MANUALE SULLE RETI NEURALI

il Mulino

DARIO FLOREANO
CLAUDIO MATTIUSSI

Psicologia

a Krisztina – D.F.
a Giovanni, per la sua intelligenza, dedizione, umanità – C.M.

Manuale sulle reti neurali

Seconda edizione

I lettori che desiderano informarsi sui libri e sull'insieme delle attività della Società editrice il Mulino possono consultare il sito Internet: **http://www.mulino.it**

ISBN 88-15-08504-1

La prima edizione di questo manuale ha incontrato il giudizio favorevole dei lettori (per lo meno di quelli che hanno voluto far sentire la loro opinione!) ed è andata rapidamente esaurita. Benché questa gratificante constatazione suggerisse di procedere senz'altro alla riedizione del volume, la rapidità degli sviluppi che caratterizza il campo delle reti neurali e la mancanza di tempo da dedicare al lavoro di aggiornamento avevano finora ostacolato questo progetto.

L'affiancarsi di un nuovo autore (C.M.) ha finalmente consentito di affrontare la revisione completa del materiale che costituiva la prima edizione e di completarlo con la descrizione dei modelli più recenti e promettenti. Il risultato è questa seconda edizione del *Manuale sulle reti neurali*, che mantiene invariati gli obiettivi e lo spirito della prima edizione. I contenuti e la bibliografia sono stati aggiornati, mentre alcune parti della precedente edizione, che descrivevano modelli troppo specifici, sono state eliminate. È stato inserito un nuovo capitolo interamente dedicato alla descrizione dei modelli neuronali dotati di dinamica interna e degli algoritmi per la loro simulazione al calcolatore, comprendente in particolare un esame dei modelli più popolari di neuroni impulsivi (*spiking neurons*).

Ringraziamo Elisabetta Dosso e Davide Marocco per le osservazioni e i preziosi suggerimenti forniti relativamente alla revisione del materiale esistente e alla stesura del nuovo capitolo.

D.F. - C.M.

Le reti neurali artificiali costituiscono un nuovo settore scientifico e tecnologico in rapida espansione (se non esplosione) che presenta due aspetti: se da un lato esse formano un campo d'indagine estremamente affascinante e creativo, dall'altro restano talvolta difficilmente accessibili per chi vi si avvicini la prima volta. D'altra parte non vi è da meravigliarsene: il paradigma «connessionista» (un aggettivo tipicamente utilizzato per indicare questo campo di ricerca) rappresenta una svolta drastica nel modo di concepire l'elaborazione dell'informazione nei sistemi viventi e artificiali. I nuovi modelli computazionali ridefiniscono o abbandonano completamente molte delle assunzioni su cui si poggiavano la scienza cognitiva e l'intelligenza artificiale classica. Inoltre alcune persone trovano questo settore di ricerca piuttosto ostico in quanto pieno di formalismi matematici e difficilmente utilizzabile come uno strumento d'indagine e di lavoro. Le reti neurali artificiali sono in effetti dei sistemi computazionali che si prestano ad essere descritti nel linguaggio matematico. Tuttavia chiunque sia in grado di fare una *somma* e una *moltiplicazione* può capire i principi di funzionamento di una rete neurale. Inoltre, se si conosce anche un linguaggio di programmazione (qualsiasi esso sia), è possibile realizzare in pochi minuti sul proprio computer una rete neurale e interagire con essa mentre questa apprende, si modifica e risponde. Se non si è in grado di programmare, niente paura! Esistono ormai moltissimi programmi disponibili commercialmente o gratuitamente con interfacce grafiche estremamente facili da utilizzare. Penso che il modo migliore per imparare a conoscere (e ad apprezzare) le reti neurali sia proprio quello di iniziare a utilizzarle nell'ambito dei propri interessi: indagare ipotesi sui principi di organizzazione dei sistemi nervosi biologici, costruire modelli percettivi e cognitivi, realizzare sistemi di predizione finanziaria o diagnosi medica, addestrare sistemi di controllo per robot, ecc.

Mi sono ispirato a tali considerazioni nell'approntare questo libro, un manuale che vuole essere una guida e uno strumento di lavoro per tutti coloro che desiderano approfondire la conoscenza sulle reti neurali e decidono di iniziare a utilizzarle. Il volume presenta un'introduzione graduale al soggetto, una semplificazione dei formalismi matematici, una descrizione delle procedure da seguire passo per passo nella realizzazione degli algoritmi neurali, numerose illustrazioni ed esempi concreti e una ricca bibliografia per chi desidera approfondire maggiormente determinati aspetti. Ho deciso inoltre di dedicare il capitolo finale agli algoritmi genetici, una famiglia di tecniche di ottimizzazione che si ispirano ai principi dell'evoluzione darwiniana e che spesso vengono trattati in testi separati, in quanto essi vengono utilizzati sempre più frequentemente sia come paradigma di sviluppo di sistemi intelligenti adattivi che come tecniche di apprendimento dei parametri neurali.

Ringrazio Domenico Parisi, Riccardo Luccio e Walter Gerbino per avermi sostenuto e incoraggiato e per aver fornito quelle strutture e condizioni di lavoro che mi hanno permesso di iniziare e completare questo libro, i membri del Centre for Cognitive and Computational Neuroscience dell'Università di Stirling (Gran Bretagna) per le discussioni interessanti e costruttive sulla struttura del secondo e terzo capitolo, e Giovanna Movia, Massimo Baldini e Daniele Malaguti della casa editrice il Mulino per i consigli preziosi e per la pazienza che hanno più volte dimostrato in varie occasioni durante le varie fasi di realizzazione di questo libro.

D.F.

Simboli

x	una variabile
$\mathbf{x}$	un vettore
x_i	l'i-esima componente del vettore $\mathbf{x}$
$\mathbf{x}_i$, $\mathbf{x}^i$	l'i-esimo vettore di una collezione
x_i^μ	l'i-esima componente del μ-esimo vettore
$\mathbf{X}$	una matrice
x_{ij}	l'elemento della matrice $\mathbf{X}$ che si trova all'incrocio dell'i-esima riga e della j-esima colonna
$\Phi(x)$	una funzione della variabile x
$\dot{\Phi}(x)$	la derivata prima della funzione $\Phi(x)$
∂	derivata parziale
$\forall i$	per tutti gli elementi i
$\in$, $\notin$	appartiene a, non appartiene a
$\neq$	diverso da
$\approx$	approssimativamente uguale a
$\propto$	proporzionale a
∞	simbolo di infinito
$\lvert x \rvert$	valore assoluto di x
$\langle x \rangle$	valore medio di x
$\lVert \mathbf{x} \rVert$	norma o lunghezza del vettore $\mathbf{x}$
$\lVert \mathbf{x} - \mathbf{y} \rVert$	distanza euclidea tra il vettore $\mathbf{x}$ e il vettore $\mathbf{y}$
exp	funzione esponenziale complessa
tan	funzione tangente
tanh	funzione tangente iperbolica
$\max(x)$	il valore massimo che la variabile x può assumere
$\min(x)$	il valore minimo che la variabile x può assumere
$rnd(\pm x)$	un numero reale casuale compreso tra $-x$ e x
Δw	il valore di cambiamento della variabile w (usato per esprimere il valore di modifica dei pesi sinaptici)
η	il tasso di apprendimento
$i \mid \max_i(\cdot)$	il valore di i che massimizza l'espressione tra parentesi

Nei rari casi in cui i simboli usati si discostino da quanto indicato qui sopra, ciò viene esplicitamente dichiarato nel testo.

1. Sistemi di calcolo seriale e intelligenza naturale

Malgrado il fatto che i computer moderni siano sempre più potenti e veloci, è ancora molto difficile utilizzarli per risolvere alcuni problemi che per gli esseri umani sono relativamente banali. Il riconoscimento di oggetti in situazioni quotidiane (ad esempio individuare una penna in mezzo ai libri aperti su una scrivania), la coordinazione motoria necessaria per spostarsi da una stanza a un'altra e la valutazione contemporanea di un insieme di circostanze per poter prendere una rapida decisione sono tutti esempi di abilità che possediamo e che non comportano sforzi particolari. Tuttavia non esiste alcun programma per computer che sia in grado di ottenere prestazioni simili a quelle umane in questi compiti apparentemente semplici. Una delle possibili cause sta nel fatto che il modo in cui i programmi tradizionali (basati su regole e sistemi di conoscenze) elaborano l'informazione è radicalmente diverso dal modo in cui funzionano i sistemi nervosi biologici.

Un computer tradizionale – anche detto computer seriale o computer von Neumann (dal nome dello scienziato ungherese che lo inventò) – è composto sostanzialmente da un'unità di calcolo, o processore, che esegue in successione un altissimo numero di operazioni (nell'ordine di parecchie centinaia di milioni al secondo) e da tre tipi di memoria: una che contiene le istruzioni necessarie a svolgere le operazioni, una temporanea da cui vengono letti i dati necessari e depositati i risultati dei calcoli effettuati e una permanente in cui questi dati rimangono registrati. Per poter svolgere un qualsiasi compito i calcolatori tradizionali necessitano di un programma e la struttura dei programmi tradizionali riflette la struttura e il funzionamento dei computer su cui operano. Un programma tradizionale è composto di un insieme di istruzioni organizzate in modo gerarchico e da tabelle di consultazione ove vengono mantenute le conoscenze; in pratica il suo funzionamento consiste nel leggere sequenzialmente i dati e nell'applicare su di essi determinate operazioni in base alle regole e conoscenze predefinite.

L'architettura e i principi di funzionamento del computer seriale sono stati utilizzati dal cognitivismo come metafora della mente [Neisser 1967; trad. it. 1976] (l'analogia è particolarmente evidente anche nei manuali di psicologia cognitivista come, ad esempio, Lindsay e Norman [1972; trad. it. 1982]); questa scelta presenta due vantaggi evidenti: da un lato la «scienza dell'informazione» (la scienza che si occupa dello studio dei sistemi di elaborazione dell'informazione) fornisce un formalismo scientifico e universalmente accettato con cui è possibile descrivere il funzionamento della mente in modo univoco, dall'altro lato questa analogia rende possibile l'impiego dei modelli della mente per la creazione di nuove tecnologie con caratteristiche di «intelligenza artificiale». Il problema è che i sistemi di elaborazione seriale sono molto rapidi ed efficienti nel risolvere compiti in cui gli esseri umani trovano normalmente difficoltà (come, ad esempio, l'esecuzione di calcoli matematici, la rotazione di complicate figure geometriche, la memorizzazione esatta di enormi quantità di dati), ma si rivelano particolarmente inefficienti e lenti nell'affrontare compiti che appartengono alla nostra vita quotidiana, come ad esempio quelli citati nel paragrafo iniziale. Se osserviamo in maggior dettaglio la natura di questi due tipi di compiti, notiamo che i primi – quelli propri del mondo dei calcolatori – sono descrivibili da una serie di regole o procedure e possiedono una soluzione analitica, mentre i secondi – quelli tipici della vita quotidiana degli esseri umani – sono difficili da descrivere attraverso regole esplicite, non sempre è possibile ricavarne una soluzione analitica e, anche se fosse possibile suggerire delle strategie, esse non necessariamente corrisponderebbero ai processi che di fatto vengono impiegati.

Questa differenza di rendimento trova riflesso anche nella struttura dei due sistemi di elaborazione sottostanti. Al contrario del calcolatore seriale, il sistema nervoso centrale (il cervello) umano contiene circa 10^{11} elementi di elaborazione (neuroni), ciascuno dei quali comunica in media con altri 10^4 elementi [si veda ad esempio Braitenberg e Schüz 1998]. Malgrado alcune differenze fisiologiche, è ragionevole ipotizzare che i neuroni funzionino pressappoco in modo simile: ciascuno di essi emette una risposta in funzione del segnale globale ricevuto e della propria soglia di attivazione. Non esiste una porzione specifica del sistema nervoso che si occupi solamente di immagazzinare le conoscenze e un'altra che applichi esclusivamente delle regole; inoltre il sistema di trasmissione dei segnali – al contrario di quanto avviene nei computer tradizionali – è molto rumoroso (ovvero la decisione di emettere un segnale è meglio descritta dalla *probabilità* di emettere il segnale) e non molto affidabile (molti elementi e linee di comunicazione tendono a morire in modo abbastanza casuale sia per processi di normale invecchiamento che per motivi patologici). Il sistema nervoso può essere paragonato a un'immensa società – Cervellopoli – che possiede un numero di individui venti volte più grande del numero di esseri umani che vivono sulla terra. Ciascuno degli abitanti di Cervel-

lopoli conosce quasi tutti gli abitanti del proprio paese o quartiere e passa il proprio tempo a parlare con tutti loro. Alcuni di questi abitanti possiedono anche relazioni con individui che vivono in zone più distanti e mantengono così la propria comunità continuamente aggiornata su quello che succede. Come accade nella nostra società, la comunicazione è spesso caratterizzata da ripetizioni, rumore e interruzioni; inoltre gli abitanti di Cervellopoli nascono, muoiono e spesso allacciano nuove amicizie o rafforzano e indeboliscono vecchie relazioni[1] (per una descrizione più convenzionale e completa del sistema nervoso umano si veda Kandel, Schwartz e Jessel [1991; trad. it. 1994]). Come il lettore avrà modo di osservare più avanti nella lettura di questo libro, l'analogia potrebbe essere portata avanti in molti altri aspetti; quello che conta, per ora, è notare che il funzionamento del sistema nervoso è radicalmente diverso dal funzionamento di un sistema di elaborazione seriale dell'informazione.

Le differenze principali riguardano i seguenti aspetti:

– l'elaborazione dell'informazione nei sistemi nervosi avviene in *parallelo*, mentre nei calcolatori tradizionali ciascun dato viene elaborato individualmente e in successione. Malgrado il fatto che ogni singolo neurone sia relativamente lento[2], il parallelismo massivo spiega – in parte – la maggior velocità del cervello nell'eseguire compiti che richiedono l'elaborazione contemporanea di un elevato numero di dati, come ad esempio il riconoscimento visivo di oggetti.

– L'elaborazione nei sistemi nervosi è *distribuita* su molti elementi, ovvero vi sono molti neuroni che si occupano della stessa operazione. L'osservazione di un sistema nervoso – attraverso le tecniche di registrazione intracellulare o di visualizzazione dell'attività cerebrale (PET o MRI) – durante lo svolgimento di semplici compiti evidenzia l'attivazione contemporanea di molti neuroni, a volte organizzati in gruppetti locali, altre volte distribuiti «a macchie» in zone diverse del cervello. Inoltre un singolo neurone può prender parte in diversi tipi di operazioni eseguibili sia contemporaneamente che in tempi diversi.

– Ogni dato nella memoria dei calcolatori è identificato da un *indirizzo* (in pratica un numero) che viene utilizzato dal processore centrale per recuperare le conoscenze necessarie allo svolgimento di un certo compito. Invece gli esseri umani accedono alle proprie memorie in base al *contenuto*: noi siamo in grado di recuperare un ricordo semplicemente in base a qualche indizio parziale o a un attributo (un profumo, una voce, una situazione simile).

– I sistemi nervosi, al contrario dei calcolatori, non devono essere programmati per svolgere un compito, bensì *imparano* autonomamen-

[1] Il compito delle neuroscienze e della psicologia consiste nel comprendere com'è organizzata questa società, qual è il compito svolto da ciascun abitante, che tipo di linguaggio viene utilizzato e che cosa gli abitanti dicono fra di loro.

[2] Un neurone è in grado di emettere qualche *centinaio* di impulsi per secondo, mentre l'unità di calcolo di un computer di media potenza elabora parecchi *milioni* di numeri interi per secondo.

te in base all'esperienza o con l'aiuto di un insegnante esterno. Si ritiene che l'apprendimento consista soprattutto nella modifica della «forza» delle connessioni attraverso cui i neuroni comunicano: quanto più una connessione – sinapsi – è forte, tanto maggiore sarà l'effetto del segnale che vi passa sul neurone ricevente. Memorizzare un nuovo vocabolo, ricordare il viso di una persona o imparare come funziona una rete neurale artificiale consisterebbe quindi nel gioco di rafforzamento e indebolimento di un gran numero di sinapsi. Un calcolatore necessita invece di un programma che contiene tutte le istruzioni necessarie per portare a termine il compito correttamente, precisamente e infallibilmente.

In conclusione i computer seriali e i relativi programmi tradizionali sono degli strumenti molto potenti per svolgere dei compiti che richiedono la ripetizione di una serie di operazioni ben definite ove l'accuratezza, l'affidabilità e la velocità sono le caratteristiche importanti. Questi sistemi di elaborazione dell'informazione sono dunque molto utili, ma non certo intelligenti: l'unico elemento di intelligenza nell'intero processo è il programmatore che ha analizzato il compito e ha creato il programma. Affinché un sistema artificiale possa definirsi intelligente, esso dovrebbe per lo meno essere in grado di risolvere i problemi che gli esseri umani trovano semplici e naturali. Per questo motivo vale la pena studiare le modalità di elaborazione dell'informazione proprie dei sistemi nervosi biologici e cercare di analizzarle computazionalmente per indurne i principi di funzionamento e capirne le proprietà emergenti a livello del comportamento.

2. Che cos'è una rete neurale artificiale?

Le reti neurali artificiali sono dei sistemi di elaborazione dell'informazione, il cui funzionamento trae ispirazione dai sistemi nervosi biologici[3]. Una rete neurale artificiale possiede molte semplici unità di elaborazione variamente connesse tra di loro (fig. 1). Alcune di queste unità ricevono informazioni dall'ambiente, altre emettono risposte nell'ambiente e altre ancora – se ve ne sono – comunicano solamente con le unità all'interno della rete: esse sono definite rispettivamente unità di ingresso (*input*), unità di uscita (*output*) e unità nascoste (*hidden*). Ciascuna unità intende simulare il ruolo di un neurone – o di un gruppo di neuroni – nelle reti neurali biologiche: per questo motivo esse vengono anche definite impropriamente *neuroni*. Altri nomi comunemente utilizzati sono *nodo* e *processore*. Ciascuna unità svolge un'operazione molto semplice che consiste nel diventare attiva se la quantità totale di segnale che riceve supera la propria soglia di attivazione. Se un'unità diventa attiva, essa emette un segnale che viene trasmesso lungo i canali di comunicazione fino alle altre unità a cui essa

[3] Il lettore può trovare un'introduzione molto chiara ai concetti di base delle reti neurali artificiali in Parisi [1989].

è connessa; ciascun punto di connessione agisce come un filtro che trasforma il messaggio ricevuto in un segnale *inibitorio* o *eccitatorio* aumentandone o diminuendone nel contempo l'intensità a seconda delle proprie caratteristiche individuali. Questi punti di connessione simulano le *sinapsi* biologiche da cui prendono spesso anche il nome; inoltre, poiché il loro ruolo consiste in effetti nel «pesare» l'intensità dei segnali trasmessi[4], essi vengono definiti anche con il nome di *pesi sinaptici* o semplicemente *pesi*.

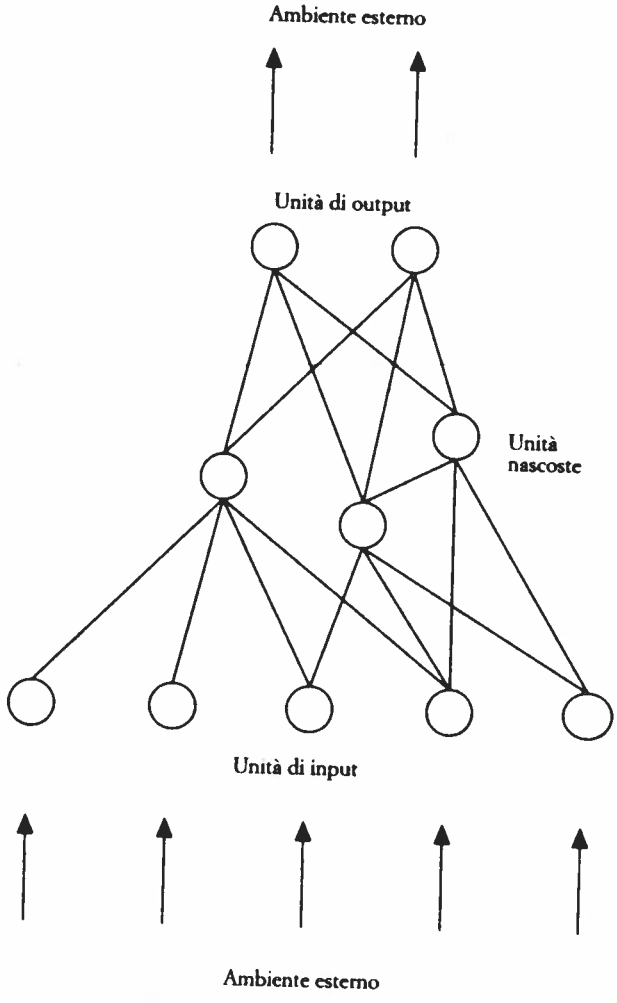


FIG. 1. Schema di una semplice rete neurale artificiale.

[4] In pratica l'operazione svolta da una sinapsi è semplicemente il prodotto tra il valore del segnale ricevuto e il valore della sinapsi stessa (che può essere un numero qualsiasi appartenente all'insieme dei numeri reali).

Formalmente, il segnale di risposta emesso da un nodo n_i è uguale a

$$n_i = \Phi\left(\sum_j w_{ij} n_j - \vartheta_i\right)$$

ovvero a una funzione Φ della somma dei prodotti dei segnali d'ingresso n_j (provenienti da altri nodi o dall'ambiente esterno) per i rispetti pesi sinaptici w_{ij}, meno il valore di soglia ϑ_i del nodo. La funzione Φ può assumere diverse forme, come vedremo in dettaglio nel capitolo successivo.

Siccome ciascun nodo può ricevere in ingresso segnali da un gran numero di altri nodi e/o recettori ambientali, ciascuno dei quali viene pesato dalla corrispondente connessione sinaptica, anche un semplice nodo è in grado di esibire comportamenti complessi. McCulloch e Pitts furono i primi a studiare questi tipi di neuroni artificiali e a dimostrare che un insieme di tali elementi connessi in rete può, in linea di principio, svolgere qualsiasi tipo di calcolo aritmetico e funzione logica [McCulloch e Pitts 1943]. Quando uno stimolo (vettore o pattern di input) viene applicato ai neuroni d'ingresso della rete neurale i segnali viaggiano in parallelo lungo le connessioni attraverso i nodi interni (sempre che ve ne siano) fino ai nodi di uscita la cui attivazione rappresenta la risposta della rete neurale. Ciascun nodo elabora solo l'informazione *locale*: questo significa che esso si attiva solo in funzione dell'informazione che riceve attraverso le proprie connessioni di ingresso, ma non sa né qual è lo scopo globale dell'elaborazione (a livello dell'intera rete) né quali operazioni vengono svolte dagli altri nodi ai quali non è collegato. Inoltre esso non è dotato di caselle di memoria ausiliarie come nel caso del processore dei computer seriali.

La configurazione delle connessioni (*architettura*) e i valori delle sinapsi artificiali determinano in gran parte il comportamento e la risposta della rete. Per questo motivo si dice che le sinapsi rappresentano la «conoscenza» o «memoria a lungo termine» della rete neurale; invece, lo stato di attivazione temporaneo dei nodi della rete in risposta a uno stimolo d'ingresso viene talvolta definito la «memoria a breve termine». È interessante notare la differenza rispetto alla nozione di «memoria» nei computer seriali (e nel cognitivismo): mentre in quest'ultimo caso la memoria è composta da diversi elementi ben distinti e localizzati, ciascuno dei quali svolge una funzione diversa dall'elemento di elaborazione, nelle reti neurali la memoria è totalmente distribuita e non richiede l'utilizzo di componenti aggiuntivi poiché è una proprietà intrinseca del sistema stesso.

Una rete neurale impara a fornire le risposte appropriate per ciascuno stimolo d'ingresso modificando i valori delle proprie connessioni sinaptiche in base a delle *regole di apprendimento*. Solitamente la fase di apprendimento è graduale e richiede molte presentazioni di uno stesso esempio, anche se vi sono alcuni modelli di rete neurale che imparano in base a una singola presentazione di uno stimolo d'in-

gresso (ad esempio il modello di Hopfield [1982]). È importante notare che le «regole» di apprendimento delle reti neurali artificiali sono ben diverse dalle «regole» utilizzate nei programmi tradizionali: mentre in quest'ultimo caso l'insieme di regole rappresenta la soluzione del problema specifico per cui è stato creato il programma (un programma che gioca a scacchi non può essere utilizzato per gestire il traffico di una stazione ferroviaria), nel primo caso la regola di apprendimento di un modello neurale indica semplicemente le condizioni locali e le modalità in cui le sinapsi si modificano a prescindere dal tipo di compito per cui la rete verrà utilizzata.

3. Impiego di reti neurali artificiali

Le reti neurali artificiali presentano alcune caratteristiche che si rivelano interessanti in molti campi di ricerca e domini di applicazione. Benché molte di queste caratteristiche varino da modello a modello, ve ne sono alcune sufficientemente generali:

Robustezza: una rete neurale è resistente al rumore, ovvero è in grado di continuare a dare una risposta corretta anche se alcune delle sue connessioni vengono eliminate («lesionate») o se viene aggiunto del rumore al segnale d'ingresso, ai canali di trasmissione o alla funzione di attivazione dei nodi. Questa proprietà è comune anche ai sistemi nervosi biologici ove la capacità di apprendere e ricordare non viene alterata in modo sostanziale dalla perdita continua di neuroni [Johnson e Brown 1988]. Man mano che il livello di rumore aumenta, le prestazioni delle reti neurali artificiali subiscono un «decadimento graduale»: la percentuale di errore potrebbe aumentare in modo lento e pressappoco uniforme su tutto il campo di risposta oppure la rete potrebbe perdere la capacità di rispondere correttamente ad alcuni stimoli e mantenere una risposta inalterata per tutti gli altri. Inoltre, come nel caso dei sistemi nervosi biologici, le reti neurali artificiali lesionate possono essere talvolta riaddestrate ad acquistare le abilità perse. Queste proprietà rappresentano un vantaggio rispetto alle modalità di funzionamento dei sistemi seriali, ove di solito la perdita di un singolo anello della catena di elaborazione comporta una caduta catastrofica delle prestazioni dell'intero sistema.

Flessibilità: un modello neurale può essere impiegato per un grande numero di finalità diverse: esso non ha bisogno di conoscere le proprietà del dominio specifico di applicazione perché le apprende in base all'esperienza. Questo non significa che un qualsiasi modello neurale possa essere utilizzato per tutti i tipi di compiti, ma implica che l'utente non deve necessariamente conoscere le soluzioni dettagliate e analitiche che caratterizzano il problema sotto indagine. In generale l'utente di una rete neurale deve essere in grado di individuare precisamente le finalità del progetto, il tipo di compito e una serie di vincoli al fine di valutare qual è il modello neurale che risulta più ap-

propriato. Ulteriori conoscenze sulla natura del problema possono essere utilizzate per decidere altri aspetti importanti del modello neurale come ad esempio l'architettura, il tipo di codifica degli stimoli esterni e della risposta della rete, alcuni parametri di addestramento, ecc. Da un lato questa caratteristica presenta notevoli vantaggi perché permette di affrontare molti problemi di cui non sono note le soluzioni analitiche; dall'altro lato vi è però il pericolo di rinunciare a cercare di comprendere a fondo la natura di un problema e di rifugiarsi in una «soluzione neurale» che non aumenta la nostra conoscenza [Dewdney 1993].

Generalizzazione: una rete neurale che è stata addestrata su un numero limitato di esempi è in grado di produrre una risposta adeguata a dei pattern d'ingresso che non ha mai visto in precedenza, ma che presentano tuttavia qualche somiglianza con gli esempi presentati durante la fase di addestramento. Questa proprietà deriva in parte dal fatto che molti modelli neurali rappresentano internamente un numero di associazioni stimolo-risposta più grande del numero di sinapsi disponibili; nel far questo la rete neurale tende a estrarre le caratteristiche invarianti dei pattern d'ingresso piuttosto che memorizzare ciascun singolo pattern. La misura in cui una rete neurale è in grado di generalizzare a un nuovo pattern d'ingresso dipende dal grado in cui queste caratteristiche invarianti possono essere ritrovate nel pattern. L'estrazione di caratteristiche invarianti risulta essere una strategia di calcolo molto potente ed economica che viene largamente utilizzata dai sistemi nervosi biologici. Ad esempio, malgrado il fatto che le caratteristiche spettrali della luce riflessa da una superficie siano molto diverse al variare delle condizioni di illuminazione, la costanza cromatica degli oggetti percepiti viene assicurata perché il sistema visivo «valuta» i colori sfruttando il rapporto fra le luci riflesse da superfici adiacenti: questo rapporto è invariante rispetto alle condizioni di illuminazione (per questo e altri esempi di invarianti percettivi, si veda Gerbino [1984]). La capacità di generalizzare a nuovi stimoli è una caratteristica molto apprezzata nei tipici campi di applicazione delle reti neurali ove spesso è impossibile ottenere una collezione esaustiva di tutti i dati su cui la rete neurale dovrà operare.

Recupero in base al contenuto: le reti neurali artificiali sono in grado di recuperare le proprie memorie in base al contenuto partendo da dati incompleti, simili o corrotti da rumore. Come nei sistemi nervosi biologici, è sufficiente un indizio per dirigere l'attivazione del sistema nella direzione appropriata completando e recuperando l'intera memoria. In alcuni modelli (ad esempio nel modello ART descritto nel quarto capitolo) il processo di recupero presenta modalità simili a quelle esibite dai soggetti umani: gli oggetti familiari vengono riconosciuti più rapidamente di quelli sconosciuti. I calcolatori seriali invece recuperano i dati impiegando un numero che rappresenta l'indirizzo di memoria corrispondente; se questo numero viene alterato o perso, non è più possibile recuperare l'intero dato.

Le reti neurali artificiali vengono impiegate in molti settori di ricerca e campi di applicazione. Una breve descrizione degli impieghi principali può fornire al lettore un'idea dell'enorme portata di questo paradigma di calcolo.

3.1. Informatica

Nel primo paragrafo di questa Introduzione ho indicato come il sistema di elaborazione dell'informazione adottato dai computer seriali sia radicalmente diverso da quello impiegato dagli esseri umani e dalle reti neurali artificiali. Le reti neurali artificiali offrono nuove prospettive di calcolo e una nuova impostazione di molti principi tradizionali alla base della scienza dell'informazione [si veda ad esempio Hecht-Nielsen 1990]. Esse sono state impiegate per:

Compressione di dati: vi è un crescente bisogno di trovare nuovi metodi per rappresentare vasti insiemi di dati in modo compatto e resistente al rumore. Le reti neurali sono state utilizzate per sviluppare codici compatti e robusti (le configurazioni sinaptiche) che possono essere utilizzati sia per la compressione che per la decompressione di immagini [Kohonen 1989]. In questo modo è sufficiente immagazzinare o trasmettere su una linea di comunicazione solamente l'attivazione dei nodi di uscita della rete neurale i quali sono solitamente in numero molto inferiore rispetto al numero dei pixel dell'immagine originaria. Un modello neurale è stato impiegato, ad esempio, per rappresentare e trasmettere in modo compatto e robusto immagini televisive ad alta definizione [Naillon e Theeten 1989].

Eliminazione del rumore: quando un segnale viene trasmesso attraverso un canale di comunicazione, il segnale ricevuto contiene del rumore che rende incerta la natura del segnale originale. Per ovviare a questo problema Widrow e Stearns [1985] hanno utilizzato una semplice rete «ADALINE» (Adaptive Linear Neuron) che impara a riconoscere i segnali eliminando il rumore aggiunto dal canale di trasmissione.

Riconoscimento di segnali sonar: le reti neurali hanno trovato molte applicazioni nel riconoscimento e classificazione di oggetti sottomarini a partire da segnali sonar. Infatti i segnali sonar marini sono spesso mescolati a una grande quantità di rumore oceanico (a volte l'intensità del rumore supera abbondantemente quella del segnale): le cause del rumore sono dovute alle proprietà di propagazione delle onde nell'acqua, ai diversi gradi di salinità, all'interferenza di oggetti sospesi, alla profondità e a molti altri fattori. Le reti neurali si sono rivelate ottimi strumenti per scoprire in tempo reale la struttura del segnale sepolto in mezzo a tutto questo rumore: esse sono state utilizzate per la rivelazione della presenza di sommergibili [Khotanzad, Lu e Srinath 1989], la distinzione fra mine e rocce di forma e dimensioni simili [Gorman e Sejnowski 1988a; 1988b] e per l'identificazione di oggetti di varie

forme e materiali con meccanismi simili a quelli impiegati dai delfini [Roitblat *et al.* 1991].

Macchina da scrivere fonetica: Kohonen ha utilizzato una rete neurale che impara a segmentare il discorso di un parlante e ad associare in tempo reale ciascun segmento alle lettere dell'alfabeto corrispondenti. Questa rete, che è in grado di adattarsi anche alle caratteristiche individuali di pronuncia del parlante, è stata utilizzata per trasformare direttamente il parlato in testo scritto [Kohonen 1988].

Riconoscimento di caratteri: molti tipi di reti neurali sono stati utilizzati per il riconoscimento di caratteri scritti a mano. Infatti le grandi differenze di dimensione, posizione e stile rendono questo compito molto arduo per le tecniche tradizionali. Le Cun ha utilizzato un modello neurale molto diffuso (Back propagation) per riconoscere con un bassissimo indice di sostituzione[5] i codici postali scritti a mano [Le Cun, Denker e Solla 1990].

3.2. Sistemi di controllo

I sistemi di controllo sono sistemi che gestiscono una serie di attuatori – come ad esempio ruote, arti meccanici, meccanismi sensoriali – con l'obiettivo di eseguire un determinato compito. Siccome il cervello dei mammiferi è la prova evidente che un sistema distribuito e parallelo è in grado di coordinare e controllare in modo molto efficiente un gran numero di attuatori diversi con moltissimi gradi di libertà, le reti neurali artificiali hanno trovato un ampio spettro di applicazioni nello sviluppo di sistemi di controllo per robot o veicoli intelligenti. Pomerleau ha impiegato una rete neurale per la guida automatica di un'automobile sia su strada che su percorsi sterrati [Pomerleau 1993]. La rete neurale osserva la strada attraverso una telecamera e impara ad associare le immagini con le azioni corrette (su sterzo, acceleratore e freni) imitando un guidatore umano. In modo analogo, una rete neurale composta da due moduli è stata utilizzata per effettuare le manovre di retromarcia di un autocarro articolato in modo tale che le porte posteriori del rimorchio vadano a coincidere con i rulli di scarico dei magazzini [Miller, Todd e Hedge 1989; Nguyen e Widrow 1989]. In una prospettiva diversa Fuminori e Fukuda [1994] hanno sviluppato un sistema di controllo neurale per un robot con due arti che impara a spostarsi appeso a delle barre; la rete neurale apprende a muoversi con la stessa tecnica utilizzata dalle scimmie correggendo i propri errori attraverso delle regole di rinforzo sinaptico.

[5] I meccanismi di riconoscimento dei caratteri esistevano già decenni prima che le reti neurali fossero applicate in questo campo. L'efficienza di questi meccanismi viene valutata in funzione dell'indice di *accettazione* (proporzione di caratteri accettati come leggibili dal sistema) e dell'indice di *sostituzione* (proporzione di caratteri classificati scorrettamente) [Hecht-Nielsen 1990].

3.3. Analisi finanziarie

L'apprendimento di caratteristiche invarianti degli stimoli di ingresso, la resistenza alle variazioni casuali e la capacità di generalizzare rendono le reti neurali degli strumenti potenzialmente molto efficaci per le predizioni finanziarie, la valutazione dei rischi in investimenti e prestiti e le analisi del mercato borsistico. Benché sia ben noto che un gran numero di compagnie specializzate in analisi finanziarie fa uso di reti neurali, vi sono relativamente poche pubblicazioni scientifiche a questo riguardo[6]. È stato comunque mostrato che alcuni modelli neurali ottengono prestazioni migliori delle tecniche tradizionali di regressione specialmente quando il modello viene applicato a nuovi dati [Dutta e Shekkar 1988]. Altri hanno invece evidenziato come le reti neurali ottengano lo stesso grado di prestazioni di sistemi a regole già esistenti [Sharda e Patil 1990]. Comunque sia, la facilità di utilizzo delle reti neurali le rende molto piu accessibili di sistemi specialistici molto costosi e complessi.

La compagnia americana Nestor Inc. ha sviluppato un modello neurale di valutazione di rischi nella concessione di prestiti; sebbene la decisione di concessione di un prestito sia nella maggior parte dei casi stabilita attraverso un processo di valutazione molto rigido e prudente, vi sono un certo numero di casi marginali in cui il rischio di errore può essere particolarmente costoso. La rete neurale impara a valutare la probabilità di rischio utilizzando l'esperienza passata fornita da molti esperti del settore [Collins, Ghosh e Scofield 1988].

Ovviamente le reti neurali non sono delle scatolette magiche, ma possono essere degli strumenti molto validi nello scoprire informazione nascosta se questa esiste. È quindi molto importante individuare il tipo di informazione su cui la rete neurale deve operare. White ha applicato senza successo una rete neurale per prevedere il valore delle azioni della IBM [White 1988]; questo fallimento non significa che le reti neurali non possano essere impiegate in questo tipo di analisi, ma che l'informazione utilizzata non era sufficiente per una corretta previsione dell'andamento borsistico di quello specifico titolo.

3.4. Medicina

La diagnosi medica è un processo molto complesso e costoso, in cui i dati di partenza sono caratterizzati da soggettività, imprecisione, un'alta dose di rumore e incompletezza [Jones 1990]; per questi motivi i sistemi esperti a regole non hanno avuto l'impatto sperato nella pratica medica. Per gli stessi motivi le reti neurali sembrano essere

[6] D'altra parte non c'è da meravigliarsene: se qualcuno sviluppa un modello neurale che si rivela particolarmente redditizio, la sua prima preoccupazione non sarà certo quella di renderlo pubblico!

particolarmente adatte come aiuto per la diagnosi e la prognosi di patologie. Una rete neurale può operare su delle immagini (come ad esempio lastre a raggi X) per suggerire la presenza di eventuali elementi patogeni (ad esempio tumori), oppure su dei dati simbolici che rappresentano una descrizione dei sintomi del paziente e di altri parametri standard (pressione, esami del sangue, ecc.) per diagnosticare la presenza di situazioni anomale. Apolloni *et al.* [1990] hanno utilizzato una rete neurale per diagnosticare la presenza di epilessia: il modello neurale è stato addestrato su 134 casi e ha ottenuto circa l'85% di generalizzazioni corrette in 22 nuovi casi di test. Anderson [1986] ha creato invece il «Medico istantaneo»[7], una rete neurale che è in grado di fornire la diagnosi e il trattamento adeguato per un paziente. La fase di addestramento consiste nella presentazione dei dati di un gran numero di pazienti (anche più di mille) che sono stati già guariti indicandone i sintomi, la diagnosi e il trattamento medico. La rete neurale immagazzina questi dati con l'algoritmo «Brain-State-in-a-Box»: quando si presenta un nuovo paziente i suoi sintomi vengono inseriti nella rete neurale che, dopo alcune iterazioni, completa il vettore di dati fornendo la diagnosi e il trattamento.

3.5. Neuroscienza

Il valore della neuroscienza risiede nella grande quantità di informazioni che è riuscita a fornire sull'anatomia e fisiologia del sistema nervoso; d'altro canto, però, la sua debolezza sta proprio nel fatto che non è riuscita a inserire il ruolo di queste strutture in un quadro più globale [Orchard e Phillips 1991]. La *neuroscienza computazionale* [Churchland 1989; trad. it. 1992; Churchland e Sejnowski 1992; trad. it. 1995; Sejnowski, Koch e Churchland 1988; Parks, Levine e Long 1998] è la disciplina che utilizza la metafora computazionale nella veste di reti neurali artificiali e semplificate per poter comprendere ulteriormente il funzionamento del sistema nervoso. Essa prevede un processo di scambio reciproco in cui le conoscenze neurofisiologiche offrano ispirazione e spunto per la costruzione di modelli artificiali, le cui proprietà computazionali emergenti possano essere accuratamente studiate e illuminino a loro volta i principi di funzionamento del sistema nervoso. Malgrado il fatto che spesso i modelli neurali artificiali siano abbastanza lontani dalla realtà biologica (come il lettore avrà modo di osservare dalla lettura dei capitoli successivi), lo studio di modelli computazionali è in grado di fornire delle spiegazioni plausibili sullo sviluppo e funzionamento dell'organizzazione nervosa (si veda ad esempio il quarto capitolo sui modelli neurali che si auto-organizzano) e può servire come ambiente di lavoro ove confrontare e valutare

[7] Il nome è stato attribuito da Hecht-Nielsen [1990, 354] ed è ormai universalmente usato per indicare questa applicazione.

diverse teorie neurali. Inoltre l'approccio computazionale offre al neurofisiologo una serie di strumenti formali con cui inquadrare coerentemente, descrivere ed eventualmente interpretare i propri risultati.

3.6. Psicologia

Le reti neurali artificiali, il cui studio in psicologia è anche noto con il nome di *Connessionismo*, hanno avuto un profondo impatto sullo studio della mente. La nuova impostazione è stata presentata per la prima volta in modo eccezionalmente chiaro e comprensivo da Rumelhart, McClelland e Gruppo PDP[8] [McClelland, Rumelhart e Gruppo PDP 1986; Rumelhart, McClelland e Gruppo PDP 1986; trad. it. 1991] (si veda anche Parisi [1989]). I modelli neurali artificiali sono visti come un'alternativa ai modelli seriali nella comprensione della «microstruttura» dei processi cognitivi. Se da un lato i modelli cognitivisti tradizionali (basati su schemi e insiemi di regole) sembrano essere adeguati per spiegare processi di alto livello che possiedono una natura seriale (ad esempio il linguaggio, il ragionamento logico, la soluzione di problemi, ecc.), dall'altro lato essi non offrono una spiegazione plausibile né dal punto di vista biologico né da quello computazionale quando si tratta di descrivere la dinamica e la struttura degli stati attraverso cui il sistema transita in modo sequenziale. L'approccio connessionista fornisce una spiegazione e un metodo d'indagine *complementare* all'approccio cognitivista tradizionale: sembra ragionevole prestare attenzione alla microstruttura dei processi cognitivi se non altro per valutarne l'integrazione con i modelli seriali delle macrostrutture cognitive (una visione ancor più radicale è presentata in Clark [1989; trad. it. 1994]). Vi è comunque una crescente fiducia che le proprietà auto-organizzative dei modelli neurali artificiali sviluppino delle abilità emergenti che possono spiegare fenomeni macrostrutturali all'interno della stessa cornice computazionale.

La complementarità dell'approccio connessionista risulta anche evidente dal fatto che i modelli neurali si sono rivelati particolarmente potenti in campi d'indagine in cui le spiegazioni cognitiviste erano superficiali o inadeguate, come ad esempio la struttura delle rappresentazioni interne, la formazione di concetti, la struttura della memoria, l'apprendimento di regole linguistiche, le prime fasi dell'elaborazione sensoriale e la coordinazione sensomotoria solo per citare alcuni esempi (alcuni tra i modelli più famosi sono raccolti in McClelland, Rumelhart e Gruppo PDP [1986]; Anderson e Rosenfeld [1988]; Anderson, Pellionisz e Rosenfeld [1990]; si veda anche Quinlan [1995] e Churchland e Sejnowski [1992; trad. it. 1995]).

[8] I contenuti dei due volumi sono basati sui contributi di circa una dozzina di ricercatori interessati allo studio del «Parallel Distributed Processing», i quali cominciarono a riunirsi e a discutere le loro idee agli inizi degli anni Ottanta presso l'Università di California a San Diego.

4. Reti neurali e computer

Malgrado il fatto che le reti neurali artificiali funzionino in modo radicalmente diverso dal computer seriale, è comunque possibile simulare una rete neurale artificiale su qualsiasi tipo di computer. Se il computer possiede un tipo di architettura seriale, come è il caso di tutti i personal computer e quasi tutti i computer da scrivania, la fase di attivazione della rete neurale viene discretizzata e gli stati dei nodi vengono calcolati in modo seriale; se la rete possiede più di uno strato di unità, prima di passare al calcolo dell'attivazione di uno strato superiore bisogna aver calcolato l'attivazione di tutti i nodi dello strato inferiore. Questa procedura richiede solitamente dei tempi abbastanza lunghi (anche se tutto è relativo perché dipende dalla velocità del processore e dalla complessità del modello neurale), ma permette a chiunque di fare degli esperimenti con reti neurali sul proprio calcolatore a casa o in ufficio.

È importante notare che il computer non è più un modello per il programmatore (come lo era nel caso dei modelli cognitivisti o nel caso degli algoritmi di intelligenza artificiale classica), bensì un semplice e comodo strumento che ci permette di affrontare l'approccio neurocomputazionale in un ambiente integrato ove possiamo registrare variabili, condurre studi parametrici, visualizzare dati e descrivere i risultati delle nostre esperienze. Di per sé le reti neurali artificiali non richiedono un hardware dedicato: è possibile ad esempio simulare una rete neurale con un sistema di bacinelle e canali a portata variabile collegati a un rubinetto. Le bacinelle costituiscono i neuroni, il segnale d'ingresso è l'acqua portata dai canali, i sistemi di apertura variabile dei canali rappresentano le sinapsi e la soglia in base alla quale i nodi decidono di rispondere è semplicemente la capienza della bacinella. Il sistema può essere reso ancora più complesso usando dei sistemi di ribaltamento laterale a equilibrio instabile per simulare i segnali inibitori oppure dei buchi di scarico per creare le caratteristiche di resistenza al segnale di certi tipi di neuroni.

Giochi d'acqua a parte, le reti neurali artificiali possono essere facilmente simulate anche sui *calcolatori paralleli*. Vi sono vari tipi di calcolatori paralleli, ma l'idea di base comune è che vi sono molti processori – ciascuno con una propria memoria limitata – collegati fra di loro. Un singolo processore può quindi ricevere diversi segnali d'ingresso e trasmettere il risultato della propria operazione a molti altri processori. L'elaborazione dell'informazione avviene in parallelo. Anche se questi tipi di computer non possiedono caratteristiche neurocomputazionali (perché i canali di comunicazione non filtrano il segnale, non sono in grado di modificare le proprie caratteristiche e il funzionamento dei singoli processori è sostanzialmente analogo al funzionamento dei processori dei computer da scrivania), tuttavia la loro architettura distribuita si presta particolarmente per le simulazioni di reti neurali artificiali facilitando spesso la programmazione di modelli

complessi e velocizzando comunque la procedura di attivazione della rete.

Al contrario dei modelli cognitivisti che si sono ispirati al funzionamento del computer von Neumann, i modelli neurali artificiali hanno invece ispirato la costruzione dei neurocomputer. Un neurocomputer – o chip neurale – è un dispositivo elettronico (od ottico) che elabora l'informazione come una rete neurale ed è – in alcuni casi – in grado di apprendere. Vi sono tre tipi di neurocomputer: coprocessori neurali, neurocomputer propriamente detti e chip neurali con tecnologie innovative (si veda il capitolo 8 in Hecht Nielsen [1990] e il capitolo 18 in Rojas [1996]). I primi due utilizzano solitamente la tecnologia impiegata per i computer tradizionali, ma i più diffusi sono i coprocessori neurali che possono essere collegati a un qualsiasi computer da tavolo come ogni altra scheda grafica o coprocessore matematico: essi richiedono un investimento finanziario minore e permettono di continuare a utilizzare il proprio calcolatore per altre funzioni (analisi, videoscrittura, ecc.). I chip neurali sono invece dei dispositivi molto piccoli realizzati spesso con varie tecnologie e possono essere usati come strumenti di controllo per determinati strumenti[9].

5. Breve storia

Il primo semplice modello di neurone artificiale fu creato da McCulloch e Pitts [1943], i quali proposero anche l'assemblaggio di molti neuroni in reti neurali artificiali per la realizzazione di complicate funzioni logiche. Queste prime reti neurali artificiali non erano in grado di apprendere e i valori sinaptici delle loro connessioni dovevano essere prestabiliti dallo sperimentatore. Ciascun neurone – caratterizzato da una soglia e da un'attivazione binaria – realizzava una semplice funzione logica, ma la combinazione in parallelo e in strati successivi di molti neuroni che svolgevano operazioni logiche diverse permetteva alla rete neurale di affrontare problemi complessi. I neuroni di McCulloch e Pitts, per quanto semplici, rappresentano il modello su cui si basano quasi tutti i modelli neurali odierni.

Qualche anno più tardi lo psicologo canadese Donald Hebb propose il primo meccanismo di apprendimento sinaptico per reti neurali [Hebb 1949; trad. it 1975]. La «regola di Hebb» prevede che se due neuroni collegati fra di loro sono attivi contemporaneamente il valore sinaptico della loro connessione viene aumentato. In realtà Hebb aveva formulato questa ipotesi in un quadro più teorico basato soprattutto su osservazioni del comportamento animale e umano, ma ben presto le sue idee trovarono impiego nelle prime simulazioni di reti

[9] Grazie alle loro proprietà di adattamento e generalizzazione, i chip neurali cominciano a trovare uso in prodotti commerciali come telecamere, lavatrici, forni a micro-onde e dispositivi di controllo per automobili.

neurali artificiali in grado di apprendere. Malgrado l'aspetto piuttosto qualitativo della formulazione originale di Hebb, queste idee rappresentarono a quell'epoca un importante avanzamento concettuale perché localizzarono le basi del condizionamento classico nei principi di funzionamento dei singoli neuroni.

Il periodo a cavallo fra gli anni Cinquanta e gli anni Sessanta fu dominato dai successi di Frank Rosenblatt [Nagy 1991], il quale propose una classe generale di reti neurali da lui definite *percettroni* e inventò un algoritmo di apprendimento basato su un procedimento iterativo di correzione dell'errore che era molto più potente della regola di Hebb [Rosenblatt 1962]. Ciascun percettrone consisteva in un singolo neurone che riceveva input da una serie di recettori di luminosità (la retina) su cui veniva proiettato un determinato pattern di input che il percettrone avrebbe dovuto imparare a riconoscere. La modifica delle connessioni sinaptiche avveniva in modo automatico quando il percettrone forniva una risposta scorretta al pattern di input. Assieme ad altri ricercatori, Rosenblatt costruì anche il primo neurocomputer – il Mark I Perceptron – in grado di apprendere a riconoscere dei caratteri proiettati su una griglia dotata di 20×20 sensori. Rosenblatt dimostrò anche che il suo algoritmo di apprendimento era in grado di trovare una configurazione di valori sinaptici per un percettrone se questa configurazione fosse esistita.

Nello stesso periodo Widrow e il suo studente Hoff [1960] svilupparono un algoritmo di apprendimento basato su un meccanismo di correzione dell'errore simile a quello di Rosenblatt, ma più potente nella capacità di generalizzare a nuovi pattern. L'idea di base era quella di modificare i valori delle connessioni sinaptiche in misura proporzionale alla discrepanza tra la risposta data dal neurone e la risposta corretta. Questo algoritmo di apprendimento – comunemente denominato «regola delta» – viene talvolta definito anche con il nome «ADALINE» (Adaptive Linear Neuron) e fu inizialmente sviluppato per applicazioni di ingegneria elettronica.

L'entusiasmo iniziale generato da queste scoperte subì una pausa nella seconda metà degli anni Sessanta e fu notevolmente smorzato dall'analisi delle limitazioni dei percettroni ad opera di Minsky e Papert. Il loro volume *Perceptrons* [Minsky e Papert 1969; 1988] dimostrò matematicamente che i percettroni – ovvero reti neurali a un solo strato di unità – non erano in grado di realizzare alcune funzioni logiche che non erano linearmente separabili, come lo XOR ($f(0, 0) = f(1, 1) = 0$; $f(0, 1) = f(1, 0) = 1$). L'autorevolezza del parere di Minsky e Papert e il tono delle loro conclusioni, unitamente alla mancanza di metodi di apprendimento efficienti per reti multistrato, distolsero l'attenzione della comunità scientifica (e soprattutto i finanziamenti governativi negli Stati Uniti) dalle reti neurali artificiali.

Durante quello che viene ora definito «il periodo del silenzio» (gli anni Settanta e la prima metà degli anni Ottanta) alcuni ricercatori continuarono a sviluppare modelli neurali che riscossero molto succes-

so solo molto più tardi. In particolar modo negli Stati Uniti Stephen Grossberg produsse un'enorme mole di lavoro caratterizzato soprattutto da analisi matematiche delle prestazioni del sistema nervoso e da modelli neurali biologicamente plausibili. I suoi lavori di quegli anni coprirono un ampio spettro di argomenti, alcuni dei quali sono riassunti in Grossberg [1987]. Più recentemente, assieme a Carpenter, egli ha proposto un modello neurale auto-organizzato, da loro definito con il nome di *Teoria della Risonanza adattiva* [Carpenter e Grossberg 1987a; 1987b; 1990], che riscosse un notevole successo per il suo carattere innovativo. Nello stesso periodo Klopf stava lavorando a un modello di neurone «edonista» (ovvero un neurone intento ad aumentare l'effetto dei segnali d'ingresso eccitatori e a diminuire l'effetto dei segnali inibitori) che lo portò poi a formulare un'elegante teoria computazionale sui principi del condizionamento classico [Klopf 1988]. Il neurofisiologo Anderson invece stava costruendo il suo modello di rete neurale auto-associativa noto con il nome *Brain-State-in-a-Box* caratterizzato dal tentativo di proporre una spiegazione computazionale di alcune proprietà fisiologiche e cognitive del sistema nervoso umano [Anderson *et al.* 1977].

Nel frattempo in Europa vi era una maggior attività, in parte dovuta al fatto che essa si svolgeva all'interno di varie discipline tradizionali e in parte perché probabilmente le critiche di Minsky e Papert non avevano avuto qui la stessa influenza. Caianiello fu tra i primi a formulare una teoria del funzionamento dei neuroni in base a principi di meccanica statistica con un tipo di apprendimento hebbiano [Caianiello 1961]: il vantaggio di questo approccio consisteva nella possibilità di sfruttare una serie di principi e strumenti di analisi già consolidati per studiare le proprietà di adattamento e funzionamento delle reti neurali artificiali. Caianiello rimase attivo fino alla sua scomparsa facendosi promotore in Italia assieme ai suoi collaboratori della nascita della Società Italiana di Reti Neuronali e dell'organizzazione delle conferenze nazionali annuali sulle reti neurali artificiali. In Finlandia Kohonen stava invece lavorando su memorie associative con caratteristiche di apprendimento hebbiano [Kohonen 1974], mentre Willshaw in Gran Bretagna e von der Malsburg in Germania proponevano un modello computazionale dello sviluppo dell'organizzazione colonnare [von der Malsburg 1973] e della formazione retinotopica della corteccia striata [Willshaw e von der Malsburg 1976].

Agli inizi degli anni Ottanta vi fu una graduale rinascita dell'interesse verso le reti neurali artificiali. Mentre le prime conferenze iniziavano ad avere luogo negli Stati Uniti e l'Agenzia dei Progetti di Ricerca Avanzata per la Difesa (DARPA) riprendeva a finanziare i progetti neurocomputazionali, il premio Nobel per la fisica John Hopfield pubblicò due articoli estremamente chiari in cui paragonò il funzionamento di una rete neurale a un sistema fisico descrivibile da una funzione di energia in cui la memorizzazione di pattern corrispondeva a un abbassamento dell'energia del sistema e il recupero consisteva inve-

ce nel raggiungimento di uno stato di equilibrio energetico [Hopfield 1982; 1984].

In Giappone Fukushima [1975] propose un modello neurale (*Cognitron*) che imparava a riconoscere caratteri in base a un processo di auto-organizzazione. Questo modello è stato poi modificato e sviluppato (*Neocognitron*) fino a renderlo in grado di riconoscere oggetti a prescindere dalla loro posizione o rotazione nel campo visivo della rete neurale [Fukushima 1988; Fukushima *et al.* 1983].

Il 1986 vide la diffusione dell'algoritmo di Back-propagation e la pubblicazione dei due volumi del Gruppo PDP [McClelland, Rumelhart e Gruppo PDP 1986; Rumelhart, McClelland e Gruppo PDP 1986; trad. it. 1991]. L'algoritmo di Back-propagation rappresentava la risposta alla critica di Minsky e Papert in quanto, a partire dall'algoritmo di Rosenblatt e basandosi sulla regola delta di Widrow-Hoff, esso proponeva un potente metodo ricorsivo per modificare i valori sinaptici di una rete neurale con un qualsiasi numero di strati composti a loro volta da un qualsiasi numero di neuroni. Questo algoritmo fu largamente diffuso da Rumelhart, Hinton e Williams [1986], ma più tardi fu reso noto che era già stato scoperto e riscoperto parecchie volte da altri ricercatori [Le Cun 1985; Parker 1985; Werbos 1974], come ogni tanto accade per alcune scoperte scientifiche. Back-propagation è tuttora l'algoritmo di apprendimento più diffuso sia come strumento di ricerca e modellizzazione che nelle applicazioni ingegneristiche e informatiche.

Verso la fine degli anni Ottanta la disponibilità dell'algoritmo di Back-propagation e di personal computer sufficientemente potenti diede luogo a un'esplosione della popolarità delle reti neurali, che sull'onda del rinato interesse vennero applicate a svariati problemi, mentre la loro efficacia veniva propagandata a volte con qualche eccesso di ottimismo.

Gli anni Novanta sono stati caratterizzati da una più equilibrata valutazione dei limiti e delle possibilità delle reti neurali. Parallelamente è cresciuta l'attenzione per il ruolo svolto dalla componente temporale nell'elaborazione neurale. Ciò ha portato la comunità dei ricercatori a studiare le proprietà di alcuni modelli neurali ispirati alla teoria dei sistemi dinamici [Beer e Gallagher 1992] e tendenti a riprodurre alcune caratteristiche della dinamica dei neuroni reali [Maass e Bishop 1999]. Un'altra linea di ricerca molto attiva è quella che cerca di mettere a frutto la mole di dettagli neurofisiologici che si vanno accumulando per costruire dei modelli di computazione neurale biologicamente plausibili, nella speranza di svelare alcuni meccanismi del funzionamento cerebrale [Rolls e Treves 1998].

Un'interessante prospettiva storica sullo sviluppo delle reti neurali dalle origini fino alla metà degli anni Novanta è offerta dalla serie di interviste raccolte nel volume curato da Anderson e Rosenfeld [1998].

Capitolo 1

1. Circuiti neurali biologici

Siccome le reti neurali artificiali aspirano alla plausibilità biologica o per lo meno si ispirano al funzionamento dei sistemi nervosi biologici, è utile passare brevemente in rassegna alcuni dei fatti più rilevanti sull'organizzazione dei circuiti neurali dei mammiferi in modo che il lettore possa inquadrare meglio la posizione dei modelli esposti in seguito.

1.1. La triade nervosa

Il sistema nervoso è organizzato in regioni e moduli ben individuabili da analisi anatomiche, citologiche e istologiche. Ciascuno di questi moduli è caratterizzato dalla presenza di tre elementi costitutivi [Shepherd 1997]: i neuroni principali, i neuroni intrinseci e le fibre nervose. Le fibre nervose sono i canali di comunicazione che trasportano i segnali a entrambi i tipi di neuroni attraverso sinapsi posizionate sull'albero dendritico o sul soma dei neuroni, i neuroni principali possiedono un lungo assone che trasmette il segnale ad altri moduli o regioni, mentre i neuroni intrinseci trasmettono segnale solamente all'interno del modulo in cui si trovano[1]. Un neurone è solitamente composto da un albero dendritico, un soma e un assone ed è caratterizzato da un alto grado di *convergenza* e *divergenza*: i segnali provenienti da migliaia di altri neuroni convergono verso il soma del neurone, il cui assone a sua volta diverge in migliaia di terminali che vanno a toccare diversi altri neuroni.

[1] Questa distinzione non è molto netta perché anche i neuroni principali trasmettono segnale ai neuroni dello stesso modulo.

1.2. Le sinapsi

Le sinapsi svolgono un ruolo centrale in questa fitta rete di comunicazione poiché esse regolano l'entità e il tipo di effetto che agisce sul neurone ricevente. Una sinapsi è il punto di contatto fra il terminale assonico di un neurone (anche detto neurone *presinaptico*) e il ramo dendritico di un altro neurone (neurone *postsinaptico*). Le sinapsi possiedono tre importanti proprietà: sono puntiformi, trasmettono segnale in una sola direzione e utilizzano neurotrasmettitori chimici. Una sinapsi ha una larghezza di circa 1 micron: queste ridotte dimensioni permettono una grande ricchezza di connettività in uno spazio molto compatto. Il punto di giunzione sinaptico è caratterizzato dalla presenza di un piccolissimo spazio tra le due membrane di trasmissione. Il segnale che proviene dal neurone presinaptico causa l'emissione di alcune sostanze chimiche (neurotrasmettitori) che si diffondono nello spazio fra le due membrane: queste sostanze chimiche agiscono sulla membrana postsinaptica facendo aprire dei microcanali che permettono l'assunzione di ioni (positivi o negativi) presenti nelle vicinanze; l'assorbimento di questi ioni provoca una carica elettrica sulla membrana postsinaptica che prosegue verso il soma del neurone postsinaptico. In sostanza una sinapsi converte un segnale elettrico (il segnale presinaptico) in un segnale chimico e quindi nuovamente in un segnale elettrico. Vi sono due importanti tipi di sinapsi: le sinapsi eccitatorie e le sinapsi inibitorie. Malgrado le sinapsi inibitorie rappresentino solamente il 16% circa del numero totale di sinapsi della corteccia cerebrale dei mammiferi, esse svolgono spesso un ruolo importante nella risposta del sistema grazie alla loro collocazione strategica e all'effetto relativamente forte sull'attività postsinaptica. Inoltre, si ipotizza che esse svolgano un ruolo fondamentale di controllo del livello di attivazione dei circuiti cerebrali e, in particolare, di prevenzione dell'innesco di focolai di attività epilettica [Braitenberg 1984].

I punti di contatto sinaptico sono variamente distribuiti sul neurone postsinaptico: la maggior parte di essi si trova sulle ramificazioni dell'albero dendritico e sulle sue piccole protuberanze che ne aumentano la superficie utile (spine), ma molte altre si trovano anche su piccole ramificazioni laterali intorno al corpo del neurone o sul corpo del neurone stesso o addirittura alla base dell'assone del neurone. Solitamente nei modelli neurali artificiali si tende a a descrivere in modo semplificato l'effetto di tutte queste sinapsi come il risultato di una grande somma algebrica dei contributi eccitatori e inibitori. In realtà la ricca configurazione geometrica e la distinzione fra sinapsi eccitatorie e inibitorie può dar luogo già a livello sub-dendritico a interessanti interazioni locali (fig. 1.1) che sono in grado di svolgere operazioni logiche di base. In generale è importante notare che le sinapsi inibitorie tendono a esercitare una specie di «veto»[2] sul segnale che viaggia ver-

[2] Se una sinapsi inibitoria diventa attiva essa blocca il passaggio di tutti i segnali che provengono da zone più a monte.

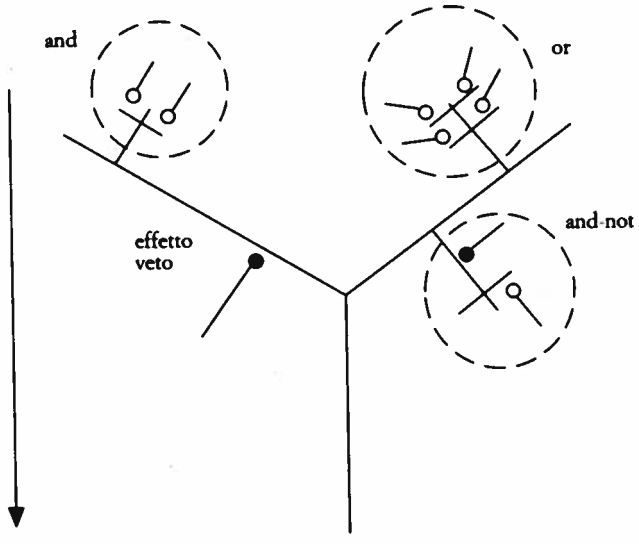


FIG. 1.1. Interazioni sinaptiche a livello sub-dendritico. I circoli vuoti sono sinapsi eccitatorie, quelli pieni sono sinapsi inibitorie. and = entrambe le sinapsi devono essere attive affinché un impulso prosegua verso il soma; or = almeno una delle sinapsi deve essere attiva per provocare un impulso; and-not = la sinapsi eccitatoria è in grado di produrre un impulso solamente se la sinapsi inibitoria è inattiva. La sinapsi inibitoria isolata può esercitare un veto sul segnale che proviene dalle sinapsi a monte quando essa diventa attiva.

so il soma del neurone postsinaptico: quindi, quanto più una sinapsi inibitoria è vicina al soma del neurone, tanto più efficace sarà il suo ruolo se essa riceve segnale presinaptico. Il ruolo distintivo dell'inibizione è testimoniato anche dal fatto che la maggior parte delle sinapsi sul soma del neurone – benché rare – sono inibitorie.

1.3. Attivazione del neurone

Due neuroni che possiedono le stesse caratteristiche morfologiche in regioni diverse del cervello possono emettere risposte completamente diverse agli stessi segnali: questa diversità è dovuta soprattutto a proprietà locali del sistema nervoso. Un neurone emette un segnale lungo il suo assone quando la differenza di potenziale fra la parte interna ed esterna della sua membrana raggiunge un certo livello di soglia. Un neurone in fase di «riposo» (quando non riceve alcun segnale esterno) rimane in una situazione di equilibrio in cui la differenza di potenziale elettrico tra l'interno e l'esterno è di circa –70 mV (i valori esatti variano per i diversi tipi di neuroni). La ricezione di segnale causa una rapida *depolarizzazione* della membrana verso valori positivi che è in grado di provocare l'emissione di una scarica (potenziale di

azione) lungo l'assone, seguita da una fase di iperpolarizzazione, ovvero di ritorno a valori negativi (ad esempio –90 mV) maggiori di quelli di riposo, e quindi a sua volta seguita da una graduale restaurazione del potenziale di riposo. Questo fondamentale ciclo, che è stato scoperto e descritto formalmente da Hodgkin e Huxley [1952], è causato dal flusso di diverse correnti di ioni (le correnti principali sono quelle di sodio e di potassio) attraverso la membrana del soma e impiega alcuni millisecondi (da 3 a 50 circa, a seconda del tipo di correnti ioniche coinvolte nel processo). Le proprietà della membrana, i tipi di correnti ioniche e la presenza di neurotrasmettitori e neuromodulatori nei pressi del soma influenzano quindi le caratteristiche di risposta del neurone. Un neurone biologico è in grado di emettere fino a 250-300 impulsi per secondo quando diventa attivo e la maggior parte di essi presentano un'attività di fondo di circa 10 impulsi per secondo in fase di riposo. Vi è un dibattito aperto sul tipo di informazione che un neurone può trasmettere. Alcuni sostengono che un neurone è in grado di trasmettere solamente un bit di informazione (che corrispone all'essere attivo o inattivo); altri sostengono [ad esempio Barlow 1972] che l'attività di un neurone trasmette più informazione grazie al numero variabile di impulsi per secondo (essi variano in funzione dell'intensità del segnale); altri ancora sostengono che l'informazione elaborata dal sistema nervoso non risiede tanto nella quantità di impulsi trasmessi da un singolo neurone, quanto invece nelle dinamiche di attività sincrona di popolazioni di neuroni [Gray *et al.* 1989].

1.4. Connettività

Malgrado la struttura del sistema nervoso centrale sia molto complessa, è possibile individuare una serie di «regole» generali sulla sua architettura e costituzione. La corteccia cerebrale è divisa in moduli che appaiono evidenti sia a livello anatomico che da un punto di vista funzionale[3]. Essa riceve i segnali sensoriali attraverso sinapsi eccitatorie dal talamo, il quale riceve a sua volta connessioni eccitatorie di retroazione dalla corteccia. Ciascun modulo corticale proietta connessioni eccitatorie verso il modulo successivo e a sua volta riceve connessioni di retroazione eccitatorie da quest'ultimo. Il ruolo delle connessioni di retroazione rimane sconosciuto, anche se alcuni autori hanno proposto l'idea che esse svolgano un ruolo importante nella sincronizzazione di popolazioni di neuroni che rispondono a caratteristiche diverse di uno stesso oggetto [Sillito *et al.* 1994; Singer 1990]. Le connessioni inibitorie hanno luogo solamente all'interno dei circuiti locali di ciascun modulo e rappresentano una minoranza.

[3] Ad esempio, la corteccia striata è composta di aree che svolgono funzioni diverse, come l'elaborazione del colore, forma e movimento [DeYoe e van Essen 1988; van Essen e Maunsell 1983].

La corteccia è anche caratterizzata da configurazioni topologiche a diversi livelli. Ad esempio i neuroni delle aree sensoriali sono disposti in modo da riflettere la disposizione geometrica dei recettori o le proprietà della stimolazione, come ad esempio le mappe retinotopiche e le mappe tonotopiche presenti nelle aree sensoriali [Knudsen, du Lac e Esterly 1987; Merzenich e Kaas 1980]. Questa topologia tende ad essere mantenuta grazie a una connettività selettiva man mano che si prosegue verso le aree superiori, anche se le distorsioni diventano sempre più evidenti. Le zone sensoriali di maggior interesse sono rappresentate da un numero maggiore di neuroni corticali: ad esempio vi è un maggior numero di neuroni che si occupano dell'elaborazione del segnale proveniente dalla fovea rispetto al numero di neuroni che rispondono ai segnali provenienti da altre parti della retina; lo stesso vale anche per la proporzione tra i neuroni che elaborano i segnali provenienti dai polpastrelli delle dita e quelli che rispondono alla stimolazione di altre parti della mano. Inoltre neuroni che rispondono a caratteristiche simili tendono a essere raggruppati e ordinati, come nel caso delle ipercolonne della corteccia striata ove ciascun neurone risponde a segmenti aventi orientazione diversa nel campo visivo. Infine, i neuroni che rispondono a caratteristiche simili dello stimolo sensoriale tendono a essere connessi fra di loro.

A livello locale i neuroni si dividono in due tipi, sostanzialmente in

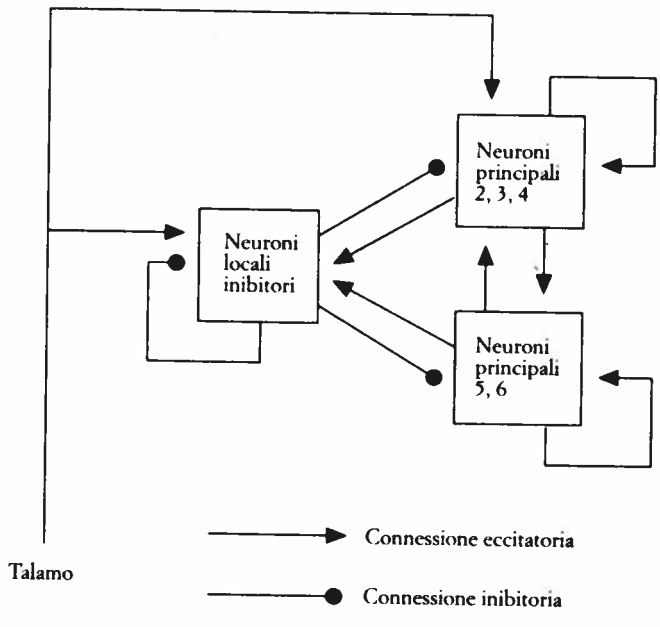


FIG. 1.2. Il circuito canonico corticale (adattato da Douglas e Martin [1990]). I numeri all'interno dei blocchi si riferiscono agli strati della corteccia: 2, 3 e 4 sono gli strati superficiali; 5 e 6 sono gli strati inferiori.

base al tipo di effetto che i loro segnali provocano sulle cellule postsinaptiche: i neuroni eccitatori e i neuroni inibitori. Un neurone biologico – contrariamente ai neuroni artificiali – non può avere terminali sinaptici sia eccitatori che inibitori. I neuroni auto-eccitatori sono molto rari: per evitare una crescita incontrollabile dell'attivazione, di solito l'eccitazione di feedback passa attraverso l'intermediazione di un altro neurone locale. L'architettura interna di un modulo è abbastanza costante in tutta la corteccia cerebrale, ma, anche se è ben nota (fig. 1.2), non si è ancora in grado di capirne la funzionalità e il funzionamento.

2. Circuiti neurali artificiali

I modelli neurali artificiali rappresentano una notevole semplificazione della loro controparte biologica al punto tale che alcuni ricercatori (soprattutto i neuroscienziati) ritengono l'aggettivo «neurale» inappropriato e preferiscono definirli sistemi di calcolo paralleli. Il grado di approssimazione biologica varia però da modello a modello e se è vero che alcuni di essi rappresentano puramente delle tecniche matematiche a scopi ingegneristici, altri tentano invece di cogliere aspetti generali del funzionamento del sistema nervoso. Dal momento che non è possibile tracciare una netta linea di demarcazione fra questi due approcci, continueremo a riferirci ai modelli descritti in questo libro con il termine di reti neurali artificiali. La descrizione degli elementi fondamentali dei circuiti neurali artificiali che segue si riferisce a proprietà generali dei modelli presentati nei capitoli successivi.

2.1. Il neurone artificiale

Un neurone artificiale è caratterizzato da un insieme di sinapsi che corrispondono ai terminali di altri neuroni, da una soglia e da una funzione di attivazione (fig. 1.3); esso verrà indicato anche con il nome di unità, nodo e processore.

Solitamente non si distingue tra fibra nervosa e sinapsi e si usa invece parlare di connessioni sinaptiche o semplicemente sinapsi. Infatti, poiché nella maggior parte dei modelli vengono ignorati i ritardi di trasmissione dovuti alla lunghezza e caratteristiche delle fibre nervose, l'effetto di un segnale x sul neurone postsinaptico è semplicemente uguale al prodotto $w \cdot x$, dove w è il peso attribuito alla sinapsi corrispondente. L'*input netto* o *potenziale di attivazione* A_i di un neurone i-esimo è la la somma algebrica dei prodotti fra tutti i segnali di ingresso x_j e i valori dei pesi delle sinapsi corrispondenti w_{ij}

$$A_i = \sum_j^N w_{ij} x_j$$

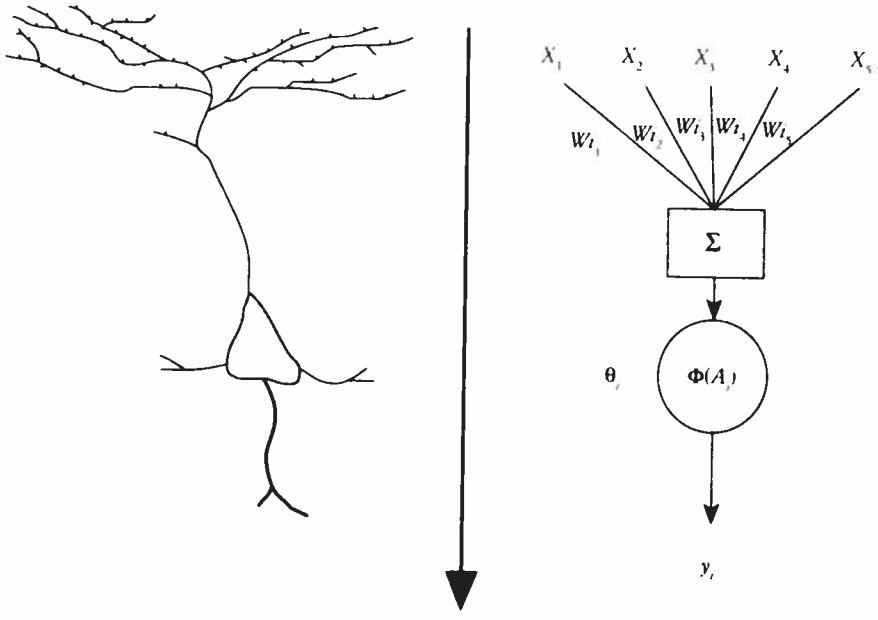


FIG. 1.3. Un neurone biologico (cellula piramidale) e un neurone artificiale.

a cui solitamente si sottrae il valore di soglia ϑ_i del neurone

$$A_i = \sum_j^N w_{ij} x_j - \vartheta_i$$

La risposta del neurone y_i viene calcolata sottoponendo il potenziale di attivazione così ottenuto all'azione di una funzione di attivazione $\Phi(A)$,

$$y_i = \Phi(A_i) = \Phi\left(\sum_j^N w_{ij} x_j - \vartheta_i\right)$$

Vi sono diverse funzioni di attivazione, alcune delle quali verranno illustrate nella sezione seguente. Nella maggior parte dei modelli il peso w_{ij} di ciascuna sinapsi può assumere valori positivi o negativi continui ed è modificabile durante la fase di apprendimento.

Poiché una rete neurale è composta di uno o più neuroni, ciascuno dei quali riceve una o più connessioni sinaptiche, è vantaggioso analizzare il sistema in notazione vettoriale. Dato che il potenziale di attivazione di un neurone è una funzione lineare dei segnali di ingresso, il potenziale di attivazione di un intero strato di neuroni $A^T = \{A_1, A_2, \ldots, A_n\}$ è riscrivibile semplicemente come

$$A = W \cdot x$$

ovvero il prodotto tra il vettore dei segnali d'ingresso $\mathbf{x}^T = \{x_1, x_2, ..., x_n\}$ (che possono essere sia i valori di input esterno che i valori di attivazione di uno strato inferiore di unità) e la matrice $\mathbf{W} = \{w_{11}, w_{12}, ..., w_{1n}, w_{21}, w_{22}, ..., w_{mn}\}$ di connessioni sinaptiche ove le righe m corrispondono ai neuroni riceventi e le colonne n ai segnali d'ingresso.

2.2. Funzioni di attivazione

La funzione di attivazione determina il tipo di risposta che un neurone è in grado di emettere. Alcune fra le più comuni sono descritte qui di seguito (fig. 1.4).

Nella formulazione originale di McCulloch e Pitts [1943], la soglia viene mantenuta fuori dal calcolo del potenziale di attivazione e la risposta del neurone è data da una funzione «a gradino»

$$\Phi(A) = \begin{cases} 1 & \text{se } A > \vartheta \\ 0 & \text{altrimenti} \end{cases}$$

dove ϑ è la soglia del neurone. Alternativamente, l'output può essere bipolare

$$\Phi(A) = \begin{cases} 1 & \text{se } A > \vartheta \\ -1 & \text{altrimenti} \end{cases}$$

In entrambi questi casi il neurone può essere in solo due stati (attivo o inattivo) e trasmette quindi solo un bit d'informazione.

Maggior informazione può essere trasmessa se viene utilizzata una funzione continua lineare

$$\Phi(A) = kA,$$

ove k è una costante. In alcune situazioni questa funzione può essere forzata a operare entro un certo intervallo (ad esempio nell'intervallo [0, 1] o [−1, 1]) per contenere l'attivazione del neurone. Le funzioni continue permettono al neurone di trasmettere una gradazione di segnali di varia intensità che può essere opportunamente sfruttata dai neuroni riceventi: questa proprietà viene considerata analoga alla frequenza di scarica (impulsi per secondo) dei neuroni biologici.

Vi è inoltre una famiglia di funzioni continue non-lineari, tra cui una delle più utilizzate è la funzione «sigmoide» o «logistica»

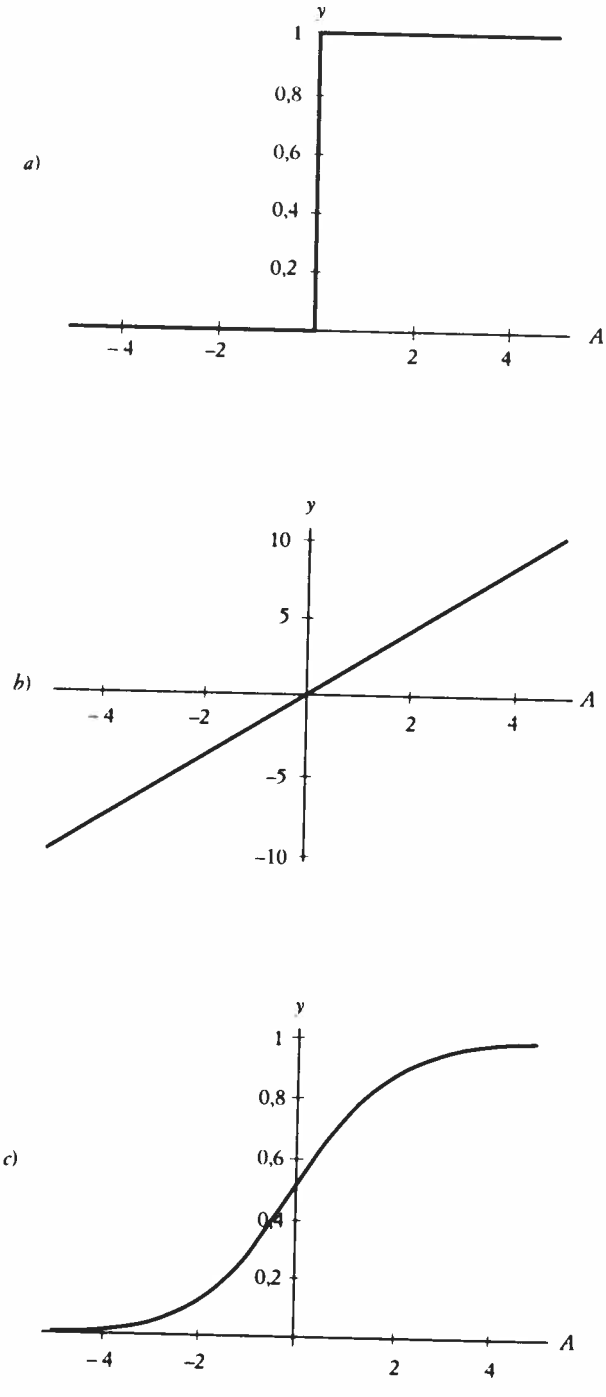


FIG. 1.4. Alcune comuni funzioni di attivazione: *a)* funzione a gradino con $\vartheta = 0$; *b)* funzione lineare continua; *c)* funzione sigmoide.

$$\Phi(A) = \frac{1}{1 + e^{-kA}}$$

ove k è una costante che controlla l'inclinazione della curva; per $k \to \infty$ la funzione sigmoide approssima la funzione a gradino. Le rette $y = 0$ e $y = 1$ sono asintoti orizzontali per la funzione sigmoide. Una funzione simile alla sigmoide è data da $\tanh(kA)$ che ha le rette $y = -1$ e $y = 1$ come asintoti orizzontali.

Nella maggior parte dei modelli tutti i neuroni della rete – eccetto i neuroni di ingresso (recettori) – utilizzano la stessa funzione di attivazione per calcolare il proprio segnale di uscita.

2.3. Architetture

L'architettura di una rete neurale è caratterizzata dalla distinzione tra neuroni di ingresso e neuroni di uscita, dal numero di strati di sinapsi (o neuroni) e dalla presenza di connessioni di retroazione.

Nelle reti *etero-associative* i nodi di ingresso che ricevono input dall'ambiente esterno sono distinti dai nodi di uscita che forniscono la risposta della rete (fig. 1.5). Il nome deriva dal fatto che questo tipo di reti apprendono ad associare coppie di vettori diversi, non necessaria-

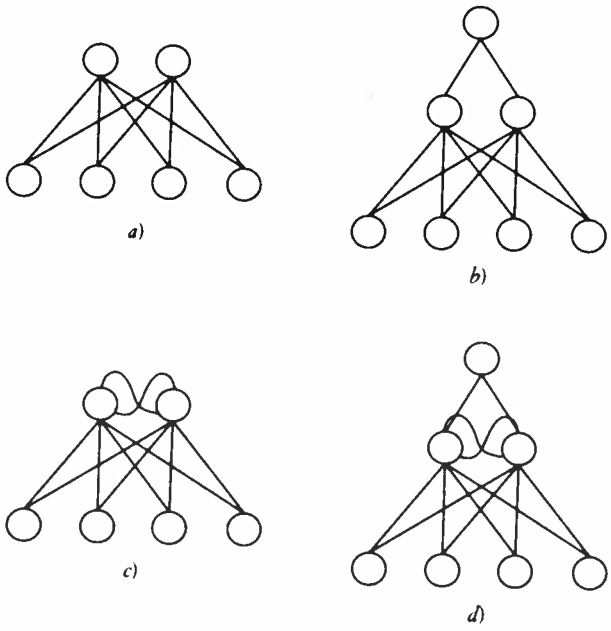


FIG. 1.5 Alcune architetture etero-associative: *a)* feedforward a uno strato di sinapsi; *b)* feedforward multistrato; *c)* uno strato con connessioni ricorrenti sui neuroni di output; *d)* multistrato con connessioni ricorrenti sui neuroni interni.

mente composti dallo stesso numero di elementi: il vettore di input e il vettore di risposta. La stessa architettura viene utilizzata anche in compiti di classificazione e di estrazione di informazione rilevante dal segnale d'ingresso. Le reti *auto-associative* possiedono invece un unico strato di unità interamente connesse fra di loro (fig. 1.6): siccome ciascuna unità riceve input sia dall'ambiente esterno che dalle altre unità, essa è in grado di emettere una risposta che varia nel tempo anche in presenza di un segnale esterno costante: la risposta della rete viene calcolata ripetendo per molte volte il calcolo dell'attivazione dei suoi nodi[4]. Le reti auto-associative vengono impiegate soprattutto per la memorizzazione di pattern.

Una prima distinzione all'interno della famiglia di reti etero-associative riguarda il numero di strati di sinapsi (alcuni autori si riferiscono invece agli strati di neuroni). I percettroni di Rosenblatt, ad esempio, erano reti con un solo strato di sinapsi (fig. 1.5a). In alcuni contesti un singolo strato di sinapsi non è sufficiente per apprendere un'associazione desiderata tra pattern di ingresso e pattern di uscita: in questi casi è necessario impiegare reti *multistrato* – anche dette «percettroni multistrato» (o MLP, *Multi-Layer Perceptron*) – che possiedono neuroni interni e più di uno strato di sinapsi (fig. 1.5b). La risposta di una rete multistrato viene ottenuta calcolando l'attivazione di uno strato di neuroni alla volta procedendo gradualmente dai nodi interni verso i nodi di uscita. Il termine «feedforward» viene usato per indicare tutte le architetture in cui ciascun nodo riceve connessioni unicamente dai nodi degli strati inferiori: in altre parole, il flusso d'informazione procede in un'unica direzione dai nodi di input verso i nodi di output. Queste reti emettono una risposta per ogni pattern di ingresso, ma non riescono a cogliere l'eventuale struttura temporale dell'informazione d'ingresso o ad esibire dinamiche temporali endogene.

Un semplice modo per dotare una rete etero-associativa di caratteristiche temporali consiste nell'aggiungere connessioni di retroazione, anche dette connessioni *ricorrenti*. Una connessione ricorrente permette la ricezione di segnale da neuroni dello stesso strato (fig. 1.5c, d) o da neuroni di strati superiori. In tal caso l'output di un nodo che possiede connessioni ricorrenti diventa

$$y_i^t = \Phi\left(\sum_j^N w_{ij} x_j^t + c\sum_l^M r_{il} q_l^{t-1} - \vartheta_i\right)$$

[4] Si noti però che è possibile associare un pattern con se stesso anche impiegando un'architettura etero-associativa che possieda lo stesso numero di neuroni di ingresso e di uscita. Inoltre è possibile dimostrare che una rete auto-associativa corrisponde a una rete etero-associativa multistrato con un numero di strati sinaptici (aventi gli stessi valori) uguali al numero di iterazioni richieste per la produzione della risposta nella rete auto-associativa [Minsky e Papert 1969; Rumelhart, McClelland e Gruppo PDP 1986, cap. 8].

ove q_i^{t-1} sono le attivazioni dei nodi dello stesso strato (incluso eventualmente il nodo i se il nodo possiede una connessione auto-ricorrente) o degli strati superiori all'istante precedente, r_{il} sono le connessioni sinaptiche ricorrenti e c è una costante.

Quando un nodo riceve segnali da tutti i nodi della rete, si dice che esso è «completamente connesso». Ovviamente è possibile utilizzare reti a connettività parziale a seconda del tipo di scopi e compiti che si vogliono affrontare. Il numero di sinapsi e il tipo di connettività sono molto importanti per il buon rendimento della rete neurale: esistono, come vedremo in seguito, delle tecniche che permettono la graduale eliminazione di connessioni superflue o la crescita di nuovi nodi e connessioni durante l'apprendimento. I pesi della sinapsi di una rete neurale sono i parametri che devono essere stimati durante l'apprendimento dei pattern di input: siccome è noto dalla statistica [si veda anche Baum e Haussler 1989] che una soluzione stabile non può essere assicurata se il numero di parametri da stimare è superiore al numero di esemplari che compongono il campione di osservazione, ciascun nodo necessiterebbe in teoria di un numero di esemplari di input almeno uguale al numero delle proprie sinapsi per poter apprendere correttamente la trasformazione da input ad output. In alcune situazioni è possibile utilizzare conoscenze a priori sui vincoli del sistema e sulle caratteristiche del compito di addestramento per scegliere il tipo di connettività e architettura più adatto.

Come abbiamo accennato sopra, la funzione primaria delle reti auto-associative consiste nell'apprendere, immagazzinare e ricostruire pattern; il loro nome deriva dal fatto che esse associano le diverse parti di uno stesso pattern l'una con l'altra (fig. 1.6).

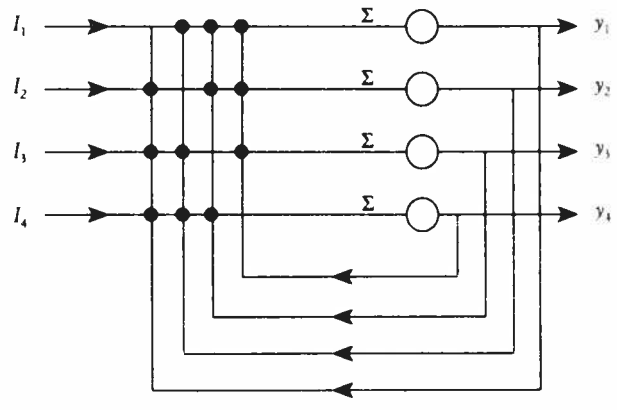


FIG. 1.6. Una rete auto-associativa.

L'apprendimento avviene presentando ciascun pattern a tutte le unità della rete e modificando i valori di tutte le connessioni sinaptiche. Una volta che il pattern è stato memorizzato, è sufficiente presen-

tarne solo una parte – o una versione indebolita dal rumore – e calcolare in modo ciclico l'output di ciascun nodo

$$y_i = \Phi\left(I_i + \sum_{j \neq i}^{N} w_{ij} y_j \right)$$

ove I_i è l'input esterno del nodo i e j è l'indice che si riferisce alle altre unità della rete; solitamente non si usano valori di soglia e ciascun neurone non possiede auto-connessioni. Dopo alcune iterazioni del calcolo dell'attivazione la rete ricostruisce la versione originale del pattern memorizzato. La ricostruzione del pattern corrisponde a uno stato di *equilibrio* in cui le attivazioni dei nodi della rete cessano di variare. Analizzeremo in seguito le dinamiche dettagliate di alcuni modelli auto-associativi.

2.4. Codifica e rappresentazioni

I valori di attivazione dei nodi della rete dipendono dal tipo di funzione che viene applicata per calcolare la loro risposta. Essi possono assumere valori binari {0, 1} o bipolari {–1, +1}, oppure valori continui, talvolta ristretti entro un certo dominio, come ad esempio [0, 1] o [–1, +1]. La scelta di una determinata funzione di attivazione comporta implicazioni matematiche che influenzano le prestazioni o l'applicabilità di un modello neurale: alcuni algoritmi di apprendimento richiedono che le funzioni di attivazione siano continue e differenziabili (come ad esempio la funzione sigmoide).

La codifica dei pattern d'ingresso di una rete neurale rappresenta una decisione più complessa e meno vincolata anche se spesso il tipo di attivazione di ciascun singolo nodo riflette le proprietà dei nodi interni della rete. In generale le codifiche bipolari si rivelano vantaggiose rispetto alle codifiche binarie in molte situazioni perché alcuni tipi di classificazione sono risolvibili da semplici modelli di apprendimento hebbiano se vengono utilizzate unità bipolari, ma non se vengono utilizzate unità binarie. Inoltre l'impiego di unità bipolari permette di trattare input incompleti in modo neutro (mettendo i componenti mancanti a zero), mentre nel caso di unità binarie lo zero possiede già un significato. La scelta del tipo di codifica dipende comunque dai requisiti matematici dell'algoritmo di apprendimento che viene utilizzato. Ma, posto ad esempio che si decida di utilizzare nodi ad attivazione continua entro il dominio [0, 1], come possiamo rappresentare l'informazione di input su cui la rete dovrà operare? Immaginiamo di impiegare una rete neurale per classificare una serie di oggetti. Nel presentare ai nodi d'ingresso della rete i vari oggetti possiamo impiegare una *codifica locale*, in cui ciascuna unità d'ingresso corrisponde a un determinato oggetto, o una *codifica distribuita* in cui molte unità contribuiscono a rappresentare ogni singolo oggetto (fig. 1.7).

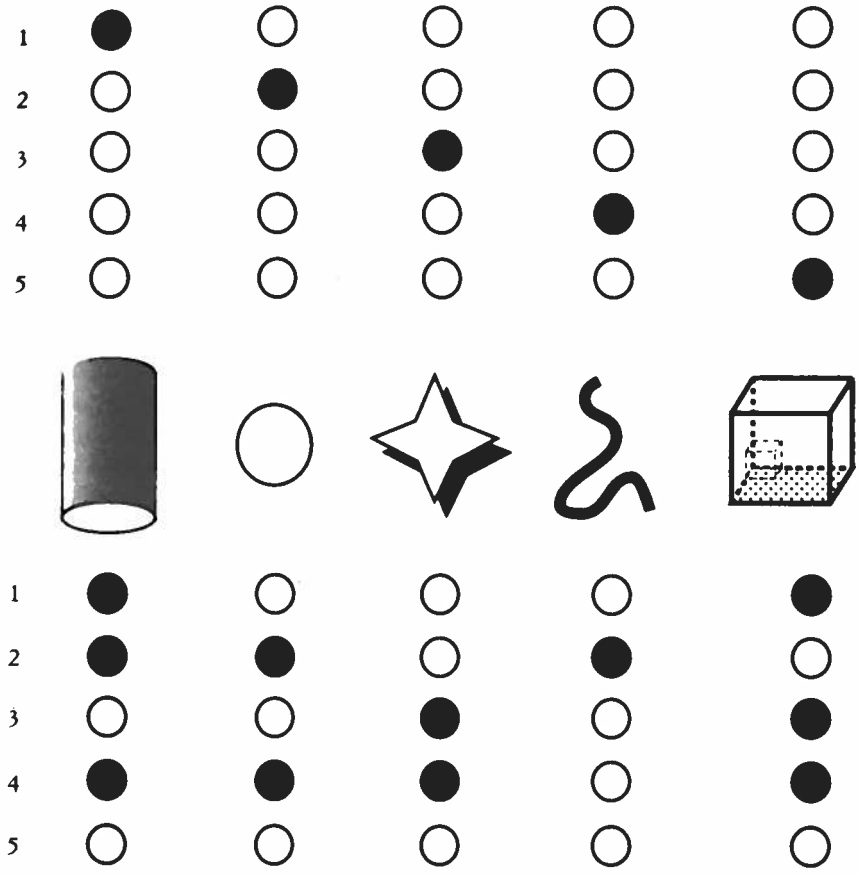


FIG. 1.7. Codifica locale e codifica distribuita per cinque nodi di ingresso di una rete neurale. Gli oggetti sono raffigurati nella porzione centrale dell'illustrazione. Ciascuna colonna di cerchietti rappresenta l'attivazione dei nodi della rete per l'oggetto corrispondente: i cerchietti neri indicano nodi attivi, mentre i cerchietti bianchi indicano nodi inattivi. La parte superiore mostra una codifica locale, mentre la parte inferiore mostra una codifica distribuita in cui ciacun nodo si attiva in presenza di una determinata caratteristica degli oggetti (profondità, presenza di spigoli, chiusura, ecc.). Si noti che la codifica distribuita non richiede l'impiego di tutti i nodi per rappresentare i cinque oggetti.

La codifica locale, benché semplice, presenta una serie di svantaggi: *a*) potenzialmente richiede un alto numero di unità, uguale al numero di oggetti che si vogliono presentare alla rete; *b*) richiede la conoscenza anticipata del numero di oggetti per poter allocare il giusto numero di unità; *c*) è molto fragile in quanto la perdita di un'unità (a causa di lesioni o rumore) corrisponde alla perdita dell'oggetto corrispondente. Per questi motivi le codifiche locali, oltre a non essere un buon candidato per la rappresentazione della conoscenza nei sistemi nervosi biologici, vengono raramente utilizzate nelle reti neurali artifi-

ciali. D'altro canto vi sono diversi tipi di codifica distribuita. Un codice binario, ad esempio, permette una rappresentazione più compatta dell'informazione: mentre con una codifica locale n nodi possono rappresentare solo n oggetti, con un codice binario n nodi possono rappresentare 2^n oggetti. Tuttavia il codice binario è soggetto alle altre due limitazioni rilevate per la codifica locale, ovvero la mancanza di flessibilità nell'accomodare nuovi oggetti e la fragilità rispetto al rumore. Una possibile soluzione consiste nel codificare le «caratteristiche» degli oggetti, non gli oggetti stessi. In quest'ottica ciascun oggetto è descrivibile da un insieme di caratteristiche, come ad esempio il peso, il colore, la rotondità, la profondità, ecc. Ciascuna unità d'ingresso della rete codifica quindi la presenza e/o il valore di una certa caratteristica: ciascun oggetto attiva una o più unità e ciascuna unità viene impiegata per uno o più oggetti: per questo motivo si parla talvolta di codifica grezza (*coarse*). Ogni singolo oggetto è definito unicamente dalla combinazione di unità attive nella rete. Questo tipo di codifica è resistente alle lesioni e al rumore: se un'unità è danneggiata un oggetto può essere ancora definito con una buona probabilità in base alle caratteristiche rimanenti; inoltre la rete è in grado di rappresentare nuovi oggetti senza bisogno di aumentare il numero di nodi. Ogni unità definisce una dimensione, ovvero un campo di possibili variazioni della caratteristica corrispondente: l'unità che codifica l'altezza di un oggetto, ad esempio, può assumere valori diversi a seconda della misura. Il grado di variazione su ciascuna dimensione corrisponde al valore di attivazione di un'unità della rete (ma è possibile anche ripartire il campo di variazione in sotto-campi e assegnare a ciascun sotto-campo una singola unità[5]). È possibile visualizzare questa rappresentazione come uno spazio multi-dimensionale con tante dimensioni quante sono le unità: ciascuna dimensione (o asse) corrisponde al campo di variazione dell'unità (caratteristica) corrispondente e ciascun oggetto corrisponde a un punto nello spazio. Siccome oggetti simili occupano posizioni vicine nello spazio della rappresentazione, questo tipo di codifica facilita la classificazione e generalizzazione della rete neurale.

Un modo diverso di ottenere una codifica grezza consiste nell'utilizzare campi recettivi sovrapposti. Immaginiamo di voler distinguere delle figure disposte in varie posizioni su una retina artificiale (fig. 1.8). Possiamo tassellare l'immagine in zone di dimensioni uguali e parzialmente sovrapposte; ciascuna di queste zone costituisce il campo recettivo di un nodo d'ingresso la cui attivazione è una funzione dei contributi di tutti i recettori o pixel della zona (ad esempio una som-

[5] Vi sono molte altre possibili variazioni su questo tipo di rappresentazione: ad esempio uno potrebbe scegliere di segmentare ciascun campo di variazione (ad esempio la luminanza di una superficie) su unità parzialmente sovrapposte, oppure di utilizzare unità che rispondono alla «congiunzione» di caratteristiche dell'oggetto. La scelta è largamente determinata dal tipo di operazione che la rete deve svolgere sull'informazione d'ingresso.

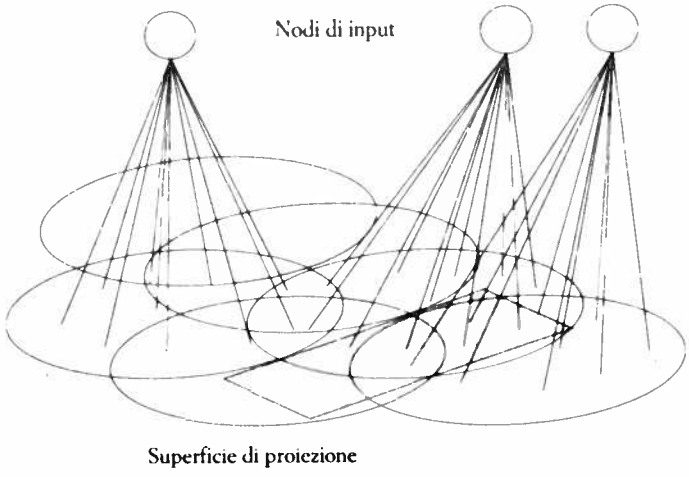


FIG. 1.8. Codifica distribuita a campi recettivi parzialmente sovrapposti.

ma). L'accuratezza a di questa rappresentazione, ovvero il grado di finezza con cui il codice neurale è in grado di distinguere diverse caratteristiche, dipende dal raggio r di ciascuna zona e dal numero di zone n che coprono regolarmente il campo retinico [Rumelhart, Hinton e Williams 1986]

$$a \propto nr$$

quando le caratteristiche giacciono su uno spazio bidimensionale. I contributi dei singoli pixel possono essere semplicemente sommati oppure pesati in base a una funzione che svolge il ruolo di «filtro». Ad esempio, Jacobs e Kosslyn [1994] hanno utilizzato una funzione gaussiana e hanno studiato l'influenza dell'ampiezza del campo recettivo (determinato dalla varianza della gaussiana) sul tipo di operazioni che una rete neurale può compiere sull'immagine. I loro risultati sperimentali indicano che quando i campi recettivi sono relativamente piccoli e non sovrapposti la rete neurale impara più facilmente a categorizzare le figure e a individuare relazioni spaziali generiche (sopra, sotto), mentre quando i campi recettivi sono grandi e parzialmente sovrapposti la rete apprende più facilmente a distinguere le caratteristiche individuali delle singole figure e a stabilire precise distanze metriche. Questo risultato non dovrebbe sorprendere poiché, a parità di nodi distribuiti regolarmente sulla superficie, larghi campi recettivi fanno sì che piccole caratteristiche possano essere codificate su un numero maggiore di nodi permettendo così una maggior discriminazione dei dettagli. Un dato interessante degli esperimenti di Jacobs e Kosslyn è che, se l'ampiezza dei campi recettivi può essere automaticamente modificata durante la fase di apprendimento, le reti esplicita-

mente addestrate a *categorizzare* figure sviluppano piccoli campi recettivi, mentre le reti addestrate a *distinguere* le singole figure sviluppano campi recettivi più larghi.

Le codifiche distribuite però non sono in grado di rappresentare separatamente oggetti distinti che vengono presentati simultaneamente alla rete: in questo caso le loro caratteristiche attivano contemporaneamente i nodi corrispondenti e l'identità degli oggetti originari viene persa. Questo è un problema condiviso da molti modelli neurali artificiali, ma non dai sistemi nervosi biologici i quali non mostrano difficoltà nel percepire e distinguere oggetti presenti contemporaneamente nel campo percettivo. Il «problema del raggruppamento» (*binding problem*) sembra risolvibile se si ipotizza che gli oggetti siano rappresentati dalla sincronia temporale di risposta di popolazioni di neuroni che si attivano in fasi diverse [Abeles 1991], ma i modelli che incorporano questa assunzione sono ancora in fase di sviluppo.

Il problema della rappresentazione degli stimoli d'ingresso è particolarmente importante quando si opera su dati ad alta dimensionalità (con un alto numero di componenti), come nel caso di immagini visive caratterizzate da una grande collezione di singoli valori (i pixel). A meno che non si desideri modellare le operazioni retiniche[6], vi sono alcune ragioni per cui è preferibile impiegare un metodo di riduzione dei dati. Una di queste consiste nel rischio di una stima scorretta dei valori sinaptici perché non è sempre possibile avere un numero di immagini di addestramento uguale o maggiore al numero di sinapsi della rete. Un altro motivo è che i valori dei singoli pixel sono altamente ridondanti e ambigui nel contempo, mentre gran parte dell'informazione utile risiede nelle relazioni di livello superiore, come ad esempio i bordi di luminanza, i rapporti di riflettanza, i centri di massa, ecc. Sembra infatti che la retina biologica e il Nucleo genicolato laterale (il primo centro extra-retinico a ricevere l'informazione ottica) svolgano un numero di operazioni volte ad estrarre l'informazione essenziale dall'attivazione dei recettori e a rappresentarla in modo compatto (si veda ad esempio [Barlow 1961; Hancock, Baddeley e Smith 1992; Linsker 1988; Sanger 1989], ma anche Field [1994] per una trattazione critica di questo argomento). Queste operazioni, che sono state formalizzate in una serie di filtri atti a descrivere in modo compatto i dati grezzi [si veda ad esempio Marr 1982; Watt 1991], possono essere impiegate per la «preparazione» dei dati prima di presentarli alla rete neurale. Alternativamente (o in combinazione) è possibile utilizzare una cascata di nodi con campi recettivi localmente ristretti in modo che l'ampiezza della «finestra» attraverso la quale ogni nodo osserva l'immagine diventa più grande man mano che si procede verso strati di elaborazione successiva.

[6] Le prime fasi dell'elaborazione retinica sono ben note e sono state riprodotte artificialmente in una retina di silicio che è soggetta alle stesse «illusioni» della retina umana [Mahowald e Mead 1991].

Si noti infine che la distinzione fra configurazioni visive che possiedono determinate proprietà geometriche potrebbe risultare difficile per una semplice rete neurale. Infatti, mentre noi siamo in grado di cogliere le relazioni geometriche tra diverse zone dell'immagine, un neurone artificiale esegue una semplice somma dei singoli contributi del proprio campo recettivo: questo significa che lo stesso valore di input netto (la somma pesata degli input) potrebbe corrispondere a una qualsiasi disposizione degli elementi presenti nel proprio campo recettivo. L'unico modo in cui questi neuroni artificiali possono imparare a distinguere pattern geometrici diversi è dato dal calcolo della correlazione fra le intensità dei singoli elementi su tutti i pattern di addestramento. Secondo alcuni autori è improprio concludere che una semplice rete neurale con connessioni feedforward che apprende a distinguere determinate configurazioni visive sia in grado di «percepire» proprietà geometriche, come ad esempio il concetto di simmetria (si vedano i risultati di Enquist e Arak [1994] e Johnstone [1994] e il dibattito successivo [Cook 1995]) e altre relazioni spaziali [si veda Kosslyn *et al.* 1992] e critiche successive [Cook, Fruh e Landis 1995]. Questa critica è sollevata però se le unità della rete sono in grado di comunicare fra di loro attraverso connessioni ricorrenti oppure se la rappresentazione di input è tale da rilevare esplicitamente le proprietà geometriche delle immagini.

Un problema che si presenta frequentemente quando si opera con dati reali è che questi non sono omogenei, sia nel caso in cui vengano prelevati continuamente attraverso dispositivi di registrazione che nel caso in cui siano estratti da liste predefinite. Ad esempio, nel caso di un robot che si muove in un ambiente vi potrebbero essere alcuni sensori che emettono risposte caratterizzate da picchi ad alta intensità oppure che operano su scale diverse: queste disomogeneità non riflettono necessariamente proprietà del mondo esterno e possono disturbare il funzionamento di una rete neurale[7]. Se si conosce l'ampiezza d'oscillazione dei singoli segnali è possibile riportarli su di una stessa scala (ad esempio [0,1]); altrimenti è possibile normalizzarli. La normalizzazione consiste nel far sì che la «norma», o lunghezza, di ciascun vettore $\mathbf{x}$

$$\|\mathbf{x}\| = \sqrt{\mathbf{x} \cdot \mathbf{x}} = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$$

sia uguale a 1 (fig. 1.9): per ottenere questa proprietà ciascun componente del vettore viene diviso dalla norma del vettore

[7] Ad esempio un nodo con attivazione sigmoide potrebbe essere saturato (ovvero rispondere sempre 1) dalla presenza di un segnale molto forte, mentre i segnali rimanenti verrebbero trascurati.

$$x'_2 = \frac{x_i}{\sqrt{\sum_{i=1}^{N} x_i^2}}$$

Siccome gran parte delle operazioni degli algoritmi neurali sono riconducibili a manipolazioni della distanza tra i vettori di addestramento e i vettori sinaptici, il processo di normalizzazione rende in molti casi queste operazioni più facili, oltre ad essere specificamente richiesto da alcuni modelli.

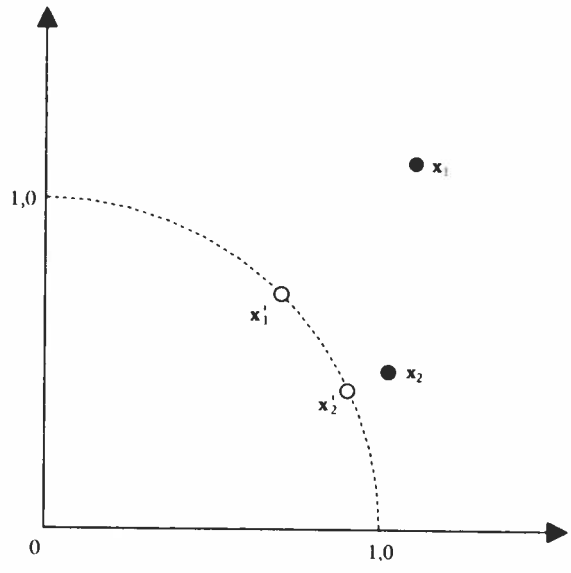


FIG. 1.9. Normalizzazione di due vettori. La normalizzazione riporta i due vettori originali (indicati dai puntini neri) a una distanza unitaria dall'origine.

In conclusione la rappresentazione dei pattern d'ingresso è molto importante per il risultato finale e merita un'accurata riflessione; si consideri che qualsiasi rete neurale può apprendere un determinato compito se i pattern di input sono stati opportunamente elaborati e rappresentati [Hertz, Krogh e Palmer 1991].

2.5. Apprendimento

La risposta di una rete neurale è determinata dai valori sinaptici delle connessioni fra i nodi. Il semplice impiego di connessioni inibito-

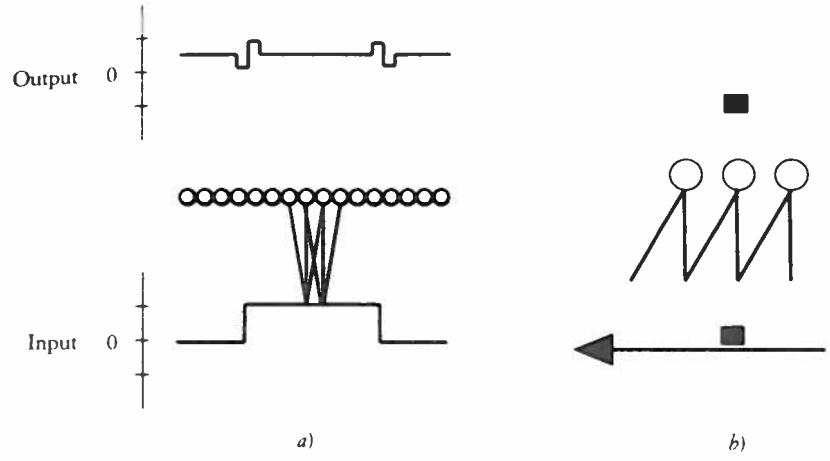


FIG. 1.10. Inibizione laterale (connessioni in grassetto) in una rete feedforward: *a*) estrazione di contorni; *b*) selettività di direzione nella percezione del movimento (vengono segnalati solamente oggetti in movimento verso sinistra).

rie laterali – una caratteristica molto comune dei circuiti biologici – è di per sé in grado di spiegare alcuni fenomeni percettivi.

L'effetto dell'inibizione laterale consiste nell'estrarre i contorni di una sagoma (fig. 1.10a) presentata come input aumentando la risposta dei neuroni che si trovano in corrispondenza di zone di contrasto e diminuendo la risposta dei neuroni che si trovano invece in zone omogenee. L'estrazione dei contorni avviene indipendentemente dall'illuminazione assoluta dell'immagine. Questo tipo di circuiteria, che è in grado di spiegare il fenomeno delle bande di Mach[8], il contrasto simultaneo di luminanza e l'illusione di Chevreul, svolge un ruolo importante nell'evidenziare le parti più informative dell'immagine retinica. L'inibizione laterale asimmetrica (fig. 1.10b) è invece in grado di spiegare la preferenza per una certa direzione di movimento nelle cellule gangliari della retina della maggior parte dei vertebrati [Shepherd 1997].

