UNIVERSITÀ DEGLI STUDI DE L'AQUILA

Facoltà di Scienze mm.ff.nn.

Tesi di laurea in

Scienze dell'Informazione

Conseguenze delle limitazioni sui pesi nelle Reti Neurali Artificiali Multistrato Feed-Forward

Il candidato

Luca Sponta

(matr. 114256)

. . . . . .

I relatori

Prof. Giovanni Russo

. . . . . .

Dott. Gabriele Di Stefano

. . . . . .

Anno Accademico 1994-95

Indice dei contenuti

Elenco delle Figure

Elenco delle Tavole

Introduzione

Circa dieci anni fa, alcuni sviluppi teorici di notevolissima importanza hanno contribuito in modo determinante a far rinascere l'interesse verso le reti neurali. Oggi questi sistemi possono considerarsi un fatto assodato.

I successi ottenuti nel campo applicativo, sia industriale che commerciale, bastano a giustificare il crescente favore di cui godono le reti neurali, tanto che oggi non sembra più possibile considerarle una soluzione di seconda scelta rispetto ai modelli di calcolo più affermati o tradizionali. Esse, piuttosto, possono costituire un modo nuovo di affrontare i problemi, certamente complementare, o addirittura da preferire laddove le questioni non siano formulabili in modo rigoroso, oppure siano di complessità tale da richiedere tempi di calcolo enormi per una soluzione che sia esaustiva.

I sistemi neurali possiedono, in effetti, diverse caratteristiche che li rendono interessanti:

- la possibilità di adeguarsi a situazioni che si evolvano nel tempo, sia direttamente che attraverso un periodo di riaddestramento;
- la capacità di trattare dati incompleti, incerti o "rumorosi", caratteristica questa che li rende adatti in tutte quelle applicazioni che elaborano forme,

figure, segnali;

- lo scadimento progressivo delle prestazioni in funzione della quantità e della qualità dei dati disponibili; questa "degradazione dolce" è certamente difficile da ottenere con altri approcci come, ad es., i Sistemi Esperti;
- la "robustezza", cioè la tolleranza ai guasti e alle disfunzioni, indispensabile per applicazioni che debbano lavorare a lungo in modo autonomo, senza possibilità di riparazione o di controllo remoto;
- la loro natura "parallela" (come quella del sistema nervoso a cui si ispirano), che li propone, tra l'altro, come modello per i futuri calcolatori a parallelismo esteso.

Accanto a questi indubbi vantaggi vanno, tuttavia, ricordati alcuni punti deboli dei sistemi neurali, come:

- la scarsa precisione, che ne esclude, tra l'altro, l'impiego nelle applicazioni tecnico-scientifiche o di contabilità;
- i lunghi periodi di apprendimento richiesti;
- la mancanza di trasparenza, cioè l'impossibilità di capire in che modo le rete giunga alle sue conclusioni, essendo la "conoscenza" della rete stessa distribuita su tutti gli elementi che la compongono e non localizzata, ad es., in regole, come avviene nei Sistemi Esperti.

Non ultima, tra le lacune, va considerata la mancanza di una solida teoria sui sistemi neurali, che sia di supporto nell'analisi e nella sintesi delle reti o nella

scelta della topologia più adatta per un determinato problema. Lacuna che, se colmata, potrebbe avere ulteriori notevoli riflessi in campo applicativo.

Gran parte della ricerca attuale sulle reti neurali ed un numero non trascurabile di "neuro-applicazioni" fanno uso di simulatori software su computers "seriali".

Il maggior vantaggio offerto dai neuro-simulatori è sicuramente la loro flessibilità: possono essere utilizzati per diverse topologie di reti (reti multistrato, reti di Hopfield, mappe di Kohonen, etc. [2]); parametri, strati, numero dei neuroni per strato, possono essere facilmente rivisitati secondo i desideri dell'utente. Non è da sottovalutare, inoltre, la possibilità di utilizzare interfacce grafiche avanzate ed amichevoli, di cui i simulatori fanno ormai largo uso.

D'altro canto, l'inerente parallelismo del paradigma neurale non trova riscontro nell'accoppiata *simulatore software-computer seriale*.
La scarsa aderenza di questa "architettura" al modello connessionista, se da un lato non penalizza la ricerca, la quale da sempre si è misurata con problemi di riferimento di portata limitata, ma anzi la favorisce in virtù della diffusione attuale di calcolatori e neuro-simulatori, comporta dall'altro un evidente scadimento delle prestazioni nel supporto di quelle neuro-applicazioni che nascono per risolvere problemi reali complessi.
Per far sì che reti aventi centinaia di neuroni e migliaia di sinapsi, alle quali debbano essere sottoposti migliaia di esempi durante la fase di apprendimento, siano in grado di generalizzare in contesti di una certa complessità, e non soltanto di memorizzare i dati di *training*, si richiede l'esecuzione di un numero enorme di calcoli in un tempo relativamente breve, cosa che rende i simulatori sostanzialmente inadeguati.

Ricorrere allora ad implementazioni hardware dei sistemi neurali, che ne sfruttino l'intrinseco parallelismo, diventa quasi una necessità.

Le implementazioni hardware garantiscono, rispetto ai neuro-simulatori, incrementi nella velocità di risposta di un fattore $10^6/10^9$. Tuttavia sono soggette a vincoli tecnologici che ne influenzano le prestazioni complessive.

Attualmente vengono seguite diverse strade nella progettazione dei chip neurali; le implementazioni esistenti possono essere suddivise in quattro gruppi:

1. implementazioni analogiche;
2. implementazioni digitali;
3. implementazioni ibride (analogico/digitali);
4. implementazioni opto-elettroniche.

Ciascuna delle filosofie implementative ha peculiarità diverse e diversi punti deboli, tanto che a tutt'oggi non esiste un consenso sul miglior approccio realizzativo. Tutte le strategie devono sottostare comunque ad alcune restrizioni (oltre a quelle legate ai limiti della tecnologia utilizzata), come:

- la necessità di limitare e/o quantizzare i valori dei parametri caratterizzanti la rete (siano essi implementati in termini di correnti e tensioni oppure come celle di n bit);
- la possibilità di poter collocare all'interno di un singolo chip "neurale" solo un certo numero di unità di calcolo.

Queste limitazioni, come già detto, non sono indolori, ma hanno piuttosto dei notevoli riflessi sulle prestazioni delle reti hardware.

Il numero limitato di neuroni attualmente presenti su di un singolo integrato, conduce la ricerca alla progettazione di chip "componibili", che consentano di tracciare in maniera dinamica topologie di reti complesse.
Inoltre, l'impossibilità di implementare "on-chip" il canonico algoritmo di apprendimento di Error Back Propagation, per via delle limitazioni imposte ai parametri, comporta la necessità di individuare algoritmi di apprendimento che siano facilmente integrabili e compensino, per quanto possibile, le approssimazioni computazionali introdotte dalla tecnologia in uso.
Tuttavia, e la questione sembra più importante, sono le stesse capacità intrinseche della rete, come vedremo nel corso del lavoro, ad essere penalizzate dalle restrizioni imposte ai parametri (tipicamente i pesi delle connessioni), a prescindere da quale sia l'algoritmo usato durante la fase di apprendimento.

La tesi si propone di investigare sulle ricadute che l'introduzione di vincoli sui valori dei parametri ha sul comportamento dei sistemi neurali.
L'obiettivo della tesi è andato definendosi sulla base di alcune osservazioni effettuate simulando al calcolatore il comportamento di un chip neurale in fase di sviluppo presso il dipartimento di Ingegneria Elettrica dell'Università dell'Aquila.
In origine ci si era proposti di elaborare un algoritmo di apprendimento "off-line" per il chip in esame, mancando lo stesso di un circuito di apprendimento. Il progetto iniziale del chip limitava i valori dei pesi nell'intervallo $[-1,\ 1]$. La funzione di trasferimento ivi integrata era non lineare e simile nel suo andamento alla più conosciuta *sigmoide*. Durante la simulazione, dalla rete venivano accettate come risposte valide soltanto uscite maggiori di 0.9 o minori di 0.1.
Ci si è subito accorti che la rete, addestrata utilizzando un algoritmo genetico,

non era in grado di risolvere il classico problema dello XOR (vedi la sezione 1.2.1), con la topologia proposta in [31]. Poiché gli algoritmi genetici sono in grado di individuare con alta probabilità una soluzione che sia almeno localmente ottimale, l'insuccesso ottenuto ci ha fatto sospettare che il mancato apprendimento della rete potesse risiedere nelle ridottesi "capacità" della stessa, a causa delle limitazioni imposte ai pesi delle connessioni, come abbiamo successivamente verificato. Ed è stato questo lo spunto da cui ha avuto inizio la nostra ricerca teorica.

Nel lavoro di tesi abbiamo fissato la nostra attenzione sulle reti feed-forward multistrato, che sono quelle più comunemente implementate "on-chip".
Dopo aver presentato alcune nozioni generali sulle reti neurali feed-forward (Cap. 1) e una breve panoramica sulle strategie di implementazione elettronica dei sistemi neurali (Cap. 2), ci addentriamo nello studio degli effetti che la limitazione (Cap. 3) e la discretizzazione (Cap. 4) dei parametri di queste reti hanno sulle loro capacità di apprendimento, in particolare sulla capacità di "separare" ingressi differenti.
Ad un breve capitolo che riassume i risultati ottenuti ed indica alcuni possibili sviluppi del lavoro di tesi, seguono delle appendici.
Nella prima abbiamo inserito dei richiami di matematica utili alla comprensione generale del testo.
Le appendici B e C costituiscono invece parte integrante del lavoro svolto.
Nell'appendice B si trovano le dimostrazioni di alcuni risultati di una certa rilevanza. Le dimostrazioni sono piuttosto articolate: abbiamo preferito non inserirle nel corpo del lavoro per garantire una continuità di lettura. Nell'appendice C, infine, viene riportata la descrizione dei punti salienti del programma scritto ed utilizzato durante la fase sperimentale del nostro studio.

Parte I

Generalità

Capitolo 1

Le Reti Neurali

1.1 Una Panoramica

Le Reti Neuronali[1] Artificiali (ANN), indicate anche come Reti Neurali o sistemi connessionistici, possono essere pensate come costituite da unità di calcolo molto semplici tra loro connesse. Ogni unità possiede un grado di attivazione che dipende dai segnali ricevuti attraverso le connessioni. Questi segnali sono mediati da pesi, che vengono associati alle connessioni che li propagano: maggiore è il peso, più intenso sarà il segnale ricevuto dall'unità. Ogni unità, quando viene "attivata", invia segnali alle altre unità cui è connessa. Le connessioni vengono utilizzate normalmente in modo asimmetrico, hanno cioè un solo verso di propagazione del segnale. Ogni tipo di rete prevede delle unità che possono ricevere segnali dall'ambiente (le unità *di input*), unità che possono inviare segnali all'esterno (le unità *di output*) ed unità interne collegate

[1]Nel seguito il termine *neuronale* farà riferimento alle reti di neuroni naturali, cioè alle cellule nervose, mentre il termine *neurale* verrà utilizzato per i sistemi artificiali.

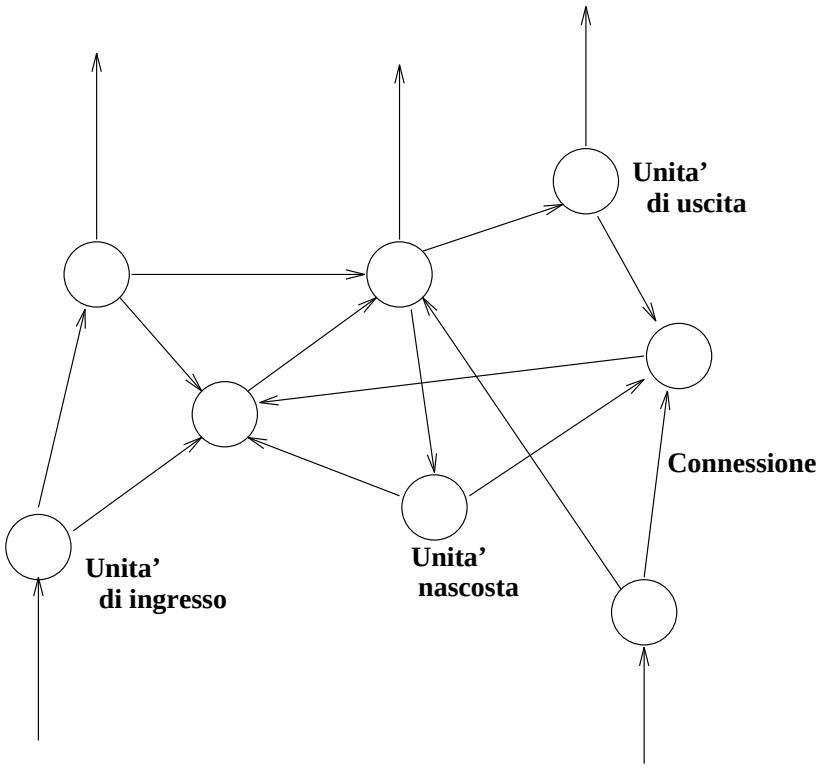

Figura 1.1: *Una rete neurale.*

solo ad altre unità (le unità *nascoste*).

Le reti sono in grado di svolgere compiti diversi ma soprattutto posseggono la notevole capacità di apprendere. Ciò significa che esse sono in grado di modificare i parametri che le caratterizzano (generalmente i pesi delle connessioni ed i parametri che regolano il funzionamento delle singole unità) aumentando gradualmente le proprie prestazioni fino a raggiungere quelle desiderate. Al termine della fase di apprendimento una rete è in grado di eseguire il compito per cui è stata addestrata.

Un'altra interessante caratteristica che avvicina i sistemi connessionistici ai modelli biologici, riguarda le modalità di rappresentazione dell'informazione. Questa infatti non viene immagazzinata in maniera simbolica o "discreta", come avviene ad esempio nei sistemi di Intelligenza Artificiale, ma in modo ana-

logico (si pensi ai parametri che caratterizzano il funzionamento di una rete). Spesso si usa dire che le informazioni in una rete sono a livello "sub-simbolico" per indicare che non sono chiaramente osservabili in un singolo neurone o in gruppi di neuroni ma, essendo distribuite su tutta la rete, possono essere messe in evidenza soltanto attraverso l'attività della rete stessa. È questa informazione distribuita ad essere modificata e "riorganizzata" durante il processo di apprendimento, finché la rete non sia in grado di rispondere correttamente alle sollecitazioni in ingresso.

Riassumendo, una rete neurale con N neuroni può essere descritta dai seguenti parametri [31, 2]:

1. un insieme di unità di calcolo $u_j,\ j = 1, \ldots, N$ (neuroni), capaci di svolgere semplici elaborazioni;
2. il loro stato di attivazione a_j;
3. lo schema delle connessioni o topologia (è possibile descriverlo attraverso un grafo orientato od una matrice);
4. la regola di propagazione, che lega le attività in ingresso ad un neurone con i pesi delle rispettive connessioni;

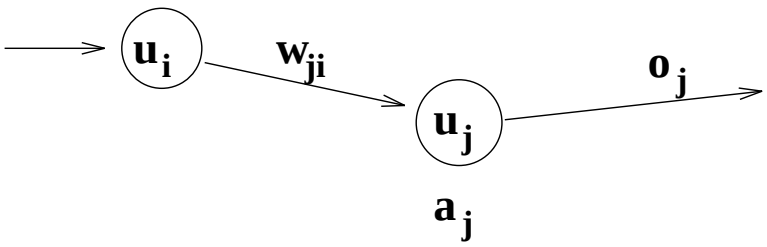

Figura 1.2: *Un neurone.*

5. la regola di attivazione, che combina gli ingressi afferenti a ciascuna unità con il suo stato di attivazione precedente;

6. la funzione di trasferimento $o_j : \mathbb{R} \longrightarrow \mathbb{R}$ (detta anche *funzione di attivazione* o *di uscita*) di ogni unità, atta a definire lo stato di attivazione dell'unità all'"istante" successivo;

7. una regola di apprendimento per modificare i pesi delle connessioni e gli altri parametri della rete.

Essendo ormai molti i lavori di tipo introduttivo disponibili in letteratura sul mondo delle reti neurali (vedere ad es. [2, 25]), eviteremo di ripercorrere in maniera approfondita quella che è stata l'evoluzione storica dei sistemi connessionistici, soffermandoci solo su quei modelli che abbiamo ritenuto più vicini ai temi da noi trattati.

1.2 Le Reti Feed-Forward

Nelle reti *feed-forward* i neuroni sono disposti per "strati". Agli strati di ingresso (o di *input*) e di uscita (o di *output*) possono aggiungersi uno o più strati intermedi di neuroni *nascosti*. Ciascun neurone è connesso a tutte le unità degli strati adiacenti con legami di peso variabile. Non esistono collegamenti tra neuroni dello stesso strato, e le connessioni vengono utilizzate in maniera unidirezionale, dall'ingresso verso l'uscita; da qui il nome di reti "in avanti".

1.2.1 Il Percettrone

Il primo modello di rete avente topologia feed-forward in grado di apprendere fu proposto nel 1958 da F. Rosenblatt [28], e venne da lui denominato "Percettrone". Anche se il Percettrone aveva in origine una topologia leggermente diversa (per una descrizione della quale rimandiamo a [2], pag. 74 e seguenti), per i nostri fini analizzeremo un modello ad esso equivalente, che riportiamo in figura 1.3. Il modello, che chiameremo "percettrone esteso" (con la lettera minuscola, per distinguerlo dal Percettrone di Rosenblatt), prevede due strati di neuroni, uno di ingresso e uno di uscita, ciascuno composto da più unità. Gli n segnali in ingresso, ricevuti dai neuroni del primo strato, vengono pesati e sommati tra loro secondo la regola di propagazione

$$net_j = \sum_{i=1}^{n} w_{ji} x_i, \qquad j = n+1, \ldots, N \tag{1.1}$$

dove net_j rappresenta l'ingresso totale al neurone di uscita u_j mentre x_i e w_{ji} sono rispettivamente i segnali in ingresso ed i pesi relativi al neurone.

La regola di attivazione è data da:

$$a_j = net_j - \theta_j, \qquad j = n+1, \ldots, N \tag{1.2}$$

dove θ_j è un valore costante detto *soglia del neurone*. La (1.2) descrive come varia l'attivazione del neurone in funzione dei segnali ad esso afferenti.

La funzione di trasferimento dei neuroni di uscita è così definita:

$$o_j = \begin{cases} 1 & a_j > 0 \\ 0 & a_j \leq 0 \end{cases} \tag{1.3}$$

Questa è la funzione comunemente denominata *funzione a gradino* o *a soglia*.

Per quanto riguarda la procedura di apprendimento, le configurazioni (*patterns*) da classificare vengono presentate al percettrone in sequenza, insieme

ai valori desiderati in uscita. I pesi delle connessioni vengono modificati se i neuroni di output non classificano correttamente un pattern, cioè se la risposta ottenuta è diversa da quella desiderata. Il processo di modifica dei pesi si conclude quando tutti i patterns sono stati classificati esattamente. A tal proposito Rosenblatt ha dimostrato che il processo di modifica ha necessariamente termine se il problema di classificazione affrontato è *linearmente separabile* (vedremo nel seguito della sezione cosa questo significhi).

La legge di apprendimento prevista dal modello di Rosenblatt, nota come *regola delta*, è data da:

$$\Delta w_{ji} = (t_j - o_j) \cdot o_i \tag{1.4}$$

dove w_{ji} è il peso associato alla connessione tra il neurone j-esimo del secondo livello e l' i-esimo neurone del primo. o_i è l'uscita dell'i-esimo neurone di input, ossia la i-esima componente del vettore di ingresso, mentre t_j sta ad indicare l'uscita desiderata per il j-esimo neurone di output. In sostanza la regola dice

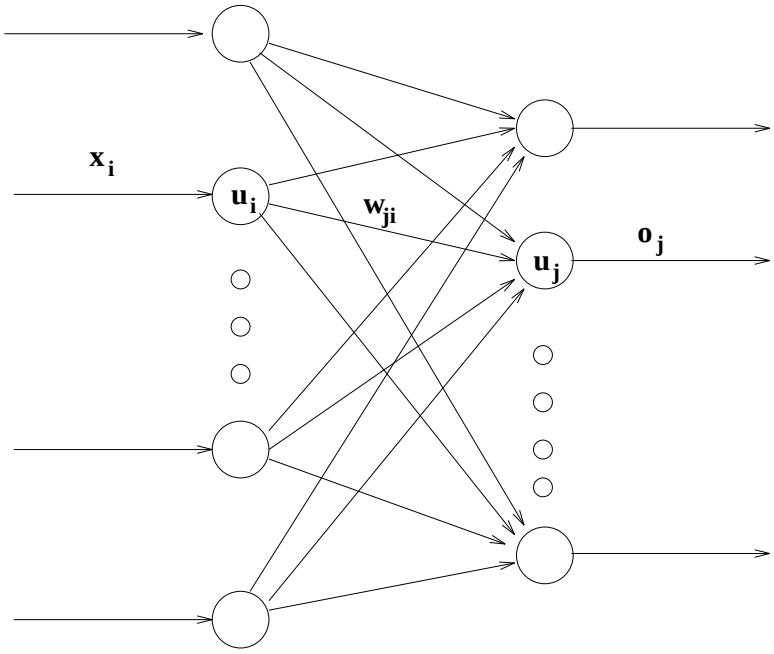

Figura 1.3: *Il percettrone esteso.*

che, per ottenere i pesi all'istante $t + 1$, ai pesi dell'istante t devono essere sottratti i valori dei rispettivi ingressi qualora il pattern di input venga erroneamente riconosciuto come appartenente alla classe da discriminare, oppure sommati se al contrario, il vettore in input non venga "individuato"; nel caso in cui il riconoscimento sia corretto, i pesi rimarranno invariati.

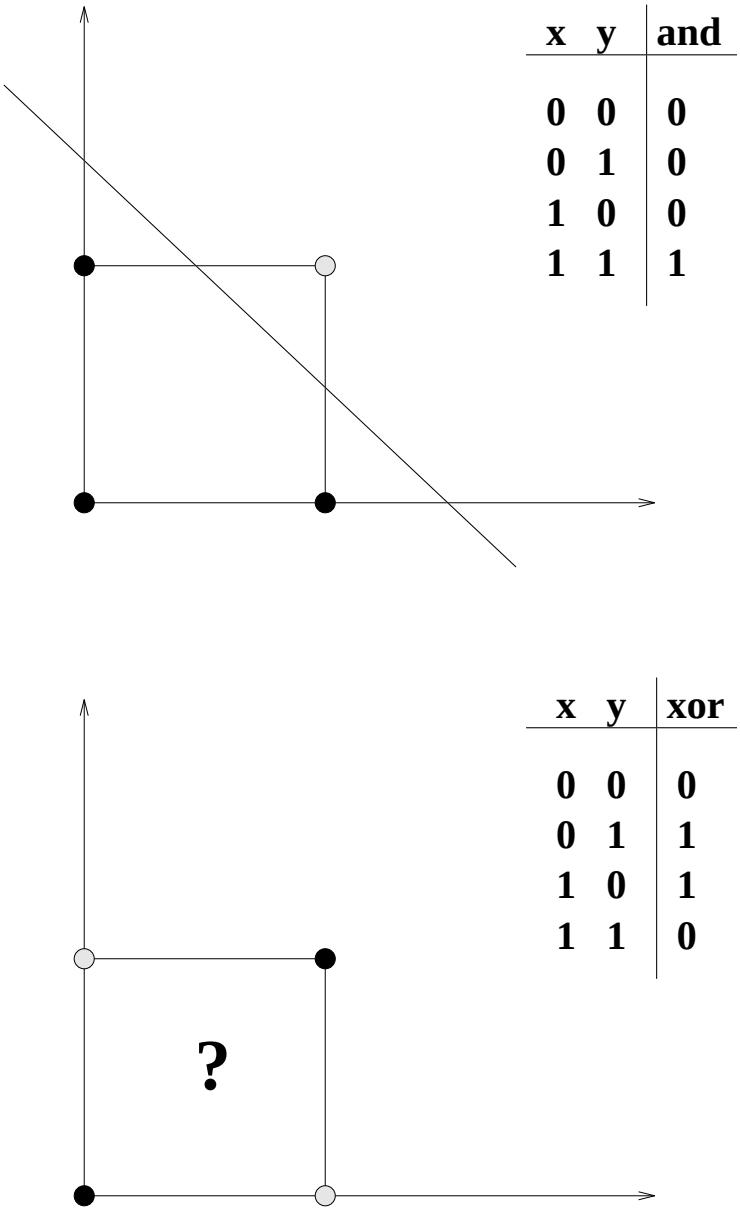

Figura 1.4: *Lo XOR non è un problema linearmente separabile: infatti non è possibile separare con una retta gli "zeri" dagli "uni".*

Le dure critiche mosse al Percettrone da M. Minsky ed S. Papert in un famoso lavoro del 1969, "Perceptrons" [21], ne mettono in evidenza i limiti sostanziali; questi sono dovuti principalmente al fatto che il modello prevede due soli strati di neuroni, quelli di ingresso e di uscita.
Se chiamiamo *spazio di input* lo spazio vettoriale di tutti i possibili ingressi alla rete si può dimostrare che il Percettrone è in grado di distinguere tra due classi di segnali in ingresso solo se i corrispondenti vettori giacciono in regioni dello spazio di input separabili con un iperpiano. Tutti i problemi di classificazione di questo tipo vengono detti *linearmente separabili* e, come abbiamo già detto, sono gli unici che un Percettrone possa risolvere.

Il Percettrone, ad es., è in grado di implementare correttamente la funzione logica AND, che risulta essere un problema linearmente separabile. Infatti, come è facile capire se si affronta la questione geometricamente, è possibile separare con un iperpiano (in questo caso una retta) gli ingressi che valgono 0 da quelli che valgono 1 (figura 1.4). Il Percettrone tuttavia non riesce a risolvere lo XOR che, pur essendo in apparenza di complessità simile all'AND, non è però linearmente separabile; per quanto ci si sforzi infatti, non si riuscirà ad individuare una retta che separi i punti di classe 0 da quelli di classe 1. Per queste sue caratteristiche lo XOR è diventato uno dei problemi canonici nell'ambito della ricerca sulle reti neurali, assurgendo a paradigma dei limiti del Percettrone.

1.2.2 Il Percettrone Multistrato

Se il Percettrone avesse avuto uno strato intermedio di neuroni, avrebbe potuto separare punti appartenenti a regioni convesse dello spazio di input; se

ne avesse avuti due sarebbe stato in grado di classificare ingressi appartenenti anche a regioni non convesse e non connesse dello spazio (vedi la figura 1.5). Ci si può domandare, allora, perché Rosenblatt non si sia rivolto a percettroni multistrato, viste le loro maggiori potenzialità e considerato che lui stesso ne era consapevole. Il fatto è che non era ancora nota una procedura di apprendimento che permettesse di modificare i pesi delle connessioni anche per gli strati di neuroni che non fossero adiacenti a quello di output.

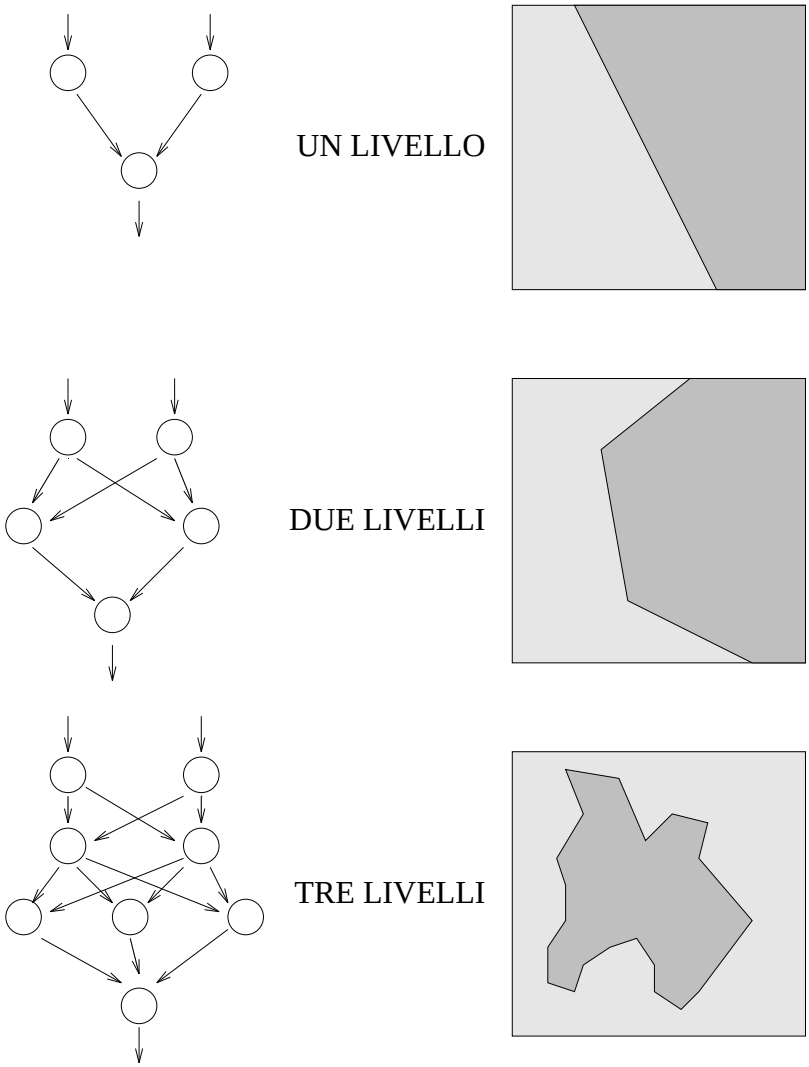

Figura 1.5: *Aumentando nel percettrone il numero degli strati nascosti è possibile riconoscere regioni (in questo caso del piano) di complessità crescente.*

Per utilizzare infatti percettroni a più livelli senza il conforto di una procedura di apprendimento, sarebbe comunque necessario individuare in altra maniera i valori dei pesi per tutte le connessioni della rete, cosa che, se rimane possibile per problemi "giocattolo" come quello dello XOR, diventa impensabile per problemi appena complessi.

La svolta, dopo anni di stagnazione, si è avuta nel 1986 quando, nel libro "Parallel Distributed Processing" [31], Rumelhart, Hinton e Williams hanno proposto l'ormai celebre algoritmo EBP, di Error Back Propagation, per l'apprendimento dei percettroni multistrato.
L'idea dell'algoritmo è quella di modificare i pesi delle connessioni in modo da minimizzare una funzione di errore E che calcola in modo opportuno la distanza tra l'uscita della rete e quella desiderata. Dopo aver calcolato la derivata parziale della funzione di errore rispetto al peso che si vuole modificare, si aumenta (o si diminuisce) quest'ultimo a seconda che la derivata sia negativa (o positiva). Facendo diminuire il valore della funzione di errore, si riduce in

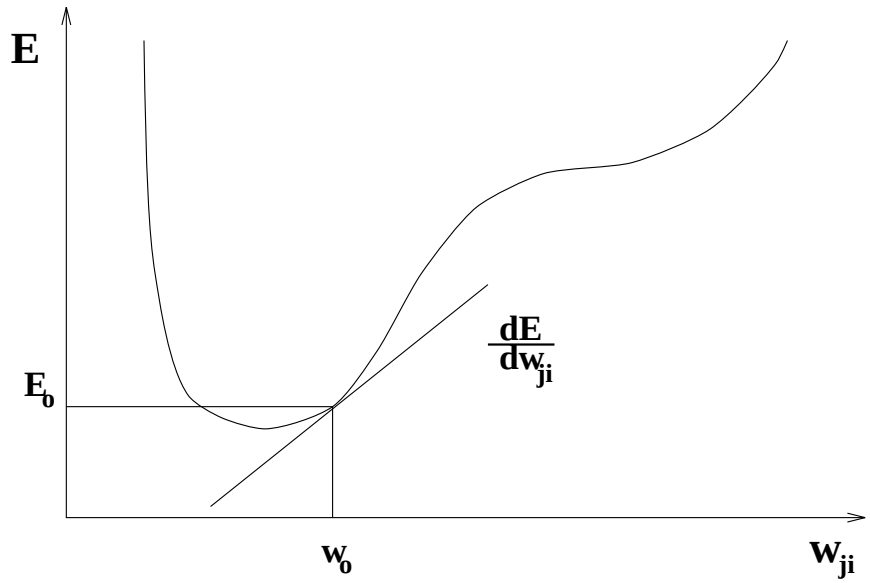

Figura 1.6: *Una derivata positiva richiede che il peso venga diminuito.*

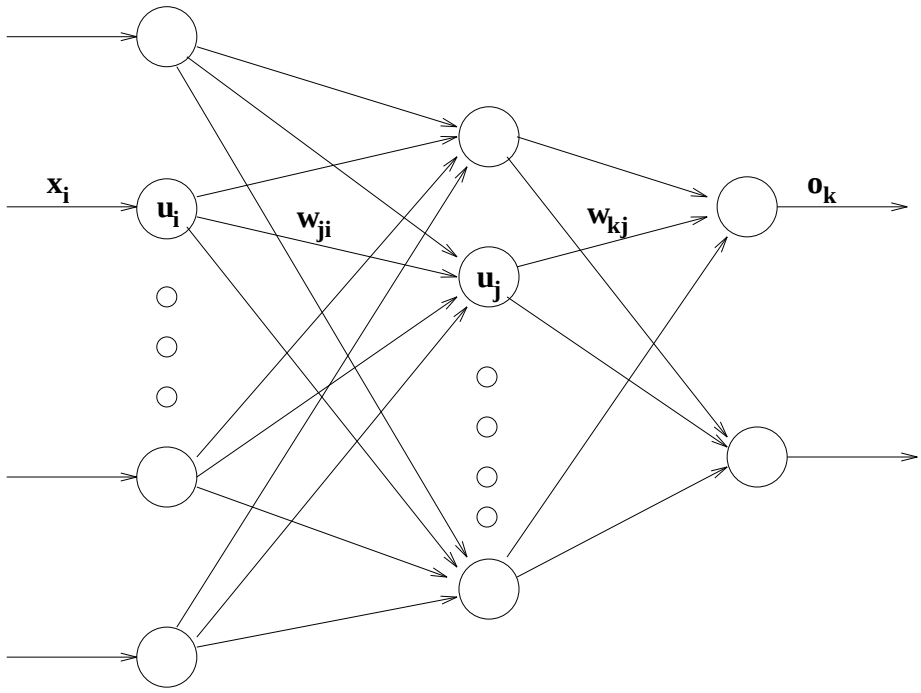

Figura 1.7: *Il percettrone multistrato.*

questo modo la distanza tra l'uscita aspettata e quella ottenuta. La modifica dei pesi avviene a ritroso, risalendo i livelli delle connessioni a partire da quello di output.

