NEURON

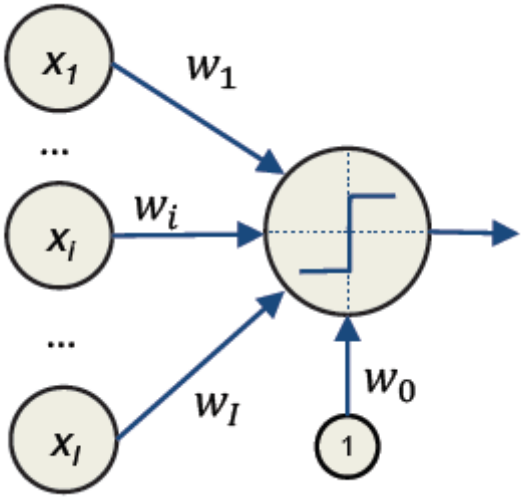
Riccardo Fontanini

Consegna

Hebbian evolution of a single neuron for OR/AND computation

Rete neurale

Una rete neurale artificiale è un modello matematico composto di "neuroni" artificiali, ispirato vagamente dalla semplificazione di una rete neurale biologica.



Una rete neurale artificiale è un sistema adattivo che cambia la sua struttura basata su informazioni esterne o interne che scorrono attraverso la rete durante la fase di apprendimento.

Un neurone è di fatto una combinazione lineare di ingressi a cui è applicata una funzione non lineare, detta anche funzione di attivazione.

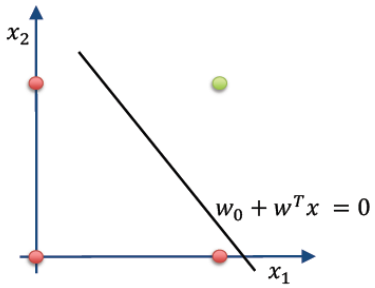
$$h_j(x|w,b) = h_j\left(\sum_{i=1}^{I} w_i \cdot x_i - b\right) = h_j\left(\sum_{i=0}^{I} w_i \cdot x_i\right) = h_j\,\mathbf{w^T x}$$

La funzione di attivazione può essere una semplice funzione a soglia oppure una più elaborata sigmoide:

$$g(x) = \frac{1}{1 + exp(-x/x_0)}$$

Più neuroni possono essere collegati insieme per creare una rete neurale.

Di fatto un neurone identifica un semispazio nel campo di variabilità degli ingressi, definendo nel caso di due valori di ingresso un semipiano di appartenenza ad una possibile soluzione.



I pesi sono fissati grazie alla fase di apprendimento, per questo progetto è stato impiegato l'apprendimento Hebbiano.

Apprendimento Hebbiano

L'apprendimento Hebbiano è una metodologia di variazione dei pesi di un determinato neurone. Il nuovo valore di un determinato peso viene calcolato in relazione a vari fattori:

- All'input associato al peso in quella determinata iterazione
- Ad un tasso di apprendimento
- All'uscita desiderata in quella determinata iterazione

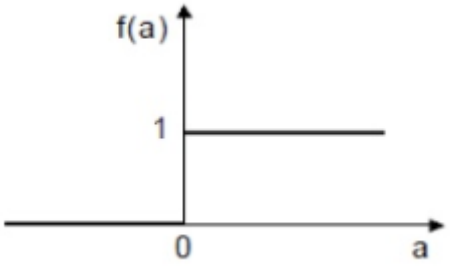
$$w_i^{k+1} = w_i^k + \Delta w_i^k$$
$$\Delta w_i^k = \eta \cdot x_i^k \cdot t^k$$

Iterando il modello di apprendimento è possibile raggiungere un determinato grado di convergenza dei pesi, in modo che a dei dati ingressi corrispondano specifiche uscite.

