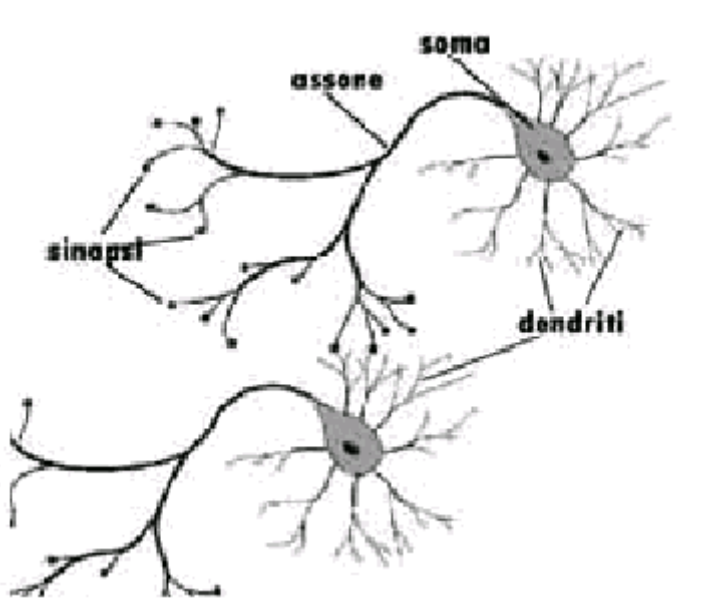


CAPITOLO 7

MACCHINE CHE APPRENDONO E SI EVOLVONO

1. Un neurone per fare calcoli.

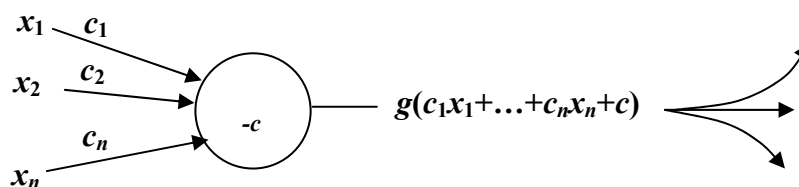
Consideriamo ora un modo di realizzare “macchine”, per meglio dire automi finiti, che è legato all'idea di "sistema nervoso" piuttosto che a quella di "marchingegno elettronico". Un esempio in tale senso è stato proposto fin dal 1943 da McCulloch (un neurofisiologo) e Pitts (un matematico) che proposero un modello matematico del sistema nervoso noto sotto il nome di "rete neurale". Il sistema nervoso degli esseri viventi, come è noto, si basa su particolari cellule, chiamate "neuroni", che sono collegate tra loro da connessioni chiamate "sinapsi".



Il funzionamento di tale sistema si basa sulla trasmissione di impulsi da un neurone all'altro, impulsi che viaggiano lungo le sinapsi. E' possibile definire un modello matematico del sistema nervoso astruendo dal tipo di materia di cui è costituito, dalla velocità di trasmissione dei segnali, dall'energia consumata per il funzionamento ed altro. Per descrivere tale modello cominciamo a definire la nozione di neurone in base alle seguenti considerazioni:

1. supponiamo che lungo le sinapsi possano viaggiare segnali di intensità diversa, intensità misurata da un numero reale;
2. supponiamo che ciascuna sinapsi ha il potere di modificare (ampliare o ridurre) l'intensità del segnale che l'attraversa tramite una costante moltiplicativa.
3. supponiamo che il neurone fornisca una risposta che dipenda dalla somma di tutti i segnali (modulati dalle sinapsi) ricevuti

Possiamo rappresentare questa idea generale di neurone al modo seguente:



dove:

- x_1, \dots, x_n sono le intensità dei segnali ricevuti lungo le sinapsi di ingresso,
- c_1, \dots, c_n sono i pesi lungo le sinapsi di ingresso che servono a modificare l'intensità dei segnali ricevuti,
- il numero $s = -c$ viene chiamato soglia
- $g : R \rightarrow R$ è una funzione detta funzione di attivazione
- la quantità $P = c_1x_1 + \dots + c_nx_n$ viene chiamata potenziale.

La quantità $g(c_1x_1+\dots+c_nx_n + c)$ rappresenta l'intensità del segnale che il neurone trasmette ad altri neuroni attraverso la sinapsi di uscita. Come vedremo, i pesi sono parametri che devono essere determinati tramite un meccanismo di apprendimento. Da un punto di vista matematico un neurone si identifica con uno strumento per calcolare il valore della funzione $g(c_1x_1+\dots+c_nx_n + c)$, cioè il valore di una funzione lineare $c_1x_1+\dots+c_nx_n + c$ opportunamente modificato tramite una funzione g .

Una classe particolarmente importante di neuroni, che chiameremo *digitali* è il corrispondente delle porte logiche considerate nel capitolo 2. Un neurone viene chiamato *digitale* se:

1. si suppone che i pesi siano solo del tipo 1 o -1,
3. si suppone che la funzione di trasferimento g sia "a soglia" cioè che sia $g(x) = 1$ se $x \geq 0$ e $g(x) = 0$ altrimenti.