Tuttavia i casi in cui è possibile individuare a priori i valori sinaptici di una rete neurale che debba esibire un certo comportamento sono rari e ristretti a situazioni estremamente semplici. Così come i sistemi nervosi biologici sono in grado di apprendere in base all'esperienza, le reti neurali artificiali apprendono modificando gradualmente i propri valori sinaptici attraverso la presentazione ripetuta di una serie di esempi. Tradizionalmente si distinguono due modalità di apprendimento:

[8] Ernst Mach attribuì il fenomeno delle bande illusorie all'inibizione laterale 130 anni fa, ben prima che fosse fornita evidenza neurofisiologica [Mach 1986; trad. it. 1975; Ratliff 1965].

– nell'*apprendimento supervisionato* la modifica dei valori sinaptici avviene impiegando una misura di errore tra la risposta fornita dalla rete neurale e la risposta desiderata per ogni vettore di input. Questa procedura viene talvolta definita «apprendimento con insegnante» poiché si ipotizza che la risposta desiderata provenga da un insegnante esterno. L'addestramento viene eseguito su un gruppo di coppie di pattern, ciascuna formata dal pattern di input x e dal pattern di risposta desiderata t. L'apprendimento supervisionato include anche una gamma di algoritmi che richiedono solamente una misura di «bontà» della risposta della rete neurale, piuttosto che la specificazione della risposta esatta per ogni pattern d'addestramento: questi tipi di algoritmi sono definiti «apprendimento per rinforzo» (*reinforcement learning*). Anche i metodi di memorizzazione di pattern (come nel caso delle reti auto-associative) possono essere inclusi nella categoria dell'apprendimento supervisionato perché la rete neurale effettivamente è forzata ad associare fra di esse le varie parti che compongono un pattern.

– Nell'*apprendimento per auto-organizzazione* invece non esiste una risposta desiderata imposta dall'esterno, ma semplicemente la definizione di alcune semplici regole di plasticità sinaptica che danno luogo a una graduale auto-organizzazione della rete durante la fase di esposizione ai pattern di input. Le reti neurali che funzionano in base a questi principi sono in grado di estrarre informazione dall'ambiente, di classificare autonomamente una serie di pattern e di sviluppare delle configurazioni interne simili a quelle riscontrate in alcuni circuiti nervosi biologici.

I confini di questa distinzione non sono però ben demarcati perché da un lato è possibile utilizzare algoritmi di apprendimento supervisionato per scoprire strutture informative dell'ambiente senza conoscere necessariamente le risposte corrette (vedremo in seguito negli esempi in cui l'insegnante è l'ambiente stesso in cui la rete neurale opera); dall'altro lato molti algoritmi di apprendimento senza insegnante derivano da una precisa formulazione dell'informazione che deve essere estratta dall'ambiente e richiedono dettagliate assunzioni sulla struttura dei pattern d'ingresso. Vedremo infine nel sesto capitolo che un particolare tipo di apprendimento (gli algoritmi genetici) sfugge a una precisa classificazione di questo tipo perché si presta a essere utilizzato all'interno di entrambi i paradigmi.

Tutti gli algoritmi di apprendimento presentano comunque degli aspetti generali comuni:

– i valori iniziali dei pesi sinaptici della rete vengono assegnati in modo casuale entro un piccolo campo di variazione (ad esempio $[-0.1, 0.1]$) oppure vengono messi tutti a zero.

– L'apprendimento consiste nella presentazione ripetuta di una serie di vettori, detti anche pattern d'addestramento. Nell'apprendimento supervisionato ogni pattern è composto da una coppia, il vettore d'ingresso e il vettore della risposta desiderata; nell'apprendimento auto-organizzato invece vi sono solo i vettori d'ingresso. La modifica

dei valori sinaptici Δw_{ij} della rete viene calcolata dopo ogni presentazione di un singolo pattern («apprendimento per cicli», oppure *on-line*) oppure solo alla fine della presentazione di tutti i pattern d'addestramento («apprendimento per epoche»). La nuova configurazione di valori sinaptici dopo un ciclo (o epoca) di addestramento è calcolata addizionando la modifica ottenuta Δw_{ij}^t alla configurazione sinaptica precedente w_{ij}^{t-1},

$$w_{ij}^t = w_{ij}^{t-1} + \Delta w_{ij}^t$$

Gli algoritmi di apprendimento riguardano quindi solo il calcolo di Δw_{ij}^t.

– L'apprendimento riguarda dunque la sovrapposizione di nuove conoscenze Δw_{ij}^t su una base già consolidata di conoscenze precedenti w_{ij}^{t-1}. Per limitare il pericolo che le nuove conoscenze cancellino o stravolgano quanto già appreso, l'apprendimento procede in modo ricorsivo (ciascun pattern viene presentato più volte alla rete neurale) e graduale (solo una frazione della modifica sinaptica viene effettivamente addizionata ai valori sinaptici). La velocità dell'apprendimento è regolata da una costante η, detta «tasso di apprendimento», che controlla la porzione di modifica che viene applicata ai valori sinaptici; è quindi possibile riscrivere l'equazione generale di apprendimento come

$$w_{ij}^t = w_{ij}^{t-1} + \eta \Delta w_{ij}^t, \qquad 0 < \eta \leq 1$$

Matematicamente η può essere maggiore di 1, ma in questi casi si introducono potenziali elementi di instabilità. Il ruolo del tasso di apprendimento verrà illustrato in maggior dettaglio nel capitolo seguente.

– Una volta che la fase di apprendimento è stata completata (in base a un determinato criterio), i valori sinaptici vengono registrati («congelati») ed è possibile studiare la risposta della rete su dei vettori di test. La fase di test consiste nella presentazione di nuovi pattern d'ingresso e nel calcolo dell'attivazione dei nodi della rete senza però modificare i pesi sinaptici. È possibile così studiare le capacità di generalizzazione a nuovi stimoli ed effettuare altre analisi sul tipo di soluzione impiegata dalla rete per risolvere un determinato compito.

La distinzione tra fase di apprendimento e fase di test non si applica però agli algoritmi di apprendimento che operano all'interno della Teoria della Risonanza Adattiva [Grossberg 1987], ove le reti neurali sono in grado di attivare autonomamente il processo di apprendimento in presenza di nuovi pattern sconosciuti (si veda il quarto capitolo).

La capacità di apprendere rappresenta forse l'elemento di maggior attrazione dei modelli neurali perché permette di impiegare una rete neurale per risolvere problemi senza dover individuare direttamente la soluzione analitica, ma semplicemente esponendo il modello neurale a

una serie di esempi. Per questo motivo le reti neurali trovano vasta applicazione in problemi caratterizzati da trasformazioni non-lineari, per cui è difficile (o impossibile) trovare soluzioni ottimali con i metodi analitici. I prossimi capitoli presenteranno un'introduzione agli algoritmi di apprendimento più diffusi illustrandone le procedure, la logica di funzionamento e il tipo di operazioni matematiche che essi compiono sull'informazione disponibile.

2.6. Analisi vettoriale di un neurone artificiale

Prima di procedere ulteriormente, è utile osservare il funzionamento di una singola unità in termini vettoriali. Consideriamo la risposta di una semplice unità lineare

$$y = a\left(\sum_{i}^{N} w_i x_i\right), \qquad a = 1$$

che è riscrivibile come il prodotto interno tra il vettore **w** di valori sinaptici e il vettore **x** di input

$$y = \mathbf{w} \cdot \mathbf{x}$$

La risposta dell'unità, ovvero il valore scalare y, è una misura della «somiglianza» tra il vettore di input e il vettore di valori sinaptici (fig. 1.11). Come abbiamo già visto nel paragrafo 2.4, la lunghezza di un vettore è data dalla sua norma

$$\|\mathbf{x}\| = \sqrt{\mathbf{x} \cdot \mathbf{x}} = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$$

L'angolo ϑ formato dai due vettori **w** e **x** (o meglio, il coseno dell'angolo) è invece dato da

$$\cos \vartheta = \frac{\mathbf{w} \cdot \mathbf{x}}{\| \mathbf{w} \| \| \mathbf{x} \|}, \quad 0 \leq \vartheta \leq \pi$$

Quindi, il prodotto interno $y = \mathbf{w} \cdot \mathbf{x}$ è

$$\mathbf{w} \cdot \mathbf{x} = \|\mathbf{w}\| \|\mathbf{x}\| \cos \vartheta$$

Questo risultato significa che se immaginiamo di muovere nello spazio i due vettori mantenendo costante la loro lunghezza, il loro prodotto interno sarà proporzionale al coseno dell'angolo ϑ che viene a formarsi. Siccome

$$\begin{aligned} \vartheta &= 0° \quad \rightarrow \cos \vartheta = 1 \\ \vartheta &= 90° \quad \rightarrow \cos \vartheta = 0 \\ \vartheta &= 180° \quad \rightarrow \cos \vartheta = -1 \end{aligned}$$

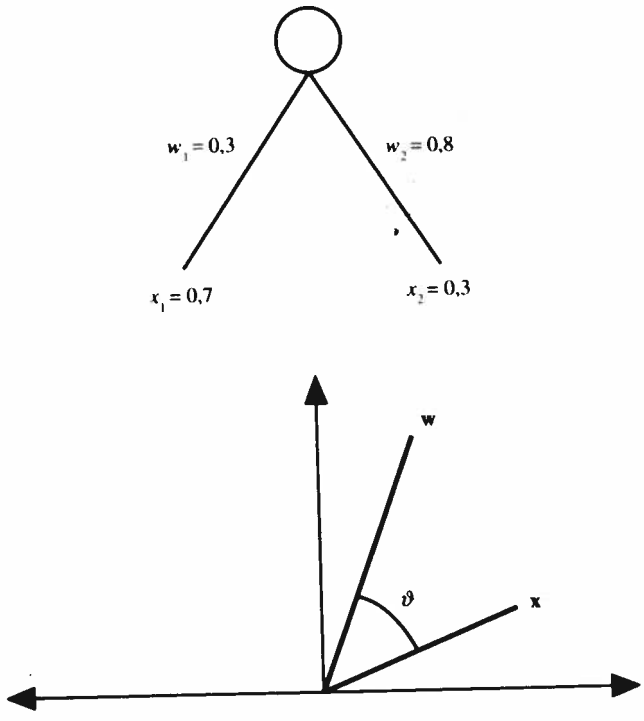


FIG. 1.11. Rappresentazione vettoriale dei pesi sinaptici e di un pattern di input per un'unità che possiede solo due sinapsi.

il prodotto interno (e quindi la risposta del neurone) sarà tanto maggiore quanto minore è la distanza angolare tra il vettore di input e il vettore sinaptico all'interno dello stesso quadrante; quando la risposta dell'unità è uguale a zero i due vettori sono ortogonali fra di loro (formano un angolo di 90 gradi); quando invece la distanza è maggiore di 90 gradi la situazione è simmetrica e l'unità assume valori negativi. Il neurone di McCulloch e Pitts, che utilizza una funzione di attivazione a gradino $\Phi(x) \in \{0,1\}$ con soglia uguale a zero, segnala quindi se il vettore di input si trova a una distanza inferiore (risposta = 1) o maggiore (risposta = 0) di 90 gradi dal proprio vettore sinaptico. Si ricordi che in una rete neurale con molte unità è possibile giudicare quale unità possieda il vettore sinaptico più simile al pattern di input in base al livello di attivazione solo se i vettori sinaptici sono stati normalizzati (possiedono tutti la stessa lunghezza).

Molti modelli neurali vengono utilizzati per classificare pattern. Nel caso di un neurone binario (o bipolare) la classificazione consiste nell'imparare ad attivarsi in presenza di un determinato gruppo di pattern di ingresso e a rimanere inattivo per tutti gli altri pattern. L'apprendimento consiste nello sviluppo di un vettore di valori sinaptici che individuano una linea di separazione nello spazio dell'input in

modo tale che tutti i pattern di un gruppo si trovino da un lato della linea e tutti gli altri pattern si trovino sul lato opposto. Se vi sono solamente due connessioni d'ingresso è lecito parlare di linea di separazione; se vi sono tre connessioni si ha un piano; infine, se vi sono più di tre connessioni, esse individuano un iperpiano di separazione nello spazio multidimensionale dell'input. Se tale linea, piano o iperpiano che separa i gruppi di input in due categorie esiste, si dice che il compito di classificazione è «linearmente separabile». La separabilità lineare dei vettori di input rappresenta un fattore importante per la scelta dell'architettura e del modello neurale: vedremo in seguito che tutte le reti neurali con un solo strato di sinapsi non sono in grado di classificare correttamente vettori di input che non siano linearmente separabili. La linea, piano o iperpiano di separazione è perpendicolare rispetto ai pesi sinaptici dell'unità (fig. 1.12). Consideriamo l'esempio di un semplice nodo ad attivazione bipolare con due connessioni d'ingresso. Siccome l'output, o decisione, del nodo è data da

$$y = \Phi(A) = \Phi\left(\sum_j w_j x_j - \vartheta\right)$$

la linea di separazione in due regioni, una in cui $A > 0$ e una in cui $A < 0$, è individuata dalla relazione

$$w_1 x_1 + w_2 x_2 - \vartheta = 0$$

ovvero

$$x_2 = \frac{\vartheta}{w_2} - \frac{w_1}{w_2} x_1$$

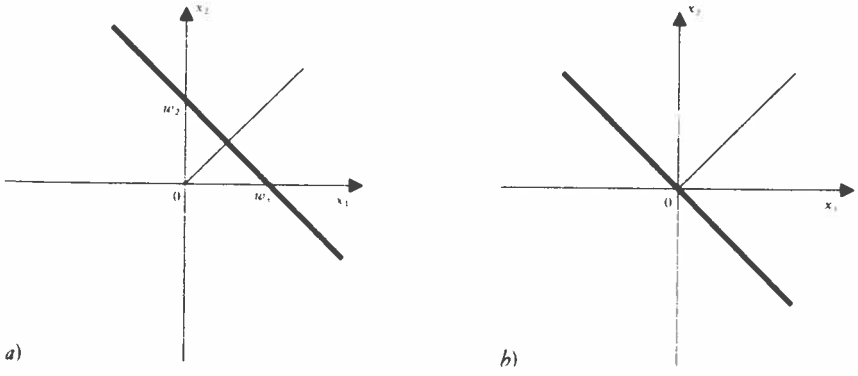


FIG. 1.12. La linea di separazione dello spazio di input individuata da un neurone con due connessioni d'ingresso entrambe fissate a 1: *a*) soglia $\vartheta = 1$, *b*) soglia $\vartheta = 0$. Linea sottile = direzione del vettore sinaptico, linea grossa = linea di separazione.

ove i pesi sinaptici e la soglia sono determinati dall'algoritmo di apprendimento. Tutti i pattern che si trovano nella regione $A > 0$ provocheranno output 1 mentre gli altri pattern provocheranno output -1. È facile osservare che se la soglia non viene utilizzata ($\vartheta = 0$), la linea di separazione passa necessariamente per l'origine dello spazio dell'input e potrebbe quindi limitare la capacità del neurone di separare certe distribuzioni di pattern.

Data una configurazione sinaptica iniziale casuale e una serie di pattern di input, l'apprendimento consiste nel muovere il vettore sinaptico in modo tale che la linea di separazione che esso individua separi correttamente i pattern in due gruppi distinti.

2.7. La soglia e il «bias»

Date le caratteristiche additive del calcolo dell'input netto del neurone, la soglia ϑ è equivalente a un peso sinaptico di valore ϑ collegato a un'ulteriore unità di input con attivazione costante -1. Il peso sinaptico si definisce «bias» e l'unità aggiuntiva «unità di *bias*»: essi vengono indicati rispettivamente con w_0 e x_0 (fig. 1.13).

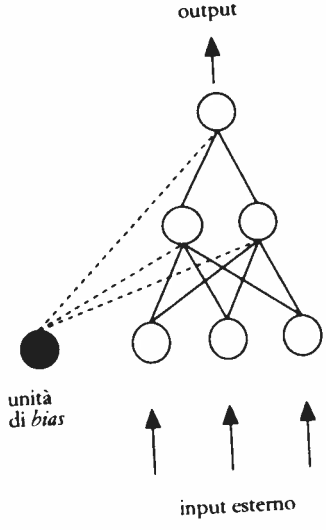


FIG. 1.13. Una rete neurale con unità di *bias* (cerchietto nero). Ciascuna unità possiede una connessione di *bias*; le unità di input non possiedono connessioni di *bias* perché la loro attivazione è data dall'elemento corrispondente del vettore di ingresso.

Adottando questa convenzione l'attivazione del neurone si indica semplicemente come

$$y = \Phi\left(\sum_{j=0} w_j x_j \right)$$

facendo partire la sommatoria da 0 e fissando $x_0 = -1$. Trattare la soglia come un ulteriore peso sinaptico proveniente da un'unità sempre attiva significa che essa può essere modificata con le stesse regole di apprendimento impiegata per gli altri pesi sinaptici della rete semplificando le notazioni matematiche e gli algoritmi di calcolo.

2.8. Strumenti di valutazione e analisi di una rete neurale

Per quanto tempo bisogna addestrare una rete neurale? I metodi di valutazione variano a seconda del paradigma di apprendimento. Nel caso dell'apprendimento supervisionato si utilizza l'errore medio fra la risposta desiderata e la risposta fornita dalla rete neurale per ciascun pattern di addestramento: quando l'errore raggiunge un livello minimo prefissato (criterio), i valori sinaptici vengono registrati e si passa alla fase di test. Il livello minimo di errore dipende da molti fattori: il grado di precisione che si vuole ottenere, la funzione di attivazione utilizzata, il numero di unità di output e la distribuzione dei pattern nello spazio. Inoltre, come vedremo in maggior dettaglio nel capitolo seguente, vi è un compromesso tra la precisione di apprendimento dei pattern di addestramento e la capacità di generalizzare a nuovi pattern di test. Per l'apprendimento senza insegnante il criterio invece cambia poiché non esiste una misura di errore. Per alcuni di questi algoritmi è possibile definire una funzione di «energia» o «funzione oggettiva» dei parametri della rete, la quale fornisce una misura continua della convergenza della rete verso una soluzione. Altrimenti si possono utilizzare altri indici, come ad esempio una misura di variazione delle sinapsi (il raggiungimento di una soluzione di solito corrisponde a una situazione di stabilità sinaptica) o una misura di variazione dello stato di attivazione dei nodi della rete per ogni singolo pattern.

La valutazione della qualità della risposta di una rete neurale addestrata dipende dalle finalità d'impiego. I criteri di valutazione per le applicazioni ingegneristiche possono essere molto diversi dai criteri utilizzati nella modellizzazione psicologica o negli studi neurocomputazionali. Nel primo caso si è interessati alla percentuale di errori come, ad esempio, classificazioni scorrette e risposte miste (in cui un pattern di ingresso provoca l'attivazione contemporanea di unità di uscita corrispondenti a diverse categorie). Un possibile metodo per ottenere queste misure – nel caso di paradigmi supervisionati – consiste nel calcolare la distanza di Hamming[9] fra la versione binarizzata del pattern fornito dalla rete neurale e il pattern corretto. Nello studio di modelli neurocomputazionali e nella modellizzazione cognitiva i criteri possono essere diversi. Ad esempio è interessante osservare in-

[9] La distanza di Hamming ρ_H è una misura di somiglianza tra vettori binari che consiste semplicemente nel numero di bit diversi in posizioni corrispondenti dei due vettori. Ad esempio, per i due vettori $\mathbf{x} = \{1,0,1,1,0\}$ e $\mathbf{y} = \{1,0,0,1,0\}$ $\rho_H(\mathbf{x},\mathbf{y}) = 1$.

nanzitutto le caratteristiche della fase di apprendimento: è possibile cogliere delle somiglianze tra la curva di apprendimento della rete neurale (individuata ad esempio dalla funzione obiettivo o dalla percentuale di risposte scorrette) e quella dei soggetti umani? quali sono i pattern che la rete neurale apprende con maggior facilità durante l'addestramento? quali sono le influenze delle manipolazioni della fase di apprendimento (presentazione casuale o presentazione nello stesso ordine dei pattern; addestramento su un gruppo delimitato di pattern in un primo tempo e addestramento successivo su un gruppo diverso, ecc.)? Un vantaggio dei modelli neurali rispetto ai modelli cognitivisti tradizionali consiste per l'appunto nella possibilità di studiare le dinamiche di sviluppo del modello neurale: questa possibilità non solo permette di comprendere meglio il funzionamento del modello finale, ma permette anche di riprodurre e studiare computazionalmente i risultati di molti esperimenti di psicologia sull'apprendimento in soggetti animali e umani. Una volta che la rete neurale è stata addestrata, è necessario confrontare le risposte dei neuroni di uscita con le risposte dei soggetti. Siccome non è possibile «interrogare» una rete neurale, come si può invece fare con i soggetti, è utile pianificare l'architettura e la metodologia simulativa fin dall'inizio in modo da poter riprodurre le situazioni sperimentali a cui si è interessati. Nella letteratura sperimentale le prestazioni dei soggetti vengono spesso espresse in termini di percentuali di risposte scorrette e tempi di reazione. Ma mentre i soggetti umani e animali possiedono un apparato sensomotorio o linguistico che permette la misurazione di queste variabili (tempi di risposta nel premere un pulsante, ripetizione di parole, lettura di testi, ecc.), una rete neurale fornisce semplicemente l'attivazione dei nodi di uscita. Piuttosto che cercare un confronto diretto fra le misure ottenute con i soggetti umani e la risposta del modello neurale, è più proficuo individuare dei confronti sulle proprietà delle prestazioni: ad esempio, nel caso di paradigmi di apprendimento supervisionato, quali sono i pattern d'ingresso che provocano un maggior errore? vi è una tendenza sistematica a rispondere scorrettamente per alcuni pattern rispetto ad altri? quando vi è un errore, quali sono le caratteristiche della risposta fornita dalla rete? In generale bisogna notare comunque che la misura di errore (qualunque essa sia) di una rete neurale è un indice del grado di difficoltà nell'elaborare un pattern d'ingresso; date queste premesse è lecito paragonare l'errore della rete neurale (ad esempio la somma delle discrepanze quadratiche medie fra il vettore di output della rete e il vettore della risposta corretta, oppure la distanza di Hamming, o altre ancora, a seconda del modello utilizzato) con le percentuali di risposte scorrette e i tempi di reazione dei soggetti. I dati importanti nei confronti non sono le scale di misura, ma l'andamento delle prestazioni nelle varie condizioni sperimentali. La letteratura presenta moltissimi esempi di confronti di questo tipo, in cui le prestazioni delle reti neurali ricalcano quelle dei soggetti umani.

Un ulteriore tipo di valutazione riguarda l'analisi interna della rete

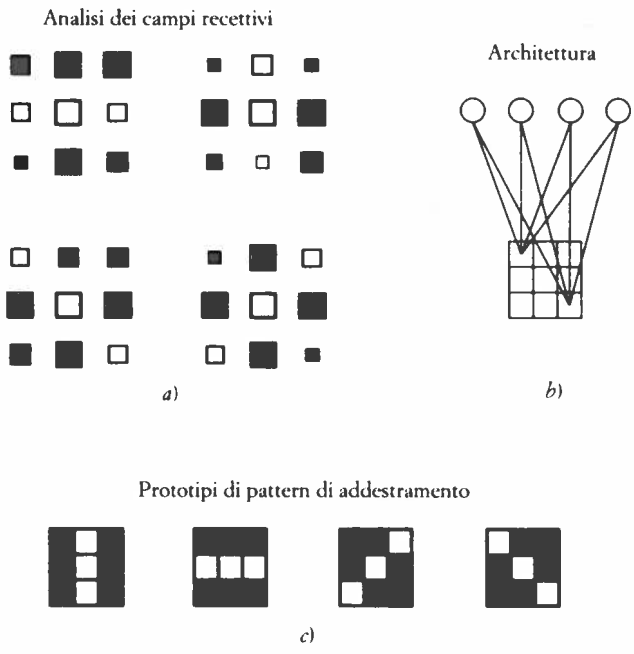


FIG. 1.14. Analisi dei pesi sinaptici di una rete feedforward con 4 unità di uscita che ricevono segnali da 9 unità d'ingresso. *a*) Ciascun blocco rappresenta i pesi sinaptici di un nodo; la dimensione dei quadratini rappresenta la forza del peso sinaptico, il colore indica il segno (nero = negativo; bianco = positivo). *b*) Ciascun nodo riceve segnali da tutta la matrice di input (sono rappresentati solo alcuni pesi sinaptici per chiarezza grafica). *c*) Durante la fase di addestramento la rete neurale vede una successione casuale di linee orizzontali, verticali e diagonali di segno positivo su uno sfondo di segno negativo in cui l'intensità di ciascun singolo elemento varia casualmente. Ciascuna unità impara a segnalare la presenza di una linea con una certa inclinazione, come è possibile osservare anche dalla configurazione di pesi sinaptici risultanti. Le variazioni delle forze sinaptiche riflettono le variazioni d'intensità dei pattern durante l'addestramento.

neurale dopo la fase di apprendimento. L'indagine della configurazione di valori sinaptici e delle caratteristiche di attivazione dei nodi interni può fornire utili indici sul tipo di soluzione impiegato dalla rete per risolvere un certo compito. Immaginiamo innanzitutto di voler analizzare una semplice rete feedforward con un solo strato di sinapsi. La visualizzazione dei «campi recettivi» consiste nella proiezione sul piano dei valori sinaptici di ciascun neurone (fig. 1.14); siccome valori assoluti più grandi corrispondono ad associazioni più forti, questa analisi ci può informare sulla specializzazione di risposta del neurone, ovvero sugli stimoli di input che contribuiscono ad attivarlo maggiormente. Nel caso di input ad alta dimensionalità, il gioco di inibizione ed eccitazione può rendere questa analisi abbastanza complessa. Un metodo diverso consiste allora nel ricavare le caratteristiche dei campi recettivi dalla misurazione dell'attività dei nodi stessi: l'attivazione di

un nodo viene misurata mentre vari tipi di stimoli (di forma e intensità diverse) vengono presentati nel suo campo recettivo. Questo metodo ci permette di costruire delle curve di risposta che evidenziano la selettività del nodo per determinate caratteristiche dello stimolo; esso si rivela molto utile per l'analisi dei nodi interni delle reti multistrato: in alcune simulazioni questo metodo ha permesso di evidenziare caratteristiche di risposta simili a quelle dei neuroni biologici [Zipser e Andersen 1988]. La visualizzazione dell'attivazione di un singolo nodo interno non ci dice molto però su come la rete organizza internamente i vari stimoli d'ingresso. Questa informazione si ottiene facendo un'analisi dei gruppi (cluster analysis) sull'attivazione di tutti i nodi interni per ciascun pattern d'ingresso: l'albero risultante indica i raggruppamenti (ovvero le somiglianze «percepite» dalla rete) degli stimoli in funzione dell'attivazione interna.

Oltre a questi metodi di valutazione «standard», il lettore interessato troverà molte variazioni nella letteratura specifica: in generale il metodo di valutazione mette in risalto gli aspetti importanti del modello in funzione delle finalità della ricerca. In molti esperimenti lo scopo non è solo quello di riprodurre dati sperimentali esistenti, ma quello di spiegare il funzionamento e lo sviluppo dei meccanismi responsabili per un certo tipo di prestazioni e di fornire possibilmente nuove predizioni.

3. Apprendimento hebbiano

La regola di modifica sinaptica di Hebb – e le sue variazioni – costituisce le fondamenta su cui si basano o derivano tutti gli algoritmi di apprendimento che prenderemo in esame nei prossimi capitoli. Per quanto semplici, le regole hebbiane sono fra i pochi meccanismi di apprendimento che godono di evidenza neurofisiologica e definiscono quindi una cornice e un punto di partenza su cui sviluppare modelli più complessi. Nelle sezioni seguenti esamineremo l'uso delle regole hebbiane in una rete etero-associativa feedforward con un singolo strato di sinapsi e con unità binarie. Il compito consiste nell'associare una serie di pattern d'ingresso con una serie di pattern di uscita ed è quindi un esempio di apprendimento supervisionato poiché indichiamo esplicitamente alla rete neurale quali siano i tipi di associazione che devono essere appresi.

3.1. La regola di Hebb

Nel 1949 lo psicologo canadese Donald Hebb [Hebb 1949; trad. it. 1975] affermò che le leggi del condizionamento classico riflettevano le proprietà di funzionamento degli elementi costitutivi del sistema nervoso, ovvero dei singoli neuroni. La regola di Hebb dice che se

due neuroni collegati fra di loro sono contemporaneamente attivi, l'efficacia sinaptica della connessione viene rinforzata. In seguito a questo rafforzamento, la sola attivazione del nodo presinaptico sarà sufficiente a causare l'attivazione del nodo postsinaptico. Questa associazione verrà ulteriormente rinforzata ogni volta che i due nodi saranno attivi contemporaneamente. Consideriamo ora una rete neurale in cui i nodi di uscita possiedono una funzione di attivazione binaria. Per ogni presentazione di un pattern d'ingresso **x** e di un pattern di uscita **y**, la modifica dei pesi sinaptici è data da

$$\Delta w_{ij} = \eta y_i x_j$$

dove η è il tasso di apprendimento. Se iniziamo la fase di apprendimento da una matrice di valori sinaptici w_{ij} uguale a 0 e fissiamo $\eta = 1$, la matrice finale sarà uguale alla somma di tutte le modifiche calcolate per ciascuna coppia di pattern μ

$$w_{ij} = \sum_{\mu} \Delta w_{ij}^{\mu}$$

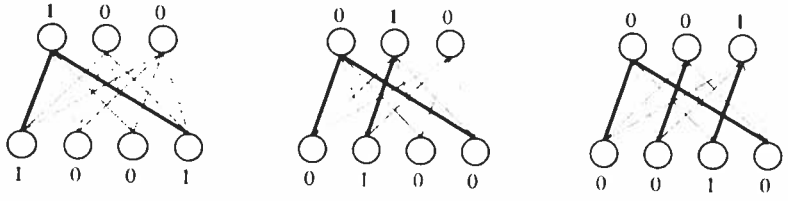


FIG. 1.15. Apprendimento di tre associazioni con la regola di Hebb. Le connessioni disegnate in grigio possiedono valore sinaptico zero, quelle evidenziate in nero possiedono valore 1.

La figura 1.15 illustra l'apprendimento di tre associazioni: la matrice sinaptica viene modificata per ogni presentazione. Una volta che le tre coppie sono state apprese, è sufficiente presentare un pattern d'ingresso per elicitare il pattern d'uscita corrispondente. Inoltre, se presentiamo una versione indebolita o incompleta di un pattern d'ingresso, la rete neurale è in grado di fornire ugualmente il pattern di uscita corretto. Tuttavia, siccome la regola di Hebb prevede solamente l'incremento delle sinapsi, la rete non è in grado di apprendere associazioni che presentano elementi in comune ma che richiedono risposte diverse. Un pattern di input che possiede elementi in comune con un altro pattern di input causa l'attivazione dei nodi di output corrispondenti a entrambi i pattern («risposta mista»). La regola di Hebb permette quindi di apprendere solamente pattern ortogonali (in cui la somma dei prodotti dei singoli componenti è zero). La produzione di risposte miste dovute alla sovrapposizione di pattern di ingresso viene definita con il nome di «interferenza».

3.2. La regola postsinaptica

Alcune limitazioni della regola di Hebb possono essere superate se si introduce la possibilità di ridurre l'efficacia sinaptica. La regola postsinaptica, detta anche regola di Stent-Singer (dal nome dei due neurofisiologi che hanno evidenziato questo meccanismo nei circuiti biologici [si veda Singer 1987; Stent 1973]), prevede che il valore della connessione sinaptica venga incrementato ogni volta che l'unità postsinaptica e l'unità presinaptica sono entrambe attive, ma venga diminuito ogni volta che l'unità postsinaptica è attiva, ma quella presinaptica è inattiva, come segue

$$\Delta w_{ij} = \eta(y_i x_j + (x_j - 1)\, y_i)$$

Questo meccanismo permette la riduzione del fenomeno di interferenza; tuttavia, quando si vogliono associare molti pattern di input parzialmente sovrapposti con lo stesso pattern di output, questa regola non è in grado di apprendere correttamente perché tende a creare troppe sinapsi inibitorie.

Per inciso, si noti che durante il processo di apprendimento una sinapsi può facilmente invertire il proprio segno in seguito all'accumulazione delle modifiche. Questa caratteristica è propria di quasi tutti i modelli neurali presi in considerazione in questo libro, ma non trova riscontro nei sistemi nervosi biologici.

3.3. La regola presinaptica

La regola presinaptica prevede invece una situazione simmetricamente opposta alla regola di Stent-Singer (si veda Stanton e Sejnowski [1989] per l'evidenza biologica), in cui il valore della connessione sinaptica viene aumentato quando sia l'unità presinaptica e l'unità postsinaptica sono attive, ma viene diminuito quando l'unità presinaptica è attiva e quella postsinaptica è inattiva, come segue

$$\Delta w_{ij} = \eta(y_i x_j + (y_i - 1)\, x_j)$$

Essa funziona meglio della regola postsinaptica quando molti pattern di input diversi e parzialmente sovrapposti debbono essere associati con lo stesso pattern.

3.4. La regola della covarianza

Questa regola, che è la combinazione delle regole precedenti, viene spesso anche definita regola di Hopfield [Hopfield 1982], dal nome del fisico che l'ha inizialmente introdotta (anche se in un conte-

sto leggermente diverso). Quando l'unità postsinaptica e quella presinaptica sono nello stesso stato (entrambe attive o entrambe inattive) la connessione viene rinforzata; quando invece le due unità sono in stati diversi, la connessione viene indebolita. La tabella 1.1 riassume il funzionamento di queste quattro regole hebbiane.

TAB. 1.1. *Funzionamento delle regole hebbiane: variazione dei pesi sinaptici in funzione dell'attività pre- e post-sinaptica*

Unità presinaptica	+	+	–	–
Unità postsinaptica	+	–	+	–
Hebb	+			
Postsinaptica	+		–	
Presinaptica	+	–		
Covarianza	+	–	–	+

Si noti che, nel caso in cui i nodi della rete utilizzano un'attivazione bipolare, la regola della covarianza assume la stessa forma della regola di Hebb: in tal caso essa viene definita come «regola di Hebb estesa». La regola di Hebb estesa è in grado di svolgere compiti di associazione e classificazione abbastanza complessi.

Regola di Hebb estesa: algoritmo

Data una rete neurale feedforward con N unità di input bipolari e una singola unità di output con attivazione a gradino bipolare:

1. Inizializzare i pesi sinaptici (*bias* incluso):

$$w_i = 0 \text{ (per } i = \{0, 1, ..., N\})$$

2. Per ciascuna coppia di pattern di input **s** e di output t eseguire 3-5

3. Calcolare attivazione unità di ingresso e di uscita

$$x_i = s_i$$
$$y = t$$

4. Calcolare le modifiche sinaptiche

$$\Delta w_i = \eta y x_i$$

5. Aggiornare i pesi sinaptici

$$w_i = w_i^{t-1} + \Delta w_i$$

3.4.1. *Esempio: apprendimento della funzione logica AND*

Consideriamo una rete con 2 unità di input, un'unità di *bias* e un'unità di output con funzione di attivazione a gradino (bipolare o binaria); fissiamo il tasso di apprendimento η a 1. La funzione AND richiede una risposta positiva quando entrambe le unità di input sono attive e la risposta opposta in tutti gli altri casi. Inizialmente tutte le connessioni assumono valore 0. Esploriamo l'apprendimento di questa funzione in due condizioni: unità bipolari e unità binarie.

Unità bipolari

Input			Uscita	Modifica			Pesi finali		
x_0	x_1	x_2	t	Δw_0	Δw_1	Δw_2	w_0	w_1	w_2
−1	1	1	1	−1	1	1	−1	1	1
−1	1	−1	−1	1	−1	1	0	0	2
−1	−1	1	−1	1	1	−1	1	1	1
−1	−1	−1	−1	1	1	1	2	2	2

La rete neurale apprende ad associare la risposta corretta per tutti i pattern d'ingresso. La figura 1.16 mostra l'evoluzione della linea di separazione

$$x_2 = \frac{w_0}{w_2} - \frac{w_1}{w_2} x_1$$

durante l'apprendimento.

Le cose vanno diversamente quando si utilizzano invece unità binarie.

Unità binarie

Input			Uscita	Modifica			Pesi finali		
x_0	x_1	x_2	t	Δw_0	Δw_1	Δw_2	w_0	w_1	w_2
1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	0	0	1	1	1
1	0	1	0	0	0	0	1	1	1
1	0	0	0	0	0	0	1	1	1

La rete apprende la prima associazione, ma non apprende le associazioni seguenti perché il pattern d'uscita è sempre 0. Siccome i pesi sinaptici non subiscono ulteriori modifiche, la linea di separazione rimane collocata nel posto scorretto e la rete non è in grado di fornire

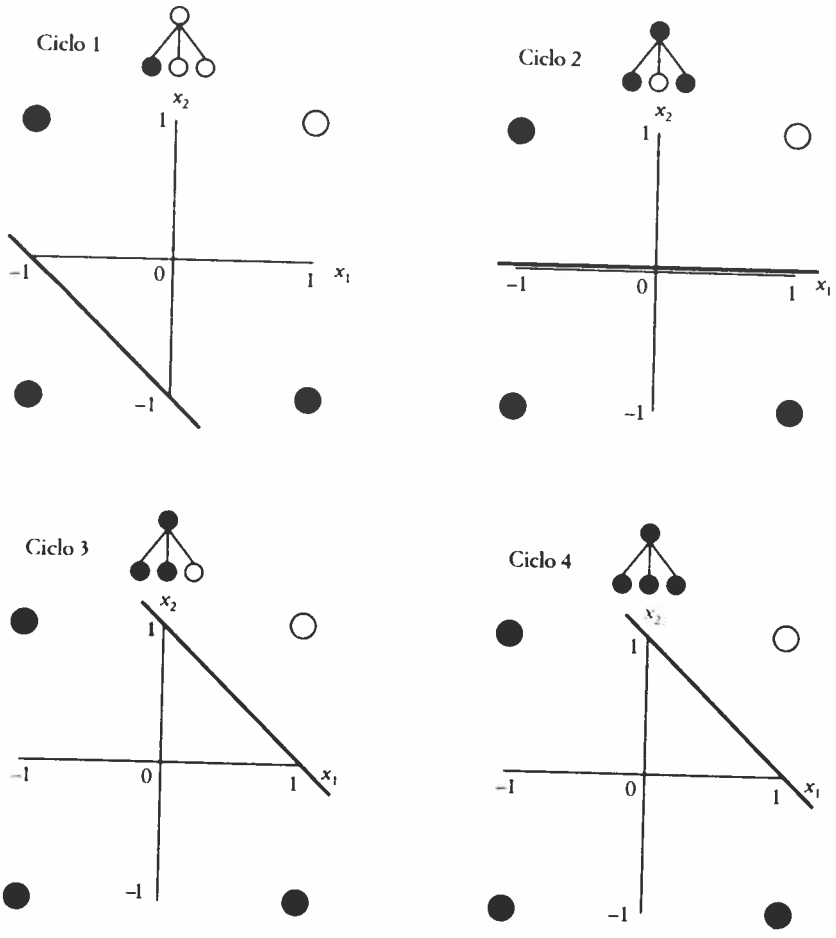


FIG. 1.16. Evoluzione della linea di separazione durante l'apprendimento della funzione AND con unità bipolari. I cerchietti all'interno del grafico indicano la risposta desiderata per ciascuno dei quattro possibili pattern binari; cerchietti neri = risposta desiderata −1; cerchietti bianchi = risposta desiderata +1. Sopra ciascun grafico è raffigurata la rete neurale con il pattern di addestramento corrispondente (cerchietti neri = −1; cerchietti bianchi = 1). La prima unità della rete è l'unità di *bias* con valore costante −1.

la risposta desiderata per i restanti tre pattern d'ingresso. In questo specifico caso (rappresentazione binaria sia per le unità di ingresso che per le unità di uscita) la regola di Hebb estesa è formalmente identica alla regola di Hebb semplice perché è in grado solamente di rafforzare le connessioni di unità attive. La regola di Hebb estesa fallisce nell'apprendere la funzione AND anche quando si usano unità d'ingresso binarie e unità di uscita bipolari.

3.5. Apprendimento ottimale con le regole hebbiane

Willshaw e Dayan [1990] hanno studiato le capacità di apprendimento della regola presinaptica, postsinaptica e di covarianza. Essi hanno mostrato che l'apprendimento ottimale (con il minor numero di errori possibili) dipende dalla proporzione p di unità attive nei pattern di input e dalla proporzione r di unità attive nei pattern di output per ciascuna coppia di addestramento. Gli autori hanno individuato per ciascuna regola le quantità ottimali di modifica in funzione dello stato dell'unità presinaptica e dell'unità postsinaptica. Per la regola postsinaptica

$$\Delta w^{+}_{++} = 1 - p, \qquad \Delta w^{-}_{+-} = p$$

dove gli apici indicano la direzione della modifica (positiva o negativa) e gli indici indicano gli stati delle unità postsinaptica e presinaptica rispettivamente.

Per la regola presinaptica invece

$$\Delta w^{+}_{++} = 1 - r, \qquad \Delta w^{-}_{-+} = r$$

Infine, per la regola della covarianza

$$\Delta w^{+}_{++} = (1 - p)(1 - r), \quad \Delta w^{-}_{--} = pr, \quad \Delta w^{-}_{-+} = (1 - p)r, \quad \Delta w^{-}_{+-} = (1 - r)p$$

Malgrado questi aggiustamenti, le regole di Hebb possiedono parecchie limitazioni nel tipo di associazioni che sono in grado di apprendere poiché sono spesso soggette a fenomeni di interferenza quando i pattern d'ingresso non sono linearmente indipendenti.

Capitolo 2

1. L'insegnante esterno

L'apprendimento supervisionato è caratterizzato dalla presenza di un «insegnante esterno[1]» che fornisce informazione utilizzata per la correzione della risposta della rete neurale. Questa informazione può essere la *risposta desiderata* dalla rete per ciascun pattern di apprendimento («apprendimento con insegnante» propriamente detto) oppure una valutazione della *bontà della risposta* della rete («apprendimento con critico» o «apprendimento per rinforzo»).

Gli esempi di apprendimento hebbiano considerati nel capitolo precedente sono effettivamente un esempio di apprendimento supervisionato perché le reti vengono addestrate ad associare ciascun pattern di input con un determinato pattern di risposta; tuttavia l'apprendimento hebbiano supervisionato è un caso abbastanza anomalo perché non è necessario calcolare l'attivazione dei nodi di output durante la fase di addestramento. Al contrario, tutti gli algoritmi considerati in questo capitolo si basano sul confronto tra la risposta fornita dalla rete e la risposta desiderata (o indice di valutazione).

2. Percettroni semplici

L'idea di modificare i pesi sinaptici di una rete neurale artificiale in base all'errore generato dal confronto tra la risposta della rete e la risposta desiderata fu inizialmente proposta da Rosenblatt [1962]. I percettroni di Rosenblatt erano delle reti neurali con due strati di connessioni impiegati soprattutto per apprendere a riconoscere dei caratteri proiettati su una matrice di recettori fotosensibili. Le sinapsi del

[1] Per «esterno» si intende estraneo alla rete neurale: un segnale esterno è un segnale che non è generato dalla rete neurale.

primo strato erano fissate in modo casuale e non potevano essere modificate: esse erano un semplice stratagemma per riportare l'attivazione dei recettori sui nodi di input della rete neurale. Le sinapsi del secondo strato erano invece sottoposte ad apprendimento. Per questo motivo, quando si parla di percettroni, ci si riferisce a una rete neurale con un unico strato di connessioni unidirezionali dai nodi di input ai nodi di uscita. Rosenblatt e altri autori [Block 1962; Minsky e Papert 1969; 1988] hanno descritto vari tipi di percettroni le cui differenze riguardano principalmente il tipo di nodi utilizzati (binari, bipolari e bipolari con una zona di indecisione): essi sono anche detti «percettroni semplici» perché i nodi di uscita utilizzano una semplice funzione a gradino. Nelle prossime sezioni esamineremo innanzitutto una semplice versione con un unico nodo di uscita ad attivazione bipolare; vedremo poi che la stessa regola sinaptica è utilizzabile anche con altri tipi di funzioni di attivazione e in reti con molteplici nodi di uscita.

2.1. Regola di apprendimento

Data una rete neurale con N nodi d'ingresso e P coppie di addestramento, ciascuna composta da un vettore d'ingresso bipolare $\mathbf{x}_p$ e da una risposta desiderata («target» o «teaching input») t_p anch'essa bipolare, l'output della rete per ciascun pattern d'ingresso è dato da

$$y = \begin{cases} 1 & \text{se } \sum_{i=0}^{N} w_i x_i > 0 \\ -1 & \text{altrimenti} \end{cases}$$

questo valore viene poi confrontato con la risposta desiderata t_p per quel determinato pattern d'ingresso. Se la risposta della rete è uguale alla risposta desiderata i valori sinaptici non vengono modificati; se invece si è verificato un errore, i pesi sinaptici vengono modificati in base alla risposta corretta,

$$\Delta w_i = \eta t x_i$$

questo valore viene poi addizionato ai valori precedenti delle sinapsi

$$w_i^t = w_i^{t-1} + \Delta w_i^t$$

I valori iniziali delle connessioni sinaptiche possono essere uguali a zero, oppure assumere piccoli valori casuali, o – infine – essere uguali a uno qualsiasi dei pattern d'addestramento [Minsky e Papert 1988]. Il tasso di apprendimento η viene di solito fissato a 1, ma può anche essere fissato a $1/\|\mathbf{x}\|$ per ciascun pattern che provoca un cambiamento

dei pesi (in questo modo il vettore di modifica dei pesi sinaptici possiede lunghezza 1).

I percettroni modificano i pesi sinaptici solamente per i pattern d'ingresso che provocano risposte scorrette e richiedono molte presentazioni della lista di coppie di addestramento: man mano che l'addestramento procede le modifiche sinaptiche diventano sempre più rare perché un numero sempre minore di pattern provoca risposte scorrette. L'apprendimento cessa automaticamente quando nessun pattern provoca errori di risposta. I percettroni possiedono due importanti caratteristiche:

- l'apprendimento è graduale e richiede la presentazione ripetuta di tutte le coppie di addestramento;
- se esiste un vettore di valori sinaptici che soddisfa la trasformazione richiesta da input ad output, la regola d'apprendimento è in grado di trovare tale vettore (o un vettore simile, se esso non è unico) in un numero finito di cicli di apprendimento.

La seconda caratteristica è forse la più sorprendente (senz'altro entusiasmò Rosenblatt e la comunità scientifica dell'epoca) perché indica che se esiste una soluzione un semplice percettrone è in grado di trovarla in un numero finito di cicli di apprendimento. La condizione necessaria affinché vi sia una soluzione è che i pattern siano linearmente separabili. Data l'importanza storica e concettuale di questa proprietà, in una prossima sezione esamineremo la prova della convergenza dei percettroni semplici.

Tornando al tipo di codifica, è facile notare che la regola di apprendimento e le proprietà del percettrone si applicano anche a input binari: in questo caso le modifiche riguardano solamente i pesi sinaptici che corrispondono a unità d'ingresso attive che hanno provocato una risposta scorretta. I percettroni sono molto più potenti della regola di Hebb: ad esempio, mentre la regola di Hebb estesa non è in grado di apprendere la funzione AND quando si usano input binari, in condizioni analoghe un percettrone trova il vettore di valori sinaptici in 10 epoche (un'epoca è una presentazione di tutte le coppie di addestramento) [Fausett 1994]. Si noti comunque l'analogia tra la forma della regola di Hebb estesa e la forma della regola del percettrone: in entrambi i casi la modifica sinaptica è funzione di un evento presinaptico e di un evento postsinaptico.

Lo stesso algoritmo di apprendimento si applica anche a reti neurali con più di un nodo di output: infatti, siccome ciascun nodo di output è indipendente dagli altri nodi, la rete è scomponibile in tanti semplici percettroni, ciascuno con un singolo nodo di uscita che possiede lo stesso vettore di input ma diverse connessioni sinaptiche.

Percettrone semplice: algoritmo

Data una rete neurale con N unità di input e una singola unità di output con attivazione bipolare:

1. Inizializzare i pesi sinaptici (*bias* incluso):

$$w_i = 0 \quad (\text{per } i = \{0, 1, ..., N\})$$

Fissare il tasso di apprendimento in modo tale che

$$0 < \eta \leq 1 \text{ (ad esempio 1)}$$

2. Fino a quando $y \neq t$ anche per una sola coppia di addestramento, eseguire 3-7.
3. Per ciascuna coppia di addestramento **s**, t.
4. Stabilire i valori dei nodi d'ingresso:

$$x_i = s_i$$

5. Calcolare l'attivazione del nodo di uscita

$$y = \begin{cases} 1 & \text{se } \sum_{i=0}^{N} w_i x_i > 0 \\ -1 & \text{altrimenti} \end{cases}$$

6. Calcolare la modifica dei pesi sinaptici:

$$\Delta w_i = \begin{cases} 0 & \text{se } y = t \\ \eta t x_i & \text{altrimenti} \end{cases}$$

7. Modificare i pesi sinaptici:

$$w_i = w_i^{t-1} + \Delta w_i$$

2.2. Prova della convergenza del percettrone semplice

È dimostrabile che, se esiste un vettore di pesi sinaptici che costituisce la soluzione, un percettrone semplice converge verso tale soluzione in un numero finito di presentazioni dell'intera lista di pattern di addestramento. La prova riportata qui sotto si basa sulla dimostrazione in notazione vettoriale di Minsky e Papert [1988], ma altri autori hanno fornito anche linee di ragionamento leggermente diverse per giungere alla dimostrazione [Hertz, Krogh e Palmer 1991; Rojas 1996; Reed e Marks 1999].

Date P coppie di addestramento, scomponiamo innanzitutto il gruppo di addestramento in funzione del tipo di risposta desiderata in due insiemi:

$$F^+ = \{\mathbf{x} \text{ tale che } t = +1\}$$

e

$$F^- = \{\mathbf{x} \text{ tale che } t = -1\}$$

in modo tale che l'esistenza di una soluzione corrisponde all'esistenza di un vettore di pesi sinaptici $\mathbf{w}^*$ per cui

$$\mathbf{x} \cdot \mathbf{w}^* > 0 \quad \text{se } \mathbf{x} \in F^+$$

e

$$\mathbf{x} \cdot \mathbf{w}^* < 0 \quad \text{se } \mathbf{x} \in F^-$$

Immaginiamo ora di creare un nuovo insieme

$$F = F^+ \cup -F^-$$

dove

$$-F^- = \{-(\mathbf{x}), \quad \text{tale che } \mathbf{x} \in F^-\}$$

in pratica abbiamo invertito il segno degli elementi dei vettori di input che corrispondevano alla risposta desiderata -1 in modo tale che ora l'esistenza della soluzione $\mathbf{w}^*$ è assicurata semplicemente dalla condizione

$$\mathbf{x} \cdot \mathbf{w}^* > 0 \quad \text{se } \mathbf{x} \in F$$

perché tutte le risposte desiderate del nuovo insieme di addestramento sono $+1$.

La regola di apprendimento prescrive che ogni volta che un vettore di ingresso provoca una risposta scorretta, ovvero

$$\mathbf{x} \cdot \mathbf{w} \leq 0$$

i pesi sinaptici vengano modificati come segue

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \eta t \mathbf{x}$$

dove k indica l'istante temporale; siccome nel nuovo insieme F abbia-

mo sempre che $t = +1$ e assumendo per semplicità che $\eta = 1$, possiamo riscrivere la regola di apprendimento come

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \mathbf{x}$$

Immaginiamo ora di presentare ciascun pattern di addestramento (in modo ciclico) alla rete e di contare solamente le occasioni in cui si verifica un errore e i pesi sinaptici vengono modificati; in tal caso avremo

$$\begin{aligned}\mathbf{w}(1) &= \mathbf{w}(0) + \mathbf{x}(0)\\ \mathbf{w}(2) &= \mathbf{w}(1) + \mathbf{x}(1)\\ \mathbf{w}(3) &= \mathbf{w}(2) + \mathbf{x}(2)\\ \mathbf{w}(k) &= \mathbf{w}(k-1) + \mathbf{x}(k-1)\end{aligned}$$

Quindi

$$\mathbf{w}(k) = \mathbf{w}(0) + \mathbf{x}(0) + \mathbf{x}(1) + \mathbf{x}(2) + \ldots + \mathbf{x}(k-1)$$

Si tratta ora di mostrare che k non cresce all'infinito. A questo scopo torniamo alla considerazione dell'esistenza di un vettore $\mathbf{w}^*$ tale che $\mathbf{x} \cdot \mathbf{w}^* > 0$ se $\mathbf{x} \in F$. Ora consideriamo un valore scalare m tale che

$$m = \min_p\{\mathbf{x} \cdot \mathbf{w}^*\}$$

ovvero uguale al potenziale di attivazione minimo tra tutti i pattern P di addestramento calcolato usando il vettore desiderato $\mathbf{w}^*$. Consideriamo ora il seguente prodotto

$$\begin{aligned}\mathbf{w}(k) \cdot \mathbf{w}^* &= (\mathbf{w}(0) + \mathbf{x}(0) + \mathbf{x}(1) + \mathbf{x}(2) + \ldots + \mathbf{x}(k-1)) \cdot \mathbf{w}^*\\ &= \mathbf{w}(0) \cdot \mathbf{w}^* + \mathbf{x}(0) \cdot \mathbf{w}^* + \mathbf{x}(1) \cdot \mathbf{w}^* + \mathbf{x}(2) \cdot \mathbf{w}^* +\\ &+ \ldots + \mathbf{x}(k-1) \cdot \mathbf{w}^*\end{aligned}$$

e visto che $\mathbf{x}(p) \cdot \mathbf{w}^* \geq m$ per tutte le coppie di addestramento P

$$\mathbf{w}(k) \cdot \mathbf{w}^* \geq \mathbf{w}(0) \cdot \mathbf{w}^* + km$$

La diseguaglianza di Cauchy-Schwartz ci dice che per due vettori $\mathbf{a}$ e $\mathbf{b}$, qualsiasi essi siano,

$$(\mathbf{a} \cdot \mathbf{b})^2 \leq \|\mathbf{a}\|^2 \, \|\mathbf{b}\|^2$$

quindi

$$(\mathbf{w}(k) \cdot \mathbf{w}^*)^2 \leq \|\mathbf{w}(k)\|^2 \, \|\mathbf{w}^*\|^2$$

per cui con una semplice manipolazione algebrica si ottiene

$$\|\mathbf{w}(k)\|^2 \geq \frac{(\mathbf{w}(k) \cdot \mathbf{w}^*)^2}{\|\mathbf{w}^*\|^2}$$

e, sostituendo $\mathbf{w}(k) \cdot \mathbf{w}^*$ con il risultato precedente, abbiamo

$$\|\mathbf{w}(k)\|^2 \geq \frac{(\mathbf{w}(0) \cdot \mathbf{w}^* + km)^2}{\|\mathbf{w}^*\|^2}$$

Riconsideriamo ora il semplice fatto che i pesi sinaptici a un dato istante k sono dati da

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \mathbf{x}(k-1)$$

e che necessariamente, visto che la correzione avviene solo quando vi è un errore,

$$\mathbf{w}(k-1) \cdot \mathbf{x}(k-1) \leq 0$$

ora, combinando questi risultati abbiamo che

$$\begin{aligned}\|\mathbf{w}(k)\|^2 &= \|\mathbf{w}(k-1)\|^2 + \|\mathbf{x}(k-1)\|^2 + 2\mathbf{x}(k-1) \cdot \mathbf{w}(k-1) \\ &\leq \|\mathbf{w}(k-1)\|^2 + \|\mathbf{x}(k-1)\|^2\end{aligned}$$

Procedendo all'indietro otteniamo

$$\begin{aligned}\|\mathbf{w}(k)\|^2 &\leq \|\mathbf{w}(k-1)\|^2 + \|\mathbf{x}(k-1)\|^2 \\ &\leq \|\mathbf{w}(k-2)\|^2 + \|\mathbf{x}(k-2)\|^2 + \|\mathbf{x}(k-1)\|^2 \\ &\leq \|\mathbf{w}(k-3)\|^2 + \|\mathbf{x}(k-3)\|^2 + \|\mathbf{x}(k-2)\|^2, + \|\mathbf{x}(k-1)\|^2 \\ &\text{ecc.}\end{aligned}$$

Se ora consideriamo un valore scalare M tale che

$$M = \max_p (\|\mathbf{x}\|^2)$$

rappresentante la lunghezza quadratica più grande di tutti i vettori di addestramento P, possiamo riscrivere la diseguaglianza precedente semplicemente come

$$\|\mathbf{w}(k)\|^2 \leq \|\mathbf{w}(0)\|^2 + kM$$

Se ora combiniamo le due diseguaglianze

$$\|\mathbf{w}(k)\|^2 \geq \frac{(\mathbf{w}(0) \cdot \mathbf{w}^* + km)^2}{\|\mathbf{w}^*\|^2}$$

e

$$\|\mathbf{w}(k)\|^2 \leq \|\mathbf{w}(0)\|^2 + kM$$

otteniamo dei margini sui possibili cambiamenti del vettore di pesi sinaptici

$$\frac{(\mathbf{w}(0) \cdot \mathbf{w}^* + km)^2}{\|\mathbf{w}^*\|^2} \leq \|\mathbf{w}(k)\|^2 \leq \|\mathbf{w}(0)\|^2 + kM$$

Dato che il valore della limitazione inferiore cresce col quadrato di k, mentre quello della limitazione superiore cresce linearmente con k, questa doppia disuguaglianza non può essere soddisfatta per valori di k arbitrariamente grandi. Ne consegue che il numero di correzioni richieste per classificare correttamente i pattern di addestramento deve essere finito.

Per avere un'idea migliore del numero di iterazioni necessarie a raggiungere la soluzione, consideriamo la situazione in cui i pesi iniziali sono tutti 0 e in cui il vettore desiderato $\mathbf{w}^*$ possiede lunghezza 1

$$\|\mathbf{w}^*\| = 1$$

in tal caso, facendo le sostituzioni richieste, otteniamo

$$(km)^2 \leq kM$$

ricordandoci che ora il valore di m risulterà in generale diverso perché il vettore desiderato $\mathbf{w}^*$ possiede lunghezza 1. Quindi il limite per il numero massimo di modifiche sinaptiche k è dato da

$$k \leq \frac{M}{m^2}$$

In generale, visto che non è possibile calcolare m perché il vettore desiderato $\mathbf{w}^*$ è sconosciuto, questo risultato ci dice solamente che un percettrone raggiungerà in un numero finito di iterazioni una soluzione vettoriale se essa esiste, ma non ci permette di sapere in partenza il numero di iterazioni necessarie. Ricordiamo infine che per unità a gradino (con soglia o *bias* diverso da 0) la condizione di esistenza di una soluzione è data dalla separabilità lineare dei pattern d'ingresso.

3. La regola delta

La regola delta è simile alla regola di apprendimento dei percettroni considerata sopra, ma mentre per i percettroni semplici si utilizzano unità con attivazione a gradino, la regola delta è applicabile a unità di output dotate di una funzione di attivazione continua e differenziabile. L'impiego di unità continue e differenziabili presenta il vantaggio di permettere la descrizione delle prestazioni di una rete neurale tramite una funzione continua $E_{\mathbf{W}}$ che misura l'errore della rete; questa misura è differenziabile rispetto alla matrice di pesi sinaptici $\mathbf{W}$, ovvero è pos-

sibile descrivere il cambiamento dell'errore in funzione del cambiamento dei pesi sinaptici.

3.1. Unità lineari

Consideriamo ora una rete neurale di tipo feedforward con unità di output ad attivazione lineare,

$$y_i = \sum_j w_{ij} x_j$$

Dato un gruppo di pattern di addestramento composto da M coppie formate dal vettore di ingresso $\mathbf{x}^\mu$ e dal vettore di risposta desiderata $\mathbf{t}^\mu$, vogliamo trovare la matrice di pesi sinaptici tale per cui

$$y_i^\mu = t_i^\mu \quad \forall i, \mu$$

Possiamo descrivere le prestazioni generali della rete tramite la funzione di errore (anche detta «funzione di costo» perché desideriamo ridurla) E_W

$$E_W = \frac{1}{2} \sum_\mu \sum_i (t_i^\mu - y_i^\mu)^2$$

ovvero lo scarto quadratico medio tra la risposta desiderata e la risposta ottenuta per ciascuna unità della rete sommato su tutte le unità e su tutte le coppie di apprendimento (la frazione serve unicamente a semplificare alcuni calcoli matematici nelle derivazioni successive). Poiché stiamo impiegando unità lineari possiamo riscrivere la funzione di costo

$$E_W = \frac{1}{2} \sum_\mu \sum_i \left(t_i^\mu - \sum_j w_{ij} x_j^\mu \right)^2$$

La funzione di errore possiede valori tanto più piccoli quanto più le risposte della rete neurale si avvicinano alle risposte desiderate. Dall'equazione precedente risulta evidente che la variazione della funzione E_W dipende unicamente dal valore delle connessioni sinaptiche $\mathbf{W}$. Infatti il gradiente (e quindi la direzione di massima crescita) della funzione è individuato da un vettore di derivate parziali di E_W rispetto a ciascun peso sinaptico w_{ij}.

Siccome noi siamo interessati a ridurre il valore di E_W, ovvero a minimizzare lo scarto quadratico medio tra la risposta desiderata e la risposta ottenuta, dobbiamo modificare i pesi sinaptici nella direzione opposta al gradiente di E_W:

$$\Delta w_{ij} = -\frac{\partial E}{\partial w_{ij}}$$

Ricordandoci che ciascuna unità di output è indipendente dalle altre e risolvendo la derivata per ciascuna singola unità, otteniamo che

$$\Delta w_{ij} = \sum_{\mu} (t_i^{\mu} - y_i^{\mu}) x_j^{\mu}$$

quindi, il cambiamento dei pesi sinaptici per ciascuna coppia di apprendimento μ è dato dalla differenza tra la risposta desiderata e l'output della rete moltiplicata per l'attività presinaptica

$$\Delta w_{ij} = \eta(t_i - y_i)x_j$$

dove η è il solito tasso di apprendimento. L'apprendimento può avvenire sia per cicli che per epoche: nel primo caso la modifica viene calcolata e addizionata ai pesi sinaptici per ogni coppia di addestramento; nel secondo caso tutte le coppie vengono presentate alla rete, le modifiche vengono progressivamente addizionate e la somma totale viene applicata ai pesi sinaptici alla fine dell'epoca. Quando vengono impiegate unità ad attivazione lineare le due modalità di apprendimento sono equivalenti; in ogni caso, la regola delta richiede una serie di presentazioni ripetute dei pattern di addestramento (come nel caso della regola del percettrone).

Rispetto ai percettroni semplici, la regola delta è in grado di operare anche con funzioni di attivazione continue. La condizione per il raggiungimento di una soluzione con la regola delta applicata a unità continue è l'indipendenza lineare dei pattern d'ingresso. Questa condizione è sufficiente, ma non necessaria. In alcuni rari casi la regola delta può individuare connessioni sinaptiche anche per pattern linearmente dipendenti (purché siano linearmente separabili), ma il successo dipende spesso da come vengono formulati i vettori di risposta desiderata.

Siccome questo algoritmo è basato sulla differenza δ tra la risposta desiderata e la risposta ottenuta, esso è universalmente indicato con il nome di «regola delta»[2] e la formulazione matematica viene talvolta [*ibidem*] abbreviata in

$$\Delta w_{ij} = \eta \delta_i x_j$$

La regola delta viene talvolta definita «regola di Widrow-Hoff», dal nome degli autori [Widrow e Hoff 1960], «regola ADALINE» dall'inglese *Adaptive Linear Neuron*, oppure «regola LMS» dall'inglese *Least Mean Square* per sottolineare lo scarto quadratico medio impiegato nella funzione di costo.

Come abbiamo avuto modo di vedere, il funzionamento della regola delta consiste nella ricerca di pesi sinaptici che corrispondano a valori minimi della funzione di errore E_w. Possiamo intuitivamente

[2] Da non confondersi con il simbolo Δ che è sempre utilizzato per indicare la modifica sinaptica.

immaginare la funzione di errore come una superficie in uno spazio n-dimensionale (dove n è il numero di pesi sinaptici della rete)[3]; poiché ciascuna dimensione rappresenta lo spazio di possibile variazione di ciascuna sinapsi, la superficie presenta un aspetto variegato caratterizzato da «minimi e massimi». Nel caso in cui i pattern di input siano linearmente indipendenti, la funzione di costo E_w per unità lineari presenta una superficie a forma di scodella con un unico minimo dove $E_w = 0$. Se utilizziamo un tasso di apprendimento sufficientemente piccolo, la modifica graduale dei pesi sinaptici corrisponde a una discesa sulle pareti della superficie verso il fondo; tuttavia, siccome i componenti ortogonali dei pesi sinaptici (che sono inizializzati a valori casuali) non vengono modificati, non è possibile predire quale sarà la matrice esatta di connessioni sinaptiche finali che corrisponde alla soluzione [Hertz, Krogh e Palmer 1991]. Nel caso in cui i pattern di addestramento non siano linearmente indipendenti, la regola delta produce pur sempre una discesa sulla superficie della funzione di errore, ma il minimo raggiungibile può essere maggiore di zero; in altre parole, la rete neurale trova solamente una soluzione parziale per i pattern di addestramento (ad esempio, solamente alcune associazioni sono apprese correttamente oppure tutte le associazioni presentano qualche unità di output scorretta).

Il tasso di apprendimento η controlla il movimento sulla superficie dell'errore: se η è troppo grande si rischia di balzare da una parete all'altra senza raggiungere il minimo; al contrario, se η è troppo piccolo, la discesa verso il minimo è assicurata (solo nel caso di unità lineari), ma è estremamente lenta e richiede molte presentazioni delle coppie di addestramento. Alcuni autori hanno suggerito di scegliere il valore in base a $0{,}1 \leq N\eta \leq 1{,}0$, dove N è il numero di unità di input per ogni unità di output [Widrow, Winter e Baxter 1987]; altri hanno indicato margini ben più ampi, come $0{,}1 \leq \eta \leq 10{,}0$ [Hecht-Nielsen 1990].

Regola delta: algoritmo

1. Inizializzare i pesi sinaptici (*bias* incluso) a piccoli valori casuali:

$$w_{ij} = rnd(\pm 0{,}1) \quad (\text{per } j = \{0, 1, ..., N\})$$

fissare il tasso di apprendimento in modo tale che

$$0{,}1 \leq N\eta \leq 1{,}0 \quad (\text{ad esempio } 0{,}1)$$

2. Eseguire i passi 3-7 fino a quando la funzione di errore E_w raggiunge un valore minimo stabile (possibilmente 0,0 se il compito è risolvibile).

[3] Questa rappresentazione è utile, ma deve essere presa con cautela in casi di reti neurali più complesse (con più connessioni e funzioni di attivazione non-lineari).

3. Per ciascuna coppia di addestramento **s**, **t** (s_i, $t_i \in \Re$).
4. Stabilire i valori dei nodi d'ingresso:

$$x_i = s_i$$

5. Calcolare l'attivazione di ciascun nodo di uscita

$$y_i = \sum_{j=0}^{N} w_{ij} x_j$$

6. Calcolare la modifica dei pesi per ciascun nodo di uscita:

$$\Delta w_{ij} = \eta(t_i - y_i)x_j$$

7. Modificare i pesi:

$$w_{ij} = w_{ij}^{t-1} + \Delta w_{ij}$$

Nell'apprendimento per epoche, le modifiche calcolate al passo 6 per ciascuna coppia di pattern di addestramento vengono sommate e il passo 7 viene effettuato solo alla fine della presentazione di tutti i pattern (un'epoca) utilizzando la somma ottenuta.

3.2. Soluzione esatta

Nel caso specifico di unità lineari è possibile calcolare esattamente il valore dei pesi sinaptici in un solo passaggio utilizzando il «metodo della pseudo-inversione» [Hertz, Krogh e Palmer 1991]. I pesi sinaptici sono esplicitamente dati da

$$w_{ij} = \frac{1}{N} \sum_{\mu} \sum_{\nu} t_i^{\mu} (\mathbf{Q}^{-1})_{\mu\nu} x_j^{\nu}$$

dove

$$\mathbf{Q}_{\mu\nu} = \frac{1}{N} \sum_{j} x_j^{\mu} x_j^{\nu}$$

è la matrice di sovrapposizione dei pattern di input ed N è il numero di elementi che compongono ciascun pattern di input[4]. La matrice inversa $\mathbf{Q}^{-1}$ esiste solo se $\mathbf{Q}$ è non-singolare, ovvero se i pattern di input sono linearmente indipendenti. Invece, se i pattern sono linearmente dipendenti (ma linearmente separabili), il problema non è risolvibile con questo metodo, ma potrebbe essere risolvibile con il metodo di discesa del gradiente esposto sopra.

[4] Torneremo in maggior dettaglio su questo metodo nel prossimo capitolo.

3.3. Unità continue non-lineari

Se l'attivazione dei nodi di uscita è una funzione continua e differenziabile non-lineare[5] $y_i = \Phi(A_i)$ del potenziale di attivazione A_i, la funzione di costo diventa

$$E_W = \frac{1}{2}\sum_{\mu}\sum_{i}\left(t_i^{\mu} - \Phi\left(\sum_{j=0} w_{ij}x_j\right)\right)^2$$

La discesa del gradiente della funzione rispetto ai pesi sinaptici produce ora la seguente regola di modifica sinaptica

$$\Delta w_{ij} = -\frac{\partial E}{\partial w_{ij}} = \sum_{\mu}(t_i^{\mu} - y_i^{\mu})\dot{\Phi}(A_i^{\mu})x_j^{\mu}$$

quindi per ogni coppia di addestramento la modifica dei pesi sinaptici è

$$\Delta w_{ij} = \eta(t_i - y_i)\,\dot{\Phi}(A_i)x_j$$

dove $\dot{\Phi}(A_i)$ è la derivata prima della funzione di attivazione. Se la funzione di attivazione è una sigmoide tra 0 e 1, $y_i = \Phi(A_i) = (1 + e^{-\beta A_i})^{-1}$, non è necessario calcolare la derivata perché questa è direttamente esprimibile in base all'output dell'unità stessa $\dot{\Phi}(A_i) = \beta y_i(1 - y_i)$; similmente, se l'attivazione è data dalla tangente iperbolica (che fornisce valori nell'intervallo tra −1 e +1) $y_i = \Phi(A_i) = \tanh(A_i)$, la derivata prima è uguale a $\dot{\Phi}(A_i) = \beta(1 - y_i^2)$. Siccome in entrambi i casi le funzioni hanno due asintoti orizzontali e il loro valore cambia rapidamente per valori intermedi dell'argomento (intorno a $y = 0{,}5$ per la funzione sigmoide e intorno a $y = 0$ per la funzione tangente iperbolica), la regola delta modifica maggiormente i pesi sinaptici di unità che possiedono valori di attivazione intermedi (per cui la derivata prima è maggiore) e cambia di poco le connessioni di unità che emettono risposte prossime agli estremi (per cui la derivata è prossima a 0): in altre parole i pesi sinaptici di unità «indecise» vengono corretti maggiormente dei pesi sinaptici di unità decise.

L'impiego di unità continue non-lineari non modifica la condizione di raggiungimento di una soluzione, che consiste sempre nell'indipendenza lineare dei pattern d'ingresso, ma aumenta – rispetto al caso in cui si usano unità lineari – il numero di «soluzioni parziali» raggiungibili quando i pattern sono linearmente dipendenti [Hertz, Krogh e Palmer 1991]. Si noti anche che con l'impiego di funzioni asintotiche (come ad esempio la funzione sigmoide e la tangente iperbolica) la rete non è in grado di raggiungere il minimo assoluto $E_W = 0$ quando

[5] E non decrescente; questi tipi di funzioni sono anche definite «semi-lineari» [Rumelhart, McClelland e Gruppo PDP 1986; trad. it. 1991]: la funzione sigmoide è un esempio.

le risposte desiderate si collocano esse stesse agli estremi (0, 1 per la funzione sigmoide e −1, +1 per la funzione tangente iperbolica), ma si porta su valori molto prossimi a 0. La superficie della funzione di errore con funzioni continue non-lineari può assumere dunque andamenti più complessi della semplice forma a scodella osservata per unità continue lineari: essa può possedere diversi «minimi locali», ovvero configurazioni di pesi sinaptici che generano un errore relativamente piccolo, ma che non corrispondono sempre alla risposta desiderata.

Le unità continue non-lineari si rivelano particolarmente utili in reti multi-strato. Le reti multi-strato, al contrario delle reti con un solo strato di connessioni, non sono più soggette alle condizioni di indipendenza lineare e di separabilità lineare dei pattern di input. Queste reti possiedono uno o più strati di unità interne che rielaborano i pattern di input formando delle rappresentazioni interne che permettono alle unità di output di fornire la risposta esatta. Le reti multi-strato utilizzano nella maggior parte dei casi unità ad attivazione non-lineare.

Infatti, siccome una trasformazione lineare di una trasformazione lineare è riducibile a un'unica trasformazione lineare, una qualsiasi rete multi-strato che utilizzi unità lineari è analoga a una rete con un solo strato di sinapsi a unità lineari (fig. 2.1) ed è quindi soggetta alle stesse limitazioni di indipendenza lineare dei pattern.

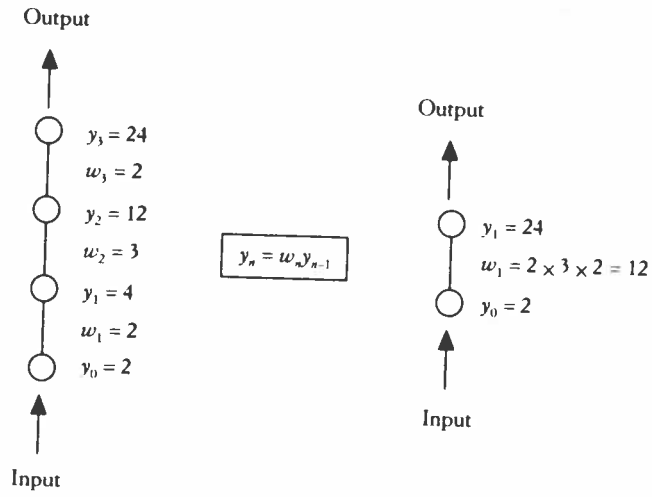


FIG. 2.1. Equivalenza tra una rete multi-strato con unità ad attivazione lineare e una rete con un singolo strato di sinapsi e unità lineari.

3.4. La regola Rescorla-Wagner

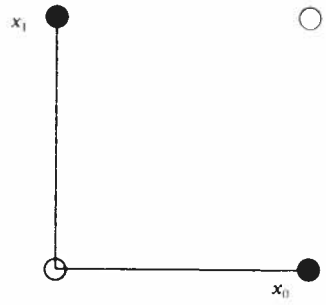
La regola delta è simile alla regola Rescorla-Wagner [1972] formulata indipendentemente in psicologia come modello di modifica delle associazioni all'interno della cornice del condizionamento classico (si veda Gluck [1991] per una trattazione dettagliata delle somiglianze). La regola di Rescorla-Wagner funziona nel seguente modo. Immaginiamo di addestrare un soggetto (animale o umano) a scegliere sempre gli oggetti rossi fra oggetti di diverso colore. Assumiamo che inizialmente la probabilità del soggetto di scegliere un oggetto rosso V_R sia prossima a 0,5 e più o meno uguale alla probabilità di scegliere un oggetto di qualsiasi altro colore V_X; il soggetto sceglierà quindi un oggetto a caso. Per ogni scelta, se l'oggetto è rosso premiamo il soggetto, altrimenti non facciamo nulla. Secondo la regola Rescorla-Wagner la modifica delle probabilità ΔV di scegliere un oggetto di un determinato colore è descrivibile da

$$\Delta V = \beta(\lambda - V)$$

dove $\lambda = 1$ se il colore è rosso (V_R) e premiamo la scelta, oppure $\lambda = 0$ se il colore non è rosso (V_X) e non facciamo nulla; la costante di apprendimento assume valori diversi a seconda del valore di λ. La regola Rescorla-Wagner è stata impiegata per modellare un gran numero di situazioni di apprendimento in psicologia comportamentale ed è stata utilizzata anche per modellare le capacità di inferenza transitiva che i piccioni sembrano possedere malgrado il fatto che Piaget sostenesse che l'inferenza transitiva richieda capacità logiche (Couvillon e Bittermann [1992]; si veda anche Floreano, De Lillo e Antinucci [1995] per un'analisi più dettagliata con una rete neurale).

4. Back-propagation

L'algoritmo di Back-propagation è un'estensione della regola delta per reti neurali multi-strato: per questo motivo esso viene spesso definito anche con il nome di «regola delta generalizzata». Le reti multi-strato con unità ad attivazione non-lineare sono in grado di classificare input che non sono linearmente separabili: esse infatti riescono a risolvere tutte le funzioni booleane, tra cui lo XOR (fig. 2.2). Lo XOR è una funzione molto semplice che viene spesso utilizzata come indice di prestazione di un algoritmo perché non è linearmente separabile e perché permette un'analisi trattabile delle soluzioni impiegate dalle reti neurali. Benché già negli anni Sessanta si sapesse del vantaggio offerto da reti multi-strato non-lineari, non esisteva ancora un algoritmo di apprendimento che fosse in grado di calcolare il contributo delle unità interne alla funzione di errore misurata sulle unità di output e non



x_0	x_1	t	
0	0	0	○
1	1	0	○
1	0	1	●
0	1	1	●

FIG. 2.2. La funzione booleana XOR non può essere appresa da un semplice percettrone perché non è possibile individuare una linea di separazione fra le due classi di risposta. I cerchietti neri indicano una risposta desiderata (t) 1, i cerchietti bianchi una risposta desiderata 0.

esisteva quindi un metodo per modificare le connessioni sinaptiche degli strati inferiori.

Il metodo di Back-propagation fornisce una soluzione a questo problema: esso è stato scoperto e riscoperto parecchie volte in modo indipendente e in diversi campi di ricerca [Bryson e Ho 1969; Le Cun 1985; Parker 1985; Werbos 1974], ma è stato denominato così nella riscoperta più recente di Rumelhart [Rumelhart, Hinton e Williams 1986], poi ulteriormente amplificata dall'inclusione dell'algoritmo nei due volumi PDP [McClelland, Rumelhart e Gruppo PDP 1986; Rumelhart, McClelland e Gruppo PDP 1986; trad. it. 1991], per sottolineare la procedura di propagazione all'indietro della misura di errore. Back-propagation rappresenta il cavallo di battaglia del connessionismo per la semplicità di funzionamento, la generalità di applicazione e la potenza del metodo: esso è stato applicato in un gran numero di campi, dalla modellizzazione cognitiva alle applicazioni ingegneristiche ed è al centro del rinnovato interesse per le reti neurali artificiali.

4.1. Il metodo di propagazione all'indietro dell'errore (Back-propagation)

Il metodo di Back-propagation è applicabile a reti neurali con un numero qualsiasi di strati di connessioni e con architetture molto diverse[6]. È formalmente simile alla regola delta considerata sopra in quanto modifica i pesi sinaptici in base alla discrepanza tra la risposta fornita dalla rete e la risposta corretta. Consideriamo per semplicità una rete feedforward a due strati di connessioni con unità di input x_k, unità interne («nascoste») h_j e unità di output y_i; indichiamo il valore della risposta corretta per ogni unità di output con il simbolo t_i^μ, dove

[6] L'esposizione del metodo data qui di seguito è analoga a quella offerta da Rumelhart, McClelland e Gruppo PDP [1986, cap. 8; trad. it. 1991] e da Hertz, Krogh e Palmer [1991].

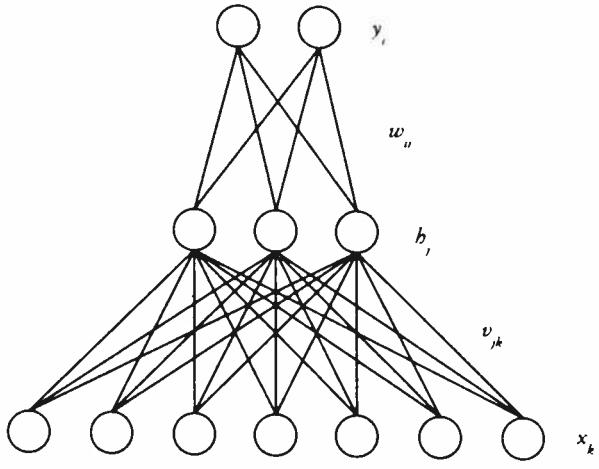


FIG. 2.3. Rete multistrato feedforward con unità non-lineari.

μ indica uno specifico pattern d'ingresso. Ciascuna unità di output riceve connessioni sinaptiche w_{ij} da tutte le unità nascoste e ciascuna unità nascosta riceve connessioni sinaptiche v_{jk} da tutte le unità di input (fig. 2.3). Tutte le unità (tranne le unità di input il cui valore corrisponde al pattern di ingresso) possiedono un'attivazione continua non-lineare, come ad esempio la funzione sigmoide $\Phi(A) = (1 + e^{-\beta A})^{-1}$ già descritta in precedenza. Le unità d'ingresso possono assumere valori binari, bipolari o continui.

Dato un pattern d'ingresso $\mathbf{x}^\mu$, si calcola innanzitutto l'attivazione di tutte le unità interne h_j

$$h_j^\mu = \Phi\left(\sum_{k=0} v_{jk} x_k^\mu \right)$$

e successivamente l'attivazione di tutte le unità di output y_i

$$y_i^\mu = \Phi\left(\sum_{j=0} w_{ij} h_j^\mu \right)$$

in entrambi i casi le unità possiedono una connessione (corrispondente all'indice 0) con un'unità di *bias* con valore costante –1.

Siccome desideriamo ridurre la differenza tra la risposta desiderata t_i^μ e la risposta ottenuta y_i^μ, la funzione di errore E_W è identica a quella già considerata per la regola delta

$$E_W = \frac{1}{2} \sum_\mu \sum_i (t_i^\mu - y_i^\mu)^2$$

che, se espansa, diventa ora

$$E_W = \frac{1}{2}\sum_{\mu}\sum_{i}\left[t_i^{\mu} - \Phi\left(\sum_{j} w_{ij}\Phi\left(\sum_{k} v_{jk}x_k^{\mu}\right)\right)\right]^2$$

La funzione E_W dipende da due gruppi di connessioni sinaptiche (w_{ij} e v_{jk}) e possiamo utilizzare il metodo di discesa del gradiente della funzione per ottenere la regola di modifica sinaptica. Per la matrice di connessioni w_{ij} tra le unità nascoste e le unità di output abbiamo lo stesso risultato già osservato per la regola delta

$$\Delta w_{ij} = -\eta\frac{\partial E}{\partial w_{ij}} = \eta\sum_{\mu}(t_i^{\mu} - y_i^{\mu})\dot{\Phi}(A_i^{\mu})h_j^{\mu}$$

che per semplicità riscriviamo come

$$\Delta w_{ij} = \eta\sum_{\mu}\delta_i^{\mu}h_j^{\mu} \quad \text{dove} \quad \delta_i^{\mu} = (t_i^{\mu} - y_i^{\mu})\dot{\Phi}(A_i^{\mu})$$

Per calcolare la modifica delle connessioni dello strato inferiore v_{jk} bisogna utilizzare la regola di derivazione delle funzioni composte; vi sono due metodi, uno in cui si considera la variazione dell'attivazione delle unità interne rispetto alla variazione dell'attivazione delle unità di input, e uno in cui si considera la variazione dell'attivazione delle unità interne rispetto alla variazione dei pesi sinaptici v_{jk} che trasmettono segnale da un input costante. Qui illustriamo il secondo metodo [si veda invece Rumelhart, McClelland e Gruppo PDP 1986, cap. 8, per l'altro metodo].

$$\Delta v_{jk} = -\eta\frac{\partial E}{\partial v_{jk}} = -\eta\sum_{\mu}\frac{\partial E}{\partial h_j^{\mu}}\frac{\partial h_j^{\mu}}{\partial v_{jk}}$$

che si sviluppa come segue

$$\Delta v_{jk} = \eta\sum_{\mu}\sum_{i}(t_i^{\mu} - y_i^{\mu})\dot{\Phi}(A_i^{\mu})w_{ij}\dot{\Phi}(A_j^{\mu})x_k^{\mu}$$

in cui possiamo individuare i delta δ_i^{μ} già calcolati per le unità di output e operare quindi la sostituzione

$$\Delta v_{jk} = \eta\sum_{\mu}\sum_{i}\delta_i^{\mu}w_{ij}\dot{\Phi}(A_j^{\mu})x_k^{\mu}$$

Si noti che i pesi sinaptici dello strato inferiore richiedono per ciascuna unità interna h_j la somma dei prodotti tra i delta δ_i^{μ} dello strato superiore di unità a cui l'unità nascosta è collegata e i corrispondenti pesi sinaptici w_{ij}. In altre parole gli errori (i delta) dello strato superiore vengono *propagati all'indietro* (da cui il nome Back-propagation) attraverso le stesse connessioni sinaptiche e sommati per ciascuna unità inferiore da cui ricevono segnale. Se stabiliamo che i delta per le unità nascoste δ_j^{μ} sono dati da

$$\delta_j^\mu = \dot{\Phi}(A_j^\mu)\sum_i w_{ij}\delta_i^\mu$$

la modifica dei pesi sinaptici v_{jk} assume la stessa forma della modifica dei pesi sinaptici w_{ij}

$$\Delta v_{jk} = \eta\sum_\mu \delta_j^\mu x_k^\mu$$

Siccome i delta di uno strato di unità sono ottenuti in base ai delta dello strato superiore, la regola di Back-propagation si applica a reti neurali con un numero qualsiasi di strati ciascuno composto da un qualsiasi numero di nodi. Riassumendo, il funzionamento dell'algoritmo consiste nel presentare i pattern di input e propagare l'attivazione strato per strato fino alle unità di uscita; quindi la modifica dei pesi sinaptici w_{mn} (dove m è l'indice delle unità dello strato superiore e n è l'indice delle unità dello strato inferiore) è data da

$$\Delta w_{mn} = \eta\sum_\mu \delta_m^\mu x_n^\mu$$

ove

$$\delta_m^\mu = (t_m^\mu - y_m^\mu)\dot{\Phi}(A_m^\mu)$$

se m corrisponde alle unità di uscita, oppure

$$\delta_m^\mu = \dot{\Phi}(A_m^\mu)\sum_s w_{sm}\delta_s^\mu$$

se m corrisponde alle unità interne; l'indice s indica le unità dello strato superiore rispetto a m. Back-propagation è un algoritmo molto potente e flessibile che è in grado di risolvere problemi non linearmente separabili: la semplicità di funzionamento dell'algoritmo e la possibilità di utilizzare reti con architetture di qualsiasi tipo hanno ulteriormente contribuito al suo successo sia in campo applicativo che per la modellistica all'interno delle scienze cognitive.

La regola di Back-propagation impiega un metodo di discesa del gradiente della funzione di errore come la regola delta. Tuttavia ora la funzione di errore E_W può assumere un andamento molto più complesso (fig. 2.4) dato dalla sequenza di operazioni non-lineari eseguite durante il calcolo dell'attivazione e spesso presenta molti minimi locali (soluzioni parziali), zone caratterizzate da ampie superfici piatte, improvvisi dirupi, valli molto strette e profonde, ecc. Questa descrizione geografica è più che altro metaforica perché non si applica direttamente a uno spazio con più di tre dimensioni, ma aiuta comunque a visualizzare il tipo di ostacoli che l'algoritmo deve superare quando si sposta alla ricerca di un minimo assoluto. I minimi locali rappresentano

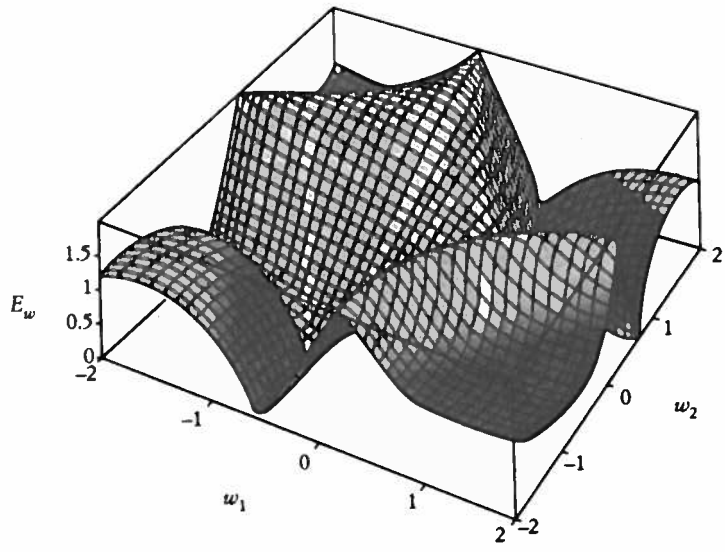


FIG. 2.4. Porzione di superficie della funzione di errore E_w per una rete multistrato con unità continue non-lineari.

degli ostacoli molto seri perché corrispondono a regioni in cui l'algoritmo può rimanere intrappolato senza poter raggiungere la soluzione. Anche le zone piatte (ad esempio, la funzione XOR possiede una superficie caratterizzata da grandi «altipiani») sono un pericolo per la convergenza perché in queste situazioni la derivata prima dell'attivazione del nodo postsinaptico è prossima a zero e i pesi sinaptici vengono quindi modificati pochissimo. A volte invece l'algoritmo ha difficoltà a trovare la soluzione perché questa corrisponde a una valle ripida e molto stretta: in questi casi la rete potrebbe «balzare» oltre l'apertura della valle e proseguire verso un'altra strada. Nonostante tutte queste difficoltà (e il fatto che non esista un prova analitica della convergenza dell'algoritmo per problemi che non siano linearmente separabili), Back-propagation è molto efficace nel raggiungere soluzioni soddisfacenti e generalizzare a nuovi esempi che non sono stati presentati nel gruppo di addestramento.

4.2. Varianti

Back-propagation è probabilmente l'algoritmo più utilizzato e studiato; per questo motivo sono state proposte diverse varianti per migliorarne le prestazioni. Alcune delle variazioni suggerite sono mirate soprattutto ad accelerare il processo di convergenza che nella versione di base descritta sopra è molto lento; altre varianti tendono invece a migliorare le caratteristiche di esplorazione della superficie dell'errore. In generale però l'efficacia di una determinata variante dipende dall'andamento della superficie dell'errore per il problema specifico che si vuole affrontare e nella maggior parte dei casi è difficile valutare a

priori quale sia la scelta migliore. Purtroppo non esistono delle regole sufficientemente generali e spesso è necessario fare diversi tentativi prima di individuare il metodo migliore.

La variante più comune consiste nell'apprendimento per cicli. La derivazione originale esposta nella sezione precedente indica che la modifica per epoche garantisce uno spostamento nella direzione inversa rispetto al gradiente della funzione di errore. In alcune situazioni questo metodo potrebbe però discendere il gradiente verso un minimo locale da cui non è più possibile uscire[7]. L'apprendimento per cicli modifica invece i pesi sinaptici subito dopo la presentazione di ciascun pattern: in questo caso l'algoritmo non esegue più strettamente la discesa del gradiente poiché utilizza solo informazioni relative a un singolo pattern. Benché per valori piccoli del tasso di apprendimento lo spostamento globale per cicli sia analogo allo spostamento per epoche, il «rumore» aggiunto dalla correzione istantanea dei pesi sinaptici potrebbe aiutare a uscire da minimi locali specialmente nelle prime fasi dell'apprendimento. Nella fase di implementazione al computer, l'apprendimento per cicli presenta anche il vantaggio di non richiedere locazioni di memoria aggiuntiva per immagazzinare la somma delle modifiche sinaptiche da applicare ai pesi alla fine di ogni epoca. Quando si utilizza il metodo di apprendimento per cicli bisogna però aver cura di presentare sempre i pattern del gruppo di addestramento in ordine casuale al fine di evitare la stagnazione in minimi locali generati da determinate sequenze di dati.

4.2.1. *Momentum*

Come nel caso della regola delta, anche per Back-propagation il valore del tasso di apprendimento η svolge un ruolo importante: un valore di η troppo piccolo comporta lunghi tempi di apprendimento, un valore troppo grande potrebbe risultare in larghe oscillazioni sulla superficie dell'errore. Un metodo per ovviare a questo problema consiste nel dotare lo spostamento sulla superficie dell'errore di una specie di «inerzia» che mantiene la direzione della discesa del gradiente. Questo effetto si ottiene aggiungendo al calcolo della modifica di ciascuna connessione una frazione della modifica ottenuta all'istante precedente [Plaut, Nowlan e Hinton 1986]

$$\Delta w_{ij}^{t} = \eta \delta_i x_j + \alpha \Delta w_{ij}^{t-1}$$

dove α è la costante «momentum» (tra 0 e 1) che controlla la «quantità di moto» fornita alla modifica sinaptica; tipicamente α viene fissa-

[7] In altri casi i pesi iniziali potrebbero essere tali per cui la somma algebrica delle variazioni calcolata su un sottogruppo di pattern potrebbe cancellare parzialmente la somma delle variazioni per il sottogruppo di pattern rimanenti. In questi casi i pesi sinaptici vengono modificati di poco alla fine di ogni epoca, ma l'errore per ciascun pattern resta comunque alto.

to intorno a 0,8. La presenza del momentum favorisce l'avanzamento su superfici piatte o semi-piatte e riduce le oscillazioni permettendo così l'adozione di valori più elevati per il tasso di apprendimento. Non vi è sufficiente evidenza per provare che il momentum giovi nel raggiungere il «minimo globale» della funzione (quello che rappresenta la soluzione al compito di apprendimento), ma molti autori riportano un notevole incremento della velocità di convergenza (fino a 10 volte per lo XOR [Fausett 1994]) rispetto alla versione di base dell'algoritmo. Il momentum può essere utilizzato sia nell'apprendimento per epoche che nell'apprendimento per cicli; in quest'ultimo caso non è garantito che esso corrisponda a un'inerzia nella direzione di discesa del gradiente, ma le eventuali deviazioni costituiscono una specie di rumore che può essere utile nella ricerca del minimo globale.

4.2.2. *Parametri adattivi*

I valori ottimali del tasso di apprendimento e del momentum variano a seconda della posizione in cui la rete neurale si trova sulla superficie dell'errore. Esistono alcuni metodi empirici per la variazione automatica di questi parametri durante l'apprendimento della rete. In generale l'idea consiste nell'aumentare il tasso di apprendimento se l'errore continua a diminuire per un certo numero di cicli di apprendimento S (perché potrebbe significare che la rete si sta dirigendo correttamente verso il minimo assoluto), e nel diminuirlo invece se l'errore non decresce o inverte tendenza entro S cicli (che potrebbe coincidere con uno spostamento nella direzione scorretta); in quest'ultimo caso il momentum potrebbe essere temporaneamente azzerato per evitare spinte inerziali scorrette.

Riassumendo,

$$\Delta\eta = \begin{cases} +b\eta & \text{se } \Delta E_W^s < 0 \quad \forall s \in S \\ -c\eta & \text{se } \Delta E_W^s \geq 0 \end{cases}$$

dove Δ indica la variazione della misura di errore tra due istanti successivi e s indica i cicli all'interno dell'intervallo S.

Alcuni autori [Cater 1987; DeRouin *et al.* 1991] hanno suggerito di variare il tasso di apprendimento per ciascun pattern di addestramento. Quando i dati provengono dal mondo reale succede spesso che alcune categorie importanti di pattern siano sotto-rappresentate (vengono viste meno frequentemente dalla rete neurale): l'impiego di un tasso di apprendimento più alto per questi pattern potrebbe contro-bilanciare la tendenza a convergere verso un minimo locale in cui le categorie sotto-rappresentate non vengono risolte adeguatamente.

Jacobs [1988] ha proposto invece un metodo in cui ciascuna connessione sinaptica possiede un proprio tasso di apprendimento che

varia in base al segno della derivata parziale della funzione di errore rispetto a quella determinata connessione. Benché la convergenza si ottenga solamente nel 90% dei casi, il suo metodo accelera l'apprendimento fino a quaranta volte nel caso dello XOR. Tuttavia la modifica automatica del tasso di apprendimento richiede la presenza di tre parametri aggiuntivi che devono essere fissati dallo sperimentatore.

Altri autori hanno utilizzato un tasso di apprendimento inversamente proporzionale al numero di connessioni per ciascun nodo, ma senza variarne il valore durante l'apprendimento [Plaut, Nowlan e Hinton 1986; Tesauro e Janssens 1988].

Anche il parametro β che controlla l'andamento della funzione sigmoide svolge una funzione importante perché la sua variazione può modificare notevolmente la superficie della funzione dell'errore. Quanto maggiore è β, tanto più velocemente l'output dell'unità si porta agli estremi (fig. 2.5) e, per $\beta \to \infty$, la funzione sigmoide è equivalente alla funzione a gradino.

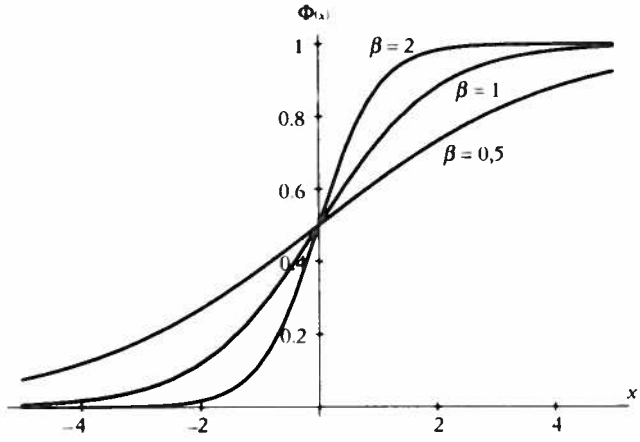


FIG. 2.5. La funzione sigmoide per diversi valori della costante β.

Solitamente $\beta = 1$, ma è possibile assegnare valori iniziali diversi per ciascun nodo e farlo variare come un qualsiasi altro peso sinaptico della rete calcolando la derivata parziale della funzione di errore rispetto ad esso con il metodo di discesa del gradiente [Tepedelenliogu *et al.* 1991]. Innanzitutto consideriamo che la derivata prima della funzione sigmoide è

$$\dot{\Phi}(x) = \beta\Phi(x)(1 - \Phi(x))$$

e quindi la modifica dei pesi sinaptici diventa

$$\Delta w_{mn} = \eta \sum_{\mu} \beta_m \delta_m^{\mu} x_n^{\mu}$$

Dal calcolo della derivata parziale rispetto ai β_m, otteniamo che la modifica di questo parametro è data da

$$\Delta\beta_m = \eta \sum_{\mu} \delta_m^{\mu} A_m^{\mu}$$

dove A_m^{μ} è il potenziale di attivazione del nodo m per il pattern μ.

4.2.3. *Pesi sinaptici iniziali*

I valori iniziali dei pesi sinaptici sono importanti in quanto determinano il punto di partenza della rete neurale sulla superficie dell'errore. Se sono troppo grandi la funzione di attivazione sigmoide o tangente iperbolica potrebbe venir saturata (produrre valori che si collocano agli estremi) fin dall'inizio e, siccome la derivata prima sarebbe prossima a zero, la rete potrebbe apprendere molto lentamente oppure restare bloccata in un minimo locale[8]. Inoltre, se fossero tutti uguali e la trasformazione da apprendere richiedesse pesi diversi, la rete neurale non potrebbe apprendere il compito perché la correzione sinaptica viene fatta in proporzione al valore di ciascun peso e risulterebbe impossibile «turbare la simmetria» [Reed e Marks 1999]. Per evitare queste situazioni, i pesi sinaptici vengono solitamente inizializzati a piccoli valori casuali entro un piccolo campo intorno allo zero (ad esempio tra −0,5 e +0,5). Un metodo analogo, ma più generale, consiste nel limitare la variazione casuale iniziale entro $\pm 1/\sqrt{k_i}$, dove k_i è il numero di connessioni afferenti al nodo i-esimo. Questa procedura stabilisce limiti diversi per i vari nodi della rete e assicura che la somma pesata degli input (input netto o potenziale di attivazione) per ciascun nodo sia intorno a 0, un valore per cui la funzione sigmoide e tangente iperbolica producono valori vicini al punto di massima rapidità di variazione (0,5 e 0 rispettivamente) generando quindi modifiche sinaptiche rilevanti.

Nguyen e Widrow [1990] hanno utilizzato una procedura simile che mantiene le attivazioni iniziali entro la zona di massima rapidità di variazione per le unità interne. I pesi dell'intera rete vengono inizializzati a valori casuali intorno allo 0 (ad esempio tra −0,5 e +0,5); solamente i pesi dalle unità di input alle unità interne v_{jk} vengono quindi ricalcolati in base a un processo di normalizzazione

$$v_{jk} = \frac{\beta v_{jk}}{\| \mathbf{v}_j \|} \left\{ \beta = 0{,}7\, \sqrt[n]{p} \right\}$$

[8] Uno studio sistematico dell'apprendimento della funzione XOR, a partire da pesi iniziali diversi, ha rivelato una sensibilità ai valori iniziali molto maggiore di quanto non si fosse sospettato in precedenza [Kolen e Pollack 1990]. Siccome non è possibile stimare la generalità di questi risultati né esiste una prova della convergenza delle reti multistrato non-lineari, resta buona norma ripetere una simulazione parecchie volte con diverse inizializzazioni casuali dei pesi sinaptici e riportare dati statistici.

dove n è il numero di unità di input e p è il numero di unità interne. Benché questa procedura fosse stata ideata per unità continue bipolari, risultati sperimentali indicano che essa migliora la percentuale di successi per lo XOR e dimezza i tempi di convergenza anche con unità continue binarie come la funzione sigmoide.

4.2.4. *Perturbazione della derivata*

Benché non esistano studi esaustivi sulle proprietà delle superfici dell'errore, molti risultati sperimentali indicano che le zone piatte rappresentano una situazione molto comune. Una zona piatta consiste in un ampio campo di variazione dei pesi sinaptici per cui la funzione di errore possiede lo stesso valore. Questa situazione si verifica tipicamente quando i pesi sinaptici crescono saturando la funzione di attivazione: quando questo avviene i nodi emettono lo stesso output (ad esempio 0 o 1 per la funzione sigmoide) per un'ampia gamma di valori di input. Una volta che la funzione è saturata non solo sono richieste grandi e ripetute modifiche sinaptiche, ma la derivata della funzione stessa genera valori prossimi o uguali a zero e di conseguenza i termini delta della regola sinaptica sono nulli. Per ovviare a questo problema Fahlman [1989] ha proposto di aggiungere una piccola costante k alla derivata dell'attivazione dei nodi di output nel calcolo dei termini delta

$$\delta_i^\mu = (\dot{\Phi}(A_i^\mu) + k)(t_i^\mu - y_i^\mu)$$

La presenza di k (0,1 negli esperimenti di Fahlman) permette la modifica della connessione sinaptica anche quando la derivata prima dell'attivazione è zero e, unita a un momentum abbastanza alto, genera un «lungo scivolamento» sulla zona piatta della superficie dell'errore. Fino a quando la rete individua nuovamente un punto di discesa.

4.2.5. *Addizione di rumore*

Un modo semplice per cercare di uscire dai minimi locali consiste nell'applicare un po' di «rumore» durante l'apprendimento della rete. Abbiamo già osservato in precedenza che l'apprendimento per cicli introduce in effetti del rumore nella discesa del gradiente. In alternativa (o in combinazione) è possibile variare lievemente i valori dei pesi sinaptici aggiungendo dei piccoli valori estratti da una distribuzione uniforme di numeri casuali intorno allo zero [von Lehman *et al.* 1988] oppure aggiungendo questi piccoli valori casuali all'attivazione dei nodi di input [Sietsma e Dow 1988].

La quantità di rumore iniettato può variare in funzione dell'andamento della misura di errore, in modo analogo alla variazione automa-

tica del tasso di apprendimento: se l'errore continua a calare per un certo numero di cicli l'intensità del rumore viene subito diminuita, altrimenti viene gradualmente aumentata. Questo metodo potrebbe rivelarsi particolarmente utile perché gli studi sperimentali sopra citati hanno mostrato che dosi inadeguate di rumore possono facilmente rallentare o impedire la convergenza della rete neurale.

Ulteriori varianti all'algoritmo di Back-propagation volte in particolare ad accelerare la convergenza del metodo sono descritte in Reed e Marks [1999].

Back-propagation: algoritmo

La descrizione dell'algoritmo data qui di seguito si riferisce alla versione base con la sola aggiunta del momentum e apprendimento per cicli (che sono le variazioni più comunemente adottate). Data una rete neurale con due strati di connessioni v_{jk} e w_{ij}, x_k unità di input, h_j unità interne e y_i unità di output (si veda descrizione al paragrafo 4.1 e fig 2.3) con attivazione binaria sigmoide

$$\Phi(A) = (1 + e^{-\beta A})^{-1} \quad \text{dove } \beta = 1$$

1. Inizializzare i pesi sinaptici (*bias* incluso) a piccoli valori casuali:

$$w_{ij} = rnd(\pm 0{,}1)$$
$$v_{jk} = rnd(\pm 0{,}1)$$

e fissare il tasso di apprendimento η e il momentum α in modo tale che

$$0{,}1 \leq \eta \leq 1{,}0$$
$$0{,}1 \leq \alpha \leq 1{,}0$$

Azzerare i valori Δw_{ij}^{t-1} e Δv_{jk}^{t-1} da utilizzare per il calcolo del momentum.

2. Eseguire i passi 3-9 fino a quando la funzione di errore E_W raggiunge un livello minimo («criterio») stabilito, oppure cessa di diminuire:

3. per ciascuna coppia **s**, **t** ($s_k, t_i \in \Re$) di input e di risposta desiderata estratta in modo casuale dal gruppo di pattern di addestramento.

4. Stabilire i valori dei nodi d'ingresso:

$$x_k = s_k$$

5. Calcolare l'attivazione di ciascun nodo interno

$$h_j = \Phi\left(\sum_{k=0} v_{jk} x_k \right)$$

e quindi l'attivazione di ciascun nodo di uscita

$$y_i = \Phi\left(\sum_{j=0} w_{ij} h_j \right)$$

6. Calcolare i delta per i nodi di uscita

$$\delta_i = (t_i - y_i)\dot{\Phi}(A_i)$$

7. Calcolare i delta per i nodi interni propagando all'indietro i delta dei nodi dello strato superiore

$$\delta_j = \dot{\Phi}(A_j)\sum_i w_{ij}\delta_i$$

8. Calcolare le modifiche sinaptiche di entrambi gli strati di connessione per il pattern di addestramento attuale

$$\Delta w_{ij}^t = \eta\delta_i h_j + \alpha\Delta w_{ij}^{t-1}$$

e

$$\Delta v_{jk}^t = \eta\delta_j x_k + \alpha\Delta v_{jk}^{t-1}$$

9. Applicare le modifiche così calcolate ai rispettivi pesi sinaptici

$$w_{ij} = w_{ij}^{t-1} + \Delta w_{ij}$$

e

$$v_{jk} = v_{jk}^{t-1} + \Delta v_{jk}$$

Se vi sono più di due strati di connessioni, bisogna ripetere più volte il passo 7 propagando all'indietro ogni volta i delta calcolati per lo strato immediatamente superiore. La versione di base della regola descritta nel paragrafo 4.1 si ottiene fissando il momentum a 0 e sommando tutte le modifiche calcolate al passo 8 fino a quando tutti i pattern di addestramento sono stati presentati prima di eseguire il passo 9.

4.3. Generalizzazione

La generalizzazione consiste nella produzione di una risposta appropriata per un pattern di input che non era stato incluso nel gruppo di addestramento. Il problema di una generalizzazione adeguata si presenta quando possediamo solo un numero limitato di pattern di addestramento ma vogliamo garantirci un buon livello di prestazioni quando, completato l'apprendimento, si possono presentare dei nuovi pattern (situazione che si verifica comunemente nelle applicazioni reali). Nel caso di una rete con un unico strato di connessioni il meccanismo di generalizzazione è facilmente visualizzabile: la risposta della rete dipenderà dalla porzione dello spazio dell'input in cui si colloca il nuovo pattern rispetto alla linea di separazione individuata dai pesi sinaptici durante la fase di addestramento. Lo stesso principio si applica

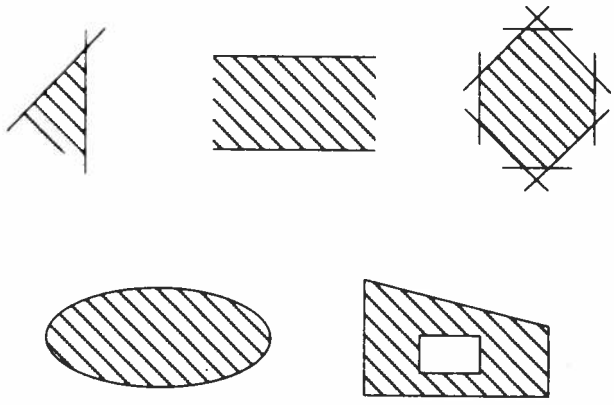


FIG. 2.6. Separazione dello spazio dell'input con reti multistrato; la combinazione delle linee di separazione generate dai nodi interni produce regioni che possono essere non connesse e non convesse.

anche alle reti multistrato con unità non-lineari, ma qui le cose sono più complesse perché la combinazione degli iperpiani individuati dalle singole unità interne fa sì che lo spazio dell'input venga suddiviso in regioni dai contorni irregolari che possono essere non connesse e non convesse (fig. 2.6).

Il numero di regioni e la loro forma dipende dal numero di pesi sinaptici e di unità interne e dalla durata dell'addestramento: se una rete neurale possiede un gran numero di unità nascoste lo spazio dell'input potrebbe essere suddiviso in un numero di regioni tali per cui ciascuna di esse racchiude al limite un unico pattern di addestramento («overfitting»)[9]. In questo caso anche una piccola distorsione potrebbe spostare il pattern al di fuori della propria regione e causare una risposta scorretta. In altre parole, un gran numero di risorse (pesi e unità) permette alla rete di apprendere una vasta serie di funzioni specifiche che sono responsabili della corrispondenza esatta tra input e output per i pattern di addestramento, diminuendo così la probabilità che la rete scopra proprio la funzione generale che descrive l'intero dominio del problema e che darebbe luogo a risposte corrette anche per i pattern di test. Il problema è analogo all'approssimazione di dati in statistica: se il numero di parametri liberi (in questo caso i pesi sinaptici della rete) da stimare è troppo alto, si ottiene una descrizione esatta dei dati ma si rischiano errori per nuovi dati (fig. 2.7). Un numero troppo grande di connessioni compromette anche la convergenza verso un minimo globale, sia perché aumenta la complessità della superficie dell'errore, sia perché richiede un alto numero di pattern di

[9] Al contrario, un piccolo numero di pesi e unità interne potrebbe non bastare a separare gli input nelle classi di risposta richieste.

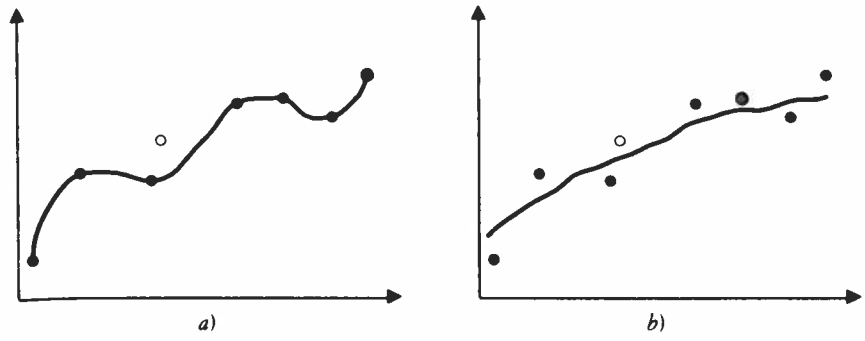


FIG. 2.7. Apprendimento della funzione che descrive i dati di addestramento (cerchietti neri). L'errore è lo scarto quadratico medio tra un dato e il corrispondente punto della funzione. *a*) Un numero troppo grande di parametri (unità interne e pesi sinaptici) non garantisce una buona generalizzazione al nuovo pattern (cerchietto bianco); *b*) un numero adeguato di parametri permette una migliore generalizzazione (lo scarto quadratico medio è inferiore).

addestramento [Baum e Haussler 1989]. La scelta del numero adeguato di unità interne e di pesi sinaptici è quindi un problema cruciale e molto complesso che verrà ripreso da diversi punti di vista nelle sezioni seguenti.