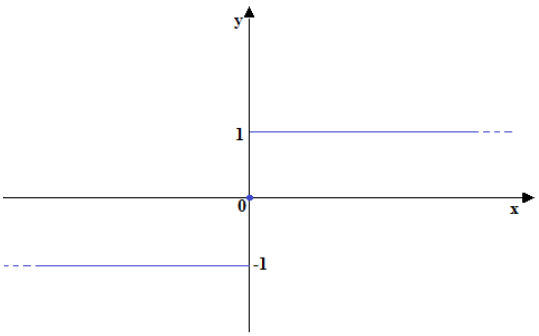
Problematiche

Valori di input

Per una prima versione del neurone si è scelto di utilizzare i classici valori di input $\{0; 1\}$ con una funzione di attivazione a soglia come in figura:



Ma con la definizione precedentemente data di apprendimento Hebbiano si è notato che non avveniva la convergenza verso dei pesi che garantivano le uscite desiderate per determinati ingressi, ad un numero di iterazioni ragionevoli. Quindi si è deciso di modificare il valore degli ingressi, cioè passare da una rappresentazione binaria $\{0; 1\}$ ad una $\{-1; 1\}$. Inoltre si è deciso di modificare la funzione di attivazione con la funzione segno:



Con questi accorgimenti il sistema si è portato a convergenza.

Sistema per il test

I test sono stati eseguiti su una macchina server che sfrutta processori Intel Xeon CPU E5-2603 e una scheda grafica NVIDIA K40.

Codice e algoritmi

Per limitare il numero di iterazioni dell'apprendimento si è scelto di imporre un numero massimo di iterazioni. Inoltre, siccome il numero di ingressi è molto ridotto, non presenta un grande sforzo computazionale testare tutte le combinazioni degli ingressi durante ogni iterazione dell'apprendimento. Quindi se per ogni combinazione degli ingressi il neurone produce l'uscita desiderata, il programma esce dal ciclo di apprendimento.

Di seguito è illustrato l'algoritmo di apprendimento.

Compilazione

Per la compilazione (in sistemi linux) è stato creato un makefile, una volta invocato il comando:

make orand

permette la compilazione di tutte le librerie utilizzate e la creazione dell'eseguibile nella cartella:

./build/linux

La compilazione avviene sfruttando lo standard C99.

È possibile specificare alcuni parametri aggiuntivi in fase di compilazione utilizzando la regola -D di gcc:

make orand D='-DOR'

Di default il programma viene generato per rappresentare una porta AND, ma impostando tale parametro si impone la generazione di una porta OR.

make orand D='-DLR=0.6'

è possibile modificare learning rate dell'algoritmo di apprendimento

make orand D=' -D W0=9 -DW1=10 -DW2=11'

Inoltre è possibile modificare i pesi iniziale dai quali l'algoritmo parte per iniziale l'apprendimento

Analisi del sistema

Convergenza

Sono stati eseguiti alcuni test per verificare la convergenza dell'algoritmo:

Come pesi inziali sono stati impostati 10 e -6, con bias -1 e learning rate pari a 0.5

Con tali parametri l'algoritmo converge ad una soluzione corretta per la AND a 61 iterazioni mentre per l'OR a 62

Se scegliamo parametri più estremi come, per esempio, 100 e 1000, con bias 100 e learning rate pari a 0.5

Con tali parametri l'algoritmo converge ad una soluzione corretta per la AND a 3993 iterazioni mentre per l'OR a 3202

In tutte le prove effettuate si trova una convergenza, però non è stato possibile trovare una regola precisa che permetta di quantificare a priori il numero di cicli necessari a raggiungere ad un risultato utile. Si può affermare però che diminuendo il learning rate, il numero di cicli necessari a raggiungere la convergenza si alza in un caso semplice come questo. Per esempio se prendiamo il primo caso analizzato, ed impostiamo il learning rate a 0.01 otteniamo la convergenza a 3397 iterazioni invece di 61.

Sommario

Consegna 1
Rete neurale 1
Apprendimento Hebbiano 2
Problematiche 2
Valori di input 2
Sistema per il test 3
Codice e algoritmi 3
Compilazione 3
Analisi del sistema 4
Convergenza 4