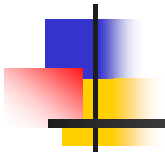
Corso di Percezione Robotica (PRo)



Modulo C.
Percezione Attiva

Richiami di Reti Neurali

9 Marzo 2006

Gioel Asuni
asuni@arts.sssup.it

Il neurone: funzionalità

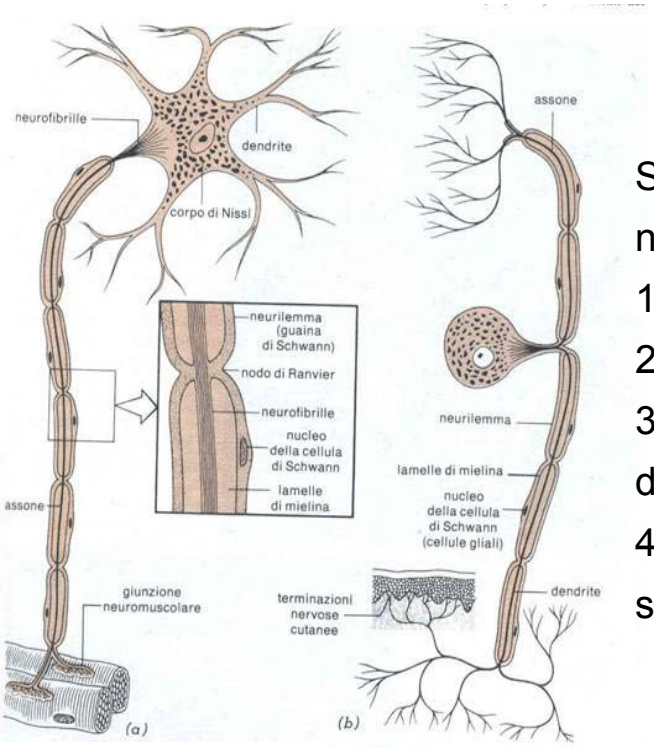
Prima di presentare le reti neurali artificiali è necessario introdurre un minino di terminologia e funzionalità di base dei neuroni del sistema nervoso.

I neuroni sono originati da cellule embrionali dette *neuroblasti* e si differenziano in diverse categorie, tra cui:

- neuroni motori – responsabili dell'attivazione dei muscoli
- neuroni sensitivi – responsabili della percezione sensoriale
- neuroni di elaborazione e proiezione corticale

A parte le differenti funzionalità che svolgono hanno tutti caratteristiche anatomiche comuni.

Il neurone: anatomia



neurone motorio neurone sensitivo

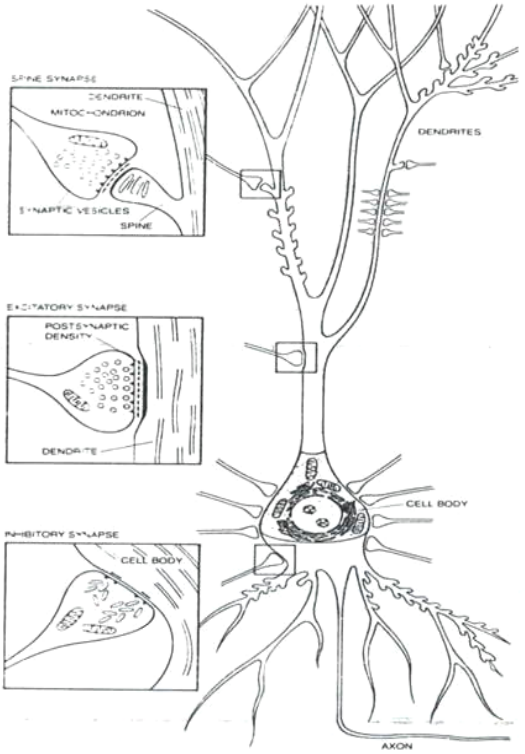
Si può schematizzare il neurone in 4 parti principali:

1. Il corpo cellulare (soma)
2. un prolungamento assonico
3. un insieme di dendriti (strato d'input)
4. un insieme di terminazioni sinaptiche (strato d'output)

Il neurone: anatomia

All'interno del *soma*, immerso nel *neuroplasma*, si trova il nucleo, che dirige l'attività cellulare in termini di funzioni vegetative. Dal soma si originano gli *assoni*, generalmente singoli, lunghi fino a diversi cm., poco ramificati con il compito di trasportare impulsi (efferenti) lontano dal soma. I *dendriti* costituiscono gli ingressi del neurone generalmente multipli, irregolari e molto ramificati. Hanno il compito di condurre gli impulsi (afferenti) verso il *corpo cellulare*.