Un modo diverso per limitare il problema della scarsa capacità di generalizzazione (senza affrontare il problema dell'architettura della rete) consiste nell'arrestare l'addestramento prima che la rete apprenda «troppo bene» i pattern di addestramento. In questo modo si tenta di riprodurre una situazione analoga a quella illustrata in figura 2.7b, in cui la rete, benché non raggiunga un minimo globale per i pattern di addestramento, è comunque in grado di ottenere un livello accettabile di prestazioni sia sui pattern di addestramento che su nuovi pattern di generalizzazione. La scelta del momento esatto in cui arrestare l'addestramento richiede lo stesso tipo di considerazioni impiegate per la scelta delle dimensioni dell'architettura [Wang, Vengatesh e Judd 1993], ma esistono anche delle semplici strategie empiriche che aiutano a prendere questa decisione. Il metodo più utilizzato consiste nel ripartire la collezione dei pattern disponibili in tre parti (non necessariamente equinumerose): una parte viene utilizzata per l'apprendimento, una per la «validazione» dell'apprendimento e un'altra per valutare le capacità di generalizzazione (test). Mentre la modifica dei pesi sinaptici si basa sull'errore calcolato per i pattern di addestramento, la decisione su quando fermare l'apprendimento si basa sull'errore ottenuto per i pattern di validazione. Assumendo che l'intero gruppo di pattern (addestramento, validazione e test) sia estratto in modo uniforme dalla stessa distribuzione di dati, l'errore sui pattern di validazione e di test inizialmente si riduce in modo simile all'errore sui pattern di addestramento, ma ad un certo punto l'errore sui pattern di validazio-

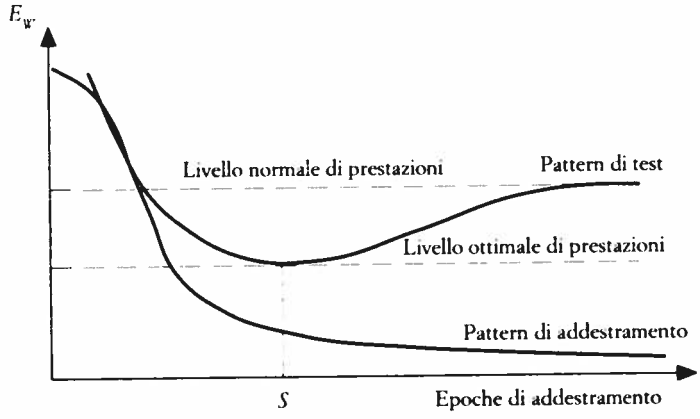


FIG. 2.8. Relazione tra errore sui pattern di addestramento ed errore sui pattern di test (+ pattern di validazione); il livello ottimale di prestazioni si ottiene se l'apprendimento viene fermato prima che si verifichi un overfit dei dati di addestramento (punto *S*) [Hecht-Nielsen 1990].

ne e di test comincia a crescere mentre l'errore sui pattern di addestramento continua a diminuire (fig. 2.8) [Weigend, Rumelhart e Huberman 1990]. Questa inversione di tendenza indica l'inizio del «sovrapprendimento» (overfit) dei dati di addestramento e costituisce un buon indicatore del momento in cui fermare l'addestramento. La decisione si basa sulla registrazione dell'errore per il gruppo di validazione nelle ultime *N* epoche di addestramento: se esso non diminuisce per tutte le *N* epoche, si cessa l'addestramento e si mantengono i pesi sinaptici corrispondenti all'errore minore nelle *N* epoche. La scelta di *N* (ad esempio 50) dipende dalle caratteristiche del compito. Alcuni autori suggeriscono anche di utilizzare la somma degli errori sul gruppo di addestramento e sul gruppo di validazione per valutare quando cessare l'addestramento [Hecht-Nielsen 1990].

La scelta del gruppo di validazione è quindi importante quanto la scelta dei pattern di addestramento. Quando il numero di pattern inizialmente disponibili è però scarso o poco ridondante, è possibile creare un «falso» gruppo di validazione duplicando i dati originali e aggiungendovi del rumore casuale o producendo delle distorsioni specifiche. Se anche questo metodo non è applicabile, conviene comunque fermare l'addestramento non appena la curva che descrive la discesa dell'errore sui pattern di addestramento cessa di scendere rapidamente.

4.4. Le capacità di rappresentazione delle reti multistrato

Nel 1957 il matematico russo Andrei Kolmogorov pubblicò un teorema nel quale egli mostrava che qualsiasi funzione continua di *N*

variabili definite sull'intervallo n-dimensionale $[0, 1]^n$ poteva essere rappresentata come una composizione di funzioni di una singola variabile[10]. Il risultato del teorema è esprimibile come

$$\Phi(x_1,\ldots,x_n) = \sum_{p=1}^{2N+1} \phi_p\left(\sum_{q=1}^{N} \psi_{pq}(x_q)\right)$$

dove $\phi(x)$ e $\psi(x)$ sono ciascuna una funzione monotona crescente di una singola variabile. Nel 1987 Hecht-Nielsen [1987a] ne fece notare la rilevanza nel campo delle reti neurali elaborando su di esso un ulteriore teorema con il quale si mostrava che una rete neurale con uno strato di unità interne sarebbe stata in grado di rappresentare qualsiasi funzione del suo input. Benché questo risultato non costituisse una prova che qualche algoritmo di addestramento fosse effettivamente in grado di apprendere qualsiasi funzione, né fornisse alcun suggerimento sul numero necessario di unità nascoste da utilizzare, esso indicava le potenzialità del nuovo approccio neurocomputazionale. Successivamente alcuni autori criticarono la validità di questa applicazione del teorema di Kolmogorov, soprattutto perché le funzioni $\phi(x)$ e $\psi(x)$ nel teorema originale non erano continue e differenziabili, ma piuttosto frattali [Girosi e Poggio 1989]; in seguito però la validità dell'applicazione del teorema è stata ristabilita sotto forma di approssimazione e impiegata per valutare il numero di unità interne necessarie [Kurkova 1991], anche se solo in via teorica. Risultati teorici simili sono stati raggiunti anche da altri autori [Hornik, Stinchcombe, White 1989] che hanno dimostrato che una rete neurale con un unico strato di unità interne non-lineari è in grado di apprendere qualsiasi funzione continua a un qualsiasi grado di approssimazione.

4.5. Architetture variabili

La scelta di un numero appropriato di unità interne e della quantità di connessioni necessarie per ciascuna unità sono una decisione spesso cruciale, come abbiamo notato in precedenza. Da un lato un numero insufficiente di unità interne compromette il raggiungimento di una soluzione; dall'altro lato un numero troppo grande potrebbe rallentare notevolemente i tempi di raggiungimento di un minimo globale e limitare comunque le capacità di generalizzazione. Benché la confidenza ispirata dal teorema di Kolmogorov possa spingerci ad aumentare il numero di unità interne quando una rete non riesce ad apprendere un compito, l'impiego di troppe connessioni aumenta il rischio che l'algoritmo di Back-propagation finisca in un minimo locale.

[10] Questa dimostrazione rappresentava la risposta a una sfida lanciata da Hilbert. Nel 1900 Hilbert propose 23 problemi che i matematici del ventesimo secolo avrebbero dovuto risolvere. Il teorema di Kolmogorov rappresentava la soluzione del tredicesimo problema [Hecht-Nielsen 1990; Gray 2000].

Una pratica abbastanza comune – ma non efficiente – consiste nel provare ad addestrare reti con un diverso numero di unità interne selezionando l'architettura che fornisce una prestazione soddisfacente. In alternativa, sono stati sviluppati vari algoritmi in cui l'architettura della rete neurale si modifica autonomamente durante il processo di apprendimento. Lo scopo di questi algoritmi è quello di mantenere la complessità della rete neurale al minimo necessario per risolvere il compito e assicurare una buona generalizzazione. Essi sono raggruppabili in due categorie principali (anche se vi sono delle sovrapposizioni): algoritmi che operano sulle connessioni e algoritmi che operano sulle unità interne della rete. Un altro modo per selezionare l'architettura più adatta consiste nell'impiego degli algoritmi genetici che verranno descritti nel sesto capitolo.

4.5.1. *Riduzione delle connessioni*

Un metodo per ridurre la complessità di una rete neurale consiste nell'eguagliare alcuni dei pesi sinaptici fra di loro: in questo modo il numero delle connessioni «indipendenti» che debbono essere calcolate è inferiore al numero delle connessioni effettivamente presenti nella rete. Questa tecnica viene definita come «condivisione dei pesi sinaptici» (*weight sharing*) [Lang, Waibel e Hinton 1990] e si rivela particolarmente efficiente quando si conoscono alcune proprietà del problema da affrontare [Le Cun *et al.* 1990]. Come vedremo più sotto, nel caso di analisi di immagini è vantaggioso impiegare una serie di unità interne che ricevono input solamente da una zona ristretta (e diversa per ogni unità) dell'intera superficie, ciascuna delle quali possiede gli stessi pesi sinaptici; questo vincolo permette di addestrare una serie di unità a rilevare determinate caratteristiche locali dell'input visivo semplicemente calcolando le modifiche sinaptiche per una sola di esse. Nowlan e Hinton [1992] hanno proposto di utilizzare una tecnica simile (ma che non richiede conoscenze sulle proprietà del compito da affrontare) in base alla quale tutti i pesi sinaptici della rete neurale sono descrivibili da un gruppo di distribuzioni gaussiane con media e ampiezza diverse. Aggiungendo questo vincolo alla funzione di errore e applicando il metodo di discesa del gradiente si ottiene una regola di apprendimento che tende a raggruppare i pesi sinaptici entro un piccolo numero di distribuzioni gaussiane (che corrispondono a connessioni vicine con valori simili), le cui medie e varianze si adattano automaticamente alla struttura del compito da risolvere.

Un altro metodo consiste nell'addestrare reti con un gran numero di connessioni ed eliminare quelle inutili nel corso dell'addestramento. Le Cun, Denker e Solla [1990] hanno suggerito di misurare la rilevanza di una connessione come il prodotto tra il quadrato del valore della connessione e la derivata seconda della funzione di errore rispetto alla connessione stessa; in questo modo le connessioni che non sono re-

sponsabili di grandi cambiamenti nell'errore totale vengono gradualmente eliminate durante il processo di addestramento. Altri autori hanno utilizzato un metodo simile per eliminare intere unità (e relative connessioni) che non contribuiscono molto alle prestazioni della rete [Mozer e Smolensky 1989; Sietsma e Dow 1988]. Thodberg [1991] ha applicato il principio del «rasoio di Ockam»[11] per ridurre la connettività di una rete neurale in modo tale da assicurare le migliori prestazioni possibili nella fase di generalizzazione. Nel suo approccio la metafora di Ockam si traduce nell'addestrare un gruppo di reti neurali – ciascuna sufficientemente grande da apprendere i pattern di addestramento correttamente – e nell'eliminare gradualmente le connessioni continuando a riaddestrare le reti per mantenere un livello accettabile di prestazioni; la rete neurale risultante con il minor numero di connessioni sarà quella che ottiene il miglior livello di generalizzazione. Thodberg ha dimostrato numericamente la validità di questo principio applicando la sua procedura al problema della contiguità[12]. Il suo algoritmo funziona nel seguente modo.

Rasoio di Ockam. 1. Separare le coppie di pattern in un gruppo di addestramento e in un gruppo di test. Scegliere un criterio di «errore accettabile» per i pattern di addestramento, ovvero il valore della funzione di errore E_W per cui l'output della rete è sufficiente per approssimare senza incertezza la risposta corretta.

2. Addestrare un gruppo di reti neurali con sufficienti unità da assicurare l'apprendimento dei pattern fino a quando esse raggiungono il criterio di errore, eliminando dal gruppo le reti che non ci riescono.

3. Per ciascuna delle restanti reti, eseguire ripetutamente (2 o 3 volte) il seguente «passaggio del rasoio»: stabilire un numero di «epoche di riaddestramento»; quindi, provare ad eliminare ciascun peso sinaptico sistematicamente (uno alla volta) riaddestrando la rete per il numero di epoche di riaddestramento. Se l'errore non aumenta oltre il criterio di errore accettabile (aumentato di un piccolo «margine di errore»), eliminare del tutto il peso, altrimenti reinserirlo e cancellare le modifiche apportate dal riaddestramento per quel peso.

4. Fra tutte le reti finali che possiedono una prestazione entro il

[11] Guglielmo di Ockam (circa 1285-1349) era un filosofo spesso ricordato per il suo principio di economia nella spiegazione, anche detto il principio del «rasoio», che può essere riformulato in termini ingegneristici dicendo che «non bisogna impiegare un gran numero di risorse per realizzare una cosa che ne richiede meno».

[12] Il problema della contiguità consiste nell'individuare il numero di blocchi di 1 nel vettore di input, dove un blocco è rappresentato da uno o più 1 consecutivi separati da almeno uno 0 dal blocco successivo (ad esempio il vettore 111000110 corrisponde a due blocchi). Malgrado esista una semplice soluzione (ciascun nodo interno dovrebbe specializzarsi nell'osservare solo due input contigui e segnalare alle unità di output la presenza della coppia 01), l'algoritmo di Back-propagation fatica a trovare la soluzione con reti pienamente connesse [Solla 1989].

criterio di errore accettabile, quelle con il minor numero di connessioni saranno le reti che forniranno la migliore generalizzazione ai pattern di test.

Ciascun passaggio richiede l'aumento del numero di epoche di riaddestramento (ad esempio 15, 60, 120) e il margine di errore è una piccola frazione dell'errore accettabile (ad esempio il 5%). Le connessioni di *bias* non vengono passate sotto la lama del rasoio perché non servono a elaborare direttamente informazione. Malgrado il fatto che una rete richieda un alto numero di epoche di addestramento per ogni passaggio (almeno 15 × 100 al primo passaggio in una rete con 100 connessioni iniziali), i risultati finali sono interessanti sia in termini di generalizzazione (corretta al 100% per il compito di contiguità studiato da Thodberg) che per le architetture risultanti.

Un approccio più semplice consiste nell'aggiungere a ciascun peso sinaptico una tendenza a decrescere verso zero con una velocità proporzionale alla forza assoluta del peso sinaptico [Hinton 1986; Kramer e Sangiovanni-Vincentelli 1989; Plaut, Nowlan e Hinton 1986; Scalettar e Zee 1988] dopo ogni epoca di apprendimento secondo la relazione

$$w_{ij}^{t} = (1 - \varepsilon) w_{ij}^{t-1}$$

dove ε è una costante che può assumere valori compresi tra 0 e 1. In questo modo i pesi sinaptici che non vengono rinforzati dall'apprendimento tendono gradualmente a scomparire. Questo procedimento equivale a modificare la funzione di errore E_W con l'aggiunta di un ulteriore termine che penalizza la presenza di pesi sinaptici

$$E = E_W + \gamma \sum_{i} \sum_{j} w_{ij}^2$$

dove γ è una costante che pesa l'importanza dell'eliminazione delle connessioni. Con il metodo di discesa del gradiente della nuova funzione di errore, otteniamo che

$$\varepsilon = 2\eta\gamma$$

e, poiché ε deve essere compresa tra 0 e 1, ricaviamo i possibili valori del parametro γ. Il procedimento di «decadimento delle connessioni» (*weight decay*) genera una matrice di connessioni i cui valori assoluti sono proporzionali alla derivata dell'errore rispetto a ciascun peso sinaptico, fornendo così informazione diretta sul funzionamento interno della rete (le connessioni più forti sono quelle più importanti). Si noti che il termine aggiuntivo alla funzione di errore non solo penalizza in generale l'uso di molte connessioni, ma implicitamente penalizza anche l'impiego di connessioni forti rispetto a connessioni deboli; infatti, l'interazione fra due connessioni deboli genera una misura di errore minore rispetto all'impiego di un'unica connessione forte giacché

$$\left(\frac{w}{2}\right)^2 + \left(\frac{w}{2}\right)^2 < w^2 + 0^2$$

Questa proprietà può essere svantaggiosa in alcune situazioni in cui il problema che deve essere appreso richiede lo sviluppo di poche connessioni, ma forti; Hertz *et al.* [1991] suggeriscono quindi di modificare il termine aggiuntivo alla funzione di errore come segue

$$E = E_W + \gamma \sum_i \sum_j \frac{w_{ij}^2}{1 + w_{ij}^2}$$

in modo tale che

$$\varepsilon_{ij} = \frac{2\eta\gamma}{(1 + w_{ij}^2)^2}$$

per cui la costante di decadimento dipende dal valore del peso sinaptico stesso e fa sì che le connessioni deboli scompaiano più rapidamente delle connessioni forti. In modo analogo, è possibile eliminare intere unità interne facendo decadere in modo uguale tutte le connessioni di uno stesso nodo i in funzione dell'intensità del suo output (in parte determinato dall'intensità delle connessioni afferenti)

$$\varepsilon_i = \frac{2\eta\gamma}{\left(1 + \sum_j w_{ij}^2\right)^2}$$

dove ε_i è uguale per tutte le connessioni w_{ij} afferenti allo stesso nodo i [Chauvin 1989; Hanson e Pratt 1989].

4.5.2. *Crescita di unità interne*

Un modo alternativo di modificare l'architettura di una rete neurale consiste nell'iniziare con un piccolo numero di unità interne e di aumentarle gradualmente durante l'addestramento se l'errore della rete non scende a livelli soddisfacenti. Diversi approcci sono stati proposti, ma in quasi tutti i casi l'idea di fondo è che ciascuna nuova unità aggiunta deve occuparsi di rispondere ai pattern che non sono stati appresi correttamente dalle unità precedenti.

L'algoritmo di Frean [1990] prevede l'utilizzo di unità binarie con attivazione a gradino (quindi è utile soprattutto per problemi di categorizzazione) e richiede solamente l'impiego della regola del percettrone semplice. Inizialmente non vengono utilizzate unità interne: ciascuna unità di output tenta di ridurre l'errore sui pattern di input per N cicli di addestramento con la regola del percettrone semplice. Dopo N

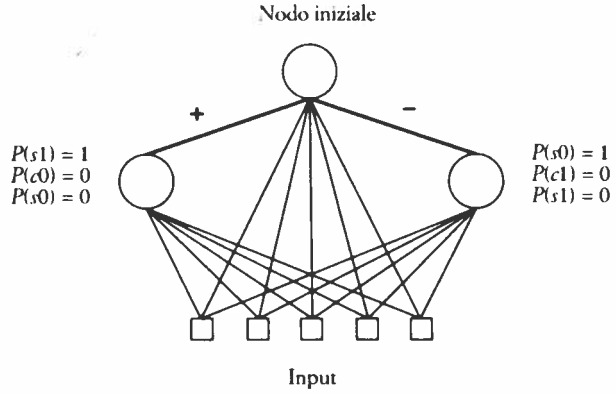


FIG. 2.9. Crescita di unità interne nell'algoritmo di Frean [1990]; si veda il testo per la spiegazione. Lo stesso procedimento si applica in modo indipendente per tutte le unità di output (qui è raffigurata una sola unità) della rete iniziale.

cicli i pattern del gruppo di addestramento possono essere suddivisi in quattro categorie rispetto a ciascuna unità: pattern con risposta corretta che richiedono l'unità attiva $P(c1)$, pattern con risposta corretta che richiedono l'unità inattiva $P(c0)$, pattern con risposta scorretta che richiedono l'unità attiva $P(s1)$ e pattern con risposta scorretta che richiedono l'unità inattiva $P(s0)$. Per ogni unità vengono aggiunte due unità laterali (fig. 2.9), una mirata a correggere la risposta per i pattern del gruppo $P(s1)$ e una mirata a correggere la risposta per i pattern del gruppo $P(s0)$[13]. La prima unità aggiuntiva viene addestrata per N cicli sul gruppo di pattern composto da $P(s1) + P(c0) + P(s0)$ con risposta desiderata uguale a 1 per tutti i pattern del gruppo $P(s1)$ e risposta desiderata uguale a 0 per tutti i restanti pattern; successivamente essa viene collegata all'unità originaria (quella di output) con un peso positivo molto forte (maggiore della somma di tutti i pesi negativi dell'unità originaria) in modo tale che quando un pattern del gruppo $P(s1)$ viene presentato in input la nuova unità fa sì che l'unità originaria produca la risposta corretta 1. In modo analogo l'altra unità aggiuntiva viene addestrata per N cicli sul gruppo di pattern $P(s0) + P(c1) + P(s1)$ a fornire la risposta desiderata 1 per i pattern del gruppo $P(s0)$ e la risposta 0 per i restanti pattern; successivamente essa viene collegata con un peso fortemente negativo (maggiore della somma di tutti i pesi positivi dell'unità originaria) in modo da far sì che l'unità originaria risponda correttamente per tutti i pattern del gruppo $P(s0)$. Lo stesso procedimento viene ripetuto per ciascuna delle unità aggiuntive; questo procedimento continua fino a quando la rete neurale produce la risposta corretta per tutti i pattern di addestramento. Verso la fine, quando le unità aggiuntive devono correggere solo pochi

[13] Se uno dei due gruppi è vuoto, l'unità corrispondente non è necessaria.

pattern, i pattern che richiedono la risposta desiderata 1 vengono presentati più frequentemente per evitare che l'unità risponda sempre 0. L'architettura risultante possiede una struttura gerarchica per ciascuna delle unità originarie della rete[14]. Esiste una particolare versione della regola del percettrone semplice che garantisce la convergenza dell'algoritmo di crescita di Frean. Quando la regola originale del percettrone non riesce a trovare una linea di separazione dei pattern, invece di fermarsi in una soluzione parziale (minimo locale), continua a modificare i pesi sinaptici spostandosi continuamente sulla superficie dell'errore, ma soffermandosi più a lungo nelle zone che corrispondono ai livelli più bassi di errore. Nell'algoritmo di Frean vengono registrati i pesi sinaptici corrispondenti alle zone in cui il percettrone si ferma più a lungo dopo un periodo iniziale di addestramento [Gallant 1986][15]. Quando viene impiegata questa versione della regola del percettrone, l'algoritmo di Frean garantisce che l'addizione automatica di nuove unità fa decrescere l'errore della rete monotonicamente verso zero. Se invece viene utilizzata la versione originale di apprendimento del percettrone, la prova formale di convergenza non è più valida, ma parecchi risultati empirici ne hanno dimostrato comunque l'efficacia in problemi complessi – come ad esempio la classificazione di oggetti tridimensionali [Burgess, Granieri e Patarnello 1992] – e l'efficienza in termini di unità utilizzate [Frean 1990].

Mézard e Nadal [1989] hanno ideato un algoritmo che produce una rete multistrato in cui ciascuno strato viene successivamente creato e addestrato con la regola modificata del percettrone semplice [Gallant 1986]. Il principio di base è che ciascuno strato deve produrre una rappresentazione fedele dell'input: questo significa che i pattern di input per cui sono richieste risposte diverse devono produrre vettori di attivazione diversi per ciascuno strato della rete. Il processo di crescita aggiunge unità al fine di ottenere questa differenziazione e nel contempo ridurre l'errore globale della rete. Anche per questo algoritmo esiste una prova di convergenza.

Fahlman e Lebiere [1990] hanno proposto il seguente algoritmo di crescita che prende il nome di *Cascade correlation*. Il loro procedimento è mirato ad aggiungere nuove unità interne ciascuna delle quali viene addestrata a fornire una risposta correlata con l'errore residuo fornito dalle unità di output sui pattern di addestramento. L'algoritmo è in grado di funzionare con unità continue lineari e non-lineari. L'algoritmo inizia con un'architettura senza unità interne: in questa prima fase le connessioni della rete vengono addestrate con la regola delta

[14] È possibile trasformare l'architettura risultante in una rete a due strati di sinapsi, dove tutte le unità corrispondenti a un'unità originaria (questa compresa) vengono allineate, le connessioni fra di loro vengono eliminate, e ciascuna di esse viene collegata a una nuova unità di output; le nuove connessioni possono essere addestrate con la regola del percettrone utilizzando l'attivazione della vecchie unità per i pattern dell'intero gruppo di addestramento originario.

[15] Per questo motivo viene definito «algoritmo del taschino» (*pocket algorithm*).

fino a quando l'errore smette di scendere. Se l'errore non è sceso al di sotto di un livello accettabile, si passa alla fase di crescita della rete. In questa seconda fase l'algoritmo opera in due tempi e su due tipi di pesi sinaptici diversi: in un primo tempo una nuova unità interna viene connessa alle unità di input e queste connessioni vengono addestrate a massimizzare la correlazione fra l'attivazione di questa unità e l'errore delle unità di output della rete con una particolare regola di apprendimento; in un secondo tempo la nuova unità interna viene collegata alle unità di output e tutte le connessioni delle unità di output (comprese le nuove connessioni dall'unità interna) vengono riaddestrate usando la regola delta (in questa fase le connessioni fra l'unità interna e le unità di input non vengono modificate). Se l'errore non è ancora sceso sotto un livello accettabile, una seconda unità interna viene collegata alle unità di input e alla unità interna precedente e tutte queste connessioni vengono addestrate a massimizzare la correlazione fra l'attivazione di questa nuova unità e l'errore residuo calcolato sulle unità di output per i pattern di addestramento; successivamente anche questa nuova unità interna viene collegata alle unità di output e tutte le connessioni delle unità di output (comprese le connessioni dalle unità interne) vengono riaddestrate per ridurre l'errore sui pattern di addestramento utilizzando la solita regola delta. Questa procedura costruisce una rete con unità interne collegate a cascata fra di loro (ciascuna nuova unità riceve input da tutte le unità interne precedentemente aggiunte) che vengono inizialmente preparate a fornire un output correlato con l'errore residuo delle unità di output (fig. 2.10): per questo motivo l'algoritmo assume il nome di *Cascade correlation* (correlazione a cascata).

Per descrivere il funzionamento dell'algoritmo, utilizzeremo la seguente notazione (si veda anche fig. 2.10). Si immagini di avere una

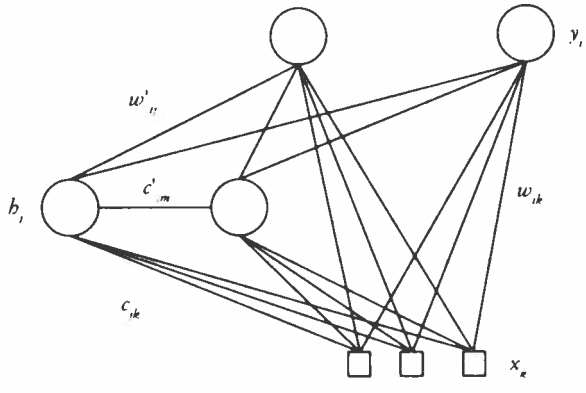


FIG. 2.10. Un esempio di rete costruita dall'algoritmo di *Cascade correlation*. Per chiarezza grafica sono illustrate solamente due unità interne. Si veda il testo per la spiegazione del funzionamento.

rete con unità di output y_i, unità di input x_k e unità interne h_j aggiunte in un secondo tempo una alla volta. Definiamo w_{ik} le connessioni fra le unità di output e le unità di input, w'_{ij} le connessioni fra le unità di output e le unità interne aggiunte, c_{jk} le connessioni fra le unità interne e le unità di input, c'_{jm} le connessioni fra ciascuna unità interna e le unità interne m precedentemente aggiunte (per cui $1 \leq m < j$). L'errore residuo E_i di ciascuna unità di output non è altro che

$$E_i = (y_i - t_i)$$

La correlazione C_j fra l'errore residuo e l'attivazione di un'unità interna è data da

$$C_j = \sum_i \left| \sum_\mu^M (h_j^\mu - \bar{h}_j)(E_i^\mu - \bar{E}_i) \right|$$

dove $|x|$ indica il valore assoluto di x, M è il numero totale di pattern di addestramento, $\bar{h}_j$ è l'attivazione media dell'unità interna calcolata su tutti i pattern di addestramento

$$\bar{h}_j = \frac{1}{M} \sum_\mu^M h_j^\mu$$

e $\bar{E}_i$ è l'errore residuo medio di ciascuna unità di output calcolato su tutti i pattern di addestramento alla fine di ogni fase di addestramento

$$\bar{E}_i = \frac{1}{M} \sum_\mu^M E_i^\mu$$

Siccome desideriamo far sì che ciascuna unità interna fornisca un output massimamente correlato con l'errore residuo della rete, dobbiamo modificare i pesi sinaptici di ciascuna unità interna in modo tale che

$$\Delta c_{jk} = \frac{\partial C_j}{\partial c_{jk}}$$

ovvero salire il gradiente della funzione di correlazione C_j; usando un metodo simile a quello impiegato per derivare la regola di Back-propagation, otteniamo la regola di modifica sinaptica per le connessioni c_{jk}

$$\Delta c_{jk} = \sum_i s_i \sum_\mu^M \dot{\Phi}(A_j^\mu) x_k^\mu (E_i^\mu - \bar{E}_i)$$

dove s_i, che assume valore -1 o $+1$, viene calcolato come segue

$$s_i = \mathrm{sgn}\left(\sum_{\mu}^{M}(h_i^{\mu} - \bar{h}_j)(E_i^{\mu} - \bar{E}_i)\right)$$

In modo analogo, le connessioni c'_{jm} fra ciascuna unità interna e le unità interne m precedentemente aggiunte sono modificate in base alla stessa regola

$$\Delta c'_{jm} = \sum_i s_i \sum_{\mu}^{M} \dot{\Phi}(A_j^{\mu}) h_m^{\mu}(E_i^{\mu} - \bar{E}_i)$$

semplicemente sostituendo l'attivazione dei nodi interni precedenti h_m^{μ} al posto dell'attivazione delle unità di input x_k^{μ}. Le connessioni w_{ik} e w'_{ij} vengono invece modificate con la regola delta per unità continue non-lineari[16]. Tutti i pesi sinaptici iniziali vengono sempre fissati a piccoli valori casuali intorno a zero. In una variante dell'algoritmo vengono addestrate più di una unità interna alla volta e viene mantenuta solamente quella che massimizza maggiormente la correlazione fra il proprio output e l'errore residuo. Shultz ed Elman [1994] descrivono alcuni metodi per analizzare reti neurali ottenute con l'algoritmo di *Cascade correlation* (come l'analisi dei componenti principali dei contributi dei pesi sinaptici in rapporto alla risposta desiderata e all'attivazione dell'unità presinaptica).

Cascade correlation: algoritmo

Consideriamo l'addestramento di una rete neurale su M coppie di pattern; ciascuna unità di output e unità interna aggiunta possiede una funzione di attivazione sigmoide. Tutte le unità di output e unità interne aggiunte possiedono una connessione di *bias* che viene considerata come un peso aggiuntivo collegato a un'unità sempre attiva. Come prima cosa è necessario individuare il criterio di errore E_W minimo per cui la risposta della rete è corretta per tutti i pattern di addestramento. Durante l'apprendimento è possibile utilizzare la regola delta sia con il metodo di addestramento per epoche che con il metodo di addestramento per cicli: per questo motivo nell'algoritmo che segue abbiamo indicato solo la forma di modifica sinaptica. Si ricordi che come sempre i pesi sinaptici vengono aggiornati aggiungendo le variazioni ottenute dalla regola di modifica sinaptica (sia per la regola che massimizza la correlazione che per la regola delta).

1. Iniziare con una rete senza unità interne. Inizializzare tutti i pesi w_{ik} della rete a piccoli valori casuali intorno allo zero (anche i pesi che verranno aggiunti successivamente vanno inizializzati con lo stesso metodo)

$$w_{ik} = \mathrm{rng}(\pm 0{,}5)$$

[16] Gli autori usano spesso una variante dell'algoritmo di Back-propagation, inventata da loro stessi, definita «Quickprop» che descriveremo in seguito.

2. Addestrare i pesi w_{ik} con la regola delta su tutti i pattern di addestramento M fino a quando l'errore E_W raggiunge un livello minimo.

$$\Delta w_{ik} = \eta(t_i - y_i)\dot{\Phi}(A_i)x_k$$

3. Se l'errore E_W non è sufficientemente basso, eseguire i passi 4-7 fino a quando E_W scende al di sotto del criterio minimo stabilito in partenza. Altrimenti fermarsi qui.

4. Ripresentare tutti i pattern e calcolare per ogni unità di output l'attivazione y_i^μ, l'errore residuo E_i^μ e l'errore residuo medio $\bar{E}_i$:

$$y_i^\mu = \Phi\left(\sum_k w_{ik}x_k^\mu + \sum_j w'_{ij}\Phi(A_j^\mu)\right)$$

(si noti che il secondo componente dell'argomento della funzione sigmoide esiste solo se sono già state aggiunte unità interne);

$$E_i^m = (y_i^m - t_i^m)$$

$$\bar{E}_i = \frac{1}{M}\sum_\mu^M E_i^\mu$$

5. Aggiungere un'unità interna, collegarla con pesi c_{jk} casuali alle unità di input e con pesi c'_{jm} casuali alle altre unità interne m (se ve ne sono) e calcolare la sua attivazione h_j^μ e l'attivazione media $\bar{h}_j$ per tutti i pattern di addestramento:

$$h_j^\mu = \Phi\left(\sum_k c_{jk}x_k^\mu + \sum_{m=1}^{m<j} c'_{jm}\Phi(A_m^\mu)\right)$$

(si noti che il secondo componente della funzione sigmoide esiste solo se sono già state aggiunte altre unità interne precedenti);

$$\bar{h}_j = \frac{1}{M}\sum_\mu^M h_j^\mu$$

6. Aggiornare solo i pesi sinaptici dell'unità interna aggiunta fino a quando diventano stabili usando le seguenti regole di apprendimento:

$$\Delta c_{jk} = \sum_i s_i \sum_\mu^M \dot{\Phi}(A_j^\mu)x_k^\mu(E_i^\mu - \bar{E}_i)$$

$$\Delta c'_{jm} = \sum_i s_i \sum_\mu^M \dot{\Phi}(A_j^\mu)h_m^\mu(E_i^\mu - \bar{E}_i)$$

(se altre unità interne precedenti esistono)

dove

$$s_i = \mathrm{sgn}(\sum_{\mu}^{M} (h_i^{\mu} - \bar{h}_i)(E_i^{\mu} - \bar{E}_i))$$

7. Registrare i pesi sinaptici dell'unità interna aggiunta e collegare le unità di output alla nuova unità interna con nuovi pesi sinaptici w'_{ij} inizializzati a piccoli valori casuali intorno allo 0. Quindi ripresentare tutti i pattern di addestramento calcolando l'attivazione delle unità interne e di output e modificare le vecchie connessioni w_{ik} e le nuove connessioni w'_{ij} con la regola delta fino a quando l'errore E_W raggiunge un nuovo minimo:

$$\Delta w_{ik} = \eta(t_i - y_i)\dot{\Phi}(A_i)x_k$$
$$\Delta w'_{ij} = \eta(t_i - y_i)\dot{\Phi}(A_i)h_j$$

4.6. Applicazioni

L'algoritmo di Back-propagation è la tecnica connessionista più utilizzata. Il modello stesso è ancora oggetto di studio e continuano tuttora ad apparire nella letteratura specialistica nuovi lavori teorici e sperimentali che presentano ulteriori variazioni o analisi delle prestazioni dell'algoritmo. Qui di seguito verranno presentati alcuni esempi classici che illustrano le modalità di applicazione del modello. Innanzitutto verranno passati in rassegna alcuni semplici problemi che costituiscono una specie di «batteria di test» spesso usati per valutare le prestazioni di un nuovo modello neurale.

4.6.1. *Una «batteria di test neurali»*

Il continuo fiorire di nuovi modelli neurali e di variazioni degli algoritmi di apprendimento già esistenti richiede la presenza di una serie di compiti standard sulla base dei quali valutare e confrontare le loro prestazioni. Malgrado non vi sia un accordo ufficiale su quali debbano essere questi compiti, alcuni di essi ricorrono frequentemente nella letteratura. Essi tendono ad essere semplici, non realistici e sono mirati soprattutto ad enfatizzare alcuni aspetti chiave della trasformazione da ingresso a uscita che il modello neurale deve essere in grado di risolvere: il loro scopo è quello di rivelare chiaramente le prestazioni, il tipo di soluzioni e le difficoltà incontrate da un determinato algoritmo di apprendimento.

Il compito più famoso è la realizzazione della funzione XOR che abbiamo già incontrato a più riprese nel corso di questo capitolo. La funzione booleana XOR non è un problema realistico (non fa certo parte del tipo di problemi quotidiani che un essere vivente deve risolvere), ma è interessante perché non è linearmente separabile, i vettori di apprendimento hanno dimensionalità ridotta (si possono quindi

analizzare i pesi interni della rete e le dinamiche di apprendimento) e la soluzione è computazionalmente trattabile.

Un altro compito molto usato, che è stato introdotto da Minsky e Papert [1969], è il problema della «parità» in cui si richiede alla rete di segnalare se il pattern binario di ingresso contiene un numero pari o dispari di 1. Questo compito si presta ad essere studiato in una varietà di architetture semplicemente modificando la dimensionalità del vettore di input. Lo XOR, ad esempio, è un particolare problema di parità a dimensionalità 2. Rumelhart, McClelland e Gruppo PDP [1986] hanno mostrato che l'algoritmo di base di Back-propagation richiede almeno N unità nascoste per risolvere un problema di parità di dimensione N (si noti però che se vi sono connessioni dirette anche dalle unità di input a quelle di output sono necessarie meno unità nascoste).

Il problema delle due spirali è stato invece introdotto dagli ideatori dell'algoritmo di *Cascade correlation* [Fahlman e Lebiere 1990]. I pattern di input individuano due spirali concentriche in uno spazio bidimensionale: il compito della rete consiste nell'individuare a quale spirale appartiene un determinato pattern di input (fig. 2.11). Il gruppo di addestramento è definito da 194 punti, ma le prestazioni finali della rete vengono studiate su un campionamento più fitto dell'intera superficie bidimensionale. Il compito è difficile in quanto richiede una com-

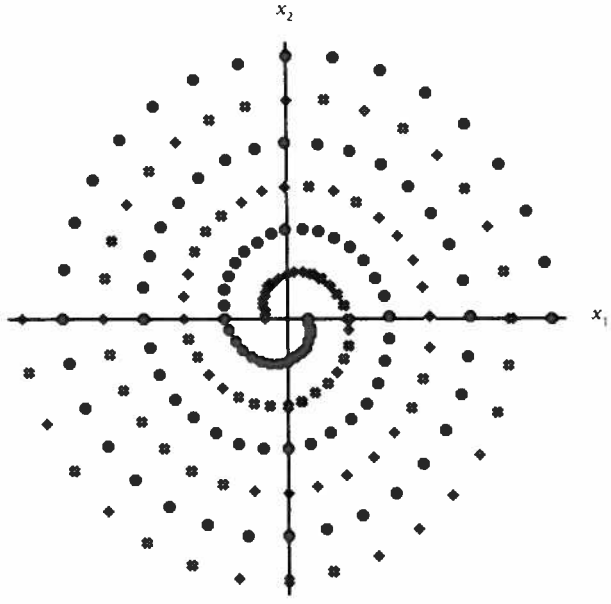


FIG. 2.11. Il problema delle due spirali. I punti neri richiedono risposta 1 e i punti grigi risposta 0. Qualsiasi punto dello spazio può essere utilizzato per osservare la risposta della rete alla fine dell'apprendimento e valutare la qualità della separazione fra le due spirali.

plessa rappresentazione interna dello spazio: reti neurali addestrate con Back-propagation necessitano di un alto numero di unità interne (alcune decine) per poter sviluppare un gruppo adeguato di linee di separazione dello spazio che approssimino la distinzione fra le due spirali; *Cascade correlation* invece è molto più efficiente nell'imparare a distinguere le due spirali in quanto segmenta in modo graduale e sistematico lo spazio durante il processo di crescita delle unità interne.

Il «problema della simmetria» è stato introdotto da Rumelhart, McClelland e Gruppo PDP [1986]. Una rete neurale deve indicare se un vettore di input possieda un asse di simmetria. Il problema è interessante perché la rete neurale deve individuare la linea di simmetria ed effettuare nel contempo un confronto fra le due porzioni del vettore di input per dare la risposta corretta; si noti tuttavia che l'apprendimento della trasformazione corretta non significa che la rete neurale abbia sviluppato un senso di simmetria, come nel caso dei mammiferi con visione retinica e corteccia visiva, perché la permutazione degli elementi di ingresso darebbe luogo alla stessa risposta (si veda il paragrafo 2.4 del primo capitolo).

Il «problema della codifica» richiede che una rete neurale codifichi una serie di pattern in un numero di unità molto inferiori alla dimensionalità dei pattern stessi. Nel caso di Back-propagation questo problema viene risolto come un problema di autoassociazione (o *identity mapping*) utilizzando una rete con N unità di input, M unità interne ed N unità di output. La rete deve apprendere a riprodurre in output il pattern di input utilizzando un numero di unità interne inferiore al numero di unità di input. Se utilizziamo pattern binari, una codifica molto efficiente richiederebbe $M \equiv \log_2 N$ unità interne: in pratica Back-propagation spesso ne usa di meno perché l'impiego dei valori continui di attivazione delle unità interne permette di ampliare le capacità di rappresentazione. Il problema della codifica è scalabile (può essere impiegato usando dimensionalità e numero di pattern diversi) e descrive i requisiti fondamentali di alcune importanti applicazioni, come la compressione di immagini e la trasmissione e ricostruzione di informazione.

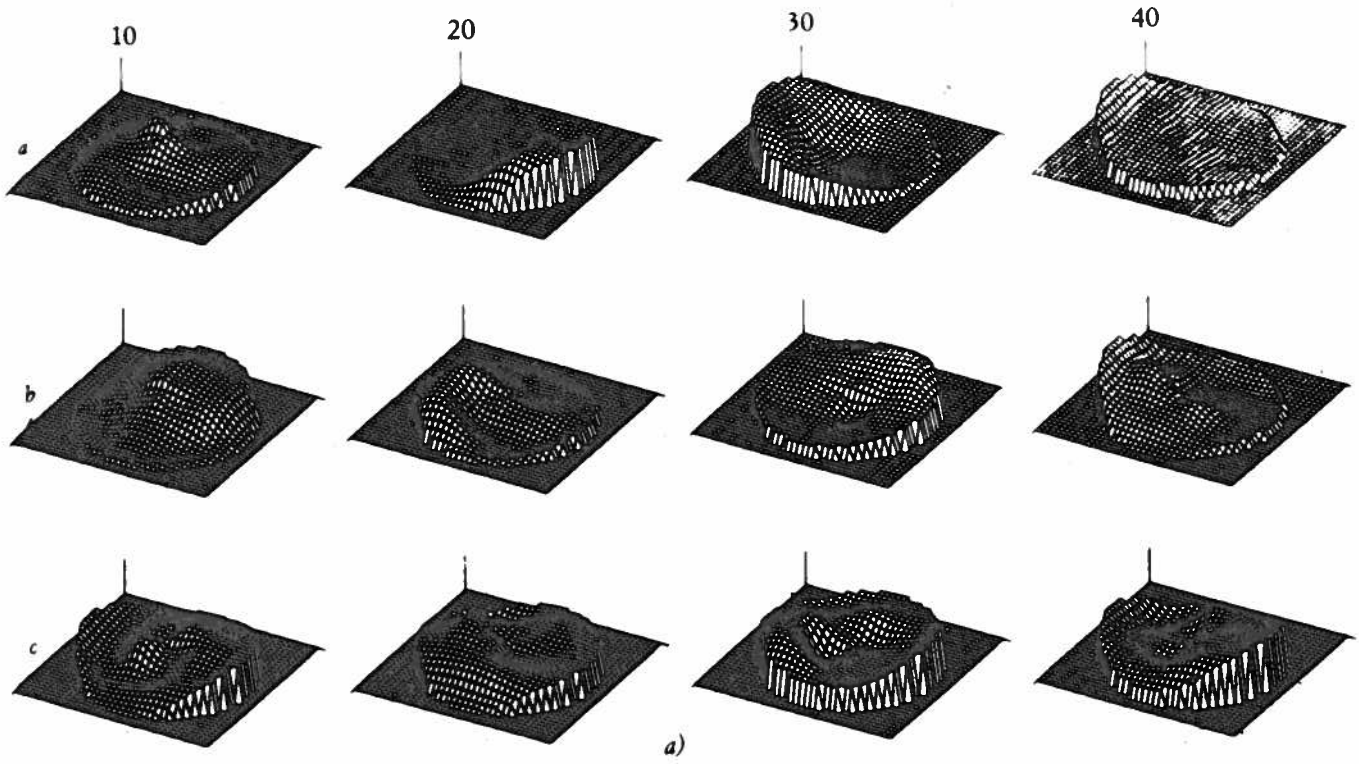
4.6.2. *Modellizzazione di processi neurali e cognitivi*

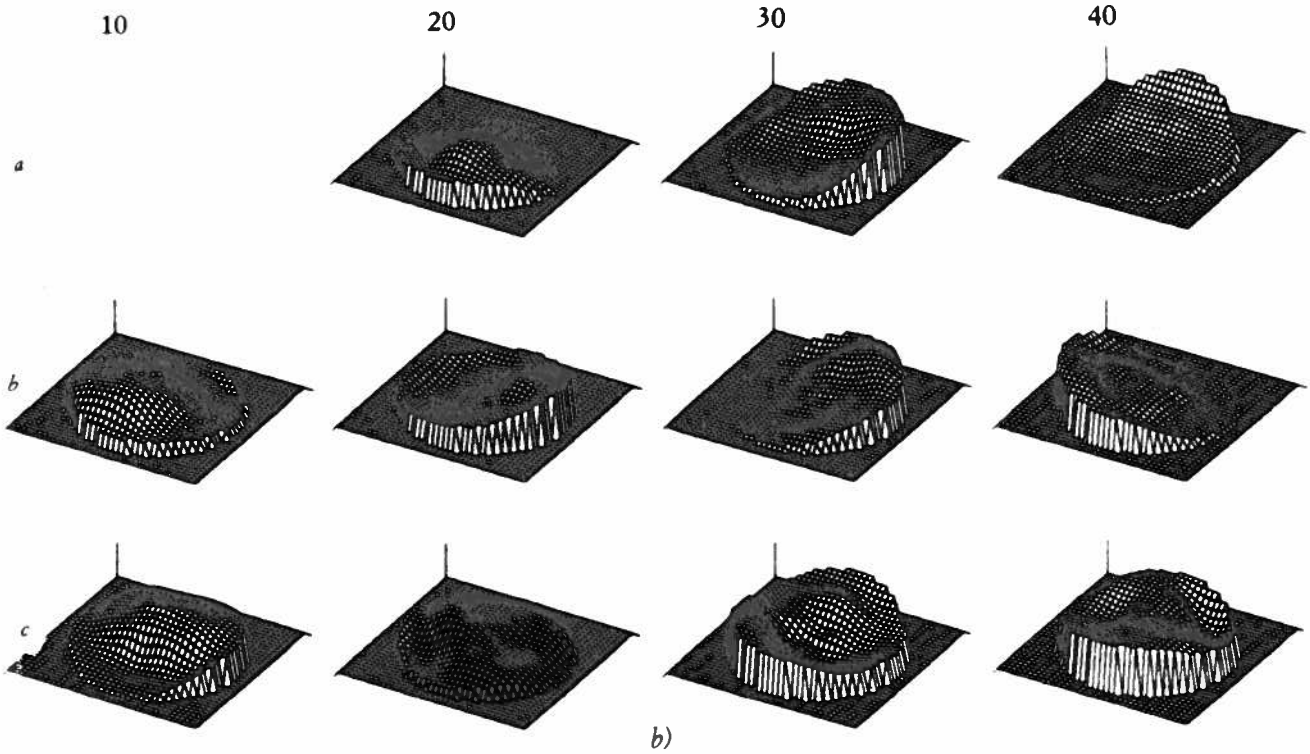
Per quanto l'algoritmo di Back-propagation non sia biologicamente plausibile, soprattutto per via della trasmissione all'indietro dei segnali d'errore attraverso le stesse connessioni sinaptiche [Crick 1989], esso è stato impiegato per modellare e spiegare i meccanismi sottostanti una grande varietà di prestazioni cognitive, visive e motorie. In questi modelli l'accento viene posto sulle caratteristiche e sui requisiti della trasformazione che la rete neurale svolge piuttosto che sul meccanismo specifico che viene impiegato per apprendere tale trasformazione.

Zipser e Andersen [1988] hanno utilizzato una rete a due strati di

connessioni per apprendere la trasformazione da coordinate retiniche a coordinate cranio-centriche che viene normalmente eseguita dal sistema nervoso visivo dei mammiferi per ottenere la posizione nello spazio degli oggetti rispetto alla testa dell'osservatore. Questa trasformazione richiede una semplice addizione di due vettori: il vettore di input dell'immagine retinica e il vettore che rappresenta l'attuale posizione degli occhi. Siccome l'addizione di due vettori è un'operazione lineare, una rete neurale con un solo strato di connessioni sarebbe in grado di eseguirla. Tuttavia gli autori hanno impiegato anche alcune unità nascoste ad attivazione sigmoide per poter studiare il tipo di rappresentazione risultante. Una parte delle unità di input intendevano simulare grossolanamente una specie di retina composta di 64 recettori su cui veniva proiettato un oggetto; le unità rimanenti codificavano invece la posizione del bulbo oculare sull'asse verticale e orizzontale rispetto al punto di fissazione fronto-centrale dell'osservatore. Le unità di output avrebbero dovuto fornire la posizione dell'oggetto in coordinate cranio-centriche: siccome questa era la parte più delicata del modello perché non è chiaro come questa informazione venga rappresentata nel sistema nervoso (e tanto meno da dove provenga la risposta necessaria all'algoritmo di Back-propagation per apprendere la trasformazione!), gli autori hanno utilizzato due diversi formati senza trovare però differenze significative nei risultati. In entrambi i casi la rete neurale apprendeva a compiere la trasformazione richiesta. Le caratteristiche di attivazione delle unità interne alla fine dell'addestramento erano qualitativamente simili a quelle dei neuroni dell'area corticale posteriore 7a che si sanno coinvolti in compiti spaziali e per i quali esiste una dettagliata serie di registrazioni neurofisiologiche (fig. 2.12). Questi risultati possiedono due risvolti: da un lato hanno fornito una chiave interpretativa al tipo di operazione svolta dai neuroni biologici coinvolti nell'operazione, dall'altro lato hanno indicato che l'algoritmo di Back-propagation è in grado di catturare il tipo di trasformazione operata da meccanismi neurofisiologici malgrado l'implausibilità di alcune sue parti e la grossolanità del modello di neurone artificiale impiegato. Da un punto di vista neurofisiologico il paradigma connessionista può quindi aiutare a interpretare dati neurofisiologici molto complessi che non sono altrimenti interpretabili in modo intuitivo. Tuttavia i modelli neurali artificiali sono la realizzazione di teorie e speculazioni sul funzionamento nervoso che necessariamente possono catturare solo aspetti generali delle operazioni che il sistema nervoso esegue. I modelli connessionisti non possono certo essere impiegati per spiegare meccanismi fisiologici, ma possono essere illuminanti nel processo di comprensione di determinati comportamenti neurali e delle prestazioni comportamentali relative ad essi.

Le reti neurali non devono necessariamente essere utilizzate per la costruzione di modelli, ma possono essere impiegate anche per far luce sulle proprietà computazionali che caratterizzano le prestazioni comportamentali degli organismi biologici. Floreano, De Lillo e Anti-





12. *a*) Registrazione sperimentale dei campi recettivi dei neuroni biologici. I dati necessari per produrre ciascun grafico provengono dalla registrazione della frequenza di risposta di un singolo neurone dell'area corticale posteriore 7a per 17 diverse zone della retina. Queste zone sono scelte a 10, 20, 30 e 40 gradi dal centro. I dati sono stati poi filtrati e normalizzati. *b*) Risposta delle unità interne della rete neurale artificiale sottoposta alle stesse condizioni di input; anche in questo caso le risposte sono state filtrate e normalizzate. Il confronto fra i due gruppi di dati si basa sull'eccentricità e sui picchi dei campi recettivi dei neuroni biologici e di quelli artificiali.

nucci [1995] hanno impiegato una rete neurale a due strati per risolvere un compito di inferenza transitiva su una serie a cinque termini. Il compito studiato con la rete neurale è stato impiegato con bambini piccoli e con varie specie di animali; esso funziona nel modo seguente. I soggetti vengono inizialmente addestrati mediante un programma di rinforzo non verbale a scegliere un oggetto per ciascuna coppia di presentazione [A+ B–], [B+ C–], [C+ D–] e [D+ E–], dove il segno + indica l'oggetto che deve essere scelto e il segno – l'oggetto che non deve essere scelto. La capacità di effettuare inferenza transitiva è evidenziata dalla scelta del termine B nella coppia di test [B, D], i cui termini non sono adiacenti, sono stati entrambi rinforzati positivamente e negativamente con la stessa frequenza durante l'addestramento, ma sono collocati diversamente nella serie A > B > C > D > E. L'interesse in questo tipo di compito consiste nel fatto che, sebbene esso fosse considerato un test per valutare l'insorgere di capacità logiche nei bambini, una serie di studi sperimentali recenti hanno rivelato che molte specie animali sono in grado di risolverlo. Un gruppo di reti neurali addestrate nelle stesse condizioni (non verbali) dei bambini pre-operatori e degli animali sono in grado di risolvere correttamente la coppia [B, D]; inoltre le reti rispecchiano le prestazioni degli animali e dei bambini per quanto riguarda le modalità di apprendimento, le prestazioni su tutte le altre coppie e i comportamenti su una serie di altre manipolazioni standard delle coppie. Il fatto che le reti neurali non riflettano però il tipo di prestazioni dimostrate dai soggetti adulti, ha permesso di ipotizzare che vi siano due tipi di meccanismi funzionali responsabili per l'inferenza transitiva: uno basato su delle semplici regole associative apprese in base a un paradigma di rinforzo (utilizzato da bambini piccoli, animali e semplici reti neurali) e uno basato su dei ragionamenti logici che richiedono un processo di seriazione (impiegato invece dai soggetti adulti).

Vi sono molti altri esempi in cui reti neurali addestrate con Back-propagation (o sue variazioni) sono state impiegate per far luce su prestazioni comportamentali e cognitive: ad esempio apprendimento del linguaggio [Parisi, Pagliarini e Floreano 1994; Seidenberg e McClelland 1989], dislessia [Plaut e Shallice 1991] e afasia ottica [Plaut e Shallice 1992], seriazione [Mareschal e Schultz 1993], effetto stroop [Cohen, Dumbar e McClelland 1990; Voltolina e Umiltà 1994], rappresentazioni semantiche [Farah e McClelland 1991] e molti altri per i quali si rimanda il lettore alle riviste specialistiche e ai testi tematici [Churchland e Sejnowski 1992; trad. it. 1995; Clark 1989; trad. it. 1994; Parisi 1989; Quinlan 1995].

4.6.3. *Applicazioni ingegneristiche*

L'algoritmo di Back-propagation è stato anche impiegato in molteplici e varie applicazioni pratiche. In questa sezione osserveremo tre

tipi di applicazioni: la lettura di testi scritti, il riconoscimento di caratteri e la guida automatica di un veicolo.

NETtalk (fig. 2.13) è una rete neurale a due strati di connessioni che impara a pronunciare un testo scritto in inglese [Sejnowski e Rosenberg 1987; Anderson e Rosenfeld 1998]. L'input della rete consiste in una finestra di sette caselle che scorre sul testo scritto avanzando di un carattere alla volta. La rete deve apprendere ad attivare l'unità di output corrispondente alla pronuncia fonetica della lettera che si trova nella casella centrale della finestra di input. Le tre caselle precedenti e successive servono unicamente a fornire l'informazione contestuale necessaria per l'attivazione del fonema corretto. Ciascuna casella di input consiste di 29 unità che codificano in modo locale le lettere dell'alfabeto e i segni di interpunzione. 26 unità di output invece codificano in modo distribuito i fonemi della lingua inglese e sono collegate a un

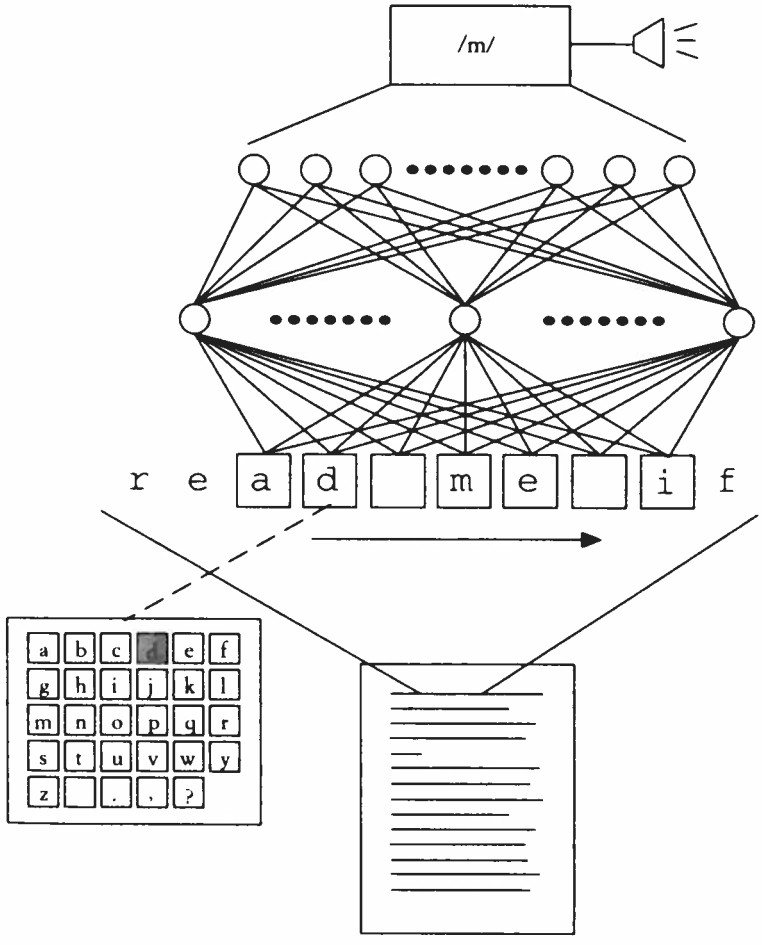


FIG. 2.13. Architettura di NETtalk, una rete che apprende a pronunciare testi scritti.

generatore di suoni. La rete possiede 80 unità nascoste (tuttavia diverse grandezze sono state impiegate nei vari esperimenti condotti dagli autori), ciascuna delle quali riceve segnale da tutte le 7 × 29 unità di input ed è connessa a tutte le unità di output. Quando addestrata su un testo di 1.024 parole, la rete ottiene un'accuratezza del 95% dopo 50 epoche di modifica sinaptica ed è in grado di leggere nuovi testi in modo comprensibile (le prestazioni possono essere migliorate se il *corpus* di parole di apprendimento viene aumentato). Una caratteristica interessante è la modalità di apprendimento: la rete impara innanzitutto a segmentare le parole, poi comincia ad emettere dei suoni simili alla lallazione dei bambini e infine comincia a pronunciare le parole – inizialmente gli articoli e le parole corte e poi quelle più lunghe. L'analisi della struttura interna della rete rivela che le unità interne rappresentano separatamente vocali e consonanti in funzione delle loro caratteristiche di pronuncia. Le prestazioni del modello sono inferiori a quelle di alcuni meccanismi attualmente in commercio, ma NETtalk è interessante perché richiede uno sforzo minimo per la realizzazione ed apprende automaticamente da esempi.

Il riconoscimento di caratteri è un settore applicativo in cui le reti neurali hanno ottenuto un grande successo in quanto esse possiedono le caratteristiche di compattezza, generalizzazione e flessibilità richieste a un sistema che deve essere in grado di operare efficacemente su un *corpus* di dati caratterizzati da moltissime distorsioni e rumore. Talvolta vengono incorporate nei programmi OCR (*Optical Character Recognition*) commercialmente disponibili per i computer da scrivania offrendo anche il vantaggio di poter continuare ad apprendere se effettuano un riconoscimento errato. Sistemi neurali simili sono attualmente in funzione anche in alcuni servizi postali per la lettura dei codici di avviamento postale. Qui descriviamo brevemente il funzionamento di una sola applicazione, anche perché la maggior parte dei modelli funzionanti sono coperti da segreto industriale. Le Cun *et al.* [1990] hanno impiegato una rete neurale a 4 strati di connessioni per il riconoscimento di numeri nei codici di avviamento postale. La rete neurale riceve input da un sistema di pre-elaborazione che individua la sequenza di numeri sulla busta, raddrizza e ridimensiona i caratteri e trasmette un numero alla volta su una matrice di 16 × 16 pixel. Un primo strato di elaborazione è composto da 12 gruppi di 8 × 8 unità; ciascun gruppo riceve segnali da un'area corrispondente sulla matrice di input spostata di due pixel rispetto agli altri gruppi in modo da coprire interamente l'area con una sovrapposizione di due pixel per lato. Ciascuno di questi gruppi possiede esattamente le stesse connessioni sinaptiche (questa tecnica è nota con il nome di «condivisione dei pesi») in modo tale che sia sufficiente addestrare un solo gruppo di connessioni e copiarne i valori per gli altri gruppi di neuroni. Uno strato successivo di unità è costruito nello stesso modo, ma riduce ulteriormente la rappresentazione dell'input: vi sono 12 gruppi di 4 × 4 unità che sfruttano nuovamente la tecnica di condivisione dei pesi.

Uno strato di 30 unità successive riceve input da tutte le unità precedenti e trasmette segnale a 10 unità di output che codificano localmente i 10 numeri da 0 a 9. Per aumentare ulteriormente le capacità di generalizzazione del modello, Le Cun, Denker e Solla [1990], hanno ridotto il numero di parametri eliminando gradualmente connessioni sinaptiche e riaddestrando la rete sull'intero gruppo di pattern di addestramento (circa 7.000 caratteri); grazie a questo metodo gli autori sono stati in grado di ottenere una generalizzazione corretta del 99% su 2.000 nuovi caratteri.

ALVINN (*Autonomous Land Vehicle In a Neural Network*) è una rete neurale a due strati di connessioni che apprende a controllare un veicolo osservando le mosse di un guidatore umano (fig. 2.14) [Pomerleau 1993]. Il veicolo è equipaggiato con una telecamera che riduce la scena frontale su una matrice 30 × 32 che viene trasmessa in tempo reale a un elaboratore in cui è simulata la rete neurale; ciascun pixel trasmette il proprio livello di attivazione a 4 unità interne ad attivazione sigmoide, le quali a loro volta sono connesse a 30 unità di uscita. Le unità di uscita codificano l'angolo di sterzo del veicolo a destra e a sinistra rispetto alla direzione frontale (che corrisponde all'unità centrale). L'apprendimento avviene mostrando alla rete neurale un film ripreso mentre una persona guida il veicolo e utilizzando la registrazione dell'angolo di sterzo come risposta desiderata per ciascuna immagine di input. Una volta che la fase di apprendimento è stata completata, la rete neurale è in grado di controllare autonomamente il veicolo in diverse condizioni di terreno: attualmente ALVINN è in gra-

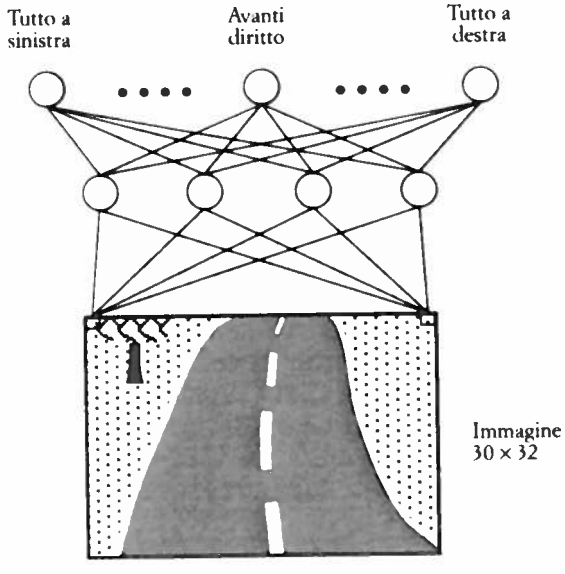


FIG. 2.14. L'architettura di ALVINN, una rete neurale per il controllo autonomo di un veicolo.

do di condurre un normale furgoncino su strade dei quartieri periferici di Pittsburgh, su tracciati sterrati ed è stato anche testato sull'autostrada a una velocità di circa 100 km/h su una distanza di circa 150 km. Un miglioramento della generalizzazione (che qui è ovviamente cruciale come nel caso della lettura del testo scritto e del riconoscimento di caratteri) è stato ottenuto utilizzando la tecnica di condivisione dei pesi sinaptici provenienti da unità vicine sulla matrice di input (questa particolare strategia, oltre a diminuire il numero di connessioni che debbono essere effettivamente addestrate, fornisce informazione topologica alle unità interne, le quali altrimenti non possiederebbero alcuna informazione sulle proprietà geometriche dell'input) e adottando un tasso di apprendimento inversamente proporzionale al numero di connessioni per ciascun nodo postsinaptico. Successivamente, le prestazioni di ALVINN sono state ulteriormente migliorate combinando Back-propagation a un algoritmo genetico (si veda il sesto capitolo). Un importante vantaggio di ALVINN rispetto ad altri sistemi di guida automatici è che essa è in grado di apprendere a guidare virtualmente su ogni tracciato perché la rete neurale estrae autonomamente le caratteristiche visive rilevanti per il controllo adeguato del veicolo; i sistemi tradizionali si basano invece su tecniche classiche di segmentazione dell'immagine e localizzazione di elementi come il bordo della strada, la linea centrale, o il retro degli altri veicoli e sono quindi molto più vincolati per quanto riguarda il tipo di ambiente operativo.

4.7. Back-propagation e metodi di regressione lineare

Sia nel caso in cui una rete debba apprendere ad associare dei pattern di input con dei pattern di uscita appropriati (ad esempio, NETtalk e ALVINN) che nel caso in cui essa debba classificare gli input in categorie (ad esempio nel caso del riconoscimento di caratteri o nel problema della parità e delle due spirali), l'apprendimento consiste nel trovare la matrice di pesi sinaptici che approssima la funzione di trasformazione richiesta da input ad output. In quest'ottica le reti neurali svolgono operazioni simili alle tecniche statistiche di regressione. In statistica l'analisi della regressione consiste nel selezionare una funzione di alcuni parametri e nel cercare di adattarla ai dati disponibili utilizzando – ad esempio – il criterio dello scarto quadratico medio. Questa tecnica, introdotta da Carl Gauss due secoli fa, è concettualmente equivalente all'individuazione dei pesi sinaptici che riducono al minimo lo scarto quadratico medio fra l'output dei neuroni artificiali e i dati disponibili[17]. Tuttavia la maggior parte dei metodi statistici di re-

[17] Molti dei problemi affrontati con reti neurali sono stati tipicamente dominio della statistica. Col diffondersi impetuoso delle reti neurali, vi è stata una presa di coscienza delle similitudini fra l'approccio classico della statistica e il nuovo metodo

gressione sono lineari [Hecht-Nielsen 1990], in quanto le funzioni di approssimazione sono solamente una combinazione lineare dei parametri; Back-propagation invece non è soggetta ai vincoli di linearità ed è molto più semplice da utilizzare. Un altro vantaggio delle reti neurali è che, dato il carattere interdisciplinare della ricerca che vi sta dietro, esse riassumono in sé le conoscenze, soluzioni e metafore di molte discipline scientifiche, dalla biologia all'informatica; esse inoltre si prestano facilmente a essere utilizzate in diversi settori applicativi. Alcuni ricercatori non sono però d'accordo con questa interpretazione e rivendicano la potenza ed efficienza dei metodi statistici tradizionali [Ripley 1992].

4.8. Quickprop

L'algoritmo di Back-propagation non è il solo metodo per effettuare la discesa del gradiente della funzione di errore E_W. Diverse tecniche sono state proposte per rendere più efficiente la ricerca del minimo globale della trasformazione da apprendere. Alcuni metodi tentano di individuare esattamente il valore ottimale del tasso di apprendimento («metodo della discesa più rapida del gradiente» e «metodo del gradiente coniugato»), altri invece fanno uso del calcolo della derivata seconda della funzione di errore rispetto allo spostamento dei pesi sinaptici: benché queste tecniche risultino talvolta più efficienti dell'algoritmo di Back-propagation, esse non sono sempre locali e richiedono spesso calcoli abbastanza complessi (si veda Kramer e Sangiovanni-Vincentelli [1989], Riedmiller [1994] e Reed e Marks [1999] per una rassegna e un confronto delle prestazioni dei vari metodi).

Fahlman [1989] ha proposto un algoritmo di discesa del gradiente completamente locale (per cui la modifica di una connessione richiede solo informazione direttamente disponibile a livello dell'unità presinaptica e di quella postsinaptica) che è in parte simile a Back-propagation. Il metodo è basato sull'assunzione che la discesa della superficie di errore di un singolo peso sinaptico è relativamente indipendente dalle modifiche che vengono apportate agli altri pesi sinaptici. Malgrado questa assunzione sia solo un'approssimazione, l'algoritmo è notevolmente più rapido di Back-propagation ed è stato quindi chiamato Quickprop (propagazione rapida). Non sempre esso trova una soluzio-

neurocomputazionale e sono apparsi una serie di lavori in cui nuovi algoritmi e modelli neurali sono espressi esplicitamente in termini statistici [Ripley 1992; Sethi e Jain 1991; Bishop 1995]. La relazione fra le tecniche statistiche tradizionali e le reti neurali è stata riassunta nel modo seguente: «Le reti neurali sono statistica per amatori. Una rete neurale costruita in modo appropriato esegue buona inferenza statistica basandosi su quello che ha visto durante l'apprendimento e su quello che vede quando risponde. La maggior parte delle reti neurali nascondono la statistica all'utente...» [Anderson, Pellionisz e Rosenfeld 1990, 541].

ne (che verrebbe invece trovata da Back-propagation a parità di pesi sinaptici iniziali), ma, data la rapidità di convergenza, è sufficiente far ripartire l'addestramento con nuovi pesi sinaptici se si nota che dopo alcune decine di epoche l'errore non cala. Quickprop è probabilmente la versione di Back-propagation più utilizzata. Il metodo consiste nel modificare ciascun peso sinaptico in base al confronto in due tempi successivi della variazione dell'errore ottenuta grazie a quel peso: se la variazione è nella stessa direzione della variazione precedente, allora il peso verrà modificato nella stessa direzione del cambiamento effettuato in precedenza, altrimenti esso verrà modificato nella direzione opposta. Per ogni peso della rete, la regola di modifica sinaptica è data da

$$\Delta w^t = \frac{S^t}{S^{t-1} - S^t} \Delta w^{t-1}$$

dove S esprime la variazione della funzione di errore E_W su tutti i pattern di addestramento per quel peso sinaptico

$$S = \frac{\partial E_W}{\partial w}$$

utilizzando la stessa notazione impiegata per la descrizione dell'algoritmo di Back-propagation (si veda il paragrafo 4.3), per i pesi sinaptici appartenenti alle unità di output, S diventa

$$S_{ij} = -\sum_{\mu} \delta_i^{\mu} h_j^{\mu} = -\sum_{\mu} (t_i^{\mu} - y_i^{\mu}) \dot{\Phi}(A_i^{\mu}) h_j^{\mu}$$

mentre per i pesi sinaptici appartenenti alle unità interne

$$S_{jk} = -\sum_{\mu} x_k^{\mu} \delta_j^{\mu} = -\sum_{\mu} x_k^{\mu} \dot{\Phi}(A_j^{\mu}) \sum_i w_{ij} \delta_i^{\mu}$$

Quickprop richiede solamente il calcolo dei delta (come nel caso di Back-propagation) e la memorizzazione di S dell'epoca precedente per ciascun peso sinaptico. Vi è comunque il problema che quando S assume la stessa direzione (segno) e valore per due epoche consecutive, il cambiamento dei pesi sinaptici cresce all'infinito sbalzando improvvisamente l'intera rete al di fuori della zona ove si colloca il minimo globale. Per evitare questa situazione, viene applicato un limite λ per cui $\Delta w^t \leq \lambda \Delta w^{t-1}$ (di solito $\lambda = 1{,}75$). Inoltre una piccola frazione della discesa di gradiente viene addizionata per evitare il «congelamento» dei pesi che potrebbe avvenire quando un S diverso da 0 viene incontrato dopo un S uguale a 0.

Quickprop: algoritmo

Data una rete neurale con due strati di connessioni v_{jk} e w_{ij}, x_k unità di input, h_j unità interne e y_i unità di output (si veda descrizione nel paragrafo 4.1 e fig. 2.3) con attivazione sigmoide $\Phi(A) = (1 + e^{-\beta A})^{-1}$ (dove $\beta = 1$).

1. Inizializzare i pesi sinaptici (*bias* incluso) a piccoli valori casuali:

$$w_{ij} = rnd(\pm 0{,}1)$$
$$v_{jk} = rnd(\pm 0{,}1)$$

2. Eseguire i passi 3-10 fino a quando la funzione di errore E_W raggiunge un livello minimo («criterio») stabilito oppure cessa di diminuire.
3. Per tutte le coppie di addestramento $\mathbf{s}, \mathbf{t}$ $(s_k, t_i \in \Re)$ eseguire i passi 4-8 (un ciclo).
4. Stabilire i valori dei nodi d'ingresso:

$$x_k = s_k$$

5. Calcolare l'attivazione di ciascun nodo interno

$$h_j = \Phi\left(\sum_{k=0} v_{jk} x_k\right)$$

e quindi l'attivazione di ciascun nodo di uscita

$$y_i = \Phi\left(\sum_{j=0} w_{ij} h_j\right)$$

6. Calcolare i delta per i nodi di uscita

$$\delta_i^\mu = (t_i^\mu - y_i^\mu)\dot{\Phi}(A_i^\mu)$$

7. Calcolare i delta per i nodi interni propagando all'indietro i delta dei nodi dello strato superiore

$$\delta_j^\mu = \dot{\Phi}(A_j^\mu)\sum_i w_{ij}\delta_i^\mu$$

8. Calcolare i gradienti di ciascun peso sinaptico da unità interne a unità di output

$$S_{ij}^\mu = -\delta_i^\mu h_j^\mu$$

e da unità di input a unità interne

$$S_{jk}^\mu = -\delta_j^\mu x_k^\mu$$

sommando il valore ottenuto per tutti i pattern di addestramento.

9. Calcolare le modifiche sinaptiche per entrambi gli strati di connessioni usando la somma dei gradienti ottenuta per ciascun peso sinaptico

$$\Delta w_{ij}^{t} = \frac{S_{ij}^{t}}{S_{ij}^{t-1} - S_{ij}^{t}} \Delta w_{ij}^{t-1} + \eta S_{ij}^{t} \qquad \text{tale che } \Delta w^{t} \leq \lambda \Delta w^{t-1}$$

e

$$\Delta v_{jk}^{t} = \frac{S_{jk}^{t}}{S_{jk}^{t-1} - S_{jk}^{t}} \Delta v_{jk}^{t-1} + \eta S_{jk}^{t} \qquad \text{tale che } \Delta v^{t} \leq \lambda \Delta v^{t-1};$$

solo nel caso della prima epoca di addestramento, invece

$$\Delta w_{ij}^{0} = -\eta S_{ij}^{0}$$

e

$$\Delta v_{jk}^{0} = -\eta S_{jk}^{0}$$

10. Applicare le modifiche così calcolate ai rispettivi pesi sinaptici

$$w_{ij} = w_{ij}^{t-1} + \Delta w_{ij}$$

e

$$v_{jk} = v_{jk}^{t-1} + \Delta v_{jk}$$

4.9. Altre funzioni di attivazione

Abbiamo precedentemente osservato che una rete multistrato è in grado di apprendere trasformazioni non-linearmente separabili solamente se vengono utilizzate funzioni di attivazione non-lineari per le unità nascoste. Oltre alla funzione sigmoide e a quella tangente iperbolica, vi sono anche altre funzioni che possiedono caratteristiche interessanti e possono essere facilmente usate in reti multistrato con apprendimento supervisionato.

4.9.1. *Funzioni a base radiale*

Una funzione a base radiale genera un output non-negativo per tutti i pattern ed è simmetrica rispetto al baricentro di un gruppo di input. Le funzioni più comunemente usate sono delle variazioni della *funzione gaussiana*

$$\Phi(x) = \exp(-x^2)$$

la cui derivata prima è esprimibile semplicemente in funzione dell'output generato

$$\dot{\Phi}(x) = -2x\Phi(x)$$

Le funzioni a base radiale sono particolarmente utili nel caso in cui i pattern di input si distribuiscano in gruppetti nello spazio; in questo caso durante l'apprendimento ciascuna unità centra la propria risposta sul baricentro (la media) di un gruppetto di pattern.

Moody e Darken [1989] hanno proposto di utilizzare una funzione gaussiana leggermente più complessa, per cui il vettore di attivazione dell'intero strato di unità interne assume lunghezza 1 (ovvero genera sempre una risposta normalizzata, qualunque sia il pattern di input)

$$\Phi_i(\mathbf{x}) = \frac{\exp\left(\frac{(\mathbf{x} - \mu_i)^2}{2\sigma_i^2}\right)}{\sum_k \exp\left(\frac{(\mathbf{x} - \mu_k)^2}{2\sigma_k^2}\right)}$$

dove μ_i e σ_i^2 sono rispettivamente la media e la varianza della funzione dell'unità i, e il denominatore è la sommatoria dell'output di tutti i nodi k dello stesso strato. I valori appropriati di μ_i si possono ottenere con la tecnica di «quantizzazione vettoriale» (si veda il capitolo quarto) che consiste nello spostare il loro valore verso il centro del gruppo di pattern per cui l'unità risponde maggiormente. Le varianze σ_i^2 vengono invece prefissate in base alle conoscenze sulla distribuzione degli input oppure usando la distanza tra ciascun μ_i e i primi vettori vicini (i più simili). Come risultato si ottiene una tassellatura completa dello spazio dei pattern di input ove ciascuna tessera (unità) è collocata sul centro di un gruppetto di pattern. A questo punto i valori dei pesi sinaptici dalle unità interne ad attivazione radiale alle unità di output possono essere addestrati utilizzando la semplice regola delta [Bishop 1995; Haykin 1999].

4.9.2. *Funzioni logaritmiche*

Talvolta, specialmente per reti che possiedono un alto numero di unità presinaptiche, è utile impiegare delle funzioni che non sono sottoposte a saturazione, come è invece il caso per la funzione sigmoide. Quando una funzione di attivazione è saturata, la correzione della risposta richiede un grosso spostamento dei pesi sinaptici; inoltre la riduzione dell'errore con unità già saturate è difficile perché equivale a spostarsi su una zona piatta della superficie della funzione di errore. Se la struttura della rete o dei dati di input è tale da causare saturazio-

ne, è utile impiegare una funzione di attivazione diversa, come ad esempio la seguente funzione logaritmica

$$\Phi(x) = \begin{cases} \log(1+x) & \text{se } x > 0 \\ -\log(1-x) & \text{se } x < 0 \end{cases}$$

la cui derivata prima è data da

$$\dot{\Phi}(x) = \begin{cases} \dfrac{1}{1+x} & \text{se } x > 0 \\ \dfrac{1}{1-x} & \text{se } x < 0 \end{cases}$$

Fausett [1994] riporta un incremento della velocità di convergenza per lo XOR quando questa funzione logaritmica viene usata solo per le unità nascoste e una semplice funzione lineare viene invece usata per le unità di output.

4.10. La dimensione temporale

Le reti neurali feedforward esaminate nelle sezioni precedenti possono apprendere solamente una trasformazione immediata da input a output, ma non sono in grado di cogliere informazione temporale. In tutti gli esempi considerati fino ad ora ciascuna coppia di pattern d'addestramento era indipendente dalle altre e l'ordine di presentazione delle coppie non era un requisito necessario per l'apprendimento del compito. Tuttavia in molte situazioni realistiche l'input di un sistema neurale (biologico o artificiale) possiede una struttura temporale che è essenziale per la produzione della risposta corretta. Un brano musicale non è definito solamente dalle note che lo compongono, ma anche dalla loro sequenza; la combinazione casuale delle parole di una frase può renderne il significato più o meno comprensibile; l'esecuzione di una certa azione motoria può basarsi su quanto è successo negli istanti precedenti; la pianificazione richiede la formulazione di una successione di azioni o schemi, ecc. Un'altra caratteristica del mondo reale è che molte situazioni richiedono una risposta solo al termine di una sequenza di eventi, mentre le reti esaminate in precedenza producono una risposta (e richiedono un segnale di correzione esterno) per ciascun singolo evento. La dimensione temporale è molto importante non solo nella modellizzazione di prestazioni cognitive e motorie (tutti gli organismi biologici vivono in una dimensione spaziale e temporale), ma anche in un gran numero di applicazioni: dal riconoscimento del parlato alla predizione dei mercati finanziari fino al controllo di sistemi robotici.

Un modo per dotare una rete neurale di una dimensione temporale consiste nel provvedere i singoli nodi della rete di una dinamica

temporale interna o memoria. Questo approccio verrà analizzato in dettaglio nel capitolo 5. Qui di seguito considereremo invece la tecnica che ottiene questo risultato introducendo nella rete dei collegamenti di retroazione.

4.10.1. *Reti parzialmente ricorrenti*

Un semplice metodo per permettere a una rete di cogliere la struttura temporale dell'input consiste nell'aggiungere connessioni di retroazione («ricorrenti») fra le unità di uno stesso strato o dalle unità di strati superiori a quelle di strati inferiori senza modificare il modello di neurone artificiale che abbiamo usato finora. Ciascuna unità calcola la propria attivazione in funzione dell'input corrente che proviene dalle connessioni feedforward e dell'output all'istante precedente delle unità a cui è collegata da connessioni ricorrenti. I due tipi di input vengono pesati dalle rispettive connessioni sinaptiche e sommati assieme. Tutto quello che si richiede è la registrazione di volta in volta delle attivazioni delle unità che emettono connessioni ricorrenti verso altre unità dello stesso strato o di strati inferiori. Da un punto di vista algoritmico questo si ottiene aggiungendo alla rete un numero di unità di «contesto» la cui attivazione corrisponde all'attivazione precedente delle unità che *emettono* segnali ricorrenti (in pratica i valori vengono copiati); le unità contestuali sono collegate a ciascuna unità che *riceve* segnali ricorrenti (fig. 2.15). A questo punto abbiamo una struttura feedforward che possiamo addestrare con gli algoritmi descritti nelle sezioni precedenti. Malgrado il fatto che le unità contestuali propaghino nella rete solo l'attivazione dei nodi allo stato precedente, la rete neurale è in grado di cogliere informazione temporale che va ben oltre il solo istante precedente codificando in modo ricorsivo questa informazione sulle unità interne.

Anche se questo semplice metodo permette di utilizzare qualsiasi tipo di architettura, si è soliti distinguere almeno due tipi di architetture ricorrenti. La rete di Elman [Elman 1990] presenta solamente uno strato di connessioni ricorrenti sulle unità interne (fig. 2.15a). In pratica questo corrisponde ad aggiungere allo strato di input un numero di unità contestuali c_i (uguale al numero di unità interne h_i) la cui attivazione è uguale a quella dell'unità interna corrispondente all'istante precedente

$$c_i^t = h_i^{t-1}$$

Queste unità aggiuntive trasmettono la loro attivazione alle unità nascoste attraverso normali connessioni sinaptiche

$$h_i^t = \Phi\left(\sum_{j=0} w_{ij}^t x_j^t + \sum_{k=1} w_{ik}^t c_k^t \right)$$

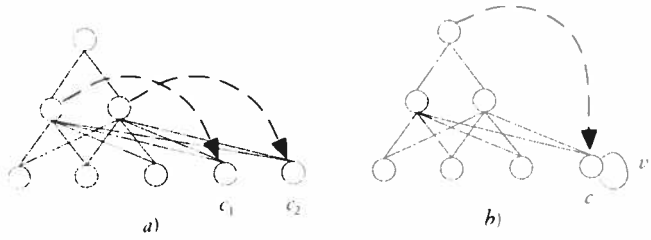


FIG. 2.15. Due tipi di reti parzialmente ricorrenti. *a*) rete di Elman; *b*) rete di Jordan. Le linee tratteggiate indicano la copia del valore di attivazione.

le quali possono essere addestrate con l'algoritmo di Back-propagation in quanto sono equivalenti alle altre unità di input. Data la presenza di connessioni ricorrenti, queste reti esibiscono dinamiche temporali, ovvero l'attivazione delle unità cambia anche se viene mantenuto lo stesso pattern di input. Una rete di Elman è in grado di funzionare in due modalità diverse: durante il calcolo ripetuto delle attivazioni delle unità in alcune situazioni essa passa ciclicamente attraverso uno stesso stato interno ogni t cicli di attivazione, mentre in altre situazioni questo non avviene e la rete tende a procedere verso uno stato finale, definito «attrattore». Nel primo caso la rete è in grado di generare una sequenza ciclica di risposte per ogni vettore di input, mentre nel secondo caso essa genera una sequenza aciclica o una sequenza finita fino al raggiungimento dell'attrattore. Le reti di Elman possono essere impiegate per la predizione delle traiettorie di oggetti sottoposti a diversi tipi di movimenti [Floreano 1993b; Floreano *et al.* 1993], per il riconoscimento e generazione di sequenze linguistiche e per la predizione del comportamento di sistemi dinamici in generale (ad esempio l'andamento di mercato finanziario).

Nelle reti di Jordan invece le unità contestuali sono una copia dell'attivazione precedente delle unità di output o_i^{t-1} e possiedono anche un'autoconnessione v_i (fig. 2.15b) [Jordan 1989]

$$c_i^t = v_i c_i^{t-1} + o_i^{t-1}$$

Non vi sono confronti rigorosi fra le prestazioni delle reti di Elman e quelle di Jordan, ma esse sembrano essere equivalenti nella maggior parte dei casi; si noti comunque che mentre nell'architettura di Jordan le unità nascoste ricevono informazione «diretta» dall'output precedente, nell'architettura di Elman l'informazione trasmessa è il vettore di attivazione delle unità interne, ovvero una *rappresentazione* della trasformazione da input ad output.

4.10.2. *Reti con ritardo temporale*

Un modo più semplice di riconoscere – ma non di produrre – sequenze temporali consiste nel trasformare la dimensione temporale in dimensione spaziale, ovvero nel moltiplicare il numero di unità di input in modo da coprire l'intervallo temporale in cui il segnale viene campionato a istanti discreti (fig. 2.16).

Questa soluzione è analoga alla «finestra» di NETtalk (si veda il paragrafo 4.7.3 e la fig. 2.13) ove il segnale scorre lungo il vettore di input. La stessa strategia può essere adottata anche per le unità interne i cui valori di attivazione entro un determinato intervallo temporale vengono trasmessi attraverso pesi sinaptici diversi fino alle unità di output. Questi tipi di reti neurali vengono spesso definite TDNN (*Time-Delay Neural Network*). Esse richiedono la conoscenza della lunghezza massima della sequenza temporale per la scelta del numero adeguato di unità di input e richiedono anche un gran numero di unità e pesi sinaptici da addestrare; tuttavia, quando questi requisiti non rappresentano una difficoltà, le TDNN mantengono tutte le caratteristiche di efficienza e robustezza proprie dell'algoritmo di Back-propagation tradizionale.

4.10.3. *Back-propagation nel tempo*

Rumelhart, Hinton e Williams [Rumelhart, McClelland e Gruppo PDP 1986, cap. 8; trad. it. 1991] hanno mostrato che per qualsiasi rete ricorrente è possibile costruire una rete feedforward equivalente i cui pesi sinaptici apprendono in base a una lieve variazione dell'algoritmo di Back-propagation. L'attivazione di un'unità nel modello ricorrente è data da

$$y_i^{t+1} = \Phi\left(\sum_{j=0} w_{ij}^t y_j^t + x_i^t \right)$$

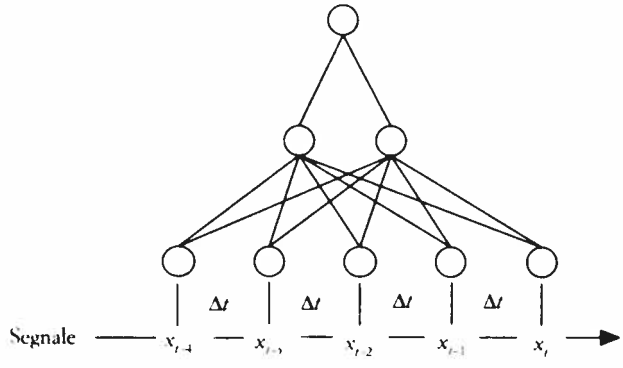


FIG. 2.16. Rete TDNN con ritardo temporale sullo strato di input.

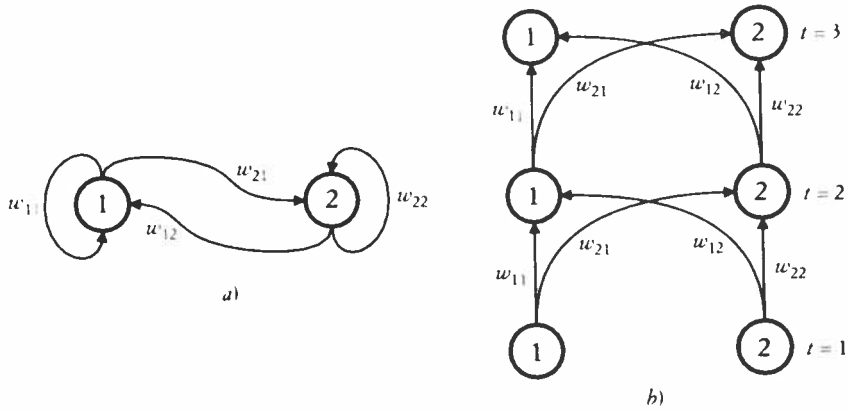


FIG. 2.17. Equivalenza tra una rete ricorrente e una rete feedforward multistrato per una sequenza temporale $T = 3$.

dove x_i^t è l'input netto al nodo i proveniente dall'esterno ($x_i^t = 0$ se non vi è input esterno all'istante t). Immaginiamo di voler addestrare una rete neurale totalmente ricorrente con due unità; se stabiliamo una sequenza temporale T a cui siamo interessati (per la quale sappiamo quale dovrebbe essere lo stato finale delle due unità, ovvero la risposta desiderata), la rete è equivalente a una rete multistrato feedforward con T strati di unità in cui ciascuno strato è una replica delle due unità e i pesi sinaptici sono uguali per tutti gli strati (fig. 2.17); la rete feedforward può essere addestrata con il normale algoritmo di Back-propagation.

L'input viene propagato attraverso la rete feedforward fino alle unità di output e l'uguaglianza dei pesi sinaptici fa sì che il loro stato di attivazione corrisponda all'attivazione delle unità della rete ricorrente equivalente dopo T istanti. La risposta ottenuta viene confrontata con la risposta desiderata e l'errore viene propagato all'indietro per calcolare le modifiche sinaptiche Δw_{ij}^t di ciascuno strato t. Siccome i pesi sinaptici devono essere uguali per ogni strato, la modifica effettivamente applicata è la somma delle modifiche calcolate per ciascuno strato t

$$\Delta w_{ij} = \sum_t w_{ij}^t$$

Una volta che la rete neurale feedforward è stata addestrata, i pesi sinaptici vengono trasferiti nella rete ricorrente la quale non è sottoposta alle limitazioni della sequenza temporale finita T. Questo approccio viene definito con il nome di «Back-propagation nel tempo» (*Back-propagation-through-time*). Esso presenta lo svantaggio di richiedere molte risorse computazionali (per la duplicazione delle unità e dei pesi) e di limitare la ricchezza della dinamica temporale a una sequenza prefissata durante la fase di apprendimento.

4.10.4. *Reti ricorrenti in tempo reale*

Williams e Zipser [1989] hanno proposto un algoritmo che applica il metodo di discesa del gradiente a una rete neurale totalmente ricorrente in tempo reale. Si consideri una rete neurale con N nodi o_i ed M recettori di input x_j. La figura 2.18 illustra un esempio in cui tutti i quattro nodi sono totalmente connessi fra di loro (per un totale di N^2 connessioni, comprese le autoconnessioni) e i recettori proiettano connessioni solamente verso alcuni di essi; infine, possediamo la risposta desiderata per un solo nodo della rete.

Definiamo l'insieme $\mathfrak{R}$ per indicare i recettori esterni, l'insieme $\aleph$ per indicare i nodi della rete e l'insieme $\Im_i$ per indicare i nodi della rete, per cui possediamo una risposta desiderata a un dato istante t. L'attivazione di un nodo è data da una funzione non lineare (ad esempio la funzione sigmoide) della somma pesata di tutti gli input che esso riceve

$$o_i(t+1) = \Phi(A_i(t)) = \Phi\left(\sum_{j \in \mathfrak{R} \cup \aleph} w_{ij}(t) n_j(t)\right)$$

dove

$$n_j(t) = \begin{cases} x_j(t) & \text{se } j \in \mathfrak{R} \\ o_j(t) & \text{se } j \in \aleph \end{cases}$$

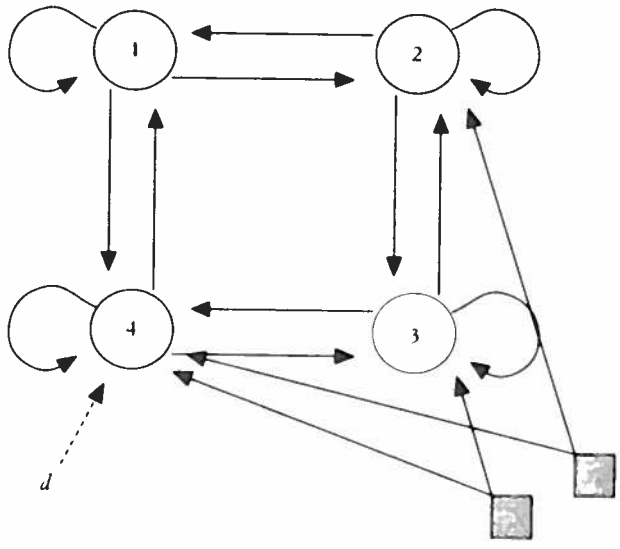


FIG. 2.18. Rete completamente ricorrente. Ciascun nodo può svolgere il ruolo di unità di input, interna o di output. In questo esempio i segnali di input esterno (quadratini scuri) raggiungono l'unità 2, 3 e 4; la risposta desiderata si applica, quando presente, all'unità 4; il nodo 1 svolge la sola funzione di unità interna.

l'errore di un nodo a un dato istante t è dato da

$$e_i(t) = \begin{cases} d_i(t) - o_i(t) & \text{se } i \in \Im_t \\ 0 & \text{altrimenti} \end{cases}$$

L'errore totale della rete a un dato istante t è dato quindi dalla somma dei singoli errori sulle unità per cui è disponibile la risposta desiderata all'istante t

$$E_W(t) = \frac{1}{2} \sum_{i \in \Im_t} e_i^2(t)$$

noi siamo interessati a ridurre l'errore totale sull'intera sequenza di istanti t, ovvero

$$\mathbf{E}_W = \sum_t E_W(t)$$

Il metodo esatto per individuare i pesi sinaptici richiederebbe il calcolo delle derivate dell'errore totale rispetto a tutti i pesi sinaptici e la loro dipendenza rispetto a tutti gli istanti precedenti, ovvero una matrice di gradienti per ciascun istante temporale. Il metodo proposto da Williams e Zipser permette invece di calcolare immediatamente (in tempo reale) le modifiche sinaptiche mediante una stima del gradiente esatto, ovvero utilizzando semplicemente la matrice di derivate della funzione di errore all'istante t rispetto ai pesi sinaptici della rete; la modifica di un qualsiasi peso sinaptico w_{kl} all'istante t è dato da

$$\begin{aligned} \Delta w_{kl}(t) &= -\eta \frac{\partial E_W(t)}{\partial w_{kl}} \\ &= \eta \sum_{i \in \Im_t} e_i(t) \frac{\partial o_i(t)}{\partial w_{kl}} \end{aligned}$$

Definiamo ora una variabile π_{kl}^i a tre indici per indicare l'insieme delle derivate dell'attivazione di ciascun nodo ($i \in \aleph$) rispetto a tutti i pesi w_{kl} della rete all'istante t

$$\pi_{kl}^i(t) = \frac{\partial o_i(t)}{\partial w_{kl}}$$

La differenziazione nel tempo della dinamica della rete indica che i termini π_{kl}^i sono ottenibili in modo ricorsivo risolvendo la seguente equazione (si veda Williams e Zipser [1989] per il procedimento completo di differenziazione)

$$\pi_{kl}^i(t+1) = \dot{\Phi}(A_i(t)) \left[\delta_{ik} n_l(t) + \sum_{j \in \Im_t} w_{ij}(t) \pi_{kl}^j(t) \right]$$

dove δ_{ik} è il «delta di Kronecker» per cui

$$\delta_{ik} = \begin{cases} 1 & \text{se } i = k \\ 0 & \text{altrimenti} \end{cases}$$

inoltre, se si utilizza la funzione di attivazione sigmoide con $\beta = 1$, la derivata prima è facilmente data da

$$\dot{\Phi}(A_i(t)) = o_i(t + 1)[1 - o_i(t + 1)]$$

Si assume che per le condizioni iniziali

$$\pi^i_{kl}(0) = 0$$

Questo procedimento corrisponde alla creazione di una matrice di π^i_{kl} per ciascun nodo i della rete (nell'esempio qui illustrato abbiamo 4 matrici), come indicato in figura 2.19; in ciascuna casella di una matrice abbiamo il π^i_{kl} per il corrispondente peso sinaptico: le righe indicano le unità postsinaptiche k ($k \in \aleph$) e le colonne le unità presinaptiche l ($l \in \aleph \cup \Re$). Se una connessione sinaptica non esiste, l'entrata corrispondente della matrice rimane vuota.

All'inizio ($t = 0$), tutti i π^i_{kl} vengono messi a zero e i loro valori vengono ricalcolati ad ogni istante utilizzando i valori dell'istante precedente. All'istante $t = 1$, dopo aver calcolato le attivazioni delle unità e gli errori in seguito all'input precedente, i π^i_{kl} della riga di ciascuna matrice corrispondente a ciascun nodo (la prima riga della prima matrice, la seconda riga della seconda matrice, ecc.) diventano non-zero perché i delta di Kronecker corrispondenti possiedono valore 1; al passo seguente ciascuna matrice si riempie di valori non-zero. Dopo aver calcolato i valori di π^i_{kl} ad un dato istante t, è possibile calcolare la modifica sinaptica del peso corrispondente in base a

$$\Delta w_{kl}(t) = \eta \sum_{i \in \Im} e_i(t) \pi^i_{kl}(t)$$

e addizionarla subito ai pesi sinaptici corrispondenti.

L'algoritmo richiede la memorizzazione delle matrici di π^i_{kl}, ma questi valori possono essere calcolati velocemente in base al loro valore precedente e non necessitano di ulteriore informazione temporale. Anche se la procedura non è locale (quindi non biologicamente plausibile), è possibile programmare l'algoritmo direttamente su un calcolatore parallelo. L'algoritmo è molto sensibile al valore del tasso di apprendimento η (che conviene mantenere molto piccolo) perché la procedura è solo una stima istante per istante del vero gradiente della funzione di errore totale. Un tipico problema dell'algoritmo è la tendenza di alcune unità a precipitare verso un minimo locale per cui esse emettono una risposta lontana dal valore desiderato (seguono una

1 / l, k	o_0	o_1	o_2	o_3	o_4	x_1	x_2
o_1	π^1_{10}	π^1_{11}	π^1_{12}	π^1_{13}	π^1_{14}		
o_2	π^1_{20}	π^1_{21}	π^1_{22}	π^1_{23}	π^1_{24}	π^1_{25}	π^1_{26}
o_3	π^1_{30}	π^1_{31}	π^1_{32}	π^1_{33}	π^1_{34}	π^1_{35}	π^1_{36}
o_4	π^1_{40}	π^1_{41}	π^1_{42}	π^1_{43}	π^1_{44}	π^1_{45}	π^1_{46}

2 / l, k	o_0	o_1	o_2	o_3	o_4	x_1	x_2
o_1	π^2_{10}	π^2_{11}	π^2_{12}	π^2_{13}	π^2_{14}		
o_2	π^2_{20}	π^2_{21}	π^2_{22}	π^2_{23}	π^2_{24}	π^2_{25}	π^2_{26}
o_3	π^2_{30}	π^2_{31}	π^2_{32}	π^2_{33}	π^2_{34}	π^2_{35}	π^2_{36}
o_4	π^2_{40}	π^2_{41}	π^2_{42}	π^2_{43}	π^2_{44}	π^2_{45}	π^2_{46}

3 / l, k	o_0	o_1	o_2	o_3	o_4	x_1	x_2
o_1	π^3_{10}	π^3_{11}	π^3_{12}	π^3_{13}	π^3_{14}		
o_2	π^3_{20}	π^3_{21}	π^3_{22}	π^3_{23}	π^3_{24}	π^3_{25}	π^3_{26}
o_3	π^3_{30}	π^3_{31}	π^3_{32}	π^3_{33}	π^3_{34}	π^3_{35}	π^3_{36}
o_4	π^3_{40}	π^3_{41}	π^3_{42}	π^3_{43}	π^3_{44}	π^3_{45}	π^3_{46}

4 / l, k	o_0	o_1	o_2	o_3	o_4	x_1	x_2
o_1	π^4_{10}	π^4_{11}	π^4_{12}	π^4_{13}	π^4_{14}		
o_2	π^4_{20}	π^4_{21}	π^4_{22}	π^4_{23}	π^4_{24}	π^4_{25}	π^4_{26}
o_3	π^4_{30}	π^4_{31}	π^4_{32}	π^4_{33}	π^4_{34}	π^4_{35}	π^4_{36}
o_4	π^4_{40}	π^4_{41}	π^4_{42}	π^4_{43}	π^4_{44}	π^4_{45}	π^4_{46}

FIG. 2.19. Tabelle di derivate dell'attivazione di ciascuna delle quattro unità rispetto a tutti i pesi sinaptici della rete neurale. Ciascuna tabella corrisponde a un'unità della rete (si veda il testo).

traiettoria diversa). Per rimediare a questa difficoltà, Williams e Zipser propongono di sostituire il valore desiderato all'output delle unità, ma solo dopo aver calcolato l'errore e i π^k_{ij}; questo metodo – detto «imposizione dell'insegnamento» – mantiene la rete sulla traiettoria individuata dalla risposta desiderata e accelera l'apprendimento. Il pericolo è che il minimo («attrattore») raggiunto con questo metodo non rimanga tale quando la risposta desiderata non è più disponibile perché il metodo di imposizione dell'insegnamento non permette alla rete di distinguere i punti di stabilità dovuti a minimi reali oppure a zone di «repulsione» (ad esempio la cima di un massimo locale).

Questo algoritmo presenta interessanti caratteristiche di generalità: non richiede la duplicazione delle unità, è in grado di operare su sequenze temporali a lunghezza variabile e imprevedibile, può essere utilizzato per riconoscere e generare sequenze temporali, si applica a qualsiasi architettura e funziona in tempo reale.

Reti ricorrenti in tempo reale: algoritmo

Si consideri una rete neurale ricorrente con un numero arbitrario di nodi e unità di input esterne collegati fra di loro in modo arbitrario (per le convenzioni simboliche si veda il paragrafo precedente).

1. Inizializzare i pesi sinaptici (*bias* incluso) a piccoli valori casuali

$$w_{kl} = rnd(\pm 0{,}1)$$

inizializzare le derivate parziali dell'attivazione di ciascun nodo rispetto a tutti i pesi della rete a zero

$$\pi^k_{ij}(0) = 0$$

2. Ad ogni istante t eseguire i passi 3-6.
3. Presentare il vettore di input $\mathbf{x}(t)$ e calcolare l'attivazione di tutti i nodi

$$o_i(t+1) = \Phi(A_i(t)) = \Phi\left(\sum_{i \in \Re \cup \aleph} w_{ij}(t) n_j(t)\right)$$

dove

$$n_j(t) = \begin{cases} x_j(t) & \text{se } j \in \Re \\ o_j(t) & \text{se } j \in \aleph \end{cases}$$

4. Calcolare le nuove matrici di derivate parziali (fig. 2.19) per ciascun nodo della rete sfruttando il valore precedente

$$\pi^k_{ij}(t+1) = \dot{\Phi}(A_i(t))\left[\delta_{ik} n_j(t) + \sum_{l \in \Im_t} w_{il}(t) \pi^k_{lj}(t)\right]$$

5. Usare le derivate così ottenute per calcolare la modifica sinaptica

$$\Delta w_{kl}(t) = \eta \sum_{i \in \mathfrak{I}_t} e_i(t) \pi'_{kl}(t)$$

dove l'errore $e_i(t)$ è dato da

$$e_i(t) = \begin{cases} d_i(t) = o_i(t) & \text{se } i \in \mathfrak{I}_t \\ 0 & \text{altrimenti} \end{cases}$$

6. Applicare le modifiche sinaptiche alle connessioni della rete

$$w_{kl}(t+1) = w_{kl}(t) + \Delta w_{kl}(t)$$

4.11. Modelli interni dell'ambiente

Il paradigma supervisionato esaminato nelle sezioni precedenti richiede la presenza costante di un insegnante che provveda la risposta desiderata per ciascun pattern di input durante la fase di addestramento. La risposta desiderata è un vettore che contiene i valori esatti che ciascuna unità di output deve assumere. Tuttavia questo tipo di informazione non è sempre disponibile per un organismo vivente: ad esempio, nel caso di un bambino che apprende a riprodurre dei suoni linguistici non vi è un insegnante che dica esattamente dove collocare la lingua, come aprire la bocca, ecc. Anche se vi fosse, l'insegnante non conoscerebbe esattamente le azioni fonoarticolatorie che il bambino deve eseguire e, quand'anche le conoscesse, non potrebbe comunque comunicargliele nello stesso sistema di coordinate motorie. Tutto quello che il bambino possiede è il suono che desidera imitare e le conseguenze sensoriali delle proprie azioni (il suono emesso e la percezione della reazione dell'insegnante). Lo stesso problema si presenta per l'apprendimento della maggior parte delle azioni motorie (apprendere a centrare un canestro, andare in bicicletta, usare gli stecchini cinesi, ecc.), ove l'unica informazione disponibile è la conseguenza sensoriale delle azioni. Una possibile soluzione consiste nel trasformare le conseguenze delle proprie azioni in un indice di valutazione della bontà delle azioni e usare questo indice per modificare la probabilità di ripetizione delle azioni corrispondenti. Questo principio sta alla base dell'apprendimento per rinforzo che esamineremo nella sezione successiva. Tuttavia ridurre l'informazione sensoriale sulle conseguenze delle proprie azioni ad un unico segnale scalare (un singolo numero) non è una strategia molto efficiente.

Jordan e Rumelhart [1992] hanno riformulato il problema dell'insegnante esterno in modo da coprire anche le situazioni in cui la risposta desiderata è data solo per le conseguenze sensoriali delle pro-

prie azioni e non per le azioni stesse. In quest'ottica ogni paradigma di apprendimento supervisionato nella sua forma più generale si compone di due fasi integrate. Nella prima fase l'agente apprende la trasformazione da azioni (qualsiasi azione) in conseguenze sensoriali, ovvero forma un modello interno dell'ambiente attraverso un normale processo di apprendimento supervisionato. L'apprendimento del modello interno permette all'agente di predire le conseguenze sensoriali delle proprie azioni. Nella seconda fase – quella in cui l'agente deve apprendere a produrre delle azioni appropriate in funzione dell'informazione sensoriale attuale e di una determinata finalità – il modello interno già appreso può essere utilizzato per trasformare gli errori distali (a livello delle sensazioni) in errori prossimali (a livello delle azioni) in almeno tre modi.

L'architettura e il funzionamento del modello neurale sono molto semplici. L'architettura è composta di due moduli (fig. 2.20). Un modulo apprende la trasformazione da azioni motorie (input) a conseguenze sensoriali (output) ovvero a predire lo stato sensoriale $\mathbf{s}_p$, formando così un modello interno («diretto») dell'interazione con l'ambiente. L'agente esegue delle azioni casuali e osserva le conseguenze delle proprie azioni: le conseguenze sensoriali effettive $\mathbf{s}_r$ vengono confrontate con le conseguenze sensoriali predette $\mathbf{s}_p$ dalla rete neurale e l'errore viene usato per correggere i pesi sinaptici. Una volta che questo modulo ha appreso a predire le conseguenze delle proprie azioni, l'agente può sfruttare questa conoscenza per apprendere ad eseguire le azioni appropriate al raggiungimento di un determinato scopo mediante il modulo dell'azione. Il compito di questo modulo consiste nell'apprendere la trasformazione da stati sensoriali desiderati

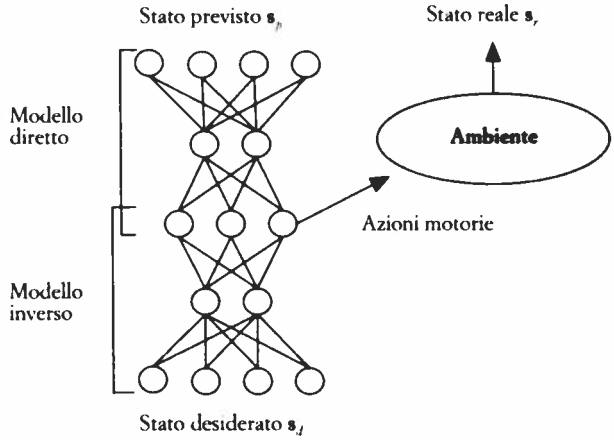


FIG. 2.20. Apprendimento con un insegnante distale. Inizialmente il modello diretto viene addestrato a fornire predizioni sensoriali di azioni casuali; successivamente il modello inverso viene addestrato a fornire azioni appropriate utilizzando il vettore di errore sensoriale propagato all'indietro fino all'unità di azioni motorie.

(input) ad azioni motorie appropriate (output), tali cioè da causare gli stati sensoriali desiderati: esso deve quindi sviluppare un modello «inverso» dell'ambiente. Inizialmente, dato un certo stato sensoriale desiderato, il modulo dell'azione non è in grado di fornire azioni corrette e non possiede un vettore di risposta desiderata; tuttavia il modello interno già appreso può essere usato per ricevere in ingresso le azioni proposte e generare una previsione delle conseguenze sensoriali di tali azioni. L'errore tra le conseguenze sensoriali previste $\mathbf{s}_p$ e le conseguenze sensoriali desiderate $\mathbf{s}_d$ viene quindi propagato all'indietro attraverso i vari strati del modello interno (senza modificarne i valori sinaptici) fino alle unità di azione motoria; a questo punto il vettore di errore così ottenuto può essere usato per correggere i pesi sinaptici del modulo di azione.

L'intera procedura si basa su tre versioni del vettore di informazione sensoriale: lo stato sensoriale desiderato $\mathbf{s}_d$, le conseguenze sensoriali reali $\mathbf{s}_r$ e le conseguenze sensoriali predette dal modello interno $\mathbf{s}_p$. Esse danno luogo a tre possibili tipi di errore. L'*errore di prestazioni* ($\mathbf{s}_d - \mathbf{s}_r$) indica la discrepanza tra lo stato desiderato e lo stato effettivamente ottenuto dall'azione motoria; l'*errore di predizione* ($\mathbf{s}_r - \mathbf{s}_p$) indica la discrepanza tra le conseguenze reali di un'azione e le conseguenze previste dal modello interno; infine, l'*errore di prestazioni previste* ($\mathbf{s}_d - \mathbf{s}_p$) indica la discrepanza tra lo stato desiderato e lo stato previsto dal modello interno. L'apprendimento del modello interno diretto avviene utilizzando l'errore di predizione. L'apprendimento del modello inverso può basarsi invece sia sull'errore di prestazioni che sull'errore di prestazioni previste: nel primo caso l'apprendimento è più rapido e il modello inverso appreso è esatto anche se il modello diretto è solo approssimativo. La scelta del tipo di errore impiegato dipende comunque dalle situazioni: nel caso in cui l'agente non sia in grado di verificare direttamente le conseguenze di tutte le proprie azioni (ad esempio quando si usa un simulatore impreciso o limitato dell'ambiente per cui non è possibile avere le conseguenze sensoriali di tutte le possibili azioni), è utile addestrare il modello inverso usando l'errore di prestazioni previste. Questa procedura è simile a un processo di «pianificazione» o «pratica mentale» e si basa fortemente sulle capacità di generalizzazione del modello interno diretto: se questo non è esatto, non è possibile assicurarsi in principio l'apprendimento di un modello inverso esatto.

Jordan e Rumelhart hanno applicato questo modello neurale a diversi compiti di controllo di un braccio meccanico a tre giunti e hanno mostrato che se viene utilizzato l'errore di prestazioni per l'apprendimento del modello inverso non è sempre necessario addestrare la rete neurale in due fasi distinte: infatti l'iniziale instabilità del modulo di predizione si riflette sul modulo di azione solo attraverso la distorsione del vettore di errore calcolato tra lo stato desiderato e lo stato effettivamente raggiunto e non attraverso una misura di errore originariamente scorretta. Gli autori hanno anche proposto diverse architet-

ture che comprendono l'aggiunta di unità contestuali [si veda anche Miyata 1988] – sia per il modello inverso che per il modello diretto – e di unità di stato corrente (che forniscono informazione sullo stato attuale dell'ambiente).

L'assunzione che l'apprendimento di comportamenti motori finalizzati si basi sullo sviluppo di un modello interno dell'ambiente fornisce spunti interessanti per la modellizzazione in psicologia. In linea con quanto teorizzato da Piaget, le fasi iniziali dello sviluppo della coordinazione sensomotoria corrisponderebbero all'apprendere a predire le conseguenze sensoriali delle proprie azioni. Il paradigma computazionale prevede che il ricco comportamento esploratorio (generazione di azioni casuali diverse) esibito dai bambini piccoli sia essenziale per lo sviluppo di un modello interno accurato su cui si basa lo sviluppo graduale della capacità di eseguire azioni finalizzate[18].