Nel modello proposto da Rumelhart, Hinton e Williams [30], i neuroni sono descritti dalle seguenti equazioni:

$$net_j = \sum_i w_{ji} x_i - \theta_j \tag{1.5}$$

$$a_j = net_j \tag{1.6}$$

Il modello prevede inoltre, affinché l'algoritmo di apprendimento EBP possa essere applicato, una funzione di attivazione che sia a valori reali, derivabile, non decrescente. Una delle funzioni più comunemente utilizzate nelle implementazioni è la *sigmoide*, avente equazione:

$$f(x) = \frac{1}{1 + e^{-x/T}} \tag{1.7}$$

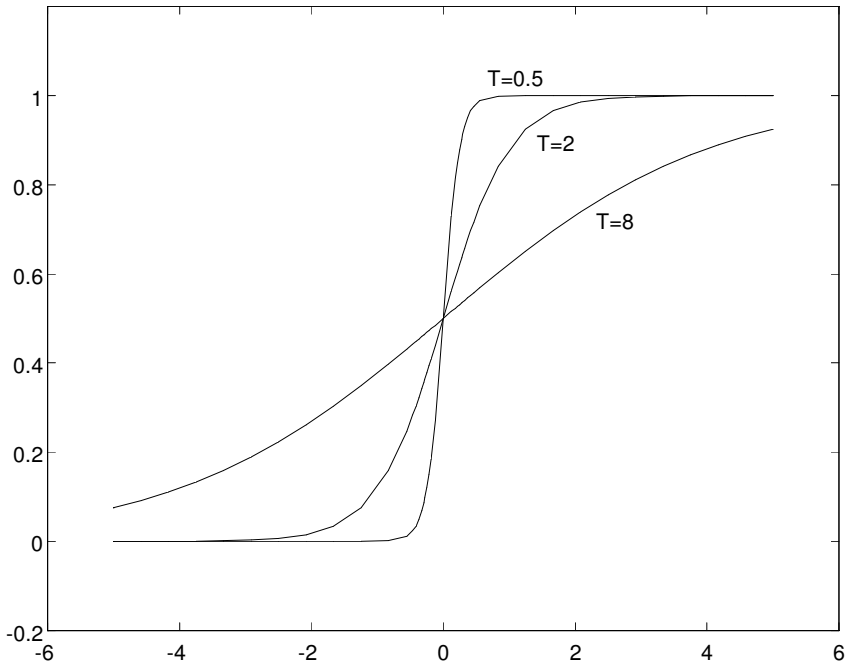

Figura 1.8: *La* sigmoide *a varie "temperature".*

Il fattore T che compare nell'esponenziale dell'equazione, denominato *temperatura*, contribuisce ad individuare la pendenza della curva (figura 1.8).

Non esiste, per le reti addestrate con l'EBP, un teorema di convergenza come per il Percettrone di Rosenblatt; inoltre non esiste, in caso di convergenza dell'algoritmo, la garanzia che il minimo trovato per la funzione di errore sia un minimo globale e non piuttosto locale; non si può essere certi cioè che la rete, qualora abbia appreso, lo abbia fatto nel modo migliore.

Nella pratica si è visto che, aumentando in maniera adeguata le dimensioni della rete, le probabilità che l'algoritmo non converga ad un minimo accettabile diminuiscono.

L'algoritmo EBP, nonostante alcuni punti deboli come la velocità di conver-

genza piuttosto bassa e fortemente dipendente dai valori dei parametri, può comunque considerarsi uno strumento potente, che ha già trovato numerose applicazioni, e che di fatto ha contribuito a rinnovare lo slancio e l'interesse della ricerca nei confronti dei sistemi connessionistici.

Capitolo 2

Circuiti Integrati Neurali

Le implementazioni software delle reti neurali mostrano il fianco quando cercano di affrontare problemi applicativi di un certo spessore, che richiedono topologie complesse ed un gran numero di neuroni, e questo a causa della lentezza degli algoritmi di apprendimento.
Questi algoritmi, essendo iterativi, hanno bisogno di un numero molto elevato di cicli per condurre la rete in uno stato stabile; se applicati a reti grandi possono allungare i tempi di apprendimento in maniera intollerabile, anche utilizzando i calcolatori seriali più veloci. Volendo beneficiare ancora dei vantaggi legati all'approccio "connessionista", volgere l'attenzione verso lo sviluppo di circuiti integrati neurali è sembrata una scelta doverosa.

I chip neurali consentono incrementi notevolissimi delle capacità di calcolo dei sistemi che implementano [11], non soltanto per il fatto di essere circuiti elettronici, ma anche perché sfruttano appieno l'inerente parallelismo delle reti neurali.

Quello legato alla progettazione dei circuiti integrati neurali può essere con-

siderato un campo di ricerca ormai consolidato. I "neuro-simulatori", dal canto loro, pur rivestendo ancora un ruolo primario sia nell'ambito della ricerca che da un punto di vista applicativo, stanno pure rientrando, a poco a poco, in quello che può essere considerato il loro alveo naturale. I simulatori infatti, in ogni area applicativa, sono essenzialmente degli strumenti di sviluppo utili per effettuare esperimenti, valutare architetture, verificare prestazioni, comparare le diverse scelte; quando però diventa necessario volgersi a soluzioni efficaci per un ben determinato problema applicativo, la scelta primaria ricade sulla circuiteria dedicata. Del resto, volendo azzardare un paragone che forse risulterà più familiare agli ingegneri elettronici, nessuno di essi penserebbe mai che usare SPICE[1] sia il modo più efficiente di introdurre un filtro in un sistema!

Come già osservato, le reti neurali traggono dal fatto di poter beneficiare di un hardware dedicato i vantaggi propri di qualunque sistema elettronico: la **velocità**, in virtù della minore complessità circuitale rispetto a quella dei sistemi *general purpose*; il **minore costo** in termini di dimensioni del circuito (vogliamo dire: quando la tecnologia avrà prodotto circuiti affidabili, converrà certamente di più spendere qualche centinaio di migliaia di lire per un chip neurale, che comperare un CRAY insieme a qualche programma di neuro-simulazione) e di consumi energetici.

Tuttavia i chip neurali risentono pure di tutte le limitazioni derivanti dalla tecnologia in uso, limitazioni che affliggono in diversi modi le varie parti funzionali di questi sistemi. La ricerca, nella progettazione dei circuiti neurali, ha seguito strade diverse, nell'intento di realizzare in modo ottimale ciascuna delle funzioni presenti nella "macchina neurale". Ciascuno degli approcci ha rivelato peculiarità e punti deboli.

[1] SPICE è un complesso e versatile simulatore circuitale.

Nell'offrire una panoramica delle varie filosofie di sviluppo dei chip neurali non pretenderemo di essere esaustivi, anzi porremo l'accento su quei temi che ci sembrano più direttamente connessi al nostro lavoro.

2.1 Una Classificazione

Potremmo pensare le reti neurali hardware come costituite da tre parti con differenti funzioni:

- **Unità di calcolo**, che possono eseguire operazioni semplici come la moltiplicazione dei segnali per i pesi sinaptici o come la somma per calcolare il valore dell'attivazione, oppure complesse, come il computo della funzione non lineare di trasferimento o l'aggiornamento dei pesi.
- **Unità di memorizzazione**, che immagazzinano l'informazione usata per i calcoli (tipicamente i pesi delle connessioni).
- **Strutture di interconnessione**, che permettono lo scambio di informazione tra unità di calcolo e di memorizzazione, nonché con il mondo esterno. È appena il caso di far notare come in un simulatore il problema delle interconnessioni non esista, poiché i neuroni sono tutti ugualmente "vicini" all'interno della RAM del sistema, e tutte le operazioni vengono eseguite dalla CPU.

Come del resto accade in qualunque circuito elettronico, ciascuna di queste funzioni di base potrebbe avere un modo diverso di "rappresentarsi" l'informazione che manipola.
Proviamo a classificare i circuiti neurali seguendo questo criterio:

a) **Il tempo ed i valori** (delle grandezze in gioco) **sono continui:** l'informazione è trasportata da livelli di tensione e/o corrente; non esiste una tempificazione delle operazioni tanto per ciò che concerne il calcolo quanto per la comunicazione.

b) **Il tempo è discreto, i valori sono continui:** l'informazione viene trasferita da tensioni e/o correnti che, tuttavia, diventano "valide" (e quindi possono essere utilizzate) solo ad intervalli di tempo definiti.

c) **Il tempo ed i valori sono discreti:** l'informazione è trasportata, immagazzinata e processata in forma binaria, usando *gates*, registri ed altri dispositivi logici.

Dal momento che, come abbiamo osservato, ciascuno dei sottosistemi che costituiscono un circuito neurale può rappresentare in modo diverso l'informazione, possono essere definite varie famiglie di chip neurali. Cercheremo di mettere in evidenza quali siano, almeno nelle linee generali, i benefici e gli svantaggi di ciascun approccio implementativo.

2.1.1 Circuiti Neurali Digitali

Le reti neurali completamente digitali usano circuiti logici per elaborare informazione discreta sia nel dominio delle ampiezze che in quello del tempo. Tipicamente vi troveremo circuiti ALU specializzati nel calcolo di somme-di-prodotti, banchi di memoria per memorizzare i pesi ed i risultati intermedi delle computazioni, bus seriali o paralleli a costituire le strutture di connessione tra le varie unità.

Sia il *time-sampling* che la discretizzazione dei valori dei parametri della rete

conducono ad errori e perdite di informazioni. In particolare, il campionamento dei segnali pone un *upper bound* sulla *larghezza di banda* (cioè sull'intervallo delle frequenze) del segnale elaborato dal sistema (si veda un qualunque testo che parli di teoria dei segnali); la discretizzazione di una variabile introduce, invece, un *errore di quantizzazione* $E_q = S/2^N$, dove S è il massimo valore che la variabile può assumere ed N è il numero di bits utilizzati.

Se è vero che i circuiti digitali sono affetti da entrambe i problemi sopra accennati, è pur vero che questi sono gli unici errori presenti in tali sistemi. Se decidiamo di accettarli potremo, nel contempo, beneficiare di:

- un'alta immunità ai "rumori",
- una grande precisione di calcolo,
- la possibilità di usare dispositivi standard per l'implementazione delle varie funzioni (ALU, memorie RAM, etc ...),
- l'alto sfruttamento dei componenti utilizzando tecniche di multiplexing,
- una maggiore facilità di integrazione di questi circuiti all'interno di altri sistemi digitali di elaborazione,
- un alto grado di connettività tra i vari circuiti neurali, a disegnare topologie complesse.

A ciò si aggiunge la possibilità, una volta definita l'architettura delle rete neurale, di sfruttare tecniche automatiche di progettazione digitale ormai consolidate.
Va tuttavia precisato che i circuiti digitali richiedono un maggiore spazio su chip rispetto a quelli analogici per implementare funzionalità equivalenti.

2.1.2 Circuiti Neurali Analogici

Sono quei circuiti che lavorano con segnali continui (siano essi tensioni o correnti) sempre validi nel tempo (non c'è campionamento del segnale).
Come tutti i sistemi di questo tipo, le reti analogiche sono afflitte da seri problemi:

- **il rumore**, che fissa un *lower bound* sul valore dei segnali che conducono informazione "utile"; le variazioni di segnale al di sotto di questa soglia non possono essere considerate valide;
- **errori di offset e non-linearità** dei dispositivi, che possono modificare il comportamento del circuito rispetto a quanto previsto dal progettista;
- **la capacità di reazione dei vari componenti (slew rate)**, che limita la velocità di elaborazione del segnale e ne riduce la larghezza di banda.

Tutti questi tipi di errore si accumulano ad ogni passo di elaborazione in modo irreversibile. Verrebbe allora da chiedersi quali siano le ragioni dell'interesse che pure questo tipo di approccio progettuale ai circuiti neurali suscita, vista la scarsa precisione che esso è in grado di fornire.

In realtà va tenuto presente che nelle operazioni critiche effettuate nelle reti neurali sono coinvolte un gran numero di variabili (si pensi al calcolo dell'attivazione); se gli errori introdotti sono uniformemente distribuiti su ciascuna di esse, il loro contributo al risultato finale probabilmente sarà vicino allo zero; verrebbero così ad attenuarsi non poco gli effetti negativi legati al rumore.

Una seconda ragione va ricercata nella natura stessa dei sistemi neurali. Nei sistemi "aperti" la precisione dei risultati dipende da quella di ciascuno

dei sottosistemi presenti nella catena; al contrario, in quelli dove è presente una retroazione (**feedback**) la precisione dell'intero circuito, come sa bene qualunque ingegnere, è legata essenzialmente a quella della rete di feedback. Così accade anche nei circuiti neurali che abbiano una qualche regola di apprendimento *on-chip*, visto che anch'essi possono essere considerati dei sistemi "chiusi". In questo caso infatti, mentre la rete impara ad eseguire il compito per cui viene addestrata, il circuito di apprendimento sarà in grado di compensare anche i disturbi propri del chip, i cosiddetti errori *in-the-loop* [8].

Se a tutto questo aggiungiamo il fatto che i circuiti analogici sono più veloci dei cugini digitali, oltre a richiedere un minore spazio su chip a parità di funzioni implementate, ecco che l'interesse verso i circuiti neurali analogici non appare più così ingiustificato.

2.1.3 Circuiti Neurali Ibridi

Nel tentativo di aggirare gli svantaggi propri degli approcci sopra descritti, parte della ricerca si è dedicata allo sviluppo di circuiti neurali che prevedano l'utilizzo di dispositivi sia digitali che analogici, con gli appropriati convertitori analogico/digitale/analogici a consentire la comunicazione tra le varie componenti (vedi, ad es., [3]). Questa soluzione mira ad implementare ciascuna funzione del circuito nel modo migliore.

Tipicamente questi circuiti usano dispositivi analogici (amplificatori operazionali) per realizzare le funzioni di somma o di trasferimento, mentre i pesi vengono immagazzinati in memorie binarie e convertiti in forma analogica prima di essere utilizzati per i calcoli. **La maggior parte dei circuiti integrati neurali prodotti a livello commerciale è attualmente di questo tipo**.

2.2 L'Implementazione dei Pesi

Le funzionalità di calcolo dei sistemi neurali, come il computo della funzione di uscita, hanno trovato una efficace implementazione analogica mediante l'utilizzo di svariati dispositivi (amplificatori differenziali, etc ...) con i quali, ad es., è possibile riprodurre le funzioni di trasferimento più in uso nelle simulazioni software (sigmoide, gaussiana).

Al contrario, qualunque sia la filosofia di progetto che si decida di abbracciare, **l'implementazione dei pesi su chip è soggetta a vincoli**, vincoli legati alla stessa natura "discreta", o comunque finita, dei circuiti integrati.
I neuro-simulatori, fatti salvi i limiti dovuti alla precisione di macchina, consentono ai pesi di assumere valori comunque elevati; ciò non può accadere in un circuito integrato, dove i valori delle tensioni e delle correnti devono sottostare a precise limitazioni. Inoltre, qualora si decida di utilizzare dei banchi di memoria per immagazzinare i pesi, questi potranno assumere soltanto un insieme finito di valori, in relazione al numero dei bits a disposizione per ciascun peso; si introdurranno così anche gli errori dovuti alla quantizzazione (per una panoramica veloce su queste problematiche si veda [17]).

Questa ulteriore limitazione, tipica dei pesi digitali, deporrebbe a favore di una integrazione analogica degli stessi, che sarebbero così soggetti soltanto a vincoli di ampiezza; questo tipo di integrazione risulta però più complessa da realizzare, per problemi di natura tecnica che fino ad oggi non hanno ricevuto una soluzione elegante.
Più precisamente, non si hanno ancora a disposizione tecniche affidabili per l'integrazione analogica di pesi che possano essere modificati, condizione questa indispensabile per la realizzazione di circuiti neurali che siano flessibili e

soprattutto capaci di apprendere.

Fino ad ora ci si è affidati a dispositivi con effetto capacitivo (transistori MOS): la quantità di carica presente nel dispositivo sarebbe proporzionale al valore del peso; questi dispositivi tuttavia sono soggetti a scarica in tempi molto brevi ($< 1ms$) e quindi inadatti a fungere da unità di memorizzazione, a meno di essere continuamente "rinfrescati" per mezzo di memorie a lungo termine. Negli ultimi tempi sono stati messi a punto dispositivi chiamati *floating gate transistors*, che mostrano una capacità di ritenzione della carica che può arrivare fino ad alcuni anni. Pur avendo ancora dei tempi di scrittura inadeguati, i floating gate transistors sembrano oggi i migliori candidati a sostituire le memorie binarie che, oltre ad avere i problemi sopra menzionati, occupano sul chip spazi enormi, se comparati a quelli degli equivalenti dispositivi analogici.

2.3 Le Strutture di Connessione

La necessità di collegare in maniera per quanto possibile completa i neuroni presenti su un chip, fa sì che la maggior parte dell'area di silicio sia occupata dalle strutture di connessione.

Questo impedisce che su un unico chip possano essere integrati più di un certo numero di neuroni (di norma qualche centinaio). Diventa decisivo allora, volendo avere a disposizione reti molto grandi, poter collegare fra loro diversi chip.

Di seguito ci limiteremo ad elencare solo alcune tra le problematiche legate al collegamento dei neuroni su chip:

- la sensibilità al rumore propria dei "fili" di connessione;

- la difficoltà di adattare i valori in uscita di un chip a quelli in ingresso al successivo, solo in parte risolubile adottando tecniche di trasmissione digitali;
- gli alti consumi energetici delle connessioni;
- le mutue interazioni tra i segnali trasferiti, che riducono la larghezza di banda "utile".

In questi ultimi anni le reti **opto-elettroniche** sono state certamente il tentativo più interessante fatto per risolvere i problemi legati alla comunicazione tra neuroni [11].
L'approccio opto-elettronico, sostituendo fasci di luce ai fili di connessione, offre diversi contributi risolutivi: abbassa in maniera drastica i consumi di energia; elimina i problemi di interazione tra i segnali trasmessi e, ciò che è più importante, consente la realizzazione di reti molto **dense**, potendo sfruttare per le connessioni anche la terza dimensione dello spazio.

I primi prototipi realizzati hanno messo in evidenza, tra l'altro, un ulteriore incremento nelle velocità di computo rispetto ai "normali" chip integrati. Rispetto ad un simulatore software supportato da hardware adeguato, le reti opto-elettroniche sembrerebbero $10^9/10^{10}$ volte più veloci. Non è male.

Parte II

Risultati

Capitolo 3

Reti a Pesi Limitati

Lavori teorici sulla capacità di approssimare funzioni a valori reali da parte delle reti feed-forward sono stati proposti da diversi autori (vedi [14, 15, 13, 33, 19]), tra i quali meritano menzione K. Hornik, M. Stinchombe e H. White. Gli ultimi due, in particolare, hanno ottenuto nel 1990 [34] un importante risultato, dimostrando che le reti feed-forward a pesi limitati aventi un unico strato di neuroni nascosti con funzione di attivazione $\in C(\mathbb{R})$, sono in grado di approssimare bene quanto si voglia funzioni continue a valori in $\mathbb{R}^n$, a patto comunque di predisporre un numero sufficiente di unità nascoste. Questo fatto, quantunque di notevole rilevanza teorica, ha contribuito forse a far diminuire l'attenzione rivolta verso la ricerca delle relazioni esistenti tra vincoli sui pesi, numero di neuroni ed effettive capacità di apprendimento delle reti neurali, quantomeno per quegli aspetti più direttamente connessi al campo delle reti hardware.

3.1 Il Problema

Il presente lavoro, come già accennato, concerne non tanto le capacità di approssimazione, quanto quelle "classificatorie" delle reti feed-forward multistrato.
Come abbiamo osservato nel capitolo precedente, le integrazioni hardware delle reti neurali introducono delle limitazioni, non tanto per ciò che riguarda le funzioni di trasferimento quanto piuttosto in relazione ai pesi, qualunque sia l'approccio progettuale di riferimento (vedi la sezione 2.2).
I pesi, infatti, saranno costretti ad assumere valori all'interno di un intervallo prefissato e, nel caso di implementazione digitale, dovranno sopportare le approssimazioni legate alla quantizzazione.

In questo capitolo cercheremo di capire se e quanto il potere separatore di una rete risulti compromesso qualora i pesi della rete siano vincolati all'interno di un intervallo prefissato che contenga lo zero; nel successivo analizzeremo invece gli effetti della quantizzazione.
Più precisamente, il problema affrontato sarà quello della classificazione di due punti dello spazio degli ingressi appartenenti a classi distinte.

Cominceremo la nostra analisi considerando il caso di reti formate da neuroni "a soglia", neuroni cioè aventi come funzione di trasferimento la funzione a gradino descritta dalla (1.3). Successivamente ci rivolgeremo alle reti aventi neuroni "sigmoidali"[1]: chiameremo così quei neuroni la cui funzione di attivazione sia "assimilabile", per il suo andamento e per i suoi effetti, alla funzione sigmoide (vedi la (1.7)). In entrambe i casi la *regola di propagazione*, che

[1] preciseremo nel seguito, in maniera formale, il significato del termine (vedi la definizione a pag. 50).

calcola l'ingresso totale al neurone, sarà definita nel modo seguente:

$$net(\overline{w}, \overline{x}) = \sum_{i=1}^{n} w_i x_i + \theta \tag{3.1}$$

dove i w_i sono le componenti del vettore $\overline{w}$ dei pesi associato agli n ingressi x_i del neurone, e θ è il valore di soglia.

Sia le reti a soglia che quelle di neuroni sigmoidali sono sempre state oggetto di particolare attenzione da parte della comunità scientifica. In letteratura troviamo, in proposito, lavori basilari, tra cui vanno certamente ricordati [21, 28, 29] ed il fondamentale [31]. Esistono inoltre molti studi relativi alle capacità di classificazione delle reti; tra quelli che presentano maggiori affinità con il lavoro da noi svolto ricordiamo [22, 35, 36].

3.2 Il Neurone a Soglia

Consideriamo innanzitutto il caso più semplice di una rete costituita da un unico neurone a soglia. Senza perdere in generalità, ci restringeremo al caso di un neurone avente due ingressi, essendo concettualmente semplice estendere quanto segue al caso $\mathbb{R}^n$.

L'ingresso totale al nostro neurone sarà espresso dunque dalla regola di propagazione:

$$net(\overline{w}, \overline{x}) = w_1 x_1 + w_2 x_2 + \theta \tag{3.2}$$

Formalmente, trovare un neurone che "separi" due punti $\overline{x}$ (x_1, x_2) ed $\overline{y}$ (y_1, y_2) del piano, vorrà dire individuare un vettore $\overline{w}$ di pesi ed una soglia

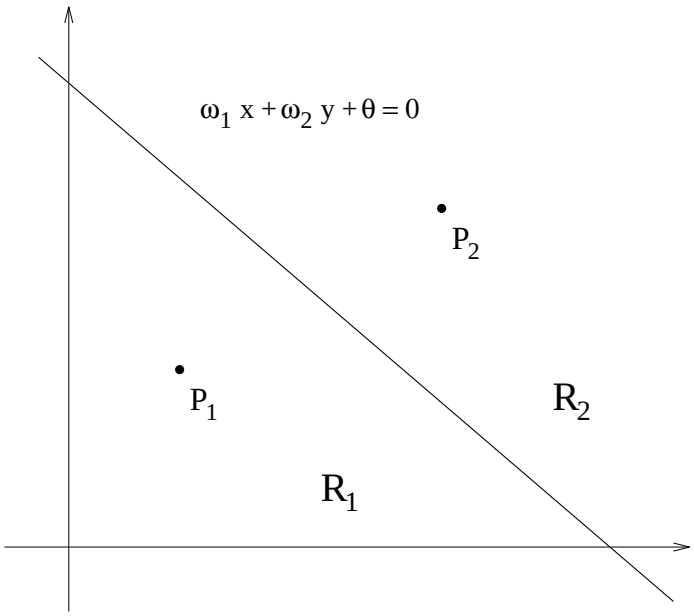

Figura 3.1: *Un neurone a soglia con due ingressi divide il piano in due regioni.*

θ tali che abbia soluzione il sistema:

$$\begin{cases} w_1 x_1 + w_2 x_2 + \theta > 0 \\ w_1 y_1 + w_2 y_2 + \theta \leq 0 \end{cases} \tag{3.3}$$

È possibile inquadrare il problema anche da un punto di vista geometrico: separare due punti P_1, P_2 del piano vorrà dire allora trovare una retta che lo divida in due regioni R_1, R_2, in modo tale che $P_1 \in R_1$ e $P_2 \in R_2$ o viceversa (vedi fig. 3.1).

La geometria ci insegna che questo è sempre possibile purché i punti siano distinti (basti pensare, se non altro, alla retta ortogonale al segmento che congiunge i punti dati e passante per il punto medio di quest'ultimo). Individuata dunque la retta in questione, sia la sua equazione:

$$ax + by + c = 0 \tag{3.4}$$

Ora, come sappiamo, se da un lato l'equazione lineare (3.4) individua la retta cercata, d'altro canto la stessa retta sarà descritta dalle infinite equazioni

lineari

$$a'x + b'y + c' = 0 \tag{3.5}$$

dove $a' = \lambda a, \qquad b' = \lambda b, \qquad c' = \lambda c,$

con $\lambda \in I\!R$, $\lambda \neq 0$.

Qualora si fosse vincolati all'uso di parametri limitati in modulo, supponiamo dal valore B, basterà scegliere

$$\lambda = \frac{B}{\max(|a|, |b|, |c|)} \tag{3.6}$$

in modo che la (3.5) abbia tutti i coefficienti, in modulo, minori o al massimo uguali a B.

In definitiva, sarà sempre possibile trovare un neurone a soglia con pesi limitati in grado di separare due punti nel piano o, se si vuole, un vettore di pesi ed una soglia, comunque limitati in modulo, che soddisfino il sistema (3.3). La limitazione sui pesi non diminuisce dunque il potere separatore di un neurone a soglia.

3.2.1 Equazioni Lineari Equivalenti

Prima di passare alle reti di neuroni a soglia, ci sembra appropriato definire in questa sede il concetto di equivalenza tra equazioni lineari. La definizione discende in maniera naturale dalle considerazioni precedenti, tuttavia la porremo tra vettori m-dimensionali, vista la corrispondenza biunivoca esistente tra le m-ple reali e le equazioni lineari $\sum_{i=1}^{m-1} a_i\, x_i + a_m = 0$.

Definizione 3.1 *Sia $I\!R^m$ l'insieme delle m-ple reali; definiamo una*

relazione $\sim$ *sugli elementi di* $\mathbb{R}^m$ *ponendo* $\forall\, \overline{x}, \overline{y} \in \mathbb{R}^m$,

$$\overline{x} \sim \overline{y} \;\; se \;\; \exists\, \lambda \in \mathbb{R},\; \lambda \neq 0 \;:\; \overline{x} = \lambda \overline{y}. \tag{3.7}$$

Si dimostra facilmente che la (3.7) definisce un'equivalenza tra vettori di $\mathbb{R}^m$.

□

Se quozientiamo $\mathbb{R}^m$ rispetto a $\sim$, otteniamo l'insieme

$$\mathbb{R}^m/\sim \; = \{\, [\overline{x}]_\sim \;:\; \overline{x} \in \mathbb{R}^m \} \tag{3.8}$$

dove

$$[\overline{x}]_\sim = \{\, \overline{y} \in \mathbb{R}^m \;:\; \overline{y} \sim \overline{x} \} \tag{3.9}$$

è la classe di equivalenza contenente $\overline{x}$.

Generalizzando le considerazioni fatte in precedenza nel piano cartesiano, potremo dire che l'insieme $\mathbb{R}^m/\sim$ si trova in corrispondenza biunivoca con l'insieme degli iperpiani distinti di dimensione $m-2$, se consideriamo anche gli iperpiani degeneri corrispondenti alle classi di equivalenza $\left[\overline{0}\right]_\sim$ e $[(0, 0, \ldots, 0, 1)]_\sim$.

Tornando nel piano cartesiano, la retta di equazione $ax + by + c = 0$ potrà, a questo punto, essere denotata utilizzando la classe di equivalenza $[(a, b, c)]_\sim$. In altre parole, a ciascuna retta corrisponderà ora un elemento di $\mathbb{R}^3/\sim$.

3.3 Reti di Neuroni a Soglia

Quanto visto per il singolo neurone accade pure per reti di neuroni a soglia con topologie comunque complesse: sarà sufficiente modificare in modo opportuno i parametri relativi a ciascun neurone per ottenere una rete *equivalente* (che

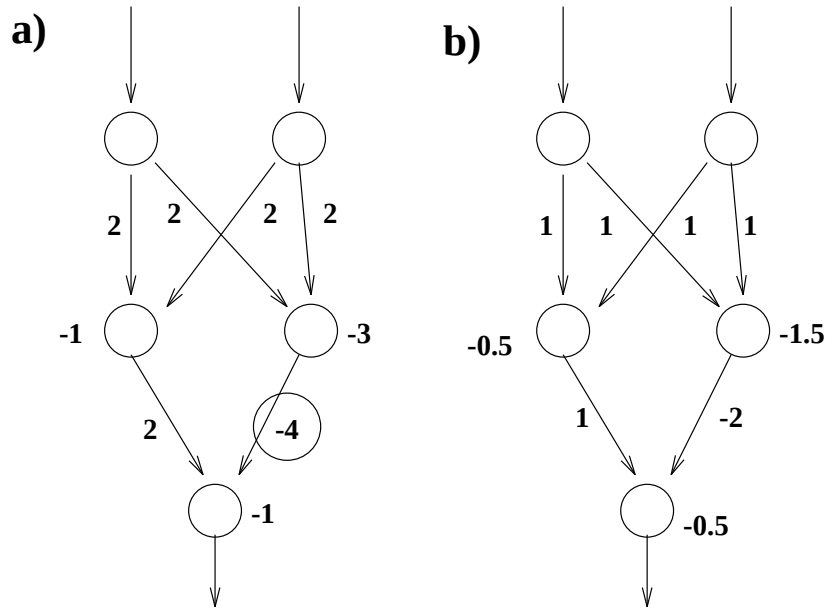

Figura 3.2: *La rete a soglia non perde le sue potenzialità limitando i pesi.*

risponda cioè in ugual maniera alle stesse sollecitazioni in ingresso) a pesi limitati.

Esempio 3.1 Consideriamo la rete di neuroni a gradino in figura 3.2a. La topologia ivi riportata, con i pesi e le soglie indicate, risolve egregiamente il problema dello XOR. Si supponga ora di volere una rete di uguale topologia e di pesi limitati a 2 in modulo, che sia ad essa equivalente: basterà moltiplicare per $h = \frac{2}{\max_i(|w_i|)}$ ciascuno dei parametri della rete (nei pesi w_i sono comprese anche le soglie), per ottenere la rete mostrata in figura 3.2b. Questa rete avrà, come si verifica facilmente, lo stesso comportamento di quella originaria.