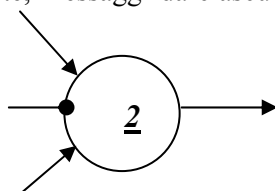
Un neurone digitale verrà rappresentato da un cerchio (il neurone) al cui interno è scritta la soglia ed in cui "entrano" linee di due tipi che prendono il nome di

sinapsi eccitatorie (in cui il peso è 1) indicate con

e *sinapsi inibitorie* (in cui il peso è -1) indicate con

Indicheremo ad esempio al modo seguente un neurone con soglia 2 due sinapsi eccitatorie ed una sinapsi inibitoria. Il comportamento di un tale neurone è descrivibile al modo seguente.

- il neurone riceve, ad un dato istante, messaggi da ciascuna sinapsi in entrata tramite impulso (indicato al

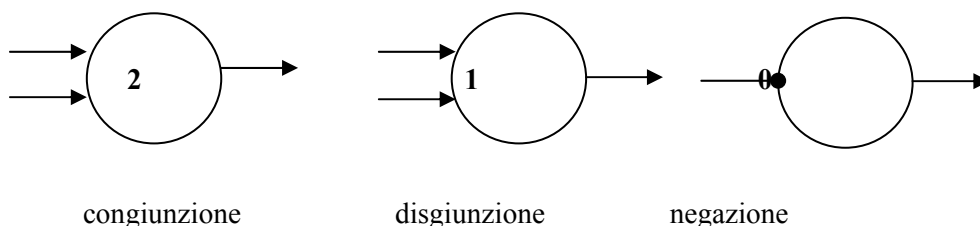


solito con 1) o non impulso (indicato con 0);

- se il potenziale che è uguale alla somma degli impulsi ricevuti dalle sinapsi eccitatorie meno la somma di quelli lungo quelle inibitorie supera la soglia allora il neurone assume lo "stato di eccitazione", altrimenti assume lo "stato di quiescenza";

- all'istante successivo, il neurone emette un impulso (in gergo "spara") se è in stato di eccitazione mentre non lo emette se è in stato di quiescenza.

Ne segue che, se si suppone che i valori input siano solo del tipo 0 ed 1, allora un neurone con n sinapsi determina una tavola di verità ad n entrate. I seguenti neuroni corrispondono ai principali connettivi logici.



Infatti il primo neurone emette un impulso solo se la somma degli impulsi ricevuti è maggiore o uguale a due. In altre parole se e solo se riceve un impulso da entrambe le sinapsi. Se chiamo con p_1 e p_2 le due sinapsi, allora il comportamento di tale neurone è descritto dalla prima delle seguenti tavole

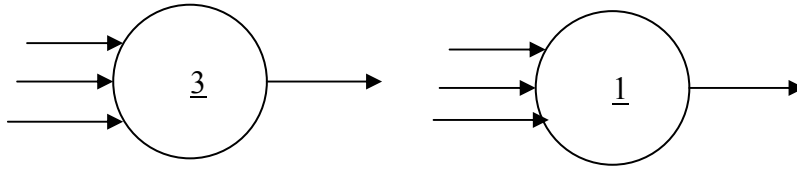
p_1	p_2	
1	1	1
1	0	0
0	1	0
0	0	0

p_1	p_2	
1	1	1
1	0	1
0	1	1
0	0	0

che coincide con la tavola di verità della congiunzione. Il secondo neurone emette un impulso quando riceve un impulso da almeno una sinapsi e quindi il suo comportamento è descritto dalla seconda delle tavole che coincide con la tavola di verità della disgiunzione. Infine nel terzo neurone se l'input è 1 allora la sinapsi inibitoria agisce e viene emesso output 0. Se l'input è 0 allora non vi è inibizione e poiché la soglia viene

comunque superata, il neurone emette 1. Questo significa che tale neurone rappresenta il connettivo della negazione.

Non è difficile generalizzare tali neuroni in modo da rappresentare le tavole di verità della congiunzione o la disgiunzione finita. Ad esempio i seguenti neuroni corrispondono alla congiunzione ed alla disgiunzione di tre formule



Esercizio. Descrivere un neurone capace di realizzare una tautologia ad n variabili (cioè una funzione ad n variabili costantemente uguale ad 1). Descrivere un neurone capace di realizzare una contraddizione ad n variabili (cioè una funzione ad n variabili costantemente uguale a 0).

2. Un neurone non è sufficiente: le reti di neurali

Anche se i singoli neuroni si presentano un sistema più potente rispetto alle porte logiche considerate nel capitolo 2, non tutte le tavole di verità possono essere ottenute tramite un unico neurone.

Proposizione 2.1. La tavola di verità dell'implicazione non può essere ottenuta tramite un unico neurone.