Il neurone: le sinapsi

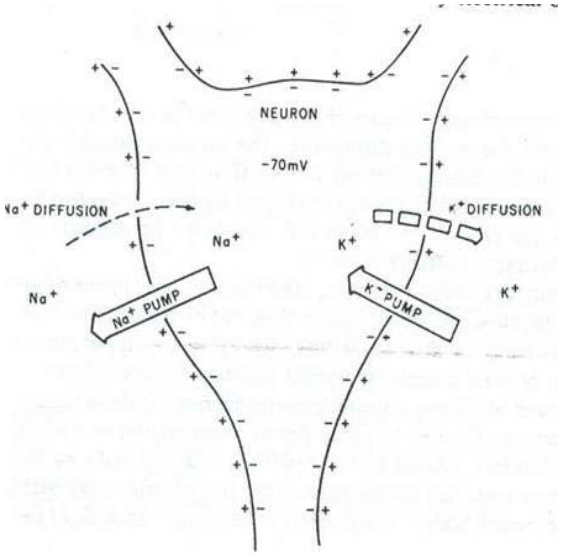


Le terminazioni sinaptiche controllano la trasmissione degli impulsi nervosi da un neurone all'altro tramite rilascio di sostanze chimiche (*neuro-trasmettitori*) contenute all'interno di vescicole presenti nelle terminazioni presinaptiche. I neuro-trasmettitori si diffondono attraverso le fessure sinaptiche e vengono "catturati" da appositi recettori post-sinaptici generando un flusso di corrente verso la membrana che avvolge il soma del neurone target.

Il neurone: inibizione e eccitazione

Il flusso della corrente può essere positivo (aumento del potenziale di membrana) o negativo (diminuzione del potenziale di membrana) a seconda del tipo di neuro-trasmettitore rilasciato. Generalmente, un particolare neurone rilascia un solo tipo di neuro-trasmettitore permettendo cosi di parlare di neuroni *inibitori o eccitatori*. I neuroni postsinaptici invece possono avere contemporaneamente sia recettori inibitori che eccitatori. La somma spazio-temporale degli ingressi inibitori/eccitatori può alterare il potenziale di membrana del neurone post-sinaptico fino a generare una scarica elettrica (*potenziale d'azione*) che viene convogliata dall'assone.

potassio



La membrana che avvolge il soma del neurone è composta da lipidi semipermeabili ad alcune sostanze. In questo modo la membrana genera un *potenziale di membrana* grazie al meccanismo detto della *pompa sodio-potassio*. La membrana risulta quasi impermeabile agli ioni del sodio cosi che gran parte di questi rimangono all'esterno una volta che questi sono stati pompati fuori.

La membrana si mostra invece relativamente permeabile agli ioni potassio, cosi gran parte di questi tornano fuori una volta che sono stati tirati dentro. Questo meccanismo fa si che si generi un potenziale di riposo di circa -70mV.

Il neurone:
alterazione del potenziale di membrana

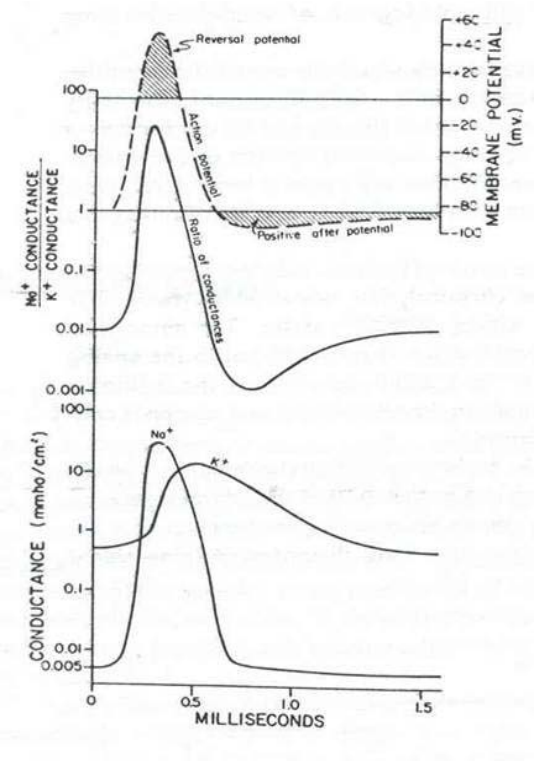
L'effetto dei neuro-trasmettitori è quello di alterare la permeabilità agli ioni della membrana;

