Lezione 4 - 20/10/2023

Reti neurali

Il cervello come calcolatore

L'elaborazione è frutto di un processo altamente parallelo
LA potenza di calcolo deriva dalla cooperazione di molti processori semplici e fortemente interconnessi:

10^{10} - 10^{11} neuroni
10^{5} connessioni/ neurone

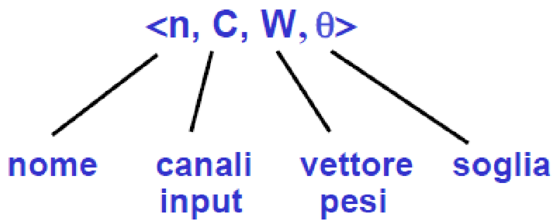
Quindi tanti neuroni, noi non riusciamo ad averne così nelle reti artificiali. E soprattutto tante connessioni, decine di migliaia di connessioni per singolo neurone.

L'apprendimento è il cambiare la rappresentazione che ho all'interno del sistema mano a mano, in base ai dati. Nel sistema biologico quello che si modifica sono le intensità delle connessioni tra i neuroni.

Neuroni formali

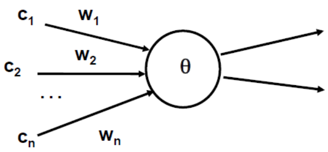
Il segnale prodotto (in uscita) dal neurone è lo stato, dovremo indicare se questo è binario, numero reale, etc.
Poi dovremo definire come calcolare il passaggio di aggiornamento, ovvero la funzione di transizione che cambia lo stato quando un segnale passa in un neurone.
C'è anche una funzione di uscita.

Modello formale di un neurone di McCulloch:



La soglia viene usata nel calcolo della funzione di transizione. Esempio:

- Insieme degli stati: {0,1} o {-1,1}
- Vettore di pesi per gli input: w_i $i = 1, \dots, n$
- Funzione di transizione: $s(t+1) = 1$ *sse* $\Sigma w_i s_i(t) \geq \theta$



In questo caso la funzione di uscita sarebbe l'identità, ovvero non altera lo stato generato dalla funzione di transizione.

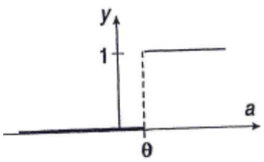
Esempi di funzioni di transizione:

A gradino

Insieme stati = {0,1}

$$f(x) = \begin{cases} 1 \text{ se } x > \theta \\ 0 \text{ altrimenti} \end{cases}$$

ove $x = \Sigma w_i s_i$

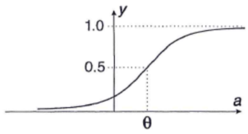


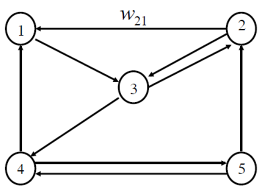
Funzione logistica o Sigmoide

Insieme stati = R

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

ove $x = \Sigma w_i s_i$





Esempio di connessioni:

tipicamente non ci sono archi doppi o loop, ma sarà più un grafo aciclico, ovvero un albero.

Formalmente:

$| \mathbf{w_{ij}} |$ matrice dei pesi

$| \vartheta_i |$ vettore delle soglie

$$n_i(t) = \sum_{j=1}^{n} w_{ij} \cdot x_j(t) - \vartheta_i$$ input netto a i al tempo t

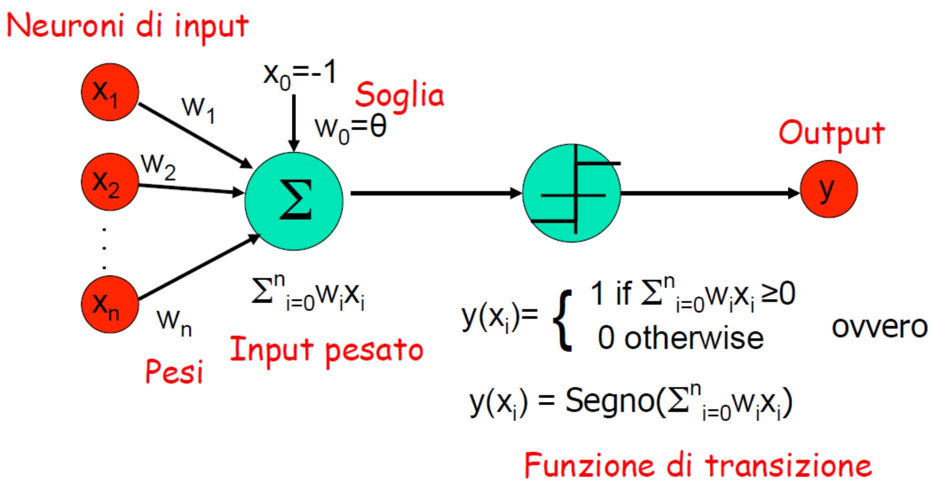
$$x_i(t+1) = g(n_i(t))$$ funzione di transizione

w_{ij} è il peso delle connessioni entranti, ovvero il peso da j a i.
Teta i è la soglia di i.
La terza formula è la somma pesata dei segnali entranti, confrontandola con la soglia del neurone.
g è lo "scatto" che può essere come visto sopra un gradino o una sigmoide.