3.4 Alcune Definizioni

Il neurone a soglia si dimostra, in definitiva, **sempre** in grado di separare due punti, indipendentemente da limitazioni sui pesi cui debba sottostare; per questo motivo non ci sembrerebbe fuori luogo indicare come "infinita" la sua capacità di classificazione.

Per poter mettere a confronto in maniera rigorosa il potere separatore di ciascun tipo di neurone, introdurremo alcune definizioni. Prima di ciò, tuttavia, vorremmo fare un'osservazione.

Mentre il neurone a soglia fornisce sempre una risposta "chiara" (0 oppure 1) qualunque sia il valore in ingresso, **la risposta di un neurone sigmoidale varia in "intensità"** al variare dell'input; se questa intensità sia "sufficiente" ad individuare la classe di appartenenza di un ingresso dipende dalle scelte di progetto. In altre parole, occorrerà prima di tutto stabilire quale debba essere, per le risposte del neurone, la distanza massima accettabile (la **indicheremo con** ε) dai valori che la funzione di attivazione può fornire "alla saturazione". Supponendo ora che il neurone restituisca un'uscita y nell'intervallo $[0, 1]$, chiameremo **"0"** (**"1"**) la classe di quegli ingressi per i quali l'output del neurone sarà minore di ε (maggiore di $1-\varepsilon$). Negli altri casi, cioè quando $\varepsilon \leq y \leq 1-\varepsilon$, la classe dell'ingresso rimarrà indeterminata.

Osservare la situazione da un punto di vista geometrico ci sarà di aiuto. La figura 3.3 mostra l'uscita di un neurone con due ingressi, pesi $w_1 = w_2 = 1$ e funzione di attivazione data dalla (1.7), con parametro $T = 1$. Come si vede dalla figura, il neurone divide lo spazio degli ingressi in tre regioni, le cui dimensioni dipenderanno dalla tolleranza scelta per le risposte, cioè dal valore di ε. Una regione in cui l'uscita del neurone sarà "1", una regione con uscita

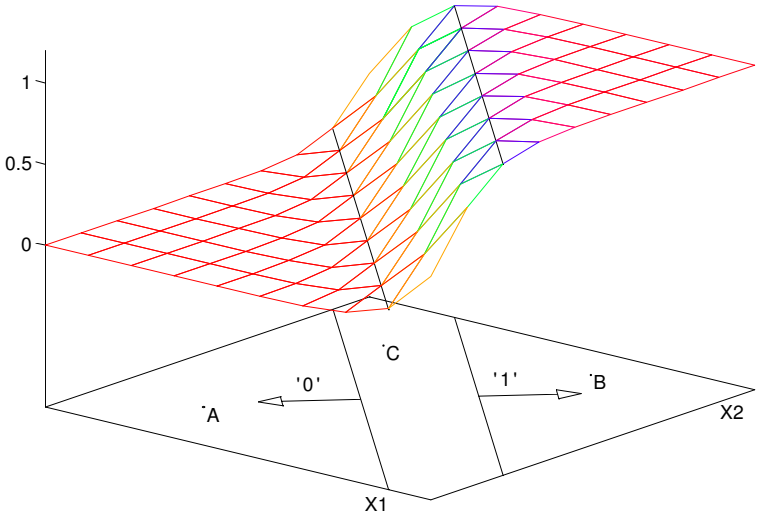

Figura 3.3: *Un neurone sigmoidale divide il piano in tre regioni.*

"0" ed una terza regione intermedia, che chiameremo *fascia di indistinguibilità*, dove la risposta non sarà sufficientemente nitida da poter essere classificata. La scelta di ε inciderà in particolare sull'ampiezza di quest'ultima regione.
Nella nostra figura i punti A e B sono "separati" dal neurone al contrario di B e C, dal momento che C si trova nella fascia di indistinguibilità.
È importante, in ultima analisi, tenere presente che la capacità di un neurone sigmoidale di separare due ingressi dipende innanzitutto da quanto nitida vogliamo che sia la sua risposta.

Detto questo, passiamo alle definizioni:

Definizione 3.2 *Sia u un neurone con funzione di uscita $f_{\overline{w}} : \mathbb{R}^n \to [\alpha, \beta]$; diremo che due punti $\overline{x_1}$ e $\overline{x_2} \in \mathbb{R}^n$ sono* **separati** *da u (oppure u* **separa** *$\overline{x_1}$ e $\overline{x_2}$) sse*

$$f_{\overline{w}}(\overline{x_1}) \leq \alpha + \varepsilon \quad e \quad f_{\overline{w}}(\overline{x_2}) \geq \beta - \varepsilon \tag{3.10}$$

o viceversa;

dove $0 \leq \varepsilon \leq \frac{\beta-\alpha}{2}$ *e* $\overline{w}$ *è un fissato vettore di parametri che influenza il comportamento della funzione* f. □

Definizione 3.3 *Sia* u *un neurone con funzione di uscita* $f_{\overline{w}} : \mathbb{R}^n \to [\alpha, \beta]$*; diremo che due punti* $\overline{x_1}$ *e* $\overline{x_2} \in \mathbb{R}^n$ *sono* **separabili** *sse*

$$\exists \ \overline{w} \ \text{ tale che } \ u \ \text{ separa } \ \overline{x_1} \, e \, \overline{x_2}. \tag{3.11}$$

□

Ai fini delle precedenti definizioni non è stato necessario specificare ulteriormente la natura della funzione f associata al neurone u, che è stata presentata in modo generico come funzione degli n ingressi ed influenzata dal vettore $\overline{w}$ dei parametri del neurone.
Di fatto restringeremo la nostra attenzione a quelle funzioni di uscita che siano composizione di due funzioni g ed h, dove h coincide con la regola di propagazione $net(\overline{w}, \overline{x})$ definita dalla (3.1), e g appartiene alla classe di funzioni di trasferimento che definiamo qui di seguito.

Definizione 3.4 *Chiameremo* $\mathcal{G}(\mathbb{R})$ *l'insieme delle funzioni* $g : \mathbb{R} \longrightarrow \mathbb{R}$ **non decrescenti** *e tali che*

$$-\infty < \alpha = \lim_{x \to -\infty} g(x) < \lim_{x \to +\infty} g(x) = \beta < +\infty \tag{3.12}$$

□

Osservazione: Se $g \in \mathcal{G}(\mathbb{R})$ ed α, β denotano i limiti di g a $-\infty$ e $+\infty$ rispettivamente, come indicato nella (3.12), allora $g : \mathbb{R} \longrightarrow [\alpha, \beta]$.

Come si vede, la classe di funzioni $\mathcal{G}(\mathbb{R})$ comprende le già citate funzione a gradino e funzione sigmoide.
Anche se il termine "funzione di uscita" è stato adoperato per denotare la funzione f delle definizioni 3.2 e 3.3, nel seguito lo utilizzeremo, seguendo la prassi comune, anche per indicare la funzione di trasferimento del neurone.

Puntualizziamo ora, sulla base della definizione 3.4, altri concetti che risulteranno utili alla nostra causa.

Definizione 3.5 *Sia u un neurone con funzione di attivazione $f \in \mathcal{G}(\mathbb{R})$; indichiamo con α e β i limiti a $-\infty$ e $+\infty$, come nella definizione 3.4; sia ε un numero reale appartenente a $\left(0, \frac{\beta-\alpha}{2}\right)$; chiameremo* **intervallo di inseparabilità** *l'intervallo $[a, b] \subset \mathbb{R}$ dove*

$$a = \sup_x\{x : f(x) \leq \alpha + \varepsilon\} \qquad b = \inf_x\{x : f(x) \geq \beta - \varepsilon\} \tag{3.13}$$

□

Definizione 3.6 *Sia $[a, b]$ l'intervallo di inseparabilità di un neurone u, e sia $k = b - a$; definiremo* **grado di separabilità** *di u il numero $\frac{1}{k}$.* □

Riteniamo sia importante sottolineare una volta ancora come il problema della separabilità sia legato alla scelta del valore ε che definisce l'errore accettabile. A rigore dunque, nelle definizioni avremmo dovuto parlare di "ε – separabilità" anche se, per nostra comodità, abbiamo preferito non farlo. Vogliamo precisare infine il significato del termine *sigmoidale* introdotto in precedenza:

Definizione 3.7 *Chiameremo* **sigmoidale** *un neurone la cui funzione di trasferimento f goda delle seguenti proprietà:*

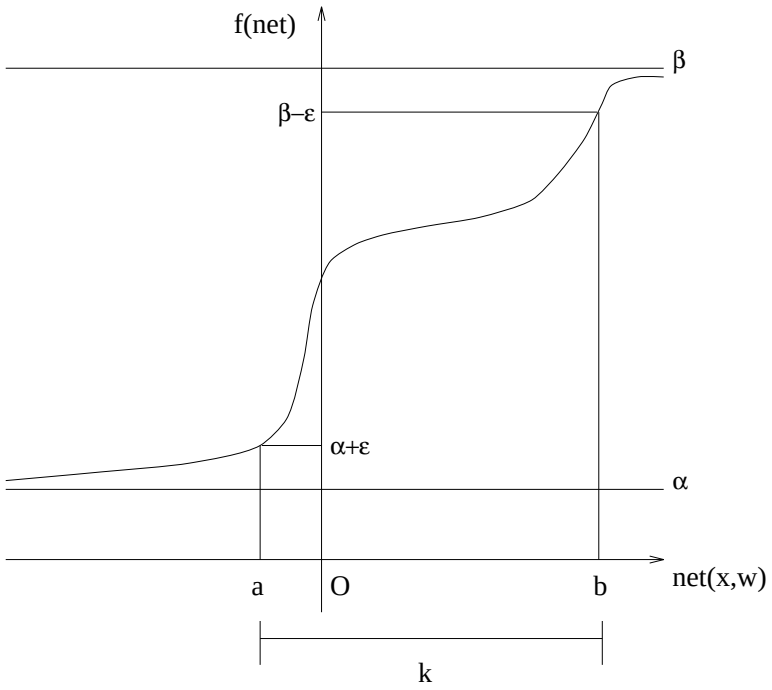

Figura 3.4: *L'intervallo di inseparabilità.*

1. $f \in \mathcal{G}(\mathbb{R})$;

2. $\exists\, \varepsilon, 0 < \varepsilon < \frac{\beta-\alpha}{2} :\ b_\varepsilon > a_\varepsilon$.

dove $a_\varepsilon, b_\varepsilon$ *sono stati definiti nella (3.13).* □

Come si arguisce dalla definizione, con il termine *sigmoidale* abbiamo voluto individuare quei neuroni il cui grado di separabilità sia effettivamente influenzato dalla scelta di ε, cosa che **non avviene per il neurone a soglia**. Infatti, come abbiamo già sottolineato, la "qualità" della risposta di un neurone avente funzione di uscita a gradino è indipendente dalla tolleranza scelta; ciò fa sì, in base alla (3.13), che divenendo a e b coincidenti, si abbia $k = 0$. Questo implica, per la definizione 3.6, che il grado di separabilità di questo neurone sia infinito. Sarà quindi sufficiente che due punti $\overline{x_1}$ e $\overline{x_2} \in \mathbb{R}^n$ siano distinti perché esista un neurone a soglia in grado di separarli secondo la

definizione 3.2.

3.5 Neuroni Sigmoidali

È possibile conseguire anche con neuroni di tipo sigmoidale (che pure dipendono, per ciò che riguarda la loro capacità di separazione, dalla scelta di ε) un grado di separabilità comunque elevato, una volta fissata la tolleranza?
Prendiamo in esame, ad es., un neurone, che chiameremo u, la cui funzione di uscita sia la funzione *sigmoide* definita dalla (1.7), e che qui riportiamo di nuovo per comodità:

$$f(x) = \frac{1}{1 + e^{-x/T}} \tag{3.14}$$

Essendo la (3.14) continua e crescente su tutto il dominio, possiamo esprimere i punti estremi dell'intervallo di inseparabilità come:

$$a = f^{-1}(\varepsilon), \qquad b = f^{-1}(1 - \varepsilon) \tag{3.15}$$

dove $f^{-1}(x) = T \cdot \ln\left(\frac{x}{1-x}\right)$ è l'inversa della funzione di uscita.
Rendiamo espliciti a e b:

$$a = T \cdot \ln\left(\frac{\varepsilon}{1-\varepsilon}\right), \qquad b = T \cdot \ln\left(\frac{1-\varepsilon}{\varepsilon}\right) \tag{3.16}$$

È facile notare che, quando T tende a zero, la stessa cosa succede ai valori di a e b; di conseguenza, andando ad infinito il suo grado di separabilità, il neurone u mostrerà lo stesso comportamento del neurone a soglia. Dalla figura 1.8 si vede come, quando $T \longrightarrow 0$, la forma della sigmoide tenda a sovrapporsi a quella della funzione a gradino.

Consideriamo ora un neurone v avente funzione di attivazione g sigmoide, in cui manchi il parametro T o, se si vuole, con $T = 1$. Il valore di attivazione

sia espresso dalla (3.1). In questo caso rimarranno fissati, come si vede dalla (3.16), anche a e b, e quindi k. Non potendo modificare k, non potremo ottenere in maniera diretta un grado di separabilità comunque grande per il neurone v. Possiamo, tuttavia, fare ancora in modo che v "simuli" il comportamento del neurone u agendo sui pesi delle connessioni che a v afferiscono: infatti, scegliendo per v un vettore di pesi $\overline{w}^*$ tale che

$$w_i^* = \frac{w_i}{T} \qquad\qquad i = 1, \ldots, n \tag{3.17}$$

dove i w_i sono i pesi[2] di u e T la temperatura della sua funzione di uscita, otteniamo

$$\begin{aligned} f(net(\overline{w}, \overline{x})) &= f\left(\sum_i w_i \cdot x_i\right) = \\ &= \frac{1}{1 + e^{-\frac{\sum_i w_i x_i}{T}}} = \frac{1}{1 + e^{-\sum_i \frac{w_i}{T} \cdot x_i}} = \\ &= g\left(\sum_i w_i^* \cdot x_i\right) = g(net(\overline{w}^*, \overline{x})) \end{aligned} \tag{3.18}$$

Se consentiamo ai pesi w_i^* di crescere senza limitazioni, anche il neurone v potrà esibire una capacità di separazione uguale a quella di u: dalla (3.17) si vede infatti che quando $T \longrightarrow 0$, $w_i^* \longrightarrow \infty$.

Quanto visto sembrerebbe condurci ad un'importante conclusione: è possibile avere neuroni sigmoidali con un grado di separabilità comunque elevato agendo su un eventuale parametro di temperatura presente nella funzione di uscita o, in alternativa, permettendo ai pesi delle connessioni di crescere senza limitazioni.

Il fatto è che praticamente tutte le implementazioni hardware delle reti neurali non consentono ai pesi delle connessioni di crescere senza limiti, né

[2]Abbiamo assimilato, in questo caso, la soglia agli altri pesi; vi vede facilmente, infatti, che la soglia si comporta come un peso associato ad un ingresso sempre uguale ad 1.

possono integrare funzioni che siano modificabili attraverso parametri di temperatura.
Ma in queste condizioni, dati due punti nello spazio degli ingressi, potrebbe non esistere un neurone capace di distinguerli.
Il teorema che segue puntualizza questa conclusione in modo formale.

Nell'esprimere la distanza tra due punti utilizzeremo non la tradizionale formula euclidea, bensi la notazione derivante dalla *norma di Manhattan,* detta anche "norma 1" (vedi l'equazione (A.5) in appendice):

$$d_1(\overline{x}, \overline{y}) = \sum_{i=1}^{n} |x_i - y_i| \tag{3.19}$$

Inoltre indicheremo con B il massimo valore in modulo che un peso w_k può assumere, cioè

$$\forall\, k \;\; |w_k| \;\leq\; B \tag{3.20}$$

Teorema 3.1 *Sia u un neurone con funzione di uscita $f \in \mathcal{G}(\mathbb{R})$; sia il valore di attivazione dato dalla $net\,(\overline{w}, \overline{x}) = \sum_i w_i x_i + \theta$ e sia $\frac{1}{k}$ il grado di separabilità del neurone; se u separa due punti $\overline{x}, \overline{y} \in \mathbb{R}^n$ allora*

$$d_1(\overline{x}, \overline{y}) \geq \frac{k}{B} \tag{3.21}$$

Dim. Poiché u separa $\overline{x}$ ed $\overline{y}$, secondo la definizione 3.2 avremo

$$f(net(\overline{w}, \overline{x})) \leq \alpha + \varepsilon \qquad e \qquad f(net(\overline{w}, \overline{y})) \geq \beta - \varepsilon \tag{3.22}$$

o viceversa;

Chiamiamo a, b gli estremi dell'intervallo di inseparabilità; essendo f non decrescente, possiamo riscrivere la condizione di separabilità:

$$net(\overline{w}, \overline{x}) \leq a \qquad e \qquad net(\overline{w}, \overline{y}) \geq b \tag{3.23}$$

o viceversa.

La precedente equazione implica

$$b - a \leq |net(\overline{w}, \overline{x}) - net(\overline{w}, \overline{y})| \tag{3.24}$$

da cui

$$\begin{aligned} k &\leq \left| \sum_{i=1}^{n} w_i x_i + \theta - \sum_{i=1}^{n} w_i y_i - \theta \right| = \\ &= \left| \sum_{i=1}^{n} w_i (x_i - y_i) \right| \leq \\ &\leq \sum_{i=1}^{n} |w_i| \cdot |x_i - y_i| \end{aligned} \tag{3.25}$$

Per ogni termine della sommatoria, $|w_i|$ può essere maggiorato da B, cosicché avremo:

$$\begin{aligned} k &\leq \sum_{i=1}^{n} |w_i| \cdot |x_i - y_i| \leq \\ &\leq \sum_{i=1}^{n} B \cdot |x_i - y_i| \leq \\ &\leq B \cdot \sum_{i=1}^{n} |x_i - y_i| \end{aligned} \tag{3.26}$$

dalla quale

$$\frac{k}{B} \leq \sum_{i=1}^{n} |x_i - y_i| \tag{3.27}$$

e quindi

$$d_1(\overline{x}, \overline{y}) \geq \frac{k}{B} \tag{3.28}$$

□

Il teorema fornisce una condizione necessaria affinché due punti siano separabili da un neurone. Con un esempio cercheremo di metterne in evidenza le implicazioni.

Esempio 3.2 Si supponga di avere due punti nello spazio a due dimensioni, $\overline{x_1} = (0, 0)$ ed $\overline{x_2} = (1, 1)$; sia u un neurone con funzione di uscita f data dalla (3.14) con il parametro $T = 1$; si scelga $B = 1$; supponiamo inoltre di volere una certa accuratezza di risposta da parte del neurone: sceglieremo, ad es., $\varepsilon = 0.1$ (ciò vuol dire che considereremo 0 l'uscita del neurone se questa sarà minore od uguale a 0.1, ed 1 se maggiore od uguale a 0.9). In questo caso l'intervallo di inseparabilità diventa

$$[f^{-1}(\varepsilon), f^{-1}(1 - \varepsilon)] = [-2.21, 2.21]$$

Essendo $k = 4.42$ e $d_1(\overline{x_1}, \overline{x_2}) = 2$, la (3.21) non viene soddisfatta. Questo ci dice che il potere separatore di u è insufficiente a distinguere i punti $\overline{x_1}$ e $\overline{x_2}$.

A questo punto abbiamo due possibilità: potremmo scegliere un valore per B più grande di 1, oppure un valore per ε più grande di 0.1. Se il nostro neurone fosse implementato in un chip, non avremmo probabilmente la possibilità di cambiare il valore di B; dovremo dunque essere più tolleranti nei confronti del neurone (poverino), ed accettare una minore accuratezza nelle sue risposte scegliendo un ε appropriato.

Possiamo chiederci quale debba essere la minima tolleranza accettabile perché la (3.21) sia soddisfatta con $B = 1$. In questo caso k non potrà essere più grande di 2; se scegliamo $[-1, 1]$ tra i possibili intervalli di inseparabilità (il che vuol dire fissare nella (3.2) un vettore $\overline{w} = (1, 1)$ e $\theta = -1$), possiamo calcolare $f(-1)$ ed $f(1)$. Avremo $f(-1) = 0.27$ ed $f(1) = 0.73$; basterà accettare una tolleranza $\varepsilon = 0.27$ perché le forze del nostro neurone siano sufficienti a separare, secondo la definizione 3.2, i punti dati.

Sfortunatamente l'equazione (3.21) esprime una condizione necessaria ma non sufficiente per la separabilità di due punti: è possibile cioè che non esista un neurone in grado di separare due punti anche qualora la (3.21) risulti soddisfatta. Va considerata infatti, ai fini della sufficienza, anche la collocazione dei punti nello spazio degli ingressi.
Dopo aver impostato il problema della sufficienza in linea generale, faremo un esempio in $\mathbb{R}^2$ che serva a chiarire le idee.

Sia u un neurone sigmoidale con funzione di trasferimento f.
Fissiamo il grado di tolleranza sulle risposte di u: sia esso $\hat{\varepsilon}$; resterà, di conseguenza, definito l'intervallo di inseparabilità $[a_{\hat{\varepsilon}}, b_{\hat{\varepsilon}}]$ (vedi la definizione 3.5) e, con esso, il numero $k_{\hat{\varepsilon}} = b_{\hat{\varepsilon}} - a_{\hat{\varepsilon}}$.
Sia B il valore che maggiora il modulo di ciascun parametro del neurone.
Prendiamo due punti $\overline{x}, \overline{y} \in \mathbb{R}^n$ che soddisfino la (3.21): si tratterà adesso di vedere se esistono un vettore $\overline{w}$ di pesi ed una soglia θ, con i $|w_i| \leq B$ e $|\theta| \leq B$, per cui il sistema

$$\begin{cases} \overline{w}\ \overline{x} + \theta \geq b_{\hat{\varepsilon}} \\ \overline{w}\ \overline{y} + \theta \leq a_{\hat{\varepsilon}} \end{cases} \tag{3.29}$$

sia soddisfatto.

Esempio 3.3 Chiamiamo in causa ancora una volta il neurone u dell'esempio 3.2; saremo tuttavia più tolleranti con lui scegliendo $\hat{\varepsilon} = 0.23$; avremo perciò $a_{\hat{\varepsilon}} = f^{-1}(\hat{\varepsilon}) = -1$ e $b_{\hat{\varepsilon}} = f^{-1}(1 - \hat{\varepsilon}) = 1$; siano $\overline{x} = (1, 10)$, $\overline{y} = (2, 8)$ i punti che soddisfano la (3.21) con $k_{\hat{\varepsilon}} = b_{\hat{\varepsilon}} - a_{\hat{\varepsilon}}$ e $B = 1$.
Si tratterà di vedere se il sistema

$$\begin{cases} \overline{w} \cdot (1, 10) + \theta \ \geq \ \ 1 \\ \overline{w} \cdot \ (2, 8) \ + \theta \ \leq -1 \end{cases} \tag{3.30}$$

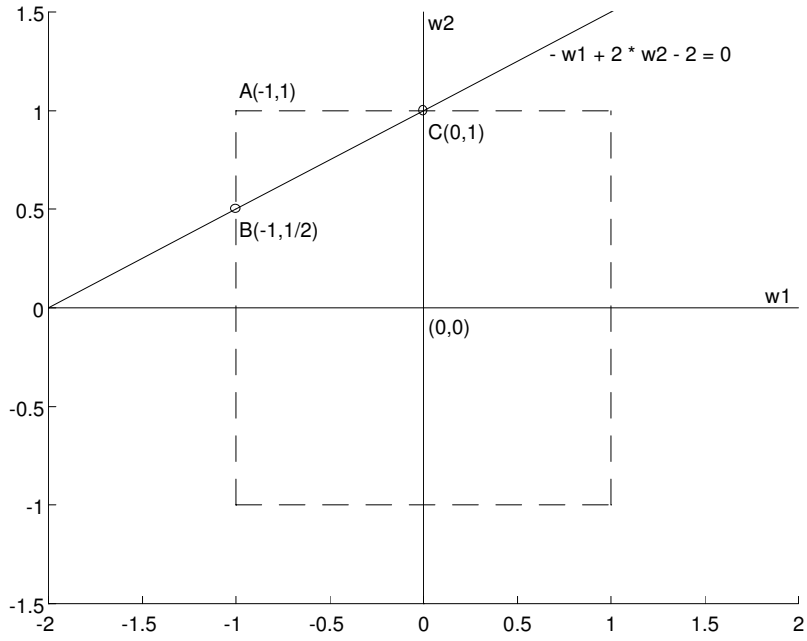

Figura 3.5: *La regione ammissibile è definita dal triangolo ABC (es. 3.3).*

con i $|w_i| \leq 1$ e $|\theta| \leq 1$, ammette soluzioni.

Sia $\overline{z} := \overline{x} - \overline{y} = (-1, 2)$; sottraendo nel sistema (3.30) la seconda equazione alla prima otteniamo

$$\overline{w} \cdot \overline{z} \geq k \tag{3.31}$$

e cioè

$$-w_1 + 2\, w_2 \geq 2 \tag{3.32}$$

Disegnamo sul piano w_1, w_2 la retta

$$-w_1 + 2\, w_2 = 2 \tag{3.33}$$

oltre alle rette che rappresentano i vincoli sul vettore dei pesi.

Dalla figura 3.5 si vede che le coppie (w_1, w_2) ammissibili si trovano all'interno del triangolo ABC; vorremmo ora individuare il corrispondente intervallo

dei valori ammissibili per la soglia θ.

Dal sistema (3.30) si vede che per θ deve valere:

$$1 - \overline{w} \cdot (1, 10) \leq \theta \leq -1 - \overline{w} \cdot (2, 8) \tag{3.34}$$

Si tratterà allora di determinare quali siano i due punti $\overline{r}, \overline{s}$ ammissibili del piano (w_1, w_2) per cui si abbia rispettivamente

$$\theta_{min} = 1 - \overline{r} \cdot (1, 10) \quad e \quad \theta_{max} = -1 - \overline{s} \cdot (2, 8). \tag{3.35}$$

Come è noto dalla geometria, il vettore $-(1, 10)$ individua il verso della normale alla retta $\theta = 1 - \overline{w} \cdot (1, 10)$; per rendere piccolo il valore di θ dovremo far scorrere la retta in senso opposto al verso della normale; così facendo e rimanendo tuttavia all'interno del triangolo ABC che costituisce la regione ammissibile, incontreremo alla fine il vertice C, in corrispondenza del quale avremo il nostro θ_{min}.

Analogamente, spostando la retta $\theta = -1 - \overline{w} \cdot (2, 8)$ lungo il verso della normale, prima di uscire dalla regione ammissibile incontreremo il punto B, in corrispondenza del quale si avrà il valore θ_{max}.

Sostituendo i punti $C\,(0, 1)$, $B\,(-1, \frac{1}{2})$ nelle espressioni (3.35) otteniamo $\theta_{min} = -9$, $\theta_{max} = -3$ rispettivamente. L'intervallo $[-9, -3]$ non rispetta i vincoli imposti sulla soglia θ e dunque il sistema (3.30) non potrà essere soddisfatto. In altre parole, non potrà esistere un neurone con funzione di trasferimento f e grado di separabilità $k = 2$ che sia capace di separare i punti $(1, 10)$ e $(2, 8)$, qualora i parametri in modulo siano maggiorati da 1.

Proseguiamo con altre considerazioni nel piano.

In sostanza, separare due punti del piano con un neurone sigmoidale vorrà

dire, geometricamente, cercare di collocare tra di essi una "fascia" di larghezza k (in norma 1),[3] delimitata dalle rette **parallele** $w_1\ x + w_2\ y + \theta = b$ e $w_1\ x + w_2\ y + \theta = a$.

Se i punti dati soddisfano la condizione necessaria per la separabilità definita dalla (3.21), saremo in grado di poter collocare in almeno un modo la fascia di indistinguibilità in maniera che i punti ne rimangano all'esterno. In altre parole, avremo la certezza di poter trovare un vettore (w_1, w_2), le cui componenti rispettino i vincoli, tale che la fascia di coefficiente angolare $-\frac{w_1}{w_2}$ passi tra i punti dati.

Tuttavia, proprio a causa dei vincoli esistenti sui parametri, non tutti i possibili modi (al limite nessuno) di collocare la fascia tra i due punti potranno essere accettati. Nella figura 3.6 si vede come sia possibile collocare in più modi tra i punti A e B la fascia generata dal neurone sigmoidale con funzione di attivazione

$$f(x) = \frac{1}{1 + e^{-x}}$$

grado di separabilità $k = 2$ e limite sui pesi $B = 1$. In realtà non uno dei "termini noti" corrispondenti alle diverse collocazioni sarà accettabile, in quanto ciascuno di questi richiede per θ dei valori che non soddisfano i vincoli posti sul parametro.

Lo stesso neurone è, invece, in grado di separare i punti C e D di figura 3.7; tutti i modi in cui è possibile collocare la fascia di indistinguibilità tra i due punti saranno accettabili, oscillando i rispettivi valori di θ fra $-\frac{1}{6}$ e $-\frac{1}{4}$.

[3] Vogliamo qui ricordare, ancora una volta, che la larghezza della fascia di indistinguibilità rimarrà definita una volta fissate la funzione di attivazione f e la tolleranza ε.

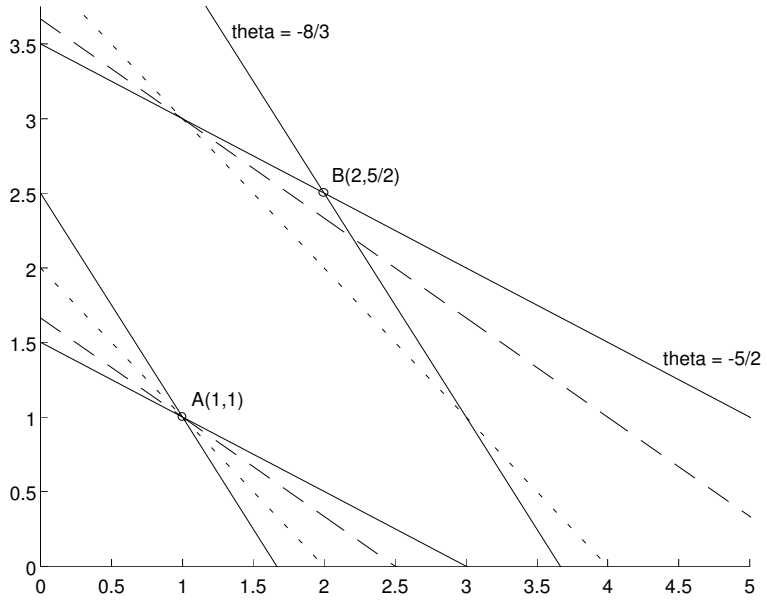

Figura 3.6: *Nessuno dei neuroni che separano A e B soddisfa i vincoli imposti ai parametri.*

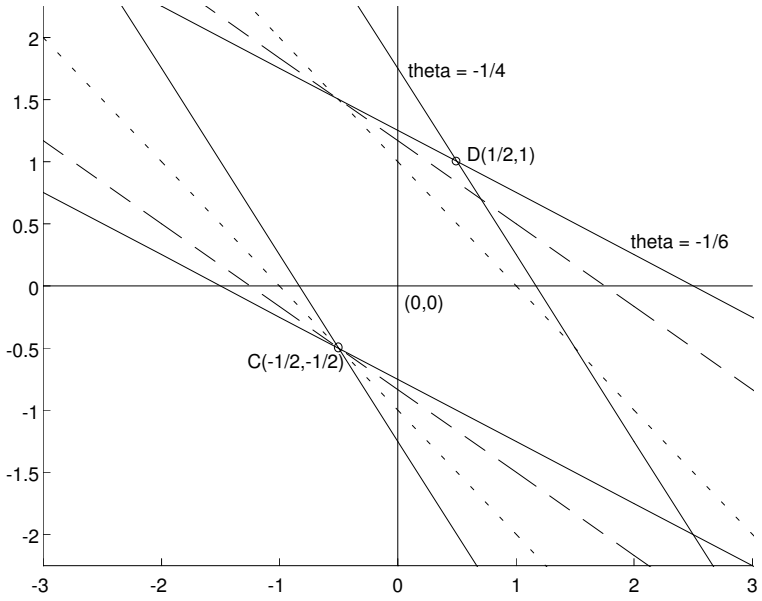

Figura 3.7: *Tutti i neuroni che separano C e D sono ammissibili.*

Prima di passare all'analisi delle capacità delle reti di neuroni sigmoidali facciamo ancora **un paio di osservazioni**:

Osservazione 1 Da quanto detto finora si comprende come, se si dà modo al parametro θ di variare liberamente, la condizione espressa dal teorema 3.1 diventi sufficiente per l'individuazione di un neurone sigmoidale che separi due punti dati.