Dim. Ricordiamo che la tavola di verità t dell'implicazione \rightarrow è rappresentata dalla tabella

p_1	p_2	
1	1	1
1	0	0
0	1	1
0	0	1

Supponiamo per assurdo che esista un neurone con due sinapsi in entrata corrispondente a tale tavola. Allora poiché $t(1,1) = 1$ tale neurone non potrebbe avere sinapsi inibitorie. D'altra parte la soglia non può essere 0 perché altrimenti sarebbe $t(1,0) = 1$. Non può essere maggiore di zero perché altrimenti $t(0,0) = 0$.

Esercizio. Dimostrare che la tavola di verità dell'equivalenza non può essere ottenuta da un unico neurone.

Esercizio. Descrivere tutti i neuroni con una sola sinapsi in ingresso.

Esercizio. Descrivere tutti i neuroni con due sinapsi in ingresso.

Per poter realizzare tutte le tavole di verità dobbiamo utilizzare più neuroni opportunamente collegati e passare quindi al concetto di rete neurale, cioè un insieme di neuroni collegati tra di loro tramite sinapsi.

Definizione 2.2. Diciamo che una terna $(Neur, p, g)$ è una *rete neurale* se:

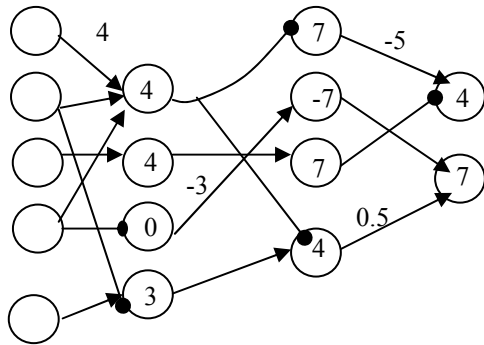
- $Neur$ è un insieme finito i cui elementi vengono detti *neuroni*,
- $a : Neur \rightarrow R$ è una funzione chiamata *funzione soglia*,
- p è una funzione di $Neur \times Neur$ in R detta *funzione peso*,
- g è una funzione di R in R detta *funzione di attivazione*.

Il valore $a(i)$ viene detto *soglia* del neurone i , il valore $p(i,j)$ *peso* della sinapsi che porta dal neurone i al neurone j . Il fatto che $p(i,j) = 0$ si esprime dicendo che *non esiste una sinapsi da i a j* . Ogni rete può essere "*disegnata*" sul piano rappresentando

- ciascun neurone con un cerchio
- ponendo all'interno del cerchio rappresentante il neurone i la relativa soglia $a(i)$
- non ponendo nessuna freccia se $p(i,j) = 0$

- tracciando una freccia del tipo \rightarrow che va da i a j se $p(i,j) = 1$
- tracciando una freccia del tipo $\xrightarrow{\bullet}$ se $p(i,j) = -1$
- ponendo una freccia con sopra il numero $p(i,j)$ negli altri casi.

I neuroni input sono quelli verso cui non arriva nessuna freccia. Più precisamente chiamiamo *neurone input* un neurone j tale che $p(i,j) = 0$ per ogni $i \in \text{Neur}$. I neuroni output sono quelli da cui non parte nessuna freccia. Più precisamente diciamo che i è un *neurone di output* se $p(i,j) = 0$ per ogni $j \in \text{Neur}$. Ecco un esempio di rete con 5 neuroni input e 2 output.



Definiamo il comportamento di una rete neurale supponendo che in ogni istante ciascun neurone sia in uno stato che identifichiamo con il numero in uscita, cioè l'intensità dell'impulso che il neurone sta per trasmettere ai neuroni a cui è collegato.

Definizione 2.3. Chiamiamo *stato* di una rete neurale una funzione $s : \text{Ne} \rightarrow R$, chiamiamo *stato di un neurone i* il valore $s(i)$.

Se in un dato istante un neurone i è nello stato $s(i)$ vuol dire che nell'istante successivo trasmetterà il valore $s(i)$ ai neuroni verso cui è collegato. Supponiamo inoltre che i neuroni input ricevano segnali dal mondo esterno e che i neuroni output emettano segnali verso il mondo esterno. Per il momento ci limitiamo a studiare le reti neurali a strati.

Definizione 2.4. Una *rete neurale ad r strati* è tale che Neur è diviso in $r+1$ parti S_0, \dots, S_r tali che:

- il primo strato S_0 contiene solo *neuroni input*,
- l'ultimo strato S_r contiene solo *neuroni output*,
- una sinapsi può solo partire da un neurone di S_i ed arrivare ad un neurone di S_{i+1} .

La rete che abbiamo disegnato ha 4 strati. In una rete a strati ogni strato trasmette impulsi allo strato successivo. Una descrizione più precisa del comportamento di una rete a strati è la seguente.