Percettrone

Ora andremo a prendere il percettrone, un tipo di neurone specifico, con il suo specifico algoritmo di apprendimento. Avremo quindi scatto al posto di sigmoide, etc. Il percettrone è sempre solitario.

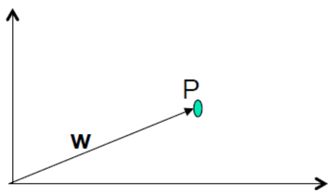
Questo è un percettrone in modo formale:



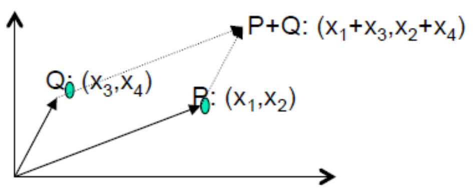
Gli stati interni possono essere polarizzati -1 +1 (il secondo cerchio verde acqua). Cioè la funzione è a scatto con uscita -1 +1.

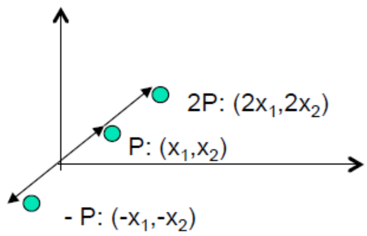
L'output (ovvero la funzione di transizione) prende la sommatoria di tutti i segnali in ingresso x_i (solo segnali, non neuroni, prendiamo il percettrone singolo senza la rete neurale), con i pesi delle connessioni w_i, con scatto sullo zero. Questo non è addestramento, è inferenza. Viene in questo modo calcolata l'uscita del percettrone dato l'ingresso. L'apprendimento starà nel cambiare questi pesi. La soglia diventerà anche cambiabile in fase di addestramento grazie a $x_0 = -1$ fisso, dato che in fase di addestramento cambieremo tutti i pesi w. Quindi nel calcolo dell'output c'è anche teta.

Concentriamoci ora sulla sommatoria. Il vettore x è un vettore di input, w è un vettore di pesi. Una coppia x e w può essere visto come un punto sul piano. Il punto può essere associato ad un vettore se lo fisso sull'origine, ottenendo le caratteristiche di verso, direzione e lunghezza dei vettori.



La relazione tra vettori può diventare calcolabile, sommando due vettori e creandone un terzo, oppure moltiplicandone uno per un numero scalare che se negativo rovescia il verso del vettore.





Il prodotto interno tra due vettori in R^n è il numero reale

$$\langle \mathbf{x}, \mathbf{y} \rangle \equiv \mathbf{x} \cdot \mathbf{y}^T = \sum_{i=1}^{n} x_i y_i$$

Proprietà:

1. $\langle x, y \rangle = \langle y, x \rangle$
2. $\langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle$ e $\langle x, y + z \rangle = \langle x, y \rangle + \langle x, z \rangle$
3. $\langle \lambda x, y \rangle = \lambda \langle x, y \rangle$ e $\langle x, \lambda y \rangle = \lambda \langle x, y \rangle$
4. $\langle x, x \rangle \geq 0$

La norma (modulo, lunghezza) di un vettore in R^n è il numero reale <u>non negativo</u>

$$\|\mathbf{x}\|_2 \equiv \sqrt{x \cdot x^T} = \sqrt{\sum_{i=1}^{n} x_i x_i} = \sqrt{\langle x, x \rangle}$$

Radice di un prodotto interno per se stesso. proprietà:

1. $\| \mathbf{x} + \mathbf{y} \| \leq \| \mathbf{x} \| + \| \mathbf{y} \|$
2. $\| \lambda \mathbf{x} \| = \lambda \| \mathbf{x} \|$
3. $\| \mathbf{x} \| > 0 \quad \text{se } \mathbf{x} \neq \mathbf{0}$

Vettore unitario

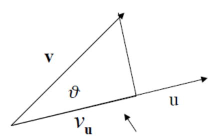
$$\frac{\mathbf{w}}{\| \mathbf{w} \|} \quad \text{se} \quad \mathbf{w} \neq \mathbf{0}$$

Crea un altro vettore con lo stesso orientamento ma unitario rispetto alla lunghezza