Potremmo chiederci allora quale sia la minima soglia θ, in modulo, che consenta ad un certo neurone sigmoidale di separare due punti che soddisfino la (3.21).
In sostanza, si tratterà di minimizzare la funzione $z = |\theta|$ con i vincoli

$$\begin{aligned} &\overline{w}\,\overline{x} + \theta \geq b \\ &\overline{w}\,\overline{y} + \theta \leq a \\ &|w_i| \leq B \qquad i = 1, \ldots, n \end{aligned} \tag{3.36}$$

dove $\overline{x}, \overline{y} \in I\!R^n$.

È possibile far vedere che, adottando tecniche proprie della programmazione lineare (P.L.), il problema in oggetto può essere trasformato in un problema di P.L. in forma *standard* ([26], pag. 25).
Il sistema lineare trasformato avrà $2n + 2$ equazioni e $4n + 4$ incognite, dove n è la dimensione dello spazio degli ingressi.
Il problema, quindi, potrà essere risolto utilizzando **l'algoritmo del simplesso**, con i ben noti vantaggi che questo comporta.

Osservazione 2 Non ci si meravigli se, nelle figure 3.6 e 3.7, la larghezza della fascia sembra variare a seconda della sua inclinazione. Quella che cambia è, effettivamente, la distanza **euclidea** tra le rette, mentre la "distanza 1", cioè quella derivante dalla norma 1, rimane costante.

È possibile, a questo proposito, stabilire quale sia la configurazione di pesi che rende minima la distanza euclidea tra le rette parallele che delimitano la fascia: se chiamiamo r ed s le due rette, come da figura 3.8, questa distanza sarà quella di uno qualunque dei punti di r dalla retta s.

Sia allora P un punto della retta r di equazione $w_1\,x + w_2\,y + \theta - b = 0$; P avrà coordinate $\left(\hat{x}, -\frac{w_1}{w_2}\hat{x} + \frac{b-\theta}{w_2}\right)$. La distanza di P dalla retta s di equazione $w_1\,x + w_2\,y + \theta - a = 0$ sarà data, secondo la (A.22), da:

$$dist(P,s) = \frac{\left|w_1\,\hat{x} + w_2\,\left(-\frac{w_1}{w_2}\hat{x} + \frac{b-\theta}{w_2}\right) + \theta - a\right|}{\sqrt{w_1^2 + w_2^2}}. \tag{3.37}$$

Sviluppando si ha:

$$dist(P,s) = \frac{|w_1\,\hat{x} - w_1\,\hat{x} + b - \theta + \theta - a|}{\sqrt{w_1^2 + w_2^2}} \tag{3.38}$$

da cui

$$dist(P,s) = \frac{|b-a|}{\sqrt{w_1^2 + w_2^2}}. \tag{3.39}$$

Come si vede dalla (3.39), la distanza minima tra le due rette si ha quando i pesi assumono il massimo valore in modulo ad essi consentito, cioè quando, per ogni i, $|w_i| = B$.

Ciò corrisponde a collocare la fascia con un'inclinazione di $\pm 45°$ rispetto all'asse x. Dalle figure 3.6 e 3.7 risulta evidente come, tra le fasce aventi $d_1 = 2$

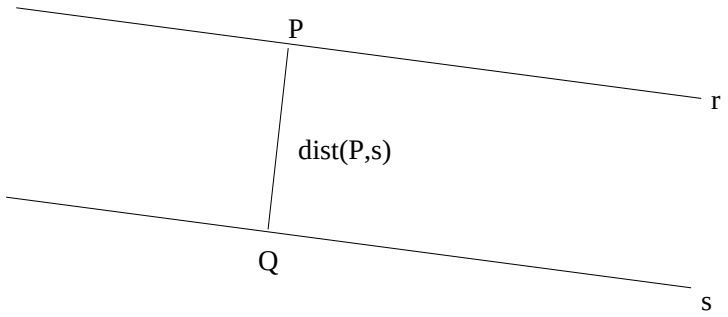

Figura 3.8: *La distanza euclidea tra due rette.*

collocate fra i punti, la "più stretta" sia quella che forma un angolo di $-45°$ con l'asse delle ascisse.

Resta inteso che tutte le considerazioni fatte in $\mathbb{R}^2$ possono essere facilmente estese al caso $\mathbb{R}^n$.

3.6 Reti Sigmoidali

Ci proponiamo ora di esaminare cosa accade alle capacità di separazione di reti con neuroni sigmoidali una volta che i pesi siano costretti ad assumere valori all'interno di un intervallo prefissato.
Affrontiamo la questione aiutandoci con un esempio e soffermiamoci ancora una volta sul problema dello XOR.

Come è noto, questo problema può essere risolto da una rete non limitata come quella riportata in figura 3.9, i cui neuroni abbiano funzione di uscita data dalla (3.40) (vedi [30]).

$$g(x) = \frac{1}{1 + e^{-x}} \tag{3.40}$$

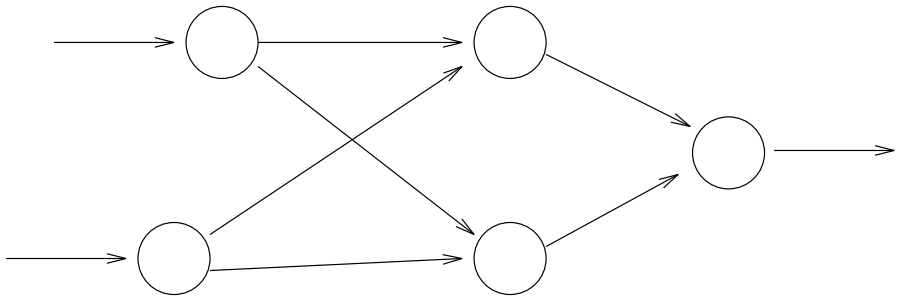

Figura 3.9: *La topologia proposta in figura risolve il problema dello XOR se i pesi non sono limitati.*

Supponiamo ora di limitare i pesi in modo che sia $B = 1$. Se scegliamo $\varepsilon = 0.1$ per il neurone di uscita, avremo che k è circa 4.42. Con questo valore di k, per il teorema 3.1 siamo sicuri che il neurone di uscita della nostra rete non sarà in grado di distinguere alcuna coppia di ingressi. Infatti, poiché i neuroni nascosti hanno uscita nell'intervallo $[0,\ 1]$, lo strato nascosto non potrà mai fornire al neurone di output una distanza maggiore di 2; perciò non sarà in alcun caso possibile separare due punti.
Da qui si vede che, limitando i pesi, non sarà sempre possibile ottenere gli stessi risultati di una rete non limitata, a meno di cambiarne la topologia. Questa è una limitazione intrinseca delle reti con pesi limitati, **che non è legata ad alcun algoritmo di apprendimento**.

Il teorema 3.2 formalizza queste osservazioni. Nel teorema restringeremo la nostra attenzione alle reti aventi un solo strato di neuroni nascosti (dette anche reti *a due livelli* dal numero dei livelli di connessione), dal momento che reti feed-forward di questo tipo a pesi limitati sono in grado di approssimare qualunque funzione continua a valori in $\mathbb{R}^n$ ([34]).
Estendiamo prima di tutto la definizione 3.2 alle reti:

Definizione 3.8 *Sia $\mathcal{N}$ una rete neurale feed-forward di N neuroni, dei quali N_I siano d'ingresso ed N_O di uscita.*
Sia $f_{\overline{w}_j} : \mathbb{R}^{n_j} \longrightarrow [\alpha, \beta],\ j = 1, \ldots, N$, la funzione di uscita di ciascun neurone. Siano $\overline{x},\ \overline{y} \in \mathbb{R}^{N_I}$ due ingressi alla rete; siano inoltre $\overline{\xi},\ \overline{\eta} \in \mathbb{R}^{N_O}$ le uscite di $\mathcal{N}$ corrispondenti ad $\overline{x}$ ed $\overline{y}$. Diremo che $\overline{x}$ ed $\overline{y}$ sono **separati** *da $\mathcal{N}$ (oppure che $\mathcal{N}$* **separa** *$\overline{x}$ ed $\overline{y}$) sse*

$$\exists\, i\ :\ \xi_i \leq \alpha + \varepsilon \quad e \quad \eta_i \geq \beta - \varepsilon \tag{3.41}$$

oppure

$$\eta_i \leq \alpha + \varepsilon \quad e \quad \xi_i \geq \beta - \varepsilon \tag{3.42}$$

dove $0 \leq \varepsilon \leq \frac{\beta-\alpha}{2}$ □

Teorema 3.2 *Sia* $\mathcal{N}$ *una rete neurale feed-forward a due livelli, con neuroni aventi funzione di trasferimento* $f \in \mathcal{G}(\mathbb{R})$ *e valore di attivazione dato dalla (3.1); sia* $\frac{1}{k}$ *il grado di separabilità (di ciascun neurone) della rete; se* $\mathcal{N}$ *separa due punti* $\overline{x}, \overline{y} \in \mathbb{R}^n$ *allora*

$$(\beta - \alpha) \cdot n \geq \frac{k}{B} \tag{3.43}$$

dove n *è il numero delle unità nascoste.*

Dim. Data la definizione 3.8, possiamo restringere la nostra attenzione ad uno dei neuroni di uscita che separano $\overline{x}$ ed $\overline{y}$; lo chiameremo u. Sia $\overline{z} = z_1, z_2, \cdots, z_n$ il vettore che rappresenta l'uscita delle unità nascoste. Siccome u, il neurone di uscita, separa $\overline{x}$ ed $\overline{y}$, dal teorema 3.1 avremo che

$$d_1(\overline{z}_x, \overline{z}_y) \geq \frac{k}{B} \tag{3.44}$$

dove $\overline{z}_x$ e $\overline{z}_y$ indicano rispettivamente le uscite delle unità nascoste quando gli ingressi $\overline{x}$ ed $\overline{y}$ sono presentati alla rete. Rendiamo esplicito il primo membro della (3.44); possiamo scrivere:

$$\begin{aligned} d_1(\overline{z}_x, \overline{z}_y) &= \sum_{i=1}^{n} |z_{x_i} - z_{y_i}| = \\ &= \sum_{i=1}^{n} |f(net_i(\overline{x})) - f(net_i(\overline{y}))| \geq \frac{k}{B} \end{aligned} \tag{3.45}$$

dove

$$net_i(\overline{x}) := net(\overline{w}_i, \overline{x}) = \sum_j w_{ij} x_j + \theta_i \tag{3.46}$$

è l'attivazione dell'i-esimo neurone nascosto.

Essendo la (3.45) una somma di n moduli, non tutti potranno essere minori di $\frac{k}{Bn}$. Dunque

$$\exists\ q : |f(net_q(\overline{x})) - f(net_q(\overline{y}))| \geq \frac{k}{Bn} \tag{3.47}$$

Ora, per come f è definita,

$$|f(net_q(\overline{x})) - f(net_q(\overline{y}))| \leq (\beta - \alpha) \tag{3.48}$$

da cui la tesi. □

Una volta fissati α, β e B (come di solito accade nelle implementazioni hardware), il teorema 3.2 fornisce un *lower bound* per il numero n dei neuroni nascosti necessari a risolvere un qualunque problema di classificazione con un certo grado di accuratezza ε. Il teorema, ad es., ci mostra immediatamente come la rete mostrata in figura 3.9 non possa convergere quando $B \leq 1$ ed $\varepsilon = 0.1$.

Il teorema nella sua enunciazione è molto generale, non facendo alcun riferimento specifico ad una particolare funzione di trasferimento.

Tuttavia, il poter centrare l'attenzione su una qualche funzione di attivazione ci consente di mettere in relazione (come è già stato fatto nel teorema 3.1) la distanza tra gli ingressi con i parametri caratteristici della rete. Vale infatti la seguente

Proposizione 3.1 *Sia $\mathcal{N}$ una rete neurale feed-forward a due livelli, con neuroni aventi funzione di uscita $f \in \mathcal{G}(\mathbb{R})$ e valore di attivazione dato dalla (3.1); sia $\frac{1}{k}$ il grado di separabilità (di ciascun neurone) della rete; se $\mathcal{N}$ separa due punti $\overline{x}, \overline{y} \in \mathbb{R}^n$ allora*

$$d_1(\overline{x}, \overline{y}) \geq \frac{t_f}{B} \tag{3.49}$$

dove t_f è un valore che dipende dall'andamento della funzione f.

Dim. Essendo la f nelle ipotesi del teorema 3.2, per essa varrà l'equazione (3.47). Affinché la (3.47) sia soddisfatta, ci chiediamo come dovrà essere la differenza

$$|net_q(\overline{x}) - net_q(\overline{y})|\,, \tag{3.50}$$

domanda lecita, considerando la non decrescenza della f. La differenza (3.50) sarà, in generale, almeno uguale ad un valore t che dipenderà dalla f. Avremo cioè:

$$|net_q(\overline{x}) - net_q(\overline{y})| \geq t_f \tag{3.51}$$

Ripercorrendo passaggi analoghi a quelli svolti nel teorema 3.1 si ha:

$$\begin{aligned} t_f &\leq |net_q(\overline{x}) - net_q(\overline{y})| = \\ &= \left|\sum_j w_{qj}(x_j - y_j)\right| \leq \sum_j |w_{qj}| \cdot |x_j - y_j| \leq \\ &\leq B \cdot \sum_j |x_j - y_j| = B \cdot d_1(\overline{x}, \overline{y}) \end{aligned} \tag{3.52}$$

da cui la tesi. □

Come calcolare t_f?

In linea di principio, dopo aver ricavato dalla (3.47) $net_q(\overline{y})$ in funzione di $net_q(\overline{x})$ (o viceversa) ed aver ottenuto quindi una espressione del tipo

$$net_q(\overline{y}) \geq h(net_q(\overline{x})) \tag{3.53}$$

dove abbiamo supposto $net_q(\overline{y}) \geq (net_q(\overline{x}))$, si tratterà di minimizzare la funzione

$$y = h(net_q(\overline{x})) - net_q(\overline{x}), \tag{3.54}$$

cosa quantomeno laboriosa[4].
Tuttavia, almeno per le funzioni di trasferimento più comuni è possibile calcolare t_f utilizzando le loro proprietà di antisimmetria.

Di seguito prenderemo in esame il caso di una rete di neuroni aventi funzione di attivazione g definita dalla (3.40) e, utilizzando i risultati precedenti, proveremo il seguente

Corollario 3.1 *Sia $\mathcal{N}$ una rete neurale feed-forward a due livelli, con neuroni aventi funzione di uscita $g = \frac{1}{1+e^{-x}}$ e valore di attivazione dato dalla (3.1); indichiamo con k, come nella definizione 3.5, l'intervallo di inseparabilità fissato per ognuno dei neuroni della rete;*
siano n i neuroni dello strato nascosto; sia inoltre B il valore che limita, in modulo, i pesi delle connessioni.
Se $\mathcal{N}$ separa due punti $\overline{x}, \overline{y} \in \mathbb{R}^n$ allora

$$d_1(\overline{x}, \overline{y}) \geq \frac{2 \ln \frac{Bn+k}{Bn-k}}{B} \tag{3.55}$$

Dim. Supponiamo che $\mathcal{N}$ separi i punti $\overline{x}$ ed $\overline{y}$ secondo la definizione 3.8; allora, per il teorema 3.2, varrà la (3.47), sarà cioè possibile individuare almeno un neurone h_q dello strato nascosto di $\mathcal{N}$ per il quale

$$\left| \frac{1}{1 + e^{-net_q(\overline{x})}} - \frac{1}{1 + e^{-net_q(\overline{y})}} \right| \geq \frac{k}{Bn} \tag{3.56}$$

dove $net_q(\overline{x})$ è definita nella (3.46).

[4] a titolo di curiosità riportiamo la derivata prima della (3.54) nel caso in cui $f = \frac{1}{1+e^{-x}}$:

$$y' = \frac{e^x + ge^x}{1 - g(e^x + 1)} - \frac{e^x + g(e^x + 1)(-ge^x)}{(e^x + g(e^x + 1))(1 - g(e^x + 1))} - 1$$

dove $g = \frac{k}{Bn}$.

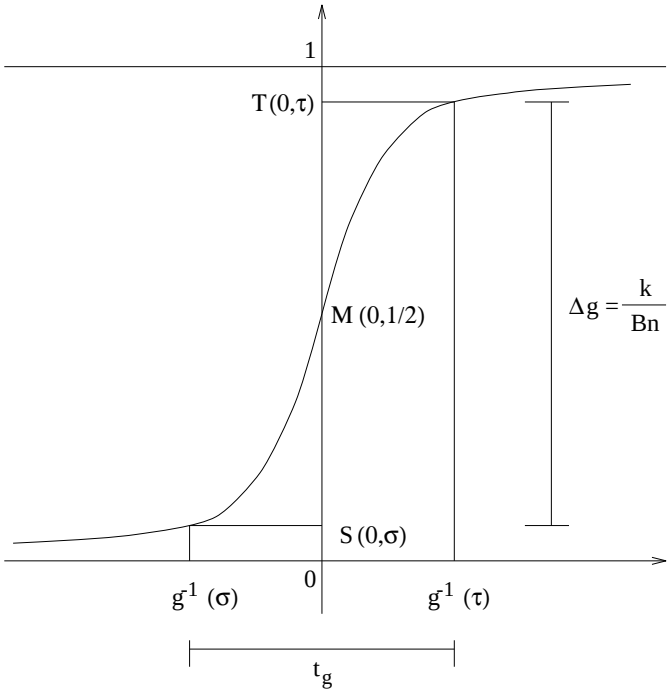

Figura 3.10: *Il segmento t_g può essere individuato grazie alla forma della curva.*

In virtù della proposizione 3.1, la distanza $d_1(\overline{x}, \overline{y})$ tra gli ingressi dovrà soddisfare la relazione

$$d_1(\overline{x}, \overline{y}) \geq \frac{t_g}{B} \tag{3.57}$$

dove $t_g = \min(|net_q(\overline{x}) - net_q(\overline{y})|)$ che rende vera la (3.56).

Nel nostro caso t_g si trova in corrispondenza del segmento Δg di misura k/Bn avente il suo centro nel punto $M\,(0, 1/2)$.

Gli estremi di Δg si troveranno dunque in corrispondenza dei punti S e T aventi ordinata

$$\sigma = \frac{1}{2} - \frac{k}{2Bn} \qquad \tau = \frac{1}{2} + \frac{k}{2Bn} \tag{3.58}$$

rispettivamente.

Il valore di t_g sarà allora così espresso:

$$t_g = \left| g^{-1}(\tau) - g^{-1}(\sigma) \right| \tag{3.59}$$

Calcoliamo $g^{-1}(\tau)$:

$$g^{-1}(\tau) = \ln \frac{\tau}{1-\tau} = \ln \frac{Bn+k}{Bn-k} \tag{3.60}$$

Analogamente

$$g^{-1}(\sigma) = \ln \frac{\sigma}{1-\sigma} = \ln \frac{Bn-k}{Bn+k} \tag{3.61}$$

Avremo quindi:

$$\begin{aligned} t_g &= \min(|net_q(\overline{x}) - net_q(\overline{y})|) = \left|g^{-1}(\tau) - g^{-1}(\sigma)\right| = \\ &= \left|\ln \frac{Bn+k}{Bn-k} - \ln \frac{Bn-k}{Bn+k}\right| = \\ &= \left|\ln \frac{Bn+k}{Bn-k} + \ln \frac{Bn+k}{Bn-k}\right| = \\ &= 2\left|\ln \frac{Bn+k}{Bn-k}\right| \end{aligned} \tag{3.62}$$

Essendo B, n e k tutti positivi, il logaritmo naturale che appare nell'ultimo membro della (3.63) rimarrà definito quando $Bn - k > 0$; ma questo è proprio quanto discende dalle ipotesi fatte e dal teorema 3.2, in base al quale deve aversi $n \geq k/B$.

Il logaritmo sarà dunque sempre definito; inoltre, per la positività di B, n e k, sarà pure maggiore di zero, cosicché saremo in grado di eliminare il modulo.

In definitiva avremo

$$t_g = 2 \ln \frac{Bn+k}{Bn-k} \tag{3.63}$$

e da qui la tesi, dopo aver sostituito la (3.63) nella (3.57). □

Il corollario, potendo sfruttare le ulteriori informazioni derivanti dalla conoscenza della funzione di attivazione, fornisce una condizione necessaria per la separabilità ancora più forte, pur se legata alla particolare rete in esame.

Esempio 3.4 Supponiamo di avere a disposizione una rete a due livelli di neuroni con funzione di trasferimento data dalla (3.40), e pesi limitati ad 1.

Quello che vorremmo fare è risolvere il problema dello XOR con una tolleranza $\varepsilon = 0.1$. Il nostro k_ε sarà uguale a 4.42.
Il teorema 3.2 ci richiede, in base alla (3.43), uno strato nascosto costituito da almeno 5 unità. "Eh, no!", risponde il corollario, "affinché la rete possa sperare di separare gli ingressi dello XOR ce ne vogliono di più, di unità nascoste ...".

Esplicitando infatti n nella (3.55) avremo:

$$n \geq \left\lceil \frac{k}{B} \frac{e^{\frac{B \cdot d(\overline{x},\overline{y})}{2}} + 1}{e^{\frac{B \cdot d(\overline{x},\overline{y})}{2}} - 1} \right\rceil \tag{3.64}$$

La (3.64) ci dice, ad es., che i punti $(0,0)$ ed $(1,1)$ non saranno in alcun caso separati dalla rete a meno di porre nello strato nascosto 10 unità.

Capitolo 4

Pesi Discreti

4.1 Introduzione

Nel corso del lavoro abbiamo più volte messo in evidenza come l'integrazione digitale dei pesi all'interno di un chip neurale li costringa a muoversi in un insieme finito di valori, in relazione al numero di bit a disposizione per ciascuno di essi.
Da questo limite non sarebbero esenti, a dire il vero, neanche le implementazioni software delle reti, la cui rappresentazione (in virgola mobile) delle variabili è pur sempre di natura "digitale". Tuttavia, il grande numero di bit che questa rappresentazione mette a disposizione per ogni parametro (almeno 32 nella maggior parte dei casi), consente ai pesi di poter assumere un numero elevatissimo di valori, e con una risoluzione tale da rendere ingiustificato un accostamento delle reti software a quelle digitali, dove normalmente i bit per peso sono intorno alla decina.

Diversi studi sperimentali, condotti attraverso simulazioni ma anche uti-

lizzando chip neurali in fase di studio o già disponibili a livello commerciale (vedere, ad es., [8, 10, 12, 20, 27, 36]), hanno messo in evidenza come una bassa risoluzione per i pesi conduca ad un degrado delle prestazioni delle reti discrete, tanto più evidente all'aumentare della complessità del problema da risolvere. Si è osservato inoltre come le capacità di queste reti possano essere reintegrate incrementando il numero dei neuroni oltreché, ovviamente, aumentando la risoluzione dei pesi.

È difficile, ad ogni modo, evincere da questi studi quali relazioni intercorrano tra risoluzione dei pesi, topologia e capacità complessive delle rete, vista la particolarità dei problemi di volta in volta affrontati; va inoltre tenuto presente che la difficoltà attuale che si riscontra nell'individuare algoritmi di apprendimento per questo tipo di reti che siano di provata efficienza, non permette di comprendere a fondo quanto le diminuite capacità di questi sistemi dipendano da fattori intrinseci e quanto dalla relativa efficacia degli algoritmi in questione.

Un'analisi teorica dei problemi in esame si presenta, parimenti, di insolita complessità; il lavoro che segue vuole essere un timido tentativo di affrontare la questione in modo sistematico, con lo scopo di offrire un quadro di riferimento entro cui valutare le scelte implementative finora proposte in letteratura.

Ci addentreremo dunque nello studio degli effetti che la discretizzazione dei valori dei pesi comporta sulle capacità di classificazione di una rete neurale; ci occuperemo in particolar modo delle ricadute che questa discretizzazione ha sul potere di classificazione del neurone a soglia, qualora lo si utilizzi per separare due ingressi dello spazio n-dimensionale.

4.2 Neurone a Soglia e Pesi Discreti

Abbiamo già osservato (vedi la sezione 3.2) come un neurone a soglia a due ingressi possa essere assimilato ad una retta nello spazio $\mathbb{R}^2$; in generale, potremmo vederlo come un iperpiano nello spazio n-dimensionale degli ingressi. Il neurone separerà due ingressi di $\mathbb{R}^n$ se, da un punto di vista geometrico, saremo in grado di collocare un iperpiano tra i due punti in esame; cosa che, potendo i pesi variare in $\mathbb{R}$, risulta sempre possibile.

Abbiamo inoltre fatto notare come le capacità del neurone rimangano intatte anche qualora i pesi associati agli ingressi possano variare soltanto in un intervallo prefissato della retta reale che contenga l'origine.

Tuttavia, nel caso in cui i pesi del neurone siano vincolati ad assumere un numero finito di valori, le sue potenzialità risultano pregiudicate. Nel piano cartesiano la limitazione imposta equivarrà a poter collocare la retta che il neurone rappresenta soltanto in determinate posizioni, e non è detto che queste siano sufficienti a separare due qualunque punti del piano.

Esempio 4.1 Sia V l'insieme $\{-2, -1, 0, 1, 2\}$, ed u un neurone a soglia la cui regola di propagazione sia espressa dalla solita equazione

$$net(\overline{w}, \overline{x}) = w_1 x_1 + w_2 x_2 + \theta \tag{4.1}$$

dove i pesi w_i e la soglia θ possano assumere soltanto valori di V.

Disegnamo sul piano cartesiano tutte le rette distinte che sarà possibile generare al variare dei parametri in V. A partire dalla figura 4.1, si vede come sia facile individuare coppie di punti del piano che non possono essere separate dal neurone u: basterà prendere entrambe i punti all'interno di una qualunque delle regioni determinate dalla suddivisione del piano indotta da

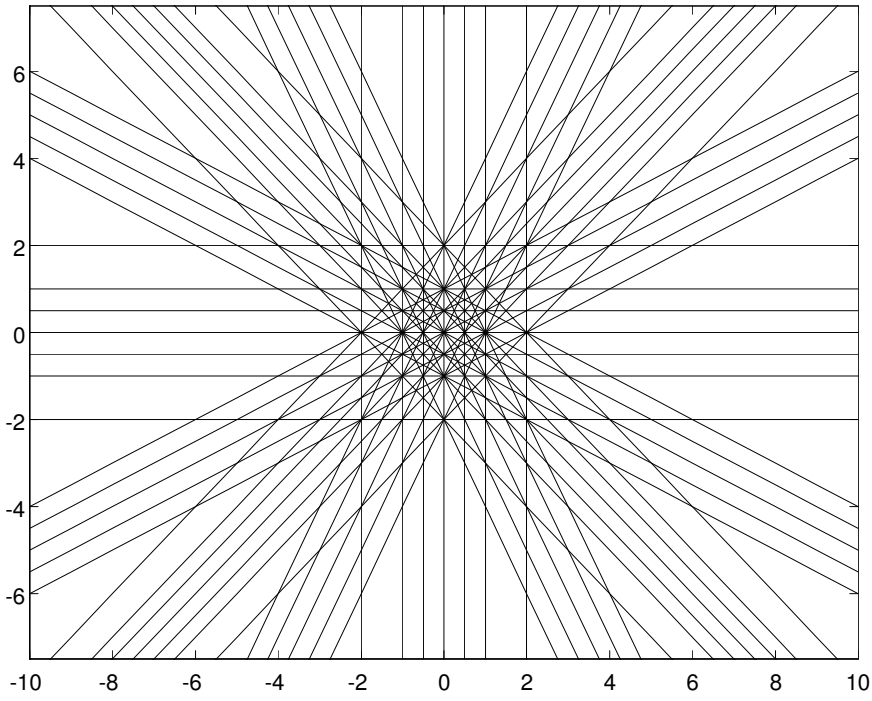

Figura 4.1: *Il neurone a soglia dell'esempio 4.1 induce la suddivisione del piano sopra riportata.*

u. Affinché i punti siano separati dal neurone essi dovranno cadere, invece, in due regioni distinte della suddivisione.

In che modo potremo giungere a dare una "misura" delle capacità di separazione di un neurone a soglia con pesi discreti?

Se teniamo in conto le considerazioni precedenti, un approccio **probabilistico** ci sembra naturale.

Seguiamo questa strada facendo nel piano le nostre riflessioni, per altro facilmente estendibili in $\mathbb{R}^n$.

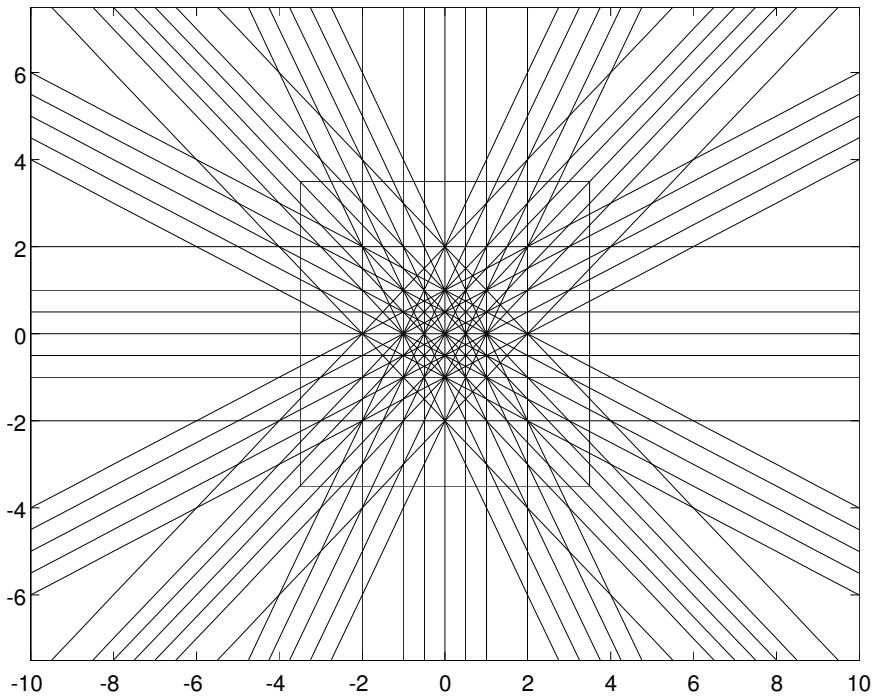

Figura 4.2: *Il quadrato all'interno della suddivisione viene diviso in un certo numero di regioni.*

4.2.1 Un Approccio Probabilistico

Supponiamo allora di individuare una regione del piano all'interno della quale gli ingressi siano distribuiti in modo uniforme. Scegliamo, per semplicità, un quadrato.

Introduciamo ora nel piano la suddivisione indotta da un neurone u a soglia i cui parametri possano assumere valori in un insieme V di cardinalità finita. Questa suddivisione determinerà all'interno del quadrato, purché questo sia stato scelto "abbastanza grande", un certo numero di regioni R_i (vedi la figura 4.2) di area A_i.

Stanti la scelta del quadrato[1] e del neurone u[2], è possibile definire **la probabilità $\mathbf{P_u}$ che il neurone separi due punti scelti a caso nel quadrato.** Se indichiamo con $\widehat{P}_u$ la probabilità che i due punti non siano separati dal neurone, la nostra probabilità sarà:

$$P_u = 1 - \widehat{P}_u. \tag{4.2}$$

Ma chi è $\widehat{P}_u$? In altri termini, quando due punti **non** potranno essere **in alcun modo** separati da u? Quando cadranno ambedue in una stessa regione R_i.
L'ipotesi di distribuzione uniforme per gli ingressi implica che la probabilità che un punto scelto nel quadrato cada in una regione di area A_i sia proprio A_i, posto che l'area del quadrato sia uguale ad 1. Nel caso più generale in cui l'area del quadrato sia A_{TOT}, la suddetta probabilità sarà data da A_i/A_{TOT}. D'ora in poi assumeremo, senza ledere le generalità, che $A_{TOT} = 1$.
Ora, se A_i è la probabilità che un punto cada nella regione R_i, la probabilità che un secondo punto cada nella stessa regione è data da A_i^2, essendo le scelte dei punti indipendenti (l'una dall'altra).
La probabilità $\widehat{P}_u$ che due punti non siano separati da u è proprio la probabilità che i due punti cadano entrambi in una qualunque delle regioni R_i del quadrato:

$$\widehat{P}_u = \sum_{i=1}^{N} A_i^2 \tag{4.3}$$

dove N indica il numero delle regioni all'interno del quadrato.

Il valore P_u espresso dalla (4.2), che resta associato al neurone una volta fissata la distribuzione degli ingressi può, a buon diritto, essere assunto come una misura delle sue capacità: se u e v sono due neuroni cui siano associate le probabilità P_u e P_v rispettivamente, e se $P_u > P_v$, allora il neurone u

[1] Più in generale potremo parlare di distribuzione degli ingressi.