• i neuro-trasmettitori eccitatori "aprono" i pori della membrana per gli ioni sodio aumentando la diffusione di questi verso l'interno della cellula, depolarizzando la membrana
• i neuro-trasmettitori inibitori "aprono" i pori della membrana per gli ioni potassio che diffondono fuori in maniera maggiore.

Le correnti negative e positive fluenti nel soma del neurone a livelli di contatto sinaptico dei dendriti e del soma non vengono semplicemente sommate in maniera algebrica; concorrono anche effetti amplificatori/limitatori a livello di sinapsi (*peso sinaptico*), la differenza temporale degli istanti d'arrivo e perfino dalla posizione relativa delle sinapsi sui dendriti e sul corpo cellulare.

Il neurone: generazione dello spike

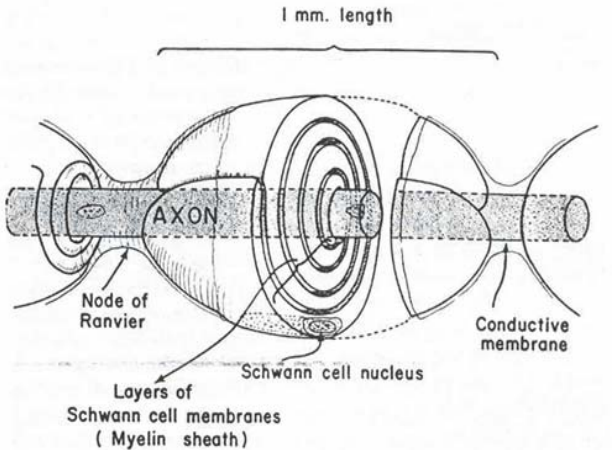
Quando il potenziale di membrana sale al disopra di circa -50mV



le pareti della base dell'assone diventano rapidamente permeabili al sodio che penetra massicciamente all'interno invertendo la polarità della membrana a ca.+50mV, dopo ca.0.5ms il controllo passa di nuovo al meccanismo della pompa sodio-potassio ripolarizzando negativamente la membrana in ca. 2ms. Il risultato è un impulso elettrico che viene propagato nell'assone chiamato potenziale d'azione o spike.

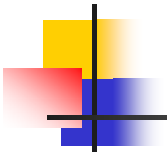
Il neurone: nodi di Ranvier

Il potenziale d'azione si propaga lungo l'assone mediante rigenerazione al livello dei nodi di Ranvier. La rigenerazione avviene grazie alla depolarizzazione.



Ogni volta che il potenziale d'azione compare in un punto dell'assone (nodo) depolarizza il nodo vicino nel quale si genera un analogo potenziale d'azione che in questo modo si propaga lungo l'assone. La propagazione per depolarizzazione può avvenire grazie alla mielina di cui sono ricoperti gli assoni.

La mielina è una sostanza isolante composta dalle cellule di Schwann che permette sia di aumentare la velocità di propagazione (da ca. 30m/s fino a 300m/s) che di non far degradare il segnale.



Il Neurone artificiale

Il neurone artificiale è un modello matematico del neurone biologico. E' l'elemento base necessario per la costruzione di una rete neurale. Il modello matematico non "cattura" appieno le proprietà biologiche e funzionali dei neuroni reali; costituisce una forte semplificazione della realtà mostrando tuttavia delle analogie che ne permette l'utilizzo.

Il Neurone artificiale

Un po' di storia:

- 1943: McCulloch e Pitts presentano un lavoro sulle reti "neurologiche" introducendo un modello formale. Le reti formate con questi neuroni sono in grado di computare funzioni della logica del I ordine.