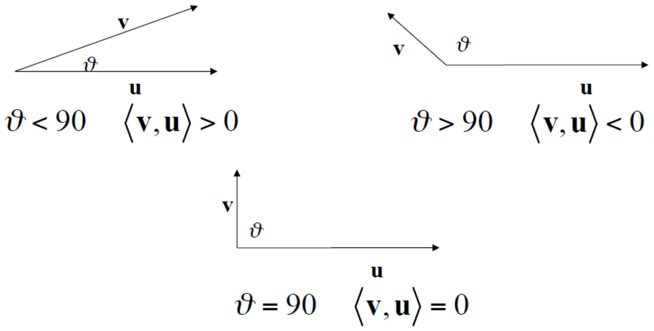
Se ho due vettori v e u posso calcolare la proiezione (cioè la sua ombra) di v su u in questo modo:

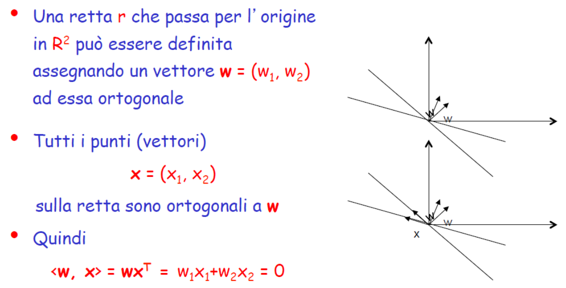
Proiezione di v su u:

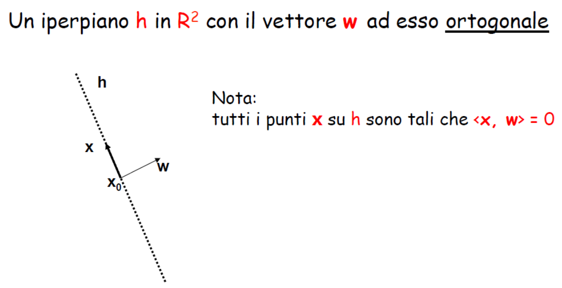
$$v_{\mathbf{u}} = (\| \mathbf{v} \| \cos \vartheta)$$

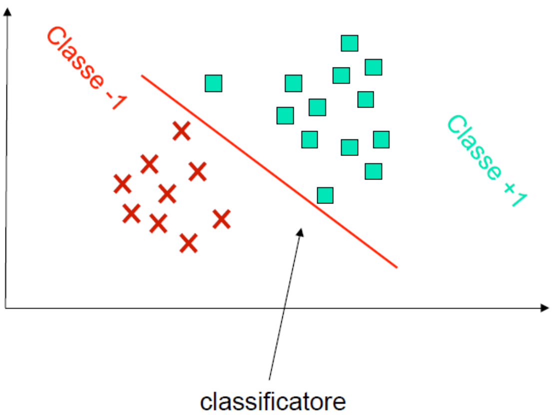


Ulteriori note:









Il neurone con i suoi pesi corrisponde all'ipotesi h, che corrisponde ad una linea che forma una classificazione.

b è lo spostamento rispetto all'origine

Apprendimento

L'apprendimento è la modifica dei pesi in base agli esempi. Il sistema sarà a due classi A e B.

La procedura

- Obiettivo è classificare vettori di input in due classi, A e B
- Si sottomette una sequenza infinita $\{x_k\}$ di vettori tale che ve ne siano un numero infinito sia di A che di B
- Per ogni x_k la rete calcola la risposta y_k
- Se la risposta è errata, si modificano i pesi, incrementando i pesi delle unità di input attive se si è risposto 0 anzichè 1, decrementandole nel caso duale:

$$w' = w \pm x$$

I dati saranno forniti in sequenza, non in lotto, in maniera infinita. Queste caratteristiche sono specifiche del percettrone.
Su ogni esempio vado a calcolare l'output, e in base alla risposta aggiusto i pesi.
Il vettore dei pesi viene aggiornato solo quando sbaglio, sommando l'esempio.
Il segno sarà diverso in base a se l'errore è dato dal risultato 0 al posto di 1, oppure 1 rispetto a 0.
0 è il "sotto" e 1 il "sopra"

Teorema di convergenza:

Comunque si scelgano i pesi iniziali, se le classi A e B sono discriminabili, la procedura di apprendimento termina dopo un numero finito di passi

Teorema di Minsky e Papert:

La classe delle forme discriminabili da un percettrone semplice è limitata alle forme linearmente separabili

Quindi quando i punti sono messi in modo non separabile, gli errori accadranno sempre. Quindi il percettrone trova la separabilità, se e solo se questa esiste.

Input	$\mathbf{x} = (x_1, \ldots, x_d)$
Input esteso	$\mathbf{x} = (x_1, \ldots, x_d, -1)$
Pesi	$\mathbf{w} = (w_1, \ldots, w_d, \theta)$

Teorema

Se l'insieme degli input estesi è partito in due classi linearmente separabili A, B allora é possibile trovare un vettore di pesi **w** tale che:

$\mathbf{w} \cdot \mathbf{x} \geq 0$	se	$\mathbf{x} \in A$
$\mathbf{w} \cdot \mathbf{x} < 0$	se	$\mathbf{x} \in B$