[2] Il neurone, come abbiamo più volte ripetuto, genera, fissato il numero di bit per ciascun peso, una precisa suddivisione dello spazio in regioni.

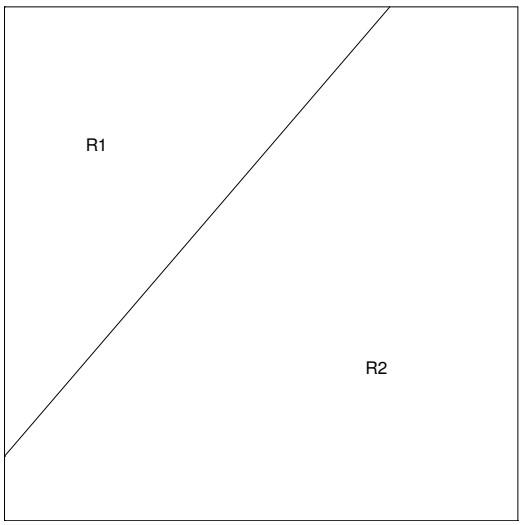

Figura 4.3: *Analisi del caso più semplice: il quadrato viene diviso due regioni.*

sarà "probabilmente" più bravo di v nel separare due punti scelti a caso nel quadrato.

4.2.1.1 Il neurone ideale

Vorremmo capire adesso come agire in modo da rendere "piccola" la probabilità $\widehat{P}$ data dalla (4.3).
A tal proposito osserviamo come la distribuzione delle aree nel quadrato incida sul valore di $\widehat{P}$.

Consideriamo il caso più semplice di un quadrato di lato 1 diviso in due regioni R_1 ed R_2 aventi, rispettivamente, area a_1 ed a_2 (vedi la figura 4.3). La probabilità $\widehat{P}$ sarà, in questo caso:

$$\widehat{P} \ = a_1^2 + a_2^2 = a_1^2 + (1 - a_1)^2 \tag{4.4}$$

Uguagliando a zero la derivata prima della $\widehat{P}$ rispetto ad a_1 troviamo il punto stazionario $a_1 = 1/2$, che è anche punto di minimo in virtù della convessità

della funzione $\widehat{P}$.

Dunque, in questo caso il minimo valore per $\widehat{P}$ si ottiene quando le aree a_1 ed a_2 sono uguali; in generale, quanto più le due regioni all'interno del quadrato saranno uniformi tanto più piccolo sarà il valore di $\widehat{P}$.

Questa semplice analisi tuttavia non può non far nascere il sospetto che, anche nel caso più generale in cui si abbiano N regioni nel quadrato unitario, **il minimo valore** per la probabilità si raggiunga quando tutte le regioni sono uguali.

Si dimostra, in effetti, il seguente

Teorema 4.1 *Siano $f, u : \mathbb{R}^n \longrightarrow \mathbb{R}$ così definite:*

$$f(p_1, \ldots, p_N) = \sum_{i=1}^{N} {p_i}^2 \quad e \quad u(p_1, \ldots, p_N) = \sum_{i=1}^{N} p_i; \tag{4.5}$$

Sia

$$G = \{(p_1, \ldots, p_N) : u(p_1, \ldots, p_N) = 1\}. \tag{4.6}$$

Allora

$$\min_G f = f\left(\frac{1}{N}, \ldots, \frac{1}{N}\right). \tag{4.7}$$

Dim. La funzione f si trova nelle ipotesi del teorema A.1; applicando dunque il *metodo dei moltiplicatori di Lagrange* saremo in grado di trovare i punti stazionari liberi per la f.

I punti di massimo e minimo vincolato dovranno essere ricercati tra le soluzioni del sistema

$$\begin{cases} \frac{\partial f}{\partial p_1} + \lambda \frac{\partial u}{\partial p_1} = 0 \\ \frac{\partial f}{\partial p_2} + \lambda \frac{\partial u}{\partial p_2} = 0 \\ \quad \vdots \\ \frac{\partial f}{\partial p_n} + \lambda \frac{\partial u}{\partial p_n} = 0 \\ u(p_1, \ldots, p_N) - 1 = 0 \end{cases} \tag{4.8}$$

Avremo dunque

$$\begin{cases} 2p_1 + \lambda = 0 \\ 2p_2 + \lambda = 0 \\ \quad \vdots \\ 2p_n + \lambda = 0 \\ \sum_{i=1}^{N} p_i = 1 \end{cases} \tag{4.9}$$

Dalle prime N equazioni si ricava $p_i = -\lambda/2$, per $i = 1, \ldots, N$; sostituendo i p_i nella $(N+1)$-esima equazione troviamo

$$\sum_{i=1}^{N} -\frac{\lambda}{2} = 1; \tag{4.10}$$

$$-\frac{N}{2}\lambda = 1; \tag{4.11}$$

$$\lambda = -\frac{2}{N} \tag{4.12}$$

da cui

$$p_i = \frac{1}{N}, \quad i = 1, \ldots, N. \tag{4.13}$$

Quindi, l'unico punto stazionario per la funzione f con vincolo G sarà

$$P_o = \left(\frac{1}{N}, \ldots, \frac{1}{N}\right), \tag{4.14}$$

che sarà anche punto di **minimo**, trovandosi la f nelle ipotesi del teorema A.2. □

Nel caso in cui le aree all'interno del quadrato siano uguali tra di loro, avremo

$$\widehat{P} = \sum_{i=1}^{N} A_i^2 = \sum_{i=1}^{N} \frac{1}{N^2} = \frac{1}{N} \tag{4.15}$$

Dalla (4.15) si vede pure che, essendo le aree uguali tra loro, la probabilità $\widehat{P}$ diminuisce all'aumentare del numero di regioni nel quadrato.

Osserviamo a questo punto che la $\widehat{P}$ definita dalla (4.3) vale anche nel caso di distribuzione uniforme degli ingressi in $\mathbb{R}^n$, pur di interpretare gli A_i come le *misure* di ciascuna regione all'interno dell'ipercubo unitario n-dimensionale.

Alla luce di quanto visto finora, è possibile trarre alcune conclusioni:

a) quello che specifica il potere di separazione del neurone a soglia è proprio il modo di disporre nello spazio gli iperpiani che il neurone stesso rappresenta;

b) il neurone "ideale" sarà quello in grado di generare una suddivisione dello spazio degli ingressi che abbia il maggior numero di regioni, con le misure quanto più possibile uguali tra loro. Ciò, come si è detto, rende massima la capacità P_u (probabilità) di separazione del neurone.

Ma cosa vorrà dire, matematicamente, andare alla ricerca di questo neurone? Mettiamoci in $\mathbb{R}^2$ e supponiamo che ciascun parametro abbia a disposizione t bit: ciò equivarrà ad **individuare una funzione** che associ una retta a ciascuna delle 2^{3t} combinazioni binarie ottenibili con $3t$ bit, in modo che la suddivisione del piano ottenuta collocandovi le rette abbia le qualità sopra elencate.

Fissato il numero dei bit per ciascun peso, questa funzione, che chiameremo *funzione di codifica*, **descrive completamente le caratteristiche del neurone cui viene associata**; per questo, nel seguito parleremo in modo equivalente di neurone e di codifica, potendo riferirci all'uno attraverso l'altra e viceversa.
Ugualmente, a livello notazionale scriveremo, a seconda dei casi, $\mathbf{u_f}$ per indicare il neurone u descritto dalla codifica f, oppure $\mathbf{f_u}$ per denotare la funzione

di codifica associata ad u. Ancora, si scriverà $\mathbf{P_u}$ (o anche $\mathbf{P(u)}$) oppure $\mathbf{P_f}$ (o anche $\mathbf{P(f)}$) a seconda che si voglia sottolineare il fatto che la probabilità di separazione è associata al neurone u oppure alla funzione di codifica che definisce il neurone stesso.

4.2.2 La Funzione di Codifica

Fino ad ora, le conseguenze che la scelta della codifica ha sulle capacità di classificazione di un neurone a soglia, pur di non poca importanza (almeno a prima vista), non sembra siano state dettagliatamente analizzate in letteratura.
Nella maggioranza dei casi questa scelta è stata dettata da esigenze di natura tecnica, o da motivi di "naturalità".
Questo non vuol dire, tuttavia, che le scelte fatte finora, sia per ciò che riguarda le realizzazioni hardware delle reti a pesi digitali sia per le loro simulazioni software, siano state "cattive"; soltanto non ci si è posti il problema di valutarne la bontà. La nostra speranza sarebbe quella di colmare tale lacuna, fornendo degli strumenti atti a valutare la qualità intrinseca delle varie codifiche; vorremmo affrontare il problema della migliore codifica in modo astratto, prescindendo dal contesto implementativo attuale e dai suoi vincoli, convinti come siamo che un esame a più ampio respiro della questione non possa che offrire nuovi spunti ai progettisti di chip neurali. Del resto nulla ci impedirà, in un secondo tempo, di sottomettere delle scelte di codifica ideali alle esigenze di natura implementativa.

Prima di passare alla definizione formale di funzione di codifica, vorremmo descrivere, a titolo di esempio, una codifica comunemente utilizzata sia a livello

hardware che nelle neuro-simulazioni delle reti digitali.

Esempio 4.2 Supponiamo che ad ogni parametro della rete spettino t bit; la legge di codifica associa all'insieme delle stringhe di t bit i valori

$$i = -2^{t-1} + 1, \ldots, -1, 0, 1, \ldots, 2^{t-1} - 1 \tag{4.16}$$

dove ogni i è la rappresentazione in base 10 dei primi $t-1$ bit della sua controimmagine binaria, con il t-esimo bit ad indicare il segno.
Ad esempio, se $t = 3$, l'insieme V dei valori possibili per ogni peso sarà

$$V = \{-3, -2, -1, 0, 1, 2, 3\} \tag{4.17}$$

Questa codifica, come si vede, è ridondante: due configurazioni di bit, infatti, codificano lo stesso valore 0; a fronte di uno spreco minimo, questa scelta tende a semplificare la progettazione dei circuiti neurali.

In realtà la codifica, così come è stata definita, potrebbe ancora non prestarsi ad una facile implementazione, in quanto i valori delle variabili nei circuiti debbono, di norma, sottostare a precisi vincoli di ampiezza.

Nel caso in cui i pesi siano limitati in modulo dal valore B, la codifica assocerà questa volta alle stringhe binarie i valori

$$w_i = i \cdot \frac{B}{2^{t-1} - 1}, \tag{4.18}$$

dove i valori di i sono dati dalla (4.16). Con $t = 3$ e $B = 2$, l'insieme dei valori assumibili da ogni peso diventa

$$V_{hw} = \left\{ w_i = i \cdot \frac{2}{3}, \quad i = -3, \ldots, -1, 0, \ldots, 3 \right\} \tag{4.19}$$

È interessante notare come sia la codifica a valori in V che quella a valori in V_{hw} inducano la stessa suddivisione dello spazio degli ingressi, simile nel suo sviluppo a quella di figura 4.1.
Il motivo diventa chiaro se (mettendoci in $\mathbb{R}^2$) osserviamo che ciascuna delle equazioni lineari generabili al variare dei parametri w_1, w_2, θ della (4.1) in V, corrisponde **a meno di un fattore** $\lambda = \mathbf{B}/\left(\mathbf{2^{t-1}} - \mathbf{1}\right)$ ad una delle equazioni generate al variare degli stessi coefficienti in V_{hw}. Abbiamo già ricordato (sezione 3.2) come una stessa retta possa essere descritta da infinite equazioni lineari i cui coefficienti differiscano tra loro per un fattore reale.

Il fatto che le due codifiche generino la stessa suddivisione del piano ci fa capire come lo studio delle codifiche possa prescindere, almeno nel caso di neuroni a soglia, dai problemi legati al *bound* sui pesi tipici delle implementazioni circuitali. In altre parole, una volta individuata una codifica teoricamente adeguata, qualora per esigenze implementative sia necessario manipolare valori più piccoli di quelli legati alla codifica, basterà scalarli in modo opportuno senza tema di influenzare il potere separatore della codifica stessa.

Veniamo alle definizioni.

Definizione 4.1 *Siano T l'insieme $\{0,1\}$ e t, m interi positivi; sia $A_t = \overbrace{T \times T \times \ldots \times T}^{t \text{ volte}}$; sia, inoltre, $A_t^m = \overbrace{A_t \times A_t \times \ldots \times A_t}^{m \text{ volte}}$;*

Si dice **codifica** *una funzione $f : A_t^m \longrightarrow C \subset \mathbb{R}^m$.* □

Definizione 4.2 *Sia $a \in A_t$; allora $a = (a_1, a_2, \ldots, a_t)$ dove $a_j = 0$ oppure $a_j = 1$; nel seguito scriveremo, in luogo della notazione sopra riportata, $a = a_1 a_2 \ldots a_t$; ad esempio, se $t = 4$, scriveremo 0111 al posto di $(0,1,1,1)$.*

Se $s \in A_t^m$*, allora sarà* $s = (s^1, s^2, \ldots, s^m)$ *dove* $s^i \in A_t$*; anche in questo caso scriveremo* $s = s^1 s^2 \ldots s^m$ *al posto della notazione classica. Ad esempio, se* $t = 4$ *ed* $m = 3$*, scriveremo* $0111\,0001\,1010$ *al posto di* $(0111, 0001, 1010)$*.*

Una funzione f *è detta* **codifica locale** *se esiste una funzione* $g : A_t \longrightarrow \mathbb{R}$ *tale che, per ogni* $s = s^1 s^2 \ldots s^m \in A_t^m$*,*

$$f(s) = f\left(s^1 s^2 \ldots s^m\right) = \Big(g(s^1),\, g(s^2), \ldots,\, g(s^m) \Big) \in \mathbb{R}^m, \tag{4.20}$$

altrimenti è detta **codifica globale**. □

Definizione 4.3 *Sia* $F_m = \{f,\ f : A_t^m \longrightarrow \mathbb{R}^m, t \in \mathbb{N}\}$ *l'insieme delle funzioni di codifica di* **rango m***; si dice* **classe di codifiche di rango m** *una successione* $\{f_t\}_{t \in N}$ *a valori in* F_m*.* □

Facciamo alcune osservazioni che servano a chiarire il senso delle precedenti definizioni.
Il dominio A_t, innanzitutto, denota l'insieme delle configurazioni binarie che fanno capo a ciascun parametro del neurone. Ogni elemento di A_t^m raccoglie, invece, tutti i bit che caratterizzano il neurone stesso.
Stando alla definizione 4.2 le codifiche proposte nell'esempio 4.2 sono locali, poiché un'unica funzione a valori in V (o V_{hw}) viene usata per codificare ciascuno dei parametri del neurone.
Al contrario, in una codifica globale, l'insieme degli m parametri del neurone viene visto come un solo oggetto, un'unica stringa binaria cui associare l'm-pla dei coefficienti dell'iperpiano.

A proposito del concetto di classe di codifiche, notiamo che l'insieme V dell'esempio 4.2, se pensato come funzione di t, descrive proprio una tale successione. Se, ad es., definiamo una funzione che associ all'insieme dei possibili

valori ottenibili con t bit i primi 2^t numeri primi, stiamo descrivendo, in realtà, una successione di funzioni di codifica al variare di t.

4.2.3 Il Problema della Migliore Codifica

Una volta fissato il numero di bit per ogni peso, per valutare quale sia la migliore codifica (il miglior neurone) per una certa distribuzione degli ingressi si potrebbe utilizzare un approccio che definiremmo "statico": potremmo calcolare, cioè, la probabilità P_f associata alle varie codifiche e scegliere la funzione con la maggiore probabilità di separazione.

D'altro canto, individuata una classe di codifiche, sarebbe interessante studiare l'andamento della probabilità ad esse associata, al variare del numero dei bit e per una determinata distribuzione degli ingressi.

Ovviamente, tra tutte, non potranno non attirare la nostra attenzione quelle classi in cui il potere separatore degli elementi cresca al crescere dei bit a disposizione per ciascun peso, classi cioè per le quali l'andamento della probabilità associata a ciascun elemento abbia caratteristiche di monotonia crescente rispetto a t.

Mettendo a confronto diverse successioni, potremmo scoprire quella la cui capacità di classificazione cresca "più velocemente" all'aumentare di t, per una certa distribuzione (nel nostro caso uniforme) dell'input.

Ciò di cui, in ultima analisi, vorremmo disporre sarebbe una classe di codifiche che abbia un "buon" andamento crescente della probabilità di separazione al crescere di t.

Decidiamo di seguire questo secondo approccio.
Considereremo nel seguito neuroni a soglia con due ingressi, e tuttavia alcuni risultati potranno valere anche nel caso n-dimensionale: quando sarà questo il caso lo metteremo in evidenza.

Per raggiungere il nostro scopo, una volta scelto il quadrato in cui la distribuzione degli ingressi sia uniforme, dovremo calcolare la probabilità

$$\widehat{P}_{f_t} = \sum_{i=1}^{N(f_t)} \left(\frac{A_{i,t}}{A_{TOT,t}} \right)^2 \tag{4.21}$$

per ciascuno degli elementi f_t della successione in esame. La notazione $N(f_t)$ e l'indice t presente negli elementi della sommatoria, ci ricordano come le regioni generate da ciascun elemento della successione dipendano, nel loro numero e nelle dimensioni, dalla codifica in esame.

Valuteremo, innanzitutto, quale sia l'andamento della probabilità nei casi estremi: per la classe, cioè, delle "migliori" codifiche al variare di t, che indicheremo con $\{f_t^{best}\}$, nonché per quella delle codifiche "peggiori", che sarà denotata da $\{f_t^{wrst}\}$. **Qualunque altra classe, a parità di rango, dovrà mostrare un andamento della probabilità che si collochi tra i primi due**.
In entrambi i casi le considerazioni svolte varranno per successioni di rango qualunque.

4.2.3.1 Il caso peggiore

Cominciamo dunque dalla situazione peggiore: le successioni di codifica peggiori saranno quelle in cui nessuno dei loro elementi dividerà l'ipercubo degli

ingressi; in tal caso, per ogni t, la (4.21) diventa semplicemente

$$\widehat{P}\left(f_t^{wrst}\right) = \sum_{i=1}^{1} \left(\frac{A_{TOT,t}}{A_{TOT,t}}\right)^2 = 1. \tag{4.22}$$

La probabilità di separare due ingressi distribuiti uniformemente nell'ipercubo sarà dunque **nulla** per queste codifiche.

4.2.3.2 Il caso migliore

La migliore classe di codifiche sarà, al contrario, quella in grado di indurre, per ogni t, la suddivisione dello spazio con il maggior numero di regioni; tutte le regioni generate, per ciascuna suddivisione, avranno pure la stessa misura. Abbiamo già sottolineato come queste due condizioni garantiscano la maggiore capacità di separazione da parte del neurone.

Al lettore attento non potrà, a questo punto, non sorgere qualche dubbio sulla esistenza di tale codifica.
Se prendiamo un foglio di carta e una penna (meglio una matita) e cerchiamo di disegnare delle rette in modo da avere sul foglio, per ogni retta aggiunta, sempre il numero massimo di regioni, ci accorgeremo presto della difficoltà di ottenere al tempo stesso una sia pur appossimativa uniformità tra le aree. Al contrario, cercando di ottenere aree uguali, sarà facile sacrificare a quest'ultima causa qualche regione.

In sostanza, sembra che una codifica che induca una suddivisione "ricca" di regioni, non possa farlo se non a spese dell'uniformità tra le aree. La migliore classe di codifiche "effettiva" sarà probabilmente quella con il miglior compromesso tra numero di regioni e uniformità tra le aree. Tutto questo, comunque, non ci crea problemi: anzi, l'andamento della probabilità della nostra

miglior classe di codifiche costituirà un limite invalicabile per qualunque altra successione, e dunque un *bound* dal punto di vista teorico.

Partiamo dunque dall'ipotesi che la nostra miglior classe riesca ad ottenere sempre regioni di uguale misura; cerchiamo di capire ora quale sia il numero massimo di regioni che è possibile generare avendo a disposizione m pesi discreti di t bit ciascuno.

In primo luogo, nulla ci vieta di pensare ad una codifica che generi, fissati gli interi m e t, il numero massimo di iperpiani distinti, cioè 2^{mt}. Basterà che tra le m-ple associate ad ognuna delle possibili stringhe di $m \cdot t$ bit non ce ne sia alcuna *equivalente* ad altre, secondo la definizione 3.1 data nella sezione 3.2.1. In modo formale, dovrà essere:

$$\# \left(C_t^{best} \right) = \# \left(C_t^{best} / \sim \right) \tag{4.23}$$

dove abbiamo indicato con C_t^{best} l'immagine di f_t^{best}, e la relazione $\sim$ è quella della definizione 3.1.

Avendo cura di non associare ad alcuna combinazione binaria gli iperpiani degeneri $[(\overline{0})]_\sim$ e $[(0, 0, \ldots, 0, 1)]_\sim$ potremo effettivamente disporre di 2^{mt} iperpiani distinti e dotati di "potere separatore".

Vale la pena di far notare che una codifica locale, utilizzando una sola funzione a valori in $\mathbb{R}$ per codificare i valori binari di ciascun peso, non sarà mai in grado di generare 2^{mt} iperpiani distinti: infatti 2^t elementi di $\mathbb{R}^m$ confluiranno nella classe di equivalenza $[(1, 1, \ldots, 1)]_\sim$. Potranno essere dunque $2^{mt} - 2^t + 1$ al massimo, gli iperpiani distinti generati da una codifica locale, a meno che l'unica funzione di codifica non contenga pure lo zero, nel qual caso altri $2^t + 1$ iperpiani si perderanno in corrispondenza delle classi $[(\overline{0})]_\sim$ e

$[(0, 0, \ldots, 0, 1)]_{\sim}$.

4.2.3.3 Una questione interessante

Ogni elemento della nostra classe $\{f_t^{best}\}$, oltre a generare il massimo numero di iperpiani distinti, li dovrà collocare in modo da ottenere con essi il più grande numero di regioni nello spazio ad $m-1$ dimensioni. Geometricamente si vede che questo è possibile aggiungendo ogni iperpiano in modo che risulti *sghembo* rispetto ad ognuno di quelli precedentemente collocati.

Se indichiamo con $R_n(r)$ il massimo numero di regioni ottenibili con r iperpiani $(n-1)$-dimensionali nello spazio ad n dimensioni, allora il legame tra regioni ed iperpiani può essere espresso dalla seguente relazione ricorsiva[3]:

$$R_n(r) = R_n(r-1) + R_{n-1}(r-1) \tag{4.24}$$

dove $R_0(r) = R_n(0) = 1$.
Partendo dalla (4.24) possiamo arrivare ad esprimere in modo non ricorsivo il valore di $R_n(r)$: nel caso in cui si abbia $n < r$, si trova che

$$R_n(r) = \sum_{i=0}^{n} \binom{r}{i}; \tag{4.25}$$

se invece $n \geq r$,

$$R_n(r) = \sum_{i=0}^{r} \binom{r}{i} = 2^r. \tag{4.26}$$

Quale sarà, nel piano cartesiano, il numero massimo di regioni generabili utilizzando r rette? Nel caso in cui $r > 2$,

$$\begin{aligned} R_2(r) &= \sum_{i=0}^{2} \binom{r}{i} = \binom{r}{0} + \binom{r}{1} + \binom{r}{2} = \\ &= 1 + r + \frac{r\,(r-1)}{2} = \\ &= 1 + \frac{r\,(r+1)}{2} = 1 + \sum_{i=1}^{r} i. \end{aligned} \tag{4.27}$$

[3]Per una giustificazione della (4.24) si veda in appendice

Se facciamo in modo che le 2^{3t} rette generate dalla nostra codifica si intersechino all'interno del quadrato degli ingressi, questo resterà diviso in $R_2\left(2^{3t}\right)$ regioni: dalla (4.28) avremo

$$\begin{aligned} R_2\left(2^{3t}\right) &= \frac{2^{3t}\left(2^{3t}+1\right)}{2}+1= \\ &= \frac{1}{2}\,2^{6t}+2^{3t-1}+1 \;\approx\; \frac{1}{2}\,2^{6t}=\frac{1}{2}\,r^2. \end{aligned} \tag{4.28}$$

Utilizzando la (4.28) possiamo calcolare la probabilità per il caso migliore:

$$\widehat{P}\left(f_t^{best}\right)=\sum_{i=1}^{N(f_t)}\left(\frac{A_{i,t}}{A_{TOT,t}}\right)^2=\frac{1}{N(f_t^{best})}\approx\frac{2}{2^{6t}}=\frac{2}{r^2}. \tag{4.29}$$

Nel piano cartesiano la probabilità $\widehat{P}_{f_t}$, per qualunque successione di codifiche, non potrà avere un andamento decrescente migliore di quello dato dalla (4.29).

Tenendo presente l'osservazione a pagina 82, possiamo scrivere l'espressione della migliore probabilità nel caso generale:

$$\widehat{P}\left(f_t^{best}\right)=\frac{1}{N(f_t^{best})}=\frac{1}{R_n\left(2^{(n+1)t}\right)}=\frac{1}{\sum_{i=0}^{n}\binom{2^{(n+1)t}}{i}}. \tag{4.30}$$

4.2.4 Alcune Codifiche Notevoli

Passiamo ora allo studio di alcune classi di codifica semplici da analizzare e tuttavia interessanti ai fini di un confronto con i risultati finora ottenuti.

4.2.4.1 La codifica a barre

Cosideriamo in primo luogo la successione di codifiche che chiameremo "a barre", e che indicheremo con la notazione $\{f_t^{prsn}\}$.

In questa successione ogni elemento genererà 2^{mt} iperpiani distinti paralleli

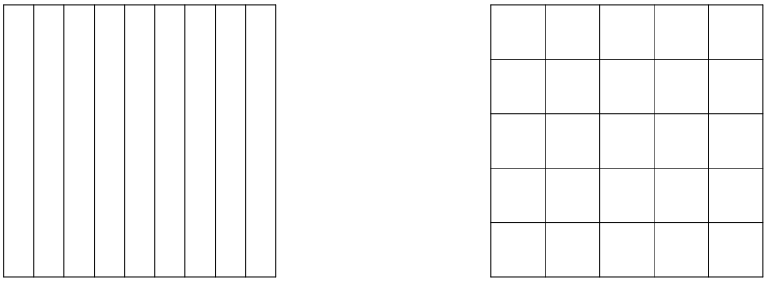

Figura 4.4: *Le codifiche "a barre" e "a griglia".*

tra loro: fissato il numero degli iperpiani, questa è la disposizione che genera il numero minimo di regioni.

Supporremo inoltre che le $2^{mt}+1$ regioni generate si trovino all'interno dell'ipercubo degli ingressi e siano uguali tra loro.

La probabilità $\widehat{P}$, associata ad ogni elemento della successione, sarà

$$\widehat{P}\,(f_t^{prsn}) = \frac{1}{N(f_t^{prsn})} = \frac{1}{2^{mt}+1} \approx \frac{1}{r}. \tag{4.31}$$

Il risultato è, ovviamente, valido anche per $\mathbb{R}^2$. Il fatto che, nel piano, la probabilità $\widehat{P}$ decresca così lentamente rispetto a quella calcolata nel caso migliore, è dovuta al basso numero di regioni presenti nella suddivisione.

4.2.4.2 La codifica a griglia

Analizziamo ora la classe di codifiche che chiameremo "a griglia", e che denoteremo con $\{f_t^{grid}\}$. Condurremo l'analisi in $\mathbb{R}^2$.

Nella successione a griglia di rango 3, ogni elemento della successione genera 2^{3t} rette distinte, 2^{3t-1} delle quali saranno disposte orizzontalmente e le altre verticalmente nel quadrato degli ingressi, a formare regioni di stessa area.

Classi	*Andamento della probabilità* $\widehat{P}$					
	con r		*con N*		*con t*	
di codifica	In $\mathbb{R}^2$	In $\mathbb{R}^n$	In $\mathbb{R}^2$	In $\mathbb{R}^n$	In $\mathbb{R}^2$	In $\mathbb{R}^n$
worst	1					
best	$\approx \frac{2}{r^2}$		$\frac{1}{N}$		$\approx \frac{2}{2^{6t}}$	$\frac{1}{\sum_{i=0}^{n}\binom{2^{(n+1)t}}{i}}$
prison	$\approx \frac{1}{r}$		$\frac{1}{N}$		$\frac{1}{2^{3t}+1}$	$\frac{1}{2^{(n+1)t}+1}$
grid	$\approx \frac{4}{r^2}$		$\frac{1}{N}$		$\approx \frac{4}{2^{6t}}$	

Tavola 4.1: *L'andamento di* $\widehat{P}$ *per le classi analizzate finora.*

È facile vedere che le regioni all'interno del quadrato saranno

$$N\left(f_t^{grid}\right) = \left(2^{3t-1}+1\right)^2 = 2^{6t-2} + 2^{3t} + 1 \approx 2^{6t-2}. \tag{4.32}$$

Per questa successione la probabilità diventa

$$\widehat{P}\left(f_t^{grid}\right) = \frac{1}{N(f_t^{grid})} \approx \frac{1}{2^{6t-2}} \approx \frac{4}{r^2}. \tag{4.33}$$

È notevole osservare come, nel piano, l'andamento della probabilità della codifica a griglia, codifica "effettiva", **non si discosti che per un fattore 2** da quello del caso migliore.

Nella tavola 4.1 riportiamo l'andamento della probabilità per ognuna delle classi di codifica finora analizzate, rispetto ai parametri di rilievo: gli iperpiani r, le regioni N ed i bit per peso t.

4.2.4.3 Il modello PQ

La classe di codifiche a griglia può essere "simulata" da un modello di suddivisione dell'ipercubo n-dimensionale che chiameremo "modello PQ". Il modello prevede una suddivisione costruttiva dell'ipercubo di lato unitario in regioni, utilizzando degli iperpiani paralleli alle facce dell'ipercubo stesso.
La suddivisione avviene secondo la regola che andiamo a descrivere.

Al primo passo di suddivisione su ciascuno degli spigoli dell'ipercubo viene individuato un punto P_i, in modo che lo spigolo i resti diviso in due parti di misura p e $q = 1 - p$ rispettivamente, con $0 \leq p \leq 1$. Un iperpiano ortogonale allo spigolo di appartenenza viene quindi fatto passare per ciascuno dei punti $P_i,\ i = 1, \ldots, n$. L'ipercubo rimarrà diviso in 2^n regioni.
Al passo successivo, ciascuno dei segmenti staccati su ogni spigolo dai punti P_i verrà a sua volta diviso in proporzioni p e q, e nei punti individuati verranno, ugualmente, fatti passare iperpiani ortogonali agli spigoli di volta in volta

	p*q^3		q^4
p^3*q	p^2*q^2		p*q^3
p^4	p^3*q		p^2*q^2

Figura 4.5: *Modello PQ.*

considerati. Reiterando il processo, al generico passo k:

- l'ipercubo sarà attraversato da

$$r_k = n\,(2^k - 1) \tag{4.34}$$

iperpiani (dove il fattore $2^k - 1$ è la somma dei primi k termini della serie geometrica[4] di *ragione* 2), e resterà diviso in

$$N_k = 2^{\,nk} \tag{4.35}$$

regioni;

- le misure di ciascuna regione saranno date dagli addendi trovati sviluppando l'espressione $(\,p+q\,)^{\,nk}$:

$$1 = (\,p+q\,)^{\,nk} = \sum_{i=0}^{nk} \binom{nk}{i} p^i q^{nk-i}; \tag{4.36}$$

- per la probabilità $\widehat{P}_k$ si avrà:

$$\begin{aligned} \widehat{P}_k &= \sum_{i=1}^{N} A_i^2 = \sum_{i=0}^{nk} \binom{nk}{i} \left(p^i q^{nk-i}\right)^2 \\ &= \sum_{i=0}^{nk} \binom{nk}{i} p^{2\,i} q^{2\,(nk-i)} = \\ &= \sum_{i=0}^{nk} \binom{nk}{i} \left(p^2\right)^i \left(q^2\right)^{nk-i} = \left(p^2+q^2\right)^{\,nk}. \end{aligned} \tag{4.37}$$

La figura 4.5 mostra la situazione in $\mathbb{R}^2$, al passo $k = 2$, per $p = 1/3$.