- 1949: Donald Hebb ipotizzò l'apprendimento biologico come fenomeno sinaptico. Propose una formula matematica per simulare l'apprendimento nelle reti conosciuta come legge di Hebb

Il Neurone artificiale

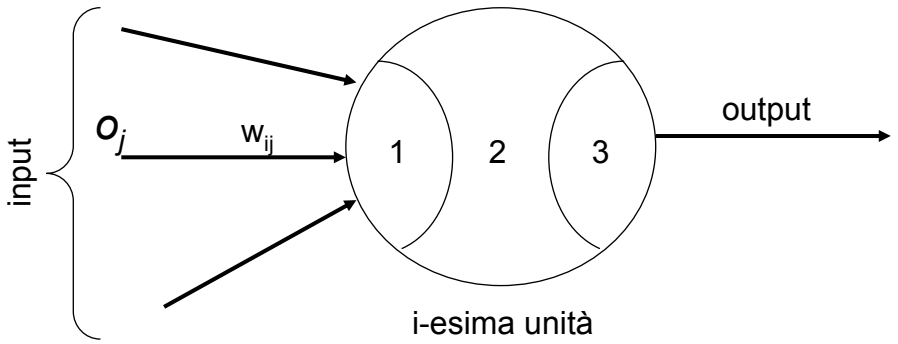
- 1957: Rosenblatt presentò il *perceptron:* una rete in grado di riconoscere immagini. Viene dimostrato un teorema per la convergenza dell'apprendimento del perceptron. Il perceptron riusciva a riconoscere correttamente anche immagini mai viste prime (generalizzazione e capacità d'astrazione). Si ebbe un forte impulso in avanti nell'utilizzo e ricerca in questo settore.

- 1969: Minsky e Papert analizzarono criticamente il perceptron evidenziandone i limiti e l'incapacità di risolvere problemi banali come lo XOR. Le ricerche subirono una forte battuta d'arresto

- anni '80: Rumelhart introdusse il "terzo strato" superando i limiti del perceptron. Si dimostrò formalmente come architetture di reti con almeno 3 strati (input, strato interno/i, output) siano in grado di computare qualsiasi funzione

Il Neurone artificiale

Vediamo adesso il modello matematico del neurone artificiale:



La i-esima unità riceve gli input o_j da strati precedenti o direttamente dall'input. Sulle connessioni sono presenti pesi sinaptici che denotano la "forza" della connessione. L'output viene calcolato secondo tre fasi denotate 1, 2 e 3. Vediamole:

Il Neurone artificiale

1) Regola di propagazione (viene calcolato il *net* dell'input al tempo t+1) $net_i(t+1) = \sum_j w_{ij}(t) o_j(t) \qquad net_i(0) = 0$

2: Regola di attivazione (viene calcolata lo *stato d'attivazione* dell'unità al tempo t+1)

$$a_i(t+1) = F_i(a_i(t), net_i(t+1)) \quad a_i(0) = 0$$

dove F_i =funzione d'attivazione.

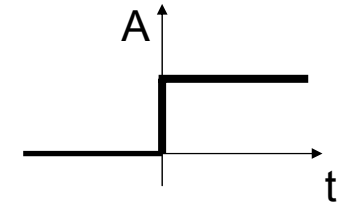
3: viene calcolata l'uscita al tempo t+1

$$o_i(t+1) = f_i(a_i(t+1))$$

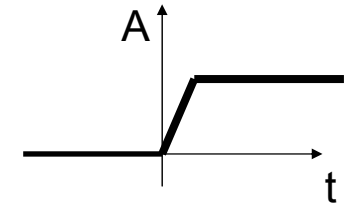
con f_i funzione di trasferimento

Il Neurone artificiale

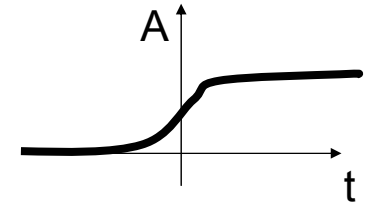
Comunemente le F_i e f_i sono della forma



gradino



soglia



sigmoide

Richiami di reti neurali

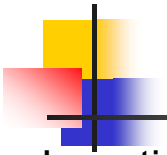
La rete neurale artificiale (ANN-Artificial Neural Network) è una struttura per elaborazione delle informazioni che interconnette neuroni artificiali con cui si tenta di simulare il sistema nervoso del cervello umano.