5. Apprendimento per rinforzo

Vi sono alcune situazioni in cui l'informazione che proviene dall'ambiente consiste solamente in un segnale (un singolo numero) che indica la bontà di un'azione o di una sequenza di azioni. In linea con la metodologia utilizzata per gli studi sull'apprendimento animale, il segnale di feedback che proviene dall'ambiente viene interpretato come un *rinforzo* positivo o negativo della risposta della rete neurale. I modelli di apprendimento per rinforzo hanno una lunga tradizione che trascende la specifica implementazione connessionista; essa si fa sentire soprattutto nei tipici domini di applicazione di questo paradigma connessionista e nella terminologia che viene utilizzata. Le prime idee sul ruolo del rinforzo per lo sviluppo del comportamento si basavano sulle osservazioni degli psicologi dell'inizio del secolo [ad esempio Thorndike 1911] e furono poi formalizzate, ulteriormente sviluppate e applicate in campo ingegneristico per i sistemi di controllo e in Intelligenza Artificiale [ad esempio Samuel 1959].

L'apprendimento per rinforzo è una forma di apprendimento supervisionato perché vi è un segnale che proviene dall'ambiente, ma questo segnale non fornisce un'informazione di gradiente che può essere usata dal sistema per individuare la risposta corretta; colui il quale genera il segnale (tipicamente l'ambiente stesso in cui il sistema opera) non indica qual è la risposta corretta, come nel caso dell'insegnante nei paradigmi supervisionati descritti nelle sezioni precedenti, ma fornisce solamente una *critica* della risposta del sistema (questo paradigma di apprendimento viene talvolta definito come «apprendimento con un critico»). Questo significa che il sistema deve essere in grado

[18] Si veda Floreano e Parisi [1992] e Parisi, Pagliarini e Floreano [1994] per una trattazione più dettagliata e un'applicazione allo studio delle fasi iniziali dello sviluppo linguistico nei bambini.

di generare molte risposte casuali ed osservare il valore di rinforzo che il critico (l'ambiente) fornisce; l'apprendimento consiste nel modificare i parametri interni del sistema in modo tale da aumentare la probabilità di emettere risposte che ricevono rinforzi positivi e di diminuire la probabilità di emettere risposte che ricevono rinforzo negativo. La generazione di risposte casuali può essere vista come un processo di esplorazione e svolge un ruolo centrale; quando lo spazio delle possibili coppie di input-output è troppo grande per essere accuratamente esplorato, il sistema deve essere in grado di indovinare sulla base dell'esperienza precedente quale sia la risposta più appropriata: questa situazione (comune a tutti i problemi reali) è definita come il «problema dell'assegnazione di credito strutturale».

Nell'implementazione connessionista la rete neurale tipicamente costituisce il sistema di controllo di un agente che opera in un ambiente. L'input della rete rappresenta l'informazione sensoriale e l'output costituisce la probabilità che l'agente emetta una certa azione la cui esecuzione genera un segnale di rinforzo.

Vi sono due classi generali di ambienti in cui l'apprendimento per rinforzo viene utilizzato:

1. ambienti statici (o stocastici) in cui per ogni coppia di input-output vi è un unico valore di rinforzo (oppure una determinata probabilità di ottenere un rinforzo positivo);
2. ambienti dinamici in cui la sequenza temporale influenza l'interazione tra agente e ambiente, modifica il valore del rinforzo e altera la frequenza con cui esso è reso disponibile all'agente.

5.1. Ambienti statici: metodo di «Associative Reward-Penalty»

L'algoritmo *Associative Reward-Penalty* (Ricompensa-Penalizzazione associativa) ARP [Barto e Anandan 1985] è applicabile in situazioni statiche (e stocastiche) in cui ad ogni coppia di input-output corrisponde sempre (o con una certa probabilità) un determinato valore di rinforzo (+1 o −1). L'attivazione di ciascuna unità di output della rete rappresenta la *probabilità* di emettere un'azione. Le unità ad attivazione probabilistica possono assumere uno di due possibili stati (ad esempio {−1, +1} o {0, 1}); la probabilità $P(s_i = \pm 1)$ che l'unità i-esima assuma lo stato 1 o −1 è data dalla funzione

$$P(s_i = \pm 1) = \frac{1}{1 + e^{\pm 2\beta A_i}}$$

dove

$$A_i = \sum_j w_{ij} x_j$$

è la somma pesata degli input dell'unità. Siccome la determinazione

della risposta s_i richiede un processo stocastico, queste unità vengono spesso definite «stocastiche»[19].

L'algoritmo ARP costruisce la risposta desiderata d_i^μ per ciascun pattern μ osservando il segnale di rinforzo r_i^μ che proviene dall'ambiente; se il rinforzo è positivo, la risposta desiderata è la risposta dell'unità stessa s_i^μ, altrimenti è l'opposto della risposta

$$d_i^\mu = \begin{cases} s_i^\mu & \text{se } r_i^\mu = +1 \\ -s_i^\mu & \text{se } r_i^\mu = -1 \end{cases}$$

La correzione dei pesi sinaptici usa il metodo di discesa del gradiente sulla superficie dell'errore (ovvero la regola delta) generato dal confronto tra la risposta desiderata e la risposta media dell'unità $\langle s_i^\mu \rangle$

$$\delta_i^\mu = d_i^\mu - \langle s_i^\mu \rangle$$

Il valore medio della risposta dovrebbe essere calcolato effettuando parecchie simulazioni del calcolo della risposta dell'unità, ma può essere ottenuto più rapidamente con la funzione tangente iperbolica della somma pesata degli input osservando che

$$\begin{aligned} \langle s_i^\mu \rangle &= 1 \cdot P(S_i = +1) + (-1) \cdot P(s_i = -1) \\ &= \frac{1}{1 + e^{-2\beta A_i}} - \frac{1}{1 + e^{2\beta A_i}} \\ &= \frac{e^{\beta A_i} - e^{-\beta A_i}}{e^{\beta A_i} + e^{-\beta A_i}} \\ &= \tanh(\beta A_i) \end{aligned}$$

La funzione tangente iperbolica ha lo stesso tipo di andamento della funzione sigmoide, ma gli asintoti si trovano a +1 e −1.

Il tasso di apprendimento viene scelto in base al valore del rinforzo r_i^μ: se il rinforzo è positivo, η è molto più grande (uno o due ordini di grandezza) per premiare le risposte corrette: la scelta esatta dei valori dipende comunque dal tipo di compito che la rete deve apprendere. La regola di modifica dei pesi sinaptici è data quindi dalla regola delta

$$\Delta w_{ij}^\mu = \eta^\pm \delta_i^\mu x_j^\mu$$

[19] Il risultato può essere ottenuto usando la funzione di generazione di numeri casuali dei linguaggi di programmazione, ma, se il tempo di calcolo non è una priorità, è meglio far ricorso a dispositivi che producono sequenze aleatorie a partire da un processo stocastico naturale, ad esempio una sorgente di rumore. Infatti la maggior parte dei generatori di numeri casuali producono delle sequenze ricorrenti che limitano l'esplorazione della rete neurale ed eventualmente ne compromettono l'apprendimento.

dove $\eta^{\pm}$ è il tasso di apprendimento in funzione del valore del rinforzo. Se la rete possiede più strati o connessioni ricorrenti, vi sono due possibili soluzioni: *a)* si utilizza il metodo di Back-propagation dei δ_i^{μ} per calcolare il contributo delle unità nascoste all'errore finale; *b)* si calcola direttamente l'errore per ciascuna unità nascosta usando lo stesso valore di rinforzo globale. Alcune semplici variazioni sono state proposte per utilizzare valori di rinforzo continui e per accelerare il processo di convergenza [Ackley e Littman 1990; Barto e Jordan 1987].

Una versione lievemente modificata dell'algoritmo ARP è stata utilizzata per apprendere la trasformazione da coordinate retiniche a coordinate cranio-centriche [Mazzoni, Andersen e Jordan 1991] impiegando lo stesso paradigma simulativo già utilizzato da Zipser e Andersen [1988]. I risultati indicano che l'apprendimento per rinforzo produce lo stesso tipo di soluzioni individuate dall'algoritmo di Back-propagation sia per quanto riguarda i tempi di apprendimento che per il confronto qualitativo tra la risposta delle unità interne artificiali e quella dei neuroni biologici. Questo fatto indica che la procedura di propagazione all'indietro degli errori non è necessaria per la simulazione della trasformazione operata dai neuroni biologici. L'algoritmo ARP presenta anche il vantaggio di essere biologicamente più plausibile. Infatti, l'apprendimento per rinforzo non richiede la propagazione all'indietro sulle stesse connessioni dei messaggi di errore (come nel caso di Back-propagation), ma applica l'unico segnale di rinforzo disponibile su ciascuna unità della rete per calcolare una stima dell'errore. Siccome le unità delle reti impiegate nell'apprendimento per rinforzo sono stocastiche e l'errore è solo una stima rispetto all'errore calcolato in Back-propagation, la curva di apprendimento presenta delle oscillazioni e i valori individuali dei pesi finali generati dai due tipi di algoritmi sono diversi, ma rappresentano lo stesso tipo di trasformazione quando vengono considerati globalmente (i pesi della rete addestrata con ARP generano lo stesso risultato quando sono trasferiti nella rete di Back-propagation, la quale impiega unità continue sigmoidi).

5.2. Ambienti dinamici: metodi di «Differenza temporale»

Vi sono molte situazioni in cui un agente deve effettuare una sequenza di azioni prima di ricevere un segnale di rinforzo (ambienti dinamici). In questi ambienti l'agente deve risolvere due tipi di problemi: il «problema dell'assegnamento di credito strutturale» (descritto nel paragrafo 5) e il «problema dell'assegnamento di credito temporale». Quest'ultimo si riferisce al modo in cui il sistema attribuisce meriti (crediti o penalizzazioni) alle singole azioni che lo hanno portato alla situazione di rinforzo. I metodi di «Differenza temporale» (TD-*learning*) rappresentano una classe di algoritmi di ap-

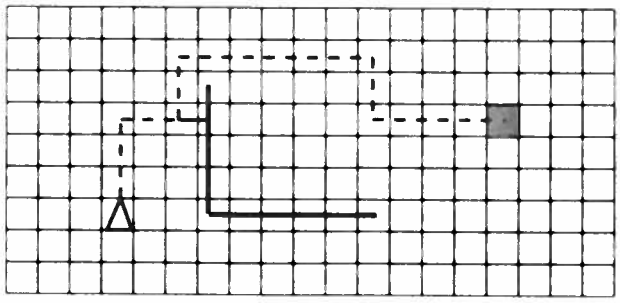


FIG. 2.21. Immagine dall'alto di un ambiente composto da una griglia di caselle. L'agente (rappresentato dal triangolino) deve raggiungere la casella meta (evidenziata in grigio) nel minor tempo possibile. Esso si può muovere di una casella alla volta in una delle quattro direzioni cardinali; se decide di muoversi contro la barriera viene rimbalzato indietro sulla casella precedente. La linea tratteggiata indica la traiettoria generata da una sequenza di azioni successive. L'agente riceve un rinforzo positivo dall'ambiente solamente quando raggiunge la meta.

prendimento per rinforzo che operano su reti neurali in ambienti dinamici[20].

Consideriamo innanzitutto un agente che si sposta in un ambiente discreto (fig. 2.21). Il compito dell'agente consiste nel raggiungere la casella meta nel minor numero possibile di passi. Lo stato dell'agente a un dato istante è dato dall'attività sensoriale. La probabilità che l'agente si sposti dallo stato $\mathbf{x}$ allo stato $\mathbf{y}$ dipende dall'azione a che esso esegue. Assumendo che l'agente impieghi una «politica» (*policy*) stazionaria $a = \pi(\mathbf{x})$, ovvero esegua sempre la stessa azione in risposta allo stato $\mathbf{x}$, la probabilità di transizione dallo stato $\mathbf{x}$ allo stato $\mathbf{y}$, $P_{xy}(\pi(\mathbf{x}))$ è indipendente dagli stati precedenti. Il processo di transizione attraverso una serie di stati successivi rappresenta quindi una «catena di Markov» e la decisione della mossa da eseguire è un «compito di decisione markoviana». Immaginiamo di conoscere il valore della ricompensa $r_{t+1} = R(\mathbf{x}_t, a_t)$ per ogni azione a eseguita nello stato $\mathbf{x}$, ovvero – 1 se lo stato raggiunto non corrisponde alla meta e 0 quando esso coincide invece alla meta. Immaginiamo anche – per semplicità – che l'ambiente sia deterministico: se l'agente decide di effettuare l'azione a, la probabilità $P_{xy}(\pi(\mathbf{x}))$ di spostarsi effettivamente sullo stato $\mathbf{y}$ è 1 e quindi la ricompensa per quella azione è proprio r_{t+1}. A questo punto possiamo definire una funzione di «valutazione» $V^{\pi}(\mathbf{x})$ di un certo stato $\mathbf{x}$ rispetto alle conseguenze future come la somma di tutte le ricompense a partire da quello stato fino all'infinito scontati di un fattore γ

$$V^{\pi}(\mathbf{x}) = \sum_{t=0}^{\infty} \gamma^t r_{t+1}$$

[20] Una trattazione dettagliata molto chiara è data in Barto, Sutton e Watkins [1990] e in Sutton e Barto [1998].

posto che lo stato dell'agente al tempo $t = 0$ sia $\mathbf{x}$ e che l'azione a sia il risultato di una politica stazionaria $\pi(\mathbf{x})$. Se il fattore di sconto γ è prefissato a un valore tale per cui $0 < \gamma < 1$, le ricompense vengono scontate maggiormente in funzione del tempo (ovvero le ricompense più recenti possiedono un peso maggiore rispetto alle ricompense più distanti nel tempo); quando invece $\gamma = 1$ la valutazione dello stato $\mathbf{x}$ è semplicemente la somma di tutte le singole ricompense fino all'infinito. Una volta che l'agente ha raggiunto la meta la ricompensa è sempre zero e quindi la sommatoria produce un risultato finito; altrimenti, nel caso in cui la politica $\pi(\mathbf{x})$ non porti mai alla meta, il risultato della sommatoria approssima $-1/(1 - \gamma)$. Se l'agente è in grado di conoscere la ricompensa per ciascuna azione la sola cosa che deve fare è sviluppare una politica che massimizzi la valutazione di ciascuno stato.

Consideriamo ora il calcolo della valutazione di uno stato rispetto a due successive azioni dell'agente. Inizialmente l'agente esegue un'azione e questa comporta un rinforzo r_{t+1}; quindi, la valutazione $V_1^{\pi}(\mathbf{x})$ di $\mathbf{x}$ rispetto a un passo nel futuro è uguale al merito ricevuto (come risulta dalla sostituzione nell'equazione precedente)

$$V_1^{\pi}(\mathbf{x}) = r_1$$

ora l'agente si trova in un nuovo stato $\mathbf{y}$ rispetto al quale esegue un'altra azione; la valutazione dello stato iniziale dopo questa seconda azione (quindi rispetto a due passi nel futuro) è uguale al merito ricevuto per l'azione precedente sommato alla valutazione (rispetto a un solo passo nel futuro) del nuovo stato scontato del fattore γ

$$V_2^{\pi}(\mathbf{x}) = r_1 + \gamma V_1^{\pi}(\mathbf{y})$$

La formula per il calcolo della valutazione di uno stato a un certo istante ha quindi la seguente struttura ricorsiva:

$$V^{\pi}(\mathbf{x}_t) = r_{t+1} + \gamma V^{\pi}(\mathbf{y})$$

5.2.1. *Il «critico euristico adattivo»*

Nelle situazioni reali non sempre si conoscono i meriti delle singole azioni, né la politica di transizione da uno stato a quello successivo. L'algoritmo AHC (*Adaptive Heuristic Critic*) prevede che l'agente stesso impari a predire la valutazione di ciascuno stato e sulla base di questa predizione impari anche a sviluppare una politica ottimale che lo porta alla meta nel minor numero possibile di passi [Barto, Sutton e Watkins 1990]. L'architettura del sistema di controllo dell'agente è composta da due reti neurali (fig. 2.22): una rete per la predizione della valutazione dello stato – EVAL – e una rete per la generazione delle azioni (i pesi della rete rappresentano la politica dell'agente) – POL.

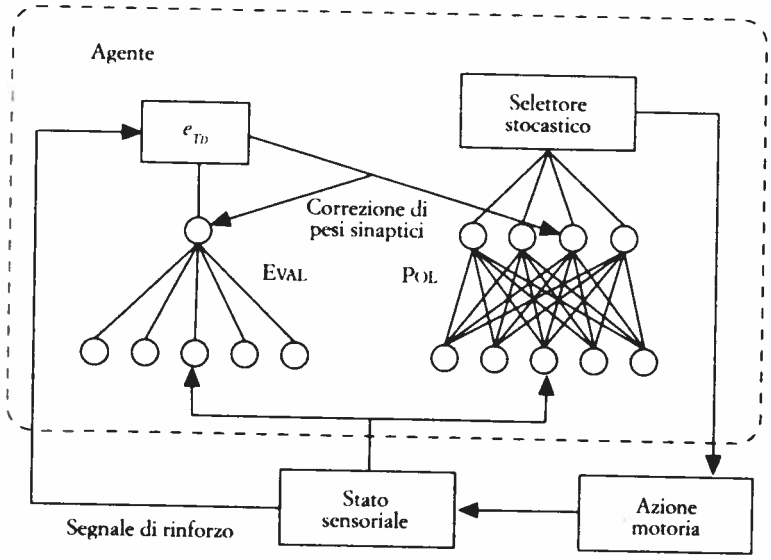


FIG. 2.22. Architettura dell'algoritmo AHC.

Entrambe le reti ricevono input sullo stato dell'ambiente dai sensori dell'agente e impiegano unità ad attivazione continua. Un ulteriore sistema stocastico sceglie l'azione da eseguire fra quelle proposte dalla rete POL.

Ad ogni istante t la rete EVAL predice una valutazione $v(\mathbf{x}_t)$ dello stato attuale; la correzione dei pesi sinaptici necessiterebbe dell'errore tra la valutazione reale $v^{\pi}(\mathbf{x})$ e la valutazione prevista $v(\mathbf{x}_t)$, ma la valutazione reale non è conosciuta. Dalla proprietà ricorsiva della funzione di valutazione possiamo però definire un errore tra la predizione della valutazione del nuovo stato e la predizione della valutazione dello stato precedente

$$e_{TD} = (r_{t+1} + \gamma v(\mathbf{y})) - v(\mathbf{x}_t)$$

dove r_{t+1} diventa nonzero solo quando l'ambiente provvede il segnale di rinforzo. L'errore tra le predizioni a due istanti successivi si chiama errore TD (*Temporal Difference*) e viene usato per correggere i pesi sinaptici con la regola delta rispetto all'input precedente

$$\Delta w_i^{EVAL} = \eta e_{TD} x_i = \eta[(r_{t+1} + \gamma v(\mathbf{y})) - v(\mathbf{x})]x_i$$

si noti che se $\gamma = 0$ (nel caso in cui non si considerano i meriti delle altre azioni nel tempo), la modifica dei pesi sinaptici si riduce alla regola delta

$$\Delta w_i^{EVAL} = \eta[r_{t+1} - v(\mathbf{x})]x_i$$

Questo risultato indica che la regola delta è in effetti un predittore adattivo rispetto a un solo passo nel futuro. Tuttavia se $\gamma > 0$, è difficile mantenere l'analogia tra il comportamento della regola delta e quello del metodo TD (alcuni risultati teorici sono esposti in Sutton [1988]).

La stessa misura di errore viene usata per correggere anche i pesi della rete POL: infatti se l'errore è positivo vuol dire che l'azione a eseguita dalla rete POL è migliore di quanto inizialmente previsto e quindi si rafforzano i valori delle connessioni che hanno generato quell'azione (aumentando così la probabilità che essa venga nuovamente selezionata); altrimenti, se l'errore TD è negativo, si diminuiscono i valori delle connessioni responsabili di tale azione. Ciascuna unità di output della rete POL rappresenta la probabilità di emettere una determinata azione. La regola delta viene dunque applicata *solo ai pesi dell'unità che ha generato l'azione a*

$$\Delta w_{ij}^{POL} = \eta e_{TD} x_j \quad \text{se } i = a$$
$$\Delta w_{ij}^{POL} = 0 \quad \text{altrimenti}$$

Durante l'apprendimento, la probabilità di emettere l'azione a_i corrispondente all'unità i-esima è calcolata da un selettore stocastico

$$P(a_i) = \frac{e^{\frac{o_i}{T}}}{\sum_k e^{\frac{o_k}{T}}}$$

dove o_i è l'attivazione dell'unità i-esima e T è una costante (detta «temperatura») che determina la casualità della selezione (quanto maggiore è T, tanto maggiore è l'incertezza della selezione dell'azione a_i); il denominatore è la sommatoria di tutte le altre unità k dello strato di output e funge da fattore di normalizzazione. La selezione stocastica delle azioni è responsabile del processo di esplorazione che è fondamentale per l'apprendimento di una politica (pesi sinaptici) ottimale permettendo di apprendere le situazioni in cui è necessario che l'agente esegua delle azioni temporaneamente svantaggiose per poter raggiungere la meta finale. Il selettore stocastico normalizza l'attivazione di ciascuna unità della rete (la somma di tutte le probabilità è uguale a 1). Esso funziona nel modo seguente. Immaginiamo di avere una ruota della fortuna truccata con tante caselle quante sono le unità della rete; la larghezza di ciascuna casella è proporzionale al valore attribuito dal selettore stocastico a ciascuna unità. La scelta dell'azione corrisponde al lanciare una pallina e osservare in quale casella cade (nelle simulazioni al computer si estrae un numero casuale da un insieme finito e si osserva l'intervallo a cui esso appartiene, dove ciascun intervallo corrisponde a un'unità e la larghezza dell'intervallo è proporzionale al valore corrispondente del selettore stocastico).

Le architetture delle due reti – EVAL e POL – sono di solito uguali,

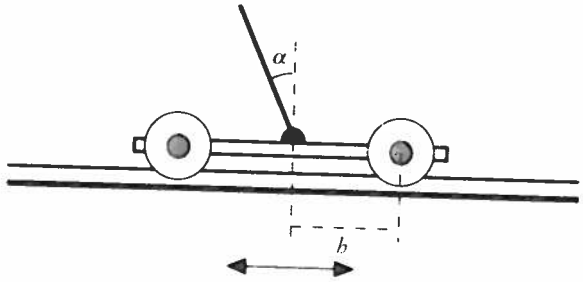


FIG. 2.23. Il problema del pendolo inverso. Un pendolo inverso è fissato con una cerniera a un carrello che si può spostare su una rotaia. La rete neurale deve apprendere a muovere il carrello per mantenere il pendolo in equilibrio senza però spostare il carrello dalla posizione originale oltre una distanza b. L'ambiente emette un rinforzo negativo (−1) se il pendolo supera l'angolo α dalla posizione perpendicolare oppure se il carrello è stato spostato oltre la distanza b, un rinforzo positivo (1) se il pendolo è esattamente in posizione perpendicolare e il carrello si trova entro gli spazi prefissati e rinforzo zero in tutte le altre situazioni. L'algoritmo AHC apprende velocemente a mantenere il pendolo in equilibrio.

anche se questa scelta non è indispensabile. Se si usano delle reti multistrato, l'errore TD viene propagato all'indietro sulle unità interne per calcolare le modifiche dei pesi del primo strato. In questo caso Lin [1992] suggerisce di usare una rete POL per ciascun tipo di azione con un'unica unità di output, invece di un'unica rete con tante unità quante sono le possibili azioni: infatti, la seconda soluzione (architettura monolitica) provocherebbe dei fenomeni di interferenza nella correzione dei pesi dalle unità di input alle unità interne.

Barto, Sutton e Anderson [1983] hanno impiegato una versione leggermente diversa dell'algoritmo AHC per controllare un pendolo inverso in equilibrio su un carrello mobile (fig. 2.23). Il sistema apprende a spostare il carrello per controbilanciare la tendenza a cadere del pendolo inverso e riesce a compensare anche spinte applicate dall'esterno.

AHC: algoritmo

EVAL è costituita da una rete con un'unica unità di output o^{EVAL}; POL è costituita da una rete ove ciascuna unità di output o_i^{POL} rappresenta la probabilità di emettere l'azione corrispondente a_i. Le unità di entrambe le reti sono continue (non-lineari se le reti sono multistrato).

1. Inizializzare i pesi sinaptici di EVAL e POL a piccoli valori casuali. Selezionare i valori del tasso di apprendimento η, del fattore di sconto γ e della temperatura T (ad esempio 0,1; 0,5 e 1,0 rispettivamente).
2. Usare il vettore di stato corrente $\mathbf{x}$ come input di ciascuna delle reti.
3. Propagare l'attivazione attraverso le connessioni della rete EVAL per ottenere la valutazione prevista dello stato $\mathbf{x}$ dall'output della rete

$$v(\mathbf{x}) = o^{EVAL}$$

4. Propagare l'attivazione attraverso le connessioni della rete POL e selezionare l'azione da eseguire usando il selettore stocastico

$$P(a_i) = \frac{e^{\frac{o_i^{POL}}{T}}}{\sum_k e^{\frac{o_k^{POL}}{T}}}$$

5. *a*) eseguire l'azione a_i individuata dal metodo esposto sopra;
   b) misurare il nuovo stato **y**;
   c) registrare il rinforzo r_{t+1} (0 se non è disponibile).
6. Propagare l'attivazione del nuovo stato attraverso le connessioni della rete EVAL e calcolare la valutazione prevista del nuovo stato **y** usando l'output della rete

$$v(\mathbf{y}) = o^{EVAL}$$

7. Calcolare l'errore TD

$$e_{TD} = r_{t+1} + \gamma v(\mathbf{y}) - v(\mathbf{x})$$

8. Modificare i pesi della rete EVAL rispetto all'input **x**

$$\Delta w_i^{EVAL} = \eta e_{TD} x_i$$

9. Modificare i pesi della rete POL rispetto all'input **x**

$$\Delta w_{ij}^{POL} = \eta e_{TD} x_j \quad \text{se } i = a$$
$$\Delta w_{ij}^{POL} = 0 \quad \text{altrimenti}$$

10. Tornare al passo 2.

Se le reti sono multistrato, l'errore TD può essere propagato all'indietro sulle unità interne con il metodo di Back-propagation per correggere i pesi del primo strato.

5.2.2. *«Q-learning»*

Invece di apprendere due compiti separatamente – la valutazione dello stato e l'azione ottimale – è possibile applicare lo stesso procedimento descritto nella sezione precedente per apprendere a *predire l'utilità* di un'azione nel raggiungere lo scopo finale. Il modello prevede la presenza di un'unica rete neurale che riceve in ingresso l'input sensoriale e produce una valutazione dell'utilità di ciascuna azione [Watkins 1989]. La rete possiede un'unità di output per ciascuna delle possibili azioni e un meccanismo stocastico per la selezione dell'azione (fig. 2.24). Il sistema esegue in modo probabilistico l'azione corrispondente all'utilità massima. La trasformazione da stato sensoriale a utilità

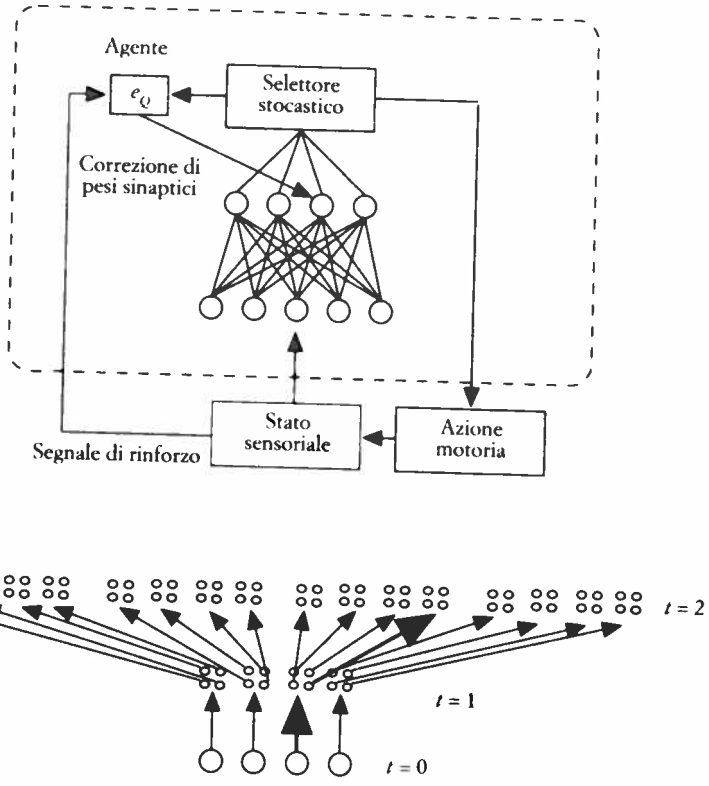


FIG. 2.24. Architettura del modello Q e rappresentazione del calcolo dell'utilità maggiore per una sequenza di soli due passi. Ciascun cerchietto rappresenta una possibile azione; il percorso temporale evidenziato dalla linea spessa rappresenta l'utilità maggiore dell'azione iniziale effettuata in risposta allo stato **x**. La rete neurale deve apprendere a prevedere per ogni stato quale sarà l'azione che comporta l'utilità maggiore da lì fino all'infinito.

di un'azione è stata definita da Watkins con il nome di «funzione Q». La funzione di utilità possiede la stessa proprietà ricorsiva della funzione di valutazione del modello AHC. L'utilità $U(\mathbf{x}, a_i)$ di un'azione a_i in risposta a uno stato **x** è la somma del rinforzo immediato più l'utilità massima che il sistema può ottenere dal prossimo stato **y** scontata del fattore γ

$$U(\mathbf{x}, a_i) = r + \gamma \max[U(\mathbf{y}, a_i)]$$

Il concetto di «utilità massima» è visualizzabile se rappresentiamo l'albero decisionale dell'agente per alcuni stati (fig. 2.24). In corrispondenza di ogni stato l'agente può eseguire una delle i azioni disponibili, ciascuna delle quali lo porterà a un nuovo stato per il quale esso potrà nuovamente scegliere tra i azioni disponibili; e così via. A ciascuna azione corrisponde un'utilità: l'utilità di un'azione si calcola proceden-

do verso l'alto e seguendo sempre i rami che corrispondono all'utilità massima.

Per la correzione dei pesi sinaptici possiamo considerare l'attivazione dell'unità con l'output massimo come un'approssimazione $u(\mathbf{x}, a_i)$ dell'utilità dell'azione corrispondente e, sfruttando la proprietà ricorsiva della funzione, richiedere che l'utilità prevista sia uguale alla somma del rinforzo ottenuto più l'utilità massima calcolata per lo stato successivo **y** scontata del fattore γ

$$e_Q = r_i + \gamma \max[u(\mathbf{y}, a_i)] - u(\mathbf{x}, a_i)$$

l'errore Q può essere quindi usato mediante la regola delta per correggere i pesi sinaptici dell'unità all'unità corrispondente all'azione eseguita

$$\Delta w_{ij} = \eta e_Q x_j \quad \text{se } i = a$$
$$\Delta w_{ij} = 0 \quad \text{altrimenti}$$

Come in precedenza, una singola rete neurale multistrato con i unità di output può essere sostituita da i reti multistrato con una sola unità di output per ciascuna al fine di evitare che le modifiche relative a un'azione influenzino anche i pesi sinaptici del primo strato i quali, nell'architettura monolitica, sono responsabili anche della generazione delle altre azioni.

Q-learning: algoritmo

Il modello è costituito da un'unica rete il cui input è lo stato corrente e ciascuna delle unità di output rappresenta l'utilità della selezione dell'azione corrispondente. Quanto maggiore è l'attivazione di un'unità, tanto maggiore è l'utilità che la rete attribuisce a quell'azione. L'esecuzione dell'azione viene effettuata da un meccanismo stocastico responsabile del processo esplorativo.

1. Inizializzare i pesi sinaptici della rete a piccoli valori casuali. Selezionare i valori del tasso di apprendimento η, del fattore di sconto γ e della temperatura T (ad esempio 0,1; 0,5 e 1,0 rispettivamente).
2. Usare il vettore di stato corrente **x** come input di ciascuna delle reti.
3. Propagare l'attivazione attraverso le connessioni della rete per ottenere l'utilità prevista di ciascuna azione rispetto allo stato **x**

$$u(\mathbf{x}, a_i) = o_i$$

4. Selezionare l'azione da eseguire usando il selettore stocastico

$$P(a_i) = \frac{e^{\frac{o_i}{T}}}{\sum_k e^{\frac{o_k}{T}}}$$

5. *a*) eseguire l'azione a_i individuata dal metodo esposto nella sezione precedente;
   b) misurare il nuovo stato **y**;
   c) registrare il rinforzo r (0 se non è disponibile).
6. Propagare l'attivazione del nuovo stato **y** attraverso le connessioni della rete per ottenere le nuove misure di utilità

$$u(\mathbf{x}, a_i) = o_i$$

e scegliere la misura di utilità maggiore

$$u_{max}(\mathbf{y}, a) = \max[u(\mathbf{y}, a_i)] : \forall a_i$$

7. Calcolare l'errore Q

$$e_Q = r + \gamma u_{max}(\mathbf{y}, a_i) - u(\mathbf{x}, a_i)$$

dove $u(\mathbf{x}, a_i)$ rappresenta l'attivazione dell'unità corrispondente all'azione individuata dal selettore stocastico.
8. Modificare solo i pesi dell'unità corrispondente all'azione a eseguita dal selettore rispetto all'input **x**

$$\Delta w_{ij} = \eta e_Q x_j \quad \text{se } i = a$$
$$\Delta w_{ij} = 0 \quad \text{altrimenti}$$

9. Tornare al passo 2.

Entrambi i modelli – AHC e Q – sono piuttosto lenti nell'apprendere: l'agente deve essere collocato molte volte nell'ambiente in posizioni casuali diverse prima di essere in grado di sviluppare una politica ottimale [Anderson 1987; Chapman e Kaelbling 1991; Mahadevan e Connell 1991]. Al contrario, gli animali non possono permettersi di provare troppe volte una certa strategia perché questo potrebbe costargli la vita. Per far fronte alla lentezza dell'apprendimento, sono state proposte una serie di varianti ai metodi di Differenza temporale [si consulti ad esempio Lin 1992 o Sutton e Barto 1998]. L'apprendimento per rinforzo rappresenta un paradigma molto interessante che sta suscitando un rinnovato interesse: esso infatti combina in una cornice teorica coerente le osservazioni e teorie sull'apprendimento animale con le esigenze di ottimizzazione dei sistemi di controllo ingegneristici. I metodi di differenza temporale hanno trovato interessanti applicazioni per lo sviluppo di agenti autonomi intelligenti sia simulati che sotto forma di robot reali [ad esempio Millan 1996].

1. Introduzione

I modelli descritti in questo capitolo hanno due caratteristiche distintive: possiedono un'architettura ricorrente e il loro funzionamento è descrivibile in base a una «funzione di energia». Come è già stato evidenziato nel capitolo precedente, una rete neurale con delle connessioni ricorrenti è in grado di esibire comportamenti dinamici per cui lo stato di attivazione delle unità del sistema evolve lungo una sequenza temporale. Questo significa anche che se una rete ricorrente riceve un input esterno stazionario, essa è in grado di generare una sequenza di stati di attivazione diversi fino al raggiungimento (possibilmente) di un punto di stabilità. Una rete neurale composta da semplici unità bipolari interconnesse rappresenta un sistema molto simile a quelli che vengono studiati dalla meccanica statistica. Le unità della rete sono paragonabili a delle particelle atomiche la cui distribuzione geometrica nello spazio rappresenta la struttura di alcuni materiali magnetici. In entrambi i sistemi i singoli componenti (unità e atomi) interagiscono in base a determinate regole.

Agli inizi degli anni Ottanta il fisico John Hopfield rese esplicite le somiglianze tra alcuni concetti della meccanica statistica e il funzionamento di una rete neurale composta di semplici neuroni McCulloch-Pitts completamente interconnessi [Hopfield 1982]; nello stesso articolo egli introdusse anche il concetto di «energia» per descrivere la dinamica di funzionamento della rete[1]. Il modello di Hopfield e le variazioni successive, tra cui la Macchina di Boltzmann, costituiscono l'argomento di questo capitolo. Così come i modelli analizzati nel capitolo precedente sono stati influenzati dai metodi e dai concetti impiegati

[1] L'estrema chiarezza della trattazione, la novità dell'analogia e la nitidezza del modello attrassero l'attenzione di molti ricercatori appartenenti a diverse discipline scientifiche; fu uno dei lavori che contribuì alla rapida diffusione delle reti neurali.

nel campo del riconoscimento di pattern e di ottimizzazione di funzioni, la classe di modelli descritti in questo capitolo si è giovata della teoria e dei metodi di analisi usati dai fisici nello studio della meccanica statistica. Questa combinazione ha dato luogo alla creazione di sistemi che, benché possano essere utilizzati per gli stessi tipi di compiti affrontati dai modelli già analizzati in precedenza (memorizzazione, riconoscimento, classificazione, ottimizzazione, ecc.), possiedono nuove e interessanti caratteristiche di funzionamento.

2. Il modello di Hopfield

Una rete di Hopfield è composta da N unità bipolari[2] completamente connesse fra di loro (fig. 3.1) la cui attivazione è data dalla familiare funzione a gradino della somma pesata A_i dei segnali provenienti dalle altre unità della rete

$$\Phi(A_i) = \begin{cases} 1 & \text{se } A_i > \vartheta_i \\ -1 & \text{altrimenti} \end{cases}$$

dove

$$A_i = \sum_j^N w_{ij} x_j$$

e ϑ_i rappresenta la soglia dell'unità che per semplicità eguagliamo a zero. L'attivazione di ciascuna unità può assumere uno di due soli stati $\{-1, +1\}$ il cui segno è dato dalla somma pesata dei suoi input. Lo «stato della rete» ad ogni istante è il vettore di attivazione di tutte le unità che la compongono. Per motivi che verranno chiariti in seguito, assumiamo che il valore delle connessioni auto-ricorrenti w_{ii} sia zero e che la matrice dei pesi sinaptici sia simmetrica ossia $w_{ij} = w_{ji}$.

Ciascuna unità della rete svolge le funzioni di input e di output. Definiamo quindi l'attivazione di una qualsiasi unità con la variabile x e utilizziamo gli indici i e j per distinguere – rispettivamente – tra unità postsinaptiche e presinaptiche dove sia necessario. Se applichiamo un pattern di input esterno $\mathbf{p}$ alle unità della rete, ossia imponiamo

$$x_i = p_i$$

e lasciamo poi che ciascuna unità modifichi la propria attivazione in base agli input provenienti dalle altre unità, la rete tenderà a procedere attraverso una sequenza di stati fino al raggiungimento di un pun-

[2] Il modello è applicabile anche a unità binarie, ma rende la matematica più complessa.

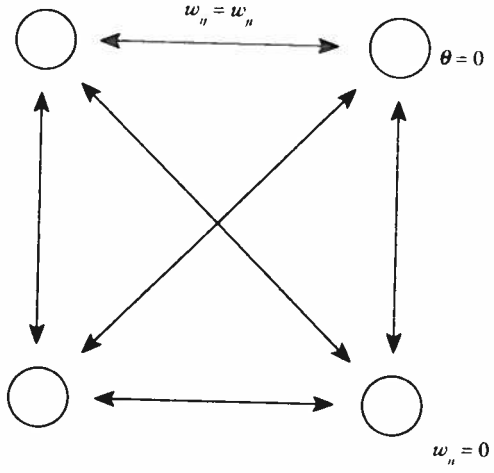


FIG. 3.1. Architettura del modello di Hopfield.

to di *stabilità* in cui nessuna unità modifica più il proprio segno[3]. Il punto di stabilità è uno stato di «equilibrio» e lo stato di attivazione delle unità in corrispondenza del punto di equilibrio rappresenta la risposta della rete al pattern **p**.

Vi sono due metodi per aggiornare lo stato della rete. Il metodo «sincrono» prevede che ad ogni istante tutte le unità ricalcolino la propria attivazione sulla base dell'attivazione corrente delle altre unità. Questo metodo permette di raggiungere uno stato stabile in una o poche iterazioni, ma richiede un orologio centrale il cui segnale venga distribuito a tutti i nodi della rete. Inoltre è altamente improbabile che i neuroni biologici di una rete neurale locale simile a quella di Hopfield (ad esempio i neuroni dell'area CA3 dell'ippocampo) modifichino il proprio stato simultaneamente. L'altro metodo consiste nell'aggiornamento «asincrono»: ogni unità ricalcola il proprio stato in tempi diversi. Questo metodo può essere realizzato scegliendo un ordine sequenziale oppure un ordine casuale: entrambi i procedimenti portano la rete verso lo stesso stato finale, ma la traiettoria definita dagli stati intermedi può essere diversa. Il procedimento consiste nello scegliere in modo casuale (o in base a un certo ordine) un'unità e ricalcolare il suo stato in base all'attivazione proveniente da tutte le altre unità; questo processo di aggiornamento continua fino a quando nessuna unità modifica più il proprio stato.

Le reti di Hopfield vengono utilizzate soprattutto per compiti di memorizzazione e riconoscimento. L'idea di base è quella di far apprendere alla rete una serie di pattern ciascuno dei quali interessa tutte le unità della rete; dopo la fase di addestramento la rete è in grado

[3] A condizione che i valori delle connessioni siano non-zero e che le attivazioni corrispondenti al pattern esterno non rappresentino già il punto di stabilità della rete.

di ricostruire la versione originale di un pattern incompleto o corrotto dal rumore oppure di produrre il pattern più simile a uno che non era mai stato presentato durante l'addestramento.

La memorizzazione dei pattern avviene mediante la «regola di Hebb estesa» (si veda il paragrafo 3.3 del primo capitolo). Ciascun pattern viene presentato alla rete e le connessioni sinaptiche (inizialmente uguali a zero) vengono modificate in base a

$$\Delta w_{ij}^{\mu} = x_i^{\mu} x_j^{\mu} \qquad i \neq j$$

L'apprendimento di ciascun pattern avviene in una sola presentazione. Nel caso di M pattern, i valori finali dei pesi sinaptici sono quindi dati da

$$w_{ij} = \begin{cases} \sum_{\mu=1}^{M} x_i^{\mu} x_j^{\mu} & i \neq j \\ 0 & i = j \end{cases}$$

È possibile dimostrare [Rojas 1996] che la condizione $w_{ii} = 0$ congiuntamente alla condizione di simmetria $w_{ij} = w_{ji}$ assicura la convergenza di una rete completamente connessa con aggiornamento asincrono a uno stato stabile.

Da questo punto di vista un'importante conseguenza della regola di Hebb estesa applicata alla rete di Hopfield è che i valori assegnati ai pesi delle connessioni sinaptiche che uniscono due nodi sono simmetrici

$$w_{ij} \equiv w_{ji}$$

La rete di Hopfield – come i sistemi nervosi biologici – memorizza e recupera i pattern in base al loro contenuto e non in base all'indirizzo numerico della casella di memoria come nel caso dei computer seriali tradizionali. Se alla rete viene presentata una versione incompleta di un pattern già memorizzato, essa procede attraverso una sequenza di stati fino alla ricostruzione del pattern originale: un pattern correttamente memorizzato rappresenta così un «attrattore» della rete. Il «bacino di attrazione» di un pattern memorizzato è la zona dello spazio individuata dai possibili stati della rete per cui essa converge verso quel pattern. La fase di apprendimento consiste nella creazione di attrattori stabili.

È importante chiedersi quale sia la «capacità» di una rete di Hopfield, ovvero, dato un numero finito N di nodi, quanti siano i pattern che possono essere memorizzati e recuperati correttamente senza fenomeni di interferenza o di instabilità. Un determinato pattern $\mathbf{p}^* = x^*$ è stabile se

$$\Phi(A_i^*) = p_i^* \quad \forall i$$

dove la somma pesata A_i^* degli input dalle altre unità è data da

$$A_i^* = \sum_j w_{ij} x_j^* = \sum_{j \neq i} \sum_{\mu} x_i^{\mu} x_j^{\mu} x_j^*$$

estraendo dalla sommatoria su tutti i pattern μ il contributo dato dal pattern $\mathbf{p}^*$, otteniamo la seguente scomposizione

$$A_i^* = (N-1)x_i^* + \sum_{j \neq i} \sum_{\mu \neq *} x_i^{\mu} x_j^{\mu} x_j^*$$

Il secondo termine della somma rappresenta l'«interferenza» (*crosstalk*) tra i pattern $\mathbf{p}^{\mu \neq *}$ e il pattern $\mathbf{p}^*$. Se il valore dell'interferenza è zero, allora in base al criterio $\Phi(A_i^*) = p_i^* \ \forall i$ sappiamo che il pattern è stabile; se il valore assoluto dell'interferenza è inferiore a $(N-1)$, il pattern è ancora stabile perché la somma pesata A_i^* non cambia di segno; se invece il valore assoluto dell'interferenza è maggiore di $(N-1)$ il pattern non è stabile e la rete evolve verso uno stato diverso da $\mathbf{p}^*$. Il valore dell'interferenza dipende unicamente dai pattern che vogliamo far memorizzare alla rete. La capacità della rete è il rapporto M/N fra il numero di pattern e il numero di nodi della rete. Assumendo dei pattern generati in modo casuale e valori di N e P molto maggiori di 1 la capacità al di sotto della quale si ottiene un errore con probabilità <0,0036 (non più di 0,36% delle unità produrranno risultati scorretti) è di $0{,}138N$ [Hertz, Krogh e Palmer 1991]. Questo valore rappresenta un limite drastico perché se vengono memorizzati anche pochi più di $0{,}138N$ pattern, le poche unità inizialmente instabili potrebbero provocare un fenomeno di caduta a valanga per cui un gran numero di unità evolverebbero verso uno stato scorretto e i pattern risultanti non sarebbero più riconoscibili. Sebbene questa capacità sia calcolata per pattern generati in modo casuale, essa rappresenta comunque un'importante cifra indicativa dei margini di sicurezza: una rete con 100 unità è in grado di memorizzare e recuperare correttamente circa 13 pattern casuali[4]. Questa capacità non è poi così piccola se pensiamo a situazioni realistiche: ad esempio, una rete di Hopfield è in grado di memorizzare stabilmente quasi 565 immagini (assumendo che esse siano casuali; siccome in realtà non lo sono, il numero è inferiore) presentate su una matrice di 64×64 pixel.

Se i pattern sono correlati, la capacità della rete diminuisce notevolmente perché il valore di interferenza diventa grande. Però, se questi pattern sono linearmente indipendenti, il metodo della «pseudo-inversione» (già introdotto nel capitolo precedente) permette di memorizzare correttamente fino a N–1 pattern. Si inizi col creare una

[4] Se i pattern sono completamente ortogonali, l'interferenza è sempre zero e la rete può memorizzare fino a N pattern. In questo caso il problema è che la rete non è in grado di completare versioni incomplete di pattern già memorizzati.

matrice di sovrapposizione dei pattern di input $\mathbf{Q}_{\mu\nu}$, dove μ e ν sono due indici che puntano entrambi agli M pattern e l'indice i punta ai componenti di ciascun pattern

$$\mathbf{Q}_{\mu\nu} = \frac{1}{N}\sum_i x_i^\mu x_i^\nu$$

i pesi sinaptici sono presto dati usando la matrice inversa $(\mathbf{Q}^{-1})_{\mu\nu}$

$$w_{ij} = \frac{1}{N}\sum_{\mu\nu} x_i^\mu (\mathbf{Q}^{-1})_{\mu\nu} x_j^\nu$$

Il recupero corretto dei pattern ha luogo se, dato un pattern di test $\mathbf{p}^* = x^*$ applicato alle unità della rete, il segno dell'input netto A_i^* per ciascun nodo è uguale al segno dell'elemento corrispondente al pattern $\mathbf{p}^*$. Dunque

$$\begin{aligned} A_i^* = \sum_j w_{ij} x_j^* &= \frac{1}{N}\sum_{j\mu\nu} x_i^\mu (\mathbf{Q}^{-1})_{\mu\nu} x_j^\nu x_j^* \\ &= \sum_{\mu\nu} x_i^\mu (\mathbf{Q}^{-1})_{\mu\nu} \mathbf{Q}_{\nu^*} \\ &= \sum_\mu x_i^\mu \delta_{\mu^*} \\ &= x_i^* \end{aligned}$$

dove δ_{μ^*} è il delta di Kronecker, definito a p. 172. Questo metodo non è biologicamente plausibile (in quanto il calcolo dei pesi delle sinapsi non è locale) e comunque vale solamente per pattern linearmente indipendenti, ma garantisce una matrice di pesi sinaptici ottimale per memorizzare fino a N–1 pattern.

2.1. La funzione di energia

La dinamica della rete di Hopfield è descrivibile da una funzione di energia H

$$H = -\frac{1}{2}\sum_i \sum_j w_{ij} x_i x_j + \sum_i x_i \vartheta_i$$

che, nel semplice caso dove le soglie e le connessioni auto-ricorrenti sono uguali a zero, diventa

$$H = -\frac{1}{2}\sum_i \sum_{j\neq i} w_{ij} x_i x_j$$

Il termine «energia» riflette l'analogia con la funzione che descrive il comportamento dei «vetri di spin», un tipo di materiali magnetici che sono studiati dai fisici con i metodi della meccanica statistica. La funzione di energia H possiede un ruolo simile a quella della funzione di errore E_W usata per i percettroni multistrato nel capitolo precedente: entrambe descrivono lo stato globale della rete in funzione dei parametri interni. Tuttavia, mentre la funzione di errore E_W richiede la conoscenza della risposta desiderata per ogni unità di output a ogni istante, la funzione di energia H si basa solamente sui parametri interni della rete (i valori dei pesi sinaptici e le attivazioni delle unità) in quanto non è possibile conoscere gli stati desiderati della rete durante il processo di evoluzione verso lo stato di equilibrio. Come per la funzione di errore E_W, anche la funzione di energia H assume valori proporzionali al comportamento desiderato: quanto più la rete è vicina alla risposta corretta, tanto più piccolo è il valore della funzione corrispondente (fig. 3.2). Nel caso dei percettroni questa proprietà è garan-

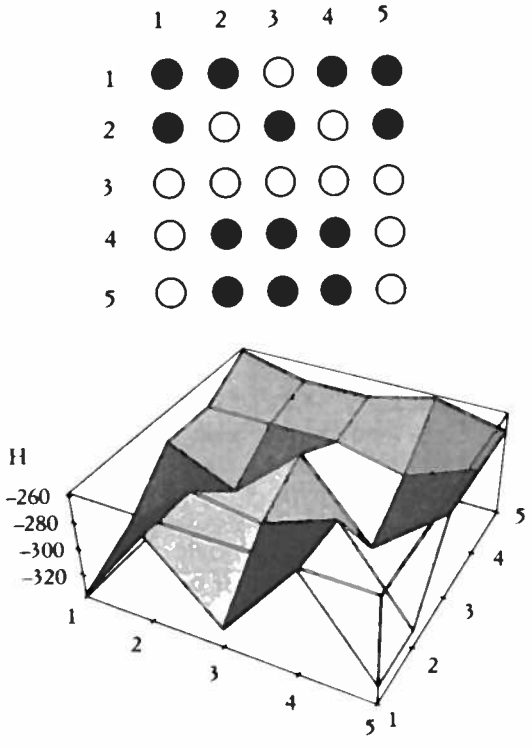


FIG. 3.2. Valori della funzione di energia rispetto agli stati della rete. Alto: la rete è costituita da 25 nodi visualizzati come una matrice quadrata sulla quale possono essere proiettati dei caratteri (alto): nero = −1, bianco = +1. Basso: in seguito alla fase di memorizzazione, la lettera A è stata ripresentata alla rete invertendo il valore di un nodo alla volta e misurandone l'energia. Ciascun punto del grafico corrisponde al valore dell'energia misurata invertendo il valore del nodo corrispondente. Il valore dell'energia misurata per la lettera A intatta è −340.

tita dal fatto che la regola delta effettua la discesa del gradiente della funzione stessa; in modo simile, è possibile dimostrare che la regola di Hebb estesa modifica i pesi della rete di Hopfield in modo da scendere il gradiente della funzione fino a un minimo che corrisponde al punto di attrazione per ogni pattern che viene appreso. L'apprendimento nella rete di Hopfield consiste dunque nel collocare ciascun pattern in un minimo diverso della funzione di energia creando dei bacini di attrazione; durante la fase di recupero, dato un pattern incompleto o corrotto, la rete diminuisce il livello di energia portandosi verso il bacino di attrazione più vicino.

2.1.1. *Riduzione dell'energia durante la memorizzazione*

La memorizzazione di un pattern consiste nell'abbassamento dell'energia della rete per quel determinato pattern in modo da collocarlo in un minimo della funzione. La modifica dei pesi sinaptici deve essere eseguita in modo tale da disturbare il meno possibile gli altri pattern già memorizzati in diversi minimi della funzione.

L'abbassamento dell'energia H

$$H = -\frac{1}{2}\sum_i \sum_{j \neq i} w_{ij} x_i x_j$$

è equivalente all'incremento dei suoi termini, dato il segno negativo posto di fronte all'equazione. Se consideriamo la memorizzazione di un nuovo pattern $\mathbf{p}^*$, possiamo scomporre la funzione di energia in due componenti

$$H = \left[-\frac{1}{2}\sum_i \sum_{j \neq i} w_{ij}^{\mu **} x_i x_j\right] + \left[-\frac{1}{2}\sum_i \sum_{j \neq i} w_{ij}^* x_i^* x_j^*\right]$$

il primo termine è l'energia dovuta a tutti pattern che sono già stati memorizzati nei pesi $w_{ij}^{\mu **}$ eccetto il pattern $\mathbf{p}^*$ e il secondo termine è il contributo apportato all'energia globale dalla memorizzazione del nuovo pattern $\mathbf{p}^*$. Per quanto riguarda il primo termine non c'è molto da fare; nel memorizzare il pattern $\mathbf{p}^*$ però dobbiamo modificare i pesi sinaptici in modo da rendere il contributo del secondo termine il più piccolo possibile. Dato il segno negativo, questo è equivalente ad aumentare il valore del termine $\frac{1}{2}\sum_i \sum_{j \neq i} w_{ij}^* x_i^* x_j^*$. Siccome il quadrato di x_i è sempre positivo la strategia migliore consiste nell'individuare una regola di apprendimento dei pesi sinaptici per cui

$$\frac{1}{2}\sum_i \sum_{j \neq i} w_{ij}^* x_i^* x_j^* = \frac{1}{2}\sum_i \sum_{j \neq i} x_i^2 x_j^2$$

è facile constatare che la regola di apprendimento che soddisfa questa eguaglianza è data dalla regola di Hebb estesa

$$w_{ij}^{*} = x_i^{*} x_j^{*}$$

Ne consegue che il modo migliore per abbassare l'energia della rete per ciascun pattern consiste nel calcolare i pesi sinaptici sommando le modifiche per tutti i pattern

$$w_{ij} = \sum_{\mu} x_i^{\mu} x_j^{\mu}$$

La regola di Hebb estesa corrisponde dunque all'abbassamento dell'energia nel modello di Hopfield. Si noti comunque che ogni modifica conduce a un aumento dell'energia della rete, ma questo è inevitabile, a meno che i pattern non siano ortogonali fra di loro.

2.1.2. *Riduzione dell'energia durante il recupero*

Il recupero di un pattern incompleto o corrotto dal rumore consiste nell'abbassamento dell'energia della rete fino a quando essa raggiunge il bacino di attrazione di quel pattern. Ora desideriamo osservare che l'azione della funzione a gradino bipolare con la quale calcoliamo l'attivazione di ciascuna unità conduce effettivamente all'abbassamento dell'energia della rete.

Scomponiamo innanzitutto l'energia totale della rete isolando il contributo dato dalla k-esima unità

$$H = \left[-\frac{1}{2} \sum_{i \neq k} \sum_{j \neq k} w_{ij} x_i x_j \right] + \left[-\frac{1}{2} x_k \sum_i w_{ik} x_i - \frac{1}{2} x_k \sum_j w_{kj} x_j \right]$$

dove il secondo termine indica il contributo del nodo k in funzione dei pesi sinaptici afferenti ed efferenti. Quando l'unità k modifica il proprio stato la differenza energetica dell'intera rete si calcola sottraendo l'energia precedente all'energia attuale; siccome il primo termine si annulla, questa differenza si riduce a

$$\Delta H = -\frac{1}{2} \left[\Delta x_k \sum_i w_{ik} x_i + \Delta x_k \sum_j w_{kj} x_j \right]$$

dove il simbolo Δ indica la differenza; ricordando ora che la matrice di pesi sinaptici è simmetrica (come consegue dall'applicazione della regola di Hebb estesa a unità bipolari), possiamo ulteriormente semplificare l'equazione

$$\Delta H = -\Delta x_k \left[\sum_j w_{kj} x_j \right]$$

Da qui si noti che il modo migliore per ridurre ΔH consiste nello scegliere il valore di x_k uguale a +1 quando la somma pesata $\sum_j w_{kj} x_j$ degli input degli altri neuroni è maggiore di zero e scegliere invece il valore di x_k uguale a −1 quando la somma pesata è minore o uguale a zero: in entrambi i casi il prodotto è positivo e, dato il segno negativo di fronte all'equazione, l'energia totale della rete risulta abbassata. Questo risultato conferma che l'aggiornamento dello stato della rete in base alla funzione di attivazione a gradino

$$\Phi\left(A_i = \sum_j w_{ij} x_j\right) = \begin{cases} 1 & \text{se } A_i > 0 \\ -1 & \text{altrimenti} \end{cases}$$

consiste effettivamente nella riduzione dell'energia totale della rete.

2.2. Vari tipi di attrattori

Il modello di Hopfield memorizza i pattern nei minimi della funzione di energia (fig. 3.3). Tuttavia la funzione di energia contiene altri minimi verso cui la rete può convergere durante la fase di recupero.

Per ciascun pattern memorizzato **p** esiste uno stato opposto **−p** che possiede lo stesso livello energetico e le stesse caratteristiche locali del bacino di attrazione. Questo stato si definisce «simmetrico» ed è facilmente identificabile e riconducibile al pattern corretto dallo sperimentatore. Esistono poi dei bacini di attrazione che corrispondono a stati «misti»: uno stato misto è una combinazione lineare di un numero dispari di pattern che sono stati memorizzati dalla rete [Amit, Gut-

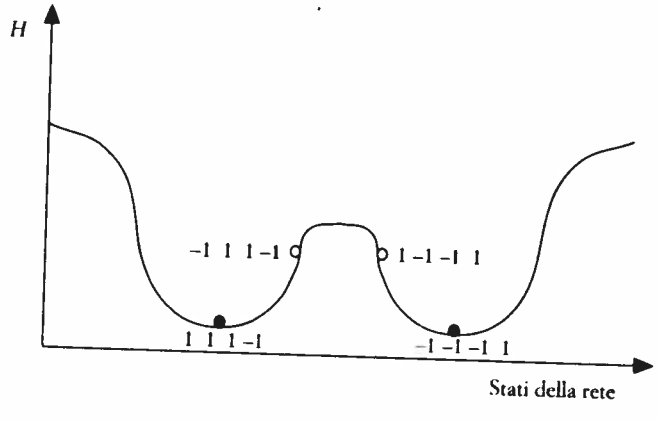


FIG. 3.3. Rappresentazione idealizzata di un attrattore originale e di un attrattore simmetrico. I cerchietti neri indicano lo stato di equilibrio; i cerchietti bianchi indicano uno stato instabile della rete. Durante la fase di recupero la rete scende verso il centro del bacino di attrazione più vicino.

freund e Sompolinsky 1985a]. Esistono infine altri stati di attrazione che non sono esprimibili in funzione dei pattern originariamente memorizzati dalla rete: essi sono definiti stati dei «vetri di spin» per indicare le somiglianze con le proprietà esibite da questi materiali magnetici. Sia gli stati misti che gli stati dei vetri di spin sono detti stati «spuri» o «metastabili»: essi possiedono un bacino di attrazione più piccolo rispetto agli attrattori dei pattern originariamente memorizzati dalla rete e solitamente sono caratterizzati da un livello energetico più alto. Gli stati dei vetri di spin, in particolare, possono apparire quando il numero di pattern memorizzati supera la capacità $0{,}138N$ [*ibidem* 1985b]. Come vedremo più sotto, un modo per evitare di rimanere bloccati in questi attrattori consiste nell'utilizzare delle unità stocastiche al posto delle unità bipolari deterministiche.

2.3. Unità stocastiche e il metodo del «raffreddamento simulato»

Le unità stocastiche sono nodi bipolari (o binari) che aggiornano il proprio stato in base a una certa probabilità: nel modello che prendiamo in considerazione in questa sezione la probabilità che il nodo assuma valore 1 è data dalla somma pesata degli input filtrata attraverso la funzione sigmoide

$$P(x_i = 1) = \frac{1}{1 + e^{-\beta\left(\sum_j w_{ij} x_j\right)}}$$

L'impiego di unità stocastiche rende l'analogia fra la rete di Hopfield e i materiali magnetici più realistica. Come è già stato anticipato in precedenza, un magnete può essere descritto come un insieme di unità elementari distribuite nello spazio e caratterizzate da un campo magnetico locale con una certa direzione; nel modello di Ising, ciascuno di questi elementi può assumere solo una fra due tipi di configurazioni opposte: direzione («spin») 1 o direzione –1. La direzione che ciascun elemento assume è data sia da influenze esterne al magnete stesso che dalla direzione del campo magnetico degli elementi vicini pesata da un opportuno coefficiente (analogo a un peso sinaptico nella rete di Hopfield). Tuttavia il calcolo esatto della direzione di ciascun elemento è valido solo alla temperatura dello zero assoluto (–273 gradi Celsius). In condizioni normali («temperatura finita») ciascun elemento modifica il proprio stato non più in modo deterministico, ma in modo probabilistico. Le fluttuazioni dal valore atteso sono tanto maggiori quanto più alta è la temperatura. Infatti questi materiali perdono le loro capacità magnetiche quando sono esposti ad alte temperature perché la direzione dei campi magnetici locali dei singoli elementi è totalmente casuale (ciascun materiale è caratterizzato da una temperatura critica oltre la quale perde le sue proprietà). Un modello più rea-

listico di questi materiali magnetici non fa uso di una regola deterministica per descrivere il cambiamento di stato di ogni singolo elemento, bensì utilizza una funzione stocastica: il modello di Glauber impiega proprio la funzione sigmoide sopra descritta ove la temperatura è inclusa nel parametro β che controlla la curvatura della funzione. Nel modello fisico

$$\beta = \frac{1}{kT}$$

dove k è la costante di Boltzmann e T è la temperatura espressa in gradi Kelvin. Quanto più la temperatura è alta, tanto più piccolo risulta β e di conseguenza la funzione sigmoide diventa più «schiacciata»; al contrario, quanto più bassa è la temperatura, tanto più la funzione sigmoide approssima la funzione a gradino[5].

L'impiego di unità stocastiche nel modello di Hopfield rende la transizione della rete verso lo stato di equilibrio più incerta, ma permette di evitare in teoria alcuni stati indesiderati come gli stati misti e gli stati dei vetri di spin: siccome questi stati presentano di solito un livello di energia più alto dei pattern memorizzati e possiedono un bacino di attrazione più ristretto (ovvero la rete viene attirata verso queste zone solo per un ristretto numero di stati di attivazione), la casualità introdotta dal meccanismo probabilistico permette alla rete di sfuggire dagli attrattori spuri e di dirigersi invece verso gli attrattori che corrispondono ai pattern memorizzati. Nel modello neurale di Hopfield la temperatura T diventa un semplice parametro della rete; possiamo riscrivere quindi

$$\beta = \frac{1}{T}$$

ignorando la costante di Boltzmann e scegliere un valore di T appropriato alla situazione (la curvatura della funzione sigmoide per alcuni valori di β è tracciata nella figura 2.5 del secondo capitolo). Per valori alti di T la transizione di stato è più incerta perché la funzione sigmoide restituisce probabilità vicine a 0,5 per un ampio spettro di valori dell'argomento. Ogni stato spurio possiede una temperatura critica al di sopra della quale la rete riesce ad uscire da quel bacino di attrazione e a dirigersi verso un attrattore caratterizzato da un livello energetico più basso (possibilmente corrispondente a un pattern memorizzato).

[5] Siccome il valore restituito dalla funzione rappresenta la *probabilità* del cambiamento di stato, una funzione sigmoide schiacciata genera un comportamento del sistema completamente casuale. Quando la temperatura è invece uguale allo zero assoluto, la funzione sigmoide è identica alla funzione a gradino: questo significa che la direzione del campo magnetico degli elementi è calcolabile in modo deterministico.

Una volta che la rete ha memorizzato i pattern in base alla regola di Hebb estesa, il recupero di un pattern può essere dunque effettuato usando le unità stocastiche. Quando un pattern incompleto o corrotto dal rumore viene presentato a una rete con unità stocastiche a temperatura finita, la rete può spostarsi verso un gran numero di possibili stati, ciascuno dotato di un livello energetico, la cui distribuzione di probabilità è data dalla distribuzione di Boltzmann. (Per questo motivo le reti di Hopfield con unità stocastiche vengono talvolta definite «Macchine di Boltzmann»; noi riserveremo invece questo termine per indicare un modello più generale caratterizzato da regole di apprendimento ben distinte che verrà descritto più sotto.) La probabilità P_1 che la rete si sposti verso uno stato a energia H_1 è data da

$$P_1 = ke^{-H_1/T}$$

e dipende dunque sia dal livello energetico dello stato che dalla temperatura usata per calcolare la transizione della rete. Se consideriamo ora la probabilità P_2 che la rete si sposti verso uno stato a energia H_2, il rapporto fra le due probabilità è dato da

$$\frac{P_1}{P_2} = \frac{ke^{-H_1/T}}{ke^{-H_2/T}} = e^{-(H_1-H_2)/T}$$

Se l'energia H_1 è più bassa di H_2

$$H_1 < H_2$$

ne consegue che

$$\frac{P_1}{P_2} = e^{-(H_1-H_2)/T} > 1$$

e quindi che

$$P_1 > P_2$$

ovvero è più probabile che la rete converga verso stati a energia più bassa.

Il calcolo probabilistico dello stato dei nodi della rete rende il modello biologicamente più plausibile perché il comportamento dei neuroni biologici è descritto meglio in termini probabilistici. Infine, l'approccio stocastico permette di introdurre del rumore sulle connessioni della rete rendendola così asimmetrica. Se la temperatura è maggiore di zero, una rete asimmetrica non presenta più gli stati spuri dei vetri di spin nella fase di recupero anche se l'asimmetria può rendere la convergenza verso l'attrattore più lunga [Parisi 1986].

La scelta del valore di temperatura deve essere fatta con accortezza perché valori troppo alti rendono il sistema instabile; i valori esatti dipendono dalle caratteristiche degli stati spuri che sono difficili da individuare. Una possibile soluzione consiste nel provare a scostarsi gradualmente da $T = 1$ in entrambe le direzioni e osservare il comportamento della rete. Una soluzione più efficiente consiste nella «ricottura simulata» (*simulated annealing*) della rete [Kirkpatrick, Gelatt e Vecchi 1983]. Questo metodo intende riprodurre un processo utilizzato in metallurgia: durante il processo di lavorazione un metallo viene portato a temperature molto alte fino al punto di fusione, colato nelle forme desiderate e quindi raffreddato in base a precise regole. La fase di riscaldamento agita le particelle che costituiscono il metallo rendendo possibile la creazione di nuove strutture. Il processo di raffreddamento è molto importante: se esso avviene troppo rapidamente la consolidazione delle nuove strutture incorpora delle discontinuità atomiche (ad esempio inversioni o punti di rottura) che risultano in un materiale con alta energia. Questi stati possono compromettere la robustezza e le caratteristiche del materiale. L'idea è quella di raffreddare il materiale gradualmente in modo che le nuove strutture si assestino gradualmente a temperature sempre più basse fino a raggiungere un punto di stabilità che corrisponde all'energia minima. Durante la fase di raffreddamento graduale le varie strutture spurie hanno la possibilità di mutare stato verso una configurazione regolare ad energia più bassa. La stessa procedura può essere applicata a una rete di Hopfield con unità stocastiche per migliorare il processo di convergenza verso l'attrattore corretto. Lo scopo della ricottura simulata consiste nel far convergere la rete verso uno stato di energia minima corrispondente al bacino di attrazione del pattern più simile a quello presentato alle unità senza finire in stati spuri di vario tipo. Inizialmente i nodi della rete vengono aggiornati per alcune volte usando un valore abbastanza alto della temperatura: in questa fase la rete si sposta verso zone a energia più bassa, ma è estremamente instabile. Successivamente il valore della temperatura viene diminuito e i nodi continuano a essere aggiornati per un certo numero di volte: ora la rete resta in una zona più circoscritta della funzione di energia spostandosi gradualmente verso punti ad energia più bassa. Questa fase è cruciale perché se i nodi vengono aggiornati solo per poche volte a una temperatura «media», la rete potrebbe restare intrappolata in uno stato spurio: infatti il processo di raffreddamento è caratterizzato da una fascia di temperatura critica per la quale l'energia del sistema può diminuire improvvisamente senza che il punto di arrivo corrisponda necessariamente all'energia minima. Nella fase finale la temperatura viene ulteriormente abbassata mentre i nodi continuano ad aggiornare il proprio stato fino a quando il sistema diventa definitivamente stabile (fig. 3.4).

La definizione di un «programma di raffreddamento» consiste nello scegliere i valori di temperatura T e il periodo trascorso P dalla

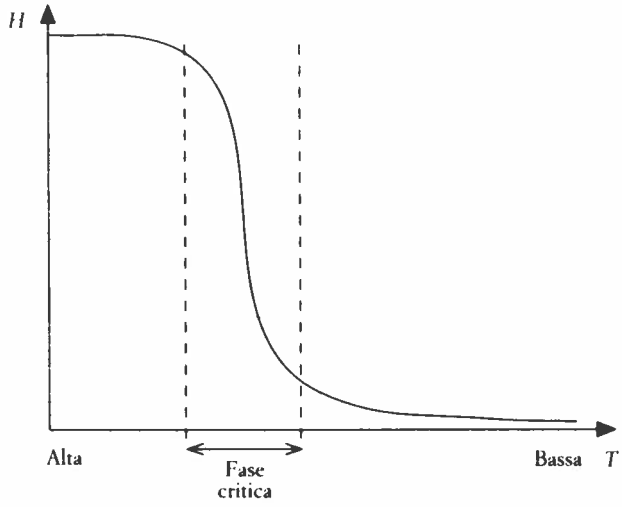


FIG. 3.4. Riduzione dell'energia durante il processo di raffreddamento simulato. Un programma di raffreddamento adeguato dovrebbe prevedere un campionamento maggiore degli stati della rete nelle zone a temperatura critica.

rete a ogni temperatura (ovvero il numero di volte in cui un nodo viene scelto a caso e aggiornato in modo probabilistico); ad esempio, per una rete con N nodi:

T	P
20	$N / 3$
15	$N / 2$
12	N
10	$N + N / 3$
5	N
1	N

La lunghezza del programma, le temperature utilizzate e la permanenza della rete a ciascuna temperatura dipendono interamente dal numero di nodi e dalle caratteristiche della superficie della funzione di energia. Malgrado la complessità della procedura e la necessità di scegliere un programma adeguato, il metodo di raffreddamento simulato risulta molto efficace.

2.4. Unità continue

Il modello di Hopfield funziona anche con unità continue la cui attivazione è data, ad esempio, dalla funzione tangente iperbolica: in

questo caso l'output di ciascuna unità assume un valore continuo nel dominio [–1, +1]

$$x_i = \Phi\left(\sum_j w_{ij} x_j\right) = \tanh\left(\beta \sum_j w_{ij} x_j\right)$$

e le altre convenzioni restano immutate: le connessioni autoricorrenti e le soglie sono uguali a zero e la memorizzazione dei pattern è ottenuta con la regola di Hebb estesa. Come nei casi precedenti, l'impiego della regola di Hebb estesa fa sì che le connessioni restino simmetriche. Ora ciascuna unità può aggiornare il proprio stato in almeno tre modi: con il metodo asincrono (un'unità alla volta), con il metodo sincrono (tutte le unità calcolano il proprio stato contemporaneamente) e con il metodo di «cambiamento continuo». In quest'ultimo caso ciascuna unità assorbe input in continuazione e modifica di conseguenza la propria attivazione. Quando viene usato il metodo di cambiamento continuo la quantità a cui siamo interessati è la variazione di attività di un'unità entro un certo intervallo di tempo τ_i

$$\tau_i \frac{\partial x_i}{\partial t} = -x_i + \Phi\left(\sum_j w_{ij} x_j\right)$$

I sistemi ad attivazione continua sono rilevanti sia per la realizzazione di circuiti neurali su silicio che per una simulazione più realistica delle dinamiche continue dei sistemi neurali biologici [Cohen e Grossberg 1983; Haykin 1999]. Ciascuna unità aumenta gradualmente il proprio livello di attivazione mentre la rete procede verso l'attrattore. Nel caso in cui invece venga usato un metodo discreto come l'aggiornamento asincrono il sistema è equivalente al modello di Hopfield a unità bipolari. Il valore continuo che un'unità assume può essere interpretato come l'attivazione media che l'unità assume nel modello stocastico. L'impiego di unità continue permette di memorizzare anche pattern continui.

Rete di Hopfield: algoritmo

Consideriamo una rete con N unità bipolari ed M pattern da memorizzare. Le connessioni autoricorrenti sono mantenute a zero e la soglia di attivazione dei nodi è zero.

1. Apprendimento

Calcolare i valori dei pesi sinaptici sommando i contributi di tutti i pattern

$$w_{ij} = \sum_{\mu}^{M} x_i^{\mu} x_j^{\mu}$$

2*a*. Recupero deterministico

Presentare una versione simile, parziale, corrotta o indebolita di un pattern memorizzato e aggiornare l'output di ciascun nodo della rete (ad esempio uno alla volta) fino a quando nessun nodo cambia più stato

$$\Phi\left(A_i = \sum_j w_{ij} x_j\right) = \begin{cases} 1 & \text{se } A_i > 0 \\ -1 & \text{altrimenti} \end{cases}$$

2*b*. Recupero stocastico

– Scegliere un valore di temperatura T (ad esempio 1) e calcolare il parametro β

$$\beta = \frac{1}{T}$$

– Presentare un pattern (simile, corrotto, incompleto o indebolito) alla rete e calcolare a turno per ciascun nodo la probabilità di assumere valore 1

$$P(x_i = 1) = \frac{1}{1 + e^{-\beta\left(\sum_j w_{ij} x_j\right)}}$$

– Estrarre un numero casuale r con distribuzione di probabilità uniforme tale che $0 \leq r < \max$; se $r < P(x_i = 1)\max$ il nodo assume valore 1, altrimenti assume valore –1.

– Ripetere fino a quando nessun nodo della rete modifica più il proprio stato.

2*c*. Raffreddamento simulato

– Scegliere un programma di raffreddamento (ad esempio quello descritto nel paragrafo 2.3); iniziare dalla temperatura più alta

$$\beta = \frac{1}{T}$$

– Selezionare a caso un nodo e calcolare la probabilità che il suo stato sia 1

$$P(x_i = 1) = \frac{1}{1 + e^{-\beta\left(\sum_j w_{ij} x_j\right)}}$$

– Estrarre un numero casuale r con distribuzione di probabilità uniforme tale che $0 \leq r < \max$; se $r < P(x_i = 1)\max$ il nodo assume valore 1, altrimenti assume valore –1.

– Ripetere il processo di selezione casuale e aggiornamento probabilistico alla stessa temperatura in base al programma di raffreddamento.

– Passare alla temperatura successiva.

2.5. Applicazioni

2.5.1. *Ricostruzione di immagini*

L'applicazione naturale del modello di Hopfield consiste nella ricostruzione in base al contenuto di pattern già memorizzati. Questa proprietà si rivela utile ad esempio nella ricostruzione di segnali trasmessi attraverso un canale rumoroso. In questo esempio osserviamo il comportamento di una rete con 25 nodi visualizzati come una matrice bidimensionale usata per la proiezione e memorizzazione di alcune lettere dell'alfabeto (fig. 3.2, alto). Una rete di queste dimensioni può memorizzare correttamente circa 3 pattern estratti da una distribuzione completamente casuale; nel caso di pattern realistici la distribuzione dei valori non è uniforme e la capacità dipende dal livello di correlazione dei valori che ciascun nodo deve assumere nel corso dell'addestramento. Durante la fase di memorizzazione vengono presentate alla rete le lettere A, B, C e M. Nella fase di recupero, dopo che i pesi sinaptici sono stati registrati, una delle lettere viene modificata azzerando o invertendo il valore di alcuni dei suoi componenti scelti in modo casuale e la versione così modificata viene ripresentata ai nodi della rete (fig. 3.5, sinistra). La presentazione di una versione distorta di un pattern già memorizzato corrisponde a un innalzamento dell'energia della rete e i nodi tendono quindi a modificare la propria attivazione. Dopo alcuni cicli di aggiornamento deterministico e asincrono (un nodo alla volta o in base a una selezione casuale), la rete raggiunge uno stato di stabilità che rappresenta l'attrattore più vicino e corrisponde alla versione originale del pattern memorizzato (fig. 3.5, destra). Ciascun attrattore è caratterizzato da un livello energetico molto più basso della versione corrotta da rumore.

La figura 3.2 visualizza l'energia della rete, dopo la memorizzazione delle 4 lettere, per 25 diverse distorsioni della lettera A. Ciascuna distorsione corrisponde all'inversione del valore di un singolo componente dell'immagine (distanza di Hamming = 1). In tutti i casi una distorsione corrisponde a un innalzamento dell'energia della rete, ma i contributi dei vari nodi sono diversi. Durante la fase di recupero la rete ricostruisce la versione originale scendendo la superficie della funzione, come è stato dimostrato nel paragrafo 2.1.2.

Lo stesso metodo di memorizzazione e recupero può essere applicato a situazioni più interessanti con matrici bidimensionali corrispondenti a immagini ad alta risoluzione e può essere impiegato anche per studiare le caratteristiche di memorizzazione e ricordo dei sistemi nervosi biologici. Virasoro [1989] ha confrontato il comportamento di una rete di Hopfield lesionata con i fenomeni di prosopagnosia, una patologia neurologica per la quale i pazienti non sono più in grado di riconoscere una categoria di visi familiari. Malgrado la semplicità del modello di Hopfield, esso contiene un numero relativamente ristretto di assunzioni formali e le regole di transizione si attengono a meccani-

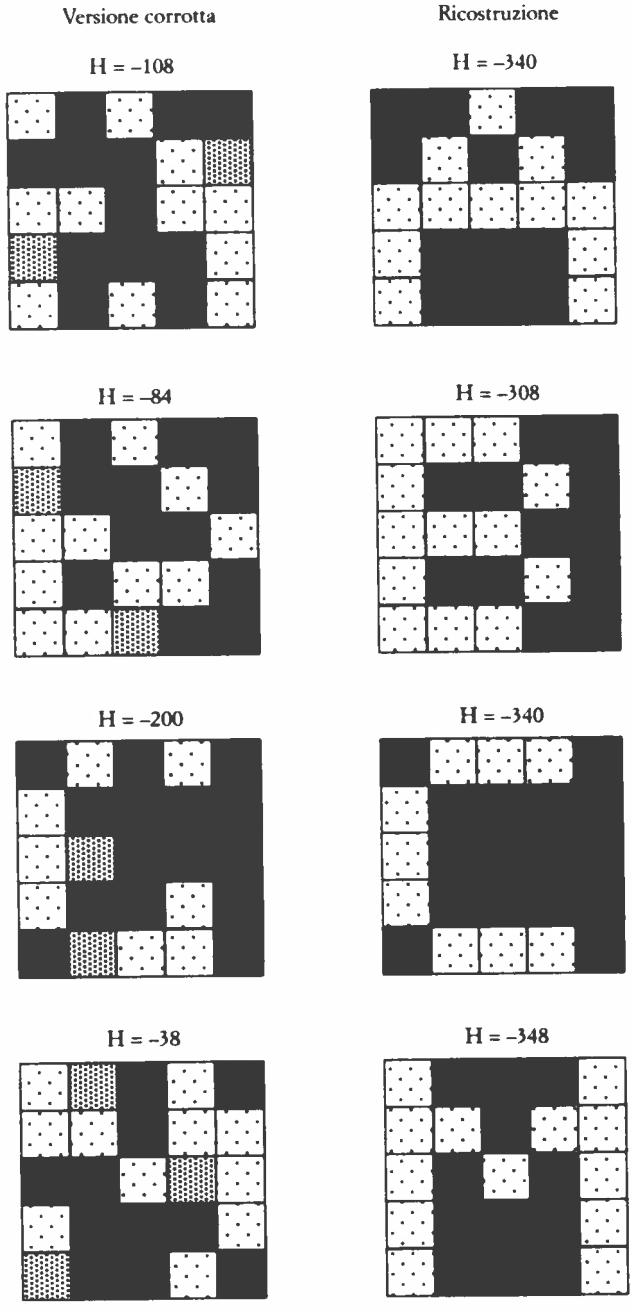


FIG. 3.5. Ricostruzione di caratteri memorizzati corrotti da rumore (invertendo o azzerando i valori di unità scelte in modo casuale). L'architettura della rete è illustrata in figura 3.2. Nero = −1, bianco = +1, grigio = 0. La ricostruzione avviene aggiornando il valore dei nodi in modo casuale fino a quando lo stato della rete è stabile. Il valore dell'energia per ciascuno stato è indicato sopra le figure. Il rilassamento verso uno stato stabile corrisponde a un abbassamento del valore dell'energia.

smi «biologicamente plausibili» (regola di Hebb e attivazione McCulloch-Pitts). Kleinfeld e Sompolinski [1989] hanno quindi applicato una versione modificata del modello di Hopfield (in cui l'attivazione di ciascun nodo possiede una traccia temporale) per la simulazione del movimento natatorio nel mollusco *Tritonia diomedea*. Il segno delle sinapsi artificiali sviluppate dal modello neurale è stato poi esattamente riscontrato nella configurazione di connessioni eccitatorie e inibitorie dell'animale.

2.5.2. *Ottimizzazione con vincoli: il problema del commesso viaggiatore*

Il modello di Hopfield può essere impiegato in compiti di ottimizzazione vincolata quando il problema può essere espresso nei termini di una funzione di energia. La soluzione viene fornita applicando le regole di evoluzione dinamica al modello così ottenuto.

Così come lo XOR è un problema tipico nel campo del riconoscimento di pattern, il problema del commesso viaggiatore – spesso abbreviato in TSP (*Travelling Salesman Problem*) – svolge una funzione analoga nel campo dell'ottimizzazione vincolata. Nel TSP classico un commesso viaggiatore deve visitare N città attraverso il percorso più breve passando per ciascuna città una sola volta e tornando alla città di partenza alla fine del giro[6]. Il problema diventa particolarmente complesso quando il numero di città aumenta.

Nel caso in cui vi siano N città da visitare, il modello di Hopfield richiede $N \times N$ unità disposte su una matrice ove le colonne indicano la posizione di ciascuna città nel giro del commesso e le righe indicano la città (fig 3.6). Ad esempio, per un problema con 10 città, la riga di attivazioni corrispondenti alla città di Trieste {0, 0, 0, 0, 1, 0, 0, 0, 0, 0} indica che Trieste è la quinta città ad essere visitata. Siccome desideriamo che ciascuna città venga visitata una sola volta, ciascuna riga deve contenere solo un'unità attiva; analogamente, siccome il commesso viaggiatore non può essere presente in più di una città contemporaneamente, desideriamo che ciascuna colonna possieda una sola unità attiva. Infine, desideriamo che il percorso totale sia il più breve possibile. Hopfield e Tank [1985] hanno costruito una funzione di energia H basata su quattro vincoli

$$H = \alpha \sum_{a} \sum_{i} \sum_{j \neq i} x_{a,i} x_{a,j} +$$

[6] Un gran numero di problemi possono esser parafrasati in questi termini come, ad esempio, la rotazione di operai all'interno di una fabbrica, la produzione di merci in una serie di ditte affiliate distribuite in vari paesi, la distribuzione di personale di bordo nelle compagnie aeree, ecc.

Rappresentazione

Città	Posizione 1	2	3	4	5	6	7	8	9	10
Firenze	1	0	0	0	0	0	0	0	0	0
Roma	0	1	0	0	0	0	0	0	0	0
Palermo	0	0	1	0	0	0	0	0	0	0
Bari	0	0	0	1	0	0	0	0	0	0
Trieste	0	0	0	0	1	0	0	0	0	0
Gorizia	0	0	0	0	0	1	0	0	0	0
Milano	0	0	0	0	0	0	1	0	0	0
Padova	0	0	0	0	0	0	0	1	0	0
Bologna	0	0	0	0	0	0	0	0	1	0
Pesaro	0	0	0	0	0	0	0	0	0	1

Architettura

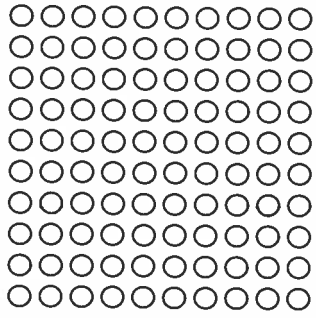


FIG. 3.6. Disposizione delle unità della rete per il problema del commesso viaggiatore.

– vi deve essere una sola unità attiva per riga (una città deve essere visitata una sola volta);

$$+\beta \sum_i \sum_a \sum_{b \neq a} x_{a,i} x_{b,i} +$$

– vi deve essere una sola unità attiva per colonna (il commesso viaggiatore non possiede il dono dell'ubiquità);

$$+\gamma \left(N - \sum_a \sum_i x_{a,i} \right)^2 +$$

– vi devono essere esattamente N unità attive nella rete (tutte le città devono essere visitate, e solo una volta);

$$+\varepsilon \sum_a \sum_{b \neq a} \sum_i d_{a,b} x_{a,i} (x_{b,i+1} + x_{b,i-1})$$

– la lunghezza del percorso deve essere la minore possibile (la somma è proporzionale alla lunghezza del percorso totale).

Nella notazione impiegata, a e b sono indici che puntano entrambi alle righe della matrice di nodi; i e j sono indici che puntano entrambi alle colonne; α, β, γ, e ε sono costanti; $d_{a,b}$ è la distanza fra due città a e b. Si noti che la distanza $d_{a,b}$ può essere espressa in modo arbitrario e non deve necessariamente rispondere ai criteri della geometria euclidea: essa può includere quindi vie preferenziali, nozioni di tempo e, in generale, tutti i costi e vantaggi associati a un certo percorso. I valori dei pesi sinaptici fra le unità sono dati da

$$w_{a,i;b,j} = -\alpha \delta_{ab}(1 - \delta_{ij}) - \beta \delta_{ij}(1 - \delta_{ab}) - \gamma - \varepsilon d_{a,b}(\delta_{i,j+1} - \delta_{i,j-1})$$

dove δ_{xy} è il delta di Kronecker

$$\delta_{xy} = \begin{cases} 1 & \text{se } x = j \\ 0 & \text{altrimenti} \end{cases}$$

e le notazioni usate sono identiche a quelle descritte sopra per la funzione di energia. La rete neurale viene inizializzata applicando dei valori casuali alle unità e viene poi lasciata libera di procedere verso uno stato di energia più bassa; l'attivazione finale dei nodi rappresenta il percorso ottimale proposto dalla rete. Il modello di Hopfield tuttavia perde efficienza con l'aumentare del numero di città [Wilson e Pawley 1988]; altri autori hanno quindi proposto diverse variazioni [si veda ad esempio Peterson e Soedeberg 1989 oppure Rojas 1996].

2.6. «Brain State in a Box»

Brain State in a Box (BSB) è una rete neurale completamente ricorrente sviluppata da Anderson *et al.* [1977] cinque anni prima del famoso articolo di Hopfield; essa meriterebbe una sezione a sé stante, ma, date le somiglianze tra i due modelli, può essere descritta assieme ai modelli neurali caratterizzati da una funzione di energia. L'idea che sta dietro al modello BSB consiste nell'utilizzare funzioni di attivazione lineari, ma ristrette entro i limiti −1 e +1

$$\Phi(A_i) = \begin{cases} +1 & \text{se } A_i > 1 \\ A_i & \text{se } -1 \le A_i \le 1 \\ -1 & \text{se } A_i < -1 \end{cases}$$

dove A_i è la solita somma pesata degli input provenienti da tutti gli altri nodi della rete. Ad ogni istante la rete può assumere un qualsiasi stato all'interno di un ipercubo i cui vertici sono definiti dal limite di attivazione dei nodi (da cui il nome dato al modello). Tutti i nodi modificano il proprio stato simultaneamente fino a quando la rete raggiunge un vertice dell'ipercubo (tutte le attivazioni delle unità hanno raggiunto gli estremi): la configurazione finale rappresenta la risposta della rete. La fase di apprendimento avviene inizializzando i pesi sinaptici a piccoli valori casuali intorno allo zero (incluse le connessioni autoricorrenti). Ciascun pattern viene presentato alla rete ed essa viene lasciata procedere verso un vertice dell'ipercubo (che rappresenta anche il punto di stabilità della rete[7]); quindi i pesi sinaptici vengono modificati in base alla regola di Hebb estesa

[7] La dimostrazione che la rete raggiunge un punto di stabilità è data dal teorema Cohen-Grossberg [Cohen e Grossberg 1983]: questo teorema delinea le condizioni di stabilità per un'ampia classe di reti neurali ricorrenti ad attivazione continua (una delle condizioni è che i pesi sinaptici siano simmetrici).

$$w_{ij}^{t} = w_{ij}^{t-1} + \eta x_i x_j$$

dove η è il tasso di apprendimento. La regola di apprendimento fissa nei pesi della rete gli stati stabili finali. Questi sono i bacini di attrazione e, come nel caso della rete di Hopfield, a ciascun bacino di attrazione corrisponde un bacino di attrazione equivalente e corrispondente al pattern inverso. Questa simmetria può essere eliminata se viene utilizzata un'unità di *bias* con valore costante 1 (si veda il secondo capitolo): in questo caso esiste un singolo bacino di attrazione per ogni stato memorizzato.

Il modello BSB è stato impiegato nello sviluppo del «medico istantaneo» (così definito da Hecht-Nielsen [1990]). Lo scopo dell'applicazione è quello di avere un sistema in grado di fornire automaticamente la diagnosi e la cura per un paziente che presenta determinati sintomi [Anderson 1986]. I dati di apprendimento sono dei vettori che rappresentano le cartelle cliniche di molti pazienti precedenti. Ciascun vettore di addestramento è composto di tre parti: i sintomi, la diagnosi e la cura. Se un certo dato è presente (sintomo, diagnosi o cura), le unità corrispondenti assumono valore +1, altrimenti sono inizializzate a −1. Una volta che la rete ha appreso un gran numero di cartelle cliniche, è possibile inserire solamente i dati relativi ai sintomi di un nuovo paziente e lasciare tutti gli altri nodi a 0. La rete evolverà presto verso un punto stabile e le attivazioni dei nodi forniranno la diagnosi e la cura per il paziente.

«Brain State in a Box»: algoritmo

Apprendimento

1. Inizializzare i pesi sinaptici (connessioni autoricorrenti e *bias* inclusi):

$$w_{ij} = rnd(\pm 0{,}1)$$

Fissare il tasso di apprendimento in modo tale che

$$0 \leq \eta \leq 1 \quad \text{(ad esempio 1)}$$

2. Per ciascun pattern di addestramento $\mathbf{x}^{\mu}$ eseguire i passi 3-5.
3. Applicare il pattern a tutte le unità della rete.
4. Calcolare simultaneamente i nuovi valori di tutte le unità

$$\Phi(A_i) = \begin{cases} +1 & \text{se } A_i > 1 \\ A_i & \text{se } -1 \leq A_i \leq 1 \\ -1 & \text{se } A_i < -1 \end{cases}$$

fino a quando la rete non cambia più stato.

5. Modificare i pesi sinaptici

$$w_{ij}^{t} = w_{ij}^{t-1} + \eta x_i^{\mu} x_j^{\mu}$$

Recupero

1. Presentare una versione corrotta o incompleta di un pattern già memorizzato, oppure un pattern completamente nuovo (con valori entro il dominio [–1, +1]).
2. Calcolare simultaneamente i nuovi valori di tutte le unità

$$\Phi(A_i) = \begin{cases} +1 & \text{se } A_i > 1 \\ A_i & \text{se } -1 \leq A_i \leq 1 \\ -1 & \text{se } A_i \leq 1 \end{cases}$$

fino a quando la rete non cambia più stato.

3. La Macchina di Boltzmann

La Macchina di Boltzmann (BM) è un modello generale per reti neurali ricorrenti stocastiche con connessioni simmetriche $w_{ij} = w_{ji}$ [Ackley, Hinton e Sejnowski 1985; Hinton e Sejnowski 1986]. Il nome deriva dal fatto che la probabilità di trovare la rete in un certo stato segue una distribuzione di Boltzmann (si veda il paragrafo 2.3). Essa può essere descritta dalla stessa funzione di energia utilizzata per la rete di Hopfield. La differenza principale tra la rete di Hopfield a unità stocastiche e la Macchina di Boltzmann è che quest'ultima può possedere in aggiunta alle unità «visibili» anche unità nascoste che servono unicamente per l'elaborazione interna. Di conseguenza l'algoritmo di apprendimento della Macchina di Boltzmann è diverso dal semplice algoritmo di Hopfield. La Macchina di Boltzmann può assumere qualsiasi architettura, purché le connessioni siano simmetriche. Essa può essere quindi impiegata per svolgere compiti di memorizzazione, categorizzazione, approssimazione di funzioni, ecc. Le unità nascoste permettono inoltre di memorizzare nelle unità visibili dei pattern che una rete realizzata secondo il modello di Hopfield non sarebbe in grado di memorizzare. Ad esempio i quattro pattern {+1, –1, +1}, {–1, +1, +1}, {–1, –1, –1} e {+1, +1, –1} richiederebbero all'uscita dell'unità corrispondente al terzo simbolo di assumere lo stesso valore quando i segnali provenienti dalle altre due unità valgano (–1, –1) e (1, 1), il che non può verificarsi nel modello di Hopfield. L'aggiunta di una unità nascosta consente di risolvere il problema.

Le unità della Macchina di Boltzmann assumono uno stato bipolare in base a una funzione probabilistica sigmoide controllata dal parametro della temperatura (si veda il paragrafo 2.3). Esse si distinguono in unità *visibili* e unità *nascoste*: le unità visibili possono essere a loro volta suddivise in unità di input e unità di output (fig. 3.7). Quando la

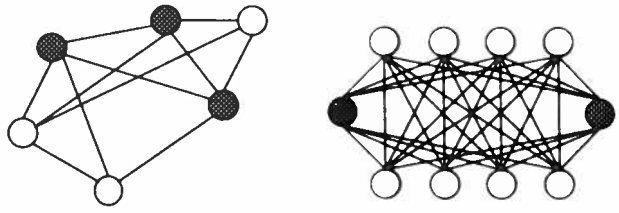


FIG. 3.7. Architetture della Macchina di Boltzmann. Ciascuna connessione è simmetrica, ovvero assume lo stesso valore in entrambe le direzioni. Le unità grigie sono le unità nascoste della rete, quelle bianche le unità visibili su cui vengono applicati pattern di addestramento. Sinistra: rete utilizzata per memorizzazione di pattern. Destra: rete utilizzata per comprimere pattern, ove le unità in basso rappresentano l'input della rete e le unità in alto l'output. Durante la fase di addestramento l'input e la risposta desiderata vengono presentati contemporaneamente alle rispettive unità; durante la fase di recupero solamente le unità di input ricevono segnale esterno e la rete genera il pattern corrispondente sulle unità di output.

rete viene impiegata per compiti di memorizzazione e di ottimizzazione ciascun pattern viene presentato su tutte le unità visibili della rete; quando invece viene impiegata per compiti di classificazione o approssimazione di funzioni, il vettore di input viene applicato alle unità di input e il vettore di risposta desiderata viene applicato contemporaneamente alle unità di output. Da questo punto di vista la procedura di addestramento della rete è biologicamente più plausibile dell'algoritmo di Back-propagation perché tratta la risposta desiderata alla stessa stregua dell'informazione di input (si ricordi che in Back-propagation la risposta desiderata viene usata per calcolare l'errore, ma non viene presentata alle unità di output).

Per comprendere il funzionamento della Macchina di Boltzmann dobbiamo pensare in termini di probabilità di stato del sistema. Una rete con S unità visibili può assumere 2^S possibili stati. Noi siamo però interessati a far sì che il sistema assuma solo un certo numero di stati che corrispondono ai pattern da memorizzare. L'apprendimento di un determinato pattern corrisponde dunque alla massimizzazione della probabilità che le unità visibili si trovino in quello stato e alla riduzione della probabilità che esse si trovino in tutti gli altri stati. Se vogliamo far apprendere più pattern dobbiamo far sì che la probabilità associata a ciascuno degli stati corrispondenti assuma un determinato valore. In conclusione, la modifica dei pesi sinaptici della Macchina di Boltzmann fa sì che gli stati delle unità visibili della rete assumano la «distribuzione di probabilità» dei vettori di input-output del gruppo di addestramento.

Riprendendo gli argomenti già introdotti per la rete di Hopfield, l'energia di uno stato della rete è data da

$$H = -\frac{1}{2}\sum_i \sum_j w_{ij} x_i x_j$$

e la probabilità P_s di trovare la rete in un certo stato **s** una volta che essa ha raggiunto l'equilibrio a una determinata temperatura, è data da

$$P_s = \frac{1}{Z} e^{-\beta H_s}$$

dove

$$\beta = 1 / T$$

e

$$Z = \sum_s e^{-\beta H_s}$$

è il fattore di normalizzazione che rappresenta la somma delle probabilità di trovare il sistema in tutti i possibili stati. Questa distribuzione di probabilità si chiama distribuzione di Boltzmann (si veda il paragrafo 2.3) e dipende unicamente dai valori dei pesi sinaptici.

Siccome nella Macchina di Boltzmann vi sono anche T (da non confondere con la temperatura) unità nascoste che possono assumere 2^T stati, il numero totale di stati dell'intera rete è 2^{S+T}. Indichiamo i possibili stati delle unità visibili con l'indice σ e i possibili stati delle unità nascoste con l'indice τ. Ora, se applichiamo un input casuale alla rete e lasciamo evolvere lo stato in base alle dinamiche di transizione stocastica, la probabilità P_σ di trovare le unità visibili in un determinato stato σ è data da

$$P_\sigma = \sum_\tau P_{\sigma\tau} = \sum_{\sigma\tau} \frac{1}{Z} e^{-\beta H_{\sigma\tau}}$$

dove

$$H_{\sigma\tau} = -\frac{1}{2} \sum_i \sum_j w_{ij} x_i^{\sigma\tau} x_j^{\sigma\tau}$$

è l'energia della rete quando le unità visibili sono nello stato σ e le unità nascoste sono nello stato τ.

Noi desideriamo però che la probabilità di stato delle unità visibili segua la distribuzione di probabilità Q_σ corrispondente ai pattern da memorizzare. La misura di errore E_{QP} tra la distribuzione di probabilità P_σ quando il sistema è in libera evoluzione e la distribuzione di probabilità Q_σ ottenuta presentando i pattern desiderati alle unità visibili è descritta da

$$E_{QP} = \sum_\sigma Q_\sigma \log \frac{Q_\sigma}{P_\sigma}$$

che in Teoria dell'Informazione è nota con il nome di Entropia relativa e indica per l'appunto la distanza tra due distribuzioni di probabilità. Siccome desideriamo ridurre E_{QP}, possiamo calcolare la modifica dei pesi sinaptici usando il metodo di discesa del gradiente della funzione di Entropia relativa rispetto ai pesi della rete

$$\Delta w_{ij} = -\eta \frac{\partial E_{QP}}{\partial w_{ij}}$$

la cui soluzione fornisce la seguente regola

$$\Delta w_{ij} = \eta\beta(\langle x_i x_j \rangle_{\text{vincolate}} - \langle x_i x_j \rangle_{\text{libere}})$$

dove

$$\langle x_i x_j \rangle_{\text{vincolate}} = \sum_{\sigma} \sum_{\tau} Q_{\sigma} P_{\tau/\sigma} x_i^{\sigma\tau} x_j^{\sigma\tau}$$

dove $P_{\tau/\sigma}$ è la probabilità condizionale di trovare le unità nascoste nello stato τ dato un certo stato σ delle unità visibili, $\langle x_i x_j \rangle_{\text{vincolate}}$ è il valore medio del prodotto fra l'unità postsinaptica e presinaptica quando le unità visibili della rete sono vincolate ad assumere i valori dei pattern da apprendere pesato dalla probabilità di occorrenza di ciascun pattern e $\langle x_i x_j \rangle_{\text{libere}}$ è il valore medio del prodotto fra l'unità postsinaptica e presinaptica quando la rete è libera di evolversi applicando valori iniziali casuali alle unità visibili. Ciascuna delle due medie deve essere calcolata solo dopo aver portato la rete in una fase di equilibrio ad esempio con un processo di ricottura simulata. Alternativamente, la regola può essere descritta dicendo che la modifica dei pesi sinaptici è proporzionale alla differenza tra la probabilità che due unità assumano lo stesso valore 1 durante la fase di presentazione dei pattern di addestramento $P_{ij}(+1)_{\text{vincolate}}$ e durante la fase di libera evoluzione $P_{ij}(+1)_{\text{libere}}$

$$\Delta w_{ij} \propto (P_{ij}(+1)_{\text{vincolate}} - P_{ij}(+1)_{\text{libere}})$$

L'idea di fondo è sempre la stessa: se la rete ha memorizzato i pattern di addestramento, allora la probabilità che le unità assumano un determinato valore è identica sia nel caso in cui il pattern è imposto dall'esterno che nel caso in cui vengono applicati dei valori iniziali casuali e la rete viene lasciata libera di evolversi. Quando questa situazione si verifica la differenza diventa zero e i pesi sinaptici cessano di mutar valore.

L'apprendimento nella Macchina di Boltzmann avviene gradualmente e richiede un numero elevato di calcoli, ma alcuni dati comparativi sembrano indicare che essa è più efficace dell'algoritmo di Back-propagation in compiti di classificazione [Kohonen, Barna e Chrisley 1988]. Inoltre sono state anche sviluppate alcune approssi-

mazioni che snelliscono notevolmente l'algoritmo esatto e talvolta ne accelerano addirittura la convergenza; ne esamineremo alcune più sotto.

Si noti nel frattempo che la regola di apprendimento della Macchina di Boltzmann è costituita di due parti. Il primo termine è di fatto la regola di Hebb estesa che è stata impiegata anche per la rete di Hopfield mentre il secondo termine rappresenta una cancellazione delle associazioni scorrette che la rete produce quando essa evolve in mancanza di input esterno (o con input esterno casuale). Se il secondo termine non fosse sottratto al primo, le unità nascoste potrebbero memorizzare degli stati che sono unicamente la produzione interna del sistema durante la fase di rilassamento perché esse non sono in grado di distinguere quando le unità visibili sono vincolate all'input esterno e quando invece il sistema è in libera evoluzione. Il confronto tra le associazioni ottenute in questi due stati è una caratteristica importante della Macchina di Boltzmann. Gli inventori hanno paragonato la fase di libera evoluzione della rete alla fase di sogno dei sistemi viventi. In un filmato divulgativo essi hanno disposto i nodi visibili della rete su una matrice bidimensionale e hanno fatto memorizzare una serie di semplici pattern raffiguranti la sagoma di una casa, di un albero, ecc. Durante la fase di libera evoluzione la rete passa attraverso una sequenza di stati in cui sono visibili combinazioni di pattern presentati durante lo «stato di veglia». Il motivo di tale paragone si basa sull'interpretazione data da Crick e Mitchinson [1983] della fase di sonno REM; secondo questi autori la configurazione di attività casuale delle aree frontali della corteccia servirebbe a cancellare le associazioni spurie che si sarebbero formate durante il giorno. Tuttavia, anche se è vero che durante la fase di sogno noi riproduciamo delle sequenze di ricordi e sensazioni in libera successione e che la mancanza di sonno limita notevolmente la capacità di apprendimento, questo confronto sembra ancora abbastanza azzardato.

Per poter calcolare il prodotto medio fra le unità, la rete deve essere in «equilibrio termico», ovvero deve trovarsi in uno stato a temperatura maggiore di zero in cui vi sia una fluttuazione dell'attivazione delle singole unità mentre la rete si mantiene sempre nella stessa zona a bassa energia; tuttavia la temperatura deve essere sufficientemente bassa per evitare di apprendere stati spuri del sistema. Il metodo della ricottura simulata permette alla rete di raggiungere una zona di bassa energia evitando che essa resti intrappolata in uno stato spurio (che verrebbe poi incorporato nei pesi sinaptici); solo quando essa ha raggiunto lo stato di equilibrio termico, vengono raccolti i dati sullo stato delle unità mentre queste continuano ad aggiornare il proprio valore per un certo numero di cicli. Siccome la temperatura deve essere maggiore di zero (per poter avere le fluttuazioni necessarie al calcolo del prodotto medio), è difficile far sì che i pesi riducano a zero la probabilità di alcuni stati non desiderati (essi dovrebbero crescere all'infinito). Un metodo per evitare questa situa-

zione di possibile difficoltà consiste nell'aggiungere del rumore ai pattern da far apprendere (invertendo il segno di alcuni componenti in modo casuale) prima di presentarli alla rete: in questo modo si riduce il rischio che alcuni stati assumano probabilità zero nella fase vincolata.

Macchina di Boltzmann: algoritmo

La realizzazione al computer dell'algoritmo richiede alcune approssimazioni e l'algoritmo descritto qui sotto si basa quindi su particolari accorgimenti proposti dagli autori stessi [Ackley, Hinton e Sejnowski 1985; Hinton e Sejnowski 1986] al fine di ridurre il carico computazionale. Consideriamo una rete con S unità visibili e T unità nascoste con connessioni simmetriche e attivazione stocastica binaria dove la funzione sigmoide della somma pesata degli input restituisce la probabilità che l'unità assuma valore 1 (lo stato opposto è 0). La modifica dei pesi sinaptici si basa sul confronto tra la probabilità che le unità pre- e postsinaptiche siano entrambe attive nella fase di presentazione dei pattern (attivazione vincolata) e nella fase di libera evoluzione: definiamo queste probabilità rispettivamente con $P_{ij}(+1)_{\text{vincolate}}$ e $P_{ij}(+1)_{\text{libere}}$. Per poter calcolare queste probabilità bisogna lasciare che la rete raggiunga la fase di equilibrio termico e aggiorni il proprio stato per un certo numero di epoche a una temperatura finale maggiore di zero. Un'epoca è costituita da N aggiornamenti delle unità non vincolate scelte a caso, dove N è il numero di unità non vincolate a un input esterno (quindi nella fase di presentazione dei pattern $N = T$, mentre nella fase di libera evoluzione $N = S + T$): la registrazione della co-attivazione di tutte le coppie di unità avviene solo alla fine dei cicli di un'epoca. Si ricordi inoltre che il raggiungimento della fase di equilibrio termico richiede la ricottura simulata mediante un programma predefinito in cui la rete viene lasciata libera di aggiornare il proprio stato per un certo numero di epoche a ciascuna temperatura. Il programma di ricottura simulata usato in questo esempio è

T	P
20	3
15	5
12	5
10	10 (temperatura finale per la registrazione dei dati)

dove T è la temperatura e P il numero di epoche.

Nell'algoritmo riportato qui sotto i pattern di addestramento vengono presentati con del rumore aggiunto per evitare la crescita infinita dei pesi sinaptici.

Inizializzazione

1. Inizializzare tutti i pesi sinaptici a zero.
2. Scegliere il programma di ricottura simulata, ad esempio come nell'esempio di cui sopra.
3. Fissare il valore di un'epoca a T aggiornamenti casuali per la fase vincolata e a $S + T$ aggiornamenti per la fase di libera evoluzione. Notare

che nella fase vincolata solamente le unità nascoste devono essere aggiornate, mentre nella fase di libera evoluzione tutte le unità possono essere aggiornate.

4. Scegliere la costante di apprendimento: ad esempio, $\eta = 2$.

5. Scegliere il numero di volte C in cui la rete viene portata in equilibrio termico per ciascun pattern di addestramento (ad esempio 2).

Apprendimento

Ripetere la procedura seguente fino a quando i pesi sinaptici cessano di crescere.

a) Fase vincolata

1. Per ciascun pattern del gruppo di addestramento, eseguire C volte i passi 2-5.

2. Invertire il valore di ciascun componente del pattern con una piccola probabilità (ad esempio 0,1); applicare il pattern modificato alle unità visibili; inizializzare le unità nascoste in modo casuale a 0 o 1.

3. Iniziando con la temperatura più alta, aggiornare solamente le unità non-vincolate (ovvero le unità nascoste) per il numero di epoche previsto dal programma di raffreddamento simulato (1 epoca = T aggiornamenti di unità nascoste scelte a caso).

4. Abbassare la temperatura e continuare l'aggiornamento delle unità nascoste in base al programma di ricottura.

5. Giunti alla temperatura finale, aggiornare le unità nascoste per il numero di epoche previsto dal programma e calcolare la probabilità $P_{ij}(+1)_{\text{vincolate}}$ che ciascuna coppia possieda lo stesso valore 1. Questa probabilità si ottiene sommando i prodotti delle attivazioni delle unità alla fine di ogni epoca e dividendo la somma ottenuta per il numero di epoche.

6. Calcolare la media di $P_{ij}(+1)_{\text{vincolate}}$ sulle C ripetizioni.

7. Calcolare la media di $P_{ij}(+1)_{\text{vincolate}}$ su tutti i pattern di addestramento.

b) Fase di libera evoluzione

1. Per lo stesso numero di volte della fase vincolata (numero di pattern di addestramento per C ripetizioni ciascuno), eseguire i passi 2-5.

2. Inizializzare tutte le unità della rete a valori casuali 0 o 1.

3. Iniziando con la temperatura più alta, aggiornare le unità non-vincolate (ovvero tutte le unità della rete) per il numero di epoche previsto dal programma di ricottura simulata (1 epoca = $S + T$ aggiornamenti di unità scelte a caso).

4. Abbassare la temperatura e continuare l'aggiornamento delle unità in base al programma di ricottura.

5. Giunti alla temperatura finale, continuare ad aggiornare le unità per il numero di epoche previsto dal programma e calcolare la probabilità $P_{ij}(+1)_{\text{libere}}$ che ciascuna coppia possieda lo stesso valore 1. Questa probabilità si ottiene sommando i prodotti delle attivazioni delle unità alla fine di ogni epoca e dividendo la somma ottenuta per il numero di epoche.

6. Calcolare la media di $P_{ij}(+1)_{\text{libere}}$ su tutte le ripetizioni.

Modifica dei pesi sinaptici

Scegliere uno dei due sistemi seguenti e impiegare sempre lo stesso.

$$\text{A.}\quad \Delta w_{ij} = \eta(P_{ij}(+1)_{\text{vincolate}} - P_{ij}(+1)_{\text{libere}})$$

$$\text{B.}\quad \Delta w_{ij} = \begin{cases} +\eta & \text{se } P_{ij}(+1)_{\text{vincolate}} > P_{ij}(+1)_{\text{libere}} \\ -\eta & \text{se } P_{ij}(+1)_{\text{vincolate}} < P_{ij}(+1)_{\text{libere}} \\ 0 & \text{altrimenti} \end{cases}$$

L'approssimazione discreta B presenta il vantaggio di modificare i pesi sinaptici anche in zone relativamente piatte della funzione di energia.

Si noti infine che l'algoritmo della Macchina di Boltzmann applicato a una rete di Hopfield (la quale non possiede unità nascoste) risulta notevolmente semplificato poiché non è necessario misurare la probabilità di co-attivazione in equilibrio termico durante la fase vincolata: infatti, siccome tutte le unità della rete sono occupate dai pattern di addestramento, è sufficiente calcolare la media di co-attivazione delle unità per ciascun pattern di addestramento un'unica volta. Questo valore viene quindi confrontato di volta in volta con il valore ottenuto nella fase di libera evoluzione.

Hinton e Sejnowski [1986] suggeriscono di aggiungere a ciascun peso un piccolo termine di decadimento verso zero proporzionale al valore assoluto del peso sinaptico (ad esempio lo 0,5%). In questo modo i pesi sinaptici rimangono piccoli e di conseguenza l'energia della rete si mantiene intorno a valori bassi creando meno ostacoli alla ricerca del minimo globale.

3.1. Approssimazione dell'attività media

La complessità dell'algoritmo della Macchina di Boltzmann è dovuta alla necessità di tenere in considerazione le fluttuazioni delle unità in un regime di attivazione stocastica. Peterson e Anderson [1987] hanno suggerito di approssimare la correlazione media dell'attività di due unità i e j con il prodotto della loro attivazione media

$$\langle x_i x_j \rangle \approx m_i m_j$$

dove

$$m_i = \langle x_i \rangle = \tanh\left(\sum_j w_{ij} m_j \right)$$

è l'attivazione media di un'unità calcolata sulla base dell'attivazione media delle altre unità della rete (si veda il paragrafo 5.1 del secondo capitolo). Questa approssimazione è detta «Mean Field Theory» (teoria del campo medio) perché identifica il contributo di tutte le singole unità della rete con il valore medio della loro attività complessiva. In base al Teorema del Limite Centrale questa approssimazione diventa esatta quando il numero di unità nella rete tende all'infinito, ma in pratica funziona correttamente anche con poche unità.