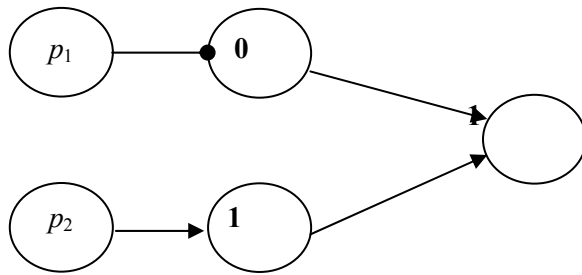
Definizione 2.5. Ogni rete neurale ad r strati con n neuroni input, $S_0 = \{i_1, \dots, i_n\}$, ed m neuroni output, $S_r = \{o_1, \dots, o_m\}$, determina una funzione di R^n in R^m che si ottiene attraverso i seguenti passi:

- passo 1 (INPUT): dato un input $x = (x_1, \dots, x_n) \in R^n$, i neuroni input i_1, \dots, i_n assumono come "stato" i valori x_1, \dots, x_n a tutti gli altri neuroni assegniamo un qualunque stato, ad esempio 1,
- passo k (per $k = 2, \dots, r$): supposto che $S_{k-1} = \{j_1, \dots, j_i\}$ e che gli stati di tale strato siano $s(j_1), \dots, s(j_i)$, ciascun neurone i dello strato S_k assume lo stato $g(p(j_1, i)s(j_1) + \dots + p(j_i, i)s(j_i) - a(i))$
- passo $r+1$: si assume come output la m -pla di stati assunti dai neuroni output in S_r .

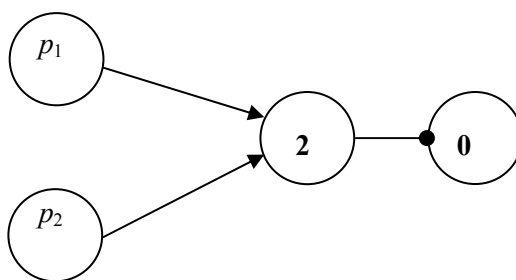
Da notare che, come avviene sempre nelle reti, entra in gioco la variabile tempo e che l'output si manifesta dopo r passi dal momento in cui è stato fornito l'input.

3. Reti neurali digitali.

In particolare, per $m = 1$, e se tutti i neuroni sono digitali e se supponiamo che in ingresso siano forniti solo input in $\{0,1\}$, allora una rete neurale a strati realizza una tavola di verità. Un esempio è dato dalla seguente rete neurale a due strati che realizza l'implicazione (che abbiamo visto non può essere realizzata con un unico neurone)



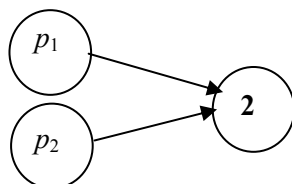
Un altro esempio è la seguente rete neurale a due strati che realizza la tavola di verità della negazione della congiunzione $\neg(p_1 \wedge p_2)$



Come era prevedibile, è bastato mettere in sequenza il neurone della congiunzione con quello della negazione.

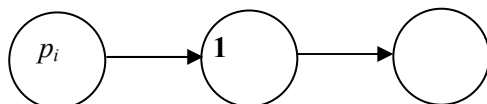
Nota 1. Se si pone uguale ad uno il tempo di risposta di un neurone ad uno stimolo, allora nelle reti ad r strati l'uscita che a noi interessa avviene con ritardo di r .

Nota 2. Stante la definizione data, gli input non vengono da linee esterne alla rete ma facendo assumere ai neuroni input opportuni stati. In base a tale convenzione, i singoli neuroni per poter essere considerati reti devono essere forniti anche di un strato di neuroni input. Ad esempio la congiunzione deve essere rappresentata da:

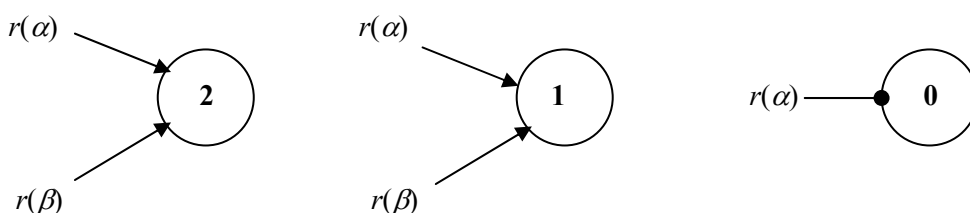


Teorema 3.1 Ogni tavola di verità può essere realizzata da una opportuna rete neurale.