Una formulazione più specifica dal punto di vista tecnico porta alla definizione data da Kohonen (*)
"struttura per elaborare informazioni basata sui modelli delle funzioni cerebrali, realizzata con un sistema dinamico altamente parallelo, con la topologia di un grafo orientato in cui i neuroni artificiali sono i nodi e le linee di connessione con i relativi pesi fra l'uscita di un neurone e l'ingresso di una altro neurone sono i rami di una rete che assume stati diversi in risposta a segnali di ingresso diversi".
Alternativamente la rete neurale viene anche chiamata "neurocomputer" o "sistema di calcolo massicciamente parallelo a struttura connessionista"

Richiami di reti neurali

Altra definizione di Rete Neurale (Hecht-Nielsen):

Una Rete Neurale è una struttura parallela che processa informazioni distribuite. Tale rete consta di elementi di processazione (PEs o neuroni- che possono avere una memoria locale e che sono in grado di processare localmente informazioni) interconnessi tramite canali (detti connessioni) che trasmettono segnali unidirezionali. Ogni neurone ha una singola connessione di uscita che si dirama in un certo numero di connessioni collaterali; ognuna di questa trasporta lo stesso segnale - il segnale d' uscita del neurone. Questo segnale d' uscita può essere di qualunque tipo matematico. La computazione compiuta all'interno di ciascun neurone può essere definita arbitrariamente con l'unica restrizione che deve essere completamente locale; cioè deve dipendere solo dai valori correnti dei segnali d'ingresso che arrivano al neurone tramite opportune connessioni e dai valori immagazzinati nella memoria locale del neurone.



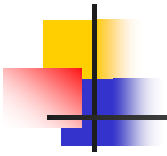
Richiami di reti neurali

Le reti neurali si caratterizzano attraverso almeno i seguenti elementi fondamentali:

L'architettura o topologia della rete, che rappresenta il modo particolare in cui gli strati sono interconnessi e ricevono gli inputs e outputs; la connessione fra due neuroni generici avviene mediante un link detto peso.

La funzione di attivazione e di trasferimento scelta per i neuroni, che in analogia con il neurone biologico rappresenta la modalita' di "sparo" del neurone cioe' di risposta a stimoli esterni. In genere viene scelta la stessa funzione per tutti i neuroni degli strati componenti la rete, ma cio' non e' un vincolo stretto, bensi' una strategia architetturale.

L'algoritmo usato per la fase d'apprendimento.



Richiami di reti neurali

Queste tre caratteristiche possono essere pensate come il livello piu' alto di visione di un modello di rete neurale.
E' bene sottolineare che tali aspetti non possono essere sempre considerati indipendenti fra loro, poiche' per esempio, determinate architetture precludono l'utilizzo di determinati algoritmi d'apprendimento o viceversa.
Inoltre le connessioni tra i neuroni possono essere rappresentate da valori reali positivi o negativi, in tal caso si parlera' di connessioni rispettivamente eccitatorie ed inibitorie; entrambe sono in grado di influenzare il grado di apprendimento di una rete.

Richiami di reti neurali

In base alla configurazione delle connessioni di rete, le diverse architetture possono essere raggruppate in due classi:

•le reti ad alimentazione in avanti (feed-forward) in cui il grafo non contiene spire formate con rami di ritorno all'indietro.