L'algoritmo resta identico nella struttura, ma i calcoli risultano semplificati notevolmente ed è anche possibile ottenere una grande accelerazione dell'apprendimento. Le unità vincolate vengono fissate ai valori corrispondenti ai pattern di addestramento, ma durante il processo di rilassamento non è più necessario simulare l'attivazione stocastica delle unità. Inoltre, il passo 5 dell'algoritmo (sia nella fase vincolata che in quella in libera evoluzione) non richiede la media su un gran numero di epoche; una volta che la rete è stata portata in equilibrio termico alla temperatura finale, è sufficiente calcolare il prodotto

$$m_i m_j$$

e farne la media su tutti i pattern di addestramento e su tutti i corrispondenti cicli di rilassamento nella fase di evoluzione libera. La correzione dei pesi sinaptici è analoga alla versione esatta

$$\Delta w_{ij} = \eta((m_i m_j)_{\text{vincolate}} - (m_i m_j)_{\text{libere}})$$

1. Introduzione

La maggior parte dei modelli considerati nei capitoli precedenti apprendono mediante la guida di un insegnante esterno che fornisce la risposta desiderata per ogni vettore di input. Negli algoritmi di rinforzo l'apprendimento viene guidato da un indice di valutazione della risposta della rete neurale. In senso lato anche l'apprendimento autoassociativo può essere considerato come un esempio di apprendimento supervisionato perché la rete è forzata ad associare ogni pattern con se stesso. Sebbene i modelli ad apprendimento supervisionato siano molto efficaci dal punto di vista computazionale, bisogna considerare il fatto che i sistemi nervosi biologici sono in grado di apprendere anche senza la guida di un insegnante esterno. Ad esempio, sebbene l'area visiva V1 della corteccia striata riceva segnali solamente dalla retina attraverso le fibre provenienti dal Nucleo Genicolato Laterale[1], vi è un evidente processo di sviluppo della struttura interna e delle caratteristiche di risposta dei singoli neuroni. È comunque lecito chiedersi anche per le aree corticali successive e per gli altri componenti del sistema nervoso centrale in che misura l'apprendimento sia dovuto a un meccanismo supervisionato (in qualche modo simile a quelli analizzati nei capitoli precedenti) e in che misura sia invece dovuto a dei meccanismi di auto-organizzazione. Probabilmente la risposta non è univoca e comunque non generale, ma l'assunzione dell'apprendimento supervisionato richiede una serie di caratteristiche non ancora supportate dalle conoscenze neurofisiologiche in nostro possesso, ovvero che ciascun neurone sia in grado di distinguere tra il segnale di input e il segnale di apprendimento (risposta desiderata, rinforzo, fase di attività vincolata, ecc.) oppure che vi siano fibre nervose distinte e specializzate nel trasporto di segnali correttivi.

[1] Essa è l'unica area della corteccia che non riceve connessioni di feedback da moduli corticali non-visivi ed è quindi improbabile che un paradigma di apprendimento supervisionato riesca a coglierne le caratteristiche di sviluppo.

In attesa di ulteriori scoperte sulle proprietà anatomiche e neurofisiologiche del sistema nervoso, è utile indagare da un punto di vista computazionale le proprietà di un sistema artificiale in grado di auto-organizzarsi solamente in base alle caratteristiche dei segnali di input. L'apprendimento auto-organizzato caratterizza la modalità di sviluppo dei modelli descritti in questo capitolo. Essi possiedono alcune proprietà generali.

1. Siccome lo studio dei sistemi neurali artificiali ad auto-organizzazione scaturisce principalmente dal desiderio di comprendere le caratteristiche computazionali del cervello, le regole di apprendimento e le dinamiche interne che governano la rete tendono ad essere semplici e più plausibili dal punto di vista biologico rispetto a quelle impiegate per i modelli descritti nei capitoli precedenti. In generale le regole di apprendimento sono di tipo hebbiano, le attivazioni delle unità sono lineari o a gradino e la rete è composta da un solo strato di sinapsi.

2. L'apprendimento consiste nell'estrazione dell'informazione che meglio descrive i pattern di input; questa informazione consiste in caratteristiche comuni o distintive dei pattern di input che permettono al modello di sviluppare rappresentazioni compatte o di eseguire classificazioni senza bisogno di una guida esterna. Le operazioni che caratterizzano gli apprendimenti non supervisionati si basano sul calcolo di indici di correlazione, di parametri della distribuzione di probabilità dei pattern di ingresso (medie, varianze), di analisi dei componenti principali, ecc.; ne consegue che i pattern di ingresso devono essere ridondanti affinché il modello neurale sia in grado di estrarre la struttura rilevante. Se l'informazione usata per addestrare la rete non è ridondante, non è possibile estrarne la struttura sottostante e quindi non vi è apprendimento e sviluppo di rappresentazioni. Secondo Horace Barlow, uno dei ricercatori più attivi in questo settore di ricerca, la ridondanza è alla base dello sviluppo della conoscenza [Barlow 1989]. Una conseguenza del fatto che questi modelli estraggono informazione rilevante da un insieme di configurazioni ridondanti consiste in architetture con un numero di unità di output generalmente minore del numero di unità di input.

3. La maggior parte dei modelli viene impiegata per lo studio dell'elaborazione dell'informazione sensoriale e in particolar modo dell'informazione visiva; siccome molte caratteristiche delle aree corticali visive sono ben note, è possibile effettuare confronti realistici con le proprietà esibite dai modelli artificiali.

Questo capitolo è organizzato in tre parti (oltre a questo paragrafo introduttivo). Nella prima parte verranno prese in esame alcune semplici variazioni della regola di Hebb che danno luogo a interessanti proprietà di auto-organizzazione e sviluppano profili di risposta neurale simili a quelli dei neuroni biologici. Nella parte successiva verrà descritto un approccio diverso basato sulla Teoria dell'Informazione: in questo contesto i neuroni apprendono a massimizzare una determinata definizione del contenuto informativo dell'input mediante regole

essenzialmente hebbiane. La terza parte offrirà infine una famiglia di modelli e algoritmi di apprendimento basati su meccanismi di competizione di attività che sono stati spesso riscontrati nei sistemi nervosi biologici.

2. Estrazione di informazione con apprendimento hebbiano

Data una certa distribuzione[2] di pattern di input $\mathbf{x}^{\mu}$, un modello con una sola unità di output lineare potrebbe sviluppare un semplice indice descrittivo dei dati e usare questa capacità per indicare quanto ogni vettore sia rappresentativo dell'intera distribuzione. Ad esempio, se questo indice fosse la direzione di massima varianza della distribuzione, l'output dell'unità dovrebbe diventare tanto maggiore quanto più un pattern è «rappresentativo» della distribuzione. Un'unità con queste caratteristiche indicherebbe così la «familiarità» di un nuovo pattern di input rispetto ai pattern visti durante il periodo di addestramento.

È facile osservare che la semplice regola di Hebb impiegata in una rete neurale con una sola unità di output lineare è in grado di sviluppare questo tipo di capacità. La modifica dei pesi sinaptici è data da

$$\Delta w_i = \eta y x_i$$

dove η è il tasso di apprendimento e l'attivazione dell'unità di output è data dalla somma pesata degli input

$$y = \sum_i w_i x_i$$

Si noti innanzitutto che se un componente del vettore di input attiva l'unità di output, la sinapsi corrispondente sarà rafforzata e la volta successiva che lo stesso componente di input sarà presente l'unità postsinaptica si attiverà ancora maggiormente: questo fenomeno – definito auto-amplificazione – è una proprietà generale dei sistemi ad auto-organizzazione [von der Malsburg 1990]. In secondo luogo, le connessioni provenienti dalle unità di input più frequentemente attive saranno rafforzate maggiormente e provocheranno una risposta più marcata dell'unità di output: quindi, se i pattern vengono estratti casualmente dall'ambiente (ad esempio attraverso il campionamento di un sistema visivo artificiale), la rete imparerà a rispondere maggiormente ai pattern che presentano i componenti più frequenti. Il proble-

[2] Il concetto di distribuzione deriva dal linguaggio usato in teoria della probabilità. Nel caso discreto, una distribuzione di pattern indica una collezione di vettori la cui distribuzione nello spazio vettoriale è descrivibile in base a qualche indice descrittivo (ad esempio la media, la varianza, ecc.).

ma è che queste due proprietà fanno sì che le sinapsi della rete tenderanno a rafforzarsi all'infinito per cui la rete non raggiungerà mai un punto di stabilità. Un semplice modo per contenere la crescita dei pesi sinaptici consiste nel normalizzare i valori sinaptici dopo ogni correzione in modo che il vettore mantenga norma (lunghezza) 1; ad esempio, ciascun peso può essere ricalcolato come segue

$$w_j = \frac{w_j}{\sqrt{w_1^2 + w_2^2 + \ldots + w_n^2}}$$

in questo modo, siccome le risorse delle sinapsi sono limitate (da un lato tendono a crescere ma dall'altro sono vincolate dal limite della somma totale di tutti i loro valori), si instaura un meccanismo di competizione per cui la regola converge verso una soluzione stabile in cui l'output dell'unità esprime la direzione di massima varianza della distribuzione degli input (il vettore sinaptico giace sull'asse principale della distribuzione dei pattern). Questo risultato significa due cose importanti: *a*) la varianza dell'attivazione dell'unità $\langle y^2 \rangle$ viene massimizzata (mentre all'inizio dell'apprendimento l'unità emette sempre la stessa risposta per tutti i pattern, gradualmente essa diversifica la propria risposta per i vari pattern) e *b*) i pattern che si collocano lungo la direzione di massima varianza della distribuzione elicitano una risposta maggiore. Il vettore dei pesi sinaptici dell'unità durante l'apprendimento si sposta verso la direzione dell'autovettore corrispondente all'autovalore massimo della matrice di correlazione **C** dei pattern di input (si veda l'appendice sull'algebra lineare) [Haykin 1999; Hertz, Krogh e Palmer 1991; Diamantaras e Kung 1996]

$$\mathbf{C} = \langle \mathbf{x}\mathbf{x}^T \rangle_\mu$$

i cui singoli componenti sono dati da $\mathbf{C}_{ij} = \langle x_i x_j \rangle_\mu$

2.1. La regola di Oja: estrazione della prima componente principale

La normalizzazione proposta nel paragrafo precedente è efficace ma presenta lo svantaggio di non essere locale perché il calcolo di ciascuna sinapsi richiede la conoscenza dei valori di tutte le altre sinapsi. Affinché una regola sinaptica sia locale, la modifica di ogni peso deve essere funzione di variabili disponibili solamente a livello presinaptico e postsinaptico[3]. Oja [1982] ha suggerito una semplice modifica alla

[3] I sistemi nervosi biologici sono interessanti anche perché le capacità che essi possiedono emergono dall'interazione puramente locale di molti semplici processi (trascurando gli effetti ancora non chiari da un punto di vista computazionale dei neuromodulatori). La caratteristica locale del calcolo neurale viene solitamente mantenuta nei modelli ad auto-organizzazione.

regola di Hebb che può essere applicata alla classe di modelli descritti nel paragrafo precedente: con la nuova regola il vettore di pesi sinaptici tende naturalmente verso lunghezza 1 senza necessità di rinormalizzazioni globali. La regola di Oja prevede una ridefinizione del componente presinaptico in funzione della capacità di trasmissione corrente della sinapsi

$$\Delta w_i = \eta y(x_i - w_i y)$$

dove il termine $-w_i y$ – talvolta definito anche «fattore di dimenticanza» – limita la crescita infinita del peso sinaptico in funzione del proprio valore corrente e possiede anche rilevanza neurofisiologica [Stent 1973]. Come nel caso della normalizzazione esplicita descritta nel paragrafo precedente, è possibile mostrare che la regola di Oja produce un vettore di pesi sinaptici corrispondente all'autovettore dell'autovalore massimo della matrice di correlazione degli input. Anche in questo caso la varianza dell'output $\langle y^2 \rangle$ viene massimizzata rispetto alla distribuzione dei pattern di input e l'unità può essere impiegata per discriminare tra pattern appartenenti alla distribuzione di addestramento e pattern tratti da distribuzioni diverse (purché entrambe le distribuzioni possiedano la stessa media).

Se i pattern di input possiedono media zero, l'output dell'unità rappresenta la prima componente principale della distribuzione dei pattern. L'Analisi delle componenti principali è un metodo che viene usato in statistica per rappresentare in modo compatto e informativo le proprietà di una distribuzione di dati [Jolliffe 1986]. Esso consiste nell'individuare una serie di vettori ortogonali fra loro nella direzione della massima varianza residua della distribuzione. La prima componente principale è un vettore collocato nella direzione di massima varianza della distribuzione; la seconda componente è un vettore ortogonale rispetto al precedente che esprime la direzione di massima varianza residua; la terza componente è ortogonale rispetto ai primi due ed esprime anch'essa la direzione di massima varianza residua; e così via. Questo procedimento permette di rappresentare in modo estremamente compatto una distribuzione composta di molti vettori. Ciascuna componente principale è data in ordine dall'autovettore dell'autovalore corrispondente (la prima componente principale corrisponde all'autovalore più grande, la seconda componente all'autovalore immediatamente seguente, ecc.) della matrice di covarianza **R** dei pattern

$$\mathbf{R}_{ij} = \langle (x_i - m_i)(x_j - m_j) \rangle_\mu$$

dove $m_i = \langle x_i \rangle$ è il valore medio della componente i-esima. Nel caso di distribuzioni centrate sullo zero, le componenti principali sono gli autovettori della matrice di correlazione **C**. La regola di Oja estrae dunque la prima componente principale della distribuzione di input (fig. 4.1).

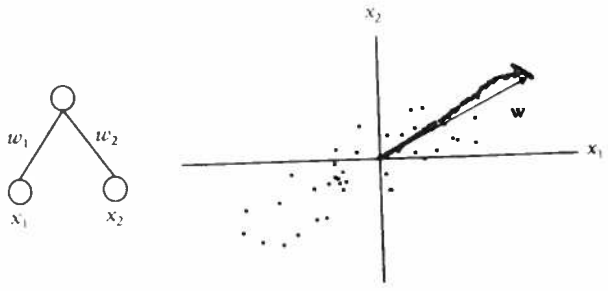


FIG. 4.1. Apprendimento della prima componente principale della distribuzione di input. La rete è stata addestrata per 1.000 cicli con pattern estratti casualmente da un gruppo di 40; il tasso di apprendimento è 0,1. Sinistra: architettura della rete. Destra: grafico raffigurante i pattern di addestramento (puntini) e l'evoluzione dei pesi sinaptici; la direzione e lunghezza del vettore finale è indicata dalla linea con la freccia. Esso si colloca lungo la direzione di massima varianza della distribuzione.

2.2. Estrazione di componenti principali in reti con *N* unità di output

L'estrazione della prima componente principale con una semplice rete neurale rappresenta un risultato interessante, ma è di scarsa utilità applicativa e non rispecchia certo la ricchezza di risorse a disposizione dei sistemi nervosi biologici. Oja stesso [Oja 1989] ha successivamente modificato la propria regola in modo che potesse essere applicata in reti con N unità di output disposte su un unico strato (fig. 4.2a). L'apprendimento delle connessioni di un neurone i avviene sottraendo all'attivazione dell'unità presinaptica l'attivazione pesata di tutte le N unità di output collegate a quella stessa unità di input

$$\Delta w_{ij} = \eta y_i \left(x_j - \sum_{k=1}^{N} w_{kj} y_k \right)$$

dove l'indice i e k puntano entrambi alle unità di output. Oja ha definito questo modello con il nome di «rete del sottospazio» perché i pesi sinaptici delle unità convergono verso un insieme di vettori che giacciono nello stesso sottospazio vettoriale definito dagli autovettori corrispondenti ai primi N autovalori principali della matrice di correlazione dell'input (fig. 4.2b).

Sanger [Sanger 1989] ha proposto una regola simile che, applicata a una rete con N unità lineari (fig. 4.2a), è in grado di estrarre in ordine le prime N componenti principali della distribuzione dell'input

$$\Delta w_{ij} = \eta y_i \left(x_j - \sum_{k=1}^{i} w_{kj} y_k \right)$$

la differenza rispetto alla regola di Oja per N unità sta nella sommato-

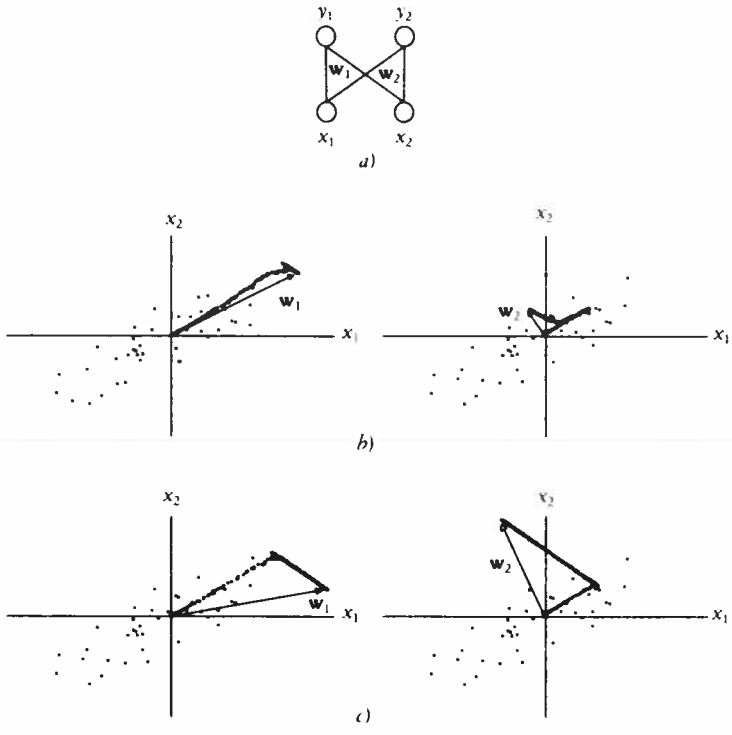


FIG. 4.2. Estrazione di componenti principali della distribuzione di input. *a)* Architettura di una rete con due unità di output; *b)* evoluzione dei pesi sinaptici quando la rete è addestrata con la regola di Oja per unità multiple; *c)* evoluzione dei pesi sinaptici quando la rete è addestrata con la regola di Sanger. I pattern di addestramento sono rappresentati dai puntini; il vettore finale dei pesi sinaptici di ciascuna unità è indicato dalla freccia.

ria: per ciascuna unità *i* di output la regola di Sanger sottrae all'attivazione dell'unità presinaptica l'attivazione di tutte le unità *precedenti* (oltre all'attivazione dell'unità stessa *i*) collegate a quella stessa unità di input. Il risultato di questa operazione è tale per cui ciascuna unità viene forzata a sviluppare un vettore di pesi sinaptici ortogonale ai vettori delle unità precedenti e nel contempo a massimizzare la varianza residua della distribuzione. L'inerente sequenzialità della modifica sinaptica fa sì che la prima unità di output estragga la prima componente principale, la seconda unità la seconda componente, la terza la terza, e così via (fig. 4.2c). L'algoritmo di Sanger però introduce delle piccole graduali distorsioni nell'estrazione delle componenti successive perché, malgrado i pesi sinaptici di tutte le unità vengano sempre modificati contemporaneamente e in parallelo, le unità successive dello strato di output devono «attendere» che le unità precedenti abbiano estratto la componente principale corrispondente prima di poter estrarre l'informazione residua. Il modello risulta più efficace quando

si vogliono scoprire solo poche componenti principali di distribuzioni ad alta dimensionalità.

Sia la regola di Oja che quella di Sanger per N unità di output sono equivalenti alla regola originale di Oja quando N è uguale a 1. Negli altri casi la regola di Oja non introduce distorsioni nell'estrazione dei vettori del sottospazio delle N componenti principali, ma l'output delle singole unità non è facilmente interpretabile come nel caso della regola di Sanger.

Estrazione di componenti principali: algoritmi di Oja e di Sanger

Consideriamo una rete con R unità di input ed N unità di output ad attivazione lineare (fig. 4.2a). Le unità di input possono assumere valori positivi e negativi, così come le unità di output. (È interessante fare esperimenti con architetture parzialmente connesse, ad esempio con unità che possiedono campi recettivi parzialmente sovrapposti.)

1. Inizializzare i pesi sinaptici a piccoli valori casuali e impiegare un basso valore per il tasso di apprendimento; ad esempio,

$$w_{ij} = rnd(\pm 0{,}01)$$
$$\eta = 0{,}05$$

2. Estrarre casualmente un pattern dalla distribuzione e calcolare l'attivazione delle unità di output

$$y_i = \sum_j w_{ij} x_j$$

3a. Regola di Oja: calcolare la modifica dei pesi sinaptici per ciascuna unità i

$$\Delta w_{ij} = \eta y_i \left(x_j - \sum_{k=1}^{N} w_{kj} y_k \right)$$

dove la variabile w_{kj} si riferisce al valore della connessione tra l'unità di output k e l'unità di input j, se questa esiste; k assume valori da 1 a N compresi.

3b. Regola di Sanger: calcolare la modifica dei pesi sinaptici per ciascuna unità progressivamente

$$\Delta w_{ij} = \eta y_i \left(x_j - \sum_{k=1}^{i} w_{kj} y_k \right)$$

dove la variabile w_{kj} si riferisce al valore della connessione tra l'unità di output k e l'unità di input j, se questa esiste; k assume valori da 1 a i compresi, dove i è l'unità di cui si stanno correggendo i pesi sinaptici.

4. Aggiornare il valore dei pesi sinaptici

$$w_{ij} = w_{ij}^{t-1} + \Delta w_{ij}$$

5. Ripetere il procedimento di presentazione e correzione sinaptica fino a quando i valori sinaptici cessano di modificarsi.

2.3. Estrazione di componenti principali con metodi di decorrelazione

I due algoritmi descritti sopra estraggono informazione «riestimando» il valore dell'input in base all'attivazione delle unità di output; gli stessi risultati possono essere ottenuti anche con algoritmi di «decorrelazione» [Becker e Plumbley 1993] che possiedono connessioni laterali fra le unità di output.

Földiák [1989] ha proposto l'impiego di una rete con N unità lineari interamente connesse tra di loro (tranne che per la connessione autoricorrente che rimane zero) le quali ricevono input da R unità

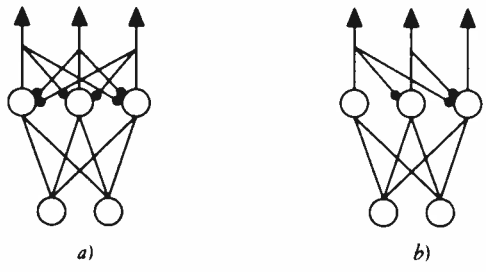


FIG. 4.3. Architetture per l'estrazione dei componenti principali con metodi di decorrelazione: *a*) Földiák, *b*) Rubner e Tavan. Le connessioni laterali tra le unità di output sono indicate dalle linee con il cerchietto nero finale.

d'ingresso (fig. 4.3a). L'output di ciascuna unità è dato da

$$y_i = \sum_{j}^{R} w_{ij} x_j + \sum_{k \neq i}^{N} v_{ik} y_k$$

dove w_{ij} sono le connessioni dalle unità di input e v_{ik} sono le connessioni laterali tra le unità di output; l'attivazione delle unità deve essere ricalcolata più volte per lo stesso input fino a quando essa si stabilizza. Mentre i pesi sinaptici w_{ij} che provengono dall'input vengono modificati in base alla regola di Oja per una singola unità, i pesi sinaptici laterali v_{ik} vengono modificati in base a una regola «anti-hebbiana»

$$\Delta v_{ik} = -\eta y_i y_k, \quad k \neq i$$

che tenta di decorrelare l'attivazione delle unità. I pesi sinaptici laterali possono essere stabilizzati impiegando un normale processo di norma-

lizzazione globale oppure aggiungendo il consueto termine di dimenticanza all'attività presinaptica (quest'ultima versione viene impiegata nella descrizione dell'algoritmo data nel prossimo paragrafo). La rete di Földiák converge verso una soluzione analoga alla regola di Oja per N unità in cui i vettori di pesi sinaptici sono ortogonali fra di loro e generano lo stesso sottospazio degli autovettori corrispondenti ai primi N autovalori della matrice di correlazione dei vettori di input; Leen [1991] ha suggerito una lieve modifica che permette ai pesi di convergere verso gli autovettori stessi anche se essi non vengono rappresentati in ordine come nella rete di Sanger.

Rubner e Tavan [1989] hanno proposto un modello simile dove però ogni unità possiede connessioni laterali provenienti solamente dalle unità precedenti (fig. 4.3b). L'attivazione delle unità è data da

$$y_i = \sum_j^R w_{ij} x_j + \sum_k^{i-1} v_{ik} y_k$$

dove la differenza rispetto al modello di Földiák sta unicamente nella seconda sommatoria che impiega i contributi delle unità precedenti all'unità i-esima; anche questo modello richiede la stabilizzazione dell'attivazione per ciascun pattern prima di procedere al calcolo della correzione dei pesi sinaptici. La modifica dei pesi sinaptici viene calcolata esattamente come nel modello di Földiák. La prima unità della rete estrae la prima componente principale della distribuzione perché non è influenzata dalle altre unità; le unità successive estraggono le componenti principali successive in ordine, come nel modello di Sanger (un modello simile è stato proposto anche da Kung e Diamantaras [1990]).

Quando l'input di una rete è composto da una combinazione di segnali indipendenti, i metodi lineari di estrazione delle componenti principali non sono in grado di separare le sorgenti di segnale. Questo è un problema abbastanza grave perché tale situazione si presenta comunemente, ad esempio a livello acustico dove il segnale che giunge alla coclea è una combinazione di diversi segnali indipendenti (un parlante su uno sfondo musicale, oppure due parlanti, ecc.). Jutten e Herault [1991a] hanno impiegato un modello molto simile alla rete di Földiák per separare le sorgenti di segnale mescolate nell'input della rete. L'output delle unità è dato da

$$y_i = I_i - \sum_{k \neq i}^{N} v_{ik} y_k$$

dove I_i è il segnale esterno proveniente all'unità i-esima. La modifica delle connessioni laterali è hebbiana

$$\Delta v_{ik} = \eta \Phi(y_i) \Psi(y_k)$$

dove Φ e Ψ indicano due funzioni non-lineari dell'attivazione dell'unità. Gli autori hanno ottenuto buoni risultati in vari problemi di separazione impiegando $\Phi(x) = x^3$, $\Psi(x) = \tan^{-1}(x)$ [Jutten e Herault 1991b].

Un altro approccio che consente di superare alcune delle limitazioni collegate alla linearità implicita nelle tecniche di estrazione delle componenti principali è quello della «kernel-based principal component analysis» [Schölkopf, Smola e Müller 1998; Müller, Mika, Rätsch, Tsuda e Schölkopf 2001; Haykin 1999].

Tale approccio si basa sull'impiego di una trasformazione non-lineare dello spazio dei pattern di ingresso, che precede l'estrazione delle componenti principali. Le operazioni lineari eseguite sullo spazio trasformato possono così individuare in modo indiretto delle regolarità presenti nello spazio dei pattern di ingresso che non sarebbero rivelabili operando direttamente un'analisi delle componenti principali in tale spazio.

Estrazione di componenti principali: algoritmi di Földiák e di Rubner e Tavan

Consideriamo una rete con R unità di input e N unità di output ad attivazione lineare connesse lateralmente da pesi sinaptici v_{ik} come in figura 4.3a o 4.3b senza connessioni autoricorrenti. Le unità di input possono assumere valori positivi e negativi, così come le unità di output.

1. Inizializzare i pesi sinaptici w_{ij} e v_{ik} a piccoli valori casuali e impiegare un basso valore per il tasso di apprendimento; ad esempio,

$$w_{ij} = rnd(\pm 0{,}01)$$
$$v_{ik} = rnd(\pm 0{,}01)$$
$$\eta = 0{,}05$$

2. Estrarre un pattern casualmente dalla distribuzione e calcolare ripetutamente l'attivazione delle unità di output fino a quando ciascun output diventa stabile; se si impiega l'architettura di Földiák, l'attivazione è data da

$$y_i = \sum_{j}^{R} w_{ij} x_j + \sum_{k \neq i}^{N} v_{ik} y_k$$

se si impiega l'architettura di Rubner e Tavan, l'attivazione è data da

$$y_i = \sum_{j}^{R} w_{ij} x_j + \sum_{k}^{i-1} v_{ik} y_k$$

3. Calcolare la modifica dei pesi sinaptici w_{ij} dall'input per ogni unità

$$\Delta w_{ij} = \eta y_i (x_j - w_{ij} y_i)$$

e la modifica dei pesi sinaptici laterali v_{ik}

$$\Delta v_{ik} = -\eta y_i (y_k + v_{ik} y_i) \quad \text{dove } k \neq i$$

(quest'ultima equazione incorpora già il fattore di normalizzazione).

4. Aggiornare il valore dei pesi sinaptici

$$w_{ij} = w_{ij}^{t-1} + \Delta w_{ij}$$

$$v_{ik} = v_{ik}^{t-1} + \Delta v_{ik}$$

5. Ripetere il procedimento di presentazione e correzione sinaptica fino a quando i valori sinaptici cessano di modificarsi.

2.4. Autoassociazione con Back-propagation

Un altro approccio all'apprendimento auto-organizzato consiste nell'impiego di un algoritmo supervisionato come Back-propagation per apprendere ad autoassociare una serie di pattern a dimensionalità R attraverso N unità nascoste, dove N di solito è molto minore di R: questo tipo di compito viene talvolta definito «associazione d'identità» (*identity mapping*) e il modello neurale diventa un «autocodificatore» (*autoencoder*). Al fine di memorizzare i pattern in modo compatto, le unità nascoste devono scoprire e rappresentare le caratteristiche importanti della distribuzione di apprendimento. Le rappresentazioni risultanti sono molto simili alle soluzioni individuate dalla rete di Oja per N unità: i vettori sinaptici convergono verso il sottospazio delle prime N componenti principali della distribuzione, ma siccome Back-propagation distribuisce l'errore quadratico medio in modo uguale sulle unità nascoste, esse possiedono egual varianza e i vettori non sono ortogonali fra di loro. Il vantaggio rispetto ai modelli lineari considerati sopra è che Back-propagation può essere teoricamente impiegata anche per scoprire componenti non-lineari dell'input visto che l'algoritmo non è vincolato dall'impiego di unità lineari; tuttavia, non vi sono risultati che siano in grado di provare questa superiorità e, anzi, sembra che questa potenzialità sia annullata dall'estrema suscettibilità dell'algoritmo a cadere in minimi locali [Bourlard e Kamp 1988] in questi tipi di compiti. Zipser [1986] ha applicato questo paradigma a immagini bidimensionali composte da nuvole di punti filtrate attraverso una funzione gaussiana: variando il numero di unità nascoste l'attivazione interna evolve da una rappresentazione distribuita ove ciascuna locazione nello spazio è unicamente determinata da un vettore di attivazione a una rappresentazione in cui ciascuna unità nascosta sviluppa un campo recettivo localizzato a una porzione specifica dello spazio. Zemel e Hinton [1994] hanno proposto alcuni vincoli sulla struttura interna della rete affinché essa sviluppi mappe topografiche dell'input. Back-propagation può rivelarsi anche utile per scoprire una struttura temporale non-lineare in un autocodificatore in cui l'output del sistema consiste nella predizione dell'input all'istante successivo; quando la rete viene addestrata su sequenze di pattern, invece che su coppie temporali isolate, è necessario impiegare architetture ricorrenti (si veda il secondo capitolo).

Uno svantaggio di Back-propagation usato come autocodificatore consiste nella scarsa capacità di completamento di pattern: infatti, di fronte a un pattern di input incompleto la rete tenderà a fornire una risposta «media» invece che convergere verso il pattern più simile (come nel caso delle reti di Hopfield). Inoltre l'efficacia dell'apprendimento decresce con l'aumentare della dimensionalità dei pattern, mentre questo fenomeno non si verifica con i metodi lineari descritti nei paragrafi precedenti.

2.5. Che senso ha impiegare un modello neurale che estrae componenti principali?

Uno dei vantaggi dell'estrazione delle componenti principali consiste nel fatto che si possono ricostruire gli input originali in modo ottimale rispetto all'errore quadratico medio tra l'immagine originale e l'immagine ricostruita [Cottrell, Munro e Zipser 1989]. Dopo che una rete di Oja o di Sanger è stata addestrata, ad esempio con una serie di immagini, il valore ricostruito di ciascun pixel di input è dato dalla somma delle attivazioni delle unità di output per l'immagine da ricostruire pesate per i corrispondenti pesi sinaptici

$$x_j^\mu = \sum_i^N w_{ij} y_i^\mu$$

dove x_j^μ è il valore del pixel che si vuole ricostruire e y_i^μ è l'attivazione dell'i-esima unità di output per l'immagine μ che si desidera ricostruire. Questo procedimento permette di compattare notevolmente i dati originari in quanto è sufficiente memorizzare il vettore di output per ciascuna immagine e la matrice di valori sinaptici. Benché questa tecnica possa essere impiegata sia con la regola di Oja che con la regola di Sanger, quest'ultima è preferibile perché è possibile identificare le singole componenti principali e impiegare di conseguenza maggior memoria (numero di bit) per le prime componenti (l'attivazione e i pesi sinaptici delle prime unità) e meno per gli ultimi; d'altra parte si ricordi che se N è grande, la regola di Sanger tende a produrre risultati degradati per gli ultimi componenti. La precisione della ricostruzione dipende comunque, oltre che dalla complessità della distribuzione, dal numero di unità di output impiegato: se esse sono troppo poche, l'immagine ricostruita tenderà a somigliare alla media della distribuzione mettendone in risalto le caratteristiche principali[4]. Quante più

[4] Un esercizio interessante: si provi ad addestrare la rete di Sanger su una serie di immagini (ad esempio i visi di alcuni amici, avendo cura che lo sfondo sia lo stesso e occupino più o meno lo stesso spazio, come nelle fototessere; è consigliabile usare il logaritmo di ogni pixel che permette alla rete di lavorare sui contrasti piuttosto che sull'intensità luminosa) e poi a ricostruirne alcune impiegando solo le prime unità di out-

unità di output vengono impiegate, tanto maggiore sarà la precisione di ricostruzione delle immagini originali. La capacità di generalizzazione può essere studiata presentando alla rete una nuova immagine (non inclusa tra quelle impiegate per l'apprendimento) e usando l'output delle unità per ricostruire i valori di attivazione dell'input: se l'immagine usata per il test è «simile» (ovvero si colloca nello spazio dell'input a cui appartiene la distribuzione di addestramento) a quelle usate per l'apprendimento, la ricostruzione è molto buona. Le reti di Oja e Sanger non sono gli unici modelli che possono essere impiegati per memorizzare e ricostruire immagini (si veda ad esempio Back-propagation, la rete di Hopfield e la macchina di Boltzmann), ma rispetto agli algoritmi supervisionati considerati nei capitoli precedenti presentano il vantaggio di non essere soggette a fenomeni di interferenza e sono molto più semplici (una proprietà importante per la realizzazione di circuiti neurali su silicio).

Resta però da chiedersi se i sistemi nervosi biologici siano effettivamente interessati a estrarre le componenti principali della stimolazione sensoriale per poi ricostruire gli input originali; siccome le capacità percettive degli organismi biologici sono finalizzate alla capacità di estrarre informazione utile all'esecuzione di azioni e decisioni importanti per la sopravvivenza dell'individuo, il quesito veramente importante è se le componenti principali rappresentino effettivamente una caratteristica utile per le fasi successive di elaborazione di un sistema neurale artificiale oppure del sistema nervoso[5]. Gli algoritmi supervisionati come Back-propagation (che impiegano metodi di discesa del gradiente della funzione di costo) sono lenti e suscettibili a minimi locali quando vi sono molte dipendenze tra i segnali che provengono dai vari canali di input: in questo caso è utile impiegare la rete di Sanger per la pre-elaborazione dei dati perché essa fornisce una rappresentazione in cui l'output di ciascuna unità (che diventa l'input del sistema supervisionato) non è correlato con l'output delle altre unità[6] e anche perché permette una certa riduzione della dimensionalità dei dati (solitamente il numero di unità di output – o componenti – è notevolmente inferiore al numero di unità di input) [Becker 1991]. Inoltre Hancock e collaboratori [Baddeley e Hancock 1991; Hancock, Baddeley e Smith 1992] hanno mostrato che se una rete di Sanger viene addestrata su una serie di immagini naturali («rappresentative» di immagini quotidiane per gli esseri umani), essa sviluppa dei campi recettivi simili a quelli dei neuroni ritrovati nelle aree visive primarie: le prime componenti principali corrispondono a cellule centro-contorno e a cel-

put; si aggiungano gradualmente poi i contributi dalle unità successive notando quante componenti principali sono necessarie affinché ciascun viso diventi riconoscibile.

[5] Torneremo in maggior dettaglio su questo aspetto della «validità ecologica» dei modelli neurali nel prossimo capitolo.

[6] Questo vantaggio non è però offerto dalla rete di Oja che sviluppa vettori di pesi sinaptici ortogonali, ma non necessariamente decorrelati.

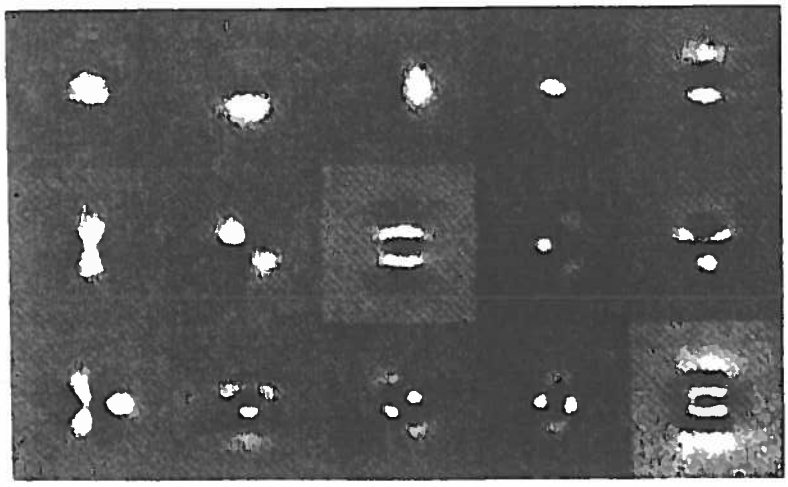


FIG. 4.4. Campi recettivi delle prime 15 unità di una rete di Sanger addestrata su una serie di immagini naturali.

Fonte: Hancock *et al.* [1992].

lule per la detenzione di barre orientate (fig. 4.4). Tuttavia la regola di Oja per N unità sembra essere un meccanismo di computazione più plausibile perché è abbastanza improbabile che il sistema nervoso estragga le componenti principali in ordine come nella regola di Sanger. Questo approccio, che caratterizza tutti i modelli neurali il cui scopo consiste nella riduzione della dimensionalità ed estrazione dell'informazione, è stato però criticato da Field [1994] il quale sostiene che esso è insufficiente per la comprensione delle proprietà dei campi recettivi dei neuroni delle aree visive; Field suggerisce che il sistema nervoso impiega un codice distribuito e «sparso» in cui ciascun neurone possiede una probabilità media di attivarsi approssimativamente uguale per tutti i pattern della distribuzione, ma che questa probabilità è bassa per ogni singola immagine. In quest'ottica la ridondanza dell'input non viene eliminata, ma viene piuttosto trasformata in una ridondanza della configurazione di attivazione dei neuroni. Un'ulteriore critica ai modelli che estraggono le componenti principali riguarda il fatto che raramente essi sviluppano unità sensibili a configurazioni di diversa frequenza spaziale quando invece sappiamo che il sistema nervoso dei mammiferi fa largo uso di filtri a diverse scale spaziali nelle prime fasi di elaborazione dell'immagine sensoriale [Watt 1991].

2.6. Il principio di massima selettività

Vi sono molti altri modelli di auto-organizzazione basati su semplici regole hebbiane che non estraggono le componenti principali della

distribuzione dei segnali di input. Fra questi ne descriviamo qui uno che dimostra comportamenti complessi e gode di un notevole supporto biologico: la regola BCM (dal nome degli autori Bienenstock, Cooper e Munro [1982]). L'idea di base consiste nel definire l'utilità di un neurone in funzione della sua «selettività»: un neurone maturo dovrebbe rispondere fortemente per alcuni pattern specifici, ma non per altri. Questo concetto può essere matematicamente tradotto in una misura di selettività

$$S(y_i) = 1 - \frac{\langle y_i \rangle}{\max(y_i)}$$

dove $\langle y_i \rangle$ è l'attività media dell'unità rispetto alla distribuzione dei segnali d'ingresso. In base a questa misura un neurone ottimale fornirebbe una risposta massima per uno specifico pattern e una risposta media molto bassa per gli altri pattern della distribuzione. Gli autori hanno quindi proposto una famiglia di regole hebbiane che sviluppano proprietà di risposta delle singole unità come richiesto dal principio di massima selettività: queste regole modificano i pesi sinaptici in base all'attività presinaptica x_j e a una funzione Φ dell'attività postsinaptica

$$\Delta w_{ij} = \eta x_j \Phi(y_i, \langle y_i \rangle) - \varepsilon w_{ij}$$

dove εw_{ij} causa un decadimento esponenziale dei pesi verso zero in assenza di input. La funzione Φ deve essere scelta in modo che il peso sinaptico diventi stabile quando l'unità ha raggiunto un punto di massima selettività. Essa deve quindi possedere le seguenti proprietà

$$sgn[\Phi(y_i, \langle y_i \rangle)] = sgn\left[y_i - \left(\frac{\langle y_i \rangle}{y_0} \right)^p \langle y_i \rangle \right] \quad \text{se } y_i > 0$$

$$\Phi(0, \langle y_i \rangle) = 0 \quad \text{altrimenti}$$

dove y_0 e p sono due costanti positive e le unità possono assumere solo un'attività positiva. Il termine $\vartheta_i = (\langle y_i \rangle / y_0)^p \langle y_i \rangle$ rappresenta una «soglia mobile» del neurone in quanto si sposta durante lo sviluppo delle caratteristiche di risposta dell'unità (fig. 4.5).

Il processo di crescita sinaptica segue un andamento tipico. Inizialmente l'attività media dell'unità è molto inferiore alla costante y_0 ($\langle y_i \rangle << y_0$) e quindi la funzione $\Phi(y_i, \langle y_i \rangle)$ genera valori positivi per tutti gli input aumentando la risposta media dell'unità. Con la crescita di $\langle y_i \rangle$ anche la soglia mobile ϑ_i si sposta: ora alcuni pattern di input provocano una risposta che supera la soglia mentre altri (quelli lontani dai valori sinaptici, ovvero ortogonali ad essi) producono un'attivazione inferiore ad essa. I pesi sinaptici vengono ora modificati in modo tale che la risposta per i primi pattern continui a crescere, mentre la risposta per questi ultimi pattern diminuisca. Questo fenomeno viene

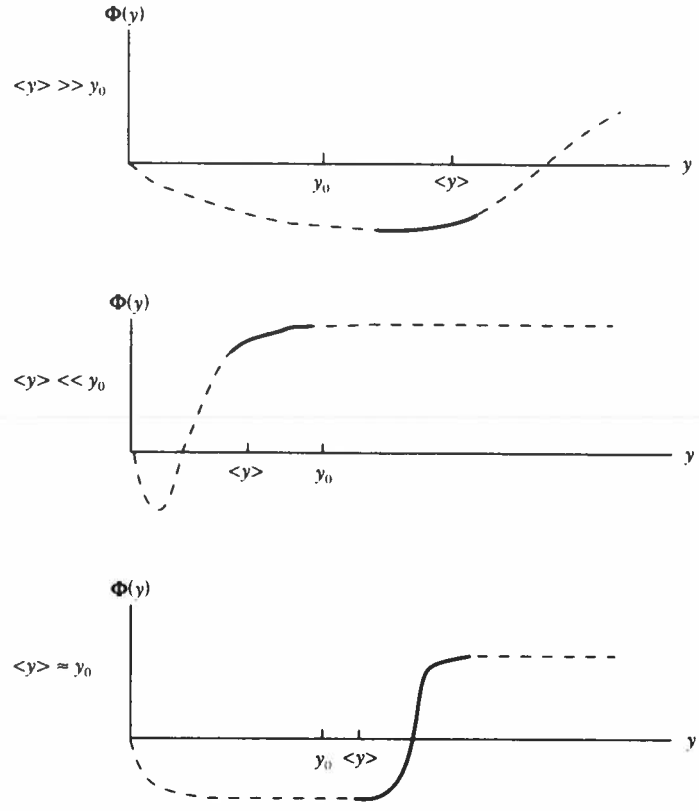


FIG. 4.5. Una funzione che soddisfa le proprietà richieste per il cambiamento sinaptico nella regola BCM. Durante l'apprendimento le condizioni di attivazione media dell'unità postsinaptica cambiano e di conseguenza la modifica sinaptica assume un andamento diverso.

Fonte: Bienenstock, Cooper e Munro [1982].

descritto dagli autori come un processo di «competizione temporale» tra i pattern. La risposta per i pattern sfavoriti decresce gradualmente verso zero dove la rete diventa stabile perché $\Phi(0,\langle y_i \rangle) = 0$. Gli autori hanno applicato il modello a distribuzioni di pattern caratterizzate da linee orientate in varie direzioni: ciascuna unità della rete sviluppa una selettività massima per linee di una certa orientazione. Questi risultati sono in linea con l'evidenza neurofisiologica che i neuroni della corteccia striata (visiva) dei gatti sviluppano durante le prime due settimane di vita una preferenza per determinate orientazioni della stimolazione visiva [Frégnac 1979; Frégnac e Imbert 1978; Hubel e Wiesel 1963]. Lo stesso modello è stato anche in grado di predire le risposte dei neuroni visivi di animali allevati in condizioni di stimolazione visiva alterata artificialmente.

Il modello BCM sembra catturare importanti proprietà generali che

un sistema sensoriale auto-organizzato deve possedere, ma presenta alcuni problemi computazionali. Innanzitutto i parametri y_0 e p sono cruciali, ma devono essere definiti in maniera euristica (ad esempio in base a esperimenti) e dipendono molto dalla particolare distribuzione dei pattern di addestramento; il modello potrebbe esibire comportamenti patologici oppure non stabilizzarsi se questi valori non vengono selezionati correttamente. Inoltre la misura di massima selettività non può essere ottimizzata direttamente (come per la funzione di costo o di energia) perché non è una funzione continua e differenziabile. Intrator e Cooper [1992] hanno in seguito proposto una riformulazione del modello BCM in cui i pesi sinaptici vengono modificati in modo da massimizzare una funzione obiettivo collegata all'inclinazione della distribuzione degli input: il nuovo modello non presenta più i vincoli di attivazione positiva per le unità ed è in grado di scoprire proiezioni di distribuzioni bimodali.

2.7. Auto-organizzazione in una rete multistrato: il modello di Linsker

I modelli descritti nei paragrafi precedenti sono caratterizzati da un unico strato di unità che elaborano il segnale proveniente dai recettori sensoriali. Tuttavia, nei sistemi nervosi biologici l'informazione visiva viene elaborata attraverso una successione di strati di neuroni e moduli subcorticali e corticali (retina → nucleo genicolato laterale e collicolo superiore → area corticale V1 → aree corticali successive). L'analisi delle proprietà di risposta delle cellule nervose indica che gli aspetti più semplici del segnale visivo, come contrasto e orientazione di bordi, vengono analizzati nelle prime fasi dell'elaborazione neurale mentre altre proprietà più complesse vengono analizzate successivamente; altri aspetti del segnale retinico, come colore e movimento, vengono analizzati in parallelo attraverso canali relativamente indipendenti [DeYoe e van Essen 1988]. Dal punto di vista anatomico i neuroni che giacciono su uno stesso strato sembrano svolgere trasformazioni simili del segnale. Se paragoniamo l'operazione svolta da ogni cellula al risultato di una convoluzione dell'immagine del proprio campo recettivo con uno specifico filtro (ad esempio una differenza di funzioni gaussiane), ogni strato della corteccia visiva corrisponde a una popolazione di filtri con caratteristiche simili. Inoltre alcune proprietà di risposta dei neuroni visivi (contrasto centro-contorno, orientazione di bordi e orientazione di linee inclinate) sono già presenti alla nascita dell'organismo e sembrano quindi svilupparsi nel feto[7].

[7] Può anche darsi che queste proprietà siano determinate geneticamente, ma questa ipotesi sembra abbastanza improbabile perché implicherebbe che il codice genetico specifichi esattamente non solo la connettività esatta e la precisa configurazione di eccitazione e inibizione, ma anche la forza esatta delle singole sinapsi. Allo stato attuale delle conoscenze sembra che il codice genetico si limiti a specificare aspetti generali del sistema nervoso e sia meglio descrivibile come un progetto di sviluppo

Il modello formulato da Linsker [1986] è un tentativo di capire quali siano i principi organizzativi di base che danno luogo a queste caratteristiche e se le regole di sviluppo del sistema visivo agiscano al fine di ottimizzare una qualche funzione che possa essere utile al resto del sistema nervoso e all'organismo stesso. La rete neurale impiegata da Linsker è caratterizzata da una gerarchia di strati di neuroni lineari che elaborano il segnale in sequenza (fig. 4.6). Ciascuno strato può essere visualizzato come una matrice bidimensionale di unità che ricevono connessioni solo da un insieme ristretto di unità dello strato inferiore. I campi recettivi delle unità sono parzialmente sovrapposti e la probabilità di connessione fra due unità appartenenti a due strati successivi è descrivibile da una distribuzione gaussiana: questa scelta fa sì che ciascuna unità riceva connessioni soprattutto dalle unità sottostanti che si trovano nella stessa posizione. Nella versione più semplice non vi sono connessioni fra i neuroni dello stesso strato.

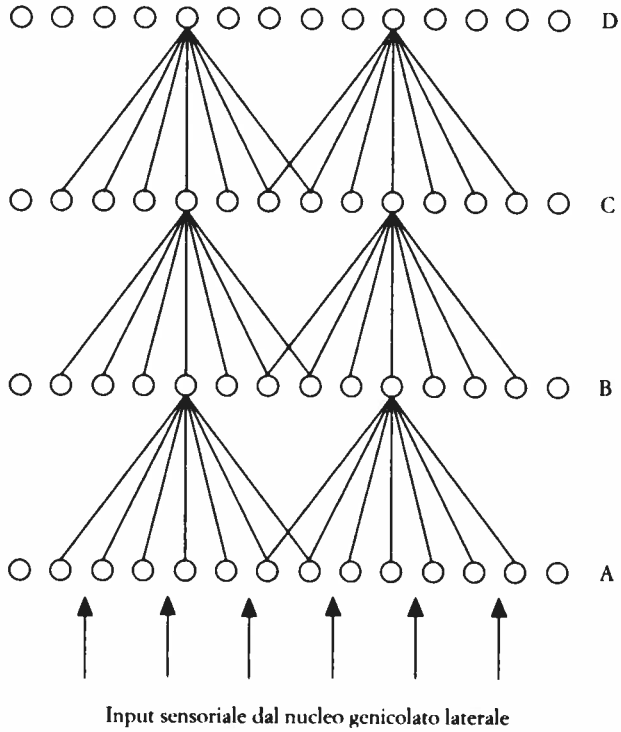


FIG. 4.6. Architettura del modello multistrato di Linsker. Solo alcune delle connessioni sono raffigurate.

piuttosto che come un preciso disegno finale (torneremo in maggior dettaglio su questi argomenti da un punto di vista computazionale nel prossimo capitolo sugli algoritmi genetici).

Dal secondo capitolo sappiamo che una serie di trasformazioni lineari eseguite in successione sono matematicamente equivalenti a un'unica trasformazione lineare eseguita da una sinapsi con un opportuno valore: potrebbe quindi sembrare strano che Linsker utilizzi un'architettura apparentemente ridondante sotto questo punto di vista. Il fatto è che sebbene la trasformazione complessiva finale possa essere realizzata con un solo strato di unità, il processo di apprendimento può cionondimeno risultare facilitato dalla presenza di un'architettura multistrato. Inoltre tale architettura permette di illustrare quali trasformazioni possano essere svolte da una rete senza supervisione le cui caratteristiche architetturali e di funzionamento obbediscono a una serie di vincoli motivati biologicamente. Uno di questi vincoli è proprio l'elaborazione del segnale attraverso una serie di strati organizzati gerarchicamente. Per le unità di ogni strato l'attivazione delle unità nel modello di Linsker è data dalla familiare somma integrata degli input

$$y_i = a_1 + \sum_j w_{ij} x_j$$

e la regola di modifica sinaptica è essenzialmente hebbiana

$$\Delta w_{ij} = a_2 y_i x_j + a_3 x_j + a_4 y_i + a_5$$

dove a_1, a_2, a_3, a_4 e a_5 sono costanti di cui a_2 è vincolata ad essere maggiore di zero; se tutte le altre costanti vengono fissate a zero, la modifica sinaptica si riduce alla semplice regola di Hebb la quale può essere ora riscritta per il valor medio della variazione dei pesi sinaptici evidenziando l'importanza della covarianza tra i singoli segnali di input

$$\langle \Delta w_{ij} \rangle = \sum_k w_{ik} C_{jk} + \left[k_1 + \left(\frac{k_2}{N} \right) \sum_k w_{ik} \right]$$

dove k_1 e k_2 sono opportune combinazioni delle costanti precedenti, N è il numero di unità presinaptiche e

$$C_{jk} = \langle (x_j - \bar{x})(x_k - \bar{x}) \rangle$$

è la covarianza fra due unità presinaptiche calcolata su tutti i pattern della distribuzione di input; $\bar{x}$ è l'attività media di un'unità e può essere considerata uguale per entrambe le unità. Questa notazione mette in risalto il fatto che in tutte le regole hebbiane quello che alla fine conta per lo sviluppo dei pesi sinaptici è la matrice di covarianza fra i pattern di ingresso; il fatto che un peso sinaptico w_{ij} dipenda non solo dall'unità presinaptica j, ma anche dalle altre unità presinaptiche k non significa che la regola hebbiana non sia locale; il motivo per cui ritroviamo questa dipendenza è dovuto al fatto che la regola di Hebb

si basa sul prodotto tra l'attivazione presinaptica x_j e l'attività postsinaptica y_i, la quale a sua volta è calcolata in funzione di tutte le unità presinaptiche (non solo l'unità presinaptica j). Per evitare la crescita dei pesi all'infinito, il modello prevede un valore massimo positivo w_{ij}^{+} e negativo w_{ij}^{-}.

L'analisi di Linsker si è concentrata soprattutto sul caso in cui l'input proveniente dal mondo esterno è «nullo», ovvero completamente casuale per cui la correlazione di attività fra ogni coppia di recettori è zero. Questa scelta intende simulare la situazione di sviluppo del sistema visivo di un organismo prima della nascita. Il modello sviluppa le connessioni sinaptiche in successione strato per strato: inizialmente vengono modificate solamente le sinapsi delle connessioni tra lo strato A (quello che riceve input dal mondo esterno) e lo strato B; una volta che queste connessioni sono maturate completamente, è possibile calcolare la matrice di covarianza dell'attivazione delle unità dello strato B e correggere i pesi sinaptici tra lo strato B e C; e così via.

I risultati delle simulazioni indicano che durante il processo di maturazione le unità sviluppano proprietà di risposta sempre più complesse man mano che si procede attraverso gli strati della rete. Le prime unità a svilupparsi sono quelle dello strato B: per alcuni valori delle costanti moltiplicative, i pesi sinaptici diventano tutti eccitatori e le unità tendono a segnalare semplicemente l'attività media delle unità presinaptiche dello strato A presenti nel proprio campo recettivo. La visualizzazione dell'attività di queste unità è simile all'«effetto neve» che si osserva sui televisori (quando il segnale captato è costituito dal solo rumore) passato attraverso un filtro a banda larga; inoltre l'attività di unità vicine tende a essere correlata (se un'unità è molto attiva, probabilmente anche le unità confinanti saranno attive a causa della topologia della connettività dallo strato inferiore). Di conseguenza le unità dello strato C sviluppano un tipo di risposta centro-contorno: esse diventano attive quando un punto luminoso viene proiettato al centro del loro campo recettivo su uno sfondo scuro. Analogamente, sullo stesso strato emergono anche unità sensibili al rapporto inverso. Entrambe queste unità sono analoghe a dei filtri sensibili al contrasto. Infine le unità dello strato D diventano sensibili all'orientazione: esse si attivano in presenza di un bordo o di una linea luminosa su uno sfondo scuro (o viceversa) quando il bordo e la linea possiedono una particolare inclinazione. Linsker ha mostrato che se la rete neurale non possiede connessioni laterali fra unità dello stesso strato, ciascuna unità svilupperà una preferenza arbitraria per una particolare orientazione. Se però vengono incluse delle connessioni laterali fra unità confinanti, la preferenza per l'orientazione si distribuisce sulla superficie dello strato in modo da formare regioni di forma irregolare che contengono unità con preferenza per orientazioni simili (fig. 4.7).

I neuroni che rispondono a stimoli centro-contorno o a barre orientate costituiscono una caratteristica predominante della corteccia

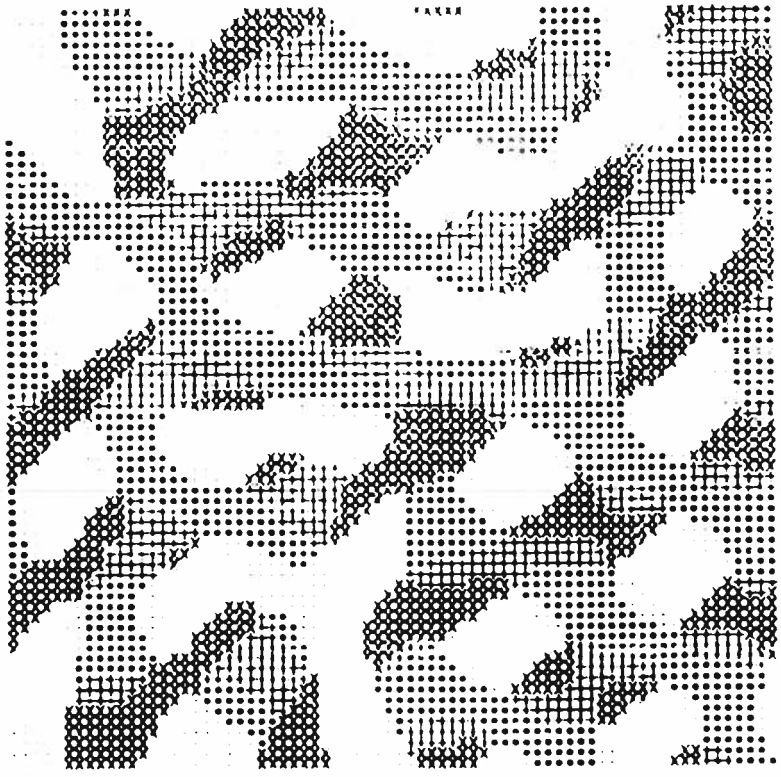


FIG. 4.7. Sviluppo di mappa topografica nella rete di Linsker. Distribuzione delle proprietà di risposta delle unità dello strato D. Ciascun simbolo indica la preferenza dell'unità per una determinata orientazione della barra di stimolo.

Legenda: punto: 9°-45°; cerchietto: 45°-81°; +: 81°-117°; x: 117°-153°; vuoto: 153°-189°.

Fonte: Linsker [1986].

striata del sistema nervoso dei mammiferi; similmente, i neuroni che rispondono all'orientazione di barre inclinate si dispongono in regioni compatte della corteccia e organizzate topologicamente in base all'orientazione dello stimolo [Hubel e Wiesel 1977]. I risultati delle simulazioni non intendono però suggerire che i meccanismi specifici di sviluppo del sistema nervoso siano identici a quelli della rete neurale artificiale, bensì che alcune proprietà dell'elaborazione dell'informazione visiva possono essere spiegate nei termini di un processo di plasticità hebbiana in una rete multistrato.

Linsker suggerisce che lo scopo dell'elaborazione dell'informazione visiva consiste nel massimizzare la varianza dell'output dei neuroni. Partendo dalla regola di Hebb, egli individua una funzione obiettivo E che viene minimizzata durante lo sviluppo dei pesi sinaptici

$$E = E_C + E_K$$

dove

$$E_{\mathrm{C}} = -\frac{1}{2}\langle (y - \bar{y})^2 \rangle$$

e

$$E_K = -k_1 \sum_i w_i - \left(\frac{k_2}{2N} \right) \left(\sum_i w_i \right)^2$$

Il cambiamento dei pesi sinaptici in pratica esegue la discesa del gradiente di questa funzione. Mentre il secondo termine contribuisce al rifinimento dei valori sinaptici già maturi, il primo dipende dalla varianza dell'output della rete: quindi, data una forza totale di connessioni $\sum_i w_i$ per una certa unità della rete, la funzione E viene minimizzata quando la varianza $\langle (y - \bar{y})^2 \rangle$ dell'output viene massimizzata.

3. Modelli basati su principi di Teoria dell'Informazione

La massimizzazione della varianza dell'attivazione di un'unità corrisponde alla massimizzazione del tasso di informazione trasmessa da quell'unità (a patto che la distribuzione dei segnali di input non sia completamente arbitraria). Infatti, un'unità che emette sempre lo stesso segnale – ad esempio 1 – qualunque sia il proprio input non fornisce alcuna informazione al resto del sistema nervoso. La regola di Hebb agisce in modo da sviluppare delle unità che mantengono il massimo dell'informazione sull'attività del proprio input. Questo concetto può essere descritto formalmente utilizzando alcuni principi di Teoria dell'Informazione[8].

3.1. Informazione, Entropia, Entropia condizionale e Informazione mutua

Immaginiamo di avere un alfabeto finito di simboli $\{x_1, x_2, ..., x_n\}$ che possono essere trasmessi attraverso un canale dotato di rumore; definiamo ora X come una variabile aleatoria che descrive il simbolo ottenuto dall'alfabeto in base a una qualche legge probabilistica. Se la probabilità che il simbolo x_i sia selezionato è p_i

$$\Pr(X = x_i) = p_i \quad (i = 1, 2, ..., n)$$

[8] La breve spiegazione del prossimo paragrafo è basata su Kay [1992] e Linsker [1988]. Si veda anche Haykin [1999].

allora X possiede la seguente distribuzione di probabilità

x	x_1	x_2	x_3	...	x_n
$\Pr(X = x)$	p_1	p_2	p_3	...	p_n

L'idea è quella di misurare in qualche modo la quantità di informazione contenuta in un simbolo x_i e nell'intero alfabeto $\{x_1, x_2, ..., x_n\}$. Se un certo simbolo avesse probabilità 1 di essere selezionato, noi continueremmo ad osservare la sua presenza e non ne saremmo affatto sorpresi perché sappiamo che esso deve apparire ogni volta che eseguiamo un'osservazione. In altre parole, possiamo dire che non otteniamo alcuna informazione dall'osservare la sua presenza: quindi, un evento sicuro non ci dà alcuna informazione. Al contrario, se osservassimo un evento improbabile – come la presenza di un simbolo raro – allora potremo dire di aver ricavato maggior informazione. Questo ragionamento potrebbe portarci a concludere che la quantità di informazione fornita da un simbolo è inversamente proporzionale alla probabilità che il simbolo venga osservato

$$\text{Informazione} \propto \frac{1}{\text{Probabilità}}$$

oppure, più formalmente

$$I(x_i) \propto \frac{1}{p_i}$$

Immaginiamo ora di osservare due simboli indipendenti fra di loro – x_i e x_j – e di voler esprimere l'informazione congiunta in funzione dell'informazione fornita da ciascuno dei due simboli separatamente. In base all'ipotesi appena formulata avremmo

$$I(x_i, x_j) \propto \frac{1}{\Pr(X = x_i \text{ e } X = x_j)}$$

e siccome i due simboli sono indipendenti

$$\Pr(X = x_i \text{ e } X = x_j) = \Pr(X = x_i)\ \Pr(X = x_j) = p_i p_j$$

Quindi l'informazione congiunta è data da

$$I(x_i, x_j) \propto \frac{1}{p_i p_j}$$

Questo risultato però non ci soddisfa perché è lecito aspettarsi che l'informazione totale trasmessa da due simboli indipendenti sia la

«somma» delle quantità di informazione trasmesse separatamente dai due simboli. Fortunatamente la funzione logaritmica possiede le proprietà necessarie a render giustizia al nostro senso comune

$$\log\left(\frac{1}{p_i p_j}\right) = \log\left(\frac{1}{p_i}\right) + \log\left(\frac{1}{pj}\right)$$

e pertanto la adottiamo nella definizione di *misura di informazione*[9] [Shannon 1948]

$$I(x_i) = \log\left(\frac{1}{p_i}\right) = -\log(p_i)$$

Immaginiamo ora di voler misurare la quantità di informazione fornita dall'intero alfabeto $\{x_1, x_2, ..., x_n\}$: ciascun simbolo x_i viene selezionato una frazione p_i delle volte e trasmesso attraverso il canale. L'informazione media contenuta nell'alfabeto è data allora da

$$H = p_1 \log\left(\frac{1}{p_1}\right) + p_2 \log\left(\frac{1}{p_2}\right) + ... + p_n \log\left(\frac{1}{p_n}\right) =$$

$$= \sum_{i=1}^{n} p_i \log\left(\frac{1}{p_i}\right) \equiv -\sum_{i=1}^{n} p_i \log(p_i)$$

H è anche detta *Entropia di Shannon* e può essere riscritta come $H(X)$, l'informazione media contenuta nell'alfabeto oppure l'informazione media contenuta nella distribuzione di probabilità della variabile aleatoria X. Si noti che siccome $0 < p_i \leq 1$, si ottiene che $\log(p_i) \leq 0$ e quindi $H(X)$ è una quantità non negativa. Il concetto di entropia (informazione ottenuta) viene quindi messo in relazione con il concetto di «incertezza» contenuta in una distribuzione di probabilità. A prima vista questa sembrerebbe una strana associazione, ma diventa subito chiara se stabiliamo che l'informazione ottenuta dall'aver effettuato un'osservazione è uguale all'incertezza rimossa dall'aver effettuato quell'osservazione.

Fino ad ora abbiamo effettivamente misurato l'informazione presente alla sorgente; torniamo ora all'esempio del canale di comunicazione. Ciascuno dei simboli viene spedito lungo il canale uno alla volta con una certa probabilità; siccome il canale è dotato di rumore intrin-

[9] Possiamo usare qualsiasi base per la funzione logaritmica (purché si resti consistenti), ma la base 2 viene spesso usata in questo contesto: in tal caso l'unità di informazione è definita «bit». Tuttavia logaritmi a base diversa possono essere convertiti usando la seguente formula

$$\log_a(x) = \frac{\log_b(x)}{\log_b(a)}$$

seco, può darsi che il simbolo ricevuto y_j non sia uguale al simbolo originariamente spedito x_i. L'informazione contenuta alla sorgente è $H(X)$ e l'informazione contenuta nel messaggio ricevuto è $H(Y)$. Dato che lo scopo della comunicazione è la trasmissione di informazione, noi siamo interessati a sapere qual è l'informazione su X che il ricevente ottiene dall'osservare Y, ovvero la probabilità che un simbolo x_i sia stato spedito data l'osservazione di un simbolo y_j

$$\Pr(X = x_i | Y = y_j) = p(x_i|y_j)$$

che è la *probabilità condizionale* che un simbolo x_i sia stato spedito lungo il canale *data* l'osservazione del simbolo y_j. Tralasciando i dettagli, l'*entropia condizionale media* è definita da

$$H(X \mid Y) = \sum_j p(y_j) \sum_i p(x_i \mid y_j) \log\left(\frac{1}{p(x_i \mid y_j)}\right)$$

L'entropia condizionale è la quantità di informazione *aggiuntiva* necessaria a ricostruire il messaggio originario. Quindi la quantità di informazione su X fornita dalla conoscenza di Y è data da

$$R = H(X) - H(X|Y)$$

che è il *tasso di informazione* per un messaggio trasmesso dalla sorgente al ricevente.

Il tasso di informazione è equivalente all'«informazione mutua» $I(X;Y)$ tra il messaggio spedito e il messaggio ricevuto (fig. 4.8), ovvero l'informazione condivisa tra di loro; essa possiede le seguenti proprietà

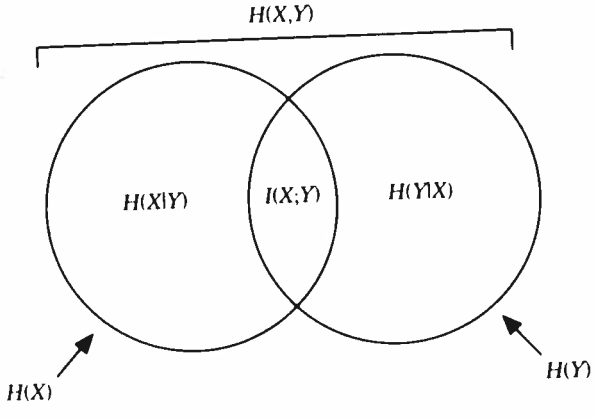


FIG. 4.8. Relazione tra informazione mutua $I(X;Y)$ e le entropie delle due distribuzioni X e Y.

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \\ &= H(X) + H(Y) - H(X,Y) \end{aligned}$$

dove $H(X,Y)$ è l'entropia congiunta della distribuzione di X e Y.

L'informazione mutua è zero se le variabili aleatorie X e Y sono indipendenti; $H(X,Y)$ è quindi un indicatore di dipendenza tra le due variabili.

3.2. Massimizzazione dell'informazione mutua tra unità presinaptica e postsinaptica

Un neurone (artificiale o biologico) è paragonabile a una sorgente di informazione misurabile in base al concetto di entropia; siccome un numero reale trasmette una quantità infinita di informazione, immaginiamo di discretizzare i possibili stati di attivazione del neurone in un certo numero di valori: nel caso più semplice il neurone può assumere solo due stati, ad esempio {0, 1}, ma è facile immaginare delle approssimazioni a valori reali più precise. Il segnale emesso dal neurone viaggia lungo la connessione sinaptica fino a raggiungere il neurone postsinaptico: se la comunicazione ha avuto successo ci si aspetta che il neurone postsinaptico esibisca un'attivazione in qualche modo collegata all'attivazione del neurone presinaptico. La misurazione dell'informazione trasmessa e le dinamiche di apprendimento si prestano quindi ad essere descritte esattamente dai principi di Teoria dell'Informazione esposti nel paragrafo precedente.

Continuando l'analisi del suo modello multistrato, Linsker [1988] ha indicato che in certe condizioni la massimizzazione della varianza dell'unità postsinaptica corrisponde alla massimizzazione dell'informazione mutua (o tasso di informazione trasmessa) tra l'unità postsinaptica e l'unità presinaptica. Consideriamo l'output dell'unità presinaptica in presenza di rumore v

$$y_i = v + \sum_j w_{ij} x_j$$

tale per cui *a*) l'attivazione di y_i possiede una distribuzione gaussiana con varianza V_i; *b*) il rumore v possiede una distribuzione gaussiana con varianza Z; *c*) il rumore v non è correlato con alcun componente dei pattern di input; ovvero $\langle vx_j \rangle = 0$ per tutti i componenti j. Trascurando i dettagli otteniamo che il tasso di informazione trasmessa è

$$R = \frac{1}{2} \log\left(\frac{V_i}{Z}\right)$$

questo significa che, per una certa varianza Z del rumore v, il tasso di informazione è massimizzato dalla massimizzazione della varianza V_i

dell'unità postsinaptica, che è proprio il risultato che si ottiene con l'apprendimento hebbiano descritto nel paragrafo 2.7. Si noti inoltre che se il rumore viene aggiunto in modo indipendente su ciascun canale sinaptico

$$y_i = \sum_j w_{ij}(x_j + v_j)$$

il tasso di informazione diventa

$$R = \frac{1}{2}\log\left(\frac{V_i}{\left(Z\sum_j w_{ij}^2\right)}\right)$$

e in questo caso esso viene massimizzato quando $V_i / \sum_j w_{ij}^2$ viene massimizzata, ovvero quando i pesi sinaptici vengono modificati impiegando un fattore di normalizzazione (che corrisponde all'estrazione delle componenti principali della distribuzione dell'input).

L'idea che le regole di maturazione dei neuroni (artificiali e biologici) tendano a massimizzare il tasso di informazione trasmessa (ovvero l'informazione mutua tra unità presinaptica e unità postsinaptica) è stato definito «principio di conservazione dell'informazione», talvolta abbreviato come «Infomax».

Altri autori hanno fornito modelli basati sul linee di ragionamento simili. Plumbley e Fallside [1988] ad esempio hanno modificato leggermente il principio di Infomax ponendo l'accento sulla riduzione della perdita di informazione definita come l'errore quadratico medio tra l'input originale e l'input ricostruibile impiegando l'attivazione dei neuroni postsinaptici maturi. Atick e Redlich [1990] hanno invece investigato il principio di minima ridondanza secondo il quale il segnale sensoriale che giunge ai recettori possiede un alto grado di ridondanza e lo scopo del sistema nervoso sensoriale sarebbe quello di ridurre questa ridondanza prima di spedire l'informazione ai sistemi corticali superiori.

3.3. Estrazione di informazione coerente nello spazio

Un'alternativa ai modelli descritti precedentemente consiste nel vincolare l'apprendimento in modo che la rete estragga delle caratteristiche del proprio input che possano essere utili per le fasi successive di elaborazione del sistema nervoso (si veda ad esempio Field [1994]). Becker e Hinton [1992] hanno suggerito che una possibile finalità del sistema percettivo consiste nell'estrazione di caratteristiche che esibiscano delle semplici forme di coerenza spaziale. Mediante l'utilizzo di una serie di moduli neurali che ricevono input da aree confinanti del-

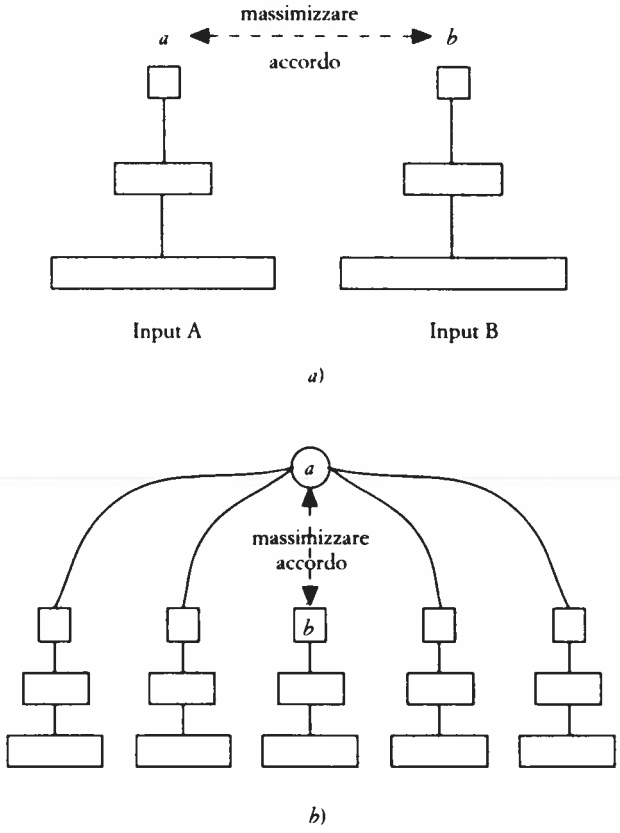


FIG. 4.9. Massimizzazione dell'informazione mutua tra l'output di ciascun modulo e l'output di moduli vicini; ciascun modulo riceve input da zone confinanti dell'immagine visiva. La rete neurale apprende a segnalare parametri dell'immagine che esibiscono coerenza spaziale, come l'indice di profondità da immagini stereoscopiche. In *b*) ciascun modulo cerca di massimizzare l'informazione mutua fra il proprio output e una combinazione lineare (ad esempio la somma) degli output degli altri moduli: nella figura è evidenziato il confronto tra l'output *b* del modulo centrale e la combinazione lineare *a* degli output dei moduli confinanti.

l'immagine visiva, ciascun modulo apprende a segnalare delle caratteristiche che obbediscono a criteri di coerenza spaziale semplicemente massimizzando l'informazione mutua fra il proprio output e l'output dei moduli confinanti (fig. 4.9).

Nel semplice caso di due moduli, ciascuno dotato di una singola unità di output (fig. 4.9a), la funzione obiettivo da massimizzare è

$$I(a;b) = \frac{1}{2}\log\left(\frac{V(a+b)}{V(a-b)}\right)$$

dove V è la varianza calcolata su tutti i pattern di addestramento. L'in-

formazione mutua $I(a;b)$ definisce la quantità di informazione fornita in media dai due moduli sul segnale comune di input, ovvero la caratteristica dell'input che è coerente nelle due aree dell'immagine visiva. Gli autori hanno applicato questo modello all'elaborazione di immagini stereoscopiche. L'input di ciascun modulo è diviso a metà: ciascuna metà riceve input da uno dei due occhi. La profondità della superficie relativa al punto di fissazione genera delle disparità di intensità nelle proiezioni sulle due retine (i punti sono spostati verso sinistra o verso destra in base alla profondità). Gli input dei due moduli provengono da aree spazialmente adiacenti sulla retina: in questo modo si assume che la disparità tra le intensità delle immagini provenienti dall'occhio destro e da quello sinistro sia l'unica caratteristica coerente fra le due immagini stereoscopiche. Nel caso di superfici piatte, la disparità è identica in entrambe le immagini stereoscopiche, ma nel caso di superfici curve essa varia leggermente nello spazio. Mediante una regola di modifica sinaptica che massimizza la funzione obiettivo, la rete neurale apprende a segnalare la profondità degli oggetti nell'input visivo, indicando se si trovano prima o dopo il punto di fissazione.

Lo stesso principio è stato impiegato per addestrare reti neurali composte di molti moduli (fig. 4.9b) in cui l'unica assunzione è che l'input dei moduli provenga da zone spazialmente adiacenti dell'immagine retinica. In questo caso l'output di ciascun modulo cerca di massimizzare l'informazione mutua tra il proprio output e una combinazione lineare degli output dei moduli vicini. Esso è in grado di estrarre un indice continuo di profondità da immagini stereoscopiche di superfici curve. Il principio di estrazione di informazione coerente fra moduli diversi viene spesso indicato con il nome di «Imax».

4. Modelli basati su meccanismi di competizione

I modelli neurali descritti in questo paragrafo si basano sul principio di competizione fra le unità di output della rete: le unità competono fra di loro per restare attive e segnalare il pattern di input corrente. Ciascuna unità occupa una precisa posizione rispetto alle altre e possiede connessioni laterali: la disposizione geometrica e il gioco di inibizione ed eccitazione fra di esse crea un meccanismo tale per cui, dato un certo pattern di input, solamente una o poche unità restano attive mentre tutte le altre vengono spinte gradualmente verso zero. La risposta della rete consiste nella configurazione di unità attive al termine del processo di competizione. Il meccanismo di apprendimento è di tipo postsinaptico: solamente i pesi delle unità attive vengono modificati in base a una regola hebbiana.

I modelli basati su meccanismi di competizione sono storicamente motivati soprattutto dal desiderio di catturare alcuni importanti principi di funzionamento del sistema corticale. Il processo di apprendimento tende a sviluppare unità specializzate nel riconoscere determi-

nati pattern (o gruppi di pattern) di input in modo tale che le caratteristiche geometriche che caratterizzano la configurazione dei recettori e le proprietà della distribuzione dei pattern di input siano riflesse nella disposizione finale della risposta della rete. Come è stato già sottolineato nel secondo capitolo, questo tipo di organizzazione è stato ampiamente riscontrato nel sistema nervoso centrale: ad esempio, la disposizione dei neuroni delle aree visive individua una mappa retinotopica che riflette la geometria della retina (neuroni vicini rispondono a segnali provenienti da recettori vicini). Questo tipo di corrispondenza è in parte dovuto al mantenimento di campi recettivi ristretti e subisce una graduale distorsione man mano che si procede verso le zone superiori dell'elaborazione (aree temporali e frontali). Un altro livello di organizzazione topologica riguarda le proprietà della stimolazione sensoriale: neuroni che rispondono a stimoli simili tendono ad occupare posizioni contigue sul tessuto corticale e sono disposti in modo da riflettere alcune proprietà dello stimolo. Alcuni esempi sono dati dalle mappe tonotopiche della corteccia uditiva e dalle colonne verticali di neuroni che rispondono a barre orientate in cui la relativa posizione del neurone all'interno della colonna varia in corrispondenza della variazione dell'inclinazione dello stimolo.

I primi modelli di auto-organizzazione neurale per competizione [von der Malsburg 1973; Willshaw e von der Malsburg 1976] furono proposti proprio per spiegare la formazione delle colonne di orientazione riscontrate nel sistema visivo dei mammiferi in quanto esse non possono essere interamente determinate dal codice genetico dell'organismo. Questi modelli possiedono caratteristiche interessanti anche da un punto di vista applicativo perché sono in grado di classificare i pattern di input senza bisogno di un insegnante esterno che guidi il processo di categorizzazione fornendo le risposte corrette. Come vedremo in seguito essi si prestano ad essere impiegati in sistemi modulari con notevoli vantaggi rispetto ai sistemi basati solamente su un processo di categorizzazione supervisionato (come nel caso dei percettroni multistrato).

4.1. Formazione di zone di attività circoscritta

In questo paragrafo mostreremo come un gruppo di unità caratterizzate da relazioni geometriche e connessioni laterali sia in grado di dar luogo a un meccanismo di competizione da cui emergono zone di attività circoscritta («bolle») comprendenti una o più unità[10]. L'evidenza anatomica e neurofisiologica sull'architettura della corteccia dei primati ha messo in risalto tre tipi di *interazione laterale* fra i neuroni: 1) ciascun neurone produce una piccola area di eccitazione a corto rag-

[10] L'esempio qui riportato è tratto dal lavoro originale di Kohonen [1982].

gio che si estende su una superficie con un raggio di circa 50-100 μm intorno ad esso; 2) quest'area di eccitazione è circondata da un anello di inibizione che si estende per un raggio di 200-500 μm; 3) un'ulteriore zona di eccitazione molto più debole circonda l'area inibitoria e si estende per un raggio di alcuni centimetri. Se osservata lungo una sola dimensione, questa configurazione assume la forma di un cappello messicano – da cui prende il nome – (fig. 4.10a) e può essere approssimata da una funzione a gradino che trascura l'area di debole eccitazione a lungo raggio (fig. 4.10c).

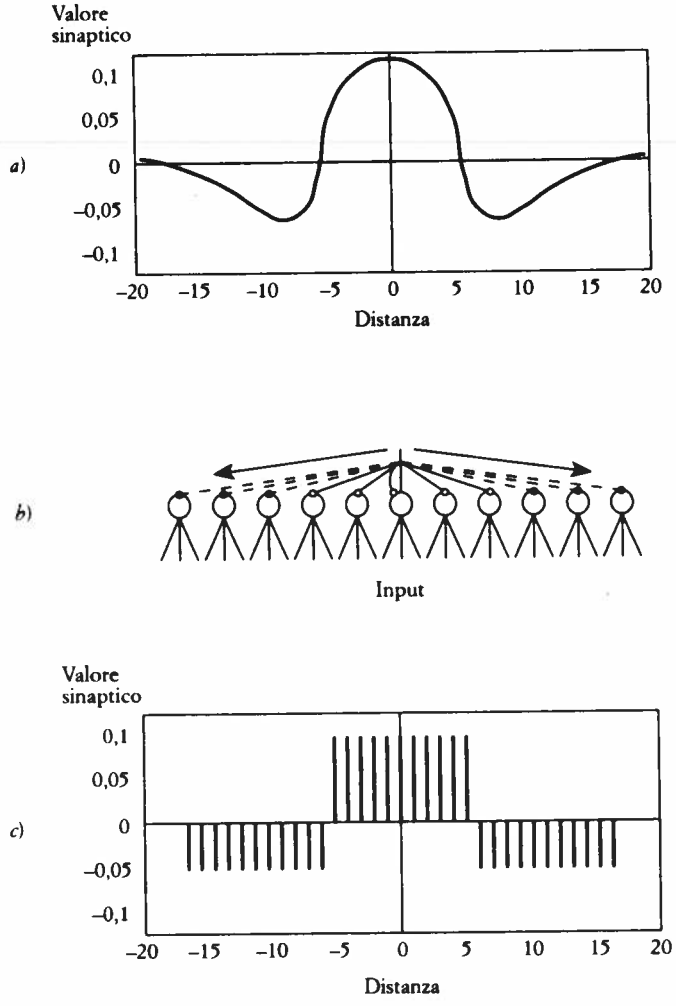


FIG. 4.10. Connessioni laterali in funzione della distanza dall'unità. *a*) Funzione a cappello messicano; *b*) architettura neurale (il numero di unità non corrisponde a quello riportato nei grafici per motivi di spazio): pallini bianchi = connessioni eccitatorie; pallini neri = connessioni inibitorie; *c*) approssimazione a gradino (ciascuna barra verticale corrisponde alla posizione di un'unità).

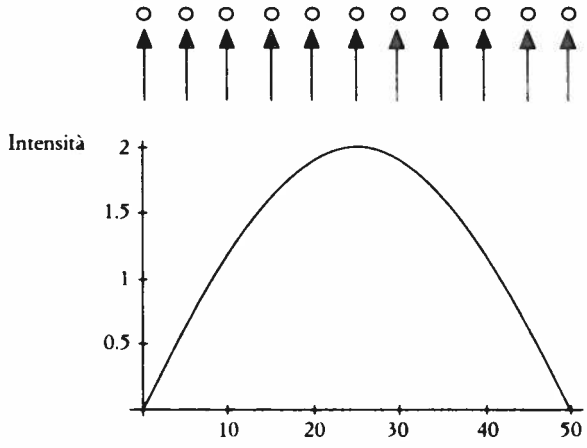


FIG. 4.11. Disposizione delle unità e intensità del segnale proveniente dall'input calcolato in base alla funzione $I_i = 2 \sin\left(\frac{\pi i}{50}\right)$, $0 \leq i \leq 50$.

Questa approssimazione si rivela utile per la simulazione numerica del processo di interazione fra unità artificiali disposte in linea (fig. 4.11) e connesse fra di loro in modo analogo a quello rappresentato in figura 4.10. L'attivazione di ciascuna unità y_i ad ogni istante t è data dalla combinazione tra il segnale I_i proveniente dallo strato di input e la somma pesata degli output delle unità circostanti y_k ad essa connessa

$$y_i^t = \Phi\left(I_i^t + \sum_{k=-r}^{+r} v_{ik} y_{i+k}^{t-1}\right)$$

dove r è la distanza massima di connessione laterale (equivalente al raggio dell'area nei modelli bidimensionali) intorno all'unità, v_{ik} indica il valore dei pesi sinaptici laterali (fissati per ciascuna unità in base alla funzione di figura 4.10c) e Φ è una funzione semilineare tale per cui

$$\Phi(x) = \begin{cases} 0 & \text{se } x < 0 \\ x & \text{se } 0 \leq x \leq a \\ a & \text{se } x > a \end{cases}$$

dove a è una costante che indica l'attivazione massima di ciascuna unità. Siccome le unità interagiscono fra di loro, la funzione di attivazione viene applicata ripetutamente fino a quando esse raggiungono un punto di stabilità; durante questo processo di «rilassamento» dell'attivazione si assume che l'intensità del segnale I_i proveniente dall'input rimanga costante.

Lo scopo della simulazione consiste nel mostrare che un gruppo di unità con queste caratteristiche si organizza in zone di attività centrata intorno all'unità che riceve il segnale più forte dall'input. Nell'esempio considerato vi sono 51 unità disposte in linea orizzontale e l'input per ciascuna di esse è dato dal valore di una funzione sinusoidale che assume un valore massimo pari a 2 al centro del gruppo e 0 agli estremi (fig. 4.11). Ciascuna unità riceve connessioni da 32 unità circostanti ($r = 16$) oltre a possedere una connessione autoricorrente eccitatoria. Il segnale di input viene mantenuto costante durante la ripetizione del calcolo di attivazione delle unità. Inizialmente l'attivazione si distribuisce sull'intera linea di unità e riflette l'intensità del segnale di input per ciascuna di esse; successivamente essa si concentra intorno all'unità centrale e aumenta mentre le unità laterali decrescono la propria attivazione verso zero (fig. 4.12a). La stabilità del sistema è assicurata dalla funzione di output che limita il valore di attivazione massimo. Il centro della bolla di attivazione corrisponde all'unità che possiede il segnale massimo proveniente dall'input; la sua larghezza dipende dalla forza relativa delle connessioni laterali: se vengono rafforzate le connessioni eccitatorie l'area di unità attive si allarga, mentre se vengono rafforzate le connessioni inibitorie essa si restringe. La formazione della bolla è invece prevenuta se la forza delle connessioni laterali non supera un certo valore critico in rapporto al segnale di input (fig. 4.12b).

Questi risultati possono essere simulati in una rete senza bisogno di introdurre connessioni laterali e il processo di rilassamento: è sufficiente portare a 1 il valore di tutte le unità che si trovano vicine all'unità con attivazione maggiore e lasciare a 0 tutte le unità rimanenti (fig. 4.13). L'unico parametro libero consiste nella scelta della *funzione di vicinanza* ovvero la larghezza della zona di unità attive intorno all'unità vincitrice.

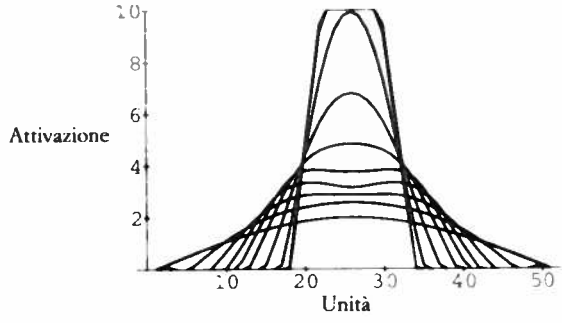
Più in generale l'unità vincente i^* è quella che possiede il potenziale di attivazione A_i di intensità maggiore

$$A_{i^*} = \sum_j w_{i^* j} x_j$$

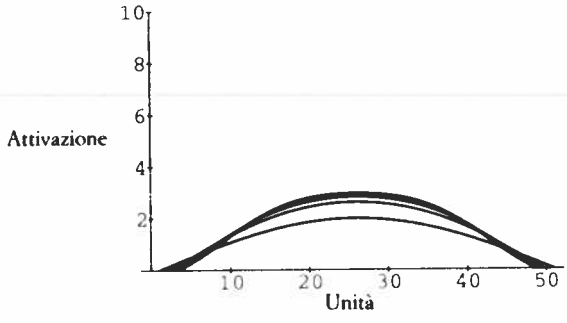
per quel determinato pattern di input $\mathbf{x}$. Nel primo capitolo abbiamo notato che quanto più un'unità è attiva per un determinato pattern, tanto più il vettore di pesi sinaptici è simile al pattern di input. Se tutti i pesi sinaptici della rete sono normalizzati, questo equivale a dire che

$$\|\mathbf{w}_{i^*} - \mathbf{x}\| \leq \|\mathbf{w}_i - \mathbf{x}\| \quad (\forall i)$$

ovvero la distanza euclidea tra il vettore di pesi sinaptici $\mathbf{w}_{i^*}$ dell'unità vincente i^* e il vettore di input $\mathbf{x}$ è la più piccola rispetto a tutte le distanze fra i pesi sinaptici delle altre unità e lo stesso vettore di input. In conclusione, la scelta dell'unità vincente può essere effettuata in



a)



b)

FIG. 4.12. Calcolo ripetuto dell'attivazione delle unità appartenenti a un gruppo competitivo. *a*) Formazione di una bolla di attività centrata intorno all'unità che riceve il segnale più forte dall'input (valori di connessione laterale: +0,15, −0,08); *b*) mancata formazione della bolla dovuta a connessioni laterali troppo deboli rispetto al segnale proveniente dall'input (valori di connessione laterale: +0,05, −0,02).

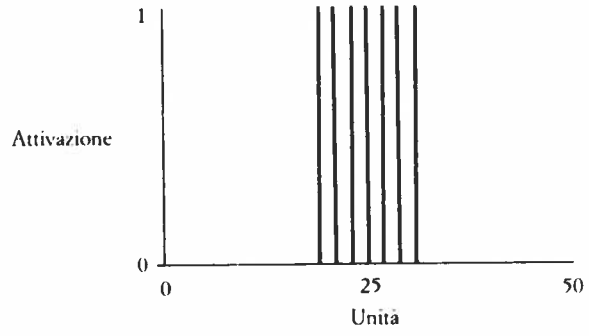


FIG. 4.13. Configurazione schematica della risposta di uno strato di unità competitive.

due modi: se tutti i vettori sinaptici $\mathbf{w}_i$ sono normalizzati, l'indice dell'unità vincente corrisponde all'unità con attivazione massima

$$i^* = i \left| \max_i \left(\sum_j w_{ij} x_j \right) \right.$$

altrimenti esso corrisponde all'unità che possiede la distanza euclidea minima tra il vettore di input e il proprio vettore di sinapsi

$$i^* = i \left| \min_i \left(\sum_j (x_j - w_{ij})^2 \right) \right.$$

Di solito si usa quest'ultimo metodo perché è più generale.

4.2. Formazione di mappe topografiche

L'apprendimento competitivo consiste nell'applicazione della regola di Hebb alle connessioni di input di un gruppo di unità disposte in un certo ordine geometrico. Assumiamo che ciascuna unità possieda un output binario

$$y_i = \Phi(A_i) = \begin{cases} 1 & \text{se entro la bolla di attivazione} \\ 0 & \text{altrimenti} \end{cases}$$

Come abbiamo già osservato all'inizio di questo capitolo, la regola di Hebb farebbe crescere i pesi all'infinito. Possiamo però introdurre un fattore di dimenticanza (come abbiamo fatto per la regola di Oja) che è una funzione non-lineare dell'output dell'unità postsinaptica

$$\Delta w_{ij} = \eta y_i x_j - \Psi(y_i) w_{ij}$$

dove

$$\Psi(y_i) = \begin{cases} \psi & \text{se } y_i = 1 \\ 0 & \text{se } y_i = 0 \end{cases}$$

dove ψ è una piccola costante maggiore di zero. Questo fattore di dimenticanza, oltre a limitare la crescita infinita dei pesi, assicura anche che essi siano normalizzati a lunghezza 1. Si noti ora che il cambiamento dei pesi sinaptici è zero quando l'unità postsinaptica assume valore zero

$$\Delta w_{ij} = \begin{cases} \eta x_j - \psi w_{ij} & \text{se } y_i = 1 \\ 0 & \text{se } y_i = 0 \end{cases}$$

questo significa che solamente i pesi delle unità che si trovano all'interno della bolla di attivazione vengono modificati. Semplificando ulteriormente, possiamo assumere che la costante ψ sia uguale al valore

del tasso di apprendimento e riscrivere la formula precedente come

$$\Delta w_{ij} = \begin{cases} \eta(x_j - w_{ij}) & \text{se } y_i = 1 \\ 0 & \text{se } y_i = 0 \end{cases}$$

Questa regola viene talvolta definita con il nome di *apprendimento competitivo standard*. Se consideriamo il valore dei pesi sinaptici finali in forma vettoriale

$$\mathbf{w}_i^{t+1} = \begin{cases} \mathbf{w}_i^t + \eta(\mathbf{x} - \mathbf{w}_i^t) & \text{se } y_i = 1 \\ \mathbf{w}_i^t & \text{se } y_i = 0 \end{cases}$$

osserviamo che l'apprendimento competitivo consiste nell'aggiungere ai pesi sinaptici delle unità attive una frazione della differenza tra di essi e il vettore di input attuale; in altre parole l'apprendimento competitivo spinge i pesi sinaptici verso i vettori di input a cui l'unità risponde maggiormente. Siccome le unità possiedono precise relazioni topologiche che vengono impiegate per determinare la zona di attività intorno all'unità vincitrice, la presentazione ripetuta dei pattern di addestramento darà luogo a una *mappa topologica* delle caratteristiche dell'input: unità vicine risponderanno a caratteristiche simili della distribuzione di input perché avranno pesi sinaptici simili. Questa semplice combinazione di unità competitive caratterizzate da relazioni geometriche e apprendimento hebbiano è stata proposta inizialmente da Kohonen [1982; 1989] per spiegare l'auto-organizzazione di mappe topologiche dell'input; l'algoritmo viene comunemente abbreviato in SOM (*Self-Organizing Maps*) o SOFM (*Self-Organizing Feature Maps*).

Il fattore più importante nell'auto-organizzazione delle mappe consiste nella scelta della *funzione di vicinanza* $\Lambda(i,i^*)$, ovvero l'ampiezza r dell'area intorno all'unità vincitrice i^* in cui le unità restano attive. È conveniente formalizzare la scelta della funzione di vicinanza nel modo seguente

$$\Delta(i, i^*) = \begin{cases} 1 & \text{se } \|\mathbf{c}_i - \mathbf{c}_{i^*}\| \leq r \\ 0 & \text{altrimenti} \end{cases}$$

dove $\mathbf{c}_i$ rappresenta il vettore di coordinate spaziali dell'unità i-esima sullo strato della rete e $\|\mathbf{c}_i - \mathbf{c}_{i^*}\|$ è la distanza euclidea tra di essa e l'unità vincitrice. Se $r = 1$ e le unità sono disposte su di una sola dimensione (come nell'esempio del paragrafo precedente), la zona di attività comprende l'unità vincitrice e le due unità ai lati; se invece la disposizione è bidimensionale, la zona di attività comprende l'unità vincitrice e le otto unità circostanti[11]. Possiamo ora riscrivere la regola ge-

[11] È possibile disporre le unità anche su tre o più dimensioni.

nerale di modifica sinaptica delle mappe topologiche

$$\Delta w_{ij} = \eta \Lambda(i,i^*)(x_j - w_{ij})$$

Non esiste un criterio teoretico per determinare l'ampiezza r della funzione di vicinanza: come anche per il tasso di apprendimento η, la scelta viene effettuata per tentativi. Da osservazioni empiriche si nota che il processo di auto-organizzazione di una mappa topologica si divide in due fasi. La prima fase («fase di ordinamento») consiste nell'ordinamento dei vettori sinaptici di ciascuna unità in base alle caratteristiche dell'input: essi si spostano in modo da rappresentare grossolanamente la superficie della distribuzione dei vettori di input. La seconda fase («fase di convergenza») consiste invece nella rifinitura della mappa durante la quale i vettori si assestano nelle posizioni finali. La fase di ordinamento è relativamente rapida (circa un migliaio di iterazioni) e cruciale per la formazione di una mappa corretta, mentre la seconda fase richiede un gran numero di iterazioni (alcune migliaia), ma non introduce cambiamenti drastici nell'ordinamento topologico dei vettori. Viste queste caratteristiche temporali, Kohonen [1989] ha suggerito di far variare durante la fase di ordinamento sia il tasso di apprendimento η che l'ampiezza r della funzione di vicinanza. All'inizio dell'addestramento η assume valori intorno a 1 e decresce gradualmente durante la fase di ordinamento fino a raggiungere un livello molto basso (circa 0,01) che mantiene costante durante la fase di convergenza. In modo simile l'ampiezza r della funzione di vicinanza viene scelta inizialmente in modo da coprire tutte le unità dello strato e decresce gradualmente durante la fase di ordinamento fino a raggiungere un valore 1 o 0 (in quest'ultimo caso vengono modificati solo i pesi sinaptici dell'unità vincente) che mantiene costante durante l'intera fase di convergenza. Questa scelta permette inizialmente a tutte le unità di spostare il proprio vettore di pesi sinaptici sulla base di molti esempi della distribuzione di input; gradualmente, mentre la zona di attività si va restringendo, ciascuna unità si specializza nel rispondere a determinate caratteristiche dell'input.

Nel modello originale delle mappe topologiche le unità possono assumere solo valori binari, ma, come abbiamo avuto modo di osservare nell'esempio di formazione della bolla di attività descritto nel paragrafo precedente, il livello di attivazione è una funzione continua della distanza dall'unità vincente. Una strategia molto semplice per reintrodurre nel modello questa proprietà consiste nel ridefinire la funzione di vicinanza in modo tale che assuma valori continui tra 0 e 1 in funzione della distanza d [Ritter, Martinetz e Schulten 1992]. Nella letteratura troviamo spesso l'impiego di una funzione gaussiana

$$\Lambda(i,i^*) = \exp\left(-\frac{d_{i,i^*}^2}{2\sigma^2}\right)$$

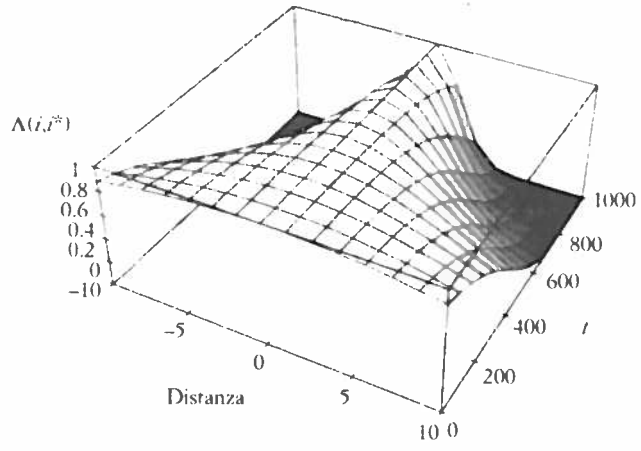


Fig. 4.14. Variazione della funzione di vicinanza di tipo gaussiano durante la fase di ordinamento (1.000 cicli di addestramento): $\sigma_0 = 10$, $\tau = 400$.

dove σ controlla l'ampiezza della funzione (fig. 4.14). Un'altra scelta comune consiste nel far decrescere in modo esponenziale i valori di σ e di η (solo durante la fase di ordinamento)

$$\sigma_t = \sigma_0 \exp\left(-\frac{t}{\tau}\right)$$

e

$$\eta_t = \eta_0 \exp\left(-\frac{t}{\tau}\right)$$

dove t indica il ciclo di apprendimento, τ è una costante temporale, σ_0 e η_0 sono i valori iniziali dei parametri; le costanti temporali τ dei due parametri non devono essere necessariamente uguali. L'apprendimento durante la fase di convergenza procede mantenendo costanti i valori finali dei due parametri.

Mappe di Kohonen: algoritmo

L'algoritmo qui presentato comprende la funzione di vicinanza di tipo gaussiano e la variazione temporale del tasso di apprendimento e dell'ampiezza della bolla di attivazione. La prima decisione consiste nella disposizione delle unità: solitamente viene utilizzata una superficie bidimensionale. Tutte le unità di output ricevono connessioni da tutte le unità di input. La seconda decisione consiste nello stabilire la durata della fase di ordinamento per poter scegliere opportunamente i valori delle costanti temporali: qui assumo arbitrariamente una durata di 1.000 cicli, ma si ricordi che la durata ottimale

dipende dal tipo di mappa e distribuzione dell'input e può essere determinata solo in base a prove empiriche.

1. Inizializzare i pesi sinaptici a valori casuali oppure inizializzarli a qualche pattern estratto dalla distribuzione di apprendimento (l'importante è che i pesi delle unità siano diversi fra di loro); ad esempio

$$w_{ij} = rnd(\pm 0{,}5)$$

inizializzare l'indice temporale a zero

$$t = 0$$

2. Scegliere i valori iniziali del tasso di apprendimento, della durata della fase di ordinamento, dell'ampiezza della funzione di vicinanza e della costante temporale; ad esempio

$$\eta_0 = 1{,}0 \qquad F = 1.000 \qquad \sigma_0 = 8 \qquad \tau = 200$$

è importante che l'ampiezza iniziale della gaussiana generi valori significativamente maggiori di 0,0 per tutte le unità della rete.

3. Estrarre un pattern di input $\mathbf{x}^\mu$ a caso dalla distribuzione ed eseguire i passi 4-8.

4. Individuare l'unità vincente usando il criterio di distanza minima

$$i^* = i \left| \min_i \| \mathbf{x}^\mu - \mathbf{w}_i \| \right.$$

5. Se $t < F$, calcolare il tasso di apprendimento e la funzione di vicinanza

$$\eta_t = \eta_0 \exp\left(-\frac{t}{\tau}\right),$$

$$\Lambda^t(i, i^*) = \exp\left(-\frac{d_{i,i^*}^2}{2\sigma_t^2}\right)$$

dove

$$\sigma_t = \sigma_0 \exp\left(-\frac{t}{\tau}\right);$$

altrimenti usare valori costanti; ad esempio,

$$\eta_t = 0{,}01 \text{ e } \sigma_t = 0{,}01$$

6. Calcolare la modifica sinaptica

$$\Delta w_{ij}^t = \eta_t \Lambda^t(i, i^*)(x_j^\mu - w_{ij})$$

7. Aggiornare i pesi sinaptici

$$w_{ij}^t = w_{ij}^{t-1} + \Delta w_{ij}^t$$

8. Aggiornare l'indice temporale

$$t = t + 1$$

9. Continuare fino a quando i pesi sinaptici diventano stabili (la fase di convergenza dura alcune migliaia di iterazioni).

4.3. Mappe di Kohonen: esempi e applicazioni

Nell'esempio presentato in questo paragrafo è stata impiegata la versione dell'algoritmo descritta sopra. La rete neurale possiede due unità di ingresso e venti unità di output: queste ultime sono disposte in modo lineare su un'unica dimensione. La distribuzione di addestramento è costituita da cento pattern estratti casualmente da una regione dello spazio a forma di C (fig. 4.15, riquadro in alto a sinistra); essi vengono presentati in modo altrettanto casuale alla rete. I pesi sinaptici sono inizializzati a valori casuali compresi fra -1 e $+1$ (fig. 4.15, riquadro in alto a destra), l'ampiezza iniziale della funzione di vicinanza σ_0 è stabilita a 10, il valore iniziale del tasso di apprendimento η_0 a 1,0, il suo valore finale dopo mille iterazioni a 0,1, e la costante temporale τ è fissata a 400. Il restringimento della funzione di vicinanza per questo esempio è raffigurato nella figura 4.14. La rete è stata addestrata per duemila iterazioni. Durante le prime 100 iterazioni (si veda la fig. 4.15) le unità si allineano e avvicinano portandosi verso il centro della distribuzione di addestramento. Successivamente cominciano ad assumere una forma più curva e a distanziarsi in modo da coprire uniformemente la superficie dei pattern di addestramento. Alla fine della fase di ordinamento (intorno alla millesima iterazione), le unità sono disposte in modo da rispecchiare la distribuzione dei pattern di addestramento; inoltre esse sono distanziate in modo abbastanza uniforme tra di loro in modo da «tassellare» uniformemente l'intera superficie di risposta. Durante le mille iterazioni successive (fase di convergenza) avvengono pochissimi cambiamenti – non riportati nella figura – che rispecchiano la microstruttura della distribuzione.

Questo semplice esempio non richiede particolari accorgimenti. Nel caso di situazioni più complesse Kohonen [Kohonen, Kangas e Laaksonen 1992] propone le seguenti considerazioni: *a*) la struttura del reticolo formato dalle unità di uscita dovrebbe possedere un certo grado di asimmetria al fine di stabilizzare il processo di apprendimento; ad esempio la disposizione rettangolare è da preferire a quella quadrata o circolare; *b*) quando i pattern vengono estratti da una distribuzione finita (come nell'esempio sopra riportato), non vi è differenza significativa fra il presentarli in modo casuale o sempre nello stesso ordine; *c*) nel caso in cui si voglia evidenziare la presenza di qualche de-

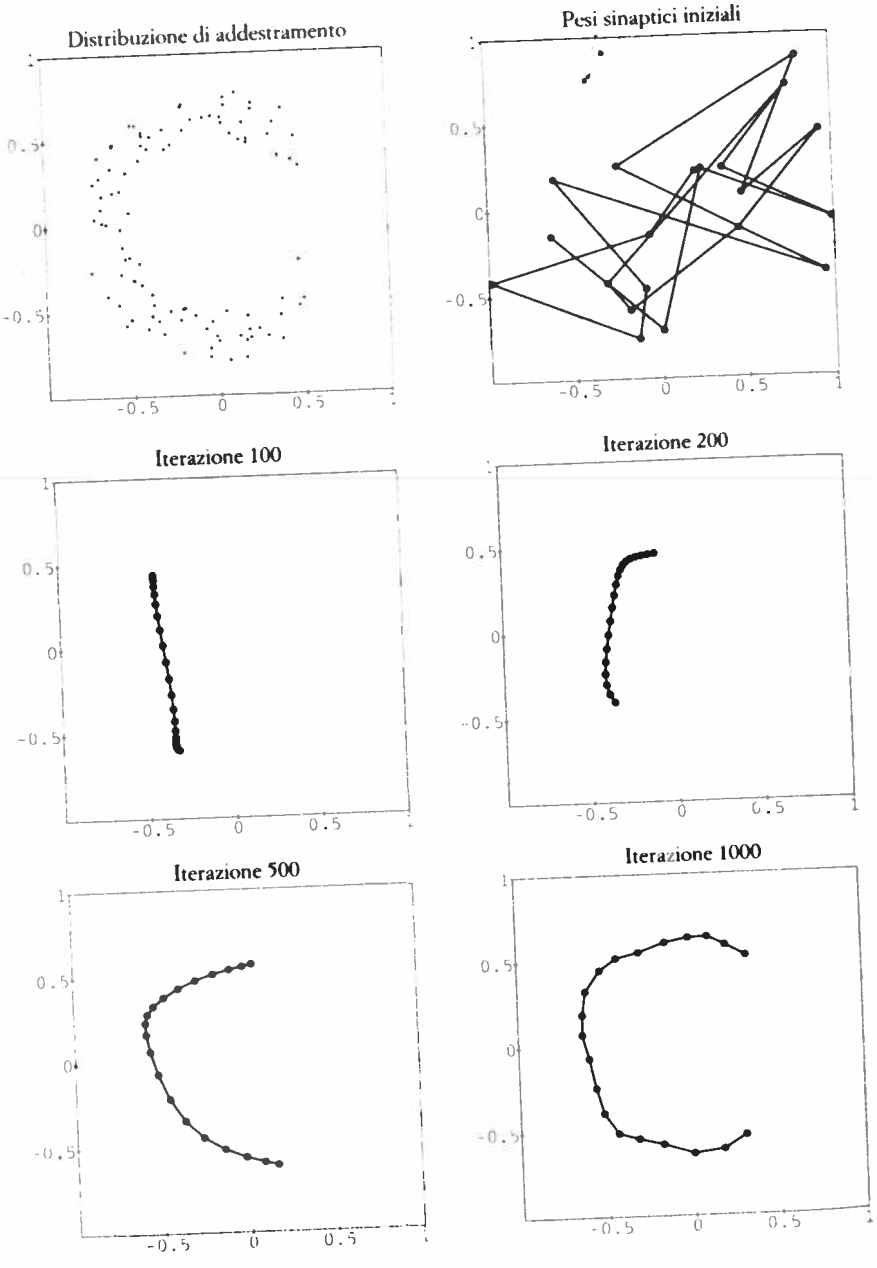


FIG. 4.15. Evoluzione della mappa di Kohonen. Il primo riquadro in alto a sinistra rappresenta la distribuzione dei pattern di addestramento; il riquadro a fianco rappresenta la distribuzione iniziale casuale delle unità della mappa. Ciascun punto rappresenta i pesi sinaptici di un'unità; essi fungono da coordinate dell'unità nello spazio dell'input; le linee congiungono unità confinanti sullo strato di output. I riquadri successivi mostrano lo sviluppo della mappa durante le prime mille iterazioni (fase di ordinamento). Siccome nelle iterazioni successive le modifiche sono minime (fase di convergenza) esse non sono state qui riportate.

terminato pattern che possieda una bassa frequenza di apparizione è sufficiente aumentare il tasso di apprendimento ogni volta che esso viene presentato alla rete; *d*) una misura della qualità di apprendimento è data dalla distanza euclidea media fra i vettori di addestramento e i vettori sinaptici delle corrispondenti unità vincenti $\frac{1}{M}\sum_{\mu=1}^{M}\|\mathbf{x}^{\mu} - \mathbf{w}_{i}\|$ (errore di quantizzazione), dove M è il numero di pattern di addestramento e $\mathbf{w}_i$ è il vettore di pesi sinaptici dell'unità vincente per il pattern $\mathbf{x}^{\mu}$; *e*) l'errore di quantizzazione può essere usato anche per scegliere i pesi sinaptici casuali iniziali (Kohonen suggerisce di provare qualche decina di inizializzazioni e di scegliere quella che genera l'errore minore); *f*) è possibile forzare rappresentazioni in zone desiderate della superficie modificando i pesi delle unità di quella zona invece che i pesi delle unità vincenti.

Kohonen [1988] ha impiegato questo algoritmo nello sviluppo della *macchina da scrivere fonetica*, un sistema che è in grado di trasformare in tempo reale il parlato in testo scritto. Il sistema è composto di tre moduli (fig. 4.16).