È possibile far vedere in modo esplicito quale sia l'andamento di $\widehat{P}$ in funzione del numero di regioni generate al passo k. Passando ai logaritmi nella (4.38) si avrà:

$$\log_2 \widehat{P}_k = n\,k \log_2 \left(p^2+q^2\right) \tag{4.38}$$

[4]Ricordiamo che la sommatoria $s_n = 1 + q + q^2 + \cdots + q^{n-1} = \sum_{i=0}^{n-1} q^i$, somma dei primi n termini della serie geometrica di ragione q, vale $\frac{1-q^n}{1-q}$

Ponendo

$$\alpha = -\log_2 \left(p^2 + q^2 \right) \tag{4.39}$$

ed osservando che, dalla (4.35),

$$n\,k = \log_2 N_k \tag{4.40}$$

potremo scrivere

$$\log_2 \widehat{P}_k = -\alpha \log_2 N_k \tag{4.41}$$

da cui

$$\widehat{P}_k = \frac{1}{N_k^{\alpha}}. \tag{4.42}$$

Infine, la probabilità $\widehat{P}_k$ in funzione del passo k di suddivisione, sostituendo la (4.35) nella (4.42), sarà

$$\widehat{P} = \frac{1}{2^{\,nk\alpha}} \tag{4.43}$$

dove α è dato dalla (4.39).

Scegliendo $p = 1/2$, il che rende le regioni della suddivisione uguali tra loro, si ha $\alpha = 1$ ed il valore di $\widehat{P}_k$ diventa $1/N$, risultato che concorda con la (4.15).

Se esprimiamo il passo di suddivisione k in funzione dei parametri n (dimensione dello spazio degli ingressi) e t (numero dei bit per peso), **è possibile utilizzare il modello PQ per simulare una codifica a griglia nello spazio n-dimensionale**.

Supponendo di avere a disposizione una codifica che generi 2^{mt} iperpiani distinti con t bit ed m pesi, e sapendo che $m = n + 1$, risolviamo rispetto a k l'equazione

$$2^{\,(n+1)\,t} = n \left(2^{\,k} - 1 \right) \tag{4.44}$$

dove al secondo membro compaiono le rette presenti nella suddivisione pq al k-esimo passo. Risolvendo si ha:

$$2^{\,k} = \frac{2^{\,(n+1)\,t}}{n} + 1; \tag{4.45}$$

da cui

$$\begin{aligned} k &= \log_2\left(\frac{2^{(n+1)t}}{n}+1\right) \approx \log_2\left(\frac{2^{(n+1)t}}{n}\right) = \\ &= \log_2\left(2^{(n+1)t}\right) - \log_2 n \end{aligned} \tag{4.46}$$

Avremo, in definitiva,

$$k \approx (n+1)\,t - \log_2 n. \tag{4.47}$$

Con $\alpha = 1$ e k dato dalla (4.47), il modello PQ simula il comportamento della classe di codifiche a griglia di rango m.

In altre parole, posto $\alpha = 1$, la probabilità associata ad una codifica a griglia $f_t^{grid} \in F_m$ sarà "circa" uguale a quella associata alla suddivisione pq al passo k, dove k è dato dalla (4.47). Un discorso analogo può essere fatto per le regioni[5].

Si avrà, in sostanza,

$$N\left(f_t^{grid}\right) = \frac{1}{\widehat{P}(f_t^{grid})} \approx N_k = \frac{1}{P_k}. \tag{4.48}$$

Calcoliamo N_k:

$$N_k \approx 2^{n\,(n+1)\,t - n\log_2 n} = \left(\frac{2^{(n+1)\,t}}{2^{\log_2 n}}\right)^n = \left(\frac{2^{(n+1)\,t}}{n}\right)^n. \tag{4.49}$$

Da cui

$$P_k \approx \left(\frac{n}{2^{(n+1)\,t}}\right)^n = \left(\frac{n}{r}\right)^n. \tag{4.50}$$

Per $n = 2$ si vede che i risultati del modello PQ sono gli stessi mostrati dalla classe a griglia di rango 3.

Aggiorniamo i valori della tavola 4.1, riportando nella tavola 4.2 i risultati relativi al modello PQ per $\alpha = 1$ e k dato dalla (4.47).

[5] L'uguaglianza del numero di iperpiani nei due casi discende dall' aver imposto la condizione (4.44)

Classi di codifica	*Andamento della probabilità* $\widehat{P}$					
	con r		*con N*		*con t*	
	In $\mathbb{R}^2$	In $\mathbb{R}^n$	In $\mathbb{R}^2$	In $\mathbb{R}^n$	In $\mathbb{R}^2$	In $\mathbb{R}^n$
worst	1					
best	$\approx \frac{2}{r^2}$		$\frac{1}{N}$		$\approx \frac{2}{2^{6t}}$	$\frac{1}{\sum_{i=0}^{n}\binom{2^{(n+1)t}}{i}}$
prison	$\approx \frac{1}{r}$		$\frac{1}{N}$		$\frac{1}{2^{3t}+1}$	$\frac{1}{2^{(n+1)t}+1}$
grid	$\approx \frac{4}{r^2}$		$\frac{1}{N}$		$\approx \frac{4}{2^{6t}}$	
PQ	$\approx \left(\frac{n}{r}\right)^n$		$\frac{1}{N}$		$\approx \left(\frac{n}{2^{(n+1)\,t}}\right)^n$	

Tavola 4.2: *L'andamento di* $\widehat{P}$ *per le classi analizzate: sono stati inseriti i valori ricavati dal modello PQ per* $\alpha = 1$ *e* k *dato dalla (4.47).*

Osservazione: avremmo, in effetti, potuto ricavare i valori espressi dalle (4.49) e (4.50) in modo diretto, estendendo il semplice ragionamento svolto per arrivare alla (4.32). Abbiamo preferito usare il modello perché di carattere più generale. La teoria del modello consente, infatti

- di simulare classi di codifiche che non generano il massimo numero di iperpiani (sarà necessario calcolare il valore di k appropriato);
- di simulare successioni che non dividono lo spazio in modo uniforme, soltanto facendo variare il valore di α o, il che è lo stesso, il valore di p.

4.2.5 La Fase Sperimentale

Accanto all'analisi teorica sopra esposta, è stata condotta l'analisi sperimentale di una delle codifiche più utilizzate nelle implementazioni discrete delle reti neurali, cioè la codifica "naturale" presentata nell'esempio 4.2. Abbiamo studiato inoltre alcune varianti della suddetta codifica, che descriveremo nel seguito, con l'obiettivo di individuare dei neuroni che esibiscano una capacità di separazione migliore di quella mostrata dalla codifica naturale. L'analisi è stata fatta per il solo caso di un neurone a soglia con due ingressi.

Nell'analisi di queste codifiche ci siamo avvalsi di un programma C scritto appositamente (per una descrizione più approfondita del programma rimandiamo in appendice), che ci ha consentito di calcolare, al variare del numero t dei bit per peso, la probabilità $\widehat{P}_{f_t}$ di ciascuna delle suddivisioni in esame, fino al punto in cui i tempi di calcolo delle macchine a nostra disposizione sono diventati inaccettabili.
Ad onor del vero lo studio dell'andamento della probabilità delle diverse classi è stato fatto al variare del numero dei valori discreti a disposizione per ognuno dei pesi, e questo per alcune ragioni:

- per avere una descrizione più "ricca" dell'andamento stesso;
- le simulazioni software delle reti discrete non sono costrette a scegliere il numero dei valori discreti tra le potenze di due;
- le stesse reti circuitali, come già osservato, spesso sacrificano qualche configurazione di bit a favore di una maggiore facilità di implementazione.

Esprimeremo dunque **i valori discreti** associati alle varie codifiche **in fun-**

zione di un intero h, che almeno per la codifica naturale rappresenterà il massimo valore intero, in modulo, che i pesi potranno assumere.

Ai fini del calcolo della probabilità $\widehat{P}$, per ciascuna delle classi di codifica esaminate abbiamo dovuto scegliere il quadrato entro cui la distribuzione degli ingressi fosse (per ipotesi) uniforme. Abbiamo deciso, per tutte le suddivisioni, di considerare il minimo quadrato che fosse attraversato da ogni retta della suddivisione stessa. Nel nostro caso si tratta del quadrato di lato $2h$ centrato nell'origine del piano cartesiano. La scelta del quadrato è stata effettuata su basi euristiche, principalmente osservando quale fosse, nel piano, la distribuzione delle rette delle diverse suddivisioni. Va dunque tenuto presente che **non si tratta di una scelta ottimale**, nel senso che potrebbe non valorizzare appieno o in ugual maniera il potere separatore delle classi esaminate.
Osservando, ad es., la figura 4.1 si vede che, scegliendo un quadrato più grande di quello in questione avremmo certamente "inglobato" più regioni, a spese tuttavia della loro uniformità; scegliendo un quadrato più piccolo le regioni al suo interno sarebbero state più omogenee, ma anche in numero inferiore.

4.2.5.1 La codifica naturale

La codifica naturale, che indicheremo anche con la lettera A, al variare di h associa a ciascun peso valori dall'insieme

$$V_h = \{-h, \ldots, 0, \ldots, h\}\,. \tag{4.51}$$

La codifica appartiene dunque al gruppo delle codifiche locali. Nella figura 4.1 mostriamo la suddivisione del piano indotta dalla codifica per $h = 2$. Nella tavola 4.3 riportiamo invece i valori elaborati dal programma di calcolo per $h = 2, \ldots, 10$. Nelle diverse colonne troviamo i valori discreti d per ciascun

Codifica naturale (A)				
h	*d*	*rette*	*regioni*	*probabilità* $\widehat{P}$
2	5	48	400	4.55186900e-03
3	7	144	4192	8.14760200e-04
4	9	288	18016	2.84517067e-04
5	11	576	74600	9.95617035e-05
6	13	864	172488	5.38799740e-05
7	15	1440	489504	2.51626280e-05
8	17	2016	970272	1.51040086e-05
9	19	2880	2001456	8.78351360e-06
10	21	3744	3416056	5.90288112e-06

Tavola 4.3: *Risultati relativi alla codifica naturale.*

peso, le rette che attraversano il quadrato degli ingressi, le regioni da esse generate, la probabilità $\widehat{P}$ associata alla suddivisione del quadrato.

Il fatto che l'andamento della probabilità $\widehat{P}$ nel modello PQ sia $1/N^{\alpha}$, ci ha portato a credere che la relazione esistente tra la probabilità $\widehat{P}_A$ e numero di regioni nella codifica naturale potesse essere simile, cioè del tipo

$$\widehat{P}_A = \frac{c}{N^{\alpha}}. \tag{4.52}$$

Poiché, passando ai logaritmi nella (4.52) si avrebbe

$$\log \widehat{P}_A = \log \frac{c}{N^{\alpha}} = -\alpha \log N + \log c, \tag{4.53}$$

affinché la congettura (4.52) sia valida, la relazione tra il logaritmo dei valori della colonna 3 e quello dei valori in colonna 4 nella tavola 4.3 dovrebbe essere lineare.

Effettivamente, mettendo su di un grafico in scala logaritmica i valori in

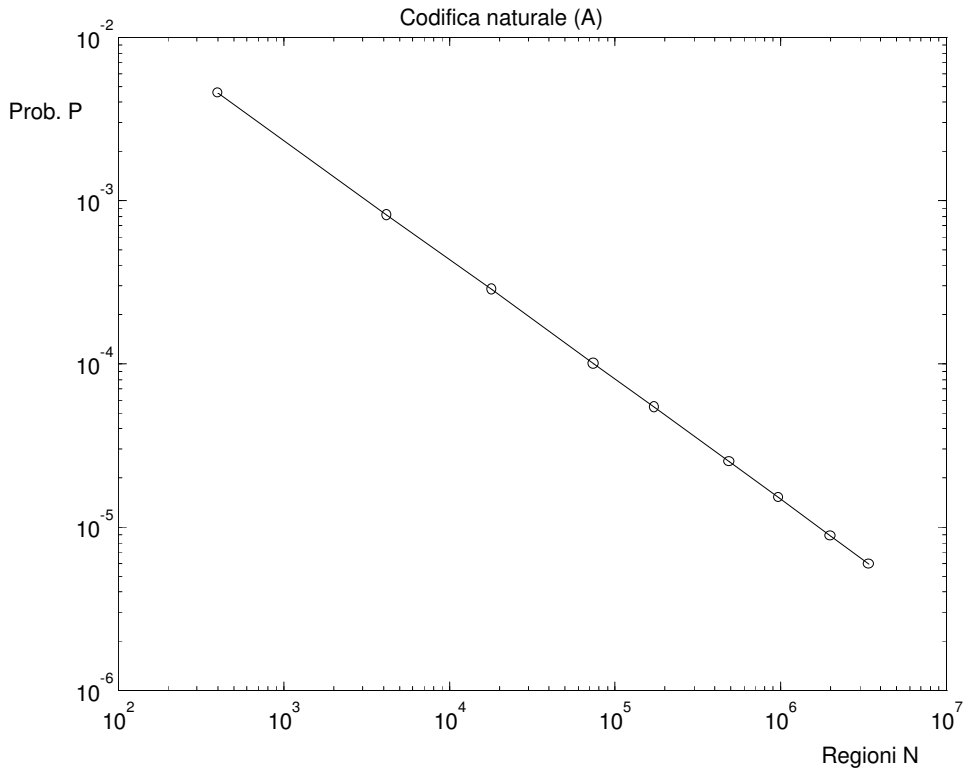

Figura 4.6: *Andamento di $\widehat{P}$ per la codifica A rispetto al numero di regioni.*

oggetto, si vede come i punti individuati si trovino, con una sorprendente approssimazione, su di una retta (figura 4.6).

Se si approssimano i punti a disposizione con un polinomio di primo grado (secondo il metodo dei *minimi quadrati*) si trovano, per α e c i valori

$$\alpha \approx 0.734, \qquad c \approx 0.373. \tag{4.54}$$

In sostanza, nel caso in esame avremo per la probabilità:

$$\widehat{P}_A \approx \frac{0.373}{N^{0.734}}. \tag{4.55}$$

Va osservato come la codifica naturale generi, al variare dei pesi in V_h, un numero di rette assai inferiore al numero massimo di 2^{3t} rette ottenibili da

una codifica di rango 3. Vorremmo sapere quale sia il numero di queste rette. Affrontiamo, comunque, la questione in termini più generali, vale a dire per successioni di codifiche naturali di rango qualunque.

Consideriamo allora la classe $\{f_h^A\}_{h\in N}$ delle codifiche naturali di rango m. Fissato h, sia $C_h^A \subset I\!R^m$ l'immagine di f_h^A. Facciamo notare che

$$C_h^A = \overbrace{V_h \times V_h \times \ldots \times V_h}^{m \ volte}, \tag{4.56}$$

e che, quindi,

$$\#(C_h^A) = (2h+1)^m. \tag{4.57}$$

Poniamo, per comodità,

$$J_h(k) = (2h+1)^k. \tag{4.58}$$

Quozientiamo ora l'insieme C_h^A rispetto alla relazione $\sim$ definita in precedenza; per quanto visto nella sezione 3.2.1, il numero r degli iperpiani distinti generati da f_h^A corrisponde proprio alla cardinalità di $C_h^A/\sim$; quale sarà questo numero?

Osserviamo intanto che, per la stessa definizione di C_h^A,

$$\forall a \in C_h^A,\ a \neq 0,\ \exists\, b \in C_h^A : b = -a. \tag{4.59}$$

Dalla (4.59) si vede pure che, essendo $a\sim b$ ($\lambda = -1$), ciascun[6] elemento di $C_h^A/\sim$ dovrà contenere almeno due m-ple, le cui rispettive componenti avranno segni opposti. Per quanto è stato detto, gli iperpiani distinti ottenibili dalla codifica potranno essere al più di $(J_h(m)+1)/2$.
Se indichiamo con

$$\overline{C_h^A} = \left\{(a_1, \ldots, a_m) \in C_h^A : M.C.D.(a_1, \ldots, a_m) = 1\right\}, \tag{4.60}$$

[6]Tutti meno la classe di equivalenza $[\overline{0}]_\sim$.

e con $\overline{C_h^A/\sim}$ l'insieme $C_h^A/\sim - \{[0]_\sim\}$, si può far vedere che (per la dimostrazione rimandiamo in appendice)

$$\#\left(\overline{C_h^A/\sim}\right) = \frac{\#(\overline{C_h^A})}{2}, \tag{4.61}$$

dove il fattore 1/2 è dovuto al fatto che, in ciascuna classe di equivalenza di $\overline{C_h^A/\sim}$, esisteranno due (e due soltanto) m-ple coprime con le rispettive componenti opposte in segno.

Dunque, si avrà

$$r = \#\left(C_h^A/\sim\right) = \#\left(\overline{C_h^A/\sim}\right) + 1 = \frac{\#(\overline{C_h^A})}{2} + 1. \tag{4.62}$$

Non è possibile dare una semplice espressione analitica della cardinalità di $\overline{C_h^A}$, a causa dell'irregolarità con cui la proprietà di *coprimalità*[7] è distribuita in $\mathbb{R}^m$. Se indicassimo con P_m^h la probabilità che m interi scelti a caso nell'intervallo $[-h,\, h]$ siano coprimi, avremmo

$$\begin{aligned} P_m^h &:= \frac{\#\overline{C_h^A}}{\#(C_h^A)}; \\ \#\overline{C_h^A} &= P_m^h\, J_h(m). \end{aligned} \tag{4.63}$$

Posto

$$W_k^h = Prob\left\{M.C.D.(a_1, \ldots, a_k) = 1,\ a_i \in \{1, 2, \ldots, h\}\right\}, \tag{4.64}$$

ed inoltre

$$W_k = \lim_{h\to\infty} W_k^h \tag{4.65}$$

si può dimostrare (vedi [32], pagg. 83–86) che

$$W_k \approx \frac{1}{\zeta(k)}, \tag{4.66}$$

[7]Ricordiamo che n interi $a_1, \ldots, a_n$ si dicono *primi tra loro (o coprimi)* se il loro *massimo comun divisore* è 1.

Rapporto tra $\#(C_h^A/\sim)$ *e* $\#(C_h^A)/2$ $\quad (1/\zeta(3) = 0.832)$							
h	2	5	10	20	30	50	100
$\div$	0.840	0.872	0.822	0.830	0.825	0.827	0.825

Tavola 4.4: *Si confrontino i risultati presentati con il valore* $1/\zeta(3)$.

dove $\zeta(s)$ è la funzione *zeta di Riemann* così definita:

$$\zeta(s) := \sum_{n=1}^{\infty} \frac{1}{n^s}. \tag{4.67}$$

In linea di principio non sarebbe possibile approssimare il valore P_m^h con il valore $\frac{1}{\zeta(m)}$, per due ragioni:

1. il valore $\frac{1}{\zeta(m)}$ esprime un limite per $h \to \infty$;
2. nella definizione (4.64) gli a_i non possono assumere il valore zero, che invece può comparire nelle m-ple di $\overline{C_h^A}$.

Tuttavia, la verifica sperimentale rivela un buon accordo tra P_m^h e $\frac{1}{\zeta(m)}$, come si vede dai valori riportati nella tavola 4.4.
Questo ci consente di scrivere

$$\begin{aligned} r &= \#\left(C_h^A/\sim\right) = \\ &= \#\left(\overline{C_h^A/\sim}\right) + 1 = \frac{\#(\overline{C_h^A})}{2} + 1 \approx \\ &\approx \frac{1}{\zeta(m)} \frac{J_h(m)}{2}. \end{aligned} \tag{4.68}$$

Per $m = 2$ si ha $\frac{1}{\zeta(2)} = 0.608$, per $m = 3$ si ha $\frac{1}{\zeta(3)} = 0.832$, per $m = 4$ si ha $\frac{1}{\zeta(4)} = 0.932$. Al crescere di m il valore di $\frac{1}{\zeta(m)}$ tende velocemente ad 1; questo ci dice che per m "grande", delle $J_h(m)$ m-ple presenti nell'immagine C_h^A della codifica f_h^A, circa $J_h(m)/2$ danno luogo ad iperpiani distinti.

4.2.5.2 La codifica B

La codifica naturale spreca, come si è visto, almeno la metà delle m-ple, tra quelle generate al variare dei pesi in V_h.

Abbiamo allora voluto analizzare una codifica, che indicheremo con B, leggermente differente e che tuttavia, eliminando i problemi di simmetria legati al segno, genera un maggior numero di iperpiani distinti. In $\mathbb{R}^2$, la codifica B associa al primo peso e alla soglia valori dallo stesso insieme V_h definito nella (4.51), mentre il peso associato al secondo ingresso assume valori nell'insieme

$$V_h^B = \{1, \ldots, 2h+1\}\,. \tag{4.69}$$

I risultati ottenuti per la codifica B sono riportati nella tavola 4.5.

I dati relativi alla probabilità $\widehat{P}$ mostrano un miglioramento rispetto a quelli della codifica naturale. Anche in questo caso la relazione esistente tra numero di regioni e probabilità è del tipo c/N^α, dove

$$\alpha \approx 0.736, \qquad\qquad c \approx 0.951. \tag{4.70}$$

Come si vede, l'andamento della probabilità è praticamente lo stesso della codifica naturale; il miglioramento dei valori di $\widehat{P}$ risulta, quindi, essere legato al maggior numero di regioni generate a parità di valori discreti.
Si trova sperimentalmente che il numero degli iperpiani distinti generati dalla codifica B è ben stimato dal valore $J_h(m)/\zeta(m)$. Questo numero è, dunque, circa 2 volte quello degli iperpiani distinti generati dalla codifica naturale.

♣		*Codifica B*			*Codifica C*		
h	*d*	*rette*	*regioni*	*probabilità* $\widehat{P}$	*rette*	*regioni*	*probabilità* $\widehat{P}$
2	5	105	2216	3.2018e-03	88	1436	2.7445e-03
3	7	297	19444	6.6303e-04	264	14640	5.2440e-04
4	9	601	83548	2.3501e-04	552	68140	1.8789e-04
5	11	1161	319412	8.4069e-05	1080	268896	6.6514e-05
6	13	1801	783124	4.4199e-05	1680	666304	3.6097e-05
7	15	2881	2033532	2.0829e-05	2760	1829140	1.6871e-05

Tavola 4.5: *Risultati relativi alle codifiche B e C.*

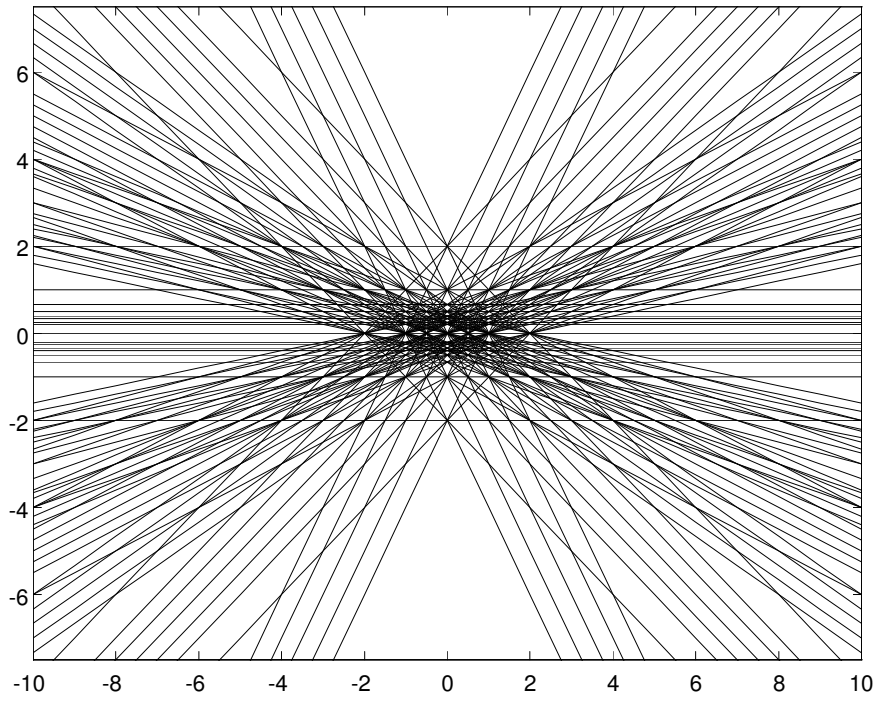

Figura 4.7: *Suddivisione del piano indotta dalla codifica B* ($h = 2$).

4.2.5.3 La codifica C

La suddivisione indotta dalla codifica B, che presentiamo in figura 4.7 per $h = 2$, mostra una evidente disuniformità tra le regioni presenti nel quadrato

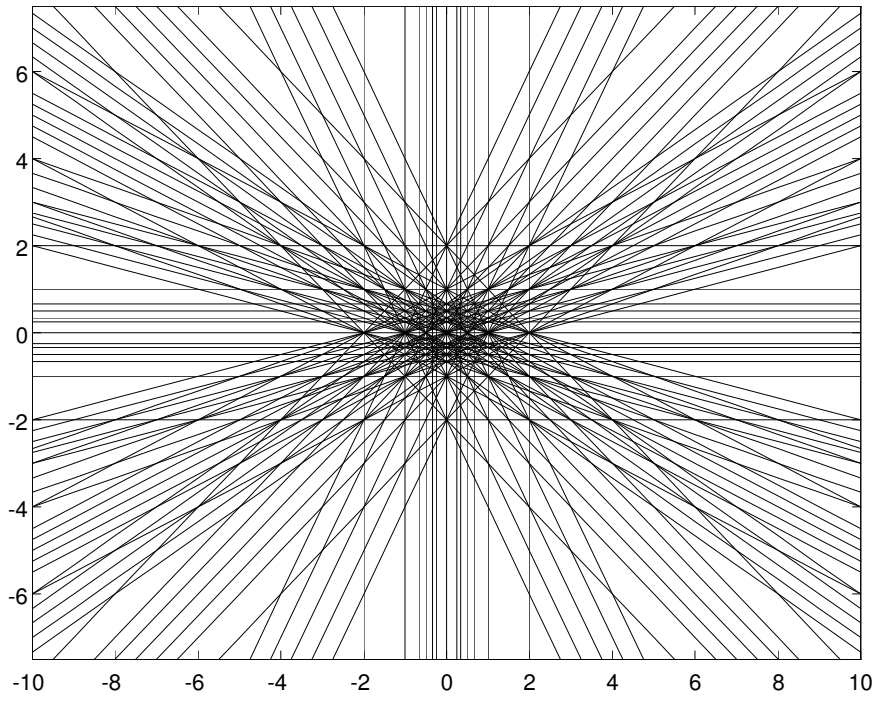

Figura 4.8: *Suddivisione del piano indotta dalla codifica C.*

degli ingressi. Questo ci ha suggerito l'opportunità di analizzare una ulteriore codifica (C) che, pur mantenendo elevato il numero di rette distinte generate, le distribuisca nel piano in modo da suddividere il quadrato degli ingressi in modo più uniforme.

La codifica C analizzata si differenzia dalla B per il fatto che i valori del peso associato al secondo ingresso vengono presi questa volta dall'insieme

$$V_h^C = \{0, 1, \ldots, 2h\}\,. \tag{4.71}$$

Questa scelta consente, in sostanza, di "recuperare" le rette parallele all'asse delle ordinate. La suddivisione indotta dalla codifica C (figura 4.8 per $h = 2$) pur generando un minor numero di regioni rispetto alla precedente, mostra un ulteriore decremento della probabilità $\widehat{P}$ a parità di valori discreti (tavola 4.5),

decremento da addebitarsi, per lo più, alla maggiore uniformità complessiva delle aree delle regioni.

L'andamento di $\widehat{P}$ rispetto ad N non si discosta dai precedenti: in questo caso l'interpolazione dei valori restituisce per α e c

$$\alpha \approx 0.710, \qquad c \approx 0.483. \tag{4.72}$$

4.2.5.4 La relazione tra rette e regioni

Se riportiamo in un grafico, per ciascuna delle classi di codifica analizzate finora, i dati relativi al numero delle regioni presenti nel quadrato degli ingressi in funzione delle rette che le generano (figura 4.9), scopriamo che le tre classi A, B, C, mostrano un andamento dei valori in tutto simile a quello della successione delle codifiche a griglia che, come si è visto, è circa $r^2/4$.

Con questo dato sperimentale, siamo ora in grado di esprimere, per ciascuna delle tre classi sopra menzionate, l'andamento della probabilità $\widehat{P}$:

1. rispetto al numero delle rette;
2. al variare del numero t dei bit.

La probabilità in funzione di r sarà, per le tre codifiche:

$$\widehat{P} \approx \frac{c\,4^{\alpha}}{r^{2\alpha}} \tag{4.73}$$

dove al posto di c ed α andranno sostituiti i valori propri di ciascuna codifica.

Infine, esprimiamo $\widehat{P}$ rispetto a t ed n per le codifiche A e B.
Abbiamo visto in precedenza che la codifica naturale genera un numero di

iperpiani

$$r \approx \frac{2^{mt}}{2\,\zeta(m)} = \frac{2^{(n+1)\,t}}{2\,\zeta(m)} \tag{4.74}$$

dove questa volta abbiamo espresso la cardinalità dell'insieme $(C_t^A) \subset I\!R^m$, immagine della codifica f_t^A, in funzione del numero dei bit.

Per il caso $n = 2$, sostituendo nella (4.73), avremo

$$\widehat{P}_A \approx \frac{c\,4^\alpha}{r^{2\alpha}} \approx \frac{c\,(4\,\zeta(3))^{2\alpha}}{(2^{6t})^\alpha} \tag{4.75}$$

In modo del tutto analogo, per la codifica B si trova

$$\widehat{P}_B \approx \frac{c\,4^\alpha}{r^{2\alpha}} \approx \frac{c\,(2\,\zeta(3))^{2\alpha}}{(2^{6t})^\alpha} \tag{4.76}$$

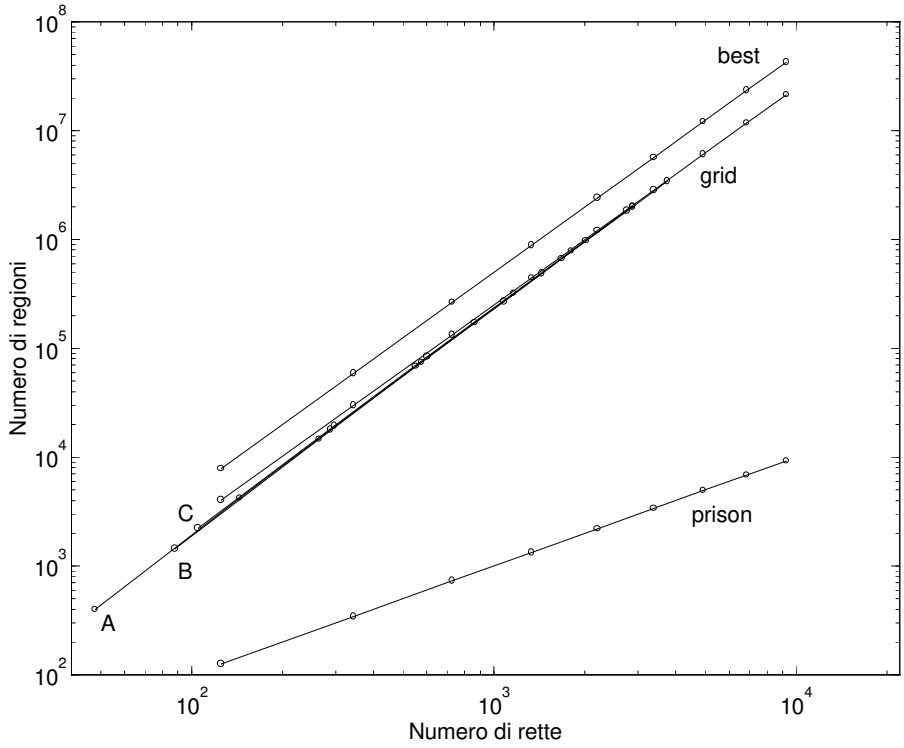

Figura 4.9: *Le regioni generate rispetto al numero di rette.*

4.2.6 Il Confronto tra i Risultati

Con l'ausilio di una tavola e di alcuni grafici, cercheremo di fare un quadro che riassuma e metta a confronto i risultati presentati fino a questo punto.

I grafici mettono in relazione l'andamento della probabilità $\widehat{P}$ con alcuni parametri di rilievo. In ognuno dei grafici i punti calcolati, per via sperimentale o in modo analitico, sono contrassegnati da cerchietti o asterischi, e congiunti da segmenti a tratto continuo.

