ADDESTRAMENTO DI UNA RETE NEURALE ED EVENTUALI CONSIDERAZIONI PRESTAZIONALI.

(a cura di Buttolo Marco).

L'algoritmo più utilizzato per addestrare una rete neurale è l'algoritmo di **back-propagation**. In sostanza la rete neurale viene addestrata con dei dati numerici. Inizialmente i pesi sinaptici vengono presi in modo casuale. La fase di addestramento avviene misurando dopo ogni ciclo di training (**epoca)** l'errore commesso dalla rete che per definizione è:

$$e(n) = y^{0}(n) - y(n)$$

Si noti che il sistema complessivo è a tempo discreto (t=n). Il valore istantaneo dell'energia dell'errore viene così definito:

$$\varepsilon(n) = \frac{1}{2} e_j^2(n)$$

Quest'ultima relazione vale se la rete neurale in questione ha un solo neurone nello strato di output. Viceversa se la rete neurale possiede più neuroni nello strato di output possiamo invece scrivere:

$$\varepsilon(n) = \frac{1}{N} \sum_{i=1}^{n} \varepsilon(n)$$

dove N è il numero totale di esempi presentati alla rete. La formula può essere considerata come una possibile misura della performance del sistema complessivo. Chiamiamo **pattern** l'insieme di tutti gli esempi di input, mentre chiamiamo epoca ogni singola iterazione dell'algoritmo di apprendimento. Abbiamo definito l'errore come la distanza tra il valore di uscita desiderato ed il valore di uscita effettivamente fornito dalla rete. Tale errore, per ogni epoca, viene reimmesso nella rete retropropagandolo all'indietro fino ai neuroni di ingresso. Durante questa propagazione tale errore va a modificare i vari pesi presenti sulle sinapsi. Tale modifica avviene secondo una relazione matematica del seguente tipo:

correzione del peso = parametro di apprendimento*gradiente locale*valore originale di input;

Quindi nell'algoritmo di propagazione all'indietro dell'errore ci sono sostanzialmente due passi fondamentali:

1. un passo relativo alla propagazione in avanti. Questa fase permette di ottenere un'uscita tale che:

$$y_j(n) = \varphi(v_j(n))$$

dove $v_j(n)$ è il potenziale di azione per il neurone J al tempo 'n'. Esso può essere così definito:

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n).y_j(n)($$

dove 'm' è il numero totale di input applicati al neurone 'j'.

2. un passo di propagazione a ritroso. In particolare la computazione di δ per ogni neurone comporta la conoscenza della derivata della funzione di attivazione $\varphi(.)$. Perché questa derivata effettivamente esisti è necessario che tale funzione di attivazione sia continua. Siccome è richiesta anche la non linearità per la funzione di attivazione di un neurone computazionale, abbiamo che un'ipotetica funzione non lineare e continua (e quindi differenziabile) è proprio la funzione sigmoidale. Tra l'altro l'ampiezza dell'uscita, utilizzando tale funzione, si mantiene nell'intervallo [0,1], perfettamente in accordo con il fatto che l'uscita della rete neurale è un valore numerico normalizzato tra zero e uno.

Quindi possiamo scrivere formalmente in questo modo:

$$\varphi_j(v_j(n)) = \frac{1}{1+e^{-(a v_j(n))}}$$

con a > 0 ed inoltre: $-\infty < v_j(n) < \infty$

In accordo con la non linearità si ha: $0 \leq y_j(n) \leq 1$. Proviamo ora a differenziare l'equazione precedente ottenendo:

$$\varphi'_j(v_j(n)) = \frac{a e^{-(a v_j(n))}}{(1+ e^{(-a v_j(n))})^2}$$

Ricordandosi che: $y_j(n) = \varphi_j(v_j(n))$. Ora eliminiamo il termine esponenziale dall'espressione, e otteniamo:

$$\varphi'_j(v_j(n)) = a\, y_j(n)(1 - y_j(n))$$

Quest'ultima relazione vale in particolare se il neurone 'j' è un neurone dello strato nascosto. Se il neurone 'j' è un neurone dello strato di output, invece si ottiene:

$$y_j(n) = o_j(n)$$

dove $o_j(n)$ è il j-esimo elemento del vettore di output. Questa uscita è confrontata con l'uscita desiderata, ottenendo così il segnale di errore. Quindi ora esprimiamo il gradiente locale per il singolo neurone 'J' come:

$$\delta_j(n) = e_j(n) \cdot \varphi'_j(v_j(n)) = a(y^{\circ}_j(n) - o_j(n)) \cdot o_j(n)(1 - o_j(n))$$

Quest'ultima relazione vale se il nodo 'J' è un nodo dello strato di output. Se il nodo 'J' è un nodo dello strato nascosto si ha:

$$\delta_j(n) = \varphi'_j(v_j(n)) \cdot \sum_k \delta_k(n) \cdot w_{kj}(n)$$

$$\Downarrow$$

$$\delta_j(n) = a \cdot y_j(n)(1 - y_j(n)) \cdot \sum_k \delta_k(n) \cdot w_{kj}(n)$$

Si noti dall'equazione che la derivata $\varphi'_j(v_j(n))$ ha il suo massimo valore per $y_j(n) = 0.5$, ed il suo minimo per $y_j(n) = 0$ oppure $y_j(n) = 1$. Da tutto ciò segue che il cambiamento dei pesi sinaptici è proporzionale alla precedente derivata. Quindi si ha un cambiamento più significativo per quanto riguarda i pesi sinaptici per quei neuroni dove i segnali di uscita hanno un valore che si attesta più o meno a metà.