Il primo modulo filtra le onde sonore provenienti da un microfono e le converte in segnali digitali; questi vengono analizzati da un algoritmo che esegue la trasformazione rapida di Fourier[12] fornendo un vettore di quindici elementi, ciascuno dei quali contiene l'ampiezza istantanea del segnale entro una delle quindici bande di frequenza (tra 200 Hz e 5.000 Hz). A ciascun componente viene sottratto il valore medio (che corrisponde a portare la distribuzione intorno allo zero) e il vettore viene poi normalizzato. Il secondo modulo è una rete neurale che possiede uno strato bidimensionale di unità competitive. Ciascuna di esse riceve input dal vettore di frequenze e impara a sviluppare una mappa fonetica del segnale: le unità imparano a rispondere alla presenza di fonemi, ma devono essere etichettate a mano dopo la fase di addestramento (presentando un fonema e osservando quale nodo si attiva). È interessante notare che lo scorrimento del parlato corrisponde a una traccia spazio-temporale sullo strato di output della rete individuata dalla sequenza di attivazione di unità vicine (Kohonen ha proposto di impiegare queste tracce per aiutare la comprensione del parlato nei non-udenti); questa traccia non viene però utilizzata nell'applicazione della macchina da scrivere fonetica. L'output istantaneo della rete viene passato a un ulteriore modulo per la disambiguazione degli effetti di coarticolazione (la pronuncia di uno stesso fonema in contesti diversi): questo modulo è un sistema di conoscenze basato su circa ventimila regole programmato in modo efficiente nella memoria permanente di un calcolatore che si occupa di trasformare il risultato in testo scritto. Applicato alla lingua finlandese (madre lingua di Koho-

[12] La trasformazione di Fourier è un metodo che permette di scomporre un'onda sonora in una serie di onde principali; in questo essa può essere *funzionalmente* paragonata al metodo di estrazione delle componenti principali.

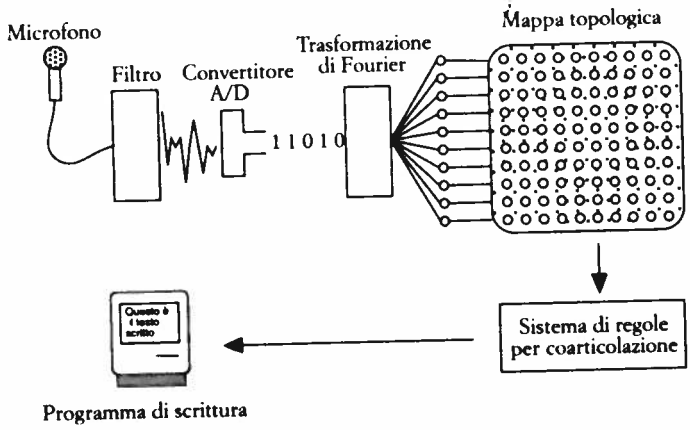


FIG. 4.16. Schema della macchina da scrivere fonetica di Kohonen.

nen), il sistema produce circa il 5 per cento di errori e può essere ulteriormente perfezionato impiegando un numero di mappe parallele che si occupano di diversi tipi di segnale (ad esempio alcune mappe elaborano solamente le consonanti plosive e labiali). Esso può essere facilmente adattato alle caratteristiche di nuovi parlanti utilizzando una tecnica di «quantizzazione vettoriale adattiva» che prenderemo in esame in seguito.

4.4. Mappe di Kohonen: alcune caratteristiche importanti

L'apprendimento nelle mappe di Kohonen tende a sviluppare un insieme di vettori sinaptici che rappresentano in modo ottimale la distribuzione dei vettori di input. Questa proprietà è analoga all'operazione di «quantizzazione vettoriale»: essa consiste nel suddividere una distribuzione di vettori in gruppi simili e individuare un vettore rappresentativo per ciascuno di essi. L'insieme dei vettori rappresentativi – anche detto *codice* – è molto più piccolo rispetto all'insieme dei vettori da elaborare. Questa procedura trova applicazione nella riduzione ottimale di enormi quantità di dati. Il criterio di ottimalità è dato dall'«errore di quantizzazione», ovvero dallo scarto medio tra il vettore originale e il vettore ricostruito tramite il codice di ricostruzione (si veda l'algoritmo). I pesi sinaptici finali di una mappa di Kohonen costituiscono il codice della distribuzione dei vettori di addestramento: ciascuno di essi rappresenta il vettore prototipico di una classe di vettori di input.

L'ordinamento delle unità dopo la fase di addestramento riflette le proprietà della distribuzione di input nel senso che unità vicine rispondono a caratteristiche simili dell'input. Questa caratteristica, assie-

me a quella precedente, risulta in una tassellazione completa della distribuzione di input ove ciascuna unità è collocata al centro del tassello corrispondente (costituito da un gruppo di vettori di input). Va notato che l'effetto dei bordi dello strato di unità fa sì che esse definiscano una zona leggermente più ristretta della distribuzione di input.

La collocazione delle unità riflette anche la densità della probabilità dei vettori di input: un numero maggiore di unità vengono collocate nelle zone corrispondenti ai pattern più frequenti. La mappa risulta quindi distorta quando la densità di probabilità non è uniforme. Si noti però che l'algoritmo non è preciso in quanto tende a sovrastimare zone caratterizzate da bassa densità e a sottostimare zone caratterizzate da alta densità. Per ovviare a questo problema sono state sviluppate delle varianti dell'algoritmo basate su una ridefinizione del processo competitivo e della regola di aggiornamento dei pesi sinaptici [Haykin 1999].

In genere dopo la fase di addestramento tutte le unità della rete vengono sfruttate per rappresentare l'input: la fase iniziale in cui la funzione di vicinanza interessa zone molto larghe permette che le unità che possiedono pesi iniziali molto distanti dai vettori di addestramento siano comunque attivate e possano avvicinare i propri pesi sinaptici verso la zona in cui si collocano i pattern di input.

Al contrario degli algoritmi supervisionati, questo processo autoorganizzato di classificazione non si deteriora quando vengono impiegati dati incompleti per l'addestramento [Samad e Harp 1989]. Un dato incompleto è un vettore costituito da N elementi di cui sono noti solamente P ($P < N$). Questa situazione è molto frequente quando si opera su dati reali. Pensiamo a una compagnia di assicurazioni che debba classificare i clienti in fasce di rischio sulla base dei loro dati personali: siccome non è possibile avere per tutti i clienti lo stesso numero di dati, alcuni vettori presenteranno degli elementi vuoti ($N - P$). La maggior parte degli algoritmi supervisionati richiede l'impiego di vettori completi per l'addestramento perché dati imparziali possono compromettere una corretta classificazione. Le mappe di Kohonen sono invece in grado di formare classificazioni corrette anche con dati imparziali e i pesi finali possono essere impiegati per «indovinare» il valore degli elementi mancanti. Quando si opera con dati incompleti, l'unità vincente corrisponde al *valore massimo* della somma degli scarti quadratici fra i componenti del vettore di input e del vettore sinaptico calcolata nel sottospazio definito dagli elementi disponibili

$$i^* = i \left| \max_i \left(\sum_{j \in P_t} (x_j - w_{ij})^2 \right) \right.$$

dove P_t rappresenta l'insieme di unità di cui sono noti i valori al tempo t. Similmente, la correzione sinaptica avviene solo per i pesi corrispondenti alle unità di cui si conoscono i valori

$$\Delta w_{ij}^{t} = \begin{cases} \eta_t \Lambda^t(i,i^*)(x_j^{\mu} - w_{ij}) & \text{se } j \in P_t \\ 0 & \text{altrimenti} \end{cases}$$

Alla fine della fase di addestramento, è possibile ricavare una predizione statistica degli elementi mancanti osservando i valori sinaptici corrispondenti all'unità vincente.

4.5. «Il vincitore prende tutto»

Una versione più semplice dell'apprendimento competitivo consiste nel modificare solamente i pesi sinaptici dell'unità con attivazione massima: questo corrisponde a una situazione in cui una sola unità risponde per ciascun pattern di input[13]. Un gruppo di unità che impiegano questa semplice regola di apprendimento e non possiedono quindi relazioni geometriche tra di loro viene definito come «gruppo WTA» – dall'inglese *Winner-Take-All* (il vincitore prende tutto). Il criterio di scelta dell'unità vincente è sempre il valore del prodotto interno tra il vettore di input e il vettore di pesi sinaptici e la regola di modifica delle connessioni è equivalente alla regola di Kohonen con funzione di vicinanza uguale a uno. I vettori di pesi sinaptici tendono a suddividere la distribuzione di input in gruppi e a collocarsi al loro centro, ma non forniscono alcuna informazione sulle loro proprietà topologiche dello spazio dei pattern di ingresso. Ciascuna unità si specializza nel rispondere alla classe di input corrispondente al proprio vettore sinaptico. Il risultato è simile a una tassellazione di Voronoi per cui la distribuzione di input viene divisa in regioni dai confini perpendicolari alle linee che uniscono i vettori prototipici: ciascuna regione contiene i vettori di input che si trovano più vicini al prototipo corrispondente che agli altri prototipi.

Queste unità vengono talvolta indicate con il nome di «neuroni della nonna» (*grandmother cells*) per indicare l'alto grado di specializzazione e selettività che esse sviluppano: la presenza di un pattern complesso (come l'immagine della nonna nel campo visivo) verrebbe segnalato dall'attivazione di un unico e specifico neurone. Indubbiamente la capacità di organizzare e categorizzare autonomamente oggetti complessi è una caratteristica molto importante dei sistemi nervosi evoluti e alcuni dati neurofisiologici sembrano indicare la presenza di neuroni nelle aree temporali della corteccia dei primati che rispondono selettivamente a facce o ad altre configurazioni visive complesse [Perrett, Mistlin e Chitty 1987]. Tuttavia questi dati non ci dicono se

[13] Nei sistemi nervosi biologici questo meccanismo di competizione estrema potrebbe emergere dalla presenza di connessioni laterali inibitorie fra unità dello stesso strato e connessioni autoricorrenti eccitatorie (vari schemi competitivi sono stati proposti da diversi autori, ad esempio Feldman e Ballard [1982]; Grossberg [1976a]).

l'attivazione di un neurone specifico sia sufficiente per permettere all'animale di riconoscere un determinato viso: in altre parole, può darsi che quel neurone faccia parte di una popolazione di cellule nervose che possiedono campi recettivi simili e rispondono contemporaneamente allo stesso stimolo[14]. Bisogna inoltre aggiungere che un meccanismo competitivo estremo come lo schema WTA non è resistente al rumore e alle lesioni: se per qualche motivo un'unità viene disabilitata il modello perde la capacità di riconoscere un'intera classe di oggetti. Infine, come già sottolineato nel primo capitolo, le capacità di rappresentazione di un codice locale sono notevolmente limitate rispetto a quelle di un codice distribuito (N oggetti con un codice locale rispetto a 2^N oggetti con un codice distribuito per una rete con N unità di output).

Da un punto di vista strettamente computazionale, lo schema WTA presenta il problema delle «unità morte»: queste sono delle unità che, siccome possiedono dei pesi iniziali distanti da tutti i pattern di input, non vincono mai la competizione e perciò non possono apprendere. Questo problema non ha luogo nel caso delle mappe di Kohonen perché durante la prima fase dell'apprendimento tutte le unità della rete apprendono e di conseguenza tutti i pesi sinaptici vengono avvicinati alla zona della distribuzione dei pattern. Hertz, Krogh e Palmer [1991] elencano alcuni provvedimenti per ovviare al problema delle unità morte negli schemi WTA:

– modificare anche i pesi delle unità perdenti, ma con un tasso di apprendimento molto più basso; questo metodo è stato adottato da Rumelhart e Zipser [1985] in una rete per il riconoscimento di caratteri;

– fare in modo che ciascuna delle N unità del gruppo WTA sia attiva in media $1/N$ delle volte; questo metodo viene soprannominato «meccanismo di coscienza» per indicare che le unità che vincono troppo spesso si sentono colpevoli e innalzano la propria soglia di attivazione [DeSieno 1988];

– aggiungere del rumore casuale ai vettori di input [Szu 1986].

4.5.1. *Quantizzazione vettoriale adattiva*

La quantizzazione vettoriale adattiva (LVQ, *Learning Vector Quantization*) è una procedura di classificazione supervisionata che permette di categorizzare i vettori di input in una serie di categorie predeterminate [Kohonen 1989; 1997]. I vettori di addestramento sono accompagnati dalla classe di appartenenza e ciascuna unità del gruppo

[14] Questa ipotesi, che è stata inizialmente formulata da Hebb, trova riscontro nell'ipotesi di Wolf Singer (fra altri) che il riconoscimento di oggetti avvenga mediante la sincronizzazione dell'attività di una popolazione di neuroni [si veda ad esempio Gray *et al.* 1989; Engel *et al.* 1992].

WTA rappresenta una specifica classe. Dopo aver trovato l'unità vincente (con la solita procedura della distanza euclidea), i pesi sinaptici di quell'unità vengono modificati diversamente a seconda che l'unità rappresenti la classe corretta o quella scorretta

$$\Delta w_{i^*,j} = \begin{cases} +\eta(x_j - w_{i^*,j}) & \text{se } i^* = \text{classe corretta} \\ -\eta(x_j - w_{i^*,j}) & \text{se } i^* = \text{classe scorretta} \end{cases}$$

Questo equivale ad avvicinare i pesi sinaptici al vettore di input se l'unità vincente rappresenta la classe corretta e ad allontanarli se invece rappresenta la classe scorretta. La quantizzazione vettoriale adattiva può essere impiegata per raffinare ulteriormente una mappa di Kohonen: la macchina da scrivere fonetica descritta nel paragrafo 4.3 viene adattata alle caratteristiche individuali di parlanti specifici impiegando questa procedura supervisionata. Poiché la mappa è già formata per un parlante generico, è possibile etichettare ciascuna unità e modificare i pesi sinaptici mentre il nuovo utente legge per circa dieci minuti un testo predeterminato. Kohonen [*ibidem*] ha proposto anche altre due variazioni del metodo di quantizzazione vettoriale adattiva – definite rispettivamente LVQ2 e LVQ2.1 – che migliorano ulteriormente la qualità della categorizzazione.

4.6. Counterpropagation

La classificazione supervisionata di pattern può essere effettuata anche impiegando il metodo di correzione dell'errore descritto nel secondo capitolo. I percettroni presentano però alcuni inconvenienti quando il gruppo di addestramento è molto vasto: richiedono molti cicli di addestramento, rischiano di restare intrappolati in minimi locali (fornendo classificazioni inadeguate) e richiedono l'impiego di dati completi. È utile allora combinare uno strato di unità competitive e uno strato (o più) di percettroni (fig. 4.17). Inizialmente vengono addestrati solamente i pesi delle unità competitive impiegando la regola di apprendimento competitivo standard oppure la regola di Kohonen se le unità possiedono relazioni geometriche fra di loro: la scelta della regola di Kohonen in questo caso è preferibile perché permette l'apprendimento di tutte le unità dello strato. Una volta che lo strato competitivo ha sviluppato una rappresentazione stabile della distribuzione di input, ciascun pattern di addestramento viene presentato nuovamente alla rete e l'attivazione viene propagata attraverso lo strato competitivo fino alle unità di output per cui è disponibile la risposta desiderata: l'errore viene utilizzato per correggere i pesi sinaptici dalle unità competitive alle unità di output con la regola delta (o Back-propagation se vi è più di uno strato di pesi sinaptici). Questo procedimento si rivela molto più robusto e rapido rispetto all'impiego del

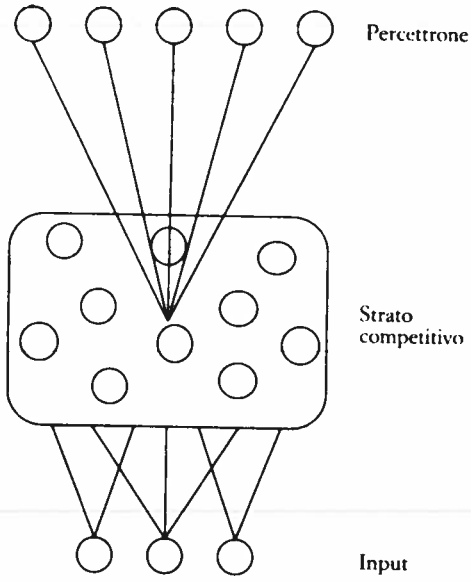


FIG. 4.17. Combinazione di uno strato competitivo e di un percettrone per la classificazione supervisionata di grandi quantità di dati (solo parte delle connessioni sono raffigurate). I due moduli vengono addestrati separatamente in successione.

solo percettrone. La combinazione dei due modelli suddivide il compito di classificazione in modo tale da sfruttare al meglio le proprietà computazionali di ciascuno di essi: lo strato competitivo svolge una pre-elaborazione dei «dati grezzi» raggruppandoli in base alla loro somiglianza, mentre il percettrone realizza l'associazione desiderata a partire da dati già organizzati. Tuttavia non è consigliabile utilizzare questa procedura quando si desidera evidenziare e classificare separatamente dati grezzi simili (come nel caso del compito di parità descritto nel secondo capitolo) perché lo strato competitivo tenderebbe a raggrupparli in un'unica classe.

Questa soluzione ibrida rappresenta lo schema di base delle reti «counterpropagation» [Hecht-Nielsen 1987b], che possiamo liberamente tradurre in «reti a propagazione incrociata». Hecht-Nielsen ha proposto due architetture, di cui una è una versione semplificata con architettura «feedforward». La versione feedforward consiste nella combinazione di uno strato competivo WTA e di uno strato di unità lineari addestrate con la regola delta; i due strati vengono addestrati separatamente come descritto sopra. In una rete con x_k unità di input, h_j unità interne e y_i unità di output, i pesi sinaptici v_{jk} da input a unità interne (strato competitivo) vengono modificati solamente per l'unità vincente j^* in base alla regola competitiva standard

$$\Delta v_{j^*k} = \eta(x_k - v_{j^*k})$$

mentre i pesi sinaptici w_{ij} da unità interne a unità di output vengono modificati con la regola delta

$$\Delta w_{ij} = \eta(t_i - y_i)h_j$$

dove t_i è la risposta desiderata. Si noti però che, siccome solo un'unità interna è attiva (con valore 1) per ciascun pattern e tutte le altre possiedono valore 0, possiamo operare le opportune sostituzioni e riscrivere la regola precedente come

$$\Delta w_{ij^*} = \eta(t_i - w_{ij^*})$$

da cui si osserva che le regole di apprendimento per i due strati assumono la stessa forma. Si noti però un'importante differenza: nel primo strato vengono modificati i pesi sinaptici dell'unità postsinaptica vincente per tutte le connessioni che provengono dall'input, mentre nel secondo strato vengono modificati i pesi sinaptici dell'unità presinaptica vincente per tutte le connessioni che essa emette verso l'output. Queste modalità di apprendimento erano state precedentemente descritte e rispettivamente definite con il nome di apprendimento «instar» e «outstar» da Grossberg [1969].

Il nome «counterpropagation» deriva comunque dalla versione completa del modello in cui lo strato competitivo calcola l'unità vincente in base al segnale proveniente sia dalle unità dello strato di input che dalle unità di output (la cui attivazione è data dalla risposta desiderata). La finalità del modello consiste nell'apprendere l'associazione desiderata tra input e output (risposta desiderata) e nel contempo essere in grado di ricostruire uno qualsiasi dei due vettori se solamente l'altro è noto. Solitamente si abbandona la nozione di risposta desiderata e si parla semplicemente di associazione tra due vettori: il vettore **x** e il vettore **y**. La rete è composta da cinque strati di unità (fig. 4.18): uno strato per il vettore **x**, uno strato per il vettore **y**, uno strato **h** di unità competitive WTA e i due strati $\hat{\mathbf{x}}$ e $\hat{\mathbf{y}}$ di ricostruzione dei rispettivi vettori. L'apprendimento consiste in due fasi. Nella prima fase vengono modificati i pesi sinaptici dello strato competitivo. Entrambi i vettori **x** e **y** vengono presentati alla rete e la loro attivazione viene propagata verso le unità dello strato competitivo **h**; l'unità vincente è quella che possiede i pesi sinaptici v_{jk} e w_{ji} più simili ai vettori di input

$$j^* = j \left| \min_j \left(\sum_k (x_k - v_{jk})^2 + \sum_i (y_i - w_{ji})^2 \right)^2 \right.$$

i pesi sinaptici corrispondenti vengono modificati in base alla regola di apprendimento competitivo standard

$$\Delta v_{j^*k} = \eta(x_k - v_{j^*k}), \quad \Delta w_{j^*i} = \eta(y_i - w_{j^*i})$$

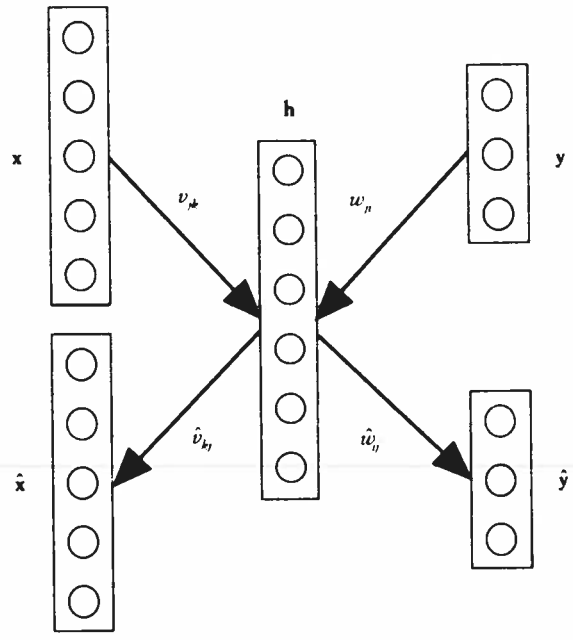


FIG. 4.18. Architettura della rete a propagazione incrociata (counterpropagation): il nome deriva dalla direzione incrociata di propagazione del segnale verso lo strato interno di unità competitive.

Nella seconda fase, dopo che lo strato competitivo ha sviluppato una rappresentazione stabile della distribuzione congiunta delle due sorgenti di input, la rete sviluppa i pesi degli strati di ricostruzione. Entrambi i vettori provenienti dallo strato **x** e **y** vengono nuovamente presentati alla rete e viene calcolata l'unità vincente dello strato interno **h**; la sua attivazione viene propagata verso gli strati di ricostruzione $\hat{\mathbf{x}}$ e $\hat{\mathbf{y}}$ attraverso i rispettivi pesi sinaptici $\hat{v}_{kj}$ e $\hat{w}_{ij}$ che vengono modificati confrontando l'attivazione di ricostruzione con l'attivazione ancora presente negli strati **x** e **y**

$$\Delta\hat{v}_{kj^*} = \eta(x_k - \hat{v}_{kj^*}), \quad \Delta w_{ij^*} = \eta(y_i - \hat{w}_{ij^*})$$

Per rendere chiaro il confronto fra i termini nelle formule precedenti sono stati impiegati gli stessi indici – naturalmente disposti in ordine opposto – per i pesi sinaptici v_{jk} e $\hat{v}_{kj}$, e w_{ji} e $\hat{w}_{ij}$. Tuttavia essi sono distinti e vengono inizializzati con valori diversi. Alla fine dell'addestramento le matrici di pesi tendono ad assumere valori simili. La rete è in grado di completare vettori incompleti e di ricostruire vettori mancanti.

La versione completa della rete counterpropagation funziona in modo corretto solo se la funzione che esprime la trasformazione da **x**

a **y** è invertibile, ovvero se esiste un valore unico per **x** dato **y**. Siccome questa limitazione è abbastanza pesante, la versione feedforward trova un impiego più diffuso. La rete counterpropagation raggiunge livelli di accuratezza simili a quelli Back-propagation, ma richiede un numero molto minore di cicli di addestramento (circa un ordine di grandezza in meno).

4.7. Teoria della Risonanza adattiva

Stephen Grossberg ha osservato che l'apprendimento competitivo standard e la maggior parte degli altri modelli neurali non sono in grado di sviluppare una rappresentazione temporalmente stabile quando esposti a un ambiente arbitrario. Se la distribuzione dei pattern di input subisce delle modifiche nel corso del tempo, la risposta della rete a un determinato pattern sempre identico a se stesso non resta costante; nel caso limite, se la distribuzione continua a mutare, la rete non raggiungerà mai un punto di stabilità. La plasticità dei pesi sinaptici fa sì che le esperienze di apprendimento più recenti cancellino drasticamente le conoscenze acquisite in precedenza: Carpenter e Grossberg hanno mostrato che in alcune situazioni questo fenomeno di instabilità può accadere anche con soli quattro pattern [Carpenter e Grossberg 1987a]. Contrariamente a questa limitazione dei modelli neurali artificiali, noi siamo in grado di riconoscere situazioni, persone e oggetti in ambienti che mutano costantemente e in modo imprevedibile e l'apprendimento di nuove conoscenze non cancella le acquisizioni precedenti. Grossberg [1980] definisce questa limitazione come il *dilemma della stabilità-plasticità*; egli si chiede quali siano i principi che permettono a un sistema di rimanere plastico in risposta a nuovi eventi significativi trascurando nel contempo variazioni irrilevanti e continuando a mantenere una rappresentazione stabile delle conoscenze già acquisite. Per poter essere biologicamente plausibile, un modello computazionale che affronti questi quesiti dovrebbe funzionare in base a principi di auto-organizzazione, ovvero senza la presenza di un insegnante esterno onnisciente.

Il problema sollevato da Grossberg è di estrema importanza perché affronta la validità ecologica di tutti i modelli neurali descritti nei paragrafi e capitoli precedenti. Solitamente il dilemma della stabilità-plasticità viene ignorato perché la distribuzione dei pattern di addestramento è statica: essi sono definiti in una lista predeterminata oppure vengono estratti casualmente da una distribuzione con caratteristiche costanti. In questi casi il termine «stabilità» comunemente usato dai vari autori si riferisce alla «capacità» di una rete neurale di operare all'interno di certi criteri di prestazione prima che entrino in gioco degli elementi di interferenza. In altri casi la stabilità, nel senso definito da Grossberg, viene assicurata a scapito della plasticità: ad esempio, la rete di Kohonen rimane pressoché stabile durante la fase di conver-

genza perché la funzione di vicinanza e il tasso di apprendimento sono prossimi a zero. Quando invece un modello rimane potenzialmente plastico, come ad esempio Back-propagation, la presentazione di input tratti da una nuova distribuzione genera errori che provocano drastiche modifiche dei pesi sinaptici: la rete apprende a rispondere correttamente ai nuovi pattern, ma può cancellare completamente le conoscenze acquisite in precedenza.

La Teoria della Risonanza adattiva (ART, *Adaptive Resonance Theory*) [Grossberg 1976b] rappresenta una proposta formale per la soluzione del dilemma della stabilità-plasticità all'interno di un modello di apprendimento competitivo. Essa si basa su due principi fondamentali: l'esistenza di connessioni di feedback dall'output verso l'input e la presenza di un meccanismo di confronto fra l'attivazione dell'input sensoriale e l'informazione proveniente dalle unità di output. Sulla base di questa teoria Grossberg e Carpenter hanno proposto tre algoritmi neurali, denominati ART1, ART2 e ART3, per l'apprendimento e riconoscimento di pattern: essi funzionano in modo simile, ma si distinguono per il crescente grado di generalità e dettaglio matematico. Ciascuno di essi è composto di una serie di moduli funzionali e unità altamente specializzate che intendono rappresentare determinati circuiti biologici. Grossberg e Carpenter pongono grande attenzione alla plausibilità biologica e psicologica dei loro modelli: parte dei formalismi matematici si preoccupa proprio di descrivere le regole di transizione che avrebbero luogo nei circuiti biologici e le proprietà del modello vengono sempre accuratamente confrontate con i dati sperimentali raccolti su soggetti. Nel prossimo paragrafo prenderemo in esame il modello ART1.

4.7.1. *ART1*

ART1 è il modello più semplice ed è in grado di funzionare solamente con pattern di input binari. Ciononostante la sua architettura e i suoi principi di funzionamento illustrano chiaramente i concetti alla base della Teoria della Risonanza adattiva (fig. 4.19). La rete si compone di due strati di unità: il primo strato (indichiamo queste unità con il simbolo c_i) riceve input dall'ambiente esterno ed esegue il processo di comparazione mentre il secondo strato (indichiamo queste unità con il simbolo r_i) effettua il riconoscimento dei pattern e rappresenta l'output del sistema. Vi sono due tipi di connessioni sinaptiche, uno (w_{ij}) dalle unità di input verso le unità di output e uno (t_{ij}) dalle unità di output verso le unità di input (connessioni di feedback). Nella versione originale le unità di output sono anche collegate fra di loro da connessioni laterali, ma siccome esse in effetti costituiscono un gruppo competitivo di tipo WTA, possiamo trascurarle e scegliere semplicemente di volta in volta l'unità vincente in funzione dell'input netto massimo. Il modello è caratterizzato inoltre dalla presenza di tre unità speciali che

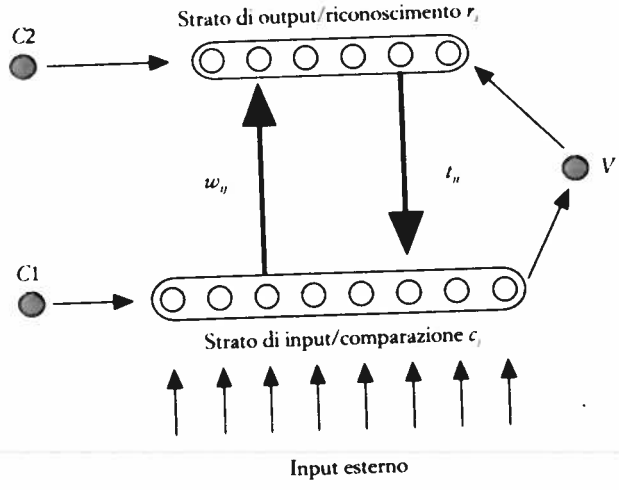


FIG. 4.19. Architettura del modello ART1.

servono a implementare un importante meccanismo di modulazione dell'attenzione e a regolare il funzionamento sequenziale dei due strati: anche per queste unità è possibile semplificare il funzionamento complessivo del sistema sostituendo ad esse delle semplici regole. La prima unità speciale (V) svolge l'importante funzione di stabilire se il processo di comparazione supera una certa soglia di vigilanza ρ e se deve essere creata una nuova unità di output che rappresenti una nuova categoria. L'unità speciale $C1$ esercita un controllo su tutte le unità di input e decide se esse svolgono la funzione di input o di comparazione: essa assume valore 1 se vi è un pattern valido (diverso da zero) proveniente dall'esterno, ma diventa 0 se qualche unità si attiva nello strato di output. L'unità speciale $C2$ esercita invece il controllo su tutte le unità di output: essa assume valore 1 per ogni pattern di input, ma diventa 0 quando il risultato della fase di comparazione non supera il test di vigilanza operato dall'unità V; in questo caso le unità di output vengono disabilitate (rimesse a zero) e un nuovo processo competitivo decide la classe di appartenenza del pattern di input. La caratteristica centrale del modello consiste nella trasmissione reciproca e ciclica (*risonanza*) di ciascun pattern fra i due strati della rete fino a quando esso viene riconosciuto e classificato. Durante questo processo di risonanza la rete modifica automaticamente la struttura sinaptica se il pattern è nuovo (sconosciuto) oppure se esso possiede caratteristiche tali per cui le conoscenze acquisite precedentemente richiedono qualche adattamento; i pattern familiari vengono invece riconosciuti automaticamente e non provocano modifiche sinaptiche. Il modello non richiede la distinzione artificiale tra una fase di apprendimento e una fase di generalizzazione (come nella maggior parte degli algoritmi considerati in precedenza), bensì funziona sempre in modo autonomo ed è in gra-

do di apprendere in continuazione pur mantenendo una rappresentazione stabile delle conoscenze già acquisite.

Il funzionamento di ART1 può essere schematizzato in cinque fasi.

Inizializzazione. Il primo passo consiste nel decidere il numero di unità di input e di output: le prime sono definite automaticamente dalla dimensionalità dei pattern di input, mentre per le seconde è sufficiente stabilire un numero massimo M. La rete utilizzerà solamente i nodi di output necessari a classificare in modo adeguato (in base al criterio di vigilanza descritto più sotto) i pattern che vengono presentati. I pesi sinaptici di feedback t_{ji} assumono inizialmente tutti lo stesso valore 1, mentre i pesi sinaptici w_{ij} vengono inizializzati come segue

$$w_{ij} = \frac{L}{L - 1 + N}$$

dove $L > 1$ ed N è il numero di unità dello strato di input. La soglia di vigilanza ρ regola il grado di dettaglio con cui la rete classifica i pattern: valori bassi provocano la creazione di poche classi, mentre valori alti causano distinzioni più precise fra i vari pattern di input e generano la creazione di un numero maggiore di classi: ρ può assumere valori compresi tra zero e uno.

Riconoscimento. Un pattern binario $\mathbf{x}^{\mu}$ estratto dall'ambiente esterno viene presentato ai nodi di input della rete

$$c_i = x_i^{\mu}$$

la cui attivazione viene propagata verso i nodi di output. Durante questa fase iniziale l'unità speciale $C1$ assume valore 1. L'attivazione dei nodi di output è data dalla somma pesata degli input

$$r_i = \sum_{j}^{N} w_{ij} x_j^{\mu}$$

siccome essi costituiscono un gruppo competitivo, il nodo che possiede attivazione massima viene messo a 1 mentre tutti gli altri vanno a 0. Poiché inizialmente i pesi sinaptici w_{ij} sono tutti uguali fra di loro, tutte le unità di output possiedono la stessa attivazione: in questi casi conviene scegliere l'unità con l'indice i minore in modo da assicurarsi uno sfruttamento progressivo delle unità. Si noti anche che siccome i pattern di input sono binari e i pesi sinaptici sono sempre positivi, le attivazioni dei nodi sono sempre positive. Durante questa fase l'unità speciale $C2$ assume valore 1. Il nodo vincente i^* rappresenta è il candidato per la classificazione e apprendimento del pattern $\mathbf{x}^{\mu}$. Tuttavia, prima di prendere questa decisione la classe rappresentata dal nodo vincente i^* viene confrontata con il pattern originale.

Confronto. In questa fase l'unità speciale $C1$ diventa 0 e i nodi di input sono pronti ad eseguire il confronto fra il segnale proveniente dall'unità vincente e quello proveniente dall'ambiente esterno. L'attivazione delle unità di input viene ora ricalcolata come segue

$$c_j = (t_{ji^*} y_{i^*}) x_j^\mu = t_{ji^*} x_j^\mu$$

Ora l'unità speciale V confronta il rapporto $\|\mathbf{c}\| / \|\mathbf{x}^\mu\|$ fra il nuovo vettore di attivazione e il pattern di input rispetto alla soglia di vigilanza ρ, dove

$$\|\mathbf{c}\| = \sum_j^N c_j, \qquad \|\mathbf{x}^\mu\| = \sum_j^N x_j$$

rappresentano la lunghezza dei due vettori.

Se

$$\frac{\|\mathbf{c}\|}{\|\mathbf{x}^\mu\|} \geq \rho$$

il nodo vincente i^* viene confermato per la classificazione del pattern $\mathbf{x}^\mu$ e si procede alla fase di modifica dei pesi sinaptici; altrimenti il controllo fallisce e si procede alla fase di ricerca di un altro nodo più idoneo alla classificazione.

Ricerca. Se il controllo eseguito dall'unità V durante la fase di comparazione è fallito, l'unità speciale $C2$ diventa zero inibendo così l'unità vincente i^*, tutti i nodi restanti della rete vengono messi a zero (le unità speciali a 1) e viene iniziata una nuova fase di riconoscimento. Lo stesso pattern di input $\mathbf{x}^\mu$ viene presentato nuovamente alle unità di input e l'attivazione viene propagata alle unità di output; questa volta però l'unità vincente precedente (che ora assume valore -1) viene esclusa dalla competizione. Il nuovo nodo vincente propaga la propria attivazione all'indietro attraverso le connessioni t_{ji} e si procede a una nuova fase di comparazione e controllo rispetto alla soglia di vigilanza. Se il controllo fallisce nuovamente, anche quest'ultima unità vincente viene messa a -1 e viene effettuata una nuova fase di riconoscimento (ora vi sono due unità che non partecipano alla scelta dell'unità vincente). Questo processo di ricerca, o risonanza tra input e output, procede fino a quando l'unità vincente riesce a superare il test di vigilanza. Se nessuna delle unità di output risulta idonea, si presentano tre possibili alternative: aggiungere alla rete ulteriori unità di output e continuare la fase di ricerca, ridurre il valore della soglia di vigilanza ρ, oppure rinunciare a classificare il pattern $\mathbf{x}^\mu$.

Modifica sinaptica. Se il controllo operato dall'unità speciale V ha successo, entrambe le connessioni w_{i^*j} e t_{ji^*} dell'unità vincente i^* vengono modificate come segue

$$w_{i^*j}^{t+1} = \frac{Lc_j}{L - 1 + \|\mathbf{c}\|}$$

dove il vettore **c** corrisponde all'attivazione delle unità di input durante la fase di comparazione, e

$$t_{ji^*}^{t+1} = t_{ji^*} x_j^m$$

Dall'equazione precedente risulta che i pesi sinaptici di feedback t_{ji}, una volta che assumono valore 0, non possono più tornare a 1: questo particolare assicura la stabilità della rete neurale. Inoltre, anche i pesi sinaptici w_{ij} dall'input all'output diventano ben presto stabili e mantengono poi lo stesso valore ogni volta che un pattern familiare (già classificato) viene presentato alla rete.

L'algoritmo presentato sopra viene spesso indicato come *apprendimento rapido* perché i pesi sinaptici w_{ij} vengono modificati interamente per ogni presentazione di un pattern; nell'*apprendimento lento* le regole di modifica sinaptica sono leggermente diverse e i pesi sinaptici w_{ij} vengono modificati solo in parte (grazie all'introduzione di un tasso di apprendimento) per ciascun pattern che viene presentato alla rete. In pratica, l'apprendimento rapido in ART1 porta agli stessi risultati ed è la modalità usata più frequentemente nella letteratura; per i modelli successivi – ART2 e ART3 – la scelta di una delle due modalità può invece comportare risultati diversi in determinate condizioni.

Il funzionamento di ART1 merita alcune considerazioni. Innanzitutto si noti che la scelta della soglia di vigilanza è molto importante perché regola il grado di discriminazione della rete e il numero di classi che vengono a formarsi: se ρ è modesto, ad esempio inferiore a 0,4 il meccanismo di vigilanza accetterà un gran numero di pattern per una determinata unità vincente; al contrario, se ρ è prossimo a 1, la fase di ricerca diventerà più accurata e piccole differenze tra i pattern di input saranno sufficienti per la creazione di una nuova classe. Quest'ultima condizione rende il modello molto sensibile al rumore e limita la sua applicazione ingegneristica, ma si presta molto bene a modellare processi cognitivi di tipo attentivo. In alcune condizioni gli esseri umani sembrano prestare grande attenzione a piccole differenze, mentre in altre situazioni le stesse differenze non provocano cambiamenti comportamentali o la necessità di esercitare un processo di apprendimento. Secondo Grossberg questo fenomeno può essere modellato attraverso un meccanismo di rinforzo che proviene da altri moduli neurali il quale modifica il valore della soglia di vigilanza a seconda delle esigenze esterne.

Il riconoscimento di pattern in ART1 è sensibile al contesto. L'operazione di confronto eseguita dall'unità speciale V si basa sul rapporto fra la *norma* dei vettori di input e di comparazione: questo fatto tiene implicitamente conto della proporzione di unità attive in ciascuno dei

due vettori ed evita che il confronto si basi unicamente sul numero assoluto di unità attive. Il risultato è che la presenza di un determinato elemento attivo in un pattern dominato da zero è molto più importante per il test di vigilanza rispetto alla presenza dello stesso elemento attivo in un pattern composto soprattutto da elementi attivi. La stessa proprietà è presente nella regola sinaptica dei pesi w_{ij} dove troviamo al denominatore la presenza del vettore di comparazione **c**: siccome esso rappresenta in effetti il numero di elementi attivi nei pesi di feedback e nel vettore di input, i vettori di comparazione con molti elementi attivi causano lo sviluppo di pesi sinaptici più piccoli e generano quindi un meccanismo di «auto-riduzione» che permette alla rete di distinguere due pattern tra di loro anche se uno di essi appartiene a un sottogruppo dell'altro.

Un'altra caratteristica molto importante della Teoria della Risonanza adattiva è che i pattern familiari – quelli già classificati – vengono riconosciuti più rapidamente dei pattern nuovi a prescindere dalla loro dimensione e complessità perché vengono subito identificati dal nodo corretto; i pattern nuovi invece devono passare attraverso un processo di ricerca che si protrae più a lungo se il pattern presenta poche caratteristiche familiari o è completamente nuovo. Questa proprietà rispecchia le modalità di riconoscimento degli esseri umani.

Infine, da un punto di vista tecnico, ART1 richiede un numero di cicli di addestramento molto minore rispetto agli altri algoritmi competitivi; questa velocità viene però controbilanciata dal processo di risonanza che può essere relativamente lungo specialmente nei primi cicli di addestramento. Le unità dello strato di output vengono utilizzate in modo progressivo, ma risentono delle stesse limitazioni di base dei modelli competitivi WTA: la classificazione richiede un grande numero di unità e la plausibilità biologica di tale codifica è limitata perché è molto sensibile al rumore e alle lesioni (la perdita di una di queste unità risulta nel mancato riconoscimento di un'intera classe di oggetti).

I modelli ART2 e ART3 mantengono la stessa architettura e gli stessi principi di funzionamento di ART1, ma ART2 [Carpenter e Grossberg 1987b] è in grado di operare anche su vettori continui (una chiara presentazione di ART2 e del problema dell'inferenza catastrofica è data in Penna [1992]) e ART3 [Carpenter e Grossberg 1990] include una serie di equazioni non-lineari che modellano le dinamiche dei neurotrasmettitori.

Capitolo 5

1. Tempo e memoria nelle reti neurali

Come accennato nel capitolo 2, vi sono dei problemi che è possibile pensare di risolvere con una rete neurale solo se questa è in grado di cogliere la struttura temporale di un processo. Questo accade, ad esempio, quando il compito assegnato alla rete implica la generazione, l'elaborazione o la predizione di sequenze temporali. In questi casi la rete neurale deve produrre una risposta che dipende non solo dal valore attuale degli ingressi e delle uscite dei singoli nodi, ma anche da quello che essi hanno assunto in passato. Perché ciò possa accadere la separazione temporale tra l'istante in cui l'informazione si è presentata alla rete e l'istante attuale deve essere colmata, tipicamente dotando la rete di qualche forma di memoria.

Nel capitolo 2 si è visto che una possibile risposta a questa esigenza consiste nell'utilizzo di un insieme di elementi di ritardo collegati in cascata a costituire un registro a scorrimento che memorizza gli ultimi n campioni della sequenza di ingresso e li mette a disposizione di una rete neurale priva di memoria (fig. 2.16). Il risultato è noto con il nome di rete neurale con ritardo temporale (*Time-Delay Neural Network*, o TDNN). Uno dei limiti delle TDNN è rappresentato dalla necessità di stabilire fin dall'inizio il numero di campioni della sequenza di ingresso che si desidera memorizzare. Dato che l'informazione risalente a istanti temporali non rappresentati nel registro a scorrimento va irrimediabilmente persa, è necessario dotare il sistema di un numero di elementi di ritardo tale da assicurare la memorizzazione di tutta l'informazione rilevante per l'esecuzione del compito previsto. È facile immaginare come la stima di questo numero possa risultare problematica.

Un approccio alternativo che non presenta questa limitazione è quello proposto da Elman [1990] (fig. 2.15). Esso prevede ad ogni passo temporale la memorizzazione dell'informazione corrispondente all'attivazione delle unità nascoste della rete su appositi neuroni detti «di contesto». Tale informazione viene poi messa a disposizione delle

unità nascoste nel successivo passo temporale. Più in generale, è possibile ottenere un risultato analogo utilizzando una rete dotata di collegamenti ricorrenti, non necessariamente del tipo proposto da Elman. A differenza della rete di ritardo impiegata dalle TDNN, la struttura ricorrente permette il mantenimento in circolo di una traccia dell'attività della rete risalente a istanti arbitrariamente remoti. Ovviamente i ripetuti passaggi dell'informazione attraverso i nodi che costituiscono le strutture ricorrenti comportano una modifica dell'informazione originaria, che non potrà in generale essere ricostruita esattamente a partire dallo stato della rete a un determinato istante.

Entrambi questi approcci all'implementazione di una dimensione temporale nelle reti neurali si basano su unità neurali che non sono intrinsecamente in grado di esibire una dinamica temporale e che per farlo devono quindi affidarsi a elementi ausiliari di memorizzazione o all'interazione con altre unità neurali. Nel resto di questo capitolo esamineremo la possibilità di definire dei modelli di unità neurali dotate di memoria locale e in grado quindi di esibire una propria dinamica temporale. Una rete neurale composta da queste unità sarà quindi anch'essa in grado di esibire tale dinamica, indipendentemente dalla topologia delle sue connessioni. Naturalmente nulla impedisce di utilizzare queste unità nell'ambito di una rete che utilizza anche sistemi di memorizzazione non locale come quelli discussi in precedenza. In effetti la presenza di una molteplicità di tipologie di memoria operanti a diverse scale temporali (memorie a breve, medio e lungo termine) sembra essere una caratteristica comune degli organismi viventi. A questo proposito va osservato che la variazione dei parametri di una rete neurale (ad esempio i pesi sinaptici) a seguito di un processo di apprendimento costituisce a sua volta una modalità di memorizzazione dei processi temporali a cui la rete è stata sottoposta.

2. Un modello elementare di neurone dinamico

Per definire un primo modello di neurone dotato di dinamica propria prendiamo come punto di partenza il modello classico di neurone artificiale presentato nel capitolo 1 (fig. 1.3), caratterizzato da un insieme di segnali di ingresso x_j, da un insieme corrispondente di pesi sinaptici w_{ij}, da un potenziale di attivazione a_i, da una soglia ϑ_i e da una risposta y_i.

Dato che siamo interessati ad aggiungere una dimensione temporale al modello, dobbiamo considerare esplicitamente questi termini come funzioni del tempo. A questo proposito è necessario distinguere tra modelli a tempo continuo e modelli a tempo discreto. I primi rappresentano il tempo come una variabile reale continua t e operano quindi in termini di funzioni di variabile reale (che indicheremo ad esempio con il simbolo $y_i(t)$), mentre i secondi si riferiscono al caso in cui la variabile temporale possa essere rappresentata da un indice intero n, e operano con sequenze di valori (che indicheremo ad esempio

con il simbolo $y_i(n)$). Il processo che porta un modello a tempo discreto dal generico istante n all'istante successivo $n + 1$, corrisponde all'esecuzione di un *passo temporale*.

Si osservi che un modello a tempo discreto può corrispondere a un processo intrinsecamente discreto (come è il caso della sequenza di caratteri presentata alla rete NETtalk descritta nel capitolo 2) o a un processo continuo la cui variabile temporale sia stata discretizzata. Quest'ultimo caso equivale tipicamente a considerare una sequenza di istanti $\{..., -\Delta t, 0, \Delta t, 2\Delta t, ..., n\Delta t, ...\}$ dove Δt è l'intervallo di discretizzazione. In questo modo è possibile istituire una corrispondenza tra modelli continui e modelli discreti, al fine ad esempio di simulare con un calcolatore digitale l'evoluzione di una rete neurale a tempo continuo.

2.1. Il modello a tempo discreto

2.1.1. *Il modello classico riconsiderato*

Prima di presentare un modello a tempo discreto di neurone artificiale dotato di una dinamica temporale non banale, è opportuno riconsiderare brevemente il modello classico che è stato introdotto nel capitolo 1, per valutare l'effetto della presenza esplicita del parametro temporale. Secondo tale modello, ad ogni istante temporale il neurone riceve in ingresso un insieme di valori $x_j(n)$ e presenta in uscita un valore $y_i(n)$. L'istituzione del legame tra gli ingressi e la risposta prevede il calcolo di un potenziale di attivazione $a_i(n)$ che è determinato univocamente dagli ingressi e a sua volta determina univocamente la risposta.

Per facilitare la transizione dal modello classico (privo di dinamica) ai modelli dinamici, è opportuno supporre che vi sia un ritardo temporale Δ (corrispondente all'esecuzione di un passo temporale) tra l'istante di applicazione di un insieme di ingressi a un neurone e la produzione della risposta corrispondente in uscita. In altre parole, quando gli ingressi $x_j(n)$ corrispondenti a un dato istante vengono applicati al neurone, essi vengono utilizzati per calcolare il valore $a_i(n + 1)$ che il potenziale d'attivazione assumerà alla fine del passo temporale in corso, mentre il valore attuale del potenziale di attivazione $a_i(n)$ e dell'uscita $y_i(n)$ corrispondente, rimangono momentaneamente invariati.

Possiamo rappresentare graficamente questo processo con uno schema a blocchi in cui gli ingressi del neurone sono separati dall'uscita da un elemento di ritardo Δ, che memorizza il valore attuale del potenziale di attivazione e mantiene l'uscita invariata mentre il nuovo valore del potenziale di attivazione viene calcolato a partire dagli ingressi attuali (fig. 5.1).

L'espressione che governa l'evoluzione del potenziale di attivazione

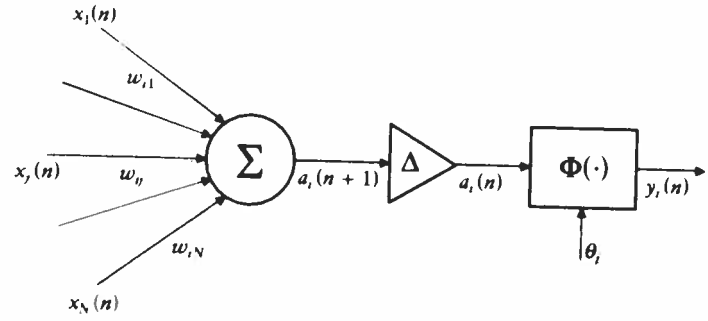


FIG. 5.1. Per semplificare la transizione dal modello classico ai modelli dinamici, è utile includere nel modello classico a tempo discreto del neurone artificiale un elemento Δ, che introduce un ritardo tra l'istante in cui il nuovo valore del potenziale di attivazione viene calcolato, e quello in cui viene presentata in uscita la risposta corrispondente.

del modello rappresentato nella figura 5.1 è

$$a_i(n+1) = \sum_{j=1}^{N} w_{ij} x_j(n)$$

che, utilizzando la notazione vettoriale, possiamo riscrivere come

$$a_i(n+1) = \mathbf{w}_i \cdot \mathbf{x}(n)$$

La risposta $y_i(n)$ si ottiene ad ogni istante applicando la funzione di attivazione $\Phi(\cdot)$ (ad esempio una delle funzioni sigmoidali rappresentate in figura 1.4) al potenziale di attivazione $a_i(n)$ presente in quell'istante, secondo la relazione

$$y_i(n) = \Phi(a_i(n) - \vartheta_i)$$

dove ϑ_i è il valore della soglia.

2.1.2. *Generazione di dinamiche temporali più complesse*

Il modello di neurone artificiale rappresentato nella figura 5.1 ha una dinamica che consiste semplicemente nella presenza di un ritardo di un passo temporale tra l'applicazione degli ingressi e la produzione della risposta corrispondente. Tale neurone è in grado di «ricordare» solo il valore dell'input netto che era presente al passo temporale appena trascorso. Per ottenere a partire da tale struttura un modello di neurone artificiale dotato di una dinamica temporale più complessa possiamo ispirarci alla struttura ricorrente proposta da Elman che prevede il rinvio all'ingresso del neurone di una copia della sua risposta al passo temporale precedente. Visto che nel modello di figura 5.1 dispo-

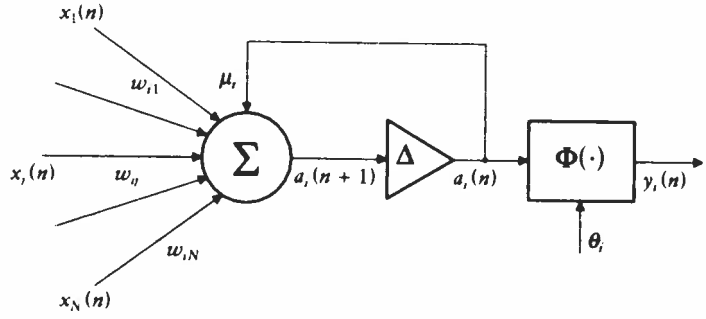


FIG. 5.2. Un modello elementare di neurone dinamico a tempo discreto si ottiene aggiungendo al modello tradizionale una connessione ricorrente che riporta in ingresso il valore attuale del potenziale di attivazione.

niamo di una copia del potenziale di attivazione $a_i(n)$, possiamo utilizzarlo al posto della risposta $y_i(n)$ utilizzata nel modello di Elman. Il modello assume dunque la struttura illustrata in figura 5.2.

L'esame della figura 5.2 ci permette di dedurre l'espressione che governa l'evoluzione del potenziale di attivazione del neurone dinamico a tempo discreto che, in notazione vettoriale, risulta essere

$$a_i(n+1) = \mu_i\, a_i(n) + \mathbf{w}_i \cdot \mathbf{x}(n)$$

Rispetto alla formula relativa al modello classico vi è di nuovo solamente la presenza del contributo relativo al valore del potenziale di attivazione al passo temporale precedente, pesato da un parametro μ_i[1]. La funzione che fornisce la risposta del neurone rimane invece invariata rispetto al modello tradizionale, ed è quindi ancora $y_i(n) = \Phi(a_i(n) - \vartheta_i)$.

Algoritmo:[2] Evoluzione (sincrona) di una rete neurale dinamica a tempo discreto

1. Assegnare il valore iniziale del potenziale di attivazione a tutti i nodi della rete in base alla relazione $a_i \leftarrow a_i(0)$ (ad esempio ponendo $a_i(0) = 0$).
2. Ripetere i punti da 3 a 5 per il generico passo temporale n, con n che va da 0 fino al passo temporale finale desiderato.

[1] Si noti che la presenza del termine aggiuntivo porta a una distinzione tra input netto $\mathbf{w}_i \cdot \mathbf{x}$ e potenziale di attivazione a_i, termini che nel modello classico tendono a confondersi dato che coincidono numericamente.

[2] Gli algoritmi descritti in questo capitolo non sono ottimizzati dal punto di vista dell'occupazione di memoria o dei tempi di esecuzione dato che, per chiarezza, si è preferito distinguere variabili a cui pure l'algoritmo assegna nella maggior parte dei casi lo stesso valore numerico (in questo caso, ad esempio, y_i e x_i).

3. Calcolare il valore della risposta di tutti i neuroni in base alla relazione $y_i \leftarrow \Phi(a_i - \vartheta_i)$.

4. Assegnare il valore attuale **x** agli ingressi di tutte le unità neurali, ponendo $x_i \leftarrow y_i$ per quelli collegati alle uscite di altri neuroni e $x_i \leftarrow s_i(n)$ per quelli collegati agli ingressi esterni **s** della rete.

5. Aggiornare il valore del potenziale di attivazione di tutte le unità neuronali ponendo $a_i \leftarrow \mu_i a_i + \mathbf{w}_i \cdot \mathbf{x}$.

Per capire come la presenza dell'elemento di ritardo in retroazione sul nodo sommatore del neurone influisce sull'evoluzione del potenziale di attivazione, analizziamo il caso semplice in cui il vettore **x** di ingresso è posto a zero (evoluzione libera) e il potenziale di attivazione evolve a partire da un valore iniziale $a_i(0)$. Dopo il primo passo temporale si avrà $a_i(1) = a_i(0)\,\mu_i$, dopo il secondo si avrà $a_i(2) = a_i(1)\,\mu_i = a_i(0)\,\mu_i^2$, e in generale all'$n$-esimo passo risulterà

$$a_i(n) = a_i(0)\,\mu_i^n, \qquad n \geq 0$$

Dunque la memoria dell'attivazione passata evolve nel tempo con un andamento esponenziale legato al valore del parametro μ_i. In particolare, nel caso in cui risulti $0 \leq \mu_i < 1$ questa evoluzione si caratterizza come una «dimenticanza» progressiva del valore iniziale del potenziale di attivazione. La figura 5.3 presenta alcuni esempi di tale tipo di evoluzione. Si osservi che ad $a_i(0)$ può risultare assegnato un valore negativo; in tal caso l'andamento della sequenza risulta speculare rispetto all'asse orizzontale relativamente a quanto illustrato in figura 5.3. Il caso $\mu_i = 0$ corrisponde al tradizionale modello di neurone, che risulta quindi un caso particolare del modello di figura 5.2.

Possiamo eseguire un calcolo simile per valutare l'effetto di un segnale presente su un ingresso del neurone a un determinato istante. Per semplificare le cose, supponiamo che all'istante 0 il potenziale di attivazione sia nullo e che contemporaneamente il vettore degli ingressi valga **x**(0), per annullarsi poi in tutti gli istanti successivi. In base all'equazione di evoluzione del potenziale di attivazione, avremo inizial-

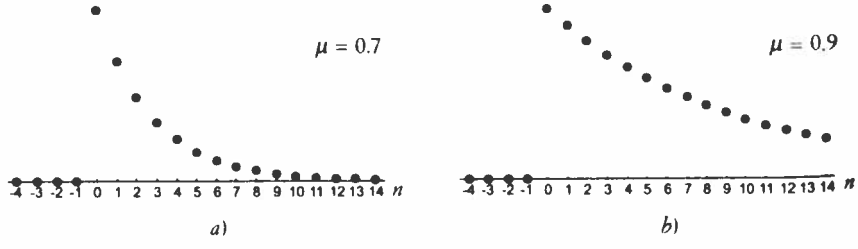


FIG. 5.3. L'evoluzione temporale libera del potenziale di attivazione del modello di neurone di figura 5.2 rappresentata per due diversi valori del parametro μ.

mente $a_i(1) = \mathbf{w}_i \cdot \mathbf{x}(0)$, ossia l'input netto determinerà univocamente il potenziale di attivazione. In seguito il potenziale di attivazione così ottenuto evolverà secondo l'andamento esponenziale che abbiamo dedotto in precedenza, ossia in base alla relazione $a_i(n) = \mu_i^{n-1} a_i(1) = \mu_i^{n-1} \mathbf{w}_i \cdot \mathbf{x}(0)$. Nel caso in cui lo stimolo sia presentato all'ingresso del neurone al generico istante k anziché all'istante 0, risulterà

$$a_i(n) = \mu_i^{n-k-1} \mathbf{w}_i \cdot \mathbf{x}(k) , \qquad n > k$$

dato che in questo caso l'evoluzione del potenziale di attivazione avrà luogo solo per $(n - k)$ passi temporali (di cui il primo serve ad acquisire l'input netto al potenziale di attivazione). In virtù della linearità del sistema che governa l'evoluzione del potenziale di attivazione l'effetto totale sarà la somma dei vari effetti considerati separatamente[3].

Ne risulta la possibilità di concepire il processo di evoluzione del potenziale di attivazione del neurone come composto da due fasi: nella prima gli ingressi all'istante attuale agiscono sul valore del potenziale di attivazione e vanno a far parte di una «memoria comune»; successivamente, questa memoria evolve in modo tale da «dimenticare» progressivamente i contributi acquisiti. Più precisamente, se si considera a partire dall'istante iniziale $n = 0$ l'evoluzione del potenziale di attivazione in presenza di sequenze di ingresso generiche si ottiene l'espressione

$$a_i(n) = a_i(0)\,\mu_i^n + \sum_{k=0}^{n} \mu_i^{n-k-1} \mathbf{w}_i \cdot \mathbf{x}(k)$$

In questa relazione ogni termine – sia esso il valore del potenziale di attivazione iniziale $a_i(0)$ o uno dei contributi $\mathbf{w}_i \cdot \mathbf{x}(k)$ dovuti agli ingressi presenti negli istanti successivi – viene moltiplicato per un fattore esponenziale che pesa il contributo del termine al valore del potenziale di attivazione in funzione della sua distanza temporale dall'istante attuale e dà origine a una risposta parziale relativa al termine in esame. La risposta complessiva è la somma delle singole risposte. Dunque il funzionamento del neurone che abbiamo presentato può essere descritto come un processo *di somma e dimenticanza graduale* dei segnali che via via si presentano agli ingressi.

Si noti che una volta che il contributo di una sequenza di ingresso è entrata a far parte della memoria comune rappresentata dal potenziale di attivazione, la sequenza stessa non è più ricostruibile univocamente a partire dal valore istantaneo del potenziale di attivazione stesso. Il processo comporta quindi una perdita di informazione. Questo

[3] Si osservi che questo non sarebbe vero in generale se la connessione ricorrente operasse sulla risposta $y_i(n)$ (come accade ad esempio nel modello di Elman), poiché in tal caso entrerebbe in gioco la funzione di attivazione $\Phi(\cdot)$, che è in genere non lineare.

fenomeno è tipico dell'operazione di combinazione di sequenze temporali rappresentato dalla formula appena descritta, operazione che prende il nome di *convoluzione temporale* della sequenza $\mathbf{w}_i \cdot \mathbf{x}(k)$ con la sequenza $(\mu_i)^k$ [Bracewell 1986].

2.2. Il modello a tempo continuo

Per sviluppare un modello a tempo continuo di neurone con memoria possiamo prendere l'ispirazione dal modello a tempo discreto sviluppato nella sezione precedente. A questo scopo riscriviamo l'espressione che governa l'evoluzione del potenziale di attivazione mettendo in evidenza al primo membro la sua variazione tra due passi temporali adiacenti

$$a_i(n+1) - a_i(n) = (\mu_i - 1)\, a_i(n) + \mathbf{w}_i \cdot \mathbf{x}(n)$$

se dividiamo questa espressione per l'intervallo temporale Δt otteniamo l'espressione

$$\frac{a_i(n+1) - a_i(n)}{\Delta t} = \frac{(\mu_i - 1)}{\Delta t} a_i(n) + \frac{\mathbf{w}_i}{\Delta t} \cdot \mathbf{x}(n)$$

In questo modo il primo membro diventa un rapporto incrementale $\Delta a_i(n)/\Delta t$ che può essere interpretato come una versione discreta della derivata temporale da_i/dt del potenziale di attivazione. Sorvolando su alcuni dettagli relativi all'istituzione della corrispondenza tra l'equazione discreta e quella continua, possiamo dunque considerare quest'equazione come una controparte dell'equazione differenziale[4]

$$\frac{da_i(t)}{dt} = -\frac{1}{\tau_i} a_i(t) + \mathbf{w}'_i \cdot \mathbf{x}(t)$$

che definirà di conseguenza l'evoluzione del potenziale di attivazione del neurone dinamico a tempo continuo. Il modello è completato dall'espressione

$$y_i(t) = \Phi(a_i(t) - \vartheta_i)$$

che, come accadeva nel caso discreto, definisce risposta del neurone in base alla funzione di attivazione $\Phi(\cdot)$ e al bias ϑ_i.

[4] L'apice aggiunto al simbolo che rappresenta i pesi sinaptici nell'equazione del modello continuo ricorda che l'interpretazione della corrispondenza tra i parametri modello discreto e quelli modello continuo richiede qualche cautela. Ulteriori dettagli a questo proposito verranno forniti parlando della discretizzazione e simulazione al calcolatore dei modelli continui.

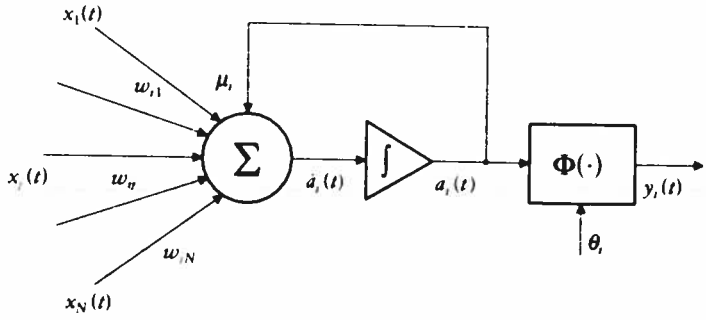


FIG. 5.4. Il modello elementare di neurone dinamico a tempo continuo.

La figura 5.4 illustra la struttura corrispondente a questa espressione. Tale struttura è simile a a quella del modello discreto rappresentato in figura 5.2, salvo che all'elemento di ritardo che nel modello a tempo discreto trasforma il valore futuro del potenziale di attivazione nel suo valore attuale si sostituisce un integratore che trasforma la derivata del potenziale di attivazione nel potenziale di attivazione stesso. Analogamente al caso discreto, possiamo determinare l'evoluzione libera del potenziale di attivazione (ossia quella che si ottiene ponendo $\mathbf{x}(t) = 0$) a partire da un valore iniziale $a_i(0)$. Essa sarà governata dalla relazione

$$a_i(t) = a_i(0)\, e^{-\frac{t}{\tau_i}}, \qquad t \geq 0$$

(fig. 5.5) come è possibile verificare sostituendo questa espressione nell'equazione differenziale in cui si sia posto $\mathbf{x}(t) = 0$. Si noti che la condizione $0 \leq \mu_i < 1$ che nel caso discreto assicura la «dimenticanza» progressiva degli ingressi passati si traduce qui nella condizione $\tau_i > 0$ applicata alla *costante di tempo* τ_i della funzione esponenziale. Più in generale, la funzione esponenziale continua $e^{-\frac{t}{\tau_i}}$ assume ora il ruolo giocato in precedenza dalla funzione esponenziale discreta μ_i^n. La corrispondenza si estende all'espressione che governa l'evoluzione del potenziale di attivazione, dato che è possibile dimostrare che nel caso

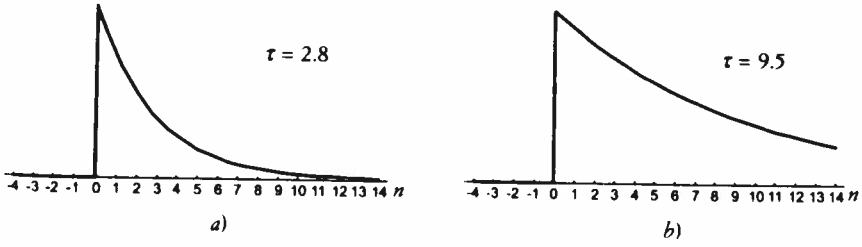


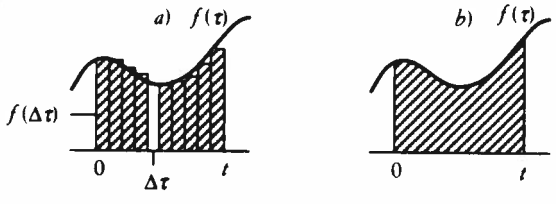
FIG. 5.5. L'evoluzione temporale libera del potenziale di attivazione del modello a tempo continuo per due valori della costante di tempo τ.

continuo essa corrisponde alla relazione

$$a_i(t) = a_i(0)\, e^{-\frac{t}{\tau_i}} + \int_0^t e^{-\frac{t-\tau}{\tau_i}} \mathbf{w}_i \cdot \mathbf{x}(\tau)\, d\tau$$

Confrontando questa espressione con quella relativa all'evoluzione del potenziale di attivazione nel caso discreto si può notare che in entrambi i casi il valore iniziale del potenziale di attivazione decade esponenzialmente per tutto l'intervallo di osservazione. Inoltre, i contributi istantanei dell'input netto $\mathbf{w}_i \cdot \mathbf{x}(\cdot)$ vengono sommati al potenziale di attivazione via via che si presentano e, una volta acquisiti, il loro contributo decade esponenzialmente. Dato che ora il tempo è rappresentato da una variabile continua, dobbiamo sostituire la sommatoria $\sum_{k=0}^{n}$ presente nel caso discreto con l'integrale[5] $\int_0^t \cdot\, d\tau$. La dinamica del

[5] Abbiamo a che fare con una funzione $e^{-\frac{t-\tau}{\tau_i}} \mathbf{w}_i \cdot \mathbf{x}(\tau)$ della variabile reale τ (indichiamola con $f(\tau)$ per comodità) che produce un effetto pari alla «somma» dei valori che essa assume negli istanti che compongono l'intervallo $[0, t]$. Dato che un intervallo contiene un'infinità continua di istanti temporali, non possiamo utilizzare una normale somma per rappresentare questo effetto. Possiamo però darne una rappresentazione approssimata considerando i valori che la funzione assume in un numero finito di istanti appartenenti all'intervallo. Ad esempio prenderemo in considerazione i valori $\{f(0), f(\Delta\tau), f(2\Delta\tau), \ldots\}$ che la funzione assume negli istanti $\{0, \Delta\tau, 2\Delta\tau, \ldots\}$. Dato che desideriamo prendere in considerazione l'effetto prodotto dall'azione della funzione durante tutto il sottointervallo $\Delta\tau$, il singolo contributo che comparirà nella somma non potrà essere semplicemente $f(n\Delta\tau)$, ma sarà invece $f(n\Delta\tau)\,\Delta\tau$. L'espressione approssimata dell'effetto globale sarà quindi $\sum_n f(n\Delta\tau)\,\Delta\tau$, che corrisponde graficamente all'area sottesa da un'approssimazione «a scalini» della funzione originaria (fig. (*a*)). Se immaginiamo di ridurre l'ampiezza dei singoli sottointervalli, possiamo migliorare via via la precisione con cui calcoliamo l'area effettivamente sottesa dalla funzione, fino a ottenere il valore esatto (fig. (*b*)). Chiamiamo questo valore esatto, *integrale definito* della funzione $f(\tau)$ sull'intervallo $[0, t]$ e lo indichiamo con l'espressione $\int_0^t f(\tau)\, d\tau$ (in cui il simbolo $\int$ è una «s» deformata, che ci ricorda la sommatoria da cui siamo partiti). Il termine $d\tau$ che compare nell'integrale allude al ruolo dell'intervallo $\Delta\tau$ nel processo di somma originario, e può essere pensato come un «intervallo infinitesimo». Inoltre, il termine $d\tau$ indica la variabile rispetto alla quale eseguiamo l'integrazione. Si noti che, una volta assegnato a t un valore, l'espressione $a = \int_0^t f(\tau)\, d\tau$ corrisponde a un numero reale (l'area della curva sottesa dalla funzione nell'intervallo $[0, t]$). Se però consideriamo t come un parametro (che può anche comparire nella funzione, che indichiamo allora con $f(t, \tau)$), otteniamo un numero per ogni valore di t. In questo caso, che è quello con cui avremo a che fare in questo capitolo, l'operazione di integrazione dà origine a una funzione $a(t) = \int_0^t f(t, \tau)\, d\tau$.



modello continuo del neurone può essere quindi interpretata come un processo di *integrazione e dimenticanza graduale* dei segnali che vengono presentati all'ingresso del neurone. Come nel caso discreto, la combinazione del segnale $e^{-\frac{t}{\tau_i}}$ con il segnale $\mathbf{w}_i \cdot \mathbf{x}(t)$ realizzata dalla formula appena descritta prende il nome di *convoluzione temporale.*

Questo modello di acquisizione degli ingressi al potenziale di attivazione è noto nella letteratura anglosassone con il nome di *leaky integrator* (integratore con perdite), con riferimento alla sua realizzazione in termini di circuito elettrico, che verrà descritta brevemente in seguito. Esso gode di una certa popolarità nell'ambito della modellistica neurale per il fatto di rappresentare un primo semplice modello dotato di plausibilità biologica per la descrizione della dinamica del potenziale di membrana dei neuroni degli organismi viventi. Le reti neurali che fanno uso del modello che abbiamo presentato prendono il nome di *continuous-time recurrent neural networks* [Beer e Gallagher 1992; Beer 1995]. Come verrà chiarito in seguito, esse sono in grado di manifestare delle dinamiche non banali e trovano quindi impiego in particolare per la realizzazione di sistemi dinamici e di sistemi di controllo per agenti che devono interagire in modo complesso con l'ambiente (ad esempio robot autonomi). In questi casi, per semplificare l'elaborazione degli input sensoriali torna spesso utile aggiungere all'equazione differenziale presentata in precedenza un termine $s_i(t)$ che rappresenta direttamente il segnale fornito da un sensore. L'equazione differenziale diventa allora

$$\frac{da_i(t)}{dt} = -\frac{1}{\tau_i} a_i(t) + \mathbf{w}_i \cdot \mathbf{x}(t) + s_i(t)$$

Il modello di figura 5.4 può essere evidentemente esteso a questo caso inserendo un segnale di ingresso corrispondente a $s_i(t)$ sul nodo sommatore.

3. Generalizzazione del modello elementare

I modelli presentati nella sezione precedente sono solo il punto di partenza per la definizione di modelli più complessi, flessibili e, in alcuni casi, biologicamente plausibili. Per semplificare la presentazione di tali modelli è opportuno introdurre una nuova rappresentazione per i modelli elementari. Con riferimento al modello continuo rappresentato in figura 5.4 (ma un discorso del tutto analogo vale per il modello discreto) osserviamo che la presenza del collegamento ricorrente dà luogo alla comparsa di una dinamica esponenziale che opera su tutti i segnali che provengono dagli ingressi. Possiamo rappresentare questa dinamica per mezzo di un elemento posto subito dopo il nodo sommatore che, come previsto dall'equazione di evoluzione del potenziale

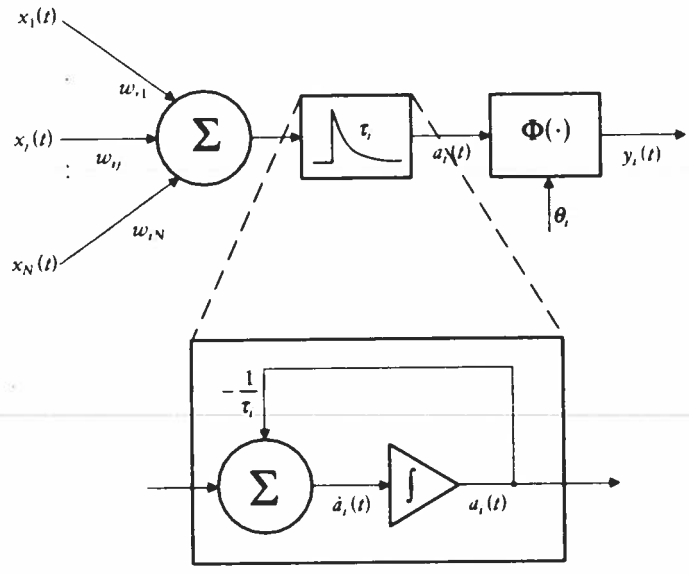


FIG. 5.6. Il modello di figura 5.4 può essere rappresentato in modo più compatto facendo uso di un elemento che segue il nodo sommatore e produce la dinamica esponenziale corrispondente alla presenza del collegamento ricorrente. Una rappresentazione del tutto analoga può essere utilizzata per il caso discreto.

di attivazione, trasformi il segnale corrispondente all'input netto nei contributi esponenziali che vanno a costituire il potenziale di attivazione (fig. 5.6). Questo tipo di rappresentazione è utilizzabile sia per i modelli a tempo continuo che per quelli a tempo discreto.

3.1. Il modello con dinamica all'ingresso

La dinamica temporale del modello di figura 5.6 è determinata unicamente dal parametro τ_i (μ_i nel caso discreto) e quindi sarà necessario agire su questo parametro per ottenere la dinamica desiderata. Eventualmente, si potrà inserire tale parametro tra quelli che la rete sarà chiamata ad apprendere nella fase di configurazione. È facile immaginare che una flessibilità molto maggiore da questo punto di vista si potrebbe ottenere se ogni ingresso potesse dar luogo a una sua propria dinamica. Oltre ad aumentare il numero di gradi di libertà del modello, quest'idea ha una certa attrattiva dal punto di vista della modellazione neurofisiologica, dato che la morfologia dei dendriti e le caratteristiche elettrochimiche delle relative sinapsi possono dar luogo in un neurone biologico a diverse tipologie di risposta per lo stesso segnale afferente. Per aggiungere al modello di figura 5.6 questi ulteriori gradi di libertà è sufficiente spostare a monte del nodo sommatore

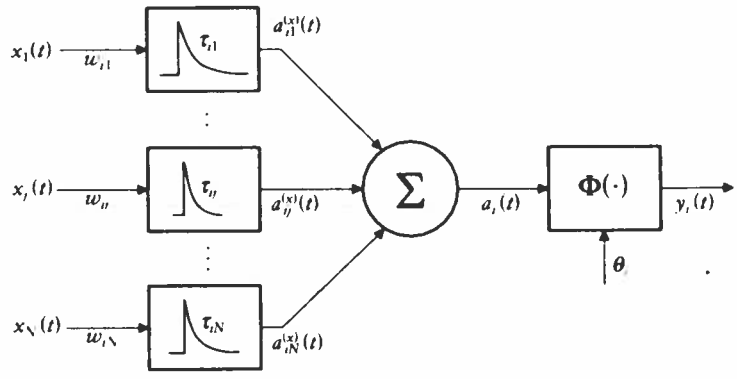


FIG. 5.7. È possibile rendere più flessibile il modello elementare supponendo che ogni ingresso possa dare luogo a una dinamica indipendente. Si ottiene così il modello di unità neurale con dinamica all'ingresso.

l'elemento che determina la dinamica, come illustra la figura 5.7. In tal modo ogni ingresso viene dotato di una sua dinamica indipendente, che è possibile specificare assegnando la corrispondente costante di tempo τ_{ij}. Indicheremo questo modello con il nome di *unità neurale con dinamica all'ingresso.*

Dato che ora ogni ingresso è dotato di una «memoria» separata, torna utile definire delle nuove variabili $a_{ij}^{(x)}(t)$ che chiameremo «potenziali di attivazione parziali», che sommati daranno il potenziale di attivazione totale $a_i(t)$, secondo la relazione

$$a_i(t) = \sum_{j=1}^{N} a_{ij}^{(x)}(t)$$

Potremo così scrivere l'equazione differenziale (o alle differenze, nel caso discreto) che governa l'evoluzione di questi potenziali

$$\frac{da_{ij}^{(x)}(t)}{dt} = -\frac{1}{\tau_{ij}} a_{ij}^{(x)}(t) + w_{ij} x_j(t)$$

che una volta integrata dà luogo a

$$a_{ij}^{(x)}(t) = \int_0^t e^{-\frac{t-\tau}{\tau_i}} w_{ij} x_j(\tau)\, d\tau$$

(dove per semplicità si è ipotizzato $a_{ij}^{(x)}(0) = 0$). Un discorso del tutto analogo può essere ripetuto per il caso discreto, che darà luogo all'algoritmo seguente:

Algoritmo: Evoluzione di una rete neurale a tempo discreto con dinamica all'ingresso

1. Assegnare il valore iniziale del potenziale di attivazione a tutti i nodi della rete in base alle relazioni

$$a_{ij}^{(x)} \leftarrow a_{ij}^{(x)}(0)$$

$$a_i \leftarrow \sum_{j=1}^{N} a_{ij}^{(x)}$$

(ad esempio ponendo $a_{ij}^{(x)}(0) = 0$).

2. Ripetere i punti da 3 a 5 per il generico passo temporale n, con n che va da 0 fino al passo temporale finale desiderato.

3. Calcolare il valore della risposta di tutti i neuroni in base alla relazione $y_i \leftarrow \Phi(a_i - \vartheta_i)$.

4. Assegnare il valore attuale **x** agli ingressi di tutte le unità neurali, ponendo $x_i \leftarrow y_i$ per quelli collegati alle uscite di altri neuroni e $x_i \leftarrow s_i(n)$ per quelli collegati agli ingressi esterni **s** della rete.

5. Aggiornare il valore del potenziale di attivazione di tutte le unità neuronali ponendo

$$a_{ij}^{(x)} \leftarrow \mu_{ij} a_{ij}^{(x)} + w_{ij} x_j$$

$$a_i \leftarrow \sum_{j=1}^{N} a_{ij}^{(x)}.$$

Fin qui abbiamo considerato dinamiche temporali ottenute come composizioni di sole funzioni esponenziali, poiché questo è l'andamento che viene prodotto dal collegamento ricorrente che ci ha consentito di trasformare il modello classico di neurone artificiale nel modello dinamico. Nulla però ci impedisce di utilizzare al posto delle funzioni esponenziali altri tipi di funzioni. Ciò può servire ad esempio a rappresentare delle operazioni di filtraggio, dei ritardi di propagazione, o altri tipi di trasformazioni applicate ai segnali di ingresso. A questo scopo sostituiamo nel modello di figura 5.7 gli elementi che producono una risposta esponenziale con elementi in grado di produrre delle generiche risposte $\eta_{ij}(t)$. Il risultato è rappresentato in figura 5.8.

Nel nuovo modello il potenziale di attivazione sarà ottenuto come la somma, estesa a tutti gli ingressi, dei potenziali di attivazione parziali $a_{ij}^{(x)}(t)$, che a loro volta sono ottenuti mediante la combinazione (convoluzione) del segnale di ingresso $x_j(t)$ con la funzione $\eta_{ij}(t)$ che specifica la dinamica relativa a quel particolare ingresso. Le espressioni corrispondenti (nell'ipotesi che valga la condizione $a_{ij}^{(x)}(0) = 0$) sono[6]

[6] Si noti che in questo caso non scriviamo le corrispondenti equazioni differenziali, dato che le risposte sono specificate direttamente in termini di funzioni $\eta_{ij}(t)$.

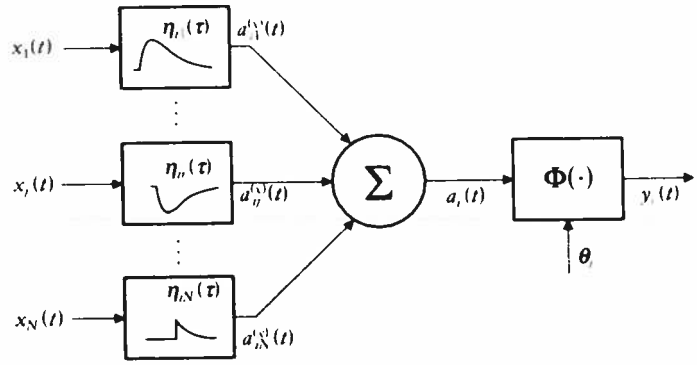


FIG. 5.8. Come ulteriore generalizzazione del modello con dinamica all'ingresso (fig. 5.7) si può immaginare che la dinamica temporale relativa agli ingressi sia descritta da funzioni $\eta_{ij}(t)$ che non sono vincolate ad essere esponenziali.

$$a_{ij}^{(x)}(t) = \int_0^t \eta_{ij}(t-\tau)\, x_j(\tau)\, d\tau$$

Le funzioni $\eta_{ij}(t)$ prendono il nome di *nuclei* della risposta e il loro andamento sarà in genere specificato da uno o più parametri che corrispondono alla costante di tempo τ_{ij} presente nel caso esponenziale. Si noti che nella definizione dei nuclei $\eta_{ij}(t)$ l'effetto del peso sinaptico w_{ij} è stato inglobato nella definizione del nucleo corrispondente. In alternativa, scriveremo $\eta_{ij}(t) = w_{ij}\varepsilon_{ij}(t)$. Il caso elementare può essere ottenuto da quello più generale ponendo semplicemente $\eta_{ij}(t) = w_{ij}\varepsilon(t)$.

Tra le funzioni che sono comunemente usate come nuclei [Gerstner 1999; Floreano e Mattiussi 2001] vi è la funzione esponenziale (fig. 5.9a)

$$\varepsilon_1(t) = \begin{cases} e^{-\frac{t-\tau_d}{\tau_m}} & t \geq \tau_d \\ 0 & t < \tau_d \end{cases}$$

in cui τ_m agisce come costante di tempo e il nuovo termine τ_d rappresenta un ritardo (*delay*) tra l'arrivo del segnale di ingresso e l'inizio del suo contributo al potenziale di attivazione (inteso a modellare un ritardo di propagazione del segnale nel suo viaggio dal neurone presinaptico a quello postsinaptico), e le funzioni

$$\varepsilon_2(t) = \begin{cases} te^{-\frac{t-\tau_d}{\tau_m}} & t \geq \tau_d \\ 0 & t < \tau_d \end{cases}$$

(fig. 5.9b),

$$\varepsilon_3(t) = \begin{cases} e^{-\frac{t-\tau_d}{\tau_m}} - e^{-\frac{t-\tau_d}{\tau_s}} & t \geq \tau_d \\ 0 & t < \tau_d \end{cases}$$

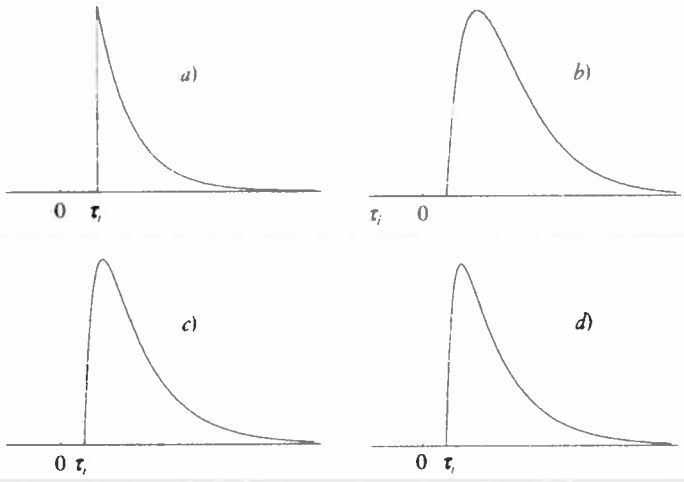


FIG. 5.9. Alcune funzioni comunemente utilizzate per generare la dinamica della risposta neuronale.

(fig. 5.9c),

$$\varepsilon_4(t) = \begin{cases} e^{-\frac{t-\tau_d}{\tau_m}}\left(1 - e^{-\frac{t-\tau_d}{\tau_s}}\right) & t \geq \tau_d \\ 0 & t < \tau_d \end{cases}$$

(fig. 5.9d). Si noti che le funzioni $\varepsilon_3(t)$ e $\varepsilon_4(t)$ danno luogo allo stesso tipo di risposta e differiscono solo per il modo in cui è specificata la costante di tempo del termine a cui si deve la fase crescente della risposta. In particolare, nell'analogia biologica possiamo interpretare τ_m e τ_s (o $\tau_{s'}$) come corrispondenti, rispettivamente, alla dinamica della membrana e della sinapsi.

Le reti neurali che utilizzano unità basate sul modello neuronale dinamico illustrato dalla figura 5.8 possiedono un'interessante caratteristica relativa alla loro capacità di dar luogo a dinamiche temporali molto generali. Nel capitolo 2 si è accennato al fatto che le reti neurali multistrato basate sul modello classico sono approssimatori universali di funzioni [Haykin 1999]. Analogamente, è possibile dimostrare che, utilizzando il modello di figura 5.8 in combinazione con una rete multistrato classica, è possibile realizzare reti in grado di approssimare una vasta classe di risposte temporali dotate della proprietà di «dimenticanza graduale» degli ingressi [Sandberg e Xu 1997a, 1997b; Haykin 1999].

Malgrado la ricchezza del modello appena presentato e la molteplicità delle dinamiche a cui è in grado di dar luogo, siamo ancora ben lontani da quello che si va scoprendo relativamente alla complessità dell'elaborazione degli segnali afferenti che caratterizza i neuroni dei

sistemi viventi [Koch 1999; Koch e Segev 1998]. Avremo modo di soffermarci in seguito sulle possibili conseguenze di tale complessità sulla formulazione e implementazione di modelli dotati di plausibilità biologica. Per il momento ci limitiamo a segnalare il fatto che nei nostri modelli abbiamo ipotizzato l'invarianza temporale dei nuclei $\eta_n(t)$ che descrivono la dinamica dei singoli ingressi. Le osservazioni neurofisiologiche suggeriscono invece che un modello più accurato, che approssimi almeno in parte la ricca dinamica delle sinapsi, dovrebbe fare uso di nuclei che variano in funzione del valore presente e passato del potenziale di attivazione e dei segnali presenti all'ingresso [Markram e Tsodyks 1996; Maass e Sontag 2000; Fortune e Rose 2001]. Inoltre, l'ipotesi di linearità dell'azione degli ingressi sul potenziale di attivazione è, in molti casi, solo una grossolana approssimazione del reale processo di integrazione dei segnali afferenti che ha luogo sui dendriti e il soma dei neuroni naturali. Ad esempio, sembra non doversi escludere la possibilità che in questo processo due o più segnali interagiscano in modo moltiplicativo o addirittura polinomiale [Koch 1999].

Un'altra strada per il raffinamento dei modelli presentati si basa sull'abbandono del modello puntiforme del neurone per tener conto della struttura spaziale del corpo neuronale e dell'albero dendritico. Ciò dà luogo a strutture nelle quali i vari segnali di ingresso si propagano e interagiscono tra di loro lungo strutture distribuite nello spazio prima di arrivare a determinare il potenziale di membrana (che corrisponde al potenziale di attivazione del nostro modello). Ne derivano operazioni di filtraggio ed elaborazione spaziotemporali ben più complesse di quanto ottenibile con i modelli presentati fin qui. Non entreremo nel dettaglio di queste modalità più sofisticate di elaborazione dei segnali di ingresso (descritte ad esempio in Koch e Segev [1998]) e ci limiteremo invece a riconsiderare il meccanismo di produzione della risposta neuronale.

4. Neuroni impulsivi («Spiking Neurons»)

Nei modelli fin qui considerati è presente una dinamica temporale solo nel processo che porta i segnali di ingresso ad influire sul potenziale di attivazione. La generazione della risposta neuronale vi è invece rappresentata per mezzo di una trasformazione che tiene conto esclusivamente del valore istantaneo del potenziale di attivazione. Dato che, solitamente, la funzione di attivazione è una funzione sigmoidale che istituisce una corrispondenza biunivoca tra il potenziale di attivazione e la risposta, in questi modelli l'informazione che viene comunicata da una unità neuronale all'altra è sostanzialmente il valore istantaneo del potenziale di attivazione.

Nella nostra analogia con i neuroni naturali il potenziale di attivazione corrisponde al potenziale di membrana e dunque, se l'analogia fosse completa, dovremmo aspettarci che il valore di quest'ultimo fos-

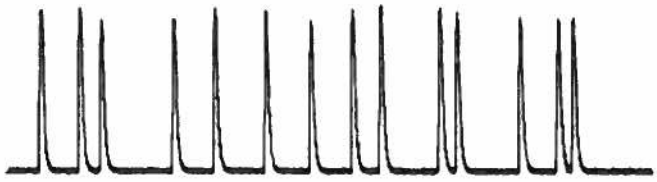


FIG. 5.10. La rappresentazione idealizzata di una sequenza di impulsi neuronali.

se l'informazione scambiata tra i neuroni dei sistemi viventi. Nella maggior parte dei casi si osserva invece che la comunicazione tra neuroni utilizza un segnale composto da sequenze di impulsi noti con il nome di *potenziali d'azione* (fig. 5.10). Come spiegato nel capitolo 1, i potenziali d'azione sono delle perturbazioni importanti del potenziale di membrana che si generano nel corpo neuronale e si propagano lungo l'assone per mezzo di un processo attivo autorigenerante che sfrutta l'energia immagazzinata nella membrana dell'assone e nelle sue immediate adiacenze.

La necessità per i neuroni naturali di ricorrere a un processo di trasmissione attivo è legato alle caratteristiche fisiche della membrana cellulare e dei sistemi di regolazione ad essa connessi. Tali caratteristiche fanno sì che piccole perturbazioni locali del potenziale di membrana subiscano una notevole attenuazione quando ci si sposta anche di pochissimo dal punto in cui la perturbazione ha avuto origine. Ne consegue che la trasmissione di informazione tra neuroni basata sulla propagazione passiva del potenziale di membrana è utilizzabile solo su distanze dell'ordine del millimetro (e effettivamente si riscontra l'impiego di tale tipo di comunicazione in alcune categorie di neuroni che comunicano solo a brevissima distanza, presenti nella retina dei vertebrati e in alcuni circuiti sensomotori degli invertebrati [Anderson 1995; Koch 1999]). Se consideriamo i neuroni naturali in una prospettiva ingegneristica, la presenza dei potenziali d'azione può essere dunque vista innanzitutto come la soluzione a un problema tecnico legato alla necessità di disporre di una comunicazione affidabile che possa operare su distanze superiori al millimetro (e che nell'uomo possono tranquillamente raggiungere l'ordine di grandezza del metro).

Le caratteristiche «esplosive» dei processi che danno origine ai potenziali d'azione fanno sì che la forma dell'impulso risultante sia una conseguenza dello stato elettrochimico della regione in cui il processo ha luogo. È quindi difficile immaginare che la forma esatta dell'impulso sia rilevante per la trasmissione dell'informazione. Possiamo quindi interpretare il potenziale d'azione come un processo stereotipato a cui assegnare una rappresentazione convenzionale atta a segnalare solamente la presenza o assenza dell'impulso. In questo senso, l'impiego di una comunicazione basata su sequenze di potenziali d'azione anziché su una grandezza fisica variabile con gradualità può ricordare la relazione che esiste tra i sistemi di memorizzazione e comunicazione

basati su una codifica digitale e quelli che fanno uso di una rappresentazione analogica (si pensi ad esempio ai compact disc in rapporto ai dischi in vinile, o alle trasmissioni radio in FM e AM). Rispetto alla rappresentazione analogica la codifica digitale mostra in genere una maggior resistenza al rumore (ad esempio, con un'opportuna scelta dei segnali che codificano i simboli 0 e 1, un rumore che rende irriconoscibile il segnale analogico può non essere in grado di trasformare un simbolo binario nell'altro) e facilita le operazioni di rigenerazione del segnale originario. D'altro canto, dato che servono in genere molti impulsi per codificare ogni campione del segnale analogico, e che è necessario campionare con una cadenza sufficientemente elevata, ciò avviene solitamente al prezzo di una maggior velocità di variazione richiesta ai segnali che trasportano l'informazione digitale (è questo uno dei motivi per i quali l'elettronica di consumo si è diffusa inizialmente in forma analogica ed ha iniziato la sua transizione verso il formato digitale solamente quando lo sviluppo tecnologico ha reso sufficientemente economica la manipolazione di segnali a frequenza elevata).

Da questo punto di vista, i neuroni biologici appaiono a tutta prima caratterizzati da una frequenza massima di operazione piuttosto modesta, se rapportata ai tempi di reazione che si osservano negli organismi viventi e che sono loro indispensabili per sopravvivere. Questa apparente contraddizione sembra testimoniare l'impiego da parte dei sistemi nervosi naturali di tecniche di codifica molto raffinate che danno luogo ad un uso ottimale degli impulsi nervosi, al punto da mettere in alcuni casi gli organismi in grado di agire in base all'informazione rappresentata dalla presenza di singoli potenziali d'azione [Rieke *et al.* 1997].

Va fatto notare, a questo proposito, che uno dei vantaggi legati all'impiego di impulsi consiste nella possibilità di contrassegnare particolari istanti temporali e di rivelare coincidenze e correlazioni temporali predefinite. Sarà questo, infatti, uno dei casi in cui l'impiego dei modelli neuronali impulsivi che descriveremo in seguito potrà risultare preferibile a quello dei modelli neuronali dinamici non impulsivi presentati in precedenza. Nei casi in cui invece il sistema nervoso è chiamato a operare su scale temporali che consentono la presenza di numerosi impulsi, si osserva spesso in natura l'impiego di una codifica che appare basata sull'andamento della densità temporale di impulsi e la cui dinamica può quindi essere approssimata abbastanza fedelmente facendo ricorso a una grandezza variabile con continuità, quale la risposta delle unità neuronali dinamiche non impulsive.

Il problema della reale natura della codifica neurale negli organismi viventi rimane in ogni caso tuttora aperto e molti sforzi sono attualmente dedicati al suo chiarimento [Singer e Gray 1995; Rieke *et al.* 1997; Koch 1999]. Ciò che appare fin d'ora probabile è il ricorso dei sistemi naturali a molteplici tecniche di codifica dell'informazione, ciascuna ottimizzata per circostanze e vincoli fisici specifici.

4.1. Modelli di neurone impulsivo

Per definire un modello neurale che esibisca le caratteristiche appena descritte, possiamo prendere lo spunto dal modello elementare di neurone dinamico illustrato in figura 5.6[7]. Per ottenere un neurone impulsivo dobbiamo innanzitutto prevedere un meccanismo in grado di produrre la dinamica impulsiva che desideriamo osservare in uscita. Esistono a questo proposito dei modelli molto accurati dei processi elettrochimici che danno origine al potenziale d'azione, primo tra tutti il famoso modello di Hodgkin e Huxley [1952] ricavato da osservazioni condotte sull'assone gigante del calamaro. Questi modelli sono in genere descritti da un sistema di numerose equazioni differenziali, la cui integrazione permette di ricostruire con esattezza la forma d'onda del potenziale d'azione.

Nel nostro caso una descrizione così dettagliata è fortunatamente superflua, dato che l'informazione che ci interessa considerare è legata esclusivamente alla presenza o assenza degli impulsi neurali. Per questo motivo possiamo trascurare i dettagli del processo di produzione degli impulsi e limitarci a ipotizzare l'esistenza di un elemento in grado di produrne a fronte di una adeguata stimolazione d'ingresso.

Nel caso dei neuroni naturali, l'innesco del processo che porta alla produzione del potenziale d'azione è legato al superamento di un valore di soglia da parte del potenziale di membrana. Analogamente, nel nostro modello possiamo legare la produzione degli impulsi al superamento di una soglia da parte del potenziale di attivazione. Dal punto di vista funzionale, possiamo pensare che la produzione degli impulsi sia gestita da un dispositivo (comparatore) che confronta il valore istantaneo del potenziale di attivazione con il valore di soglia ϑ_i e, in caso di superamento, attiva la produzione di un impulso in un dispositivo posto in cascata. La struttura costituita da questa coppia di dispositivi andrà quindi a sostituire nel nostro modello il blocco rappresentante la funzione di attivazione (fig. 5.11).

Il processo che porta alla generazione di un impulso in uscita è dunque il seguente: a partire da un istante iniziale – in cui possiamo immaginare che il potenziale di attivazione assuma un valore di riferimento posto al disotto del valore di soglia ϑ_i – il neurone riceve in ingresso dei segnali e li integra (con «dimenticanza») nel potenziale di attivazione $a_i(t)$, fino a che la soglia viene eventualmente superata e il neurone emette un impulso in uscita. La condizione di generazione dell'impulso è $a_i(t) > \vartheta_i$. Contrassegnamo con il simbolo $t_i^{(f)}$ (*firing time*) i singoli istanti in cui la condizione è verificata, e con $\mathcal{F}_i =$

[7] Si noti che non avrebbe senso considerare un neurone privo di dinamica poiché verrebbe a mancare il meccanismo di memorizzazione degli impulsi presentatisi agli ingressi. Il potenziale di attivazione sarebbe quindi costituito da una sequenza di impulsi, che il neurone potrebbe tutt'al più riprodurre in uscita.

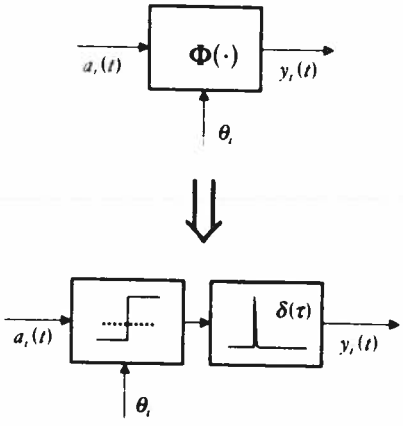


FIG. 5.11. La trasformazione del modello dinamico non impulsivo nel modello impulsivo richiede la sostituzione della funzione di attivazione con un sistema di generazione della dinamica impulsiva d'uscita. Tale sistema può essere composto da un comparatore a soglia seguito da un generatore di impulsi.

$\{t_i^{(1)}, ..., t_i^{(n)}, ...\}$ l'insieme degli istanti in cui l'i-esimo neurone emette gli impulsi, ossia[8]

$$\mathcal{F}_i = \{t \mid a_i(t) > \vartheta_i\}$$

Perché il modello di neurone dotato di questa logica di produzione degli impulsi possa funzionare con continuità, è necessario prevedere la presenza di un ulteriore elemento, il quale, dopo ogni impulso, assicuri il ritorno del potenziale di attivazione al disotto della soglia, in modo che il processo appena descritto possa ricominciare. Ciò corrisponde alla fase di iperpolarizzazione della membrana che nei neuroni naturali fa seguito al passaggio di un potenziale d'azione. Nel nostro modello questo comportamento può essere ottenuto collegando l'uscita del neurone al nodo sommatore attraverso un coefficiente negativo $-r_i$, in cui il simbolo vuole ricordare la fase «refrattaria» che, nei neuroni naturali, segue la presenza di un potenziale d'azione e rende per un certo periodo più difficoltoso (refrattarietà relativa) – o addirittura impossibile (refrattarietà assoluta) – l'innesco di ulteriori potenziali d'azione. Il modello appena descritto prende il nome di modello «integrate and fire». La sua struttura è rappresentata nella figura 5.12.

Come nel caso del modello non impulsivo possiamo trasformare facilmente questo modello in un modello dotato di dinamica all'ingresso (fig. 5.13). Si noti che con lo spostamento dell'elemento generatore della dinamica a monte del nodo sommatore, anche il processo di ripristino del potenziale d'azione conseguente alla produzione di un

[8] Dovremmo scrivere $\mathcal{F}_i = \{t \mid a_i(t) > \vartheta_i, a_i(t^-) \leq \vartheta_i\}$ per precisare che il superamento deve avvenire «dal basso». Tuttavia, i modelli che presenteremo assicurano automaticamente il rispetto di questa condizione.

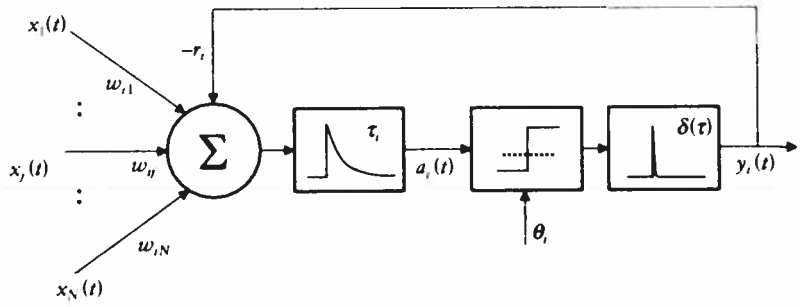


FIG. 5.12. Il modello elementare «integrate and fire» di neurone impulsivo. Il collegamento retroattivo riporta il potenziale d'azione al disotto del valore di soglia dopo la produzione di un impulso, dando di nuovo inizio al processo che porta alla generazione di ulteriori impulsi. Con un valore sufficientemente elevato del parametro r_i il suo effetto corrisponde all'iperpolarizzazione della membrana che si osserva nei neuroni naturali e permette di modellarne la fase di refrattarietà.

impulso viene dotato di una sua dinamica, descritta dal nucleo $\rho_i(t)$. Questa dinamica dà origine a un contributo $a_i^{(r)}(t)$ al potenziale di attivazione, che chiameremo «potenziale di ripristino». Una opportuna scelta della funzione $\rho_i(t)$ permette la modellazione delle fasi di refrattarietà assoluta e relativa del neurone.

4.2. Lo «Spike Response Model»

Per la descrizione formale dell'evoluzione del potenziale d'azione dei modelli di neuroni impulsivi, si applicano in linea di principio le relazioni che sono state fornite in precedenza per i neuroni non impulsivi. In particolare, il contributo al potenziale di attivazione di un im-

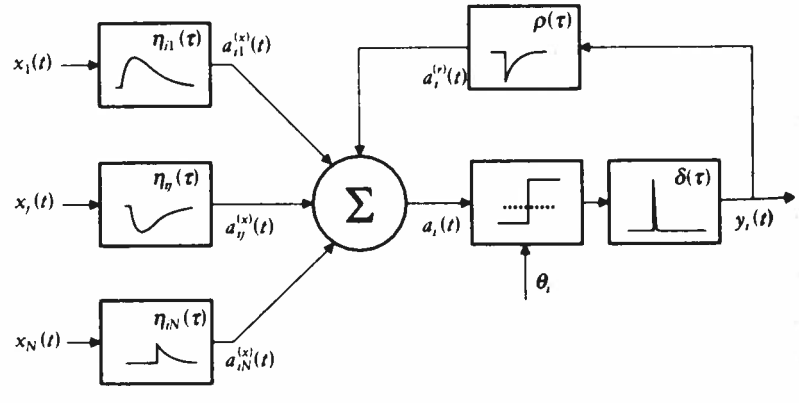


FIG. 5.13. La struttura del neurone impulsivo con dinamica all'ingresso. Il collegamento retroattivo è ora caratterizzato da una dinamica di ripristino (o di refrattarietà) indipendente, che dà origine a un «potenziale di ripristino» $a_i^{(r)}(t)$.

pulso ricevuto in ingresso si otterrà calcolando la convoluzione dell'impulso con il nucleo della risposta corrispondente. Si è visto in precedenza che, nel caso continuo, ciò corrisponde a valutare l'espressione $a_{ij}^{(x)} = \int_0^t \eta_{ij}(t - \tau)\, x_i(\tau)\, d\tau$, nella quale $x(\tau)$ rappresenta l'andamento dell'impulso.

Come si è detto, però, non vogliamo specificare la forma d'onda esatta degli impulsi e quindi non siamo a rigore in grado di valutare questo integrale. Per superare questa impasse è conveniente immaginare di avere a che fare con degli impulsi ideali istantanei noti con il nome di «delta di Dirac». Per indicare un impulso ideale presente all'istante $t = 0$ utilizzeremo il simbolo $\delta(t)$, e quindi un generico impulso presente all'istante $t = t^{(f)}$ sarà rappresentato da $\delta(t - t^{(f)})$. Gli impulsi ideali hanno la proprietà di produrre, quando ricevuti in ingresso da un neurone dinamico, la risposta relativa al nucleo corrispondente (nuclei che prendono perciò il nome di *risposte impulsive*). In termini matematici ciò corrisponde ad assumere che valga la relazione

$$\int_0^t \eta(t - \tau)\, \delta(\tau - t^{(f)})\, d\tau = \eta(t - t^{(f)})$$

Questa proprietà risulta molto utile per semplificare il calcolo del potenziale di attivazione, poiché comporta la scomparsa dell'operazione di integrazione dalla formula corrispondente così che, ad esempio, la relazione $a_{ij}^{(x)} = \int_0^t \eta_{ij}(t - \tau)\, \delta(\tau - t_j^{(f)})\, d\tau$, corrispondente a un singolo impulso ricevuto sul j-esimo ingresso, diventa

$$a_{ij}^{(x)}(t) = \eta_{ij}(t - t_j^{(f)})$$

e, più in generale, il potenziale di attivazione parziale generato da tutto l'insieme degli impulsi ricevuti sullo stesso ingresso sarà dato da

$$a_{ij}^{(x)}(t) = \sum_{t_j^{(f)} \in \mathcal{F}_j} \eta_{ij}(t - t_j^{(f)})$$

dove $\mathcal{F}_j$ è l'insieme degli istanti nei quali è presente un impulso sul j-esimo ingresso. Una relazione del tutto analoga vale per il potenziale di ripristino $a_i^{(r)}(t)$

$$a_i^{(r)}(t) = \sum_{t_i^{(f)} \in \mathcal{F}_i} \rho_i(t - t_i^{(f)})$$

Il valore del potenziale d'attivazione per un neurone impulsivo con dinamica all'ingresso sarà quindi dato da

$$a_i(t) = a_i^{(r)}(t) + \sum_{j=1}^{N} a_{ij}^{(x)}(t) = \sum_{t_i^{(f)} \in \mathcal{F}_i} \rho_i(t - t_i^{(f)}) + \sum_{j=1}^{N} \sum_{t_j^{(f)} \in \mathcal{F}_j} \eta_{ij}(t - t_j^{(f)})$$

Il modello appena presentato (ossia il modello impulsivo con dinamica all'ingresso, nell'ipotesi di impulsi ideali) è noto con il nome di «Spike Response Model» [Gerstner 1999] e si presta particolarmente bene all'implementazione su calcolatore. Nel caso si vogliano mettere in evidenza i pesi sinaptici, l'espressione dei potenziali di attivazione parziali è

$$a_{ij}^{(x)}(t) = w_{ij} \sum_{t_j^{(f)} \in \mathcal{F}_j} \varepsilon_{ij}(t - t_j^{(f)})$$

che, nell'ipotesi di nuclei di ingresso uguali per tutte le connessioni sinaptiche, diventa

$$a_{ij}^{(x)}(t) = w_{ij} \sum_{t_j^{(f)} \in \mathcal{F}_j} \varepsilon(t - t_j^{(f)})$$

Se anche la dinamica di ripristino condivide il nucleo che caratterizza gli ingressi, si ha

$$a_i^{(r)}(t) = -r_i \sum_{t_i^{(f)} \in \mathcal{F}_i} \varepsilon(t - t_i^{(f)})$$

Nel qual caso è possibile spostare il blocco che rappresenta l'azione del nucleo comune dopo il nodo sommatore, e si ottiene il modello privo di dinamica all'ingresso (fig. 5.12).

Solitamente le funzioni $\varepsilon(t)$ utilizzate come nuclei assumono esclusivamente valori non negativi. Dato che anche gli impulsi prodotti dai neuroni si suppongono assumere valori non negativi, sarà il segno del peso sinaptico a stabilire se un ingresso fornisce al potenziale di attivazione un contributo positivo o negativo. Poiché incrementando il valore del potenziale di attivazione si favorisce la produzione di impulsi in uscita, definiamo *eccitatorie* le sinapsi caratterizzate da un coefficiente w_{ij} positivo e *inibitorie* quelle caratterizzate da un valore negativo.

Una semplificazione possibile per lo Spike Response Model consiste nel supporre che gli impulsi generati da un neurone siano separati da un intervallo sufficientemente grande da rendere solo l'impulso più recente efficace dal punto di vista del ripristino del potenziale d'azione. Ciò corrisponde a porre

$$a_i^{(r)}(t) = \rho_i(t - \hat{t}_i)$$

dove $\hat{t}_i$ è l'istante più recente in cui l'i-esimo neurone ha prodotto un impulso. Infine, possiamo tener conto della presenza di ingressi non impulsivi $s_i(t)$ (ad esempio un sensore che fornisca un segnale analogico) ripristinando, per quegli ingressi, il termine integrale e scrivendo quindi

$$a_{ij}^{(s)}(t) = \int_0^t \sigma_i(t - \tau)\, s_i(\tau)\, d\tau$$

dove $\sigma_i(t)$ è il nucleo che specifica la risposta dell'ingresso e $a_{ij}^{(s)}(t)$ è il potenziale di attivazione parziale corrispondente, che andrà sommato a quelli degli ingressi impulsivi e al potenziale di ripristino.

Lo Spike Response Model può evidentemente essere definito anche per il caso a tempo discreto. Basterà a questo scopo sostituire la variabile continua t con la variabile discreta n, e definire $\mathcal{F}_i = \{n_i^{(1)}, ..., n_i^{(m)}, ...\}$ come l'insieme degli istanti in cui l'i-esimo neurone produce un impulso discreto $\delta(n)$, dove

$$\delta(n) = \begin{cases} 1 & \text{se } n = 0 \\ 0 & \text{altrimenti} \end{cases}$$

Nel caso in cui si utilizzi un singolo nucleo esponenziale, l'implementazione del modello può basarsi sulla solita equazione alle differenze che produce la dinamica libera esponenziale del potenziale d'azione, come descritto dall'algoritmo seguente:

Algoritmo: Evoluzione di una rete neurale «Spike Response Model» con nucleo esponenziale

1. Assegnare il valore iniziale del potenziale di attivazione a tutti i nodi della rete in base alla relazione $a_i \leftarrow a_i(0)$ (ad esempio ponendo $a_i(0) = 0$).
2. Ripetere i punti da 3 a 5 per il generico passo temporale n, con n che va da 0 fino al passo temporale finale desiderato.
3. Calcolare il valore della risposta di tutti i neuroni in base alla relazione

$$y_i \leftarrow \begin{cases} 1 & \text{se } a_i > \vartheta_i \\ 0 & \text{altrimenti} \end{cases}$$

4. Assegnare il valore attuale $\mathbf{x}$ agli ingressi di tutte le unità neurali, ponendo $x_i \leftarrow y_i$ per quelli collegati alle uscite di altri neuroni e $x_i \leftarrow s_i(n)$ per quelli collegati agli ingressi esterni $\mathbf{s}$ della rete.
5. Aggiornare il valore del potenziale di attivazione di tutte le unità neuronali ponendo $a_i \leftarrow \mu_i a_i + \mathbf{w}_i \cdot \mathbf{x} - r_i y_i$.

Se, pur continuando ad operare con nuclei esponenziali, si desidera dotare i neuroni di dinamiche di ingresso indipendenti, è necessario definire un insieme di potenziali di attivazione parziali. L'algoritmo va allora modificato come segue [Jahnke, Roth e Schönauer 1999]:

Algoritmo: Evoluzione di una rete «Spike Response Model» con dinamica esponenziale all'ingresso

1. Assegnare il valore iniziale dei potenziali di attivazione parziali e totale a tutti i nodi della rete, in base alle relazioni

$$a_{ij}^{(x)} \leftarrow a_{ij}^{(x)}(0),$$

$$a_i^{(r)} \leftarrow a_i^{(r)}(0),$$

$$a_i \leftarrow a_i^{(r)} + \sum_{j=1}^{N} a_{ij}^{(x)}$$

(ad esempio ponendo $a_{ij}^{(x)}(0) = a_i^{(r)}(0) = 0$).

2. Ripetere i punti da 3 a 5 per il generico passo temporale n, con n che va da 0 fino al passo temporale finale desiderato.