Nel primo di questi (figura 4.10) confrontiamo i valori di $\widehat{P}$ esibiti dalle classi analizzate in funzione del numero di regioni all'interno del quadrato degli ingressi. I cerchietti indicano i punti calcolati al variare del numero d dei

Classi	*Andamento della probabilità* $\widehat{P}$					
	con r		*con N*		*con t*	
di codifica	In $\mathbb{R}^2$	In $\mathbb{R}^n$	In $\mathbb{R}^2$	In $\mathbb{R}^n$	In $\mathbb{R}^2$	In $\mathbb{R}^n$
worst	1					
best	$\approx \frac{2}{r^2}$		$\frac{1}{N}$		$\approx \frac{2}{2^{6t}}$	$\frac{1}{\sum_{i=0}^{n}\binom{2^{(n+1)t}}{i}}$
prison	$\approx \frac{1}{r}$		$\frac{1}{N}$		$\frac{1}{2^{3t}+1}$	$\frac{1}{2^{(n+1)t}+1}$
grid	$\approx \frac{4}{r^2}$		$\frac{1}{N}$		$\approx \frac{4}{2^{6t}}$	
PQ	$\approx \left(\frac{n}{r}\right)^n$		$\frac{1}{N}$		$\approx \left(\frac{n}{2^{(n+1)\,t}}\right)^n$	
A	$\approx \frac{c\,4^\alpha}{r^{2\alpha}}$		$\approx \frac{0.373}{N^{0.734}}$		$\approx \frac{c\,(4\,\zeta(3))^{2\alpha}}{(2^{6t})^\alpha}$	
B	$\approx \frac{c\,4^\alpha}{r^{2\alpha}}$		$\approx \frac{0.951}{N^{0.736}}$		$\approx \frac{c\,(2\,\zeta(3))^{2\alpha}}{(2^{6t})^\alpha}$	
C	$\approx \frac{c\,4^\alpha}{r^{2\alpha}}$		$\approx \frac{0.483}{N^{0.710}}$			

Tavola 4.6: *L'andamento di* $\widehat{P}$ *per tutte le classi analizzate.*

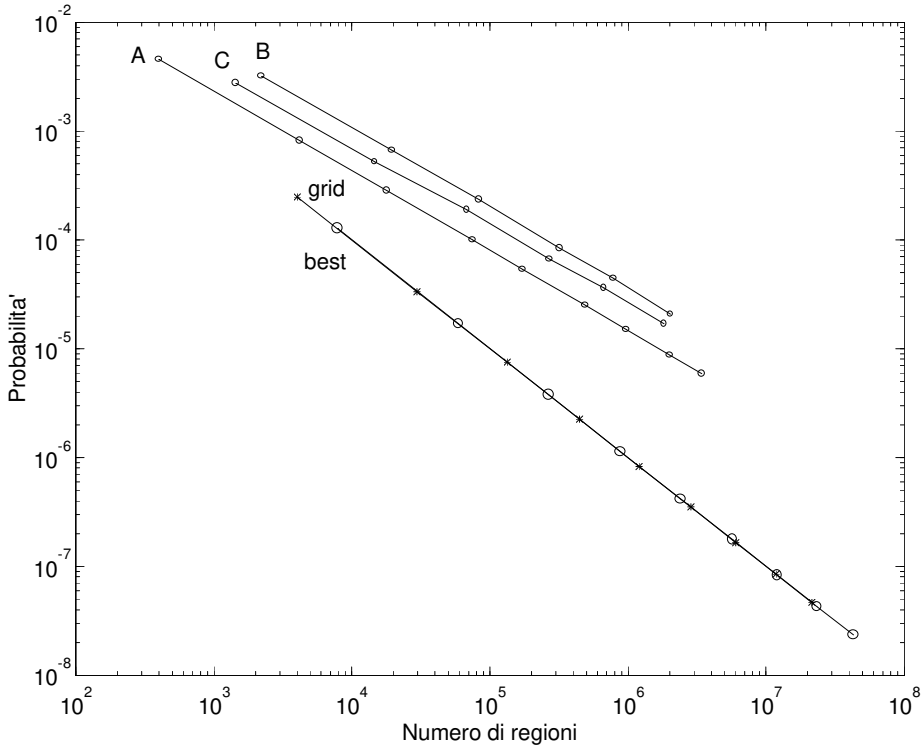

Figura 4.10: *La probabilità $\widehat{P}$ rispetto alle regioni nel quadrato degli ingressi.*

valori discreti per peso.

Il grafico è in scala logaritmica e mette in risalto il legame lineare esistente in tutti i casi tra il logaritmo di $\widehat{P}$ e quello del numero delle regioni. Legame che, se era stato imposto come condizione nei casi teorici (nel momento in cui è stata richiesta l'uguaglianza tra le aree delle regioni), si è trovato pure per le classi esaminate sperimentalmente.

La classe "best" mostra, come deve essere, l'andamento migliore ed i migliori valori per la probabilità a parità di valori discreti, e va quindi a collocarsi al di sotto delle altre classi.

I tre casi sperimentali mostrano un andamento della probabilità praticamente uguale, a dispetto del modo diverso di suddividere lo spazio degli ingressi.

Andamento che è, comunque, peggiore di quello esibito dalla classe a griglia, codifica "effettiva", e che pure si discosta soltanto per un fattore 2 dal miglior caso teorico.

Nella figura 4.11, dove i valori di $\widehat{P}$ sono riportati rispetto ai valori discreti per peso, si apprezzano meglio le distanze tra i valori di $\widehat{P}$ esibiti dalle varie codifiche. Per rendere la figura leggibile è stata mantenuta una scala semilogaritmica.

Infine, la figura 4.12, ancora in scala logaritmica, riporta i valori di $\widehat{P}$ in funzione dei bit per peso; come si vede, anche la relazione esistente tra logaritmo di $\widehat{P}$ e numero di bit è di natura lineare; la cosa, del resto, si sarebbe

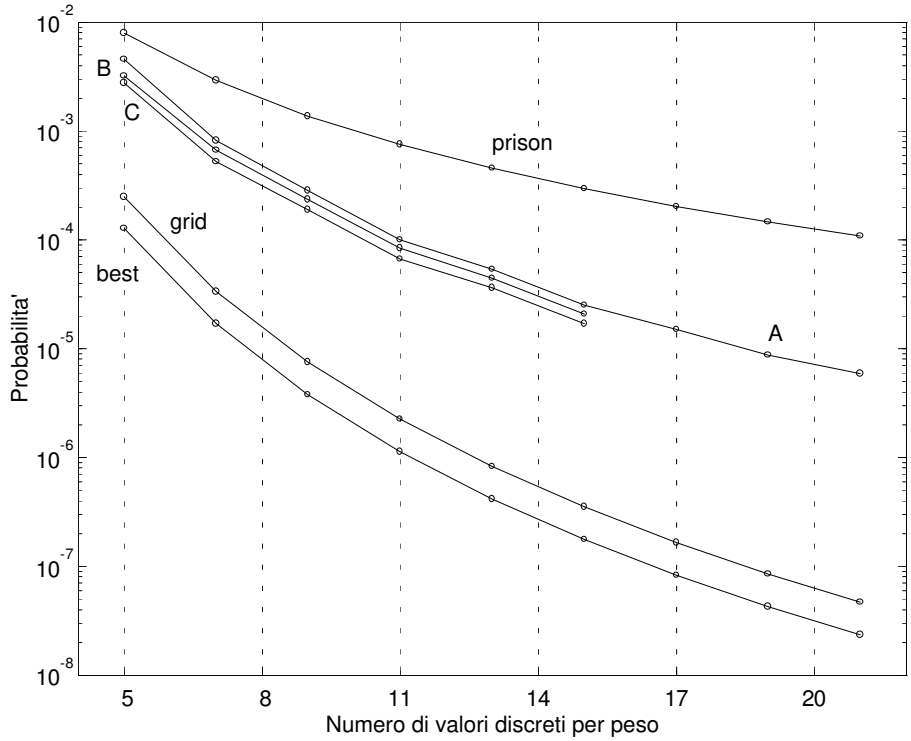

Figura 4.11: *La probabilità $\widehat{P}$ in funzione dei valori discreti per peso.*

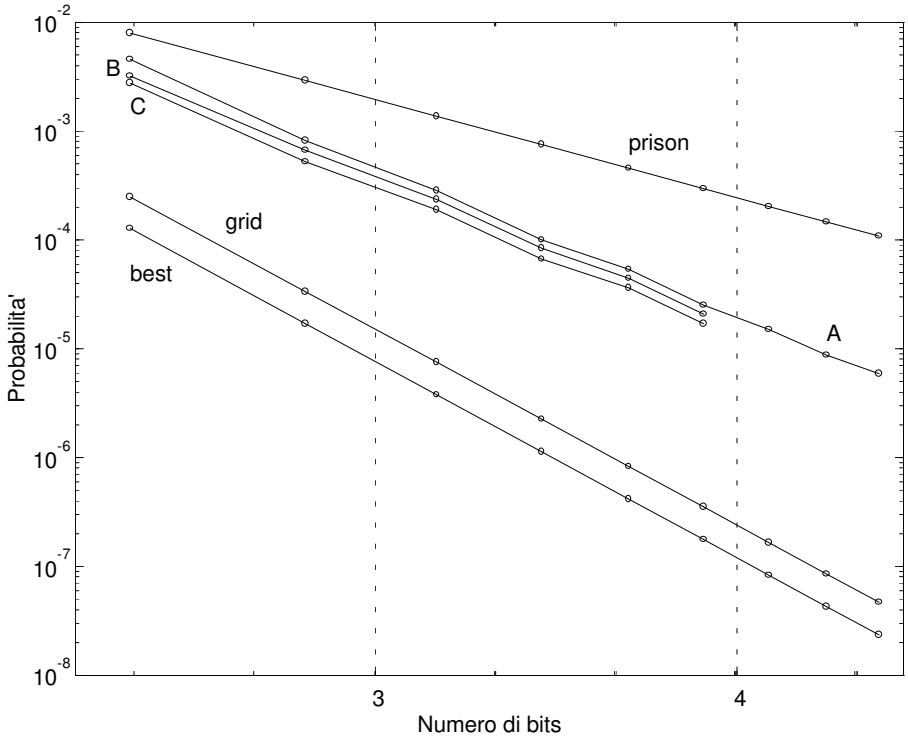

Figura 4.12: *La probabilità $\widehat{P}$ in funzione dei bit per peso.*

potuta facilmente ricavare per via analitica, ad esempio passando ai logaritmi nella (4.33):

$$\log_2 \widehat{P}\left(f_t^{grid}\right) \approx 2 - 6t. \tag{4.77}$$

In definitiva, **la classe di codifiche a griglia mostra**, tra le codifiche effettive prese in esame, **la migliore capacità di separazione**.

Viene naturale, allora, pensare ad una sua integrazione circuitale. Lasciamo a persone competenti il compito di indagare su quelli che potrebbero essere i problemi legati ad una eventuale implementazione.

A noi sarebbe piaciuto, a questo punto, verificare se, rispetto ad uno stesso problema, una rete di neuroni "a griglia" si comporti meglio di una rete di neuroni "naturali". Il tempo, pur sempre limitato, di un lavoro di tesi non ce

lo ha consentito.

Il nostro lavoro si ferma qui, lasciando aperte diverse questioni interessanti, che meriterebbero certamente di essere investigate. Sui possibili sviluppi ci proponiamo di offrire delle indicazioni in sede di conclusioni.

4.2.6.1 Un passo indietro

Vorremmo ritornare per un momento sulla questione della scelta del quadrato degli ingressi (sezione 4.2.5) che, per il modo in cui è stata effettuata, potrebbe aver lasciato qualche perplessità.
Abbiamo voluto verificare quanto questa scelta potesse aver inciso sui dati elaborati. Per far questo abbiamo valutato nuovamente il comportamento della classe di codifiche B di rango 3, ma questa volta su un diverso "quadrato" degli ingressi, e precisamente il rettangolo di base $2h$ ed altezza h centrato nell'origine degli assi. Dalla figura 4.7 si vede come questa scelta tenga maggiormente conto delle caratteristiche della codifica: viene eliminato infatti, rispetto al caso precedente, il contributo al valore di $\widehat{P}$ dovuto alle aree di maggiore taglia.

Abbiamo messo a confronto in figura 4.13 i risultati ottenuti in questo caso con quelli relativi alla codifica B valutata sul quadrato di lato $2h$. Il grafico mette in evidenza come l'andamento della probabilità esibito dalla successione sui due differenti rettangoli sia praticamente lo stesso, differendo soltanto per un fattore costante. È vero, d'altronde, che la classe B, se valutata sul rettangolo di area $2h^2$, mostra dei valori di probabilità più bassi di quelli della classe di codifiche C, che pure risultava migliore nel primo caso (figura 4.12).

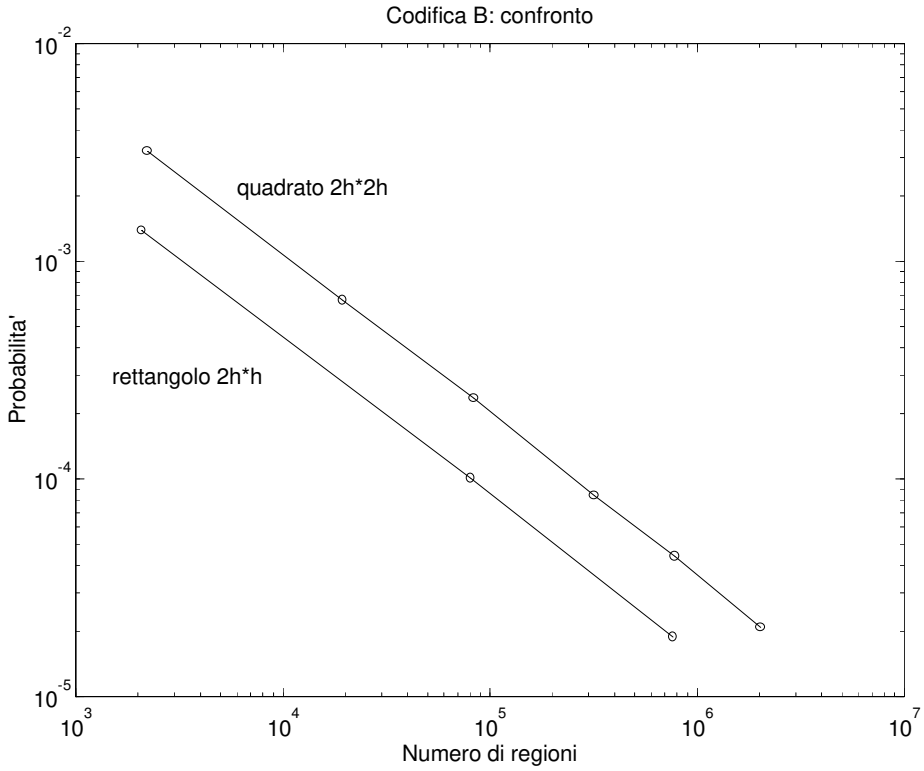

Figura 4.13: *Confronto dei valori di $\widehat{P}$ per la codifica B calcolati su due differenti distribuzioni degli ingressi.*

La scelta del quadrato, in definitiva, non ha pesato in maniera sostanziale sui dati sperimentali. D'altro canto le osservazioni precedenti suggeriscono l'opportunità di calibrare la scelta della codifica da utilizzare in relazione alla distribuzione degli ingressi o, in altri termini, al problema da trattare.

4.3 Neuroni Sigmoidali e Pesi Discreti

All'inizio del nostro lavoro ci eravamo proposti di indagare anche sulle conseguenze che la discretizzazione dei pesi ha sulle capacità dei neuroni aventi funzione di trasferimento sigmoidale, obiettivo andato disatteso per limiti di tempo.
Vorremmo ugualmente, tuttavia, fare alcune osservazioni che mettano in luce le relazioni esistenti tra i risultati di questo e del capitolo precedente.

Dal complesso dei risultati ottenuti, risulta chiaro che il fatto di limitare i parametri in un certo intervallo non pregiudica le capacità di separazione del neurone a soglia, cosa non più vera però se i pesi, oltre ad essere limitati, vengono discretizzati.

Nel neurone sigmoidale, agli effetti negativi legati alla limitazione dei parametri, vengono a sommarsi quelli dovuti alla discretizzazione, pregiudicandone ulteriormente il potere separatore. Quest'ultima affermazione diventa più comprensibile ripensando alla rappresentazione geometrica dei due neuroni nel piano.

Abbiamo più volte osservato come, mentre il neurone a soglia possa essere assimilato ad una retta, al neurone sigmoidale corrisponda una "fascia", la cui larghezza dipende dalla precisione richiesta al neurone nella risposta, oltreché dalle caratteristiche della sua funzione di uscita.

Un ritaglio di suddivisione del piano operata da un neurone sigmoidale a pesi limitati e discreti potrebbe presentarsi come in figura 4.14, dove le linee a tratto continuo indicano la corrispondente suddivisione, indotta però da un neurone a soglia. Dalla figura si comprende come due punti che si trovino nella

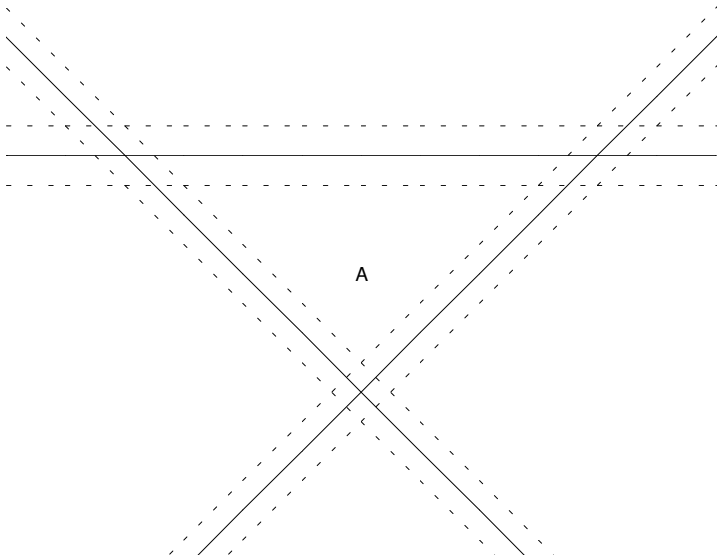

Figura 4.14: *La suddivisione del piano indotta da un neurone sigmoidale.*

regione A delimitata dal tratto continuo e che, quindi, non siano separati dal neurone a soglia, tantomeno possano esserlo dal neurone sigmoidale; inoltre, quand'anche i punti fossero separati dal neurone a soglia, potrebbero ancora non esserlo da parte del neurone sigmoidale: questo accadrebbe se almeno uno dei due punti cadesse all'interno della *fascia di indistinguibilità* relativa alla retta che li separa.

Conclusioni

Le implementazioni circuitali delle reti neurali debbono sopportare delle restrizioni ai valori dei parametri che le caratterizzano, restrizioni dovute ai limiti della tecnologia utilizzata.

In sostanza, i pesi integrati su chip non possono assumere tutti i valori reali, come il modello connessionista richiederebbe, ma solo valori in un intervallo prefissato; questo accade perché, i segnali in tensione e/o corrente che veicolano l'informazione nel circuito non possono avere, per ovvie ragioni, valori indefinitamente grandi. Inoltre, qualora i pesi vengano implementati utilizzando memorie binarie, potranno assumere soltanto un numero finito di valori, in funzione del numero di bit a disposizione per ciascun peso.

Obiettivo della nostra tesi era investigare sui riflessi che l'introduzione di questi vincoli ha sulle capacità di classificazione delle reti neurali feed-forward multistrato, costituite da neuroni a soglia oppure da neuroni aventi funzione di trasferimento assimilabile alla funzione sigmoide data dalla (1.7). Il modello scelto per i neuroni esaminati prevedeva che l'ingresso totale ad ogni unità fosse espresso dalla (3.1).

Siamo partiti analizzando quali siano i problemi derivanti dall'introduzione di un limite di ampiezza sui valori dei pesi.

Si è visto che le reti di neuroni a soglia, di topologia comunque complessa, non risentono di questa limitazione.
Al contrario, le capacità di separazione di un neurone sigmoidale subiscono un degrado, legato sia alla precisione della risposta richiesta al neurone che ai limiti di ampiezza imposti ai valori dei pesi sinaptici.

A questo proposito è stata stabilita (vedi il teorema 3.1) una condizione necessaria affinché due punti dello spazio n-dimensionale degli ingressi possano essere separati da un neurone sigmoidale, condizione che mette in relazione la distanza dei due punti con il massimo valore (in modulo) che può essere assunto dai pesi delle connessioni e con la qualità della risposta che al neurone si richiede.

Per quanto concerne le reti di neuroni sigmoidali, un primo risultato stabilisce un *lower bound* per il numero delle unità nascoste di una rete a pesi limitati che voglia separare due punti con un certo grado di accuratezza (teorema 3.2).

Inoltre, abbiamo fatto vedere come la conoscenza della funzione di trasferimento dei neuroni della rete consenta, in modo analogo a quanto è stato fatto per il singolo neurone, di mettere direttamente in relazione la distanza tra i punti da classificare con il *bound* sui pesi, il grado di accuratezza richiesto per la risposta ed il numero di unità presenti nell'unico strato nascosto della rete (proposizione 3.1).

La condizione necessaria per la separabilità stabilita, a titolo di esempio, per il caso di una rete di neuroni aventi funzione di uscita data dalla (3.40) (corollario 3.1), si rivela più forte di quella espressa nel teorema 3.2.

Siamo passati, di seguito, allo studio degli effetti della discretizzazione dei

valori dei pesi sulle capacità di classificazione di un neurone a gradino.

Abbiamo fatto notare come l'espressione 3.1, che descrive la regola di propagazione del neurone, rappresenti l'equazione di un iperpiano nello spazio n-dimensionale degli ingressi. Separare due punti equivarrà a scegliere i coefficienti dell'equazione dell'iperpiano in modo che esso si collochi tra i punti dati. Il fatto che i pesi possano assumere soltanto un numero finito di valori, consentirà all'iperpiano di poter essere collocato in un numero finito di posizioni, condizione questa non sufficiente a dividere due qualunque punti dello spazio. Le capacità di classificazione di un neurone a soglia a pesi discreti risultano, dunque, menomate.

Una misura di queste capacità è stata definita in termini probabilistici; in altre parole, abbiamo associato al neurone un valore di probabilità: la probabilità che questo separi punti scelti a caso in una fissata regione dello spazio degli ingressi.

È stato osservato come il numero e la posizione dei possibili iperpiani rappresentati da un neurone varino al variare della scelta dei valori discreti da far assumere ai pesi delle connessioni. Di conseguenza varierà la probabilità di separazione del neurone, strettamente legata al modo di collocare gli iperpiani nello spazio.

Per formalizzare queste osservazioni è stata introdotta la nozione di codifica, funzione associata al neurone, che definisce la posizione dei possibili iperpiani che lo stesso neurone può rappresentare al variare dei pesi discreti.

La funzione di codifica descrive in modo completo le caratteristiche del neurone cui viene associata tanto che, nel seguito, ci è stato possibile parlare

di probabilità di separazione di una codifica.

Ci siamo posti, a questo punto, il problema di individuare quale fosse la migliore codifica: la codifica, cioè, che rendesse massima la probabilità di separazione del neurone rispetto ad una fissata distribuzione degli ingressi, che nel nostro caso abbiamo considerato uniforme.

Abbiamo analizzato diversi tipi di neuroni o, se si vuole, di codifiche e per ciascuna di esse abbiamo calcolato, per via analitica o sperimentale, il valore di probabilità associato.

Abbiamo visto che la codifica che dispone gli iperpiani in modo da suddividere lo spazio degli ingressi in una "griglia" n-dimensionale con regioni di uguale misura, esibisce una probabilità di separazione molto vicina a quella del miglior caso teorico, e comunque nettamente migliore di quella mostrata dai neuroni più comunemente utilizzati nelle implementazioni hardware delle reti neurali. I pesi di questi neuroni assumono, di norma, valori interi compresi nell'intervallo $[-2^{t-1}+1, 2^{t-1}-1]$, dove t indica il numero di bit di cui ciascun peso dispone.

Fin qui il lavoro svolto.

La codifica "a griglia" meriterebbe ulteriori attenzioni, in primo luogo per verificarne il miglior comportamento rispetto ad altri tipi di neuroni a soglia; questo potrebbe essere fatto sottoponendo uno stesso problema di classificazione a due reti distinte, costituite una da neuroni "a griglia" e l'altra, ad es., dai neuroni sopra descritti. In secondo luogo sarebbe necessario studiare l'effettiva implementabilità circuitale della codifica.

Come si è visto, gli obiettivi iniziali del lavoro sono andati in parte disattesi. Avremmo voluto estendere i risultati ottenuti al caso dei neuroni sigmoidali.

Nella parte finale del lavoro abbiamo brevemente messo in evidenza come un neurone sigmoidale a pesi discreti veda le sue capacità ulteriormente pregiudicate, rispetto a quelle di un neurone a soglia. Uno studio sistematico del problema avrebbe, tuttavia, ben altra rilevanza, vista la centralità di questo tipo di neuroni nel campo dell'integrazione circuitale delle reti.

Il lavoro svolto, per quanto abbracci una tipologia di reti, quelle feed-forward multistrato, che gode di attenzioni privilegiate, lascia fuori diversi importanti modelli neurali. La stessa analisi meriterebbe di essere condotta anche su modelli ad apprendimento diretto, cioè non supervisionato, come le Reti di Hopfield o le Reti di Kohonen. I risvolti della limitazione dei pesi potrebbero essere valutati pure nelle topologie aventi connessioni intra-strato e nelle reti con *feedback*, in cui i segnali in uscita vengono retropropagati verso i neuroni d'ingresso.

Altro campo di indagine secondo noi interessante potrebbe essere quello relativo alla regola di propagazione.
Il valore di attivazione dato dalla (3.1) è, come si vede, una funzione lineare degli ingressi. Questa è, tuttavia, solo una possibile scelta per la regola di propagazione. Potremmo, in linea di principio, utilizzare regole di propagazione in cui gli ingressi al neurone siano legati in modo più complicato ai pesi delle connessioni.
Consideriamo, per esempio, la regola di propagazione così definita:

$$net(\overline{x}, \overline{w}) = \sum_{i=1}^{n}(x_i - w_i)^2 - \theta^2. \qquad (\ .1)$$

La (.1) rappresenta, se posta uguale a zero, l'equazione di una sfera n-

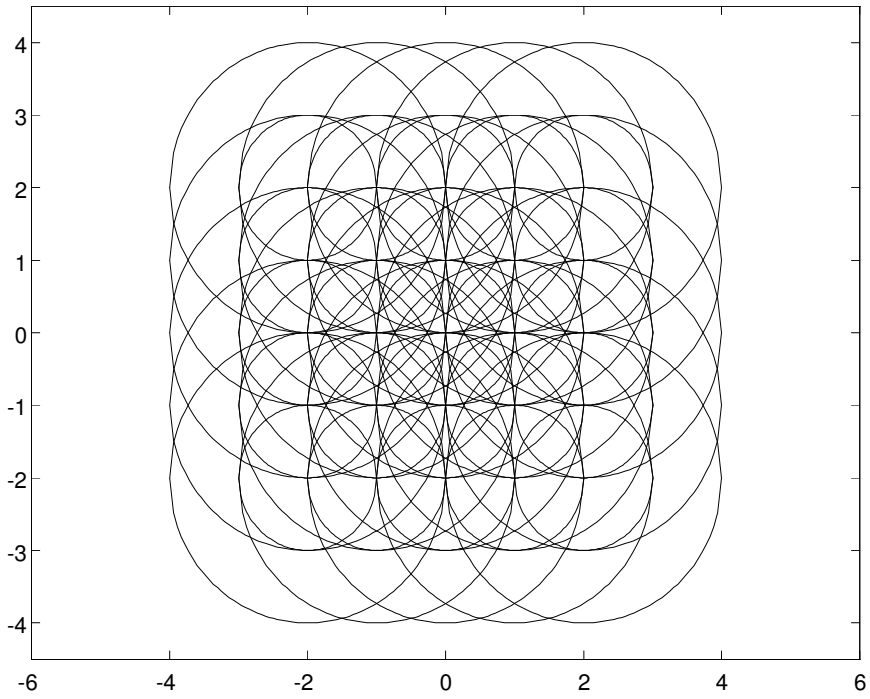

Figura .1: *La suddivisione del piano generata da un neurone a soglia con regola di propagazione data dalla (.1) e funzione di codifica naturale per h=2 (vedi sez. 4.2.5).*

dimensionale di centro $\overline{w}$ e raggio θ. In altri termini, assumendo la (.1), un neurone a soglia sarebbe rappresentato in $\mathbb{R}^n$ non più da un iperpiano, ma da una sfera. Una suddivisione del piano operata da tale neurone potrebbe presentarsi come quella in figura .1. Sarebbe interessante investigare se, a parità di numero di bit a disposizione per ciascun peso, neuroni a soglia a pesi discreti che utilizzino la (.1) siano in grado di esibire una capacità di separazione migliore di quella mostrata dai neuroni da noi analizzati.

Appendice A

Richiami di Matematica

A.1 Norma e distanza

Definizione A.1 *Sia X un insieme; si dice* **distanza** *o* **metrica** *una funzione $d(x, y)$, che ad ogni coppia di punti x ed y associa un numero reale, con le seguenti proprietà:*

a) $d(x, y) \geq 0$; $d(x, y) = 0$ *se e solo se* $x = y$.

b) $d(x, y) = d(y, x)$ *per ogni* $x, y \in X$.

c) $d(x, y) \leq d(x, z) + d(z, y)$ *per ogni* $x, y \in X$.

□

Definizione A.2 *Sia X un insieme e $d(x, y)$ una metrica in X; la coppia (X, d) si dice* **spazio metrico**. □

Definizione A.3 *Sia X uno spazio vettoriale; si dice* **norma** *una funzione $N : X \longrightarrow \mathbb{R}$, che verifica le seguenti proprietà:*

a) $N(x) \geq 0$; $N(x) = 0$ *se e solo se* $x = 0$.

b) $N(\lambda x) = |\lambda| \, N(x)$ *per ogni* $x \in X, \lambda \in \mathbb{R}$.

c) $N(x + y) \leq N(x) + N(y)$ *per ogni* $x, y \in X$.

Si scrive di solito $\|x\|$ in luogo di $N(x)$. □

Definizione A.4 *Si dice* **spazio normato** *uno spazio vettoriale dotato di norma.* □

Nel caso particolare dello spazio $\mathbb{R}^n$, si definisce come **norma p**, per $1 \leq p < \infty$, la quantità:

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} \tag{A.1}$$

Per $p = 2$ si ha la usuale **norma euclidea**, detta anche **norma 2**; per $p = 1$ la norma corrispondente è nota anche come **norma di Manhattan** o **norma 1**, ed è così definita:

$$\|x\|_1 = \sum_{i=1}^{n} |x_i| \tag{A.2}$$

Nel caso $p = \infty$ si ha la **norma del massimo**, definita da:

$$\|x\|_\infty = \max_{1 \leq i \leq n} |x_i| \tag{A.3}$$

A.1.1 Distanza indotta dalla norma

Uno spazio normato è anche uno spazio metrico, con la distanza $d : X^2 \longrightarrow \mathbb{R}$ indotta dalla norma e così definita:

$$d(x, y) := \|x - y\| \tag{A.4}$$

Si verifica facilmente che la distanza (A.4) soddisfa le proprietà elencate nella definizione.

La distanza indotta dalla norma 1, che indicheremo con la notazione $d_1(x, y)$, sarà allora così espressa:

$$d_1(x, y) = \|x - y\|_1 = \sum_{i=1}^{n} |x_i - y_i| \tag{A.5}$$

mentre dalla norma 2 discende la ben nota **distanza euclidea**, che indicheremo con $d_2(x, y)$ e che sarà espressa dalla (A.6):

$$d_2(x, y) = \|x - y\|_2 = \left(\sum_{i=1}^{n} (x_i - y_i)^2 \right)^{1/2} \tag{A.6}$$

A.2 Questioni di minimo

Richiamiamo alcuni risultati relativi alla teoria dei minimi di funzioni di più variabili reali.