Come abbiamo già detto, La fase di addestramento della rete neurale sostanzialmente viene suddivisa in epoche. Durante ogni epoca viene effettivamente propagato all'indietro l'errore di predizione. Tale algoritmo ha un certo costo computazionale sia di tipo spaziale sia di tipo temporale. Infatti, se consideriamo il caso di due neuroni A e B, che sono connessi alla stessa sinapsi e sono entrambi attivi, allora tale sinapsi o sarà "svegliata" o sarà eliminata. L'efficienza della sinapsi è legata alle attività presinaptiche e postsinaptiche presenti nella rete. Consideriamo per esempio il peso sinaptico w_{kj}. Abbiamo:

$x_j \Rightarrow$ segnale presinaptico

$y_j \Rightarrow$ segnale postsinaptico

Allora la variazione del peso sinaptico sarà funzione dei due segnali citati in precedenza:

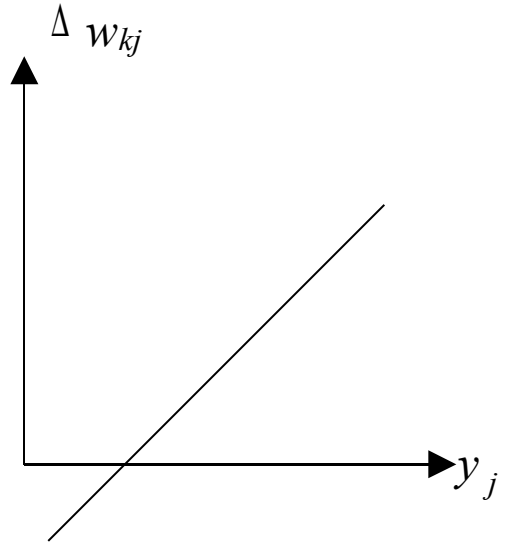
$$\Delta w_{kj} = F(x_k(n), y_j(n)) \qquad \textbf{(7.1)}$$

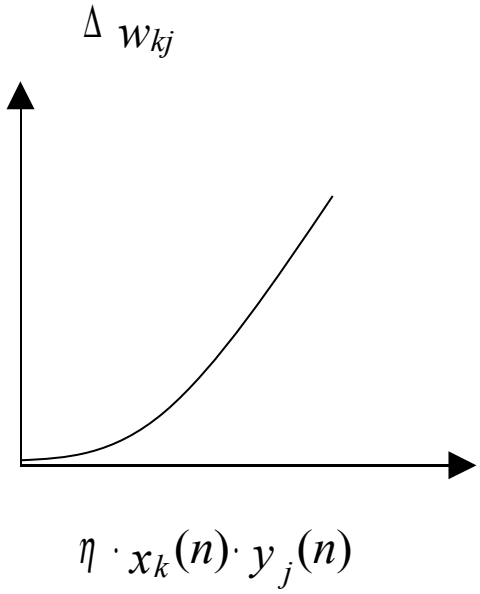
Quindi le regola di produzione delle attività diventa:

$$\Delta w_{kj} = \eta \cdot x_k(n) \cdot y_j(n) \qquad (7.2)$$

Pertanto una crescita di x_k comporta una crescita ovvia di y_j e di conseguenza una crescita che

può anche essere esponenziale di Δw_{kj}. Se per esempio si ha una crescita lineare tra y_j e x_k allora si possono ottenere i seguenti grafici:





In fase di addestramento, si nota che aumentando il numero di neuroni presenti nello strato nascosto della rete, aumenta il tempo di addestramento necessario alla rete. Inoltre aumentando il numero di dati di train aumenta anche il tempo di addestramento della rete stessa. Questo accade perché la rete neurale è un modello data-intensive, e quindi quando si vuole costruire e gestire una rete neurale ottimale, è necessario cercare di rendere ottimale il set di informazioni di train da utilizzare. Per rendere più rapida la fase di addestramento, si può agire sul tasso di apprendimento abbassandolo. Questo però comporta un minor cambiamento dei pesi sinaptici della rete e di conseguenza la predizione risulterà essere meno precisa. Viceversa alzando il tasso di apprendimento, si ha un allungamento del tempo necessario per la fase di addestramento, ma la predizione risulterà essere più accurata. Un buon compromesso potrebbe essere per esempio quello di associare un valore intermedio (0.5) al tasso di apprendimento. In questo modo si ottiene una discreta velocità in fase di addestramento e contemporaneamente una buona predizione. Anche il momento deve essere scelto in modo che renda effettivamente stabile la rete neurale e permettere la reale convergenza dell'algoritmo di apprendimento.