3. Calcolare il valore della risposta di tutti i neuroni in base alla relazione

$$y_i \leftarrow \begin{cases} 1 & \text{se } a_i > \vartheta_i \\ 0 & \text{altrimenti} \end{cases}$$

4. Assegnare il valore attuale **x** agli ingressi di tutte le unità neurali, ponendo $x_i \leftarrow y_i$ per quelli collegati alle uscite di altri neuroni e $x_i \leftarrow s_i(n)$ per quelli collegati agli ingressi esterni **s** della rete.

5. Aggiornare il valore del potenziale di attivazione di tutte le unità neuronali ponendo

$$a_{ij}^{(x)} \leftarrow \mu_{ij}\, a_{ij}^{(x)} + w_{ij} x_j$$

$$a_i^{(r)} \leftarrow \mu_i\, a_i^{(r)} - r_i y_i$$

$$a_i \leftarrow a_i^{(r)} + \sum_{j=1}^{N} a_{ij}^{(x)}.$$

5. Implementazione dei modelli dinamici

Nelle pagine precedenti abbiamo presentato dei modelli dapprima elementari e poi via via più complessi di unità neuronali dinamiche. Per i casi più semplici, ossia in presenza di risposte esponenziali e nell'ipotesi di tempo discreto, sono stati forniti gli algoritmi che permettono di calcolare l'evoluzione di una rete neurale composta di unità dinamiche. Vediamo ora come viene affrontato in generale il problema della realizzazione fisica o algoritmica di unità neurali con dinamiche generiche, sia a tempo continuo che discreto.

5.1. Modelli a tempo discreto

Per generalizzare i modelli a tempo discreto che abbiamo preso in esame fin qui, prendiamo lo spunto dall'equazione alle differenze che dà luogo al modello con risposta libera esponenziale. Riscriviamo l'equazione come segue

$$a(n+1) = \mu a(n) + \gamma c(n)$$

dove il termine $\gamma c(n)$ può rappresentare l'input netto $\mathbf{w}_i \cdot \mathbf{x}(n)$ del modello elementare con dinamica centralizzata, o il contributo $w_{ij} x_j(n)$ del singolo ingresso dotato di dinamica. In altre parole, interpretiamo la sequenza $c(n)$ come un «ingresso generalizzato», da particolarizzare di volta in volta a seconda delle circostanze. Possiamo estendere questa equazione di aggiornamento del potenziale di attivazione, includendo nell'equazione un numero finito di termini aggiuntivi che tengano conto della storia passata dell'ingresso e dell'uscita. Otteniamo la seguente espressione

$$a(n+1) = \mu_0 a(n) + \mu_1 a(n-1) + \ldots + \mu_m a(n-m) + \\ + \gamma_0 c(n) + \gamma_1 c(n-1) + \ldots + \gamma_g c(n-g)$$

che corrisponde alla struttura illustrata in figura 5.14.

Il modello rappresentato in figura 5.14 ha come caso particolare il modello elementare con risposta esponenziale. Inoltre, si può notare che l'insieme di elementi che mette a disposizione i valori passati dell'ingresso corrisponde alla linea di ritardo di una *Time-Delay Neural Network*. In effetti, scegliendo opportunamente i coefficienti γ e μ che vi compaiono, è possibile generare o approssimare con questa struttura (che è quella di un *filtro digitale lineare*) una vasta classe di dinamiche. Il problema sarà dunque rappresentato dalla determinazione dei parametri necessari per ottenere la risposta impulsiva desiderata. Non c'è spazio qui per affrontare questo argomento in tutta la sua generalità. Rimandiamo quindi il lettore all'abbondante letteratura riguardante l'elaborazione numerica dei segnali (DSP) e la sintesi dei filtri digitali (si veda ad esempio [Smith 1999]), e ci limitiamo ad osservare che, una volta determinati i coefficienti che compaiono nell'equazione generalizzata, per implementare il modello corrispondente è sufficiente sostituire tale formula a quella elementare utilizzata negli algoritmi descritti in precedenza.

5.2. Modelli a tempo continuo

Malgrado la diffusione sempre più estesa dei calcolatori digitali tenda a favorire l'impiego di modelli a tempo discreto, esistono validi motivi per occuparsi dei modelli neurali dinamici a tempo continuo.

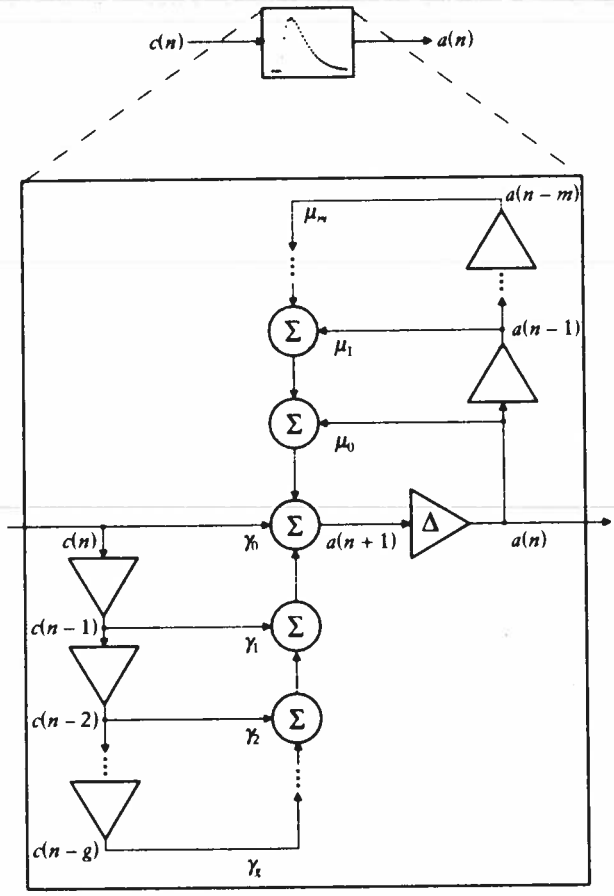


FIG. 5.14. La struttura che generalizza l'equazione di aggiornamento del potenziale di attivazione e permette l'implementazione di una vasta classe di risposte impulsive a tempo discreto.

Una prima ragione è costituita dal fatto che le osservazioni neurofisiologiche riguardanti la struttura e il funzionamento dei neuroni riguardano dei fenomeni elettrochimici per i quali è agevole fornire dei modelli continui, solitamente in termini di sistemi di equazioni differenziali[9]. Di conseguenza, questa è la forma in cui sono espressi i modelli più sofisticati derivanti dalle osservazioni neurofisiologiche.

[9] Va però detto che l'adozione di una rappresentazione continua anche nei casi in cui si intende successivamente procedere a una implementazione al calcolatore, è in parte legata a una tradizione scientifica che ha le sue origini in epoche in cui le macchine calcolatrici non erano disponibili e una formulazione discreta appariva perciò sconsigliabile. Date le mutate condizioni, l'attitudine tradizionale andrebbe in questi casi rimessa in discussione, al fine di formulare per quanto possibile fin dall'inizio i problemi in termini discreti.

A proposito del grado di sofisticazione dei modelli neuronali che è opportuno impiegare, vale la pena di osservare che probabilmente ha senso considerare modelli in grado di riprodurre il dettaglio fine della dinamica dei neuroni naturali solo nel caso in cui lo scopo finale sia la comprensione del funzionamento di questi ultimi. Infatti, possiamo aspettarci che molte delle peculiarità che si riscontrano nel comportamento dei neuroni naturali siano dettate dall'esistenza di vincoli fisici a cui questi ultimi devono sottostare, ovvero a contingenze evolutive, e vadano quindi considerate come degli ostacoli rispetto alla realizzazione di una data funzione. Ne consegue che, se l'obiettivo è la realizzazione di una data funzione mediante una rete di neuroni artificiali, sarà opportuno innanzitutto interrogarsi sulle caratteristiche che è opportuno includere nel modello relativamente allo scopo prefisso, in modo da evitare l'adozione di un modello inutilmente complesso. Ciò è del tutto analogo a quanto accade nell'ambito della modellazione dei circuiti elettronici dove, nel caso si desideri studiare solo la logica di funzionamento, si tende a non rappresentare le non-idealità del funzionamento dei componenti che lo compongono.

Un ulteriore motivo per interessarsi ai modelli a tempo continuo risiede nella possibilità di dedurne una realizzazione fisica, solitamente per mezzo di circuiti elettrici o elettronici. Ad esempio le equazioni del modello del «leaky integrator» corrispondono a quelle di un circuito elettrico in cui il potenziale di attivazione è rappresentato dalla tensione ai capi di un condensatore e in cui i pesi sinaptici e la costante di tempo dell'esponenziale sono determinati dal valore di un insieme di elementi resistivi [Haykin 1999]. Modelli più sofisticati, che comprendono la categoria dei neuroni impulsivi, possono essere realizzati mediante circuiti elettronici comprendenti elementi attivi (diodi, transistori, amplificatori e comparatori). Particolarmente promettenti appaiono gli sforzi che si stanno compiendo per la realizzazione di circuiti a larghissima scala di integrazione (VLSI) in grado di eseguire trasformazioni analoghe a quelle operate dai primi stadi di elaborazione dei segnali sensoriali negli organismi viventi (si veda, ad esempio, Mortara e Venier [1999]).

Nel caso in cui si intenda invece simulare un modello continuo mediante un circuito logico o un calcolatore digitale, è necessario procedere innanzitutto a una discretizzazione del modello stesso. Abbiamo visto che, per il modello elementare con dinamica esponenziale, esiste una corrispondenza stretta tra il caso a tempo discreto e continuo. Più in particolare, la dinamica continua rappresentata dall'esponenziale $e^{-\frac{t}{\tau_i}}$ si riduce, nel caso di discretizzazione con passo temporale Δt, all'esponenziale discreto e $e^{-\frac{n\Delta t}{\tau_i}} = \left(e^{-\frac{\Delta t}{\tau_i}}\right)^n = \mu_i^n$. Dunque, in questo caso è possibile riprodurre esattamente l'andamento della dinamica continua negli istanti considerati dal modello discreto.

Questa proprietà è purtroppo limitata a una classe piuttosto ristretta di modelli e di tipologie di segnali di ingresso esterni [Rotter e Diesmann 1999]. In generale è invece necessario ricorrere all'integrazione numerica delle equazioni differenziali che costituiscono il modello continuo. Esiste a questo proposito una vasta letteratura: si veda ad esempio Hansel, Mato, Meunier e Neltner [1998] per una metodologia ottimizzata per i modelli «integrate and fire», l'articolo di Jahnke, Roth e Schönauer [1999] per il caso dello «Spike Response Model» e, più in generale, l'ormai classico testo di Press, Teukolsky, Vetterling e Flannery [1992] per la descrizione di algoritmi applicabili a equazioni differenziali generiche.

Senza entrare nel dettaglio del problema rappresentato dall'integrazione numerica delle equazioni differenziali ordinarie, consideriamo a titolo di esempio la corrispondenza che viene a stabilirsi tra l'equazione originaria e le formule discrete, applicando il metodo di Eulero[10] all'equazione

$$\frac{da_i(t)}{dt} = -\frac{1}{\tau_i} a_i(t) + \mathbf{w}_i' \cdot \mathbf{x}(t)$$

La relazione di aggiornamento del potenziale di attivazione che ne deriva è

$$\frac{a_i(t + \Delta t) - a_i(t)}{\Delta t} = -\frac{1}{\tau_i} a_i(t) + \mathbf{w}_i' \cdot \mathbf{x}(t)$$

(si noti che stiamo ora facendo il percorso inverso a quello che ci ha portato a definire l'equazione differenziale a partire da quella discreta). Adottando la notazione a tempo discreto basata sulla corrispondenza $t = n\Delta t \rightarrow n$, possiamo riscrivere questa relazione nella forma

$$a_i(n + 1) = \left(1 - \frac{\Delta t}{\tau_i}\right) a_i(n) + \Delta t\, \mathbf{w}_i' \cdot \mathbf{x}(n)$$

che è dotata della ormai familiare struttura del tipo $a_i(n + 1) = \mu_i a_i(n) + \mathbf{w}_i \cdot \mathbf{x}(n)$ (con $\mu_i = 1 - \frac{\Delta t}{\tau_i}$ e $\mathbf{w}_i = \mathbf{w}_i' \Delta t$) che abbiamo utilizzato negli algoritmi descritti in precedenza.

Consideriamo infine il problema della determinazione del passo temporale Δt da utilizzare nel processo di evoluzione di un modello

[10] Facciamo notare che la scelta del metodo di Eulero è motivata in questo caso unicamente dalla semplicità e perspicuità delle formule a cui dà origine, viste le mediocri caratteristiche di precisione e stabilità che lo contraddistinguono, e che rendono consigliabile nella pratica il ricorso a metodi più avanzati, come ad esempio i metodi di Runge-Kutta [Press, Teukolsky, Vetterling e Flannery 1992].

continuo discretizzato, o di un modello discreto che debba interagire con segnali esterni con dinamica assegnata. Nel caso in cui il modello discreto sia ottenuto con un'integrazione numerica di un'equazione differenziale, è consigliabile far ricorso a algoritmi di integrazione in grado di adattare automaticamente il passo temporale alle caratteristiche della funzione integrata e degli obiettivi di precisione prefissati [Press, Teukolsky, Vetterling e Flannery 1992]. Negli altri casi, è invece difficile stabilire delle regole generali per la scelta di Δt, dato che essa dipenderà tra le altre cose dalla velocità di variazione dei segnali di ingresso della rete e dai tempi di reazione desiderati (che influiranno anche sulla scelta dei nuclei $\eta(t)$ dei modelli neuronali). Ovviamente la scelta di un passo temporale ridotto permette di riprodurre dinamiche molto veloci, ma impone l'esecuzione di una maggior quantità di operazioni per una stessa durata della simulazione. Nel caso si operi con modelli di neuroni impulsivi su scale temporali paragonabili a quelle che caratterizzano il sistema nervoso umano, considerando che il singolo potenziale d'azione ha una durata dell'ordine del *ms*, la scelta $\Delta t = 1$ *ms* è solitamente adeguata [Jahnke, Roth e Schönauer 1999].

5.3. Modelli dinamici e apprendimento

Rispetto al modello classico di neurone, che risulta determinato dai pesi sinaptici e dai parametri della funzione sigmoidale d'uscita, i modelli neuronali dinamici sono in genere caratterizzati da un più elevato numero di parametri. Si pensi ad esempio alla presenza delle costanti di tempo di un modello con dinamica esponenziale all'ingresso, o ai parametri τ_m, τ_s e τ_d degli esempi di nucleo $\varepsilon(t)$ presentati in precedenza. In una rete neurale basata su questi modelli, tutti questi nuovi parametri possono quindi potenzialmente essere l'oggetto di un processo di apprendimento, il che amplia considerevolmente la categoria degli algoritmi che si possono sviluppare in questo ambito.

Un possibile approccio alla selezione di questi insiemi estesi ai parametri consiste nell'utilizzo di tecniche di ottimizzazione che non richiedono la disponibilità di espressioni analitiche dell'errore in funzione dei parametri. Ad esempio, questo è il caso degli algoritmi genetici, che verranno descritti brevemente nel prossimo capitolo. Non entreremo qui nel dettaglio di questi ulteriori sviluppi e ci limiteremo a considerare l'apprendimento in termini di modifica dei pesi sinaptici. A questo proposito, per quanto riguarda l'addestramento delle *continuous-time recurrent neural networks*, facciamo notare che è possibile utilizzare per l'addestramento una delle molte varianti della tecnica di *back-propagation* nel tempo introdotta nel capitolo 2 (si veda ad esempio la rassegna presentata in Pearlmutter [1995] o gli algoritmi descritti da Haykin [1999]).

Più interessante appare il caso delle reti basate su neuroni impulsivi, per le quali gli algoritmi di addestramento sviluppati per i modelli

non impulsivi non appaiono in grado di fare un uso adeguato delle peculiarità del modello. Consideriamo ad esempio il seguente modello di apprendimento hebbiano applicabile a reti basate su modelli impulsivi, e in particolare sullo Spike Response Model [Gerstner, Kempter, van Hemmen e Wagner 1999; Kempter, Gerstner e van Hemmen 1999a, 1999b].

Nel capitolo 1 si è visto che la regola di Hebb prevede che i pesi sinaptici vengano modificati come conseguenza congiunta dello stato di attività presinaptica e postsinaptica di un neurone. Per il modello classico, privo di dinamica temporale, questo corrisponde a legare la variazione dei pesi sinaptici al livello dei segnali di ingresso e di uscita. Nel caso di neuroni impulsivi entra in gioco anche la dimensione temporale. In particolare, possiamo definire una *finestra temporale di apprendimento* descritta da una funzione $p(t^{pre} - t^{post})$ che pesa gli impulsi che si presentano in ingresso e in uscita in funzione della loro distanza temporale e ne deduce la variazione dei pesi sinaptici corrispondente. Un esempio di funzione $p(t)$, corrispondente all'espressione

$$p(t^{pre} - t^{post}) = p(t^d) = \begin{cases} (a_+ - a_-)\, e^{\frac{-(t^* - t^d)}{\tau_s}} & \text{per } t^d < t^* \\ a_+ e^{\frac{-(t^d - t^*)}{\tau_+}} - a_- e^{\frac{-(t^d - t^*)}{\tau_-}} & \text{per } t^d > t^* \end{cases}$$

è rappresentato in figura 5.15. Si noti che tale funzione pesa in maniera positiva (il che si tradurrà in un rafforzamento del peso sinaptico) nel caso di impulsi presinaptici che precedono quelli postsinaptici, con un massimo in corrispondenza di un leggero anticipo dell'impulso presinaptico su quello postsinaptico.

Indichiamo con $x_{ij}(t)$ la sequenza di impulsi presente sul generico ingresso dell'i-esimo neurone, e con $y_i(t)$ quella prodotta dal neurone stesso. Una relazione abbastanza generale per la variazione del peso

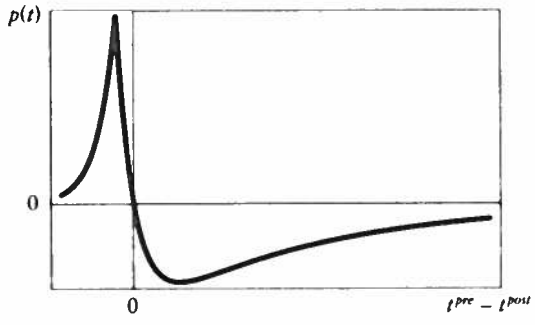


FIG. 5.15. La funzione che descrive la finestra temporale di apprendimento e determina le caratteristiche dell'apprendimento hebbiano per il modello neuronale impulsivo.

sinaptico corrispondente a un intervallo di apprendimento $[0, T]$ è [Gerstner, Kempter, van Hemmen e Wagner 1999]

$$\Delta w_{ij} = \int_0^T \int_0^T p(t' - t)\, x_j(t')\, y_i(t)\, dt\, dt' + b_{ij} \int_0^T x_j(t')\, dt' + b_i \int_0^T y_i(t)\, dt + a_i T$$

Se supponiamo che la funzione $p(\tau)$ assuma valori trascurabili al di fuori di un intervallo molto più breve di quello di apprendimento, e dividiamo questa espressione per la durata T dell'intervallo di apprendimento, possiamo interpretare la formula di aggiornamento dei pesi in termini di medie temporali $\langle \cdot \rangle$ (valutate sull'intervallo $[0, T]$), mettendo in evidenza l'effetto della correlazione tra i segnali di ingresso e di uscita

$$\frac{\Delta w_{ij}}{T} = \left\langle \frac{dw_{ij}}{dt} \right\rangle = \int_{-\infty}^{\infty} p(s)\, \langle x_j(t + s)\, y_i(t) \rangle\, ds + b_{ij} \langle x_j(t) \rangle + b_i \langle y_i(t) \rangle + a_i$$

(in cui è stata eseguita la sostituzione $s = t' - t$). Nell'ipotesi aggiuntiva in cui la densità di impulsi in ingresso e in uscita varino lentamente durante l'intervallo di apprendimento e che non vi sia correlazione tra i singoli impulsi presenti in ingresso e in uscita, possiamo trasformare questa relazione in

$$\Delta w_{ij} = \nu_{ij}^{(x)} \nu_i^{(y)} T \int_{-\infty}^{\infty} p(s)\, ds + b_{ij} T\, \nu_{ij}^{(x)} + b_i T\, \nu_i^{(y)} + T\, a_i$$

dove $\nu_{ij}^{(x)}$ e $\nu_{ij}^{(y)}$ sono le densità di impulsi di ingresso e di uscita, rispettivamente. Questo mostra che il modello dà effettivamente luogo a un apprendimento hebbiano in termini di codifica nella densità di impulsi. Una descrizione dettagliata dell'algoritmo e di un insieme di parametri numerici che rendono il modello biologicamente plausibile è fornita da Kempter, Gerstner e van Hemmen [1999a].

1. Algoritmi genetici ed evoluzione naturale

Gli algoritmi genetici[1] [Holland 1975] sono una famiglia di tecniche di ottimizzazione che si ispirano all'evoluzione naturale. I sistemi biologici esibiscono un alto grado di robustezza ed efficienza nel risolvere una vasta serie di problemi essenziali per la loro sopravvivenza, ma, al contrario dei sistemi artificiali creati dall'uomo, essi non sono il frutto di un disegno progettuale, bensì sono il risultato di un processo evolutivo basato sulla riproduzione selettiva degli individui migliori, sulla ricombinazione genetica dei loro cromosomi e su alcune mutazioni casuali. Anche se l'esatto funzionamento dei meccanismi evolutivi è ancora oggetto di discussione, vi è un generale accordo su alcuni principi di base che contraddistinguono lo sviluppo filogenetico degli organismi viventi:

– l'evoluzione naturale opera sui cromosomi degli individui invece che sugli individui stessi, ovvero su una codificazione genetica (il «genotipo») delle caratteristiche fisiche dell'organismo (il «fenotipo»);

– il processo di selezione naturale favorisce la riproduzione generazionale dei cromosomi che hanno dato luogo agli organismi più efficienti dal punto di vista adattivo: esso è quindi il momento in cui il fenotipo esercita una influenza sul genotipo, anche se questa influenza è solo indiretta perché (al contrario di quanto pensava Lamarck) non prevede la trascrizione diretta nel genotipo delle caratteristiche acquisite dal fenotipo nel corso della sua vita;

– i meccanismi biologici della riproduzione costituiscono il cuore del processo evolutivo: la combinazione dei codici genetici dei due genitori e l'introduzione di piccole mutazioni casuali dà luogo alla creazione di nuove strutture genetiche la cui bontà adattiva – e dun-

[1] Un ottimo testo introduttivo è Goldberg [1989]; si veda anche Davis [1991] e Mitchell [1996; trad. it. 1999].

que la probabilità di riproduzione – assume un significato solo in relazione alle capacità di sopravvivenza degli individui corrispondenti;

– l'evoluzione naturale opera su popolazioni di individui attraverso un processo ciclico e generazionale che non possiede memorie storiche, ma si basa esclusivamente su contingenze ambientali definite a livello dell'interazione fra ogni singolo individuo e il proprio ambiente ecologico.

Il funzionamento dell'evoluzione naturale è stato paragonato da Richard Dawkins [1986; trad. it. 1988] al lavoro di un orologiaio cieco: benché le modalità operative siano basate essenzialmente su dei processi casuali senza alcuna finalità definita a priori, i sistemi biologici possiedono un grado di complessità e di robustezza perfettamente adeguato ai compiti che essi devono svolgere per poter sopravvivere. I sistemi biologici possiedono dunque molte caratteristiche di robustezza, di auto-organizzazione, di adattamento e di efficienza che sono altamente desiderabili se catturate e incorporate nei sistemi artificiali creati dall'uomo. Tuttavia i metodi tradizionali di costruzione dei sistemi artificiali difficilmente riescono ad ottenere risultati eguagliabili alle prestazioni degli esseri viventi anche nei compiti più semplici, come ad esempio la navigazione, la ricerca di un rifugio, la fuga da un predatore, il riconoscimento di oggetti, ecc. Uno dei motivi di questo fallimento risiede probabilmente nel fatto che i metodi tradizionali si basano su procedimenti analitici (isolamento del problema, definizione teorica, identificazione delle variabili e derivazione formale di una soluzione specifica) che sono molto diversi dai principi di sviluppo dei sistemi biologici.

Gli algoritmi genetici invece si basano su procedimenti molto simili a quelli impiegati dall'evoluzione naturale e possiedono due precise finalità: il tentativo di comprendere meglio i meccanismi di sviluppo filogenetico dei sistemi viventi attraverso un processo simulativo di sintesi e il desiderio di sfruttare questi procedimenti per la creazione di sistemi artificiali che esibiscano le stesse caratteristiche di robustezza, efficienza e flessibilità degli organismi viventi nell'eseguire compiti difficilmente risolvibili con i metodi tradizionali. In quest'ottica le motivazioni che stanno dietro alla ricerca sugli algoritmi genetici sono molto simili a quelle del connessionismo rispetto ai sistemi di intelligenza artificiale tradizionale (si veda l'Introduzione).

Nei prossimi paragrafi verrà dunque presentato il funzionamento generale degli algoritmi genetici e nella seconda parte del capitolo verranno descritti alcuni metodi di ricerca che combinano gli algoritmi genetici con le tecniche connessioniste.

2. Gli elementi fondamentali di un algoritmo genetico

Gli algoritmi genetici operano su una popolazione di cromosomi artificiali che vengono fatti riprodurre selettivamente in base alle

prestazioni dei fenotipi a cui danno origine rispetto al problema da risolvere; durante il processo riproduttivo le repliche dei cromosomi degli individui migliori vengono accoppiate casualmente e parte del materiale genetico viene scambiato, mentre alcune piccole mutazioni casuali alterano localmente la struttura del codice genetico; le nuove strutture genetiche vanno quindi a rimpiazzare quelle dei loro genitori dando luogo a una nuova generazione di individui. Il processo continua fino a quando nasce un individuo che rappresenta una soluzione accettabile per il problema in esame. Gli algoritmi genetici si basano quindi su tre operatori principali: la *riproduzione selettiva* degli individui migliori, la *ricombinazione genetica* (*crossover*) e le *mutazioni casuali* dei cromosomi.

Prima di scendere nei dettagli di funzionamento di questi operatori, osserviamo che vi sono due importanti meccanismi necessari alla soluzione di un problema con un algoritmo genetico: la codificazione genetica e la funzione di valutazione. La *codificazione genetica* si riferisce al tipo di rappresentazione che viene utilizzata per codificare le soluzioni del problema nei cromosomi artificiali. Un cromosoma artificiale è una sequenza di simboli: per questo motivo viene comunemente definito con il nome di «stringa genetica». Un tipo di codice usato molto frequentemente è quello binario: in questo caso il cromosoma di ciascun individuo della popolazione è una stringa di lunghezza finita composta di simboli binari. Vi sono molti altri tipi di codificazione possibile: un cromosoma può essere composto di numeri reali, oppure dei quattro segni delle carte da poker, oppure di una serie finita di simboli appartenente a un qualsiasi alfabeto arbitrario. Il funzionamento dell'algoritmo genetico non è compromesso dal tipo di rappresentazione – purché il processo di decodificazione resti consistente durante il processo evolutivo – perché gli operatori genetici si limitano a selezionare le stringhe corrispondenti ai fenotipi migliori e a ricombinarne i vari pezzi a prescindere dal tipo di materiale su cui essi lavorano. La scelta del tipo di codificazione però è importante per il tipo di problema che si vuole risolvere: non esiste una codificazione che vada bene per tutti i problemi, né esistono delle regole generali che permettano di fare delle scelte ottimali. Facciamo due esempi concreti: se il problema consiste nel trovare il massimo di una funzione matematica, un codice binario, oppure un codice composto di numeri reali, si presta molto bene alla codificazione dei valori delle variabili; se invece il problema consiste nell'evolvere un programma di videoscrittura, un codice genetico più efficiente si baserà su una serie di simboli che corrispondono a determinate funzioni del linguaggio di programmazione, i quali non devono necessariamente riflettere le proprietà elementari dei numeri [si veda, ad esempio, Koza 1992]. Il problema della rappresentazione genetica e delle regole di decodificazione da genotipo a fenotipo è quindi molto importante per poter sfruttare al meglio le potenzialità di ricerca dell'algoritmo genetico; in questo

capitolo descriveremo vari tipi di rappresentazioni e di regole di decodificazione.

La *funzione di valutazione* serve invece per giudicare le prestazioni di ciascun fenotipo rispetto al problema che vogliamo risolvere: essa è composta di alcuni parametri che noi riteniamo importanti per la soluzione del nostro problema e fornisce un valore numerico per ciascun individuo proporzionale alla bontà della soluzione offerta. Essa svolge un ruolo analogo a quello dell'ambiente fisico per gli organismi biologici in quanto misura le prestazioni dell'individuo: per questo motivo essa viene comunemente definita «funzione di *fitness*». Per i due esempi concreti descritti sopra, le funzioni di *fitness* potrebbero essere nel primo caso la funzione matematica che deve essere massimizzata e nel secondo caso un indice relativo alla corrispondenza tra i tasti premuti sulla tastiera e le lettere che appaiono sullo schermo. In entrambi i casi la versione decodificata di ciascun cromosoma – il fenotipo – costituisce una soluzione che genera una certa prestazione – o *fitness* – la quale verrà poi impiegata dall'operatore genetico di riproduzione selettiva. Una caratteristica interessante degli algoritmi genetici è che la funzione di *fitness* può assumere vari gradi di complessità a seconda delle conoscenze disponibili: l'unico requisito è che essa permetta un ordinamento delle stringhe genetiche in base alle loro prestazioni sul problema da risolvere.

Una volta definito il tipo di rappresentazione genetica e la funzione di *fitness*, il primo passo consiste nella creazione di una popolazione iniziale di stringhe genetiche. Solitamente la popolazione iniziale è composta di stringhe casuali. Ciascuna stringa di questa generazione iniziale viene a turno decodificata e valutata in base alla funzione di *fitness*: data la diversità delle strutture genetiche, alla fine del processo di valutazione ciascuna stringa possiederà un «valore di *fitness*» diverso (fig. 6.1).

Il processo di riproduzione selettiva consiste nella creazione probabilistica di un numero di copie di ciascuna stringa proporzionale al valore di *fitness* ottenuto dal fenotipo corrispondente. Ricopiare ciascuna stringa in proporzione al proprio valore di *fitness* significa che le stringhe che hanno riportato un valore di *fitness* maggiore avranno una probabilità maggiore di produrre uno o più figli: l'operatore di riproduzione selettiva svolge dunque un ruolo simile alla legge di sopravvivenza del più forte in natura. Vi sono diversi modi di realizzare al computer la riproduzione selettiva probabilistica: il metodo più diffuso fa ricorso all'utilizzo di una ruota della fortuna truccata[2]. Consideriamo la situazione comune in cui manteniamo una popolazione di dimensione costante ad ogni generazione e in cui rimpiazziamo completamente tutti i membri della popolazione ad ogni ricambio generazionale. La

[2] Questo metodo viene impiegato anche per la realizzazione del selettore stocastico di azioni negli algoritmi di rinforzo descritti nel secondo capitolo.

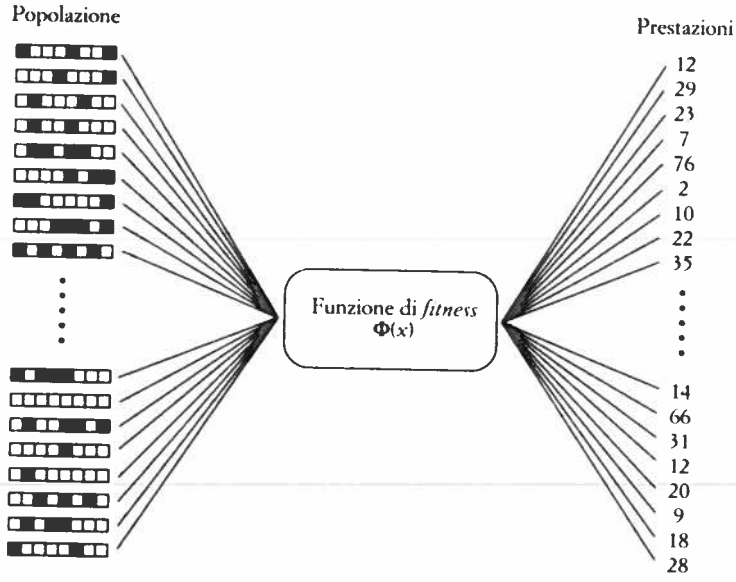


FIG. 6.1. Valutazione delle prestazioni degli individui di una popolazione. Ciascuna stringa viene a turno decodificata e valutata dalla funzione di *fitness* la quale assegna un valore di *fitness*. La diversità delle strutture genetiche assicura una diversità delle prestazioni dei membri della popolazione.

ruota della fortuna avrà dunque tante caselle quanti sono gli individui della popolazione, ma la dimensione di ciascuna casella sarà proporzionale ai valori di *fitness* di ciascun individuo. La riproduzione selettiva consiste nel far girare la ruota tante volte quanti sono gli individui da generare e nel creare ogni volta una copia della stringa corrispondente alla casella in cui si trova alla fine l'indice della ruota (fig. 6.2). Formalmente, il processo di valutazione e riproduzione selettiva avviene assegnando al fenotipo x_i di ciascuna stringa una valore di *fitness* f_i

$$f_i = \Phi(x_i)$$

dove $\Phi(x_i)$ è la funzione di valutazione. Questo valore viene utilizzato per calcolare la probabilità P_i della stringa di essere selezionata per la riproduzione (che corrisponde all'ampiezza della casella della ruota della fortuna truccata)

$$P_i = \frac{f_i}{\sum_{i}^{N} f_i}$$

pesando il valore di *fitness* della stringa per la somma dei valori di *fitness* di tutte le stringhe della popolazione. Il valore atteso del numero

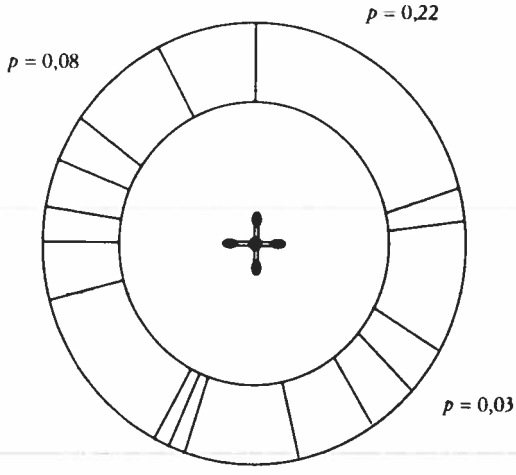


FIG. 6.2. Il processo di riproduzione selettiva realizzato tramite una ruota della fortuna truccata. L'ampiezza di ciascuna casella è proporzionale al valore di *fitness* della stringa genetica corrispondente: la generazione di ciascun individuo della nuova popolazione avviene facendo girare la ruota e creando una copia della stringa corrispondente alla casella in cui si trova alla fine l'indice della ruota. Stringhe con valori di *fitness* maggiori avranno una probabilità maggiore di lasciare una o più copie di se stesse nella generazione seguente.

di figli per ciascuna stringa è dato da NP_i, ma il numero effettivo di figli si ottiene facendo ricorso a un generatore di numeri casuali.

Le nuove stringhe così create vengono poi disposte a coppie in modo casuale e sottoposte all'azione dell'operatore di ricombinazione genetica (*crossover*). Ciascuna coppia viene incrociata con una determinata probabilità p_c. Per ciascuna delle coppie selezionate viene scelto un punto d'incrocio casuale attorno al quale avviene uno scambio reciproco di materiale genetico (fig. 6.3). Infine, tutte le stringhe della popolazione subiscono un processo di mutazione in base al quale ciascun elemento del cromosoma cambia il proprio valore in base a una probabilità p_m. Vi sono diversi tipi di operatori di mutazione a seconda del tipo di rappresentazione genetica impiegata: ad esempio, nel caso del codice binario gli elementi selezionati per essere mutati modificano il loro stato (da zero a uno o viceversa); nel caso di un codice composto di numeri reali, invece, gli elementi selezionati assumono un nuovo valore estratto in modo casuale da un certo intervallo.

La nuova popolazione di stringhe genetiche rimpiazza parzialmente o completamente le vecchie stringhe e il processo di decodifica, valutazione, riproduzione selettiva, incrocio e mutazione riprende ciclicamente per parecchie generazioni fino a quando viene ottenuta una stringa che codifica una soluzione soddisfacente.

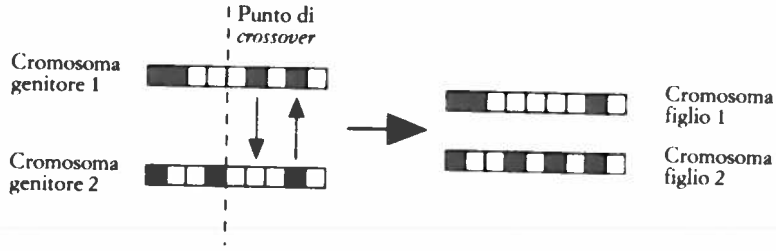


FIG. 6.3. Ricombinazione genetica di due stringhe (*crossover*). Sulla sinistra sono raffigurate due stringhe copie dei genitori della generazione precedente accoppiate in modo casuale. Il punto di incrocio e scambio genetico viene selezionato in modo casuale.

2.1. Un esempio concreto

In questo paragrafo osserviamo il funzionamento di un algoritmo genetico in una situazione molto semplice (l'esempio è tratto da Goldberg [1989]). Il problema consiste nel trovare i valori della variabile x che massimizzano la funzione potenza $\Phi(x) = x^2$, dove x può variare tra 0 e 31. Il primo passo consiste nella scelta della rappresentazione genetica: utilizziamo un codice binario e stringhe di lunghezza 5 che ci permettono di codificare i numeri da 0 (00000) a 31 (11111)[3]. Creiamo dunque una piccola popolazione composta solamente di quattro stringhe generate casualmente (tab. 6.1a). Ciascuna stringa viene decodificata e il suo valore di *fitness* viene calcolato applicando la funzione Φ al numero decimale ottenuto. Alla fine del processo di valutazione ciascuna stringa riceve una probabilità di riproduzione proporzionale alla propria *fitness* e il generatore di numeri casuali assegna un numero di copie a ciascuna di esse in base a questa probabilità. Osserviamo nel frattempo due importanti indici evolutivi: il valore medio e il valore massimo della *fitness*. Siccome una popolazione è sempre – anche nelle ultime generazioni – composta di individui che ottengono prestazioni diverse, il valore medio è sempre più basso del valore massimo.

Le nuove stringhe vengono poi accoppiate e per ciascuna di esse (qui la probabilità di *crossover* $p_c = 1$) viene scelto un punto d'incrocio casuale attorno al quale esse si scambiano parte del loro materiale. Infine ciascun elemento viene mutato con probabilità $p_m = 0{,}001$: in

[3] Il processo di trasformazione da numero binario a numero decimale è dato da $x = \sum_{p=0}^{N-1} i\,2^p$ dove p indica la *posizione* del numero intero binario a partire dall'ultima cifra (la numero zero), N è il numero di cifre e i è il valore binario dell'intero corrispondente (zero o 1). Ad esempio $10100 = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 16 + 4 = 20$.

TAB. 6.1. *Schema di funzionamento di un semplice algoritmo genetico*

A stringa numero	popola- zione iniziale	decodifi- cazione x	valutazione $F(x) = x^2$	probabilità ripr. $P_i = \frac{f_i}{\sum_i f_i}$	numero di figli	numero effettivo con ruota fortuna
1	01101	13	169	0,14	0,58	1
2	11000	24	576	0,49	1,97	2
3	01000	8	64	0,06	0,22	0
4	10011	19	361	0,31	1,23	1
somma			1170	1,00	4,00	4
media			293	0,25	1,00	1
massimo			576	0,49	1,97	2

B accoppiamento con punto di *crossover* (!)	incrocio $p_c = 1,0$ (tutti)	mutazioni $p_m = 0,001$ (nessuna)	nuova popolazione	decodifi- cazione x	valutazione $\Phi(x) = x^2$
0110!1	01100	01100	01100	12	144
1100!0	11001	11001	11001	25	625
11!000	11011	11011	11011	27	729
10!011	10000	10000	10000	16	256
somma					1754
media					439
massimo					729

questo caso nessun elemento viene mutato. La nuova popolazione di cromosomi così ottenuta viene nuovamente decodificata e valutata: questa volta sia il valore medio che il valore massimo di *fitness* sono superiori al valore della generazione iniziale.

Se osserviamo il risultato dell'operatore di *crossover*, notiamo che lo scambio casuale non produce necessariamente stringhe ottimali: in questo caso per entrambe le coppie viene generata una stringa con prestazioni uguali o addirittura peggiori della media della generazione precedente (*fitness* 246 e 144), ma vengono generate anche stringhe molto migliori (*fitness* 625 e 729). Il processo di riproduzione selettiva però fa sì che mediamente solo le stringhe migliori contribuiscano alla creazione della popolazione successiva.

2.2. La Teoria degli schemi

John Holland, considerato l'inventore degli algoritmi genetici, ha sviluppato una teoria – la Teoria degli schemi – che permette di analizzare il funzionamento del processo evolutivo. Uno «schema» è una stringa; utilizzando un simbolo aggiuntivo * rappresenta un insieme di

stringhe genetiche; il simbolo aggiuntivo indica la presenza di un qualsiasi simbolo appartenente all'alfabeto impiegato per la codificazione genetica. Ad esempio lo schema 1111* rappresenta la stringa 11111 e 11110; lo schema 1***0 rappresenta invece le otto stringhe che possiedono un 1 iniziale e uno 0 finale. Gli schemi sono uno strumento utile per analizzare le informazioni contenute in una popolazione di stringhe e osservarne le somiglianze. Se consideriamo le due popolazioni dell'esempio riportato nel paragrafo precedente notiamo che le stringhe migliori appartengono allo schema 11*** (la stringa 11000 della prima generazione e le stringhe 11001 e 11011 della seconda generazione): la ricerca degli schemi ottimali ci permette di individuare efficientemente l'informazione importante presente nel codice genetico della popolazione di individui.

Gli schemi sono anche uno strumento molto efficace per esplorare lo spazio delle possibili soluzioni del problema che vogliamo risolvere. Una stringa binaria composta di l elementi appartiene a 2^l schemi diversi (siccome ciascun elemento può assumere due diversi stati): questo significa che una popolazione di N stringhe contiene un numero di schemi compreso tra 2^l e $N2^l$, a seconda della diversità della popolazione. Quindi, da un lato gli schemi permettono di isolare i componenti importanti per la soluzione del problema e dall'altro lato permettono di esplorare un numero di potenziali soluzioni molto più grande del numero di stringhe della popolazione. Il processo di riproduzione selettiva fa sì che nel corso delle generazioni venga allocato un numero sempre maggiore di esemplari di schemi ad alta *fitness*: nell'esempio del paragrafo precedente notiamo infatti che lo schema 11*** ha duplicato la sua presenza alla seconda generazione. Tuttavia non tutti gli schemi possibili vengono elaborati e trasmessi con egual efficienza dall'algoritmo genetico: a parità di *fitness*, gli schemi caratterizzati da elementi significativi molto distanti tra di loro (ad esempio 1***1) possiedono una maggior probabilità di essere spezzati dall'operatore di *crossover*, mentre gli schemi caratterizzati da un piccolo numero di elementi significativi raggruppati in posizioni contigue (ad esempio *11***) tendono ad essere trasmessi e riprodotti con maggior frequenza. La Teoria degli schemi dimostra per l'appunto che schemi caratterizzati da una «lunghezza significativa» corta e da un alto valore di *fitness* vengono propagati attraverso le generazioni attraverso la creazione di un numero di esemplari crescente in modo esponenziale sulla base della semplice osservazione delle stringhe migliori. Un ulteriore risultato della teoria [Goldberg 1989] dimostra che gli algoritmi genetici vagliano ad ogni generazione l'equivalente di almeno N^3 schemi diversi, benché vengano valutate effettivamente solamente N stringhe: questa caratteristica viene definita con il nome di «parallelismo implicito» degli algoritmi genetici.

La dimostrazione fornita dalla Teoria degli schemi è importante perché indica che gli algoritmi genetici possono in alcuni casi esplorare in modo quasi ottimale lo spazio delle possibili soluzioni di un pro-

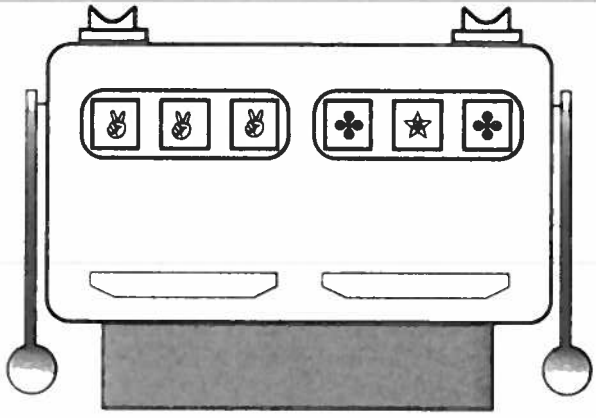


FIG. 6.4. Una slot-machine a due leve: con quale leva conviene giocare?

blema. L'esempio usato tradizionalmente per spiegare questa conclusione fa riferimento a una slot-machine con due leve (fig. 6.4). Se dovessimo giocare con questa macchinetta, vorremmo ovviamente usare la leva che ci darà un guadagno medio maggiore; però, siccome non sappiamo in partenza quale sia la leva migliore, ci troviamo di fronte al problema di dover fare due cose nello stesso tempo: raccogliere continuamente informazioni sulla leva migliore e nel contempo scegliere di volta in volta la leva con la quale giocare. Holland [1975] ha dimostrato dal punto di vista statistico che la strategia ottimale (definita come quella che riduce al minimo il rischio di perdita) consiste nell'aumentare in modo più che esponenziale il numero di giocate per la leva che produce il miglior guadagno. Benché questo risultato non possa essere tradotto direttamente in una strategia di gioco (perché non è possibile sapere in partenza quale sarà la leva che darà il miglior guadagno), esso sembra indicare che gli algoritmi genetici, con la loro crescita esponenziale del numero di individui corrispondenti agli schemi migliori, sono in grado di esplorare lo spazio delle soluzioni in modo quasi ottimale.

L'elaborazione di schemi di lunghezza significativa ridotta e ad alta *fitness* riduce notevolmente la complessità della ricerca di una soluzione: invece di cercare di costruire stringhe che possiedano un'alta *fitness* provando tutte le possibili combinazioni degli elementi disponibili, gli algoritmi genetici procedono gradualmente combinando le migliori soluzioni parziali degli individui precedenti. Gli schemi rappresentano quindi i mattoni elementari usati dall'evoluzione per la costruzione di complesse strutture. Affinché la procedura di isolamento e combinazione degli schemi risulti in un graduale miglioramento delle soluzioni offerte è importante che la rappresentazione genetica sia appropriata al problema da risolvere, perché rappresentazioni inade-

guate aumentano la complessità e lunghezza significativa degli schemi necessari a codificare una soluzione adeguata. Vi sono due regole di base per scegliere una rappresentazione adeguata: la rappresentazione dovrebbe permettere la presenza di schemi rilevanti e corti e dovrebbe basarsi su un alfabeto ristretto che permetta un'espressione naturale del problema.

Algoritmo

L'algoritmo qui descritto fa uso di un codice binario e della versione di base dei tre operatori genetici descritti nel paragrafo precedente. Ogni cromosoma può codificare parecchi parametri assegnando a ciascuno di essi una porzione della stringa genetica. Per ogni generazione registrare la *fitness* media e la *fitness* massima.

1. Definire i seguenti parametri:
   - lunghezza di ciascun cromosoma l;
   - numero di cromosomi nella popolazione N (almeno 50);
   - probabilità di *crossover* p_c;
   - probabilità di mutazione p_m.
2. Creare la popolazione iniziale di cromosomi in modo casuale.
3. Decodificare e assegnare un valore di *fitness* a ciascun cromosoma (per poter applicare la versione elementare di riproduzione selettiva descritta in seguito la funzione di *fitness* deve essere costruita in modo da restituire solo valori uguali o maggiori di zero).
4. Applicare l'operatore di riproduzione selettiva con il metodo della ruota della fortuna truccata (passi 5-8).
5. Sommare tutti i valori di *fitness* della popolazione.
6. Per ciascun cromosoma da creare.
7. Generare un numero casuale r compreso tra 0 e il valore di *fitness* totale della popolazione.
8. A partire dal primo membro della popolazione, selezionare il primo cromosoma la cui *fitness*, sommata ai valori di *fitness* dei membri precedenti della popolazione, possiede un valore maggiore o uguale a r.
9. Organizzare casualmente tutti i nuovi cromosomi in coppie.
10. Applicare a ciascuna coppia con probabilità p_c l'operatore di *crossover* (passi 11-12).
11. Scegliere il punto d'incrocio estraendo un numero casuale c tra 0 e l.
12. Scambiare fra i due cromosomi la parte della stringa compresa tra c e l.
13. Per ciascun elemento di tutti i nuovi cromosomi, invertirne il valore con probabilità p_m.
14. Tornare al passo 3 e continuare fino a quando la *fitness* media e la *fitness* massima si stabilizzano oppure fino a quando viene raggiunta una soluzione al problema.

3. Gli algoritmi genetici in maggior dettaglio

La versione di base degli operatori genetici descritta nei paragrafi precedenti è sufficiente per produrre buoni risultati nella maggior parte dei casi. Tuttavia decenni di teoria e sperimentazione nel campo dell'evoluzione artificiale hanno prodotto molteplici variazioni sui diversi componenti degli algoritmi genetici e hanno aperto interessanti dibattiti[4]. In questo paragrafo prenderemo brevemente in esame solamente alcuni aspetti principali.

Un punto molto importante è che gli algoritmi genetici operano su elementi casuali e utilizzano processi di transizione probabilistici: è quindi buona norma ripetere ogni esperimento più volte iniziando da popolazioni casuali diverse per assicurarsi la stabilità della soluzione raggiunta. Quanto minore è la dimensione della popolazione tanto maggiore è il rischio di non riuscire a replicare i dati oppure di non riuscire a ottenere una soluzione soddisfacente.

Il corretto funzionamento degli algoritmi genetici si basa sulla diversità dei cromosomi della popolazione. La mancanza di diversità corrisponde a una stagnazione che non è desiderabile né durante il processo evolutivo né quando le *fitness* media e massima della popolazione si sono stabilizzate: anche in questo ultimo caso potrebbero sempre verificarsi delle situazioni per cui potrebbe tornar utile continuare il processo evolutivo al fine di individuare nuove soluzioni. Inizialmente la diversità è assicurata dalla generazione casuale delle stringhe. Nel corso delle generazioni il processo di riproduzione selettiva tende a ridurre la diversità della popolazione e, anche se il processo di selezione fosse completamente casuale, dopo qualche generazione tutti i cromosomi si assomiglierebbero (fenomeno della «deriva genetica»). L'operatore *crossover* tende a contrastare la riduzione della diversità creando nuove strutture dalla combinazione di parti delle stringhe esistenti, ma esso non è sufficiente in tutti i casi. Immaginiamo ad esempio che un determinato «allele» (il valore specifico di una determinata caratteristica genetica, ad esempio il valore 1 in terza posizione nelle stringhe di cinque elementi considerate nel paragrafo 2.1) scompaia dalla popolazione perché nessuno degli individui che lo possedevano è stato in grado di riprodursi: in questo caso il meccanismo di *crossover* non è in grado di ricreare l'allele mancante, ma potrebbe riuscirci l'operatore di mutazione. Nell'interpretazione tradizionale degli algoritmi genetici le mutazioni sono essenzialmente un modo per mantenere un certo grado di diversità nella popolazione. Tuttavia, siccome l'operatore di mutazione opera ciecamente senza rispettare l'ingegnoso processo di elaborazione degli schemi, la maggior parte dei ricercatori tende a utiliz-

[4] Il lettore interessato può approfondire lo studio consultando gli estratti dei convegni internazionali sugli algoritmi genetici (ad esempio i *Proceedings of the International Conference on Genetic Algorithms* pubblicati da Morgan Kaufmann) e la rivista «Evolutionary Computation» pubblicata da MIT Press.

zare una probabilità di mutazione molto bassa (valori inferiori all'1%). Un altro metodo diffuso per mantenere la diversità consiste nell'introdurre ad ogni generazione nuove stringhe casuali nella popolazione sostituendole a quelle più scadenti.

Sono stati proposti diversi tipi di *crossover*: quello descritto nei paragrafi precedenti viene anche definito «*crossover* a taglio singolo» (*single-point crossover*) perché viene selezionato un unico punto casuale attorno al quale viene scambiato parte del materiale genetico. Esistono però anche diverse versioni di «*crossover* a taglio multiplo» (*multi-point crossover*) dove vengono selezionati diversi punti casuali di scambio genetico. Questa soluzione permette un maggior numero di gradi di libertà nella ricombinazione di soluzioni parziali in strutture di complessità maggiore e potrebbe accelerare il processo evolutivo. Essa diventa interessante anche quando la stringa genetica di un individuo codifica diversi parametri non strettamente collegati fra di loro, come ad esempio la lunghezza degli arti e il numero di moduli corticali di un organismo: in questi casi la mescolanza dei geni potrebbe risultare in un'evoluzione meno efficiente o più lunga per cui conviene piuttosto effettuare tagli e incroci separati all'interno delle rispettive aree genetiche. Il *crossover* svolge un ruolo molto importante nella Teoria degli schemi e ha tradizionalmente occupato una posizione centrale tra i teorici e «praticanti» degli algoritmi genetici. Recentemente però questa tradizione è stata messa in discussione. Notiamo innanzitutto che la Teoria degli schemi è valida solamente in un contesto limitato in quanto assume un alfabeto genetico discreto e finito: questo non significa che gli algoritmi genetici falliscano in tutti gli altri contesti (al contrario!), ma in questi casi il processo di elaborazione degli schemi non è più dimostrabile. In secondo luogo, esiste un vasto corpus di esperimenti recenti su codici genetici composti di numeri reali in cui non viene utilizzato il *crossover*, ma solamente la riproduzione selettiva e una probabilità di mutazione maggiore. Infine, come vedremo più sotto, esiste una versione «europea» di evoluzione artificiale che si rivela altrettanto efficace, ma che non incorpora l'operatore di *crossover*.

L'operatore di riproduzione selettiva svolge il ruolo di guida del processo evolutivo: esso assegna ad ogni stringa genetica un numero di copie grosso modo proporzionale alla sua *fitness*. Il meccanismo di riproduzione interagisce strettamente con la funzione di *fitness*. Talvolta la funzione di *fitness* possiede delle discontinuità tali per cui il migliore individuo della popolazione ottiene delle prestazioni altissime, mentre tutti gli altri membri ottengono delle prestazioni molto basse. In questo caso la generazione successiva tenderà ad essere dominata da copie genetiche del miglior individuo, le quali non avranno la possibilità di combinarsi con stringhe diverse: il processo evolutivo finirà presto in uno stato di convergenza prematura. In altri casi potrebbe accadere la situazione opposta: tutte le stringhe riportano valori di *fitness* molto simili per cui quasi tutti gli individui ottengono un numero

simile di copie. Anche in questo caso il processo evolutivo tenderà a stagnare diventando simile a un processo di ricerca casuale. Nei casi limite l'individuo migliore potrebbe addirittura fallire nel riprodursi e qualsiasi soluzione, anche raggiunta in modo più o meno casuale, sarà persa nel ricambio generazionale. Esistono alcuni metodi – le *tecniche di fitness* – per far fronte a questi problemi [Baker 1987; Goldberg e Deb 1991]. Uno di essi è il metodo della finestra (*windowing*) [Grefenstette 1986]. Questa tecnica consiste nel trovare il valore di *fitness* minimo della popolazione e assegnare a ciascuna stringa un nuovo valore di *fitness* uguale alla differenza tra la *fitness* della stringa e il valore minimo. Un'altra tecnica consiste nell'ordinare tutte le stringhe rispetto alla loro *fitness* e nel far riprodurre solamente le migliori (ad esempio il 20%) assegnando a ciascuna di esse un identico numero di copie [Floreano, Miglino e Parisi 1991; Parisi, Cecconi e Nolfi 1990]. La tecnica più diffusa consiste in una normalizzazione lineare dei valori di *fitness* della popolazione: questa tecnica permette di ridistribuire in un intervallo pressoché costante i valori di *fitness* della popolazione, allargandone la distanza se essi risultano troppo vicini o compattandoli se le stringhe migliori ottengono prestazioni notevolmente superiori alla media.

4. Diversi algoritmi genetici

Esistono molte varianti dell'algoritmo genetico descritto precedentemente. Una di esse prevede che la migliore stringa di ciascuna generazione venga ricopiata immutata nella generazione successiva. Questa tecnica, definita «strategia elitista», svolge due importanti funzioni: assicura che il cromosoma migliore sia presente nella generazione successiva anche se il processo di riproduzione probabilistica fallisce nel riprodurlo e preserva la struttura genetica nel caso in cui l'operatore di *crossover* distrugga lo schema ottimale sottostante. Siccome solitamente la strategia elitista produce un miglioramento del processo evolutivo, essa viene impiegata nella maggior parte degli algoritmi genetici assieme a un processo di normalizzazione della *fitness* per evitare che l'individuo migliore domini la popolazione.

Normalmente l'operatore di riproduzione selettiva rimpiazza tutti i membri della vecchia popolazione con le nuove stringhe. Questa scelta comporta due potenziali problemi: anche con l'impiego di una strategia elitista potrebbe succedere che molti degli individui migliori non riescano a riprodursi a causa del generatore probabilistico di copie; inoltre potrebbe succedere che gli operatori di *crossover* e di mutazione alterino la struttura genetica dei cromosomi in modo da distruggere quanto di positivo era già stato raggiunto. Una soluzione a questo problema consiste nel modificare l'operatore di riproduzione in modo da rimpiazzare solamente pochi individui alla volta: questa tecnica viene definita con il nome di «riproduzione a regime stazio-

nario» (*steady-state reproduction*) [Syswerda 1989; Whitley 1988]. Questa tecnica richiede l'impiego di un parametro aggiuntivo, ovvero il numero n di cromosomi da rimpiazzare, ma in pratica quasi tutti i ricercatori sostituiscono uno o due individui alla volta. Il processo di creazione dei nuovi individui è identico a quello della ruota della fortuna, ma in questo caso vengono creati solo le prime n copie che vengono incrociate, mutate e sostituite agli n individui peggiori della popolazione. Una versione più efficiente della riproduzione a regime costante consiste nel gettar via le stringhe che sono uguali a quelle già presenti nella popolazione (riproduzione a regime stazionario senza duplicazione): questa soluzione assicura costantemente la diversità della popolazione allontanando così il pericolo di una convergenza prematura.

Come è stato accennato all'inizio di questo capitolo, si è soliti attribuire l'invenzione degli algoritmi genetici all'americano John Holland; in realtà essi erano già stati inventati due anni prima dal tedesco Rechenberg [1973]. Gli algoritmi evolutivi tedeschi, chiamati con il nome di «strategie evolutive» (*evolutionstrategie*), sono leggermente diversi dagli algoritmi genetici in quanto sono basati soprattutto sull'operatore di mutazione e operano su stringhe di numeri reali. Successivamente Schwefel [1981], apparentemente all'oscuro di quanto stava avvenendo negli Stati Uniti, sviluppò ulteriormente le strategie evolutive aggiungendovi un operatore di *crossover* (che comunque continuava a svolgere un ruolo secondario) e lasciando evolvere alcuni parametri di mutazione assieme agli individui della popolazione. In un confronto tra gli algoritmi genetici e le strategie evolutive su una serie di problemi standard, queste ultime sono state in grado di convergere verso una soluzione ottimale in un numero più grande di casi, probabilmente grazie all'adattamento autonomo dei parametri di mutazione durante il processo evolutivo [Hoffmeister e Bäck 1991].

5. Vantaggi offerti dagli algoritmi genetici

Come è stato evidenziato all'inizio del capitolo, gli algoritmi genetici possono essere considerati una tecnica di ricerca dei massimi di funzioni. È quindi lecito chiedersi quali vantaggi essi possano offrire rispetto agli altri metodi di ricerca dell'ottimo di una funzione. Goldberg [1989] ha messo a confronto gli algoritmi genetici con i tre metodi principali: metodi di ottimizzazione basati sul calcolo infinitesimale, metodi enumerativi e metodi di ricerca casuale.

I metodi basati sul calcolo infinitesimale si dividono a loro volta in due classi: i metodi di ricerca diretta e i metodi di ricerca indiretta. I metodi di ricerca diretta si basano sulla risoluzione analitica di un sistema di equazioni – solitamente non-lineari – ottenuto eguagliando il gradiente della funzione a zero. I metodi di ricerca indiretta consistono invece nel partire da un punto qualsiasi dello spazio dei parametri

della funzione spostandosi nella direzione del gradiente della funzione (ad esempio l'algoritmo di Back-propagation descritto nel secondo capitolo è un metodo di ricerca indiretta basato sul calcolo delle derivate della funzione). Entrambi i metodi spesso non sono applicabili nella soluzione di problemi tipici del mondo reale i quali raramente rispettano l'assunzione matematica di continuità della funzione necessaria per il calcolo del gradiente. Inoltre essi operano partendo da un singolo punto nello spazio e finiscono coll'individuare il massimo più vicino (locale) della funzione. Il risultato della ricerca può essere scadente quando la funzione da ottimizzare possiede diversi massimi locali[5], come è il caso per la maggior parte dei problemi del mondo reale (fig. 6.5): una volta che il metodo ha raggiunto un picco locale, è ben difficile scenderne per raggiungerne uno più alto.

I metodi di ricerca enumerativi consistono nell'esame sistematico di uno spazio finito opportunamente discretizzato. Ciascun punto dello spazio della funzione viene valutato fino a quando viene trovato il punto di massimo. Ovviamente questi metodi tendono a comportare tempi di ricerca estremamente lunghi non appena il numero di punti dello spazio di ricerca comincia a crescere e possono quindi operare solamente su spazi estremamente ridotti rispetto a gran parte dei problemi di reale interesse.

I metodi di ricerca casuale consistono nel valutare in successione diversi punti dello spazio scelti in modo aleatorio registrando di volta in volta i punti migliori. Per certi versi essi sono simili ai metodi di ricerca enumerativa, in particolare per quanto riguarda i problemi di efficienza descritti sopra. Tuttavia, i metodi di ricerca casuale sono stati resi molto più efficienti con l'impiego delle tecniche di ricottura simulata di cui si è fatto cenno al capitolo 3 (si veda Davis [1987] per un confronto tra ricottura simulata e algoritmi genetici).

Anche gli algoritmi genetici fanno uso di strumenti di ricerca casuale, ma l'intero processo è guidato dalla riproduzione selettiva e si basa su una codifica dei parametri da ottimizzare piuttosto che sui parametri stessi. Queste caratteristiche danno agli algoritmi genetici quelle doti di robustezza che permettono loro di risolvere problemi del mondo reale che risultano invece difficili per le altre tecniche. Essi sono sufficientemente generali per poter essere applicati a una vasta gamma di problemi (è sufficiente trovare una rappresentazione genetica dei parametri del problema), non sono soggetti ai requisiti di con-

[5] Il massimo assoluto è il punto per cui la funzione assume il valore massimo relativamente all'intero dominio (l'ottimo), mentre i massimi locali sono dei punti subottimali nei quali la funzione assume un valore massimo relativamente a un intorno del punto. Massimo assoluto e massimi locali sono l'equivalente del minimo assoluto e dei minimi locali menzionati a proposito della funzione di errore; la differenza di terminologia rispecchia l'interpretazione data al concetto di «ottimo»: mentre nel caso degli algoritmi genetici si desidera *massimizzare* la funzione di *fitness*, nel caso di Back-propagation si desidera *ridurre* la funzione di errore.

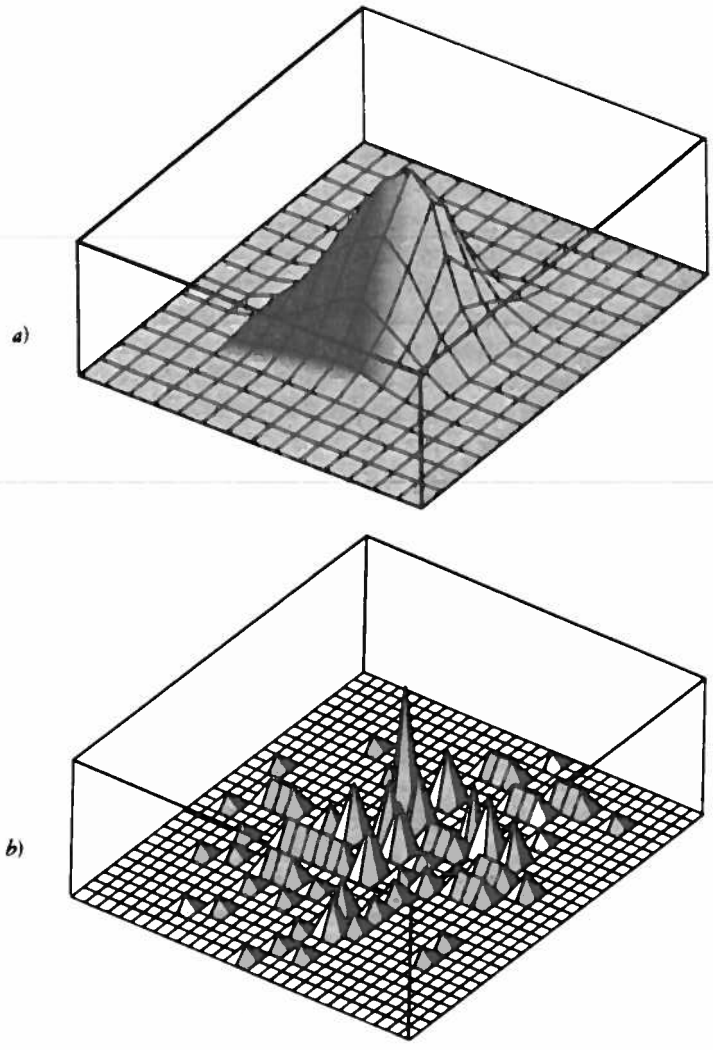


FIG. 6.5. I metodi di ricerca basati sul calcolo infinitesimale individuano facilmente il massimo della funzione *a*), ma – al contrario degli algoritmi genetici – falliscono nel caso della funzione *b*) che è discontinua e presenta molti massimi locali.

tinuità della funzione, non richiedono la visita dell'intero spazio del problema e operano in parallelo su una popolazione di soluzioni variamente distribuite sulla superficie. Un'altra caratteristica interessante riguarda la funzione di *fitness*: essa guida il processo evolutivo svolgendo un ruolo simile alla funzione di errore o di energia impiegata in alcune reti neurali. Tuttavia, mentre queste ultime richiedono una precisa formulazione matematica del problema da risolvere e richiedono la risposta desiderata per ogni vettore di input della rete, la funzione di *fitness* può assumere qualsiasi grado di complessità, a seconda delle conoscenze disponibili, perché il funzionamento degli algoritmi genetici non si basa sull'informazione di gradiente.

6. Reti neurali e algoritmi genetici

Gli algoritmi genetici e le reti neurali sono tecnologie ispirate a processi biologici di adattamento che si svolgono su scale temporali molto diverse: l'evoluzione filogenetica ha luogo attraverso il succedersi di molte generazioni di individui, mentre lo sviluppo del sistema nervoso avviene durante la vita di un singolo individuo. Per valutare l'effetto congiunto di questi due processi, un numero crescente di ricercatori ha iniziato a combinare queste due tecnologie cercando di creare dei sistemi di calcolo che possiedano una potenza adattiva maggiore di quella fornita singolarmente da ciascuno dei due approcci. Oltre all'interesse ingegneristico, la combinazione di un processo evolutivo generazionale con un adattamento rapido durante la vita di un singolo individuo rappresenta un potente strumento per l'investigazione computazionale di tematiche inerenti la biologia, le neuroscienze, la psicologia e l'etologia. Il sistema nervoso degli organismi viventi possiede fin dalla nascita una struttura determinata in gran parte dal codice genetico, la quale viene ulteriormente sviluppata e raffinata dai processi di apprendimento. Questa interazione tra evoluzione e apprendimento solleva una serie di interessanti quesiti: innanzitutto, qual è il vantaggio che deriva da questa combinazione? quali sono i livelli di interazione fra i due processi nello sviluppo del sistema nervoso? in che misura il comportamento di un individuo è determinato da strutture determinate geneticamente e quanto invece influisce l'apprendimento durante la vita?

Quasi tutti gli studi sulla combinazione di questi due diversi approcci computazionali hanno impiegato gli algoritmi genetici per assistere e potenziare le caratteristiche di funzionamento delle reti neurali. Benché esistano già alcune rassegne su questo settore di ricerca [Belew, McIrney e Schraudolph 1992; Mühlenbein e Kindermann 1989; Rudnick 1990; Schaffer, Whitley e Eshelman 1992; Weiss 1990; Yao 1993; 1999], esso manca ancora di quella sistematicità e rigore d'analisi che caratterizza le discipline già mature. Possiamo tuttavia individuare a una prima approssimazione tre tipi di ricerche: l'evoluzione dei pesi sinaptici, l'evoluzione dell'architettura e di altri parametri della rete e l'evoluzione delle regole di apprendimento.

6.1. Rappresentazione genetica di una rete neurale

La combinazione di algoritmi genetici e reti neurali richiede innanzitutto la scelta di una rappresentazione genetica per la rete neurale. Nella maggior parte delle ricerche i pesi sinaptici della rete neurale vengono rappresentati mediante l'uso di codici binari oppure direttamente come numeri reali. In entrambi i casi si presenta il problema della «competizione tra convenzioni» [Schaffer, Whitley e Eshelman 1992; Rojas 1996]. Come abbiamo notato nei paragrafi precedenti, l'algoritmo opera sui genotipi, ovvero sulle stringhe genetiche degli in-

dividui. Per poter assegnare un valore di *fitness* alle stringhe è necessario trasformare ciascun genotipo nel fenotipo corrispondente. Se la trasformazione è del tipo molti-a-uno, lo stesso fenotipo può essere generato da genotipi diversi tra di loro. Consideriamo due genotipi costituiti dalla lista di tutti i pesi sinaptici di due reti (fig. 6.6a). Il processo di decodificazione dà luogo alla costruzione di due reti neurali la cui unica differenza è lo scambio di posizione dei due nodi interni (fig. 6.6b). Siccome i contributi dei nodi interni all'attivazione del nodo di output è dato da una semplice somma pesata, l'inversione di posizione non ha alcun effetto sulle proprietà di funzionamento della rete neura-

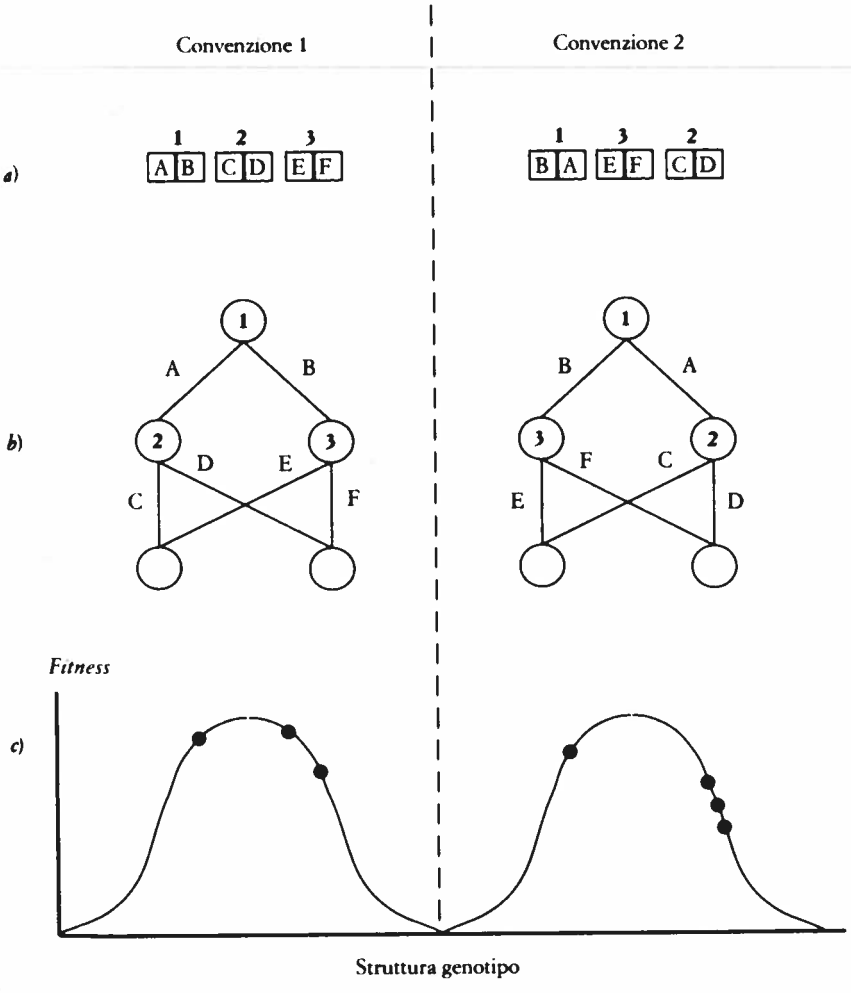


FIG. 6.6. Il problema della competizione fra convenzioni. *a*) Due genotipi diversi; *b*) i due fenotipi corrispondenti la cui unica differenza è l'inversione di posizione dei due nodi interni; *c*) la superficie della *fitness* diventa multimodale.

Fonte: Schaffer, Whitley e Eshelman [1992].

le e le due reti otterranno lo stesso valore di *fitness*. Siccome l'algoritmo genetico non è a conoscenza delle convenzioni scelte per la rappresentazione e decodificazione dei fenotipi, esso si trova ad operare su genotipi completamente diversi. Immaginiamo ora che le due reti neurali abbiano riportato una buona *fitness* e i due genotipi siano stati selezionati entrambi per la riproduzione. Il *crossover* tra questi due individui potrebbe produrre degli individui scadenti se essi si scambiano i due nodi interni: avremmo così due figli ciascuno dei quali possiede due nodi dello stesso tipo (che eseguono lo stesso calcolo) e non è più in grado di svolgere le funzioni che il nodo mancante svolgeva nelle reti dei genitori. La competizione tra convenzioni diverse (la posizione di un determinato nodo nella stringa genetica) può quindi trasformare un singolo massimo della funzione di fitness in parecchi massimi distinti, complicando in questo modo il processo di ricerca (fig. 6.6c). Gli incroci fra genitori che utilizzano la stessa convenzione avranno una maggior probabilità di produrre buoni figli rispetto agli incroci tra genitori che appartengono a convenzioni diverse. Radcliffe [1990; 1991] ha fatto notare che il numero di convenzioni diverse cresce in modo esponenziale rispetto al numero di nodi interni della rete perché ciascuna permutazione rappresenta un ordinamento diverso dei nodi, ovvero una diversa convenzione. La presenza di molte convenzioni diverse potrebbe rendere lo sviluppo evolutivo inefficace, per lo meno fino a quando, per un processo di deriva genetica, l'intera popolazione converge verso un'unica convenzione[6].

Di fatto il problema della competizione tra convenzioni è più che altro teorico. Hancock [1992] ha affrontato empiricamente questo problema presentando una serie di risultati in cui il processo evolutivo non sembra essere disturbato dalla presenza di convenzioni diverse. Inoltre il problema si presenta solamente quando i tagli di *crossover* avvengono esattamente tra due nodi dello stesso strato: quanto maggiore è il grado di connettività tanto minore è la probabilità di scambi infruttuosi. Infine, se la rappresentazione genetica include connessioni ricorrenti e laterali oppure fa uso di schemi più evoluti (che esamineremo più sotto), la posizione di ciascun nodo assume un significato importante per il funzionamento della rete, eliminando così del tutto il problema di convenzioni diverse.

7. Evoluzione dei pesi sinaptici

Vi sono una serie di motivi per cui potrebbe risultare vantaggioso ottimizzare i pesi sinaptici di una rete neurale con gli algoritmi geneti-

[6] Secondo Schaffer, Whitley e Eshelman [1992] questo momento coincide di solito anche con una riduzione della varietà del materiale genetico disponibile e corrisponde quindi con una convergenza prematura del processo evolutivo verso una soluzione sub-ottimale.

ci. Mentre gli algoritmi di apprendimento considerati nei capitoli precedenti ricercano la soluzione partendo da una singola configurazione iniziale di valori sinaptici, gli algoritmi genetici operano su una popolazione di reti neurali con pesi sinaptici diversi: questa ricerca globale dello spazio dei pesi risulta in una probabilità minore di convergere verso una soluzione sub-ottimale della funzione da ottimizzare. Consideriamo la funzione raffigurata nella porzione inferiore della figura 6.5. Una rete neurale addestrata con Back-propagation finirebbe probabilmente per stabilizzarsi sulla cima del picco più vicino al punto di partenza dato dalla configurazione iniziale dei pesi sinaptici. Nel caso degli algoritmi genetici invece l'inizializzazione casuale dei pesi sinaptici di tutte le reti neurali della popolazione colloca le varie reti in punti diversi dello spazio, permettendo all'operatore di riproduzione selettiva di scegliere le reti più vicine alle zone di massimo della funzione.

Gli algoritmi genetici potrebbero rivelarsi efficienti anche in situazioni in cui la funzione che la rete neurale deve apprendere non è continua (per cui non è possibile calcolarne il gradiente per ogni punto), oppure in situazioni in cui non è sempre disponibile la risposta desiderata necessaria alla correzione dei pesi sinaptici. Entrambe queste situazioni si presentano frequentemente nei problemi di controllo di sistemi che interagiscono con un ambiente reale. L'apprendimento per rinforzo rappresenta una possibile soluzione, ma esso presenta serie limitazioni quando la funzione da apprendere presenta forti discontinuità oppure quando il rinforzo viene fornito dopo lunghe sequenze. Come vedremo nell'ultimo paragrafo di questo capitolo, la combinazione di algoritmi genetici e reti neurali sta riscontrando un grande successo per il controllo di agenti sensomotori.

Un altro vantaggio offerto dall'ottimizzazione evolutiva dei pesi sinaptici consiste nel fatto che – al contrario di quanto richiesto dagli algoritmi di apprendimento – non vi sono restrizioni sull'architettura della rete, sul modo in cui vengono combinati i segnali provenienti dalle connessioni e sul tipo di funzione di attivazione delle unità.

Montana e Davis [1989] hanno confrontato le prestazioni di un algoritmo genetico e di Back-propagation in un compito di classificazione di segnali sonar provenienti da una serie di ricevitori acustici collocati sul fondale oceanico. La classificazione di questi dati risulta particolarmente difficoltosa per sistemi a regole perché il segnale è mescolato a una grande quantità di interferenza causata da correnti marine, oggetti in sospensione, diverse zone di conduttività, rumore oceanico di fondo e zone a diversa salinità e temperatura (che influiscono sulla conduttività acustica dell'acqua). Gli autori hanno impiegato un algoritmo genetico con riproduzione a regime costante e hanno codificato i pesi sinaptici delle reti neurali nelle stringhe genetiche usando numeri reali («codificazione diretta»); la funzione di *fitness* era l'inverso della somma dei quadrati degli errori riportati da ciascuna rete sull'intera lista di pattern. Per ogni sostituzione di una stringa ge-

netica la *fitness* migliore della popolazione veniva confrontata con il valore ottenuto dalla rete migliore di un gruppo di dieci reti ciascuna addestrata con Back-propagation: la generazione e sostituzione di una stringa veniva dunque equiparata a un ciclo di addestramento su un pattern casuale con Back-propagation. I risultati indicano che l'algoritmo genetico ottiene prestazioni migliori di Back-propagation in quanto converge più rapidamente e la qualità della soluzione è migliore in quanto la somma degli errori quadratici è minore. Questi risultati sono stati successivamente confermati anche per un diverso compito di classificazione usando la stessa metodologia [Whitley, Starkweather e Bogart 1990]. Un altro aspetto interessante della ricerca riguarda il nuovo operatore di mutazione impiegato dagli autori: ai componenti delle stringhe selezionati per la mutazione viene *aggiunto* un numero reale estratto casualmente da un piccolo intervallo intorno allo zero (compreso tra −1,0 e +1,0). Questa modifica probabilmente funziona meglio perché, invece di sostituire i pesi sinaptici delle stringhe migliori con valori casuali completamente nuovi (come prevederebbe l'operatore di mutazione tradizionale), la mutazione mantiene la tendenza di sviluppo dei pesi sinaptici spostandoli di poco intorno al loro valore originale.

Whitley, Dominic e Das [1991] hanno studiato il funzionamento di algoritmi genetici che operano su popolazioni ridotte (con meno di 50 individui) di stringhe composte di numeri reali con una probabilità di mutazione abbastanza alta. In questi casi, quando viene impiegato un regime di riproduzione stazionario, l'algoritmo genetico perde le proprietà di ricerca globale e parallela e diventa invece uno strumento di ottimizzazione stocastico non dissimile dalla produzione e valutazione di singole stringhe generate aleatoriamente.

7.1. Evoluzione e apprendimento

Le caratteristiche di ricerca globale dell'algoritmo genetico possono essere combinate con le caratteristiche di ricerca locale di un algoritmo di apprendimento neurale. Belew, McIrney e Schraudolph [1992] hanno utilizzato un algoritmo genetico per individuare i pesi iniziali di reti neurali addestrate con Back-propagation. Le prestazioni di Back-propagation, come tutte le tecniche di discesa del gradiente della funzione, dipendono molto dalle condizioni iniziali [Kolen e Pollack 1990]. Questo problema può essere parzialmente risolto se gli algoritmi genetici vengono usati per evolvere i pesi iniziali di una popolazione di reti usando come funzione di *fitness* l'inverso della somma dei quadrati degli errori dopo un certo numero di epoche di apprendimento con Back-propagation. I risultati indicano che gli algoritmi genetici riescono a localizzare delle configurazioni di pesi iniziali che permettono a Back-propagation di convergere in tempi molto ridotti (fino a due ordini di grandezza rispetto a inizializzazioni casuali).

Nello stesso lavoro gli autori hanno impiegato gli algoritmi genetici anche per evolvere i valori ottimali dei parametri di addestramento di Back-propagation, ovvero il tasso di apprendimento η e il momentum α: sorprendentemente, l'algoritmo genetico ha sviluppato sempre valori due o tre volte maggiori di quelli tradizionalmente utilizzati.

Ackley e Littman [1992] hanno studiato l'interazione fra evoluzione e apprendimento in organismi artificiali che interagiscono con un ambiente fisico che contiene pezzi di cibo, rifugi, predatori e altri oggetti. Ciascun organismo è controllato da una rete neurale composta di due moduli: un modulo per l'azione e un modulo per la valutazione (fig. 6.7), come per le architetture usate nei modelli di apprendimento per rinforzo. Il modulo dell'azione è una rete neurale feedforward che trasforma l'informazione sensoriale in probabilità di emettere una certa azione; il modulo di valutazione invece fornisce una valutazione della situazione e fornisce il segnale di rinforzo per il modulo di azio-

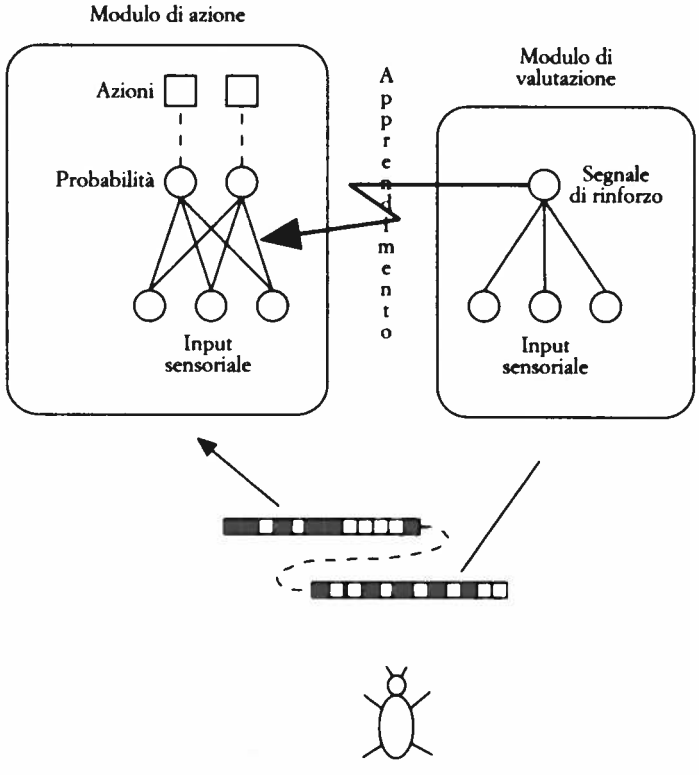


FIG. 6.7. Apprendimento per rinforzo evolutivo. Un organismo, il suo codice genetico e il sistema di controllo corrispondente. Il genotipo codifica i pesi sinaptici del modulo di valutazione e del modulo di azione; il modulo di azione modifica i propri pesi sinaptici durante la vita dell'organismo mediante il segnale fornito dal modulo di valutazione, ma le modifiche apportate durante la vita non vengono trascritte nel codice genetico.

ne. L'algoritmo genetico evolve sia i pesi sinaptici della rete di azione che quelli della rete di valutazione, ma durante la vita di ciascun individuo la rete di azione modifica i propri pesi sinaptici utilizzando il segnale di rinforzo fornito dalla rete di valutazione. L'algoritmo di modifica sinaptica durante la vita mira a rinforzare i pesi sinaptici della rete di azione quando la rete di valutazione fornisce un rinforzo positivo e tende invece a diminuirli quando il rinforzo è negativo: in accordo con il modello darwiniano dell'evoluzione, i pesi sinaptici finali della rete di azione sviluppati durante la vita dell'organismo non vengono ricopiati nel codice genetico. Ciascun organismo è in grado di riprodursi quando ha accumulato sufficiente energia (mangiando cibi e sfuggendo ai predatori) e si trova in prossimità di un altro organismo atto alla riproduzione; la creazione dei figli avviene impiegando l'operatore di *crossover* e di mutazione sulle copie delle stringhe genetiche dei due genitori. Gli organismi muoiono quando non riescono a mantenere un livello energetico sufficiente oppure quando oltrepassano una certa età. Il livello energetico rappresenta dunque la funzione di *fitness* intrinseca del modello evolutivo. Gli autori hanno confrontato tre diverse situazioni: evoluzione di pesi sinaptici senza apprendimento, apprendimento solamente (partendo da pesi sinaptici iniziali casuali) e combinazione di evoluzione e apprendimento. Quest'ultima combinazione – definita «apprendimento per rinforzo evolutivo» – è risultata migliore delle altre riuscendo a mantenere la popolazione in vita per oltre tremila generazioni. Da un'analisi delle popolazioni succedutesi nel corso delle simulazioni è emerso che mentre durante le prime 600 generazioni le reti neurali di azione non erano in grado di localizzare fin dalla nascita le piante distribuite nell'ambiente per cibarsene, ma dovevano imparare a farlo tramite il rinforzo fornito dalla rete di valutazione, nelle generazioni successive questa conoscenza era presente fin dalla nascita nelle reti di azione degli organismi. L'influenza dell'apprendimento ontogenetico sull'evoluzione filogenetica, noto in biologia con il nome di «effetto Baldwin» dal nome del naturalista che per primo osservò questo effetto [Baldwin 1896], sembrerebbe a prima vista confermare le ipotesi di Lamarck, il quale pensava che le conoscenze acquisite durante la vita rimanessero impresse nel codice genetico degli organismi. Esiste però una spiegazione di questo fenomeno all'interno del quadro teorico darwiniano. L'apprendimento durante la vita di un organismo svolgerebbe una funzione simile a quella di una «torcia» che illumina il paesaggio della superficie della *fitness* intorno all'organismo favorendo così la riproduzione selettiva di individui, il cui corredo genetico – tenendo conto dell'effetto potenziale dell'apprendimento – corrisponde a una regione (e non più solamente a un punto) nella quale la funzione di *fitness* assume valori elevati [Hinton e Nowlan 1987]. Consideriamo il caso specifico di due stringhe – A e B – che codificano i pesi sinaptici di reti neurali diverse. La figura 6.8 illustra la posizione delle due stringhe sulla superficie della *fitness* individuata dal campo di possibili variazioni dei pesi sinaptici

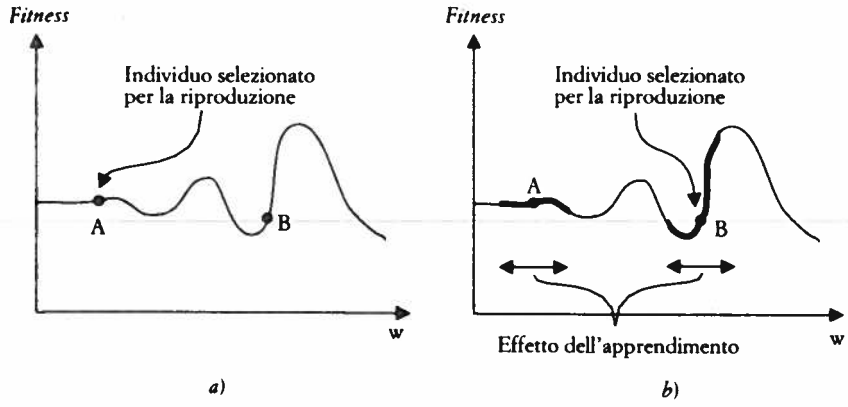


FIG. 6.8. Spiegazione computazionale dell'effetto Baldwin. L'apprendimento durante la vita permette un'esplorazione del paesaggio locale della *fitness* di ciascun individuo favorendo così la riproduzione di individui che si trovano in zone migliori. La figura mostra la collocazione di due stringhe genetiche (A e B) corrispondenti a reti neurali diverse in due diverse condizioni evolutive. *a)* Solo evoluzione: la stringa A possiede una probabilità maggiore di riprodursi perché possiede una *fitness* più alta della stringa B; *b)* l'apprendimento durante la vita modifica i pesi sinaptici spostando la rete neurale sulla superficie della *fitness*: questo campionamento delle caratteristiche locali della funzione di *fitness* intorno alla stringa genera una probabilità di riproduzione maggiore per la stringa B che per la stringa A. La stringa B infatti è quella più vicina al massimo della funzione di *fitness*.

delle reti. Durante un normale processo evolutivo senza apprendimento durante la vita (fig. 6.8a), l'operatore di riproduzione selettiva assegna una maggior probabilità di riproduzione alla stringa A poiché essa ottiene una *fitness* maggiore rispetto alla stringa B, anche se la stringa A si trova in una zona relativamente scadente della funzione di *fitness*. Quando invece l'evoluzione è combinata all'apprendimento (fig. 6.8b) la possibilità offerta dall'apprendimento di modificare i pesi sinaptici durante la vita dei due fenotipi corrisponde a un'esplorazione dei dintorni del punto di partenza individuato dai pesi sinaptici iniziali: questo campionamento delle prestazioni per un certo intervallo di variazione nei dintorni della posizione iniziale produce un valore di *fitness* maggiore per la stringa B, la quale avrà quindi una maggiore probabilità di riprodursi rispetto alla stringa A. Il processo evolutivo risulta così favorito anche se i pesi sinaptici finali non rimangono impressi nel genotipo dell'individuo perché le stringhe selezionate sono quelle che si collocano in prossimità di regioni migliori del dominio della funzione di *fitness* e necessitano quindi di poche modifiche genetiche per raggiungere una soluzione ottimale. È importante osservare che il processo evolutivo sarebbe avvantaggiato anche da semplici mutazioni casuali delle connessioni sinaptiche durante la vita dell'individuo: in effetti i campi di variazione delle due stringhe nella parte destra della figura 6.8 corrispondono a mutazioni casuali uniformi. Un processo di apprendimento tuttavia esplora lo spazio della *fitness* in modo più ef-

ficace di una semplice ricerca casuale; ad esempio, se l'apprendimento delle due reti neurali è basato sull'informazione di gradiente (come Back-propagation), la stringa B si sposterà rapidamente verso la cima del picco più alto e otterrà una probabilità ancor maggiore di riprodursi rispetto alla stringa A. L'apprendimento ontogenetico migliora l'efficacia del processo evolutivo, il quale a sua volta migliora la capacità di apprendimento posizionando le reti neurali vicino ai massimi della funzione da ottimizzare.

Parisi e collaboratori [Nolfi, Elman e Parisi 1994; Parisi, Cecconi e Nolfi 1990] hanno studiato ulteriormente le caratteristiche dell'interazione tra evoluzione e apprendimento in reti neurali di organismi artificiali che interagiscono con un ambiente. I loro esperimenti sono motivati dall'osservazione che in molte situazioni quello che viene appreso durante la vita di un organismo è collegato solo in modo indiretto agli scopi di un comportamento determinato geneticamente; in altre parole, quello che viene appreso non è il comportamento stesso, ma qualche altra capacità che potrebbe risultare utile a quel comportamento. Gli autori hanno analizzato l'evoluzione di reti neurali di organismi artificiali che devono sviluppare la capacità di raggiungere dei cibi distribuiti nell'ambiente. A differenza del sistema di Ackley e Littman [1992], le reti neurali possiedono una struttura monolitica con due gruppi di unità di output (fig. 6.9): due unità codificano le azioni eseguite dall'organismo e le altre due unità codificano la predi-

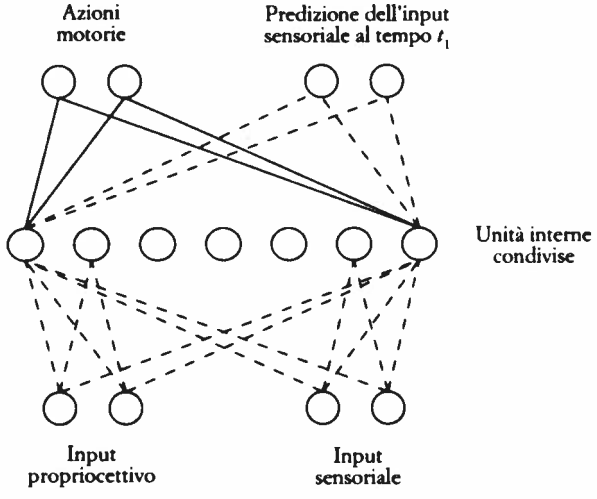


FIG. 6.9. La rete neurale impiegata negli esperimenti di Parisi, Cecconi e Nolfi [1990] per la generazione di azioni motorie e predizione delle conseguenze sensoriali di tali azioni. Tutti i pesi sinaptici sono evoluti con un algoritmo genetico, ma i pesi sinaptici tratteggiati sono modificati anche durante la vita dell'organismo per l'apprendimento della predizione sensoriale. Le modifiche acquisite durante la vita dell'organismo non restano impresse nel codice genetico.

zione del cambiamento sensoriale verificatosi in seguito all'azione eseguita. Entrambi i gruppi di unità ricevono segnali dallo stesso gruppo di unità nascoste, ma le unità per la predizione delle conseguenze sensoriali possono sfruttare l'informazione sensoriale dopo l'esecuzione dell'azione per generare un vettore di errore che viene impiegato per modificare con l'algoritmo di Back-propagation i pesi sinaptici fino alle unità di input della rete[7].

I risultati indicano che gli organismi che apprendono a predire le conseguenze delle proprie azioni durante la vita sviluppano più rapidamente la capacità di raggiungere il cibo (riportando anche prestazioni migliori) rispetto agli organismi che non possono apprendere. I compiti di predizione e di raggiungimento dei cibi si supportano a vicenda: organismi in grado di predire meglio riescono a individuare quantità maggiori di cibo, mentre gli organismi delle ultime generazioni riescono a predire meglio rispetto a quelli delle prime generazioni (che non sono ancora in grado di raggiungere molti cibi). Questi risultati non solo confermano l'effetto Baldwin, ma indicano anche che l'apprendimento di un compito diverso (predizione delle conseguenze sensoriali), ma collegato positivamente con il compito evolutivo (raggiungimento di cibi), è in grado di accelerare e migliorare il processo evolutivo.

Infine, Yamauchi e Beer [1994] hanno evoluto i valori sinaptici di reti neurali completamente ricorrenti che devono essere in grado di produrre determinate sequenze di stati in funzione di un segnale di rinforzo esterno applicato a un nodo di input della rete; anche se la rete può modificare i valori sinaptici solo mediante gli operatori genetici, il processo evolutivo genera sistemi dinamici che esibiscono cambiamenti di stato analoghi a quelli generati dagli algoritmi di apprendimento.

8. Evoluzione dell'architettura

L'architettura di una rete neurale influisce sensibilmente sulle sue prestazioni. Come è stato osservato nel secondo capitolo, esistono alcuni metodi per modificare dinamicamente l'architettura della rete neurale durante l'apprendimento, ma essi impongono una serie di vincoli computazionali (in quanto gli algoritmi sono essenzialmente basati su una misura di errore) che non sempre possono essere soddisfatti. Un metodo alternativo consiste nell'evolvere l'architettura della rete con un algoritmo genetico. Gli algoritmi genetici rappresentano un'opzione interessante per l'ottimizzazione dell'architettura perché la superficie della funzione corrispondente all'ottimizzazione di architetture

[7] Si noti che le condizioni di apprendimento sono ecologicamente plausibili perché la risposta corretta di predizione non proviene da un insegnante esterno, ma è fornita direttamente dall'ambiente stesso mentre l'organismo si sposta.

è discontinua e quindi non differenziabile e la corrispondenza tra architetture e prestazioni è «rumorosa» e non univoca [Miller, Todd e Hedge 1989].

L'impiego degli algoritmi genetici richiede la scelta di una rappresentazione degli elementi della rete da evolvere. Negli approcci descritti nei paragrafi precedenti ciascuna connessione della rete neurale viene esplicitamente codificata nella stringa genetica: questa codificazione diretta permette l'ottimizzazione «fine» di tutti i parametri della rete neurale, ma si rivela adeguata solamente per reti neurali relativamente piccole. Reti di dimensioni maggiori richiedono codici genetici molto lunghi, per cui gli schemi migliori tendono ad essere distrutti dall'operatore genetico di ricombinazione e l'algoritmo genetico non produce sempre soluzioni ottimali (Maniezzo [1994] però ha presentato un metodo evolutivo per codificare sia la topologia che i valori dei pesi sinaptici della rete neurale in modo compatto dove l'algoritmo sceglie automaticamente il grado di precisione con cui i valori sinaptici vengono codificati in numeri binari). Una scelta alternativa consiste nel codificare solamente alcuni parametri importanti dell'architettura, come ad esempio il numero di nodi, il tipo di connettività, le funzioni di attivazione, ecc. Questo tipo di rappresentazione viene definita con il nome di «codificazione indiretta» [Yao 1993]. Mentre l'evoluzione si occupa dell'ottimizzazione di queste caratteristiche architetturali, l'apprendimento durante la vita si preoccupa di ottimizzare i valori sinaptici.

Uno dei primi studi sull'evoluzione delle architetture neurali è stato presentato da Harp, Samad e Guha [1989]. La rappresentazione genetica della rete neurale consiste in un «progetto» costituito di uno o più segmenti binari, ciascuno dei quali codifica le proprietà di uno strato della rete. Ogni segmento è composto di due parti: una parte descrive alcuni parametri dello strato (l'indirizzo dell'area, il numero di nodi, il modo in cui sono disposti geometricamente e il tasso di modifica delle soglie) e l'altra parte descrive le caratteristiche delle connessioni efferenti (densità di connessione, indirizzo dell'area di proiezione, proprietà geometriche della proiezione, tasso di modifica sinaptica delle connessioni). Sia le aree che le due parti che compongono ciascuna area sono separate da marcatori genetici che vengono rispettati dall'operatore di *crossover* affinché la ricombinazione non crei stringhe genetiche senza senso. Ciascuna stringa genetica viene decodificata nella rete neurale corrispondente e i valori sinaptici, inizialmente generati in modo casuale, vengono modificati con Back-propagation per apprendere un determinato compito. Le prestazioni delle reti (gli autori hanno impiegato diversi criteri di valutazione) vengono impiegate dall'algoritmo genetico per riprodurre selettivamente le stringhe migliori, incrociarle e applicare delle mutazioni casuali. I risultati indicano che l'algoritmo genetico è in grado di produrre architetture efficienti dal punto di vista dell'apprendimento solo definendo opportunamente la funzione di *fitness*. In un compito di riconosci-

mento di caratteri, le reti migliori delle ultime generazioni possedevano relativamente poche connessioni quando la *fitness* includeva un termine di costo per il «materiale neurale» impiegato; quando invece la *fitness* premiava la velocità di apprendimento, le reti neurali apprendevano circa dieci volte più rapidamente che nel caso precedente, ma possedevano molte più connessioni.

Kitano [1990] ha utilizzato invece uno schema di sviluppo in cui il genotipo descrive una serie di regole di sviluppo dell'architettura piuttosto che codificare direttamente i vari parametri di tutti i nodi e aree della rete. Una regola di sviluppo consiste in un'equazione ricorsiva che viene applicata a una semplice matrice di connessioni iniziali; l'applicazione ripetuta di questa equazione determina la crescita della matrice tramite la duplicazione delle strutture create al passo precedente. Lo schema di sviluppo di Kitano si ispira ai «sistemi L» di Lindenmayer che sono stati impiegati per spiegare il processo di morfogenesi delle piante e di alcuni organismi marini [Lindenmayer 1968; 1971; Prusinkiewicz e Lindenmayer 1992]. Poiché il genotipo può codificare 16 diversi tipi di regole di sviluppo, questa rappresentazione genetica è in grado di generare reti neurali di varie dimensioni e architetture diverse a partire da stringhe genetiche molto corte. Tuttavia, siccome le regole di sviluppo determinano solamente l'architettura della rete, il processo di ottimizzazione dei valori sinaptici deve essere effettuato con un algoritmo di apprendimento tradizionale. Un approccio simile è stato applicato con successo anche da Boers e Kuiper [1992] per l'evoluzione di reti modulari che ottengono prestazioni migliori delle reti monolitiche che vengono tradizionalmente impiegate quando l'architettura è predefinita dallo sperimentatore. Gruau [1992] ha impiegato alberi grammaticali per la codificazione del processo di sviluppo cellulare di famiglie di reti neurali booleane (i cui pesi sinaptici possono assumere solo valore +1 o –1); successivamente egli ha aggiunto la possibilità di apprendere confrontando l'effetto Baldwin con l'ipotesi lamarckiana [Gruau 1993].

Nolfi e collaboratori [Nolfi, Miglino e Parisi 1994; Nolfi e Parisi 1994] hanno combinato l'evoluzione genetica con un processo di «maturazione» delle reti neurali di organismi artificiali. Le reti neurali si sviluppano sulla base di un genotipo che codifica la crescita dell'architettura: al contrario dei lavori descritti sopra, in questo approccio la rete neurale rimane plastica per un certo periodo dopo la nascita dell'individuo. Così come avviene in natura, la fase di crescita della rete neurale occupa una parte considerevole della vita dell'individuo e si adatta alle contingenze ambientali in funzione delle soluzioni genetiche ereditate per la sopravvivenza dell'individuo in un certo ambiente fisico. Il codice genetico possiede una lunghezza costante ed è suddiviso in blocchi ciascuno dei quali codifica una serie di proprietà di un nodo della rete: queste proprietà definiscono la posizione, le dinamiche di maturazione e la connettività. La rete neurale è concepita come una struttura fisica che si stende su uno

spazio bidimensionale. Il gene di ogni nodo codifica in coordinate cartesiane x e y la posizione del nodo sul tessuto neurale, una variazione di attività minima oltre la quale il nodo comincia a sviluppare un assone, l'angolo di ramificazione dell'assone, la lunghezza dei segmenti assonali, la soglia del neurone e un valore unico per tutte le sinapsi corrispondenti ai punti di contatto delle ramificazioni assonali con altri nodi della rete. Le sinapsi vengono create solamente quando le ramificazioni assonali raggiungono un altro nodo sul tessuto neurale; i segmenti assonali che terminano in zone ove non vi sono nodi muoiono e vengono eliminati. Il processo di maturazione dipende sia da quanto codificato nel codice genetico che dall'attività neurale durante la vita dell'individuo perché la crescita dipende anche dalla variazione di attivazione di ciascun nodo, la quale a sua volta dipende dall'interazione fra l'organismo e l'ambiente in cui esso vive. Questo schema è in grado di creare strutture modulari interessanti da un punto di vista biologico che si sono rivelate particolarmente efficienti anche per l'evoluzione di sistemi di controllo di robot mobili. Successivamente il modello è stato reso ancor più plausibile da un punto di vista biologico aggiungendovi uno schema di divisione e migrazione cellulare [Cangelosi, Parisi e Nolfi 1994]: la rete neurale si sviluppa a partire da un embrione che si divide in due parti ciascuna delle quali si sposta sul tessuto cellulare. Alla fine del processo ciclico di suddivisione e migrazione, comincia la fase di crescita e ramificazione assonale. Questo complesso schema evolutivo, che è stato impiegato per lo sviluppo di moduli neurali di controllo di organismi artificiali che interagiscono con un ambiente, ha permesso di indagare importanti aspetti di sviluppo genetico e biologico in uno schema integrato che prende in considerazione aspetti genetici, biologici, comportamentali e computazionali.

9. Evoluzione di regole di apprendimento

Gli algoritmi genetici sono stati impiegati anche per l'evoluzione delle regole di apprendimento sinaptico. Una regola di apprendimento è una funzione che combina alcune variabili, solitamente l'attività presinaptica x_j, l'attività postsinaptica y_i e il valore corrente della sinapsi w_{ij}

$$\Delta w_{ij} = \eta \Phi(x_j, y_i, w_{ij})$$

dove η è il tasso di apprendimento. Data un'architettura e un problema da apprendere, l'algoritmo genetico viene usato per individuare la combinazione delle tre variabili che dà luogo alle prestazioni migliori della rete neurale. Solitamente la rappresentazione genetica delle regole di apprendimento richiede la riscrittura della regola come una combinazione lineare dei prodotti fra le variabili stesse ciascuno pesato da

un coefficiente; ad esempio, per la funzione di tre variabili descritta sopra

$$\Delta w_{ij} = \eta(a_1 x_i x_j + a_2 x_i y_k + a_3 x_i w_{ij} + a_4 y_j y_k + a_5 y_k w_{ij} + a_6 w_{ij} w_{ij})$$

se prendiamo in considerazione solamente i prodotti di primo e secondo ordine. Nel caso più semplice i coefficienti a_n possono assumere valori discreti {–1, 0, +1}, ma è anche possibile attribuire loro valori continui nell'intervallo [–1, +1]. Il codice genetico codifica i valori dei coefficienti a_n in una stringa binaria; ciascuna stringa viene a turno decodificata in una regola di apprendimento, la quale viene poi impiegata per addestrare i pesi sinaptici della rete neurale su uno o più problemi di apprendimento (ad esempio il riconoscimento di caratteri). La *fitness* è data dalle prestazioni della rete neurale alla fine dell'apprendimento su un insieme di pattern di generalizzazione.

Chalmers [1990] è stato il primo autore a utilizzare questo schema per evolvere le regole di apprendimento di una rete con un solo strato di connessioni sinaptiche e una sola unità di output su otto compiti linearmente separabili dove la funzione di *fitness* era data dall'errore quadratico medio tra la risposta della rete e la risposta desiderata alla fine dell'apprendimento; ciascuna stringa genetica codificava solamente i coefficienti della funzione di apprendimento e i pesi sinaptici iniziali della rete venivano sempre assegnati casualmente entro un piccolo intervallo intorno allo zero. L'algoritmo genetico ha «riscoperto» la nota regola delta (si veda il secondo capitolo) e alcune sue varianti. Fontanari e Meir [1991] hanno utilizzato lo stesso schema di Chalmers per evolvere le regole di apprendimento del percettrone.

Dasdan e Oflazer [1994] hanno impiegato un approccio simile per evolvere le regole di apprendimento di mappe topologiche neurali. Anche questi autori hanno utilizzato sei diversi compiti su cui addestrare e testare le regole di apprendimento evolute per evitare che l'algoritmo genetico sviluppasse soluzioni specifiche per un determinato problema. La loro funzione di *fitness* è data dalla differenza tra i pattern di addestramento di ciascun problema e i rispettivi pesi sinaptici sviluppati dalla rete alla fine dell'addestramento (si ricordi dal quarto capitolo che l'apprendimento delle mappe topologiche neurali consiste nello spostare ciascun vettore sinaptico su uno specifico gruppo di pattern di addestramento). In questo caso l'algoritmo genetico è riuscito a evolvere regole di apprendimento più efficaci della regola di Kohonen, anche se più complesse nel loro funzionamento.

Baxter [1994] invece ha evoluto sia l'architettura che le dinamiche di apprendimento di una rete neurale. Il codice genetico di ciascuna rete neurale codifica il segno della sinapsi (eccitatoria o inibitoria), i nodi che essa unisce e il fatto che essa possa essere modificata in base alla regola di Hebb oppure è costretta a mantenere un valore costante. Ciascuna stringa viene decodificata in una rete neurale e addestrata su quattro funzioni booleane di una singola variabile. Le reti migliori ot-

tenute da Baxter possiedono pochi nodi variamente collegati fra di loro e utilizzano la regola di Hebb in modo distribuito per generare comportamenti dinamici: dato un input costante la rete passa attraverso una serie di stati fino a stabilizzarsi sulla risposta corretta (l'attrattore).

Floreano e Mondada [1995] hanno impiegato uno schema simile dove la codificazione genetica delle sinapsi specifica il segno, l'effetto postsinaptico del segnale trasportato (principale o modulatorio), il tasso di apprendimento e quattro diverse regole hebbiane (Hebb semplice, presinaptica, postsinaptica e covarianza; si veda il primo capitolo). Ciascuna stringa viene decodificata in una rete neurale e trasferita in un robot mobile in tempo reale, dove la funzione di *fitness* consiste nella capacità di navigazione ed evitamento di ostacoli del robot. L'algoritmo genetico ha evoluto in meno di cinquanta generazioni una rete neurale in grado di generare abilità di navigazione in meno di due secondi a partire da pesi sinaptici inizializzati con valori casuali; inoltre, siccome la rete neurale può apprendere continuamente, il robot è in grado di adattarsi a qualsiasi ambiente fisico modificando le proprie strategie a seconda delle caratteristiche locali dell'ambiente (presenza di ostacoli, curvatura delle pareti, ecc.).

10. Vita artificiale e Psicologia sintetica

La Vita artificiale è un nuovo campo di ricerca che si propone di comprendere i principi costitutivi della vita in un quadro più ampio di quello definito dal particolare tipo di vita che si è evoluta sul nostro pianeta, le cui caratteristiche sono studiate dalla biologia. La Vita artificiale vuole studiare non solo la logica e i principi costitutivi della vita biologica terrestre, ma anche delle possibili vite e forme intelligenti che potrebbero esistere nell'universo. Le domande a cui essa tenta di rispondere sono: che cos'è la vita? come si è evoluta su questo pianeta? quali sono le altre forme di vita possibili? quali sono i principi di formazione dei sistemi intelligenti? Chris Langton, uno dei fondatori di questo campo di ricerca, definisce la Vita artificiale come un nuovo settore di ricerca che impiega un approccio «sintetico» allo studio della vita «come-potrebbe-essere» perché essa interpreta la vita come una proprietà dell'organizzazione della materia, piuttosto che come una proprietà della materia che si è organizzata in un certo modo [Langton 1992a]. Il metodo sintetico, ovvero il tentativo di comprendere attraverso un processo di ricostruzione, è giustificato dal fatto che le attuali tecniche analitiche non permettono di comprendere pienamente fenomeni caratterizzati da trasformazioni non-lineari, corrispondenze molti-a-uno e proprietà caotiche quando gli unici elementi che abbiamo a disposizione sono i prodotti finali di tali trasformazioni, come nel caso delle forme viventi [Langton 1992b]. La Vita artificiale rappresenta la base costruttiva dell'Intelligenza artificiale perché la

capacità di mantenersi in vita rappresenta la preoccupazione principale per un organismo [Belew 1991; Steels 1993; 1994]. In altre parole, le abilità di base che permettono a un organismo di mantenersi in vita costituiscono le fondamenta e i mattoni con i quali è possibile costruire forme di intelligenza più elevate.