Teorema A.1 (dei moltiplicatori di Lagrange) *Sia* $f : \mathbb{R}^n \longrightarrow \mathbb{R}$, $f \in C^1(\mathbb{R}^n)$; *sia* $G = \{ (p_1, \ldots, p_n) \in \mathbb{R}^n : u(p_1, \ldots, p_n) = 0 \}$ *dove* $u \in C^1(\mathbb{R}^n)$ *e* ∇u *(gradiente di* u*)* $\neq \overline{0}$ *in* G.
I punti di massimo e minimo vincolato di f *(con vincolo* G*) sono punti stazionari liberi per la funzione di* $n + 1$ *variabili*

$$H(p_1, \ldots, p_n, \lambda) = f(p_1, \ldots, p_n) + \lambda u(p_1, \ldots, p_n). \tag{A.7}$$

Teorema A.2 *Sia $f : \mathbb{R}^n \longrightarrow \mathbb{R}$, continua, e sia $K \subset \mathbb{R}^n, K$ chiuso. Sia $\lim_{\|x\|\to+\infty} f(x) = +\infty$; allora $\exists \min_K f$.*

Dim. Sia $z_o \in K$; scegliamo $M =\mid f(z_o) \mid + 1 > 0$; in corrispondenza ad M

$$\exists \rho > 0 : \parallel x \parallel \geq \rho \Longrightarrow f(x) > M =\mid f(z_o) \mid + 1 \geq f(z_o) + 1 > f(z_o). \quad \text{(A.8)}$$

Prendiamo $\hat{\rho} = \max\{\rho, \parallel z_o \parallel +1\}$. Consideriamo l'insieme $(K \cap \overline{B(0,\hat{\rho})})$, intersezione di K con la palla chiusa avente centro nell'origine e raggio $\hat{\rho}$ (vedi la figura A.1); nel seguito indicheremo l'insieme $\overline{B(0,\hat{\rho})}$ con la lettera B. Essendo K e B dei chiusi, sarà chiusa anche la loro intersezione; inoltre, poichè $(K \cap B) \subset B$, $(K \cap B)$ è anche limitato. Ma,

$$(K \cap B) \textit{ chiuso e limitato in } \mathbb{R}^n \Longrightarrow (K \cap B) \textit{ compatto.} \quad \text{(A.9)}$$

Essendo la f continua in un compatto, in virtù del teorema di Weierstrass, ammetterà minimo in $(K \cap B)$ (vedi [9], vol. 2, pag. 31). Perciò

$$\exists x_o \in (K \cap B) : f(x_o) \leq f(x),\ \forall x \in (K \cap B). \quad \text{(A.10)}$$

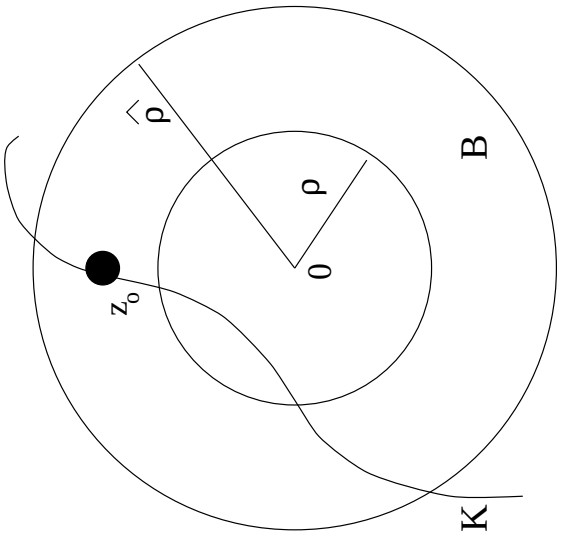

Figura A.1: *L'insieme B del teorema A.2*

Per come $\hat{\rho}$ è stato scelto, $z_o \in (K \cap B)$, e dunque

$$f(x_o) \leq f(z_o). \tag{A.11}$$

Sia $x \in K$; distinguiamo due casi:

a) $\| x \| > \hat{\rho}$; allora, in virtù delle (A.8) e (A.11), si ha $f(x) > f(z_o) \geq f(x_o)$.

b) $\| x \| \leq \hat{\rho}$; quest'ipotesi implica che $x \in B$; ma allora, per la (A.10), $f(x) \geq f(x_o)$.

Dunque $\forall x \in K,\ f(x) \geq f(x_o)$, e quindi $f(x_0) = \min_K f$. $\square$

A.3 Nozioni metriche nel piano

A.3.1 Significato geometrico dei coefficienti dell'equazione della retta

Sia $a\,x+b\,y+c = 0$ l'equazione di una retta r e sia P_0 un suo punto di coordinate x_0, y_0. Dovendo essere allora $c = -a\,x_0 - b\,y_0$, la retta potrà scriversi anche nel seguente modo:

$$a\,(x - x_0) + b\,(y - y_0) = 0. \tag{A.12}$$

Indichiamo con $\overline{v}$ un vettore di coordinate (v_1, v_2) parallelo alla retta n perpendicolare ad r. Condizione affinché un punto P di coordinate x, y appartenga alla retta r è che che il vettore $\overline{P_0P}$ risulti perpendicolare a $\overline{v}$, cioè che sia:

$$\overline{P_0P} \cdot \overline{v} = 0. \tag{A.13}$$

Esplicitando la (A.13) si ha:

$$v_1 (x - x_0) + v_2 (y - y_0) = 0. \tag{A.14}$$

Confrontando la (A.14) con la (A.12) si ha che deve essere:

$$a = \rho\, v_1 \qquad\qquad b = \rho\, v_2 \tag{A.15}$$

dove ρ è un fattore di proporzionalità diverso da zero, e quindi

Proposizione A.1 *I coefficienti a, b dell'equazione $a\,x + b\,y + c = 0$ di una retta sono proporzionali alle componenti di un vettore perpendicolare alla retta stessa.*

Dalle (A.15) è possibile ricavare il valore delle componenti del versore della normale alla retta r imponendo che queste soddisfino la relazione:

$$v_1^2 + v_2^2 = 1 \tag{A.16}$$

Avremo quindi:

$$\frac{a^2}{\rho^2} + \frac{b^2}{\rho^2} = 1 \tag{A.17}$$

da cui

$$\rho = \pm\sqrt{a^2 + b^2} \tag{A.18}$$

Le componenti del versore $\overline{n}$ della retta normale ad r saranno dunque, a meno del segno,

$$n_1 = \frac{a}{\pm\sqrt{a^2 + b^2}} \qquad\qquad n_2 = \frac{b}{\pm\sqrt{a^2 + b^2}} \tag{A.19}$$

dove i segni superiori oppure inferiori andranno scelti in base all'orientazione della normale.

A.3.2 Distanza di un punto da una retta

Siano $P_0\,(x_0, y_0)$ un punto del piano ed r una retta di equazione $a\,x+b\,y+c=0$; vogliamo determinare la distanza (euclidea) di P_0 da r. Sia Q il piede della perpendicolare condotta da P_0 ad r; la distanza cercata sarà data dalla misura del segmento P_0Q.

Sia P_1 un punto qualunque sulla retta r; la detta distanza sarà uguale alla componente ortogonale del vettore P_1P_0 secondo la retta P_0Q comunque orientata.
Siano $\overline{n}$ il versore della retta P_0Q ed x_1, y_1 le coordinate di P_1; per quanto detto potremo scrivere:

$$dist(P_0, r) = \left|\overline{P_1P_0} \cdot \overline{n}\right| \tag{A.20}$$

Le componenti di $\overline{n}$ sono indicate nella (A.19), mentre quelle del vettore $\overline{P_1P_0}$ sono $x_0 - x_1$, $y_0 - y_1$; perciò, esplicitando la (A.20), avremo

$$dist(P_0, r) = \frac{|a\,(x_0 - x_1) + b\,(y_0 - y_1)|}{\sqrt{a^2 + b^2}}. \tag{A.21}$$

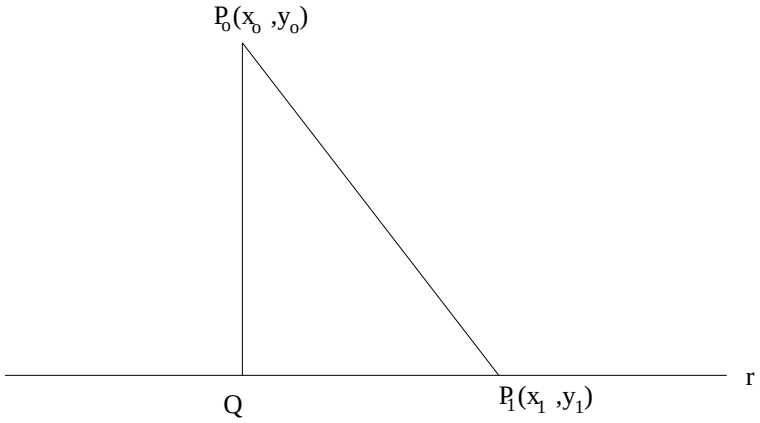

Figura A.2: *Distanza euclidea tra un punto ed una retta.*

Poiché P_1 appartiene ad r, deve essere

$$c = -a\,x_1 - b\,y_1$$

e quindi la (A.21) diventa

$$dist(P_0, r) = \frac{|a\,x_0 + b\,y_0 + c|}{\sqrt{a^2 + b^2}}. \tag{A.22}$$

Appendice B

Dimostrazioni

B.1 La Formula Ricorsiva

La formula (4.24) a pag. 92 si giustifica sulla base di alcune osservazioni. Consideriamo una suddivisione d dello spazio $\mathbb{R}^n$ in regioni, ottenuta con r iperpiani $(n-1)$-dimensionali. Il numero di regioni che vengono ad aggiungersi alla suddivisione collocando l'$(r+1)$-esimo iperpiano dipende:

1. dal modo in cui verrà collocato l'iperpiano;
2. dalla posizione degli r iperpiani già sistemati.

Indichiamo con d_{max} la suddivisione di $\mathbb{R}^n$ costruita collocando ogni iperpiano in modo che le regioni di volta in volta aggiunte siano sempre in numero massimo. Sia $R_n^{max}(r)$ il numero delle regioni ottenibili con r iperpiani seguendo la strategia sopra descritta. Sia, inoltre, $A_n^{max}(r)$ il numero di regioni aggiunte

alla suddivisione d_{max} con l'r-esimo iperpiano. Potremo scrivere

$$R_n^{max}(r) = R_n^{max}(r-1) + A_n^{max}(r) \tag{B.1}$$

Si tratta ora di caratterizzare in modo preciso il numero $A_n^{max}(r)$. Come vedremo, tale numero è dato da $R_{n-1}^{max}(r-1)$, coincide cioè con il numero massimo di regioni di $n-1$ dimensioni che $r-1$ iperpiani, intersecandosi con l'iperpiano r, sono in grado di *staccare* su di esso.

Sia $R_n^d(r)$ il numero di regioni presenti in una suddivisione d, ottenute con r iperpiani. La differenza $R_n^d(r+1) - R_n^d(r)$ risulterà massima

> *se aggiungeremo l'*$(r+1)$*-esimo iperpiano in maniera che non sia parallelo ad alcun iperpiano appartenente ad uno qualunque dei* fasci (di iperpiani) propri o impropri *individuati da ogni coppia di iperpiani della suddivisione.*

Questa condizione può essere espressa in termini analitici: siano

$$\sum_{i=1}^{n} a_i x_i + \theta_a = 0, \qquad \sum_{i=1}^{n} b_i x_i + \theta_b = 0 \tag{B.2}$$

due iperpiani a, b della suddivisione d, e sia

$$\sum_{i=1}^{n} c_i x_i + \theta_c = 0 \tag{B.3}$$

l'iperpiano c da aggiungere. Consideriamo le matrici

$$A = \begin{pmatrix} a_1 & a_2 & \cdots & a_n \\ b_1 & b_2 & \cdots & b_n \end{pmatrix} \tag{B.4}$$

e

$$\overline{A} = \begin{pmatrix} a_1 & a_2 & \cdots & a_n \\ b_1 & b_2 & \cdots & b_n \\ c_1 & c_2 & \cdots & c_n \end{pmatrix} \tag{B.5}$$

La condizione sopra indicata è equivalente alla seguente relazione:

$$Rank(\overline{A}) > Rank(A). \tag{B.6}$$

Infatti, nel caso in cui la coppia a, b di iperpiani presa in esame sia costituita da piani paralleli, dovrà essere

$$Rank(\overline{A}) > Rank(A) = 1. \tag{B.7}$$

Se, invece, i due piani si intersecano,

$$Rank(\overline{A}) > Rank(A) = 2. \tag{B.8}$$

Dalle considerazioni precedenti si deduce pure che, per avere la garanzia che la suddivisione contenga sempre il massimo numero di regioni per un numero di iperpiani fissato, dovranno valere le seguenti condizioni:

a) *tutti gli iperpiani dovranno intersecarsi tra loro;*

b) *tra gli iperpiani* $(n-2)$*-dimensionali che descrivono i centri dei fasci individuati da una qualunque coppia di iperpiani della suddivisione non dovrà esistere alcuna coppia di elementi paralleli.*

Pensiamo non sia facile comprendere, almeno in prima approssimazione, quale sia il significato preciso delle condizioni precedenti. Riteniamo sia molto utile costruirsi dei piccoli esempi in $\mathbb{R}^2$ ed $\mathbb{R}^3$.

Osserviamo ancora (anche in questo caso sarà opportuno aiutarsi con degli esempi) come, il numero $A_n^d(r)$ di regioni che si aggiungono al passo r di una suddivisione,

corrisponda a quello delle regioni individuate sull'r-esimo iperpiano dalle intersezioni di quest'ultimo con gli altri iperpiani della suddivisione.

Siamo in grado, a questo punto, di dire chi sia $A_n^{max}(r)$.
Nel caso in cui valgano le condizioni sui fasci della suddivisione espresse in italico, avremo che

$$A_n^{max}(r) = R_{n-1}^{max}(r-1) \tag{B.9}$$

Infatti, le condizioni varranno anche per tutti gli $r-1$ fasci giacenti sull'r-esimo iperpiano.
Se consideriamo i fasci, a loro volta, come iperpiani $(n-2)$-dimensionali nello spazio ad $n-1$ dimensioni costituito dall'iperpiano r, le regioni in cui questo iperpiano risulta diviso sono proprio $R_{n-1}^{max}(r-1)$.
Sostituendo la (B.9) nella (B.1), ritroviamo proprio la (4.24).

B.2 Sulla cardinalità

Teorema B.1 *Siano* $V_h = \{-h, -h+1, \ldots, 0, \ldots, h\}, \quad h \in N,$
e $C_h = \overbrace{V_h \times V_h \times \ldots \times V_h}^{m \text{ volte}}$.
Sia $C_h/\sim$ *l'insieme quoziente di* C_h *modulo* $\sim$, *dove* $\sim$ *è la relazione di equivalenza definita nella sezione 3.2.1.*
Definiamo l'insieme $\overline{C_h/\sim} = C_h/\sim - [0]_\sim$. *Sia ancora*

$$\overline{C_h} = \{(a_1, \ldots, a_m) \in C_h : M.C.D.(a_1, \ldots, a_m) = 1\}. \tag{B.10}$$

Allora

$$\#(\overline{C_h}) = 2\,\#(\overline{C_h/\sim}). \tag{B.11}$$

Dim. Per costruzione, C_h è un insieme finito di cardinalità $(2h+1)^m$.

Allora, saranno finiti anche gli insiemi $C_h/\sim$ (l'applicazione che costruisce l'insieme è suriettiva) e $\overline{C_h}$ (si ha $\overline{C_h} \subseteq C_h$).

Per dimostrare la nostra tesi basterà far vedere che in ogni classe di equivalenza di $\overline{C_h/\sim}$ esistono due e due soli elementi di $\overline{C_h}$.

Facciamo vedere che in ogni elemento di $\overline{C_h/\sim}$ esistono almeno due elementi di $\overline{C_h}$.

Sia $[x]_\sim$ una classe di equivalenza di $\overline{C_h/\sim}$.

Supponiamo che

$$\forall\ y \in [x]_\sim,\ M.C.D.(y_1, \ldots, y_m) \neq 1. \tag{B.12}$$

Allora

$$\forall\ y \in [x]_\sim,\ y = \lambda u. \tag{B.13}$$

dove $M.C.D.(u_1, \ldots, u_m) = 1$ e $2 \leq \lambda \leq h,\ \lambda = M.C.D.(y_1, \ldots, y_m) \in Z$.

Facciamo vedere che $2 \leq \lambda \leq h$:

sappiamo per ipotesi che $y \neq 0,\ \ y_i \in V_h,\ i = 1, \ldots, m$. Dalla (B.13) abbiamo $y_i = \lambda u_i,\ i = 1, \ldots, m$. Consideriamo un'indice j per cui si abbia $y_j \neq 0$. Se fosse $\lambda > h$ avremmo intanto $u_j \neq 0$.

Supponiamo che si abbia $u_j > 0$: si avrebbe $u_j \geq 1$, da cui $y_j > h$, contro l'ipotesi.

Nel caso in cui sia $u_j < 0$ si avrebbe $u_j \leq -1$, da cui $y_j < -h$, contro l'ipotesi.

Per ipotesi, inoltre, λ non potrà essere uguale a zero oppure ad uno.

Facciamo vedere ora che $u_i \in V_h,\ i = 1, \ldots, m$:

consideriamo gli indici per cui $y_i > 0$; si avrebbe

$$0 < u_i = \frac{y_i}{h} \leq \frac{y_i}{2} \leq y_i \leq h. \tag{B.14}$$

Nel caso in cui $y_i < 0$ avremmo

$$0 > u_i = \frac{y_i}{h} \geq \frac{y_i}{2} \geq y_i \geq -h. \tag{B.15}$$

Infine, se $y_i = 0$ si avrà

$$u_i = 0. \tag{B.16}$$

Ma, $\forall\ i,\ u_i \in V_h \Longrightarrow u \in C_h$ e, poiché $M.C.D.(u_1, \ldots, u_m) = 1$ per costruzione, $u \in \overline{C_h}$.

Poiché dall'ipotesi, $u \in [x]_\sim$, abbiamo individuato un primo elemento di $\overline{C_h}$ che appartiene a $u \in [x]_\sim$.

Il secondo elemento in questione è proprio $-u$: infatti $u \in C_h \Longrightarrow -u \in C_h$ e $M.C.D.(-u_1, \ldots, -u_m) = 1$; inoltre $-u \in [u]_\sim = [x]_\sim$.

Abbiamo così dimostrato la prima parte del teorema.

Facciamo vedere adesso che in ogni classe di equivalenza esistono al più due elementi di $\overline{C_h}$.

Siano $x, y \in \overline{C_h}$, con $x \sim y$; Allora $y = \lambda x,\ \ \lambda \in I\!R,\ \lambda \neq 0$.

λ non potrà appartenere ad $I\!R - Q$, altrimenti sarebbe possibile esprimere un numero irrazionale come rapporto tra elementi di Z, quali sono le componenti di x ed y.

λ non potrà neanche essere un razionale $p/q,\ p, q \in Z,\ |p| \neq |q|$; infatti, in questo caso potremmo scrivere

$$q\, y = p\, x. \tag{B.17}$$

e ci troveremmo di fronte a due vettori uguali con diverso massimo comun divisore, il che è impossibile.

λ dunque potrà essere soltanto un razionale p/q con $|p| = |q|$, da cui $|p|/|q| = 1$ e $\lambda = \pm 1$. Quindi $y = x$ oppure $y = -x$.

Siano adesso tre **distinti** elementi $a, b, c \in \overline{C_h}$ nella stessa classe di equivalenza; allora, per il ragionamento precedente $b = -a$ e $c = -b$; quindi $c = -(-a) = a$: assurdo. $\square$

Appendice C

Il Programma

La scrittura del programma C per il calcolo della probabilità associata alle classi di codifica che sono state analizzate sperimentalmente e di cui abbiamo riferito nel cap. 4, si è rivelata più complessa di quanto ci si potesse attendere; questo è accaduto non tanto in ordine all'individuazione delle strategie da utilizzare per il calcolo della probabilità, quanto per i problemi legati alla loro ottimizzazione.

Il programma è stato scritto per risolvere il problema in due dimensioni, cioè per il caso di un neurone a soglia avente due ingressi.
Alle varie fasi del calcolo, non tutte ugualmente complesse, abbiamo fatto corrispondere diversi moduli eseguibili, scritti in maniera che i dati in uscita di ciascun modulo fossero immediatamente utilizzabili dal modulo successivo. Con l'ausilio di una procedura *batch* abbiamo automatizzato l'esecuzione in cascata dei vari moduli. Al tempo stesso, la suddivisione in moduli ci ha consentito di gestire in modo efficiente i tempi di elaborazione. La produzione delle tabelle di cui al cap. 4 ha richiesto, in ultima analisi, alcuni mesi di lavoro.

Per ogni classe di codifica analizzata (si faccia riferimento alla sezione 4.2.5), fissato il numero dei valori discreti a disposizione per ogni peso, si è trattato, in sostanza, di:

1. individuare il numero di rette distinte generate dalla codifica;
2. calcolare tutti i punti di intersezione tra le rette all'interno del quadrato degli ingressi;
3. individuare i vertici di ognuna delle regioni formate, nel quadrato degli ingressi, dalla suddivisione del piano indotta dalle rette;
4. calcolare l'area di ciascuna regione;
5. calcolare la probabilità $\widehat{P}$ data dalla (4.21).

Il punto che ha richiesto i maggiori sforzi per la scrittura di un algoritmo efficiente e che, nonostante tutto, ha mostrato i tempi di elaborazione più lunghi, è stato senza dubbio quello relativo all'individuazione dei vertici di ciascuna regione del quadrato.
Vorremmo brevemente descrivere i punti salienti dell'algoritmo che risolve questo punto; la procedura potrà risultare più comprensibile facendo riferimento alla figura C.1.

Il modulo riceve dai moduli precedenti due files: il primo con i valori dei coefficienti di tutte le rette che attraversano il quadrato; il secondo contenente le coordinate dei punti di intersezione tra le varie rette.

Di seguito, la procedura prende in esame in modo ordinato ognuna delle rette in ingresso. Scelta la prima, dall'unico insieme dei punti di intersezione

delle rette nel quadrato, costruisce due insiemi di punti: quelli che si trovano alla sinistra e quelli che si trovano alla destra della retta stessa. I punti di intersezione che appartengono alla retta vengono collocati in entrambe gli insiemi.

La seconda retta in esame verrà utilizzata per suddividere ciascuno degli insiemi di punti generati in precedenza (in questo caso due) in altri due insiemi: quello dei punti alla sinistra e quello dei punti alla destra della retta in esame. Per i punti giacenti sulla retta vale quanto detto per la prima retta. La terza retta dividerà gli insiemi formatisi fino a questo punto, e così via.

Quando tutte le rette saranno state considerate, in ognuno degli insiemi di punti così generati si troveranno tutti e soli i punti di vertice di una delle regioni del quadrato.

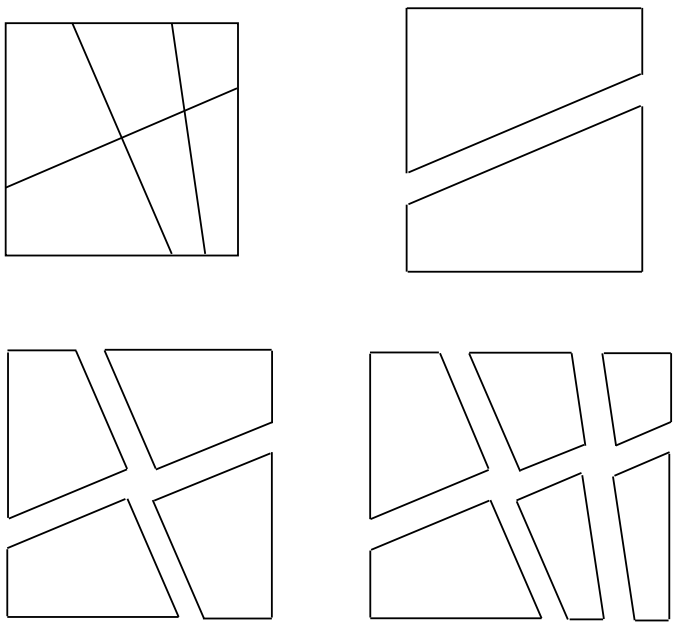

Figura C.1: *Le varie fasi della procedura che individua i vertici delle regioni all'interno del quadrato degli ingressi.*

La struttura convessa delle regioni che si formano all'interno del quadrato degli ingressi consente di ordinare i vertici di ognuna di esse (in senso orario oppure antiorario) in base alle loro coordinate. Il calcolo dell'area di ciascuna regione risulta semplice utilizzando la seguente formula:

$$2A = \left| \sum_{i=1}^{n} (x_i y_{i+1} - x_{i+1} y_i) \right|, \tag{C.1}$$

dove n sono i vertici della regione ed $(x_i,\, y_i),\ i = 1, \ldots, n$, le coordinate di ciascun vertice. La formula si ricava facilmente scomponendo in triangoli il poligono convesso che ogni regione costituisce.

L'algoritmo descritto per l'individuazione dei vertici presenta una "buona" complessità. Abbiamo visto che ciascuna retta viene utilizzata per suddividere ogni insieme di punti formatosi fino a quel momento. Se R sono le rette della suddivisione, rispetto ad R l'algoritmo ha una complessità

$$C(R) \propto \sum_{r=1}^{R} r^2 \propto R^3. \tag{C.2}$$

Rispetto alle regioni N la complessità dell'algoritmo sarà

$$C(N) \propto N^{3/2}, \tag{C.3}$$

tenuto conto della relazione $N \propto R^2$, esistente tra rette e regioni.

Un approccio alternativo per il calcolo delle aree potrebbe essere l'utilizzo di un metodo di tipo Montecarlo.
Il metodo, in effetti, si presenta attraente per la sua semplicità e soprattutto per la sua generalità, che permetterebbe di affrontare il problema in esame anche per dimensioni maggiori di 2. Tuttavia, le fluttuazioni intrinseche nel metodo stesso fanno sì che un'accuratezza ragionevole nei risultati non possa essere ottenuta se non a prezzo di una complessità computazionale elevata, specie se paragonata a quella esibita dall'algoritmo descritto in precedenza.

Un'applicazione *naive* del metodo al nostro problema consisterebbe nello scegliere a caso nel quadrato degli ingressi un numero di punti congruo alla precisione con cui si desidera calcolare il valore delle aree e, per ognuno dei punti, individuare l'area di appartenenza in base alla sua posizione rispetto a ciascuna delle rette della suddivisione.
L'area di ogni regione dovrebbe risultare proporzionale al numero di punti caduti nella regione stessa.

Ad esempio, se si volesse conoscere il valore delle aree con un margine d'errore che oscilli intorno al 10%, il numero di punti da estrarre dovrebbe essere circa 10^2 volte quello delle regioni presenti nel quadrato. Se si fa riferimento alla tavola 4.3 si vede che, per $h = 10$, si sarebbero dovuti estrarre circa 10^8 punti. Per stabilire l'area di appartenenza di ogni punto sarebbero necessarie circa 10^4 operazioni in virgola mobile, per un totale di circa 10^{12} operazioni. Con un processore 486, in grado di elaborare $\approx 10^6$ operazioni in virgola mobile al secondo, la procedura di calcolo avrebbe impiegato circa 15 giorni, contro le circa 30 ore impiegate, sulla stessa macchina, dall'algoritmo effettivamente utilizzato.

Vanno fatte, di contro, due importanti osservazioni:

a) tenendo conto delle caratteristiche particolari del problema in esame, sarebbe possibile apportare alla strategia descritta poc'anzi notevoli migliorie, che meriterebbero di essere studiate attentamente;

b) il metodo Montecarlo mostra la sua reale efficacia quando le dimensioni del problema aumentano: la caratteristica saliente del metodo consiste, infatti, nel poter esibire una **complessità indipendente dalle dimensioni del problema affrontato** [23, 24].

Bibliografia

[1] Arai M. Bounds on the number of hidden units in binary-valued three-layer neural networks. *Neural Networks*, 6:855–860, 1993.

[2] Cammarata S. *Reti Neuronali – Una introduzione all'altra intelligenza artificiale*. Collana di Informatica diretta da M. Ricciardi. ETASLIBRI, Milano, 1990.

[3] Cardarilli G. C., D'Alessandro C., Marinucci P. & Bordoni F. VLSI implementations of a modular and programmable neural architecture. In *Proceedings of Microneuro*, Torino, 26-29 Sep. 1994.

[4] Del Corso D. Hardware implementations of artificial neural networks. In *Lect. in Computer Science*, numero 686, pagine 405–417. Springer Verlag, 1993.

[5] Di Stefano G. *Reti Neurali per Diagnosi Automatiche di Campi Visivi*. Tesi di dottorato, Università degli Studi *La Sapienza*, Roma, 1992.

[6] Di Stefano G., Fabrizi G. & Sponta L. Consequences of limitation on weights in feed-forward multilayer neural networks. In Dagli C. H. & al., editors, *ANNIE '95 - Artificial Neural Networks In Engineering*, volume 5, pagine 21–26, St. Louis, MO, 1995. ASME Press.

[7] Fiesler E., Choudry A. & Caulfield H. A weight discretization paradigm for optical neural networks. In *Proceedings of the International Congress on Optical Science and Engineering, Bellingham*, volume SPIE–1281, pagine 164–173, Washington D. C., 1990.

[8] Frye R. C., Rietman E. A. & Wong C. C. Back-propagation learning and nonidealities in analog neural network hardware. *IEEE Trans. on Neural Networks*, 2(1):110–117, 1991.

[9] Giusti E. *Analisi Matematica*. 2 Voll. Boringhieri, Torino, 1983.

[10] Golea M. & Marchand M. On learning perceptrons with binary weights. *Neural Computation*, 5:767–782, 1993.

[11] Hegt J. A. Hardware implementations of neural networks. Rapporto tecnico, Eindhoven University of Tecnology, 1992.

[12] Hollis P., J. H. & Paulos J. The effects of precision constraints in a back-propagation learning network. *Neural Computation*, 2(3):363–373, 1990.

[13] Hornik K. Some new results on neural network approximation. *Neural Networks*, 6:1069–1072, 1993.

[14] Hornik K., Stinchombe M. & White H. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[15] Hornik K., Stinchombe M. & White H. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3:551–560, 1990.

[16] Jabri M. A. Practical performance and credit assignment efficiency of analog multilayer perceptron perturbation based training algorithms. In *Proceedings of Microneuro*, Torino, 26-29 Sep. 1994.

[17] Jutten C., Guerin A. & Herault J. Simulation machine and integrated implementation of neural networks. In *Lect. in Computer Science*, numero 412, pagine 244–266. Springer Verlag, 1990.

[18] Leonetti F. Esercizi di analisi superiore. Dip. di Matematica Pura ed Applicata, Università de L'Aquila, 1992.

[19] Leshno M., Lin V. Y., Pinkus A. & Schocken S. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6:861–867, 1993.

[20] Meir R. & Fontanari J. On the precision constraints of threshold elements. *Network*, 4:381–391, 1993.

[21] Minsky M. L. & Papert S. *Perceptrons: an Introduction to Computational Geometry*. The MIT Press, Cambridge, Massachusetts, 1969.

[22] Mitchell J. A geometric interpretation of hidden layer units in feedforward neural networks. *Network*, 3:19–25, 1992.

[23] Morokoff W. J. & Caflish R. E. Quasi-montecarlo integration. *J. Comp. Phys.*, 1994.

[24] Morokoff W. J. & Caflish R. E. Quasi-random sequences and their discrepancies. *SIAM J. Sci. Comp.*, 15, Novembre 1994.

[25] Patarnello S. *Le Reti Neuronali – Semplificare la complessità con l'aiuto dell'informatica*. FrancoAngeli, Milano, seconda ed., 1992.

[26] Pezzella F. *Elementi di Programmazione Lineare*, volume 1. Liguori Editore, Napoli, 1981.

[27] Pimmel R. L. & Kirk W. J. Effects of range and resolution on the performance of neural network classifiers with discrete weights. In Dagli C. H. & al., editors, *ANNIE '92 - Artificial Neural Networks In Engineering*, volume 2, pagine 299–304, St. Louis, MO, 1992. ASME Press.

[28] Rosenblatt F. The perceptron, a probabilistic model for information storage and organization in the brain. *Psycological Review*, 65:386–408, 1958.

[29] Rosenblatt F. Two theorems of statistical separability in the perceptron. In *Mechanisation of thought processes: Proc. of a symposium held at the National Physical Laboratory*, volume 1, pagine 421–456, London, 1959. HM Stationery Office.

[30] Rumelhart D. E., Hinton G. E. & Williams R. J. *Learning Internal Representations by Error Propagation*, capitolo 8, pagine 318–362. Volume 1: *Foundations* of Rumelhart and McClelland [31], 1986.

[31] Rumelhart D. E. & McClelland J. L., editors. *Parallel Distributed Processing*, volume 1: *Foundations*. The MIT Press, Cambridge, Massachusetts, 1986.

[32] Schroeder M. R. *La Teoria dei Numeri*. Manuali Scientifici. Franco Muzzio editore, Padova, 1986.

[33] Stinchombe M. & White H. Universal approximation using feedforward networks with non-sigmoid hidden layer activation functions. In *Proceedings of the International Joint Conference on Neural Networks,*

Washington D. C., volume 1, pagine 613–617, New York, 1989. IEEE Press.

[34] Stinchombe M. & White H. Approximating and learning unknown mappings using multilayer feedforward networks with bounded weights. In *Proceedings of the International Joint Conference on Neural Networks, Washington D. C.*, volume 3, pagine 7–16, New York, 1990. IEEE Press.

[35] Takahashi H., Tomita E. & Kawabata T. Separability of internal representations in multilayer perceptrons with application to learning. *Neural Networks*, 6:689–783, 1993.

[36] Woodland P. Weight limiting, weight quantisation and generalisation in multilayer perceptrons. In *Proceedings of the International Conference on Artificial Neural Networks*, pagine 297–300, Piscataway, N. Y., 1989. IEEE Press.