Dim. Poiché abbiamo già visto che esistono neuroni capaci di simulare i connettivi logici di congiunzione, disgiunzione e negazione, il teorema si prova allo stesso modo in cui nel capitolo 2 è stato provato l'analogo teorema sulle porte logiche. Infatti si associa ad ogni variabile proposizionale p_i la seguente rete $r(p_i)$

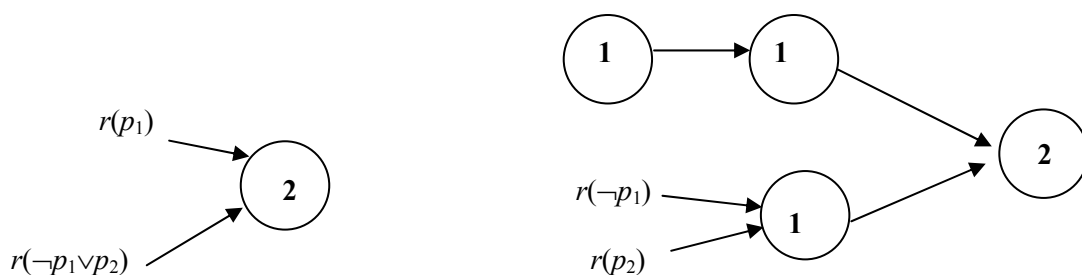


Inoltre se α e β sono due formule, e $r(\alpha)$ sia una rete che realizza α ed $r(\beta)$ una rete che realizza β allora le reti

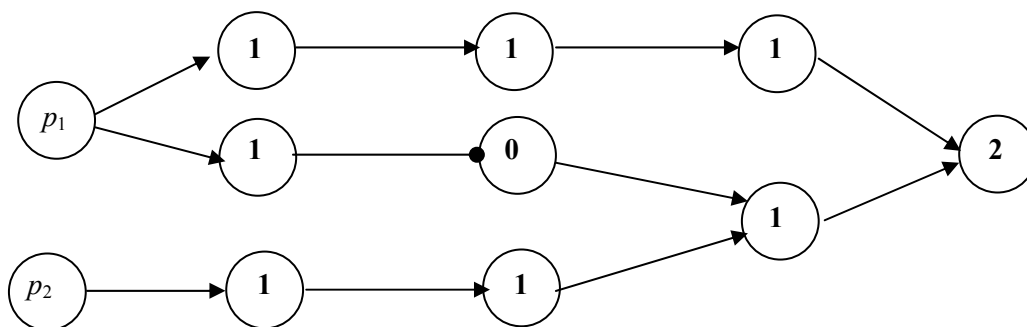


realizzano le tavole di verità di $\alpha \wedge \beta$ e $\alpha \vee \beta$ e $\neg \alpha$, rispettivamente. Precisamente le prime due reti vanno intese al modo seguente. Se α e β contengono una stessa variabile proposizionale p_i allora si deve provvedere ad aggiungere un neurone che si dirama nei due neuroni che in $r(\alpha)$ e $r(\beta)$ corrispondono a p_i .

Esempio. Data la formula $p_1 \wedge (\neg p_1 \vee p_2)$ si procede a costruire le reti $r(p_1)$ e $r(\neg p_1 \vee p_2)$ per poi "congiungerle". A sua volta la rete $r(\neg p_1 \vee p_2)$ si ottiene disgiungendo le reti $r(\neg p_1)$ e $r(p_2)$ ed $r(\neg p_1)$ negando la rete $r(p_1)$. In definitiva si ottiene



e quindi

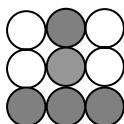


Proposizione 3.2. Ogni rete combinatoria può essere realizzata tramite una opportuna rete neurale.

Dim. Abbiamo già osservato che una rete combinatoria con m linee di uscita equivale ad m reti combinatorie ad una uscita. Allora è sufficiente collegare opportunamente le corrispondenti m reti neurali per realizzare la data rete combinatoria.

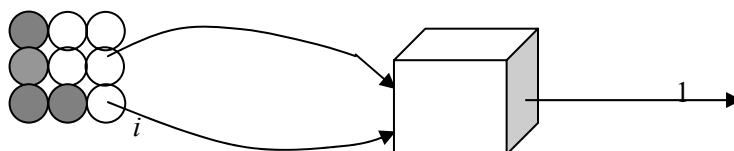
4. Macchine che riconoscono.

Da quanto è stato detto nel paragrafo precedente sembrerebbe che le reti neurali non facciano niente di più di quanto siano capaci di fare le porte logiche, cioè che siano solo un particolare tipo di porte logiche. La grande novità delle reti neurali invece si manifesta quando ci si imbatte nel problema della complessità dei problemi da affrontare. Per fare un esempio ci riferiamo alla questione di “riconoscere” i testi scritti a mano. Possiamo immaginare che una lettera dell’alfabeto scritta a mano si guardata tramite un occhio elettronico costituito da un rettangolo di p per q punti che chiameremo *pixels*. Se un pixel “guarda” un tratto scuro di penna manda un segnale, altrimenti non manda nessun segnale. Questo significa che quando l’occhio elettronico guarda una figura la trasforma in una serie di impulsi-non-impulsi, cioè una serie di 1 e 0. Se enumeriamo tali pixels man mano che vengono “letti” tale input sarà costituito da una n -pla, con $n = p \times q$ di valori booleani di tipo 0 ed 1. Ad esempio, supposto che l’occhio abbia $n = 3 \times 3$ pixels, se come input si presenta la seguente figura

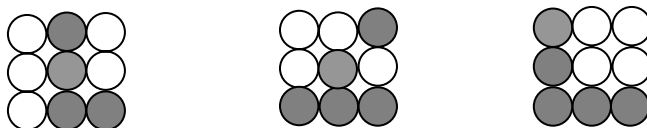


tale input viene trasformato nel vettore $(0,1,0,0,1,0,1,1,1)$.