L'evoluzione darwiniana occupa un ruolo centrale in gran parte della ricerca svolta in Vita artificiale e varie tecniche simulative sono state utilizzate per evolvere forme di vita artificiali in campi che spaziano dalla biologia molecolare alla morfogenesi delle piante fino all'evoluzione di comportamenti intelligenti in organismi dotati di sistemi nervosi artificiali. L'idea di base consiste nel creare un sistema iniziale costituito di semplici parti, definire alcune regole di trasformazione e lasciare che il sistema evolva autonomamente studiandone le fasi di sviluppo e i prodotti finali emergenti. Questo approccio è stato impiegato anche per comprendere la nascita di comportamenti intelligenti in semplici organismi artificiali. Questo particolare settore della Vita artificiale, definito anche con il nome di Psicologia sintetica [Braitenberg 1984] o Neuroetologia computazionale [Beer 1990; Cliff 1991; Floreano 1995], cerca di superare le inadeguatezze della modellizzazione cognitiva che caratterizzano sia il cognitivismo (e l'Intelligenza artificiale) che il connessionismo: entrambi questi approcci tendono a proporre modelli di prestazioni percettive e cognitive che finiscono con l'essere dei meccanismi passivi di input-output in grado di assorbire informazione esterna ed emettere risposte in base a qualche criterio imposto esternamente dallo sperimentatore. Molti esperimenti e teorizzazioni sulle reti neurali sembrano assumere che l'apprendimento abbia luogo nel vuoto [Parisi, Cecconi e Nolfi 1990]. È invece importante notare che qualsiasi organismo dotato di un sistema nervoso è in grado di eseguire azioni nell'ambiente e dunque di modificare la successione di stimoli sensoriali che gli si presentano. L'ambiente possiede una struttura che determina, attraverso l'interazione con il comportamento dell'individuo, le condizioni in cui avviene l'apprendimento; in quest'ottica l'apprendimento può essere interpretato come una progressiva costruzione da parte del sistema nervoso di un modello interno dell'ambiente in cui l'organismo vive: se non vi è un ambiente, l'apprendimento non può essere altro che arbitrario e non esibirà molte delle caratteristiche dell'apprendimento biologico [*ibidem*]. Per poter tenere in debita considerazione l'influenza dell'ambiente esterno e del processo di interazione tra l'organismo e l'ambiente in cui esso si sviluppa, è necessario ricorrere a modelli di organismi artificiali con un apparato sensomotorio e un ambiente fisico in cui essi apprendono [Nolfi e Floreano 1998]. Questo approccio permette così di studiare da un punto di vista computazionale l'influenza delle caratteristiche dell'ambiente sullo sviluppo delle abilità dell'organismo e il processo di adattamento del sistema nervoso alle caratteristiche fisiche e sensomotorie dell'organismo stesso in un'ottica piagetiana.

Nei paragrafi precedenti abbiamo già descritto alcune ricerche che

fanno uso di questa metodologia combinando gli algoritmi genetici e le reti neurali per lo sviluppo di organismi artificiali che interagiscono autonomamente con un ambiente fisico. Questo nuovo approccio ecologico ha permesso di far luce sulla nascita di molte capacità complesse che spaziano dalla coordinazione sensomotoria per la navigazione alla nascita di comportamenti motivazionali fino al comportamento sociale e alla comunicazione. Beer e colleghi [Beer e Gallagher 1992] hanno utilizzato un algoritmo genetico per sviluppare e simulare la nascita di comportamenti di navigazione chemiotattica in semplici organismi di forma circolare e per l'apprendimento della capacità di locomozione in agenti con sei zampe controllati da reti neurali ricorrenti [Beer *et al.* 1992]. Secondo alcuni autori questo approccio presenta importanti risvolti applicativi in quanto – al contrario dell'approccio classico che prevede la modellizzazione a priori del robot e dell'ambiente fisico – esso permette la creazione di sistemi di controllo per robot mobili che debbano operare autonomamente in ambienti reali e non sempre prevedibili [Cliff, Harvey e Husbands 1993]. I lavori di questi autori, centrati soprattutto sull'evoluzione di comportamenti di organismi artificiali che possiedono un sistema visivo di tipo retinico, documentano l'evoluzione di apparati visivi la cui morfologia e modalità di elaborazione dell'immagine rispecchiano le proprietà dell'ambiente fisico in cui gli organismi evolvono [Cliff e Bullock 1993; Cliff, Harvey e Husbands 1992; Cliff, Husbands e Harvey 1993; Husbands, Harvey e Cliff 1993]. Floreano e Mondada [1994a] hanno impiegato un algoritmo genetico per l'evoluzione in tempo reale di reti neurali ricorrenti che controllano un piccolo robot mobile. L'intero processo evolutivo si svolge su un singolo robot collegato mediante dei cavetti al computer ove il programma genetico [Floreano 1993d] decodifica ciascuna stringa genetica in una rete neurale corrispondente e la trasferisce nel robot; il robot viene quindi lasciato libero di muoversi per alcuni minuti controllato dalla rete neurale mentre il programma valuta automaticamente la sua *fitness* in termini di velocità e assenza di collisioni. Allo scadere del tempo fissato, una nuova stringa viene decodificata e trasmessa al robot; una volta che l'intera popolazione di stringhe genetiche è stata testata sul robot, il programma applica gli operatori genetici per la costruzione della nuova popolazione. In meno di cinquanta generazioni (equivalenti a una decina di ore di funzionamento) l'algoritmo genetico evolve reti neurali in grado di esibire comportamenti molto efficienti di navigazione ed evitamento di ostacoli. Quando lo stesso robot controllato dallo stesso algoritmo genetico e dalla stessa funzione di *fitness* viene collocato in un ambiente più complesso con gradienti luminosi e un caricatore automatico di batterie, il processo evolutivo sviluppa comportamenti ben più complessi che permettono al robot di localizzare autonomamente la sorgente di energia e ritornarvici quando necessario [Floreano e Mondada 1996]. L'analisi dell'attivazione dei nodi interni evidenzia la formazione di una mappa dell'ambiente funzionalmente simile a quella generata da

alcuni neuroni dell'ippocampo dei ratti impegnati in compiti di navigazione e orientamento [O'Keefe e Nadel 1978; Taube, Muller e Ranck 1990]. Strutture neurali artificiali simili sono emerse dall'analisi del comportamento di organismi simulati che hanno appreso geneticamente a raggiungere zone dell'ambiente nascoste facendo uso di punti di riferimento distribuiti nell'ambiente [Treves, Miglino e Parisi 1992] e in organismi che hanno appreso a raccogliere del cibo e a riportarlo in un nido [Floreano 1993c]. Un aspetto interessante di queste ricerche è l'evoluzione autonoma di soluzioni neurali che rispecchiano il funzionamento di neuroni biologici di organismi impegnati in situazioni simili, a prescindere dal fatto che l'organismo artificiale sia simulato o reale [Miglino, Nafasi e Taylor 1995] (si veda Nolfi *et al.* [1994] per una rassegna delle diverse metodologie).

Infine, alcuni autori hanno studiato l'emergere di comportamenti collettivi e sociali in organismi artificiali che condividono lo stesso ambiente fisico. Anche in questo caso le diverse condizioni ambientali causano l'evoluzione di strutture neurali e comportamenti complessi a livello della popolazione [Floreano 1993a; Piazzalunga e Parisi 1993] e in alcune situazioni danno luogo alla nascita della comunicazione sociale [MacLennan 1992].

Il tema di fondo di tutte queste ricerche consiste nell'evidenziare l'importante ruolo svolto dall'interazione fra l'individuo e il suo ambiente fisico durante il processo di formazione delle capacità sensomotorie e cognitive. La possibilità di creare e studiare comportamenti emergenti che esibiscono caratteristiche di efficienza, adattamento e intelligenza – ben difficili da programmare o descrivere con sistemi di regole, ma che possiedono rilevanza ecologica e biologica – rende questo approccio molto promettente sia a livello teorico che applicativo per la creazione e comprensione di organismi intelligenti autonomi.

Appendice

Appendice

Le attivazioni delle unità di uno strato della rete, i valori sinaptici di un'unità, gli stimoli d'ingresso e le risposte desiderate sono tutte delle collezioni di numeri che possono anche essere opportunamente combinati in vettori o matrici e manipolati in modo rapido e compatto. L'algebra lineare è il ramo della matematica che si occupa dello studio di vettori e matrici e delle operazioni che possono essere compiute su di essi; per questo motivo, essa rappresenta uno strumento ottimale per l'analisi e la sintesi di modelli neurali artificiali. In questa appendice passeremo brevemente in rassegna alcune nozioni di algebra lineare che sono state utilizzate nel testo. Data la compattezza dell'esposizione, questa appendice non è un'introduzione all'algebra lineare (di cui manca il rigore e la completezza e per la quale si dovrebbe consultare un testo specifico), bensì un mezzo di «sopravvivenza» per chi non abbia mai incontrato questi concetti e uno strumento di consultazione per chi si sia dimenticato alcune definizioni.

1. Vettori

Un vettore è una collezione di n elementi

$$\mathbf{x} = \{x_1, x_2, x_3, \ldots, x_n\}$$

solitamente si considerano vettori di elementi disposti in colonna

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{pmatrix}$$

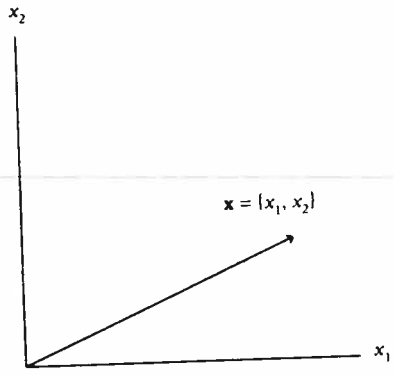


FIG. A1. Rappresentazione grafica di un vettore a due componenti nello spazio bidimensionale.

e si definisce il vettore analogo disposto su una riga come il vettore «trasposto»

$$\mathbf{x}^T = (x_1, x_2, \ldots, x_n)$$

Gli elementi di un vettore sono chiamati «componenti» e sono solitamente dei numeri reali. Un vettore può essere rappresentato come un punto o una freccia nello spazio a n dimensioni (fig. A1), dove n è il numero di componenti (o «dimensionalità») del vettore. Il vettore con tutte le componenti uguali a zero si dice «vettore nullo».

Moltiplicazione di un vettore per uno scalare

Nell'ambito dell'algebra lineare un singolo numero reale viene definito con il nome di «scalare». La moltiplicazione fra un vettore e uno scalare produce un vettore con la stessa dimensionalità del vettore originario in cui ciascuna delle componenti è il prodotto fra l'elemento corrispondente del vettore originario e lo scalare dato

$$\mathbf{x} = \begin{pmatrix} 1 \\ -3 \\ 4,5 \end{pmatrix}, \quad 2\mathbf{x} = \begin{pmatrix} 1 \cdot 2 \\ -3 \cdot 2 \\ 4,5 \cdot 2 \end{pmatrix} = \begin{pmatrix} 2 \\ -6 \\ 9 \end{pmatrix}$$

Geometricamente parlando, la moltiplicazione tra un vettore e uno scalare corrisponde all'allungamento o accorciamento del vettore il quale continua però a puntare sempre nella stessa direzione, eventualmente con verso opposto se lo scalare è negativo (fig. A2). Due vettori **a** e **b** che sono multipli scalari l'uno dell'altro (ad esempio, $\mathbf{b} = 3\mathbf{a}$) sono detti «collineari».

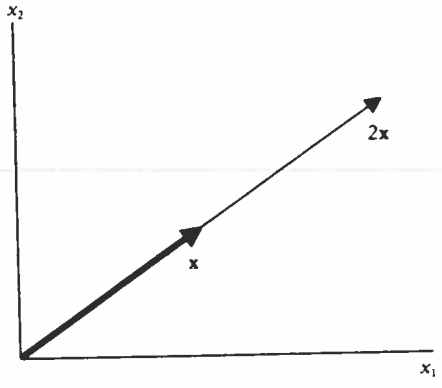


FIG. A2. Moltiplicazione di un vettore per uno scalare.

Addizione di vettori

L'addizione tra due vettori aventi la stessa dimensionalità si ottiene sommando le componenti corrispondenti:

$$\mathbf{v}_1 = \begin{pmatrix} 1 \\ 3 \end{pmatrix} \quad \text{e} \quad \mathbf{v}_2 = \begin{pmatrix} 3 \\ 1 \end{pmatrix}, \qquad \mathbf{v}_1 + \mathbf{v}_2 = \begin{pmatrix} 1+3 \\ 3+1 \end{pmatrix} = \begin{pmatrix} 4 \\ 4 \end{pmatrix}$$

Da un punto di vista geometrico la somma di due vettori corrisponde alla diagonale del parallelogramma con due lati individuati dai due vettori (fig. A3).

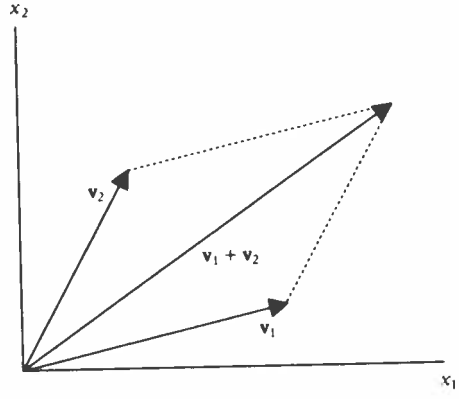


FIG. A3. Addizione di vettori.

Prodotto interno o scalare

Il prodotto interno è un tipo di moltiplicazione fra due vettori con la stessa dimensionalità che dà luogo a uno scalare

$$\mathbf{x} \cdot \mathbf{w} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ \dots \\ w_n \end{pmatrix} = x_1 w_1 + x_2 w_2 + \dots + x_n w_n = \sum_{i=1}^{n} x_i w_i$$

Il prodotto interno è alla base di molte operazioni impiegate nei modelli neurali: il potenziale di attivazione di un neurone artificiale è il prodotto interno tra il vettore di ingresso e il vettore dei pesi sinaptici.

Il prodotto interno permette di definire la lunghezza, o «norma euclidea», di un vettore

$$\|\mathbf{x}\| = \sqrt{\mathbf{x} \cdot \mathbf{x}} = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2}$$

La «normalizzazione» di un vettore non-nullo si ottiene dividendo il vettore per la propria norma (fig. A4).

$$\frac{\mathbf{x}}{\|\mathbf{x}\|}$$

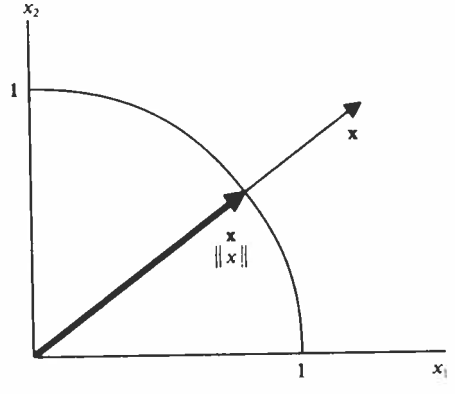


FIG. A4. Normalizzazione di un vettore.

Relazioni fra vettori

Dati due vettori **a** e **b**, risulta spesso utile avere una misura della loro distanza (fig. A5), ad esempio tra il vettore che rappresenta lo stato di attivazione delle unità di output di una rete e il vettore di risposta desiderata, oppure la distanza tra i pesi sinaptici di un'unità e i pattern di input. Una misura adeguata è la «distanza euclidea», ovvero la norma euclidea della loro differenza

$$\|\mathbf{a} - \mathbf{b}\| = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2}$$

È facilmente dimostrabile che per vettori binari ($a_i, b_i \in \{1,0\}$), il quadrato della distanza euclidea corrisponde alla «distanza di Hamming» $\rho_H(\mathbf{a},\mathbf{b})$: quest'ultima è data dal numero di componenti corrispondenti che sono diverse tra i due vettori.

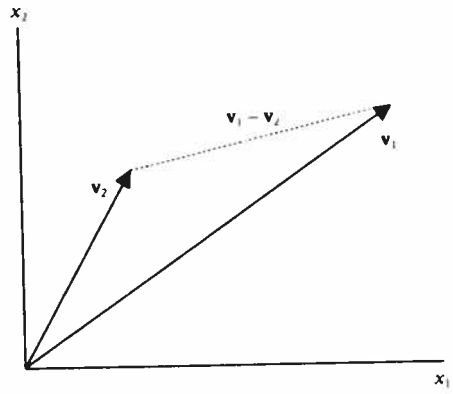


FIG. A5. Distanza tra due vettori.

La «diseguaglianza di Cauchy-Schwartz» dice che il valore assoluto del prodotto interno di due vettori è minore o eguale al prodotto tra le norme dei due vettori

$$|\mathbf{a} \cdot \mathbf{b}| \leq \|\mathbf{a}\| \cdot \|\mathbf{b}\|$$

Possiamo quindi definire l'«angolo» ϑ formato dai due vettori ponendo

$$\cos \vartheta = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \cdot \|\mathbf{b}\|}, \quad 0 \leq \vartheta \leq \pi$$

e possiamo ridefinire ora il prodotto interno tra due vettori

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \cdot \|\mathbf{b}\| \cos \vartheta$$

Questo risultato significa che se manteniamo costante la lunghezza dei vettori, il prodotto interno sarà proporzionale al coseno dell'angolo tra i due vettori: date le proprietà del coseno, il prodotto interno sarà tanto maggiore quanto più piccolo è l'angolo formato da essi (si veda il paragrafo 2.6 del primo capitolo per una discussione più approfondita di questo argomento in relazione ai neuroni artificiali). Ricordiamoci però che un valore elevato del prodotto interno non significa necessariamente che due vettori abbiano un angolo più piccolo fra di essi rispetto ad altri due vettori, a meno che i vettori non siano stati normalizzati (questa osservazione è rilevante per la fase di selezione dell'unità vincente nell'apprendimento competitivo).

Separabilità lineare

Un insieme di vettori con la stessa dimensionalità è linearmente separabile in due classi se è possibile individuare un iperpiano nello spazio vettoriale che separi le due classi di vettori. Ad esempio, la funzione logica AND divide i vettori di ingresso in due classi linearmente separabili il che non accade per la funzione XOR (si veda il primo capitolo). Vi sono alcuni metodi per determinare se un insieme di vettori sono linearmente separabili, ma il metodo più generale è la regola di apprendimento del percettrone: se la rete non riesce a distinguere i pattern in due classi distinte, allora essi non sono linearmente separabili.

Indipendenza lineare

Un insieme di vettori non nulli $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_n$ si dice linearmente dipendente se uno qualsiasi dei vettori è esprimibile come una combinazione lineare degli altri vettori. Per sapere se un insieme di vettori è linearmente indipendente è sufficiente risolvere l'equazione

$$a_1\mathbf{x}_1 + a_2\mathbf{x}_2 + a_3\mathbf{x}_3 \ldots + a_n\mathbf{x}_n = 0$$

ove a_n sono valori scalari; se l'unica soluzione è data dalla condizione

$$a_1 = a_2 = a_3 \ldots = a_n = 0$$

allora i vettori sono linearmente indipendenti. Altrimenti, se esiste una soluzione in cui almeno uno degli scalari è diverso da zero, i vettori sono linearmente dipendenti. È possibile costruire insiemi di vettori linearmente indipendenti di dimensionalità n aventi al più n elementi. Nel caso delle reti neurali, ogni qualvolta il numero di pattern da apprendere è maggiore del numero di unità d'ingresso, i pattern sono li-

nearmente dipendenti (si veda ad esempio la funzione AND). Per questo motivo la maggior parte dei problemi interessanti che vengono affrontati con le reti neurali non possiedono caratteristiche di indipendenza lineare.

2. Matrici

Una matrice è una collezione di numeri reali composta di m righe ed n colonne

$$\mathbf{M} = \begin{bmatrix} 1 & 5 & 6 \\ -2 & 0 & 3 \end{bmatrix}$$

i cui componenti si indicano usando il doppio indice m_{ij}, dove i sono le righe e j sono le colonne

$$\mathbf{M} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \end{bmatrix}$$

Moltiplicazione di una matrice per uno scalare

Una matrice può essere moltiplicata per uno scalare moltiplicando ogni singolo elemento per lo scalare

$$2\mathbf{M} = \begin{bmatrix} 1 \cdot 2 & 5 \cdot 2 & 6 \cdot 2 \\ -2 \cdot 2 & 0 \cdot 2 & 3 \cdot 2 \end{bmatrix} = \begin{bmatrix} 2 & 10 & 12 \\ -4 & 0 & 6 \end{bmatrix}$$

Addizione di matrici

Due matrici possono essere sommate solo se possiedono entrambe lo stesso numero di righe e di colonne addizionando gli elementi corrispondenti

$$\mathbf{M} + \mathbf{N} = \begin{bmatrix} 1 & 5 & 6 \\ -2 & 0 & 3 \end{bmatrix} + \begin{bmatrix} 3 & -4 & 0 \\ -1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1+3 & 5-4 & 6+0 \\ -2-1 & 0+1 & 3+1 \end{bmatrix} = \begin{bmatrix} 4 & 1 & 6 \\ -3 & 1 & 4 \end{bmatrix}$$

Moltiplicazione di una matrice per un vettore

Data una matrice

$$\mathbf{W} = \begin{bmatrix} 0,6 & 0,5 & -0,1 \\ -1 & 0,3 & 1 \end{bmatrix}$$

e un vettore

$$\mathbf{x} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

con lo stesso numero di componenti del numero di colonne della matrice **W**, definiamo un nuovo vettore **y** che sia il prodotto **Wx**

$$\mathbf{Wx} = \begin{bmatrix} 0,6 & 0,5 & -0,1 \\ -1 & 0,3 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} (0,6 \cdot 1) + (0,5 \cdot 0) + (-0,1 \cdot -1) \\ (-1 \cdot 1) + (0,3 \cdot 0) + (1 \cdot -1) \end{bmatrix} = \begin{bmatrix} 0,7 \\ -2 \end{bmatrix}$$

tale per cui il numero di componenti di **y** è uguale al numero di righe della matrice **W** e ciascun componente è il prodotto interno tra una riga della matrice **W** e il vettore **x**.

Si noti che questo prodotto permette di calcolare in un solo passo l'attivazione di un intero strato di nodi di una rete neurale. Il vettore **x** rappresenta il pattern di input della rete e la matrice **W** rappresenta un intero strato di pesi sinaptici, dove ciascuna riga raccoglie tutti i pesi di ciascuna unità di output; il vettore **y**, che possiede tanti elementi quante sono le righe della matrice, contiene i valori di attivazione dello strato di unità di output.

Moltiplicazione di matrici

Il prodotto fra due matrici $\mathbf{A} \cdot \mathbf{B}$ è valido solo se il numero di colonne della matrice **A** è uguale al numero di righe della matrice **B**; in tal caso l'elemento *ij* della matrice prodotto è il prodotto interno tra il vettore di riga *i* della matrice **A** e il vettore di colonna *j* della matrice **B**. Consideriamo il seguente esempio

$$\mathbf{A} \cdot \mathbf{B} = \begin{bmatrix} 3 & 1 & -1 \\ 2 & 0 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 5 \\ 3 & -1 \\ -2 & 2 \end{bmatrix} =$$

$$= \begin{bmatrix} [(3 \cdot 1) + (1 \cdot 3) + (-1 \cdot -2)] & [(3 \cdot 5) + (1 \cdot -1) + (-1 \cdot 2)] \\ [(2 \cdot 1) + (0 \cdot 3) + (4 \cdot -2)] & [(2 \cdot 5) + (0 \cdot -1) + (4 \cdot 2)] \end{bmatrix} = \begin{bmatrix} 8 & 12 \\ -6 & 18 \end{bmatrix}$$

La moltiplicazione di matrici non è commutativa, ovvero, in generale, $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{B} \cdot \mathbf{A}$ anche se entrambi i termini sono definiti.

Matrice unitaria

La matrice unitaria **I** è una matrice quadrata (stesso numero di righe e colonne) in cui ciascun componente *ij* è dato dal delta di Kronecker $\delta(ij)$

$$\delta(ij) = \begin{cases} 0 & \text{se } i \neq j \\ 1 & \text{se } i = j \end{cases}$$

per cui

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Il prodotto tra una qualsiasi matrice **A** e la matrice unitaria **I** corrispondente restituisce la matrice originaria. Ad esempio

$$\mathbf{A} \cdot \mathbf{I} = \mathbf{I} \cdot \mathbf{A} = \mathbf{A}$$

Matrice trasposta

Data una matrice **A** con *m* righe e *n* colonne si definisce matrice trasposta $\mathbf{A}^T$ la matrice con *n* righe e *m* colonne tale che il suo elemento *ij* è ugualmente all'elemento *ji* di **A**. Ad esempio se

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

allora la sua trasposta è

$$\mathbf{A}^T = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

Dalla definizione segue che è sempre possibile calcolare entrambi i prodotti $\mathbf{A}\mathbf{A}^T$ e $\mathbf{A}^T\mathbf{A}$, ma si ricordi che i risultati sono in genere diversi.

Metodo di eliminazione di Gauss

Il metodo di eliminazione di Gauss permette di eseguire una serie di «operazioni elementari di righe» su di una matrice per ridurla a una

matrice equivalente che possiede un numero crescente di zero iniziali sulle righe

$$\begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \dots & \dots & \dots & \dots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & * & * & * \\ 0 & 1 & * & * \\ 0 & 0 & 1 & * \\ 0 & 0 & 0 & * \end{bmatrix}$$

dove il simbolo * indica un qualsiasi valore ottenuto dalle operazioni. Vi sono solo tre tipi di operazioni elementari che devono essere applicate a tutti gli elementi di una riga R_i:

1. moltiplicazione di una riga per uno scalare diverso da 0, ad esempio

$$R_i \Rightarrow 3R_i$$

2. scambio di righe, ad esempio

$$R_1 \Leftrightarrow R_2$$

3. addizione del multiplo di una riga a un'altra riga, ad esempio

$$R_1 \Rightarrow R_1 + 5R_2$$

Queste operazioni possono essere applicate in sequenza fino a quando la matrice non può essere ridotta ulteriormente: il numero di righe non nulle ottenuto si chiama il «rango» della matrice e indica il numero di vettori di riga linearmente indipendenti; esiste anche una versione analoga in cui si opera sulle colonne, invece che sulle righe (il rango si calcola contando il numero di colonne non nulle ottenuto e produce lo stesso risultato numerico). Il metodo di eliminazione di Gauss possiede un gran numero di applicazioni, tra cui la soluzione di un sistema di equazioni lineari; esso verrà richiamato più sotto per ricavare alcuni importanti risultati.

Inversione di matrice

Data una matrice quadrata **A** composta di n righe ed n colonne è talvolta possibile trovare la sua matrice inversa $\mathbf{A}^{-1}$ tale che

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$$

dove **I** è la corrispondente matrice d'identità. La matrice inversa esiste solamente se **A** possiede rango n, ovvero se è composta di vettori linearmente indipendenti: in tal caso la matrice si dice «non-singolare»

ed è invertibile. Il procedimento consiste nell'applicare il metodo di eliminazione di Gauss contemporaneamente sia alla matrice **A** che alla matrice **I** fino a quando la matrice **A** diventa una matrice d'identità. Consideriamo il seguente esempio

$$\mathbf{A} = \begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & -8 \\ -4 & 1 & 1 \end{bmatrix} \quad \text{e} \quad \mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Uniamo ora le due matrici operando su **A** ed effettuando le stesse operazioni su **I**; per ogni passaggio vengono specificate le operazioni elementari di riga applicate

$$\left[\begin{array}{ccc|ccc} 2 & 2 & 2 & 1 & 0 & 0 \\ 2 & 2 & -8 & 0 & 1 & 0 \\ -4 & 1 & 1 & 0 & 0 & 1 \end{array}\right] \xrightarrow[R_3 \Rightarrow R_3 + 2R_1]{R_2 \Rightarrow R_2 - R_1} \left[\begin{array}{ccc|ccc} 2 & 2 & 2 & 1 & 0 & 0 \\ 0 & 0 & -10 & -1 & 1 & 0 \\ 0 & 5 & 5 & 2 & 0 & 1 \end{array}\right]$$

$$\xrightarrow{R_2 \Leftrightarrow R_3} \left[\begin{array}{ccc|ccc} 2 & 2 & 2 & 1 & 0 & 0 \\ 0 & 5 & 5 & 2 & 0 & 1 \\ 0 & 0 & -10 & -1 & 1 & 0 \end{array}\right] \xrightarrow[R_1 \Rightarrow 5R_1 + R_3]{R_2 \Rightarrow 2R_2 + R_3} \left[\begin{array}{ccc|ccc} 10 & 10 & 0 & 4 & 1 & 0 \\ 0 & 10 & 0 & 3 & 1 & 2 \\ 0 & 0 & -10 & -1 & 1 & 0 \end{array}\right]$$

$$\xrightarrow{R_1 \Rightarrow R_1 - R_2} \left[\begin{array}{ccc|ccc} 10 & 0 & 0 & 1 & 0 & -2 \\ 0 & 10 & 0 & 3 & 1 & 2 \\ 0 & 0 & -10 & -1 & 1 & 0 \end{array}\right] \xrightarrow{R_3 \Rightarrow -R_3} \left[\begin{array}{ccc|ccc} 10 & 0 & 0 & 1 & 0 & -2 \\ 0 & 10 & 0 & 3 & 1 & 2 \\ 0 & 0 & 10 & 1 & -1 & 0 \end{array}\right]$$

$$\xrightarrow[R_3 \Rightarrow 1/10 R_3]{\substack{R_1 \Rightarrow 1/10 R_1 \\ R_2 \Rightarrow 1/10 R_2}} \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 1/10 & 0 & -2/10 \\ 0 & 1 & 0 & 3/10 & 1/10 & 2/10 \\ 0 & 0 & 1 & 1/10 & -1/10 & 0 \end{array}\right]$$

$$\mathbf{A}^{-1} = \frac{1}{10} \begin{bmatrix} 1 & 0 & -2 \\ 3 & 1 & 2 \\ 1 & -1 & 0 \end{bmatrix}$$

Calcolo dei pesi sinaptici con il metodo della pseudoinversione

In questa sezione prendiamo in considerazione un metodo che permette di individuare in un solo passo i pesi sinaptici di una rete neurale. Dato un gruppo di vettori di input $\mathbf{x}_\mu$ e un gruppo di risposte desiderate $\mathbf{t}_\mu$, il problema consiste nel trovare la matrice di pesi sinaptici **W** che minimizza la distanza euclidea tra i vettori di risposta della rete e i corrispondenti vettori di risposta desiderata $\mathbf{t}_\mu$. Possiamo radunare tutti i vettori di input e di risposta desiderata in due matrici

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1\mu} \\ x_{21} & x_{22} & \dots & x_{2\mu} \\ \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & \dots & x_{n\mu} \end{bmatrix} \quad \text{e} \quad \mathbf{T} = \begin{bmatrix} t_{11} & t_{12} & \dots & t_{1\mu} \\ t_{21} & t_{22} & \dots & t_{2\mu} \\ \dots & \dots & \dots & \dots \\ t_{n1} & t_{n2} & \dots & t_{n\mu} \end{bmatrix}$$

dove le righe indicano le unità della rete. Ora si tratta di trovare la matrice $\mathbf{W}$ che minimizza la distanza euclidea tra la matrice di risposte desiderate $\mathbf{T}$ e la matrice di risposte fornite dalla rete $\mathbf{WX}$ (le attivazioni pesate ottenute per ciascun pattern di input); formalmente, vogliamo trovare la matrice ottimale $\hat{\mathbf{W}}$ che minimizzi la distanza euclidea $\|\mathbf{T} - \hat{\mathbf{W}}\mathbf{X}\|$. Tale matrice è data da

$$\hat{\mathbf{W}} = \mathbf{T}\mathbf{X}^{+}$$

dove $\mathbf{X}^{+}$ è la «matrice pseudoinversa» della matrice $\mathbf{X}$ dei vettori di input. Tale matrice soddisfa la relazione $\mathbf{X}^{+}\mathbf{X} = \mathbf{I}$, dove $\mathbf{I}$ è la matrice identità; risulta quindi

$$\hat{\mathbf{W}}\mathbf{X} = \mathbf{T}\mathbf{X}^{+}\mathbf{X}$$
$$\hat{\mathbf{W}}\mathbf{X} = \mathbf{T}$$

ovvero la matrice $\hat{\mathbf{W}}$ produce vettori di risposta desiderata per ciascun vettore di input della rete. Ora si noti che la matrice pseudoinversa è calcolabile solo quando le colonne della matrice $\mathbf{X}$ sono dei vettori linearmente indipendenti, la qual cosa implica che il numero di unità di input della rete sia maggiore o uguale al numero di vettori di input (condizione necessaria, ma non sufficiente); se la condizione di indipendenza è vera, la matrice pseudoinversa è data da

$$\mathbf{X}^{+} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}$$

Il metodo della matrice pseudoinversa viene applicato nel secondo capitolo per i percettroni e nel terzo capitolo per la rete di Hopfield.

Determinante di una matrice

Data una matrice quadrata $\mathbf{A}$ avente n righe ed n colonne è possibile definire il suo determinante. Per fare subito un esempio concreto,

se

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

allora

$$\det \mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = (a_{11}a_{22}) - (a_{12}a_{21})$$

(Esiste anche un semplice metodo per calcolare il determinante di matrici con un numero di righe e colonne superiore a 2.) Il determinante fornisce alcune importanti informazioni sulla matrice:

1. se i vettori di riga – o di colonna – della matrice sono linearmente dipendenti, il determinante è zero;
2. se due righe – o colonne – qualsiasi vengono scambiate, il determinante cambia di segno;
3. aggiungendo a una qualsiasi riga – o colonna – un multiplo di un'altra riga – o colonna rispettivamente –, il determinante rimane immutato.

Da queste proprietà è semplice inferire che se una matrice possiede una riga o una colonna nulla, allora il determinante della matrice è zero (perché il vettore nullo è sempre linearmente dipendente dagli altri vettori). Inoltre, si noti che se il determinante di una matrice è diverso da zero, la matrice è invertibile.

Autovettori e autovalori

Data una matrice quadrata **A** dotata di n righe ed n colonne, vi sono alcuni vettori (di dimensionalità n) che vengono trasformati da **A** in multipli scalari di se stessi (fig. A6). Questi vettori sono detti «autovettori».

La definizione di autovettore è la seguente: se **A** è una matrice quadrata n per n, allora un vettore non nullo **x** di n componenti è detto autovettore di **A** se il prodotto **Ax** è un multiplo scalare di **x**, ovvero

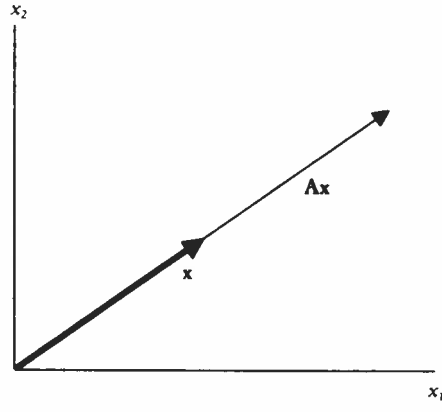


FIG. A6. Trasformazione subita da un autovettore quando moltiplicato per la sua matrice.

$$\mathbf{Ax} = \lambda\mathbf{x}$$

per un determinato scalare λ: lo scalare λ è detto «autovalore» di **A** e si dice che **x** è l'autovettore corrispondente all'autovalore λ. Se $\mathbf{Ax} = \lambda\mathbf{x}$, allora la moltiplicazione di **x** per **A** dilata, contrae o rovescia la direzione di **x** in base al valore di λ. Ora, siccome

$$\mathbf{Ax} = \lambda\mathbf{x} = \lambda\mathbf{Ix}$$

possiamo riscrivere

$$(\lambda\mathbf{I} - \mathbf{A})\mathbf{x} = 0$$

Siccome la matrice $(\lambda\mathbf{I} - \mathbf{A})$ non è invertibile (perché **x** è una soluzione non nulla di questo sistema di equazioni), allora il suo determinante è zero

$$\det(\lambda\mathbf{I} - \mathbf{A}) = 0$$

Questa viene definita l'«equazione caratteristica» di **A** e l'espansione del determinante viene definito il «polinomio caratteristico» di **A**. Il polinomio caratteristico di **A** è di grado n e gli autovalori sono le sue N radici (alcune delle quali possono avere lo stesso valore: in tal caso si dice che l'autovalore è multiplo).

Vediamo in un esempio concreto come individuare gli autovalori e gli autovettori di una semplice matrice $\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 6 & 2 \end{bmatrix}$. Risolviamo innanzitutto l'equazione caratteristica

$$\det(\lambda\mathbf{I} - \mathbf{A}) = 0$$

$$\det\begin{vmatrix} \lambda - 1 & -2 \\ -6 & \lambda - 2 \end{vmatrix} = 0$$

$$(\lambda^2 - 2\lambda - \lambda + 2) - (12) = 0$$

$$\lambda^2 - 3\lambda - 10 = 0$$

e impiegando la formula risolutiva delle equazioni di secondo grado otteniamo i due autovalori

$$\lambda = \frac{3 \pm \sqrt{9 + 40}}{2} \Rightarrow \lambda_1 = 5, \quad \lambda_2 = -2$$

Ora troviamo i due autovettori corrispondenti sostituendo i valori così ottenuti nell'equazione $(\lambda\mathbf{I} - \mathbf{A})\mathbf{x} = 0$; iniziamo con $\lambda = 5$

$$\begin{bmatrix} \lambda - 1 & -2 \\ -6 & \lambda - 2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

$$\begin{bmatrix} 4 & -2 \\ -6 & 3 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

$$\begin{bmatrix} 4x_1 & -2x_2 \\ -6x_1 & +3x_2 \end{bmatrix} = 0$$

otteniamo $x_2 = 2x_1$, il che significa che ogni vettore $\mathbf{x} = \begin{bmatrix} t \\ 2t \end{bmatrix}$ con $t \neq 0$ (ad esempio il vettore $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$) è un autovettore corrispondente all'autovalore $\lambda = 5$.

Troviamo ora il secondo autovettore per $\lambda = -2$ usando lo stesso procedimento

$$\begin{bmatrix} \lambda - 1 & -2 \\ -6 & \lambda - 2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

$$\begin{bmatrix} -3 & -2 \\ -6 & -4 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

$$\begin{bmatrix} -3x_1 & -2x_2 \\ -6x_1 & -4x_2 \end{bmatrix} = 0$$

otteniamo $x_2 = -3/2x_1$, per cui l'autovettore corrispondente ha la forma $\mathbf{x} = \begin{bmatrix} t \\ -3/2t \end{bmatrix}$ con $t \neq 0$, ad esempio $\begin{bmatrix} 1 \\ -3/2 \end{bmatrix}$.

Prodotto tensoriale di vettori

Il prodotto tensoriale tra il vettore **x** con m componenti e il vettore **y** con n componenti è una matrice di m righe ed n colonne composta dai prodotti di tutti gli elementi del vettore **x** con tutti gli elementi del vettore **y**. Il prodotto esterno si indica con $\mathbf{x} \otimes \mathbf{y}$ oppure $\mathbf{x}\mathbf{y}^T$.

$$\mathbf{x}\mathbf{y}^T = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} [y_1 \; y_2 \; y_3 \; y_4 \; y_5] = \begin{bmatrix} x_1y_1 & x_1y_2 & x_1y_3 & x_1y_4 & x_1y_5 \\ x_2y_1 & x_2y_2 & x_2y_3 & x_2y_4 & x_2y_5 \\ x_3y_1 & x_3y_2 & x_3y_3 & x_3y_4 & x_3y_5 \\ x_4y_1 & x_4y_2 & x_4y_3 & x_4y_4 & x_4y_5 \end{bmatrix}$$

Il prodotto tensoriale viene impiegato per calcolare la matrice di

pesi sinaptici della rete di Hopfield (secondo capitolo). Vi sono le seguenti importanti differenze tra prodotto interno e prodotto tensoriale tra due vettori:

– prodotto interno: vale solo per vettori della stessa dimensionalità, il risultato è uno scalare, gode della proprietà commutativa;

– prodotto tensoriale: vale anche per vettori di dimensionalità diverse, il risultato è una matrice, non gode della proprietà commutativa.

Riferimenti bibliografici

Abeles, M.
1991 *Corticonics*, Cambridge, Cambridge University Press.
Ackley, D.H., Hinton, G.E. e Sejnowski, T.J.
1985 *A Learning Algorithm for Boltzmann Machines*, in «Cognitive Science», vol. 9, pp. 147-169.
Ackley, D.H. e Littman, M.L.
1990 *Generalization and Scaling in Reinforcement Learning*, in *Advances in Neural Information Processing Systems II*, a cura di D.S. Touretzky, San Mateo, CA, Morgan Kaufmann.
1992 *Interactions between Learning and Evolution*, in *Artificial Life II: Proceedings Volume of Santa Fe Conference*, a cura di C.G. Langton, J.D. Farmer, S. Rasmussen e C. Taylor, Reading, MA, Addison Wesley.
Amit, D., Gutfreund, H. e Sompolinsky, H.
1985a *Spin-Glass Models of Neural Networks*, in «Physical Review A», vol. 32, pp. 1007-1018.
1985b *Storing Infinite Numbers of Patterns in a Spin-Glass Model of Neural Networks*, in «Physical Review Letters», vol. 55, pp. 1530-1533.
Anderson, C.W.
1987 *Strategy Learning with Multilayer Connectionist Representations*, in *Proceedings of the Fourth International Workshop on Machine Learning*, Irvine, CA, Morgan Kaufmann, pp. 103-114.
Anderson, J.A.
1986 *Cognitive Capabilities of a Parallel System,* in *Disordered Systems and Biological Organization*, a cura di E. Bienenstock, F. Fogelman-Souli e G. Weisbuch, Berlin, Springer-Verlag, NATO-ASI Series.
1995 *An Introduction to Neural Networks*, Cambridge, MA, MIT Press-Bradford Books.
Anderson, J.A., Pellionisz, A. e Rosenfeld, E. (a cura di)
1990 *Neurocomputing 2: Directions for Research*, Cambridge, MA, MIT Press-Bradford Books.

Anderson, J.A. e Rosenfeld, E. (a cura di)
1988 *Neurocomputing: Foundations of Research*, Cambridge, MA, MIT Press-Bradford Books.
1998 *Talking Nets: An Oral History of Neural Networks*, Cambridge, MA, MIT Press-Bradford Books.

Anderson, J.A., Silverstein, J.W., Ritz, S.A. e Jones, R.S.
1977 *Distinctive Features, Categorical Perception, and Probability Learning: Some Applications of a Neural Model*, in «Psychological Review», vol. 84, pp. 413-451.

Apolloni, B., Avanzini, G., Cesa-Bianchi, N. e Ronchini, G.
1990 *Diagnosis of Epilepsy via Backpropagation*, in *Proceedings of the International Joint Conference on Neural Networks*, San Diego, CA, vol. II, pp. 471-474.

Atick, J.J. e Redlich, A.N.
1990 *Towards a Theory of Early Visual Processing*, in «Neural Computation», vol. 2, pp. 308-320.

Baddeley, R.J. e Hancock, P.J.
1991 *A Statistical Analysis of Natural Images Matches Psychophysically Derived Orientation Tuning Curves*, in «Proceedings of the Royal Society London B», vol. 246, pp. 219-223.

Baker, J.E.
1987 *Reducing Bias and Inefficiency in the Selection Algorithm*, in *Proceedings of the Second International Conference on Genetic Algorithms*, a cura di J.J. Grefenstette, Hillsdale, NJ, Lawrence-Erlbaum Associates, pp. 14-21.

Baldwin, J.M.
1896 *A New Factor in Evolution*, in «American Naturalist», vol. 30, pp. 441-451, 536-553.

Barlow, H.B.
1961 *Current Problems in Animal Behavior*, Cambridge, Cambridge University Press.
1972 *Single Units and Sensation: A Neuron Doctrine for Perceptual Psychology?*, in «Perception», vol. 1, pp. 371-394.
1989 *Unsupervised Learning*, in «Neural Computation», vol. 1, pp. 295-311.

Bartlett, P.L. e Anthony, M.M.
1999 *Neural Network Learning: Theoretical Foundations*, Cambridge, Cambridge University Press.

Barto, A.G. e Anandan, P.
1985 *Pattern Recognizing Stochastic Learning Automata*, in «IEEE Transactions on Systems, Man and Cybernetics», vol. 15, pp. 360-375.

Barto, A.G. e Jordan, M.J.
1987 *Gradient Following without Propagation in Layered Networks*, in *Proceedings of the IEEE First International Conference on Neural Networks*, a cura di M. Caudill e C. Butler, New York, IEEE Press, vol. II, pp. 629-636.

Barto, A.G., Sutton, R.S. e Anderson, C.W.
1983 *Neuronlike Adaptive Elements that can Solve Difficult Learning Control Problems*, in «IEEE Transactions on Systems, Man and Cybernetics», vol. 13, pp. 834-846.
Barto, A.G., Sutton, R.S. e Watkins, C.J.C.H.
1990 *Learning and Sequential Decision Making*, in *Learning and Computational Neuroscience*, a cura di M. Gabriel e J.W. Moore, Cambridge, MA, MIT Press.
Baum, E.B. e Haussler, D.
1989 *What Size Net Gives Valid Generalization?*, in «Neural Computation», vol. 1, pp. 151-160.
Baxter, J.
1994 *The Evolution of Learning Algorithms for Artificial Neural Networks*, rapporto tecnico, School of Information Science and Technology, The Flinders University of South Australia.
Becker, S.
1991 *Unsupervised Learning Procedures for Neural Networks*, in «International Journal of Neural Systems», vol. 2, pp. 17-33.
Becker, S. e Hinton, G.E.
1992 *Self-Organizing Neural Network that Discovers Surfaces in Random-Dot Stereograms*, in «Nature», vol. 355, pp. 161-163.
Becker, S. e Plumbley, M.
1993 *Unsupervised Neural Network Learning Procedure for Feature Extraction and Classification*, in «International Journal of Applied Intelligence», vol. 1, pp. 3-25.
Beer, R.D.
1990 *Intelligence as Adaptive Behavior: An Experiment in Computational Neuroethology*, San Diego, CA, Academic Press.
1995 *On the Dynamics of Continuous-Time Recurrent Neural Networks*, in «Adaptive Behavior», vol. 3, pp. 469-509.
Beer, R.D. e Gallagher, J.C.
1992 *Evolving Dynamical Neural Networks for Adaptive Behavior*, in «Adaptive Behavior», vol. 1, pp. 91-122.
Beer, R.D., Chiel, H.J., Quinn, R.D., Espenschied, K.S. e Larsson, P.
1992 *A Distributed Neural Network Architecture for Hexapod Robot Locomotion*, in «Neural Computation», vol. 4, pp. 356-365.
Belew, R.K.
1991 *Artificial Life. A Constructive Lower Bound for Artificial Intelligence*, in «IEEE Expert», febbraio, pp. 8-15.
Belew, R.K., McIrney, J. e Schraudolph, N.N.
1992 *Evolving Networks: Using the Genetic Algorithm with Connectionist Learning*, in *Artificial Life II: Proceedings Volume of Santa Fe Conference*, a cura di C.G. Langton, J.D. Farmer, S. Rasmussen e C. Taylor, Reading, MA, Addison Wesley.
Bienenstock, E.L., Cooper, L.N. e Munro, P.W.
1982 *A Theory for the Development of Neuron Selectivity; Orientation Specificity and Binocular Interaction in Visual Cortex*, in «Journal

of Neuroscience», vol. 2, pp. 32-48, ristampato in Anderson e Rosenfeld [1988].

Bishop, C.M.
1995 *Neural Networks for Pattern Recognition*, Oxford, Clarendon Press.

Block, H.D.
1962 *The Perceptron: A Model for Brain Functioning, I.*, in «Review of Modern Physics», vol. 34, pp. 123-135, ristampato in Anderson e Rosenfeld [1988].

Boers, E.J.W. e Kuiper, H.
1992 *Biological Metaphors and the Design of Modular Artificial Neural Networks*, tesi di master, Computer Science, University of Leiden, The Netherlands.

Bourlard, H. e Kamp, Y.
1988 *Auto-Association by Multilayer Perceptrons and Singular Value Decomposition*, in «Biological Cybernetics», vol. 59, pp. 291-294.

Bracewell, R.N.
1986 *The Fourier Transform and Its Applications*, New York, McGraw-Hill, 2ª ed.

Braitenberg, V.
1984 *Vehicles. Experiments in Synthetic Psychology*, Cambridge, MA, MIT Press; trad. it. *Veicoli*, Milano, Garzanti, 1988.

Braitenberg, V. e Schüz, A.
1998 *Cortex: Statistics and Geometry of Neuronal Connectivity*, Berlin, Springer-Verlag, 2ª ed.

Bryson, A.E. e Ho, Y-C.
1969 *Applied Optimal Control*, New York, Blaisdell.

Burgess, N., Granieri, M.N. e Patarnello, S.
1992 *3-D Object Classification: Application of a Constructive Algorithm*, in «International Journal of Neural Systems», vol. 2, pp. 275-282.

Caianiello, E.R.
1961 *Outline of a Theory of Thought and Thinking Machines*, in «Journal of Theoretical Biology», vol. 1, pp. 204-235.

Cangelosi, A., Parisi, D. e Nolfi, S.
1994 *Cell Division and Migration in a Genotype for Neural Networks*, in «Network», vol. 5, pp. 497-515.

Carpenter, G.A. e Grossberg, S.
1987a *A Massively Parallel Architecture for Self-Organizing Neural Pattern Recognition Machine*, in «Computer Vision, Graphics, and Image Processing», vol. 37, pp. 54-115.
1987b *ART2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns*, in «Applied Optics», vol. 26, pp. 4919-4930.
1990 *ART3: Hierarchical Search Using Chemical Transmitters in Self-Organizing Pattern Recognition Architectures*, in «Neural Networks», vol. 3, pp. 129-152.

Cater, J.P.
1987 *Successfully Using Peak Learning Rates of 10 (and Greater) in*

Back-propagation Networks with the Heuristic Learning Algorithm, in *Proceedings of the IEEE First International Conference on Neural Networks*, a cura di M. Caudill e C. Butler, New York, IEEE Press, vol. II, pp. 645-652.

Chalmers, D.J.

1990 *The Evolution of Learning: An Experiment in Genetic Connectionism*, in *Proceedings of the 1990 Connectionist Models Summer School*, a cura di D.S. Touretzky, J.L. Elman, T.J. Sejnowski e G.E. Hinton, San Mateo, CA, Morgan Kufmann, pp. 81-90.

Chapman, D. e Kaelbling, L.P.

1991 *Input Generalization in Delayed Reinforcement Learning: An Algorithm and Performance Comparisons*, in *Proceedings of the International Joint Conference on Artificial Intelligence*, Washington, DC.

Chauvin, Y.

1989 *A Back-Propagation Algorithm with Optimal Use of Hidden Units*, in *Advances in Neural Information Processing Systems I*, a cura di D.S. Touretzky, San Mateo, CA, Morgan Kaufmann, pp. 519-526.

Churchland, P.M.

1989 *The Neurocomputational Perspective*, Cambridge, MA, MIT Press; trad. it. parziale *La natura della mente e la struttura della scienza*, Bologna, Il Mulino, 1992.

Churchland, P.M. e Sejnowski, T.J.

1992 *The Computational Brain*, Cambridge, MA, MIT Press; trad. it. *Il cervello computazionale*, Bologna, Il Mulino, 1995.

Clark, A.

1989 *Microcognition: Philosophy, Cognitive Science, and Parallel Distributed Processing*, Cambridge, MA, MIT Press; trad. it. *Microcognizione. Filosofia, scienza cognitiva e reti neurali*, Bologna, Il Mulino, 1994.

Cliff, D.T.

1991 *Computational Neuroethology: a Provisional Manifesto*, in *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*, a cura di J.-A. Meyer e S.W. Wilson, Cambridge, MA, MIT Press-Bradford Books.

Cliff, D.T. e Bullock, S.G.

1993 *Adding «Foveal Vision» to Wilson's Animat*, in «Adaptive Behavior», vol. 2, pp. 49-72.

Cliff, D.T., Harvey, I. e Husbands, P.

1992 *Incremental Evolution of Neural Architectures for Adaptive Behaviour*, rapporto tecnico CSRP 256, School of Cognitive and Computing Science, University of Sussex.

1993 *Explorations in Evolutionary Robotics*, in «Adaptive Behavior», vol. 2, pp. 73-110.

Cliff, D.T., Husbands, P. e Harvey, I.

1993 *Evolving Visually Guided Robots*, in *From Animals to Animats II: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, a cura di J. Meyer, H.L. Roitblat e S.W. Wilson, Cambridge, MA, MIT Press-Bradford Books.

Cohen, M.A. e Grossberg, S.
1983 *Absolute Stability of Global Pattern Formation and Parallel Memory Storage by Competitive Neural Networks*, in «IEEE Transactions on Systems, Man and Cybernetics», vol. 13, pp. 815-826.
Cohen, J.D., Dumbar, K. e McClelland, J.L.
1990 *On the Control of Automatic Processes: A Parallel Distributed Processing Account of the Stroop Effect*, in «Psychological Review», vol. 97, pp. 332-361.
Collins, E., Ghosh, S. e Scofield, C.L.
1988 *An Application of Multiple Neural Network Learning System to the Emulation of Mortgage Underwriting Judgements*, in *Proceedings of the IEEE International Conference on Neural Networks*, a cura di M. Caudill e C. Butler, New York, IEEE Press, vol. II, pp. 459-466.
Conquet, F., Bashir, Z.I., Davies, C.H., Daniel, H., Ferraguti, F., Bordi, F., Franz-Bacon, K., Reggiani, A., Matarese, V., Conde, F., Collingridge, G.L. e Crepel, F.
1994 *Motor Deficit and Impairment of Synaptic Plasticity in Mice Lacking mGluR1*, in «Nature», vol. 372, pp. 237-243.
Cook, N.D.
1995 *Artefact or Network Evolution?*, in «Nature», vol. 374, pp. 313-314.
Cook, N.D., Fruh, H. e Landis, T.
1995 *The Cerebral Hemispheres and Neural Network Simulations: Design Considerations*, in «Journal of Experimental Psychology: Human Perception and Performance», in stampa.
Cottrell, G.W., Munro, P.W. e Zipser, D.
1989 *Image Compression by Back-Propagation: A Demonstration of Extensional programming*, in *Advances in Cognitive Science*, a cura di N.E. Sharkey, Norwood, NJ, Abbex.
Couvillon, P.A. e Bitterman, M.E.
1992 *A Conventional Conditioning Analysis of «Transitive Inference» in Pigeons*, in «Journal of Experimental Psychology: Animal Behavior Processes», vol. 18, pp. 308-310.
Crick, F.
1989 *The Recent Excitement About Neural Networks*, in «Nature», vol. 337, pp. 129-132.
1994 *The Astonishing Hypothesis. The Scientific Search for the Soul*, London, Simon and Schuster.
Crick, F. e Mitchison, G.
1983 *The Function of Dream Sleep*, in «Nature», vol. 304, pp. 111-114.

Dasdan, A. e Oflazer, K.
1994 *Genetic Synthesis of Unsupervised Learning Algorithms*, rapporto tecnico, Department of Computer Engineering and Information Science, Bilkent University of Ankara, Turkey.
Davis, L.
1987 *Genetic Algorithms and Simulated Annealing*, London, Pitman.

1991 *Handbook of Genetic Algorithms*, New York, Van Nostrand Reinhold.

Dawkins, R.
1986 *The Blind Watchmaker*, London, Penguin Books; trad. it. *L'orologiaio cieco*, Milano, Rizzoli, 1988.

DeRouin, E., Brown, J., Beck, H., Fausett, L. e Schneider, M.
1991 *Neural Network Training and Unequally Represented Classes*, in *Intelligent Engineering Systems Through Artificial Neural Networks*, a cura di C.H. Dagli, S.R.T. Kumara e Y.C. Shi, ASME Press, pp. 135-141.

DeSieno, D.
1988 *Adding a Conscience to Competitive Learning*, in *Proceedings of the IEEE International Conference on Neural Networks*, a cura di M. Caudill e C. Butler, New York, IEEE Press.

Dewdney, A.K.
1993 *Misled by Metaphors: Two Tools that Don't Always Work*, in *The Machine as Metaphor and Tool*, a cura di H. Haken, A. Karlqvist e U. Svedin, Berlin, Springer-Verlag.

DeYoe, E.A. e van Essen, D.C.
1988 *Concurrent Processing Streams in Monkey Visual Cortex*, in «Trends in Neuroscience», vol. 11, pp. 219-226.

Diamantaras, K.I. e Kung, S.Y.
1996 *Principal Component Neural Networks: Theory and Applications*, New York, John Wiley and Sons.

Domany, E., van Hemmen, J.L. e Schulten, K. (a cura di)
1994 *Models of Neural Networks II: Temporal Aspects of Coding and Information Processing in Biological Systems*, Berlin, Springer-Verlag.

Douglas, R.J. e Martin, K.A.C.
1990 *Neocortex*, in *The Synaptic Organization of the Brain*, a cura di G.M. Shepherd, Oxford, Oxford University Press, pp. 459-509.

Dutta, S. e Shekkar, S.
1988 *Bond Rating: A Non-Conservative Application of Neural Networks*, in *Proceedings of the IEEE International Conference on Neural Networks*, a cura di M. Caudill e C. Butler, New York, IEEE Press, vol. II, pp. 443-450.

Elman, J.L.
1990 *Finding Structure in Time*, in «Cognitive Science», vol. 14, pp. 179-211.

Engel, A.K., Koenig, P., Kreiter, A.K., Schillen, T.B. e Singer, W.
1992 *Temporal Coding in the Visual Cortex: New Vistas on Integration in the Nervous System*, in «Trends in Neuroscience», vol. 15, pp. 218-226.

Enquist, M. e Arak, A.
1994 *Symmetry, Beauty and Evolution*, in «Nature», vol. 372, pp. 169-172.

Fahlman, S.E.
1989 *Fast-Learning Variations on Back-Propagation: An Empirical Study*, in *Proceedings of the 1988 Connectionist Models Summer School*, a cura di D. Touretzky, G. Hinton e T. Sejnowski, San Mateo, CA, Morgan Kaufmann, pp. 38-51.

Fahlman, S.E. e Lebiere, C.
1990 *The Cascade-Correlation Learning Architecture*, in *Advances in Neural Information Processing Systems 2*, a cura di D.S. Touretzky, San Francisco, CA, Morgan Kaufmann, pp. 524-532.

Farah, M.J. e McClelland, J.L.
1991 *A Computational Model of Semantic Memory Impairment: Modality Specificity and Emergent Category Specificity*, in «Journal of Experimental Psychology: General», vol. 120, pp. 339-357.

Fausett, L.
1994 *Fundamentals of Neural Networks*, Englewood Cliffs, NJ, Prentice-Hall.

Feldman, J.A. e Ballard, D.H.
1982 *Connectionist Models and Their Properties*, in «Cognitive Science», vol. 6, pp. 205-254.

Field, D.J.
1994 *What Is the Goal of Sensory Coding?*, in «Neural Computation», vol. 4, pp. 559-601.

Fine, T.L.
1999 *Feedforward Neural Network Methodology*, Berlin, Springer-Verlag.

Floreano, D.
1993a *Patterns of Interactions in Shared Environments*, in *Proceedings of the III European Conference on Artificial Life*, Bruxelles, ULB, vol. II.
1993b *Anticipatory Tracking of a Moving Object*, in *Artificial and Natural Visual Systems*, a cura di V. Roberto, Berlin, Springer-Verlag.
1993c *Emergence of Home-Based Foraging Strategies in Ecosystems of Neural Networks*, in *From Animals to Animats II: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, a cura di J. Meyer, H.L. Roitblat e S.W. Wilson, Cambridge, MA, MIT Press-Bradford Books.
1993d *ROBOGEN: A Software Package for Evolutionary Control Systems. (Release 1.1)*, rapporto tecnico LabTeCo 93-01, Laboratorio Tecnologie Cognitive, AREA Parco Scientifico, Trieste.
1995 *Neuroetologia Artificiale: alla ricerca delle basi adattive dell'intelligenza*, tesi di dottorato, Dipartimento di Psicologia, Università di Trieste.

Floreano, D., De Lillo, C. e Antinucci, F.
1995 *Two Different Mechanisms Involved in Transitive Inference? A Computational Investigation*, rapporto tecnico LabTeCo 95-10, Laboratorio di Tecnologie Cognitive, AREA Parco Scientifico, Trieste.

Floreano, D. e Mattiussi, C.
2001 *Evolution of Spiking Neural Controllers for Autonomous Vision-Based Robots*, in *Evolutionary Robotics: From Intelligent Robotics to Artificial Life*, LNCS 2217, a cura di T. Gomi, Berlin, Springer-Verlag.

Floreano, D., Miglino, O. e Parisi, D.
1991 *Emergent Complex Behaviours in Ecosystems of Neural Networks*, in *Parallel Architectures and Neural Networks*, a cura di E. Caianiello, Singapore, World Scientific Press.

Floreano, D. e Mondada, F.
1994a *Automatic Creation of an Autonomous Agent: Genetic Evolution of a Neural-Network Driven Robot*, in *From Animals to Animats III: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, a cura di D. Cliff, P. Husbands, J.-A Meyer e S.W. Wilson, Cambridge, MA, MIT Press-Bradford Books.
1994b *Active Perception, Navigation, Homing, and Grasping: An Autonomous Perspective*, in *Proceedings of the conference From Perception to Action*, a cura di J-D. Nicoud e P. Gaussier, Los Alamitos, CA, IEEE Computer Press.
1995 *From Evolution of Instincts to Evolution of Learning Structures in Robotic Agents*, rapporto tecnico, Laboratorio di Microinformatica, Politecnico Federale di Losanna.
1996 *Evolution of Homing Navigation in a Real Mobile Robot*, in «IEEE Transactions on Systems, Man and Cybernetics», in stampa.

Floreano, D. e Parisi, D.
1992 *Internal Prediction and Environment Models: An Application to the Development of Language*, rapporto tecnico CCCN-10, Centre for Cognitive and Computational Neuroscience, University of Stirling, U.K.

Floreano, D., Parisi, D., Antinucci, F. e Natale, F.
1993 *Eye Tracking with a Recurrent Neural Network*, in «Cognitive Systems», vol. 3-4, pp. 331-349.

Földiák, P.
1989 *Adaptive Network for Optimal Linear Feature Extractions*, in *Proceedings of the International Joint Conference on Neural Networks*, Washington, DC, vol. I, pp. 401-405.

Fontanari, J.F. e Meir, R.
1991 *Evolving a Learning Algorithm for the Binary Perceptron*, in «Network», vol. 2, pp. 353-359.

Fortune, E.S. e Rose, G.J.
2001 *Short-term synaptic plasticity as a temporal filter*, in «Trends in Neurosciences», vol. 24, pp. 382-385.

Frean, M.R.
1990 *The Upstart Algorithm: A method for Constructing and Training Feedforward Neural Networks*, in «Neural Computation», vol. 2, pp. 190-209.

Frégnac, Y.
1979 *Development of Orientation Selectivity in the Primary Visual Cortex of Normally and Dark Reared Kittens*, in «Biological Cybernetics», vol. 34, pp. 187-204.

Frégnac, Y. e Imbert, M.
1978 *Early Development of Visual Cortical Cells in Normal and Dark-Reared Kittens. Relationship between Orientation Selectivity and Ocular Dominance*, in «Journal of Physiology (London)», vol. 278, pp. 27-44.

Fukushima, K.
1975 *Cognitron: A Self-Organizing Multi-Layered Neural Network*, in «Biological Cybernetics», vol. 20, pp. 121-136.
1988 *Neocognitron: A Hierarchical Neural Network Model Capable of Visual Pattern Recognition*, in «Neural Networks», vol. 1, pp. 119-130.

Fukushima, K., Miyake, S. e Ito, T.
1983 *Neocognitron: A Neural Network Model for a Mechanism of Visual Pattern Recognition*, in «IEEE Transactions on Systems, Man, and Cybernetics», vol. 13, pp. 826-834.

Fuminori, S. e Fukuda, T.
1994 *Two-Link-Robot Brachiation with Connectionist Q-Learning*, in *From Animals to Animats III: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, a cura di D. Cliff, P. Husbands, J.-A. Meyer e S.W. Wilson, Cambridge, MA, MIT Press-Bradford Books.

Gallant, S.I.
1986 *Optimal Linear Discriminants*, in *Proceedings of the Eighth International Conference on Pattern Recognition*, New York, IEEE Press, pp. 849-852.

Gerbino, W.
1984 *La percezione*, Bologna, Il Mulino.

Gerstner, W.
1999 *Spiking Neurons*, in *Pulsed Neural Networks*, a cura di W. Maass e C.M. Bishop, Cambridge, MA, MIT Press-Bradford Books.

Gerstner, W., Kempter, R., van Hemmen, J.L. e Wagner, H.
1999 *Hebbian Learning of Pulse Timing in the Barn Owl Auditory System*, in *Pulsed Neural Networks*, a cura di W. Maass e C.M. Bishop, Cambridge, MA, MIT Press-Bradford Books.

Girosi, F. e Poggio, T.
1989 *Representative Properties of Networks: Kolmogorov's Theorem is Irrelevant*, in «Neural Computation», vol. 1, pp. 465-469.

Gluck, M.A.
1991 *Stimulus Sampling and Distributed Representations in Adaptive Network Theories of Learning*, in *Festschrift for W.K. Estes*, a cura di A. Healy, S. Kosslyn e R. Shiffrin, Hillsdale, NJ, Lawrence-Erlbaum Associates.

Goldberg, D.E.
1989 *Genetic Algorithms in Search, Optimization and Machine Learning*, Reading, MA, Addison-Wesley.

Goldberg, D.E. e Deb, K.
1991 *A Comparative Analysis of Selection Schemes Used in Genetic Algorithms*, in *Foundations of Genetic Algorithms*, a cura di G.J.E. Rawlins, San Mateo, CA, Morgan Kaufmann.

Gorman, R. e Sejnowski, T.
1988a *Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets*, in «Neural Networks», vol. 1, pp. 75-89.
1988b *Learned Classification of Sonar Targets Using a Massively Parallel Network*, in «IEEE Transactions on Acoustics, Speech, and Signal Processing», vol. 36, pp. 1135-1140.

Gray, C.M., König, P., Engel, A.K. e Singer, W.
1989 *Oscillatory Responses in Cat Visual Cortex Exhibit Inter-Columnar Synchronization which Reflects Global Stimulus Properties*, in «Nature», vol. 338, pp. 334-337.

Gray, J.J.
2000 *The Hilbert Challenge*, Oxford, Oxford University Press.

Grefenstette, J.J.
1986 *Optimization of Control Parameters for Genetic Algorithms*, in «IEEE Transactions on Systems, Man, and Cybernetics», vol. 16, pp. 122-128.

Grossberg, S.
1969 *Embedding Fields: A Theory of Learning with Physiological Implications*, in «Journal of Mathematical Psychology», vol. 6, pp. 209-239.
1976a *Adaptive Pattern Classification and Universal Recoding: I. Parallel Development and Coding of Neural Feature Detectors*, in «Biological Cybernetics», vol. 23, pp. 121-134.
1976b *Adaptive Pattern Classification and Universal Recoding: II. Feedback, Expectation, Olfaction, and Illusions*, in «Biological Cybernetics», vol. 23, pp. 187-202.
1980 *How does a Brain Build a Cognitive Code?*, in «Psychological Review», vol. 87, pp. 1-51.
1987 *Competitive Learning: From Interactive Activation to Adaptive Resonance*, in «Cognitive Science», vol. 11, pp. 23-63.

Gruau, F.
1992 *Genetic Systems of Boolean Neural Networks with a Cell Rewriting Developmental Process*, in *Combination of Genetic Algorithms and Neural Networks*, a cura di D. Whitley e J.D. Schaffer, Los Alamitos, CA, IEEE Computer Society Press.
1993 *Adding Learning to the Cellular Development of Neural Networks: Evolution and the Baldwin Effect*, in «Evolutionary Computation», vol. 1, pp. 213-233.

Hancock, P.J.B.
1992 *Genetic Algorithms and Permutation Problems: A Comparison of*

Recombination Operators for Neural Structure Specification, in *Proceedings of an International Workshop on the Combinations of Genetic Algorithms and Neural Networks*, a cura di L.D. Whitley e J.D. Schaffer, Los Alamitos, CA, IEEE Press.

Hancock, P.J.B., Baddeley, R.J. e Smith L.S.
1992 *The Principal Components of Natural Images*, in «Network», vol. 3, pp. 61-70.

Hansel, D., Mato, G., Meunier, C. e Neltner, L.
1998 *Numerical simulations of integrate-and-fire neural networks*, in «Neural Computation», vol. 10, pp. 467-483.

Hanson, S.J. e Pratt, L.
1989 *A Comparison of Different Biases for Minimal Network Construction with Back-Propagation*, in *Advances in Neural Information Processing Systems I*, a cura di D.S. Touretzky, San Mateo, CA, Morgan Kaufmann.

Harp, S.A., Samad, T. e Guha, A.
1989 *Toward the Genetic Synthesis of Neural Networks*, in *Proceedings of the Third International Conference on Genetic Algorithms*, a cura di J.D. Schaffer, San Mateo, CA, Morgan Kaufmann.

Haykin, S.
1999 *Neural Networks: A Comprehensive Foundation*, Upper Saddle River, NJ, Prentice Hall, 2ª ed.

Hebb, D.O.
1949 *The Organisation of Behavior*, New York, Wiley and Sons; trad. it. *L'organizzazione del comportamento. Una teoria neuropsicologica*, Milano, Angeli, 1975.

Hecht-Nielsen, R.
1987a *Kolmogorov's Mapping Neural Network Existence Theorem*, in *Proceedings of the 1st IEEE International Conference on Neural Networks*, New York, IEEE Press, vol. III, pp. 11-13.
1987b *Applications of Counterpropagation Networks*, in «Applied Optics», vol. 26, pp. 4979-4984.
1990 *Neurocomputing*, Reading, MA, Addison-Wesley.

Hertz, J., Krogh, A. e Palmer, R.G.
1991 *Introduction to the Theory of Neural Computation*, Reedwood City, CA, Addison-Wesley.

Hinton, G.E.
1986 *Learning Distributed Representations of Concepts*, in *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, Hillsdale, NJ, Lawrence Erlbaum Associates.

Hinton, G.E. e Nowlan, S.J.
1987 *How Learning Can Guide Evolution*, in «Complex Systems», vol. 1, pp. 495-502.

Hinton, G.E. e Sejnowski, T.J.
1986 *Learning and Relearning in Boltzmann Machines*, in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, a cura di D.E. Rumelhart e J.L. McClelland, Cambridge, MA, MIT Press, vol. 1, pp. 282-317.

Hodgkin, A.L. e Huxley, A.F.
1952 *A Quantitative Description of Membrane Current and its Application to Conduction and Excitation in Nerve*, in «Journal of Physiology (London)», vol. 108, pp. 500-544.
Hoffmeister, F. e Bäck, T.
1991 *Genetic Algorithms and Evolution Strategies: Similarities and Differences*, in *Parallel Problem Solving from Nature*, a cura di H-P. Schwefel e R. Maenner, Berlin, Springer-Verlag, pp. 455-469.
Holland, J.H.
1975 *Adaptation in Natural and Artificial Systems*, Ann Arbor, The University of Michigan Press.
Hopfield, J.J.
1982 *Neural Networks and Physical Systems with Emergent Collective Computational Abilities*, in «Proceedings of the National Academy of Sciences USA», vol. 79, pp. 2554-2558, ristampato in Anderson e Rosenfeld [1988].
1984 *Neurons with Graded Responses Have Collective Computational Properties Like Those of Two-State Neurons*, in «Proceedings of the National Academy of Sciences USA», vol. 81, pp. 3088-3092, ristampato in Anderson e Rosenfeld [1988].
Hopfield, J.J. e Tank, D.W.
1985 *«Neural» Computation of Decisions in Optimization Problems*, in «Biological Cybernetics», vol. 52, pp. 141-152.
Hornik, K., Stinchcombe, M. e White, H.
1989 *Multilayer Feedorward Networks Are Universal Approximators*, in «Neural Networks», vol. 2, pp. 359-366.
Hubel, D.H. e Wiesel, T.N.
1963 *Receptive Fields of Cells in Striate Cortex of Very Young, Visually Inexperienced Kittens*, in «Journal of Neurophysiology», vol. 26, pp. 994-1002.
1977 *Functional Architecture of a Macaque Monkey Visual Cortex*, in «Procedings of the Royal Society of London B», vol. 198, pp. 1-59.
1979 *Brain Mechanisms of Vision*, in «Scientific American», vol. 241, pp. 150-162.
Husbands, P., Harvey, I. e Cliff, D.T.
1993 *Analysing Recurrent Dynamical Networks Evolved for Robot Control*, rapporto tecnico CSRP 265, School of Cognitive and Computing Science, University of Sussex.

Intrator, N. e Cooper, L.N.
1992 *Objective Function Formulation of the BCM Theory of Visual Plasticity: Statistical Connections, Stability Conditions*, in «Neural Networks», vol. 5, pp. 3-17.

Jacobs, R.A.
1988 *Increased Rates of Convergence Through Learning Rate Adaptation*, in «Neural Networks», vol. 1, pp. 295-307.

Jacobs, R.A. e Kosslyn, S.M.
1994 *Encoding Shape and Spatial Relations: The Role of Receptive Field Size in Coordinating Complementary Representations*, in «Cognitive Science», vol. 18, pp. 361-386.
Jahnke, A., Roth, U. e Schönauer, T.
1999 *Digital Simulation of Spiking Neural Networks*, in *Pulsed Neural Networks*, a cura di W. Maass e C.M. Bishop, Cambridge, MA, MIT Press-Bradford Books.
Johnson, R.C. e Brown, C.
1988 *Cognizers: Neural Networks and Machines that Think*, New York, John Wiley and Sons.
Johnstone, R.A.
1994 *Female Preference for Symmetrical Males as a By-Product of Selection for Mate Recognition*, in «Nature», vol. 372, pp. 172-175.
Jolliffe, I.T.
1986 *Principal Component Analysis*, New York, Springer-Verlag.
Jones, D.
1990 *Neural Networks for Medical Diagnosis*, in *Handbook of Neural Computing Applications*, a cura di A.J. Maren, C.T. Harston e R.M. Pap, London, Academic Press.
Jordan, M.I.
1989 *Serial Order: A Parallel, Distributed Processing Approach*, in *Advances in Connectionist Theory: Speech*, a cura di J.L. Elman e D.E. Rumelhart, Hillsdale, NJ, Erlbaum.
Jordan, M.I. e Rumelhart, D.E.
1992 *Forward Models: Supervised Learning with a Distal Teacher*, in «Cognitive Science», vol. 16, pp. 307-354.
Jutten, C. e Herault, J.
1991a *Blind Separation of Sources, Part I: An Adaptive Algorithm Based on Neuromimetic Architectures*, in «Signal Processing», vol. 24, pp. 1-10.
1991b *Blind Separation of Sources, Part II: Problems Statement*, in «Signal Processing», vol. 24, pp. 11-20.

Kandel, E.R., Schwartz, J.H. e Jessell, T.M. (a cura di)
1991 *Principles of Neural Science*, New York, Appleton and Lange, 3ª ed.; trad. it. *Principi di neuroscienze*, Milano, CEA, 1994.
Kay, J.
1992 *Some Notes on Entropy and Mutual Information*, Lecture Notes for the M.Sc. course in Neural Computation, Centre for Cognitive and Computational Neuroscience, University of Stirling, U.K.
Kelso, S.R., Ganong, A.H. e Brown, T.H.
1986 *Hebbian Synapses in Hippocampus*, in «Proceedings of the National Academy of Sciences USA», vol. 83, pp. 5326-5330.
Kempter, R., Gerstner, W. e van Hemmen, J.L.
1999a *Spike-Based Compared to Rate-Based Hebbian Learning*, in *Advances in Neural Information Processing Systems 11*, a cura di M.S. Kearns, S.A. Solla e D.A. Cohn, Cambridge, MA, MIT Press.

1999b *Hebbian Learning and Spiking neurons*, in «Physical Review E», vol. 59, pp. 4498-4514.

Khotanzad, A., Lu, J. e Srinath, M.

1989 *Target Detection Using a Neural Network Based Passive Sonar System*, in *Proceedings of the IEEE/INNS International Joint Conference on Neural Networks*, San Diego, CA, IEEE Press, vol. I.

Kirkpatrick, S., Gelatt, C.D. e Vecchi, M.P.

1983 *Optimization by Simulated Annealing*, in «Science», vol. 220, pp. 671-680.

Kitano, H.

1990 *Designing Neural Networks Using Genetic Algorithms with Graph Generation System*, in «Complex Systems», vol. 4, pp. 461-476.

Kleinfeld, D. e Sompolinski, H.

1989 *Associative Network Models for Central Pattern Generators*, in *Methods in Neuronal Modeling: From Synapses to Networks*, a cura di C. Koch e I. Segev, Cambridge, MA, MIT Press, pp. 195-246.

Klopf, A.H.

1988 *A Neuronal Model of Classical Conditioning*, in «Psychobiology», vol. 16, pp. 85-125.

Knudsen, E.I., du Lac, S. e Esterly, S.D.

1987 *Computational Maps in the Brain*, in «Annual Review of Neuroscience», vol. 10, pp. 41-65.

Koch, C.

1999 *Biophysics of Computation*, Oxford, Oxford University Press.

Koch, C. e Segev, I. (a cura di)

1998 *Methods in Neuronal Modeling: From Ions to Networks*, Cambridge, MA, MIT Press-Bradford Books, 2ª ed.

Kohonen, T.

1974 *An Adaptive Associative Memory Principle*, in «IEEE Transactions on Computers», vol. C-23, pp. 444-445.

1982 *Self-Organized Formation of Topologically Correct Feature Maps*, in «Biological Cybernetics», vol. 43, pp. 59-69.

1988 *The «Neural» Phonetic Typewriter*, in «Computer», vol. 21, pp. 11-22.

1989 *Self-Organization and Associative Memory*, Berlin, Springer-Verlag.

1997 *Self-Organizing Maps*, Berlin, Springer-Verlag, 2ª ed.

Kohonen, T., Barna, G. e Chrisley, R.

1988 *Statistical Pattern Recognition with Neural Networks: Benchmarking Studies*, in *Proceedings of the IEEE International Conference on Neural Networks*, New York, IEEE Press, vol. I, pp. 61-68.

Kohonen, T., Kangas, J. e Laaksonen, J.

1992 *The Self-Organizing Map Program Package. Version 1.2*, rapporto tecnico, Laboratory of Computer and Information Science, Helsinki University of Technology, Finland.

Kolen, F.J. e Pollack, J.B.
1990 *Back Propagation is Sensitive to Initial Conditions*, rapporto tecnico TR 90-JK-BPSIC, Computer and Information Science Department, The Ohio State University.
Kosslyn, S.M., Chabris, C.F., Marsolek, C.J. e Koenig, O.
1992 *Categorical Versus Coordinate Spatial Relations: Computational Analyses and Computer Simulations*, in «Journal of Experimental Psychology: Human Perception and Performance», vol. 18, pp. 562-577.
Koza, J.R.
1992 *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, Cambridge, MA, MIT Press.
Kramer, A.H. e Sangiovanni-Vincentelli, A.
1989 *Efficient Parallel Learning Algorithms for Neural Networks*, in *Advances in Neural Information Processing Systems I*, a cura di D.S. Touretzky, San Mateo, CA, Morgan Kaufmann, pp. 40-48.
Kruse, P. e Stadler, M. (a cura di)
1995 *Ambiguity in Mind and Nature. Multistable Cognitive Phenomena*, Berlin, Springer-Verlag.
Kung, S.Y. e Diamantaras, K.I.
1990 *A Neural Network Learning Algorithm for Adaptive Principal Component extraction (APEX)*, in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Albuquerque, NM, vol. II, pp. 861-864.
Kurkova, V.
1991 *Kolmogorov's Theorem Is Relevant*, in «Neural Computation», vol. 3, pp. 617-622.

Lang, K.J., Waibel, A.H. e Hinton, G.E.
1990 *A Time-Delay Neural Network Architecture for Isolated Word Recognition*, in «Neural Networks», vol. 3, pp. 23-43.
Langton, C.G.
1992a *Artificial Life*, in *Artificial Life, Proceedings Volume of Santa Fe Conference*, a cura di C.G. Langton, Reading, MA, Addison-Wesley.
1992b *Preface*, in *Artificial Life II: Proceedings Volume of Santa Fe Conference*, a cura di C.G. Langton, J.D. Farmer, S. Rasmussen e C. Taylor, Reading, MA, Addison-Wesley.
Le Cun, Y.
1985 *Une Procedure d'Apprentissage pour Réseau à Seuil Asymétrique*, in *Cognitiva 85: À la Frontiere de l'Intelligence Artificielle des Sciences de la Connaissance des Neurosciences*, Paris, CESTA.
Le Cun, Y., Boser, B., Denker, J.S., Henderson, R.E., Howard, R.E., Hubbard, W. e Jackel, L.D.
1990 *Handwritten Digit Recognition with a Backpropagation Network*, *Advances in Neural Information Processing Systems 2*, a cura di

D.S. Touretzky, San Mateo, CA, Morgan Kaufmann, pp. 396-404.

Le Cun, Y., Denker, J.S. e Solla, S.A.
1990 *Optimal Brain Damage*, in *Advances in Neural Information Processing Systems 2*, a cura di D.S. Touretzky, San Mateo, CA, Morgan Kaufmann, pp. 598-605.

Leen, T.K.
1991 *Dynamics of Linear Feature-Discovery Networks*, in «Network», vol. 2, pp. 85-105.

Lin, L-J.
1992 *Self-Improving Reactive Agents Based On Reinforcement Learning, Planning and Teaching*, in «Machine Learning», vol. 8, pp. 293-321.

Lindenmayer, A.
1968 *Mathematical Models for Cellular Interactions in Development*, in «Journal of Theoretical Biology», vol. 18, pp. 280-299.
1971 *Developmental Systems without Cellular Interactions, their Languages and Grammars*, in «Journal of Theoretical Biology», vol. 30, pp. 455-484.

Lindsay, P.N. e Norman, D.A.
1972 *Human Information Processing*, New York, Academic Press; trad. it. *L'uomo elaboratore di informazioni*, Firenze, Giunti, 1982.

Linsker, R.
1986 *From Basic Network Principles to Neural Architecture (Series)*, in «Proceedings of the National Academy of Sciences of the USA», vol. 83, pp. 7508-7512, 8390-8394, 8779-8783.
1988 *Self-Organization in a Perceptual Network*, in «Computer», vol. 3, pp. 105-117.

Maass, W. e Bishop, C.M. (a cura di)
1999 *Pulsed Neural Networks*, Cambridge, MA, MIT Press-Bradford Books.

Maass, W. e Sontag, E.D.
2000 *Neural Systems as Nonlinear Filters*, in «Neural Computation», vol. 12, pp. 1743-1772.

Mach, E.
1896 *Die Analyse der Empfindungen und das Verhaeltnis des Physischen zum Psychischen*, Jena, Verlag von Gustav Fischer; trad. it. *L'analisi delle sensazioni e il rapporto fra fisico e psichico*, Milano, Feltrinelli, 1975.

MacLennan, B.
1992 *Synthetic Ethology: An Approach to the Study of Communication*, in *Artificial Life II: Proceedings Volume of Santa Fe Conference*, a cura di C.G. Langton, J.D. Farmer, S. Rasmussen e C. Taylor, Reading, MA, Addison-Wesley.

Mahadevan, S. e Connell, J.
1991 *Scaling Reinforcement Learning to Robotics by Exploiting the Sub-*

sumption Architecture, in *Machine Learning: Proceedings of the Eighth International Workshop*, a cura di L.A. Birnbaum e G.C. Collins, San Mateo, CA, Morgan Kaufmann.

Mahowald, M.A. e Mead, C.
1991 *La retina di silicio*, in «Le Scienze», vol. 275, pp. 34-41.

Makram-Ebeid, S., Sirat, J.-A. e Viale, J.-R.
1989 *A Rationalized Back-Propagation Learning Algorithm*, in *Proceedings of the International Joint Conference on Neural Networks*, New York, IEEE Press, vol. II, pp. 373-380.

Maniezzo, V.
1994 *Genetic Evolution of the Topology and Weight Distribution of Neural Networks*, in «IEEE Transactions on Neural Networks», vol. 5, pp. 39-53.

Mareschal, D. e Shultz, T.R.
1993 *A Connectionist Model of the Development of Seriation*, in *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, Hillsdale, NJ, Erlbaum.

Markram, H. e Tsodyks, M.
1996 *Redistribution of synaptic efficacy between neocortical pyramidal neurons*, in «Nature», vol. 382, pp. 807-810.

Marr, D.
1982 *Vision*, New York, Freeman.

Mazzoni, P., Andersen, R.A. e Jordan, M.I.
1991 *A More Biologically Plausible Learning Rule for Neural Networks*, in «Proceedings of the National Academy of Science USA», vol. 88, pp. 4433-4437.

McClelland, J.L., Rumelhart, D.E. e Gruppo PDP
1986 *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Psychological and Biological Models*, vol. II, Cambridge, MA, MIT Press-Bradford Books; trad. it. parziale in *Microstruttura dei processi cognitivi*, Bologna, Il Mulino, 1991.

McCulloch, W. e Pitts, W.
1943 *A Logical Calculus of the Ideas Immanent in Nervous Activity*, in «Bulletin of Mathematical Biophysics», vol. 5, pp. 115-133.

Merzenich, M.M. e Kaas, J.H.
1980 *Principles of Organization of Sensory-Perceptual Systems in Mammals*, in *Progress in Psychobiology and Physiological Psychology*, a cura di AA.VV., London, Academic Press, vol. IX, pp. 1-42.

Mézard, M. e Nadal, J.-P.
1989 *Learning in Feedforward Layered Networks: The Tiling Algorithm*, in «Journal of Physics A», vol. 22, pp. 2191-2204.

Miglino, O., Nafasi, K. e Taylor, C.
1995 *Selection for Wandering Behavior in a Small Robot*, in «Artificial Life», vol. 2, pp. 101-116.

Millan, J.
1996 *Real Robot Control with a Method of Reinforcement Learning*, in «IEEE Transactions on Systems, Man and Cybernetics-Part B», in stampa.

Miller, G.F., Todd, P.M. e Hedge, S.U.
1989 *Designing Neural Networks Using Genetic Algorithms*, in *Proceedings of the Third International Conference on Genetic Algorithms and Their Applications*, a cura di J.D. Schaffer, San Mateo, CA, Morgan Kaufmann, pp. 379-384.
Miller, W.T., Sutton, R.S. e Werbos, P.J. (a cura di)
1990 *Neural Networks for Control*, Cambridge, MA, MIT Press.
Minsky, M.L. e Papert, S.A.
1969 *Perceptrons*, Cambridge, MA, MIT Press.
1988 *Perceptrons. Expanded Edition*, Cambridge, MA, MIT Press.
Mitchell, M.
1996 *An Introduction to Genetic Algorithms*, Cambridge, MA, MIT Press-Bradford Books; trad. it. *Introduzione agli algoritmi genetici*, Milano, Apogeo, 1999.
Miyata, Y.
1988 *The Learning and Planning of Actions*, rapporto tecnico ICS 8802, University of California at San Diego.
Montana, D. e Davis, L.
1989 *Training Feedforward Neural Networks Using Genetic Algorithms*, in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, San Mateo, CA, Morgan Kaufmann.
Moody, J. e Darken, C.
1989 *Fast Learning in Networks of Locally-Tuned Processing Units*, in «Neural Computation», vol. 1, pp. 281-294.
Mortara, A. e Venier, P.
1999 *Analog VLSI Pulsed Networks for Perceptive Processing*, in *Pulsed Neural Networks*, a cura di W. Maass e C.M. Bishop, Cambridge, MA, MIT Press-Bradford Books.
Mozer, M.C. e Smolensky, P.
1989 *Using Relevance to Reduce Network Size Automatically*, in «Connection Science», vol. 1, pp. 3-16.
Mühlenbein, H. e Kindermann, J.
1989 *The Dynamics of Evolution and Learning – Towards Genetic Neural Networks*, in *Connectionism in perspective*, a cura di R. Pfeifer, Z. Schreter, F. Fogelman-Soulie e L. Steels, Amsterdam, Elsevier Science Publishers B.V. (North-Holland), pp. 173-197.
Müller, K.-R., Mika, S., Rätsch, Tsuda, K. e Schölkopf, B.
2001 *An Introduction to Kernel-Based Learning Algorithms*, in «IEEE Transactions on Neural Networks», vol. 12, pp. 181-200.

Nagy, G.
1991 *Neural Networks – Then and Now*, in «IEEE Transactions on Neural Networks», vol. 2, pp. 316-317.
Naillon, M. e Theeten, J.-B.
1989 *Neural Approach for TV Image Compression Using a Hopfield Type Network*, in *Advances in Neural Information Processing Systems 1*, a cura di D.S. Touretzky, San Mateo, CA, Morgan Kaufmann.

Neisser, U.
1967 *Cognitive Psychology*, New York, Appleton-Century-Crofts; trad. it. *Psicologia cognitivista*, Firenze, Martello-Giunti, 1976.
Nguyen, D. e Widrow, B.
1989 *The Truck Backer-Upper: An Example of Self-Learning in Neural Networks*, in *Proceedings of the International Joint Conference on Neural Networks*, Washington, DC, vol. II, pp. 357-363.
1990 *Improving the Learning Speed of Two-Layer Neural Networks by Choosing Initial Values of the Adaptive Weights*, in *Proceedings of the International Joint Conference on Neural Networks*, San Diego, CA, vol. III, pp. 21-26.
Nolfi, S., Elman, J.L. e Parisi, D.
1994 *Learning and Evolution in Neural Networks*, in «Adaptive Behavior», vol. 3, pp. 5-28.
Nolfi, S. e Floreano, D.
1998 *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*, Cambridge, MA, MIT Press-Bradford Books.
Nolfi, S., Floreano, D., Miglino, O. e Mondada, F.
1994 *How to Evolve Autonomous Robots: Different Approaches in Evolutionary Robotics*, in *Proceedings of the Fourth Workshop on Artificial Life*, a cura di R. Brooks e P. Maes, Boston, MA, MIT Press.
Nolfi, S., Miglino, O. e Parisi, D.
1994 *Phenotipic Plasticity in Evolving Neural Networks*, in *Proceedings of the conference From Perception to Action*, a cura di J-D. Nicoud e P. Gaussier, Los Alamitos, CA, IEEE Computer Press.
Nolfi, S. e Parisi, D.
1994 *Genotypes for Neural Networks*, in *The Handbook of Brain Theory and Neural Networks*, a cura di M.A. Arbib, Cambridge, MA, MIT Press.
Nowlan, S.J. e Hinton, G.E.
1992 *Simplifying Neural Networks by Soft Weight-Sharing*, in «Neural Computation», vol. 4, pp. 473-493.

Oja, E.
1982 *A Simplified Neuron Model as a Principal Component Analyzer*, in «Journal of Mathematical Biology», vol. 15, pp. 267-273.
1989 *Neural Networks, Principal Components, and Subspaces*, in «International Journal of Neural Systems», vol. 1, pp. 61-68.
O'Keefe, J. e Nadel, L.
1978 *The Hippocampus as a Cognitive Map*, Oxford, Clarendon Press.
Orchard, G.A. e Phillips, W.A.
1991 *Neural Computation. A Beginner's Guide*, London, Lawrence Erlbaum Associates.
Orr, G.B. e Müller, K.-R. (a cura di)
1998 *Neural Networks: Tricks of the Trade*, Lecture Notes in Computer Science LCS 1524, Berlin, Springer-Verlag.

Parisi, D.
1989 *Intervista sulle reti neurali*, Bologna, Il Mulino.
Parisi, D., Cecconi, F. e Nolfi, S.
1990 *Econets: Neural Networks that Learn in an Environment*, in «Network», vol. 1, pp. 149-168.
Parisi, D., Pagliarini, L. e Floreano, D.
1994 *Un modello neurale dell'apprendimento del linguaggio: imitazione, coordinamento arbitrario di schemi, rappresentazione mentale*, in *Il Lessico: processi e rappresentazioni*, a cura di A. Laudanna e C. Burani, Firenze, La Nuova Italia Scientifica.
Parisi, G.
1986 *Asymmetric Neural Networks and the Process of Learning*, in «Journal of Physics A», vol. 19, pp. L675-L680.
Parker, D.B.
1985 *Learning Logic*, rapporto tecnico TR-47, Center for Computational Research in Economics and Management Science, MIT, Boston.
Parks, R.W., Levine, D.S. e Long, D.L. (a cura di)
1998 *Fundamentals of Neural Network Modeling: Neuropsychology and Cognitive Neuroscience*, Cambridge, MA, MIT Press.
Pearlmutter, B.A.
1995 *Gradient Calculations for Dynamic Recurrent Neural Networks*, in «IEEE Transactions on Neural Networks», vol. 6, pp. 1212-1228.
Penna, M.A.
1992 *Il problema dell'interferenza catastrofica nei modelli connessionistici*, rapporto tecnico, Dipartimento di Psicologia, Università di Roma «La Sapienza».
Perrett, D.I., Mistlin, A.J. e Chitty, A.J.
1987 *Visual Neurons Responsive to Faces*, in «Trends in Neurosciences», vol. 10, pp. 358-364.
Peterson, C. e Anderson, J.R.
1987 *A Mean Field Theory Learning Algorithm for Neural Networks*, in «Complex Systems», vol. 1, pp. 995-1019.
Peterson, C. e Soedeberg, B.
1989 *A New Method for Mapping Optimization Problems onto Neural Networks*, in «International Journal of Neural Systems», vol. 1, pp. 3-22.
Piazzalunga, U. e Parisi, D.
1993 *Spatial Distribution of an Evolving Population of Neural Networks*, in *Artificial Life III: Proceedings Volume of Santa Fe Conference*, a cura di C.G. Langton, Reading, MA, Addison-Wesley.
Pinker, S. e Price, A.
1988 *On Language and Connectionism: Analysis of a Parallel Distributed Processing Model of Language Acquisition*, in «Cognition», vol. 28, pp. 73-193.
Plaut, D., Nowlan, S. e Hinton, G.
1986 *Experiments on Learning by Back-Propagation*, rapporto tecnico

CMU-CS-86-126, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Plaut, D.C. e Shallice, T.

1991 *Deep Dyslexia: A Case Study of Connectionist Neuropsychology*, rapporto tecnico CRG-TR-91-3, Department of Computer Science, University of Toronto, Canada.

1992 *Perseverative and Semantic Influences on Visual Object Naming Errors in Optic Aphasia: A Connectionist Account*, rapporto tecnico PDP.CNS.92.1, Department of Psychology, Carnegie Mellon University, Pittsburgh, PA.

Plumbley, J.C. e Fallside, F.

1988 *An Information-Theoretic Approach to Unsupervised Connectionist Models*, in *Proceedings of the 1988 Connectionist Models Summer School*, a cura di D. Touretzky, G. Hinton e T. Sejnowski, San Mateo, CA, Morgan Kaufmann, pp. 239-245.

Pomerleau, D.A.

1993 *Neural Network Perception for Mobile Robot Guidance*, Boston, Kluwer Academic Publishing.

Press, W.H., Teukolsky, S.A., Vetterling, W.T. e Flannery, B.P.

1992 *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge, Cambridge University Press, 2ª ed.

Prusinkiewicz, P. e Lindenmayer, A.

1992 *The Algorithmic Beauty of Plants*, New York, Springer-Verlag.

Quinlan, P.

1995 *Connessionismo e psicologia*, trad. it. Bologna, Il Mulino.

Radcliffe, N.J.

1990 *Genetic Neural Networks on MIMD Computers*, Tesi di Dottorato, Department of Computer Science, University of Edinburgh, U.K.

1991 *Forma Analysis and Random Respectful Recombination*, in *Proceedings of the Fourth International Conference on Genetic Algorithms*, a cura di R.K. Belew e L.B. Booker, San Mateo, CA, Morgan Kaufmann.

Ratliff, F.

1965 *Mach Bands: Quantitative Studies of Neural Networks in the Retina*, San Francisco, CA, Holden-Day.

Rechenberg, I.

1973 *Evolutionstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, Stuttgart, Friedrich Fromann Verlag.

Reed, R.D. e Marks, R.J.

1999 *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*, Cambridge, MA, MIT Press-Bradford Books.

Rescorla, R.A. e Wagner, A.R.

1972 *A Theory of Classical Conditioning: Variations in the Effectiveness of Reinforcement and Nonreinforcement*, in *Classical Conditioning*

II: Current Research and Theory, a cura di A.H. Black e W.F. Prokasy, New York, Appleton-Century-Crofts.

Riedmiller, M.
1994 *Advanced Supervised Learning in Multi-Layer Perceptrons – From Backpropagation to Adaptive Learning Algorithms*, in «International Journal of Computer Standards and Interfaces», vol. 5, pp. 1-11.

Rieke, F., Warland, D., de Ruyter van Steveninck, R. e Bialek, W.
1997 *Spikes: Exploring the Neural Code*, Cambridge, MA, MIT Press-Bradford Books.

Ripley, B.D.
1992 *Statistical Aspects of Neural Networks*, in *Proceedings of SemStat*, London, Chapman and Hall.

Ritter, H.J., Martinetz, T.M. e Schulten, K.J.
1992 *Neural Computation and Self-Organizing Maps: An Introduction*, Reading, MA, Addison-Wesley.

Roitblat, H.L., Moore, P.W.B., Nachtigall, P.E. e Penner, R.H.
1991 *Natural Dolphin Echo Recognition Using an Integrator Gateway Network*, in *Advances in Neural Information Processing Systems 3*, a cura di D.S. Touretzky e R. Lippman, San Mateo, CA, Morgan Kaufmann.

Rojas, R.
1996 *Neural Networks: A Systematic Introduction*, Berlin, Springer-Verlag.

Rolls, E.T. e Treves, A.
1998 *Neural Networks and Brain Function*, Oxford, Oxford University Press.

Rosenblatt, F.
1962 *Principles of Neurodynamics*, New York, Spartan Books.

Rotter, S. e Diesmann, M.
1999 *Exact digital simulation of time-invariant linear systems with applications to neuronal modeling*, in «Byological Cybernetics», vol. 81, pp. 381-402.

Rubner, J. e Tavan, P.
1989 *A Self-Organizing Network for Principal Component Analysis*, in «Europhysics Letters», vol. 10, pp. 693-698.

Rudnick, M.
1990 *A Bibliography of the Intersection of Genetic Search and Artificial Neural Networks*, rapporto tecnico CS/E 90-001, Department of Computer Science and Engineering, Oregon Graduate Center.

Rumelhart, D.E., Hinton, G.E. e Williams, R.J.
1986 *Learning Representations by Back-Propagation of Errors*, in «Nature», vol. 323, pp. 533-536.

Rumelhart, D.E., McClelland, J.L. e Gruppo PDP
1986 *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, vol. I, Cambridge, MA, MIT Press-Bradford Books; trad. it. parziale in *Microstruttura dei processi cognitivi*, Bologna, Il Mulino, 1991.

Rumelhart, D.E. e Zipser, D.
1985 *Feature Discovery by Competitive Learning*, in «Cognitive Science», vol. 9, pp. 75-112.

Samad, T. e Harp, S.A.
1989 *Self-Organization with Partial Data*, in «Network», vol. 3, pp. 205-212.

Samuel, A.L.
1959 *Some Studies in Machine Learning Using the Game of Checkers*, in «IBM Journal on Research and Development», pp. 210-229, ristampato in *Computers and Thought*, a cura di E.A. Feigenbaum e J. Feldman, New York, McGraw-Hill, 1963.

Sandberg, I.W. e Xu, L.
1997a *Uniform Approximation of Multidimensional Myopic Maps*, in «IEEE Transactions on Circuits and Systems - I: Fundamental Theory and Applications», vol. 44, pp. 477-485.
1997b *Uniform Approximation and Gamma Networks*, in «Neural Networks», vol. 10, pp. 781-784.

Sanger, T.D.
1989 *Optimal Unsupervised Learning in a Single Layer Network*, in «Neural Networks», vol. 2, pp. 459-473.

Scalettar, R. e Zee, A.
1988 *Emergence of Grandmother Memory in Feed Forward Networks: Learning with Noise and Forgetfulness*, in *Connectionist Models and Their Implications: Readings from Cognitive Science*, a cura di D. Waltz e J.A. Feldman, Norwood, Ablex.

Schaffer, J.D., Whitley, L.D. e Eshelman, L.J.
1992 *Combinations of Genetic Algorithms and Neural Networks: A Survey of the State of the Art*, in *Proceedings of an International Workshop on the Combinations of Genetic Algorithms and Neural Networks*, a cura di L.D. Whitley e J.D. Schaffer, Los Alamitos, CA, IEEE Press.

Schölkopf, B., Smola, A.J. e Müller, K.-R.
1998 *Nonlinear Component Analysis as a Kernel Eigenvalue Problem*, in «Neural Computation», vol. 10, pp. 1299-1319.

Schwefel, H-P.
1981 *Numerical Optimization of Computer Models*, Chichester, Wiley.

Seidenberg, M.S. e McClelland, J.L.
1989 *A Distributed, Developmental Model of Word Recognition and naming*, in «Psychological Review», vol. 96, pp. 523-568.

Sejnowski, T.J. e Rosenberg, C.R.
1987 *Parallel Networks that Learn to Pronounce English Text*, in «Complex Systems», vol. 1, pp. 145-168.

Sejnowski, T.J., Koch, C. e Churchland, P.S.
1988 *Computational Neuroscience*, in «Science», vol. 241, pp. 1299-1306.

Sethi, I.K. e Jain, A.K. (a cura di)
1991 *Artificial Neural Networks and Statistical Pattern Recognition. Old and New Connections*, New York, North Holland.

Shannon, C.E.
1948 *A Mathematical Theory of Communication*, in «Bell System Technical Journal», vol. 27, pp. 379-423, 623-656.
Sharda, R. e Patil, R.B.
1990 *Neural Networks as Forecasting Experts: An Empirical Test*, in *Proceedings of the International Joint Conference on Neural Networks*, San Diego, CA, vol. II, pp. 491-494.
Shepherd, G.M. (a cura di)
1997 *The Synaptic Organization of the Brain*, Oxford, Oxford University Press, 4ª ed.
Shultz, T.R. e Elman, J.L.
1994 *Analyzing Cross Connected Networks*, in *Advances in Neural Information Processing Systems 6*, a cura di J.D. Cowan, G. Tesauro e J. Alspector, San Francisco, CA, Morgan Kaufmann, pp. 1117-1124.
Sietsma, J. e Dow, R.J.F.
1988 *Neural Net Pruning – Why and how*, in *Proceedings of the IEEE International Conference on Neural Networks*, New York, IEEE Press, vol. I, pp. 325-333.
Sillito, A.M., Jones, H.E., Gerstein, G.L. e West, D.C.
1994 *Feature-Linked Synchronization of Thalamic Relay Cell Firing Induced by Feedback from the Visual Cortex*, in «Nature», vol. 369, pp. 479-482.
Singer, W.
1987 *Activity-Dependant Self-Organisation of Synaptic Connections as a Substrate of Learning*, in *The Neural and Molecular Bases of Learning*, a cura di J.P. Changeux e M. Konishi, London, Wiley.
1990 *Search for Coherence: A Basic Principle of Cortical Self-Organization*, in «Concepts in Neuroscience», vol. 1, pp. 1-26.
Singer, W. e Gray, C.M.
1995 *Visual feature integration and the temporal correlation hypothesis*, in «Annual Review of Neuroscience», vol. 18, pp. 555-586.
Smith, S.W.
1999 *The Scientist and Engineer's Guide to Digital Signal Processing*, San Diego, CA, California Technical Publishing, 2ª ed.
Solla, S.A.
1989 *Learning and Generalization in Layered Neural Networks: The Contiguity Problem*, in *Neural Networks from Models to Applications*, a cura di L. Personnaz e G. Dreyfus, Paris, IDSET.
Stanton, P.K. e Sejnowski, T.J.
1989 *Associative Long-Term Depression in the Hippocampus Induced by Hebbian Covariance*, in «Nature», vol. 339, pp. 215-218.
Steels, L.
1993 *Building Agents out of Autonomous Behavior Systems*, in *The «Artificial Life» Route to «Artificial Intelligence». Building Situated Embodied Agents*, a cura di L. Steels e R. Brooks, New Haven, NJ, Lawrence Erlbaum.

1994 *The Artificial Life Roots of Artificial Intelligence*, in «Artificial Life», vol. 1, pp. 75-110.

Stent, G.S.
1973 *A Physiological Mechanism for Hebb's Postulate of Learning*, in «Proceedings of the National Academy of Sciences USA», vol. 70, pp. 997-1001.

Sutton, R.S.
1988 *Learning to Predict by the Methods of Temporal Differences*, in «Machine Learning», vol. 3, pp. 9-44.

Sutton, R.S. e Barto, A.G.
1998 *Reinforcement Learning: An Introduction*, Cambridge, MA, MIT Press-Bradford Books.

Syswerda, G.
1989 *Uniform Crossover in Genetic Algorithms*, in *Proceedings of the Third International Conference on Genetic Algorithms*, a cura di J.D. Schaffer, San Mateo, CA, Morgan Kaufmann.

Szu, H.
1986 *Fast Simulated Annealing*, in *Neural Networks for Computing*, a cura di J.S. Denker, New York, American Institute of Physics, pp. 420-425.

Taube, J.S., Muller, R.U. e Ranck, J.B. Jr.
1990 *Head-Direction Cells Recorded from the Postsubiculum in Freely Moving Rats. I. Description and quantitative analysis*, in «Journal of Neuroscience», vol. 10, pp. 420-435.

Tepedelenliogu, N., Rezgui, A., Scalero, R. e Rosario, R.
1991 *Fast Algorithms for Training Multilayer Perceptrons*, in *Neural and Intelligent Systems Integration*, a cura di B. Soucek e il gruppo IRIS, New York, Wiley and Sons.

Tesauro, G. e Janssens, B.
1988 *Scaling Relationships in Back-Propagation Learning*, in «Complex Systems», vol. 2, pp. 39-44.

Thodberg, H.H.
1991 *Improving Generalization of Neural Networks through Pruning*, in «International Journal of Neural Systems», vol. 1, pp. 317-326.

Thorndike, E.L.
1911 *Animal Intelligence*, Darien, CT, Hafner.

Treves, A., Miglino, O. e Parisi, D.
1992 *Rats, Nets, Maps, and the Emergence of Place Cells*, in «Psychobiology», vol. 20, pp. 1-8.

van Essen, D.C. e Maunsell, J.H.R.
1983 *Hierarchical Organization and Functional Streams in the Visual Cortex*, in «Trends in Neuroscience», settembre, pp. 370-375.

Virasoro, M.A.
1989 *Categorization and Prosopagnosia in a Neural Network*, in «Physics Report», vol. 24, p. 301.

Voltolina, M. e Umiltà, C.
1994 *Un modello computazionale dell'effetto Stroop sottoposto a stress*, in «Giornale Italiano di Psicologia», vol. 20, pp. 1-8.

von der Malsburg, C.
1973 *Self-Organization of Orientation Sensitive Cells in the Striate Cortex*, in «Kybernetik», vol. 14, pp. 85-100, ristampato in Anderson e Rosenfeld [1988].
1990 *Network Self-Organization*, in *An Introduction to Neural and Electronic Networks*, a cura di S.F. Zorneter, J.L. Davis e C. Lau, San Diego, CA, Academic Press, pp. 421-432.

von Lehman, A., Paek, G.E., Liao, P.F., Marrakchi, A. e Patel, J.S.
1988 *Factors Influencing Learning by Back-propagation*, in *Proceedings of the IEEE International Conference on Neural Networks*, New York, IEEE Press, vol. I, pp. 335-341.

Wang, C., Vengatesh, S.S. e Judd, J.S.
1993 *Optimal Stopping and Effective Machine Complexity in Learning*, in *Proceedings of NIPS 93*, San Mateo, CA, Morgan Kaufmann.

Watkins, C.J.C.H.
1989 *Learning from Delayed Rewards*, tesi di dottorato, King's College, Cambridge.

Watt, R.
1991 *Understanding Vision*, London, Academic Press.

Weigend, A.S., Rumelhart, D.E. e Huberman, B.A.
1990 *Predicting the Future: A Connectionist Approach*, in *Proceedings of the 1990 Connectionist Models Summer School*, a cura di T. Sejnowski, G. Hinton e D. Touretzky, San Mateo, CA, Morgan Kaufmann, pp. 105-116.

Weiss, G.
1990 *Combining Neural and Evolutionary Learning: Aspects and Approaches*, rapporto tecnico FKI-132-90, Institut für Informatik, Technische Universität, München.

Werbos, P.
1974 *Beyond Regression: New Tools for Prediction and Analysis of Behavioral Sciences*, tesi di dottorato, Harvard University.

White, H.
1988 *Economic Prediction Using Neural Networks: The Case of IBM Daily Stock Returns*, in *Proceedings of the IEEE International Conference on Neural Networks*, a cura di M. Caudill e C. Butler, New York, IEEE Press, vol. II, pp. 451-458.

Whitley, D.
1988 *GENITOR: A Different Genetic Algorithm*, in *Proceedings of the Rocky Mountain Conference on Artificial Intelligence*, Denver, CO.

Whitley, D., Dominic, S. e Das, R.
1991 *Genetic Reinforcement Learning with Multilayer Neural Networks*, in *Proceedings of the Fourth International Conference on Genetic Algorithms*, a cura di R.K. Belew e L.B. Booker, San Mateo, CA, pp. 562-569.

Whitley, D., Starkweather, T. e Bogart, C.
1990 *Genetic Algorithms and Neural Networks: Optimizing Connections and Connectivity*, in «Parallel Computing», vol. 14, pp. 347-361.
Widrow, B. e Hoff, M.E.
1960 *Adaptive Switching Circuits*, in *IRE WESCON Convention Record*, vol. IV, pp. 96-104, ristampato in Anderson e Rosenfeld [1988].
Widrow, B. e Stearns, S.D.
1985 *Adaptive Signal Processing*, Englewood Cliffs, NJ, Prentice Hall.
Widrow, B., Winter, R.G. e Baxter, R.A.
1987 *Learning Phenomena in Layered Neural Networks*, in *IEEE First International Conference on Neural Networks*, a cura di M. Caudill e C. Butler, New York, IEEE Press, vol. II, pp. 411-429.
Williams, R.J. e Zipser, D.
1989 *A Learning Algorithm for Continually Running Fully Recurrent Neural Networks*, in «Neural Computation», vol. 1, pp. 270-280.
Willshaw, D.J. e von der Malsburg, C.
1976 *How Patterned Neural Connections Can Be Set Up by Self-Organization*, in «Proceedings of the Royal Society of London B», vol. 194, pp. 431-445.
Willshaw, D.J. e Dayan, P.
1990 *Optimal Plasticity from Matrix Memories: What goes up Must Come down*, in «Neural Computation», vol. 2, pp. 85-93.
Wilson, G.V. e Pawley, G.S.
1988 *On the Stability of the Travelling Salesman Problem Algorithm of Hopfield and Tank*, in «Biological Cybernetics», vol. 58, pp. 63-70.

Yamauchi, B.M. e Beer, R.D.
1994 *Sequential Behavior and Learning in Evolved Dynamical Neural Networks*, in «Adaptive Behavior», vol. 2, pp. 219-246.
Yao, X.
1993 *A Review of Evolutionary Artificial Neural Networks*, in «International Journal of Intelligent Systems», vol. 4, pp. 203-222.
1999 *Evolving Artificial Neural Networks*, in «Proceedings of the IEEE», vol. 87, pp. 1423-1447.

Zemel, R.S. e Hinton, G.E.
1994 *Developing Topographic Representations by Minimizing Description Length*, in *Advances in Neural Information Processing Systems 6*, a cura di J.D. Cowan, G. Tesauro e J. Alspector, San Francisco, Morgan Kaufmann.
Zipser, D.
1986 *Programming Neural Nets to do Spatial Computations*, rapporto tecnico 8086, Institute of Cognitive Science, University of California at San Diego.
Zipser, D. e Andersen, R.A.
1988 *A Back-Propagation Programmed Network that Simulates Response Properties of a Subset of Posterior Parietal Neurons*, in «Nature», vol. 331, pp. 679-684.

Le reti neurali sono sistemi di elaborazione dell'informazione i cui meccanismi di funzionamento si ispirano ai circuiti nervosi biologici. Le reti neurali, grazie alle loro caratteristiche per molti versi simili a quelle dei sistemi nervosi biologici, sono state protagoniste di una vera e propria rivoluzione nel modo di concepire e studiare la mente e nella realizzazione di sistemi di intelligenza artificiale. Da un lato, esse forniscono al ricercatore uno strumento per verificare le proprie ipotesi costruendo modelli adattivi della mente sul proprio calcolatore, dall'altro, hanno fatto il loro ingresso anche in una vasta serie di applicazioni pratiche in ambiti che spaziano dall'informatica, all'ingegneria, alla medicina. Questo volume, qui presentato in una nuova edizione, introduce alla comprensione del funzionamento delle reti neurali e fornisce inoltre, grazie all'ausilio di numerose illustrazioni ed esempi pratici, gli elementi necessari per la simulazione e la sperimentazione di svariati tipi di modelli principali.

Indice del volume: Prefazione. - Introduzione. - I. Fondamenti. - II. Apprendimento supervisionato. - III. Reti neurali e meccanica statistica. - IV. Auto-organizzazione. - V. Modelli neurali dinamici. - VI. Algoritmi genetici e reti neurali. - Appendice: Elementi di algebra lineare. - Riferimenti bibliografici.

Orazio Miglino è professore in Sistemi adattivi bio-ispirati presso il Politecnico Federale di Losanna (EPFL). Le sue attività di ricerca riguardano le reti neurali, la vita artificiale e la robotica evolutiva.

Claudio Mattiussi si è occupato della formulazione discreta e risoluzione numerica dei problemi di campo, e dell'applicazione dei metodi evolutivi alla progettazione elettromagnetica. Svolge attività di ricerca nel campo della robotica evolutiva e dei sistemi adattivi presso il Politecnico Federale di Losanna (EPFL).

ISBN 88-15-08504-1
9788815085047

€ 23,50