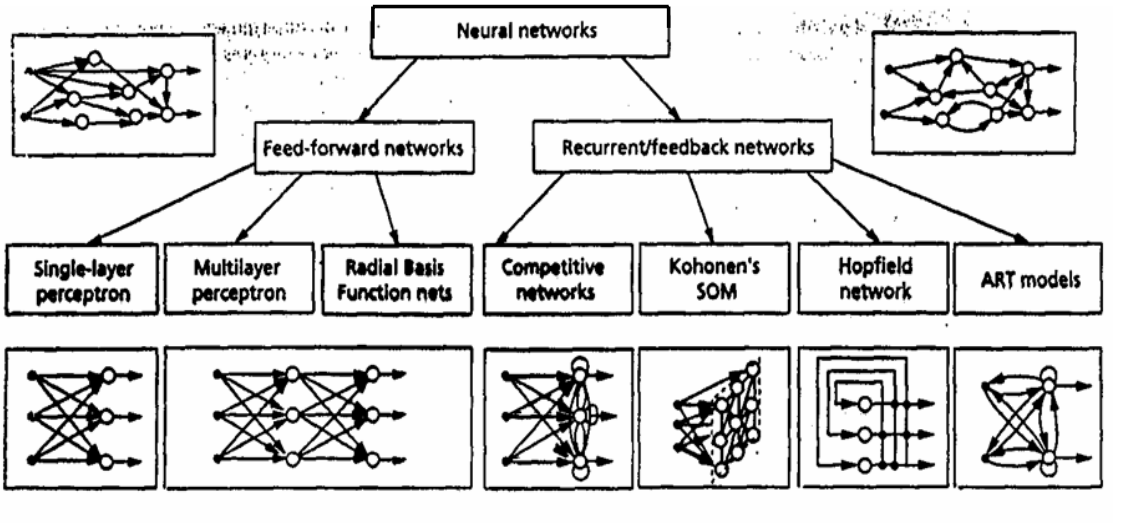
- •singolo strato
- •multistrato

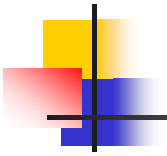
•le reti ricorrenti con reazione (recurrent or feedback) in cui si formano spire di reazione a causa di connessioni orientate all'indietro (verso neuroni dello stesso strato o di strati precedenti).

- •Apprendimento competitivo
- •SOM-Self Organizing Maps
- •Hopfield
- •ART-Adaptive Resonance Theory models

Richiami di reti neurali

Una possibile Tassonomia (Jain 97)



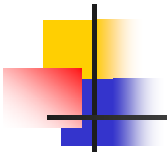


Richiami di reti neurali: apprendimento

Nel contesto delle reti neurali "il processo di apprendimento può essere visto come il problema di aggiornare l'architettura di rete e i pesi delle connessioni cosicché la rete possa compiere efficacemente il suo specifico compito" (A.K. Jain et all. 1996).

Le sue prestazioni migliorano aggiornando progressivamente i pesi e/o modificando l'architettura della rete nel prosieguo del tempo con la presentazione ripetuta di "esempi".

L'abilità che così si ottiene, di imparare automaticamente "per presentazioni di esempi" è ciò che rende di estremo interesse la rete neurale.



Richiami di reti neurali: apprendimento

•***supervised learning***- viene adottato quando si conosce il target che le unità d'uscita devono raggiungere presentando uno specifico ingresso. L'apprendimento consiste nel valutare l'errore commesso da ciascuna unità e quindi agire di conseguenza sul valore dei pesi delle connessioni delle unità per ridurre l'errore nel caso di nuova presentazione dello stesso input.

•***reinforcement learning***- questo modello, che è una variante del modello classico supervisionato, si basa sugli errori fatti nel confrontare per tentativi le configurazioni di ingresso con quelle di uscita in modo da massimizzare un parametro detto "segnale di rinforzo"

• ***unsupervised learning***-l'ambiente esterno non fornisce esempi di configurazioni da apprendere. La rete esplora non i dati d'ingresso ma la struttura sottostante a questi dati: ad esempio calcola le correlazioni fra le configurazioni di ingresso e da esse organizza queste configurazioni in classi e categorie. Allorquando la rete si sintonizza sulle regolarità statistiche dei segnali di entrata, allora essa diventa capace di formare rappresentazioni interne in cui si codificano le caratteristiche tipiche dell'ingresso

Richiami di reti neurali: apprendimento

L'apprendimento, in questo contesto, significa variare opportunamente i pesi sulle connessioni in funzione dell'esperienza (ossia valutando le prestazioni passate).

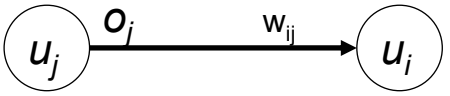
In pratica l'apprendimento fa si che la struttura della rete si modifichi in accordo alla conoscenza che essa elabora.

Generalmente parlando l'apprendimento modifica il modello di connettività della rete secondo:

- sviluppo di nuove connessioni
- perdita delle connessioni esistenti