Poniamoci ora il problema di riconoscere una lettera dell’alfabeto, ad esempio la lettera L , allora si deve costruire un marchingegno che riceve in input vettori booleani di lunghezza n e che restituisca 1 se tali vettori sono un modo di scrivere a lettera L mentre restituisce 0 altrimenti.



Nella figura abbiamo disegnato una macchina che riconosce la lettera L , macchina che utilizza un occhio costituito da 3×3 pixels. La figura indicata in grigio, che rappresenta uno dei modi per scrivere L , viene codificata tramite il vettore booleano $(1,0,0,1,0,0,1,1,0)$. Gli altri modi di scrivere L potrebbero essere i seguenti



che vengono codificati rispettivamente tramite i vettori

$$(0,1,0,0,1,0,0,1,1), (0,0,1,0,1,0,1,1,1), (1,0,0,1,0,0,1,1,1).$$

Se indichiamo con F l'insieme dei vettori che rappresentano i diversi modi di scrivere L allora tale macchina:

- deve emettere 1 per ogni vettore in F
- deve emettere 0 per tutti i vettori che non appartengono ad F .

In definitiva si tratta di costruire una rete di porte logiche corrispondente ad una tavola di verità con 9 variabili proposizionali in cui nella colonna degli output compare 1 in 4 parti. Ma abbiamo già visto che il teorema di completezza funzionale ci assicura che una tale macchina esiste e ci dice anche come è fatta.

Basta

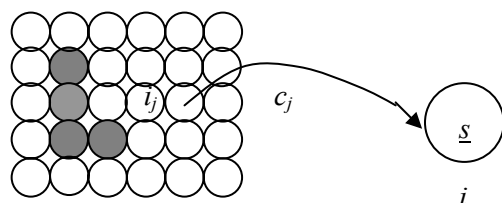
- considerare una opportuna formula ottenuta come disgiunzione di 4 formule ognuna delle quali è congiunzione di nove letterali
- trasformare tale formula in una rete di porte logiche.

In definitiva se chiamiamo *forma* un qualunque insieme $F \subseteq \{0,1\}^n$ di vettori booleani di data lunghezza n , abbiamo il seguente teorema.

Teorema 1. Comunque si fissi il numero n di pixels di un sensore e qualunque sia la forma F , esiste una rete di porte logiche capace di riconoscere F .

Fino a qui tutto sembra banale. Il problema nasce dal fatto che se vogliamo un sistema di riconoscimento ragionevole dobbiamo aumentare notevolmente il numero dei pixels. In tale caso il problema di costruire la rete, pur essendo teoricamente semplice, non diventa praticamente trattabile. Infatti se tale numero è n allora sono coinvolte tavole di verità con n ingressi e quindi con 2^n righe. Già nel caso trattato di 9 pixels si tratta di considerare tavole con $2^9 = 512$ righe. Se poi si vuole un sensore ragionevole che abbia 10×10 pixels allora dobbiamo affrontare una tavola di verità con un numero di righe uguale a $2^{100} = 1.267.650.600.228.229.401.496.703.205.376$.

Per evitare il problema della complessità è necessario cambiare completamente la nostra idea di "macchina". Non avendo la capacità di costruire la rete di porte logiche che ci serve, vogliamo ora costruire macchine più semplici ma che, in un certo senso, "imparino" ad avere il comportamento che noi vogliamo. Per illustrare tali tipo di macchine, consideriamo una rete neurale molto semplice che prende il nome di *perceptron*. Un *perceptron* è una semplice rete neurale a strati costituita dal primo strato i_1, \dots, i_n di neuroni input e da un unico neurone output i con soglia s .



Immaginiamo che lo strato input sia disposto in un rettangolo come disegnato in figura. La connessione da un neurone input i_j al neurone output i ha però la capacità di modificare il segnale ricevuto moltiplicandolo per una quantità c_j (quindi ampliandolo se c_j è maggiore di 1 o diminuendolo se c_j è minore di 1). Se la figura percepita si trasforma in un vettore booleano $x = (x_1, \dots, x_n)$, allora al vettore output viene convogliato un segnale totale $\sum_i c_i x_i$ che prende il nome di *potenziale*. Nel perceptron si suppone che il neurone i spari solo se il suo potenziale supera la soglia di eccitazione. Pertanto

- se $\sum_i c_i x_i \geq c$ il neurone spara
- se $\sum_i c_i x_i < c$ il neurone non spara.

Se $F \subseteq \{0,1\}^n$ è la forma da riconoscere, allora noi vogliamo che ad ogni elemento di F dato come input deve corrispondere 1 come output e quindi uno sparo del neurone, ad ogni elemento non in F deve corrispondere 0 e quindi un non-sparo. In termini matematici, ponendo $c = -s$, ciò significa che i pesi e la soglia devono essere tali che:

$$x \in F \Rightarrow \sum_i c_i \cdot x_i + c \geq 0 \quad \text{e} \quad x \notin F \Rightarrow \sum_i c_i \cdot x_i + c < 0. \quad (1)$$

Definizione 2. Diremo che una forma F è *separabile* se esistono dei pesi c_1, \dots, c_n , c per cui valgono le condizioni dette sopra.

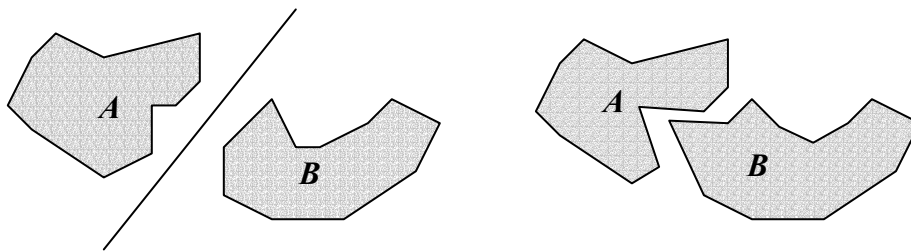
Pertanto F è separabile se esiste un perceptron capace di dire se una forma appartiene o meno ad F . Si osservi che se esistono c_1, \dots, c_n , c che verificano (5.1) allora esistono anche c_1, \dots, c_n , c tali che la prima disuguaglianza vale in senso stretto, cioè tali che

$$x \in F \Rightarrow \sum_i c_i \cdot x_i + c > 0 \quad \text{e} \quad x \notin F \Rightarrow \sum_i c_i \cdot x_i + c < 0. \quad (2)$$

La parola “separabile” deriva dalla seguente interpretazione geometrica. Associamo ad ogni vettore booleano (x_1, \dots, x_n) il punto di coordinate x_1, \dots, x_n nello spazio euclideo di dimensione n . Tali punti si disporranno lungo i vertici di un ipercubo. Ora parte di tali vertici corrisponderanno a vettori in F ed i rimanenti vertici, chiamiamoli F' , a vettori che non appartengono ad F . Ricordiamo ora una semplice nozione di geometria euclidea.

Definizione 3. Diciamo che due insiemi di punti A e B sono *separabili* dall'iperpiano di equazione $\sum_i c_i \cdot x_i + c = 0$ se risulta $\sum_i c_i \cdot x_i + c > 0$ per ogni $(x_1, \dots, x_n) \in A$ e $\sum_i c_i \cdot x_i + c < 0$ per ogni $(x_1, \dots, x_n) \in B$.

Ecco una coppia di insiemi A e B che sono separabili ed una coppia che non sono separabili



Allora una forma F è separabile se è separabile dall'insieme F' dei punti dell'ipercubo unitario che non appartengono ad F . Da notare che esistono esempi di forme che non sono separabili e quindi che non sono riconoscibili da un Perceptron.

5. Macchine che apprendono dall'esperienza.

Sarebbe possibile, in linea di principio, determinare i pesi in un Perceptron mediante opportuni calcoli. Tutto sommato si tratta di un semplice problema geometrico che consiste nel trovare un opportuno iperpiano e che analiticamente si traduce in un sistema di disequazioni nelle incognite c_1, \dots, c_n , c . Purtroppo però per tale approccio geometrico vale lo stesso discorso che abbiamo fatto per la costruzione della rete di porte logiche. Nei casi interessanti il numero n dei sensori è altissimo e quindi il numero di incognite è altissimo. Non è quindi possibile effettuare gli opportuni calcoli in tempi ragionevoli. Si è pensato allora di utilizzare, per la determinazione dei pesi, tecniche di "apprendimento" per tentativi ed errori. Ciò è possibile proponendo al perceptron in successione tutti i possibili input $x = (x_1, \dots, x_n)$ e “comunicandogli” ogni volta se ha dato la risposta giusta o meno. Senza entrare nei dettagli, dirò che, dopo aver assegnato a caso i pesi iniziali ed aver ordinato in modo ciclico gli elementi di $\{0,1\}^h$, si procede al modo seguente. Si suppone che c sia uguale a zero per semplificare i calcoli (si vede facilmente che ciò non è restrittivo), poi:

- i) si propone la prima forma x al perceptron;
- ii) se la risposta è corretta ($x \in F$ e l'output è 1 oppure $x \notin F$ e l'output è 0) il perceptron lascia immutati i propri pesi;
- iii) se la risposta è scorretta in quanto $x \in F$ e l'output è 0 ($\sum_i c_i \cdot x_i < 0$) allora il perceptron si auto-modifica aumentando i propri pesi di una data quantità (ad esempio pone $c_i := c_i + x_i$); in tale modo si cerca di aumentare il potenziale in modo da avere la risposta desiderata 1

iv) se la risposta è scorretta in quanto $x \notin F$ e l'output è 1 ($\sum_i c_i \cdot x_i \geq 0$) allora il perceptron si auto-modifica diminuendo i pesi di una data quantità (ad esempio pone $c_i := c_i - x_i$); in tale modo si cerca di diminuire il potenziale in modo da avere la risposta desiderata 0

v) si sottopone al perceptron la forma successiva ad x e si riprende il procedimento a partire da ii).

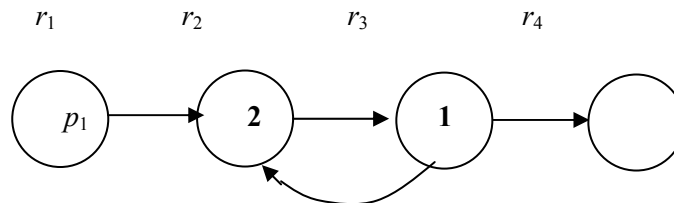
Si dimostra che, se F è separabile, dopo un numero finito di passi di tale procedimento il perceptron finisce col dare sempre risposte esatte e con questo finisce la fase di apprendimento.

A volte si propongono gli input non in maniera ciclica ma in maniera casuale. In situazioni molto complesse (che poi sono quelle in cui vale la pena di utilizzare le reti neurali) a volte ci si contenta di un comportamento quasi-corretto della rete, nel senso che si può considerare finita la fase di apprendimento quando una alta percentuale di risposte è corretta.

Nota. Quello delle macchine "che apprendono" è un capovolgimento totale dell'ordinario modo di considerare i calcolatori. Infatti usualmente un calcolatore viene considerato come un semplice esecutore di ordini, ordini dati tramite una successione di istruzioni elementari raccolte in un programma. Invece con le reti neurali all'attività di "programmare" viene sostituita l'attività di "addestrare" con una successione di atti di tipo "premio" (se la risposta è esatta) o "punizione" (se la risposta è sbagliata). Ciò comporta ad esempio che le prestazioni di una macchina possono migliorare con il tempo. Inoltre, mentre nella programmazione è essenziale l'attività del programmatore, l'apprendimento può avvenire anche tramite una diretta interazione della macchina con l'ambiente o con una persona non esperta di programmazione. Infine, mentre nella programmazione classica si conosce esattamente quale è il programma scritto ed il comportamento della macchina, nel caso di reti neurali che apprendono, se il numero di neuroni è molto grande ed il processo di apprendimento lungo, allora non si è in grado di conoscere la configurazione interna della rete (la distribuzione dei pesi). Tale configurazione infatti con il tempo si è andata modificando.

6. Reti neurali che ricordano

Fino ad ora ci siamo limitati ad utilizzare le reti neurali solo per simulare le tavole di verità e le reti combinatorie, cioè gli automi digitali senza memoria. Se si toglie la restrizione per cui le reti sono divise a strati, è possibile però riprodurre comportamenti dove entra in gioco la memoria. Infatti è sufficiente considerare reti in cui l'impulso trasmesso da un neurone può, in un certo senso "tornare indietro". Consideriamo ad esempio la seguente rete



Il suo comportamento dipende non solo dall'input nel neurone p_1 ma anche dallo "stato" del terzo neurone. Per reti neurali di tale tipo chiameremo "stato" della rete una certa distribuzione degli stati possibili (di eccitazione o meno) dei singoli neuroni (esclusi quelli di input o di output). In termini matematici uno stato si può rappresentare con una funzione che ad ogni neurone associa 0 od 1. Lo stato dei neuroni input è definito dall'input, lo stato dei neuroni output viene determinato dalla dinamica della rete e letto come output. Ad esempio, se lo stato di r_3 è di eccitazione e viene fornito output 1, allora la soglia di r_2 viene superata, viene superata anche quella di r_3 e viene fornito output 1. Se invece lo stato di r_3 è 0 allora lo stesso output 1 non è sufficiente a superare la soglia di r_2 e quindi l'output finale sarà 0.

Nel nostro corso non ci inoltriamo nello studio generale delle le reti neurali, limitandoci a quelle a strati. Ci limitiamo ad enunciare il seguente teorema.

Teorema 1. Le reti neurali sono automi finiti (in cui uno stato è rappresentato dall'insieme degli stati dei singoli neuroni) ed ogni automa finito può essere realizzato da una opportuna rete neurale.

Nota. Si osservi che nel cervello di un individuo adulto esistono circa 10 miliardi di neuroni che da ogni neurone è collegato mediamente con 10.000 altri neuroni. Queste cifre danno un'idea della complessità e quindi della "potenza di calcolo" del cervello umano.

7. Algoritmi genetici (da scrivere)

8. Vita artificiale (da scrivere)

Bibliografia.

C. Domenicani, M. Jordan, Discorsi sulle reti neurali e l'apprendimento, Franco Angeli Editore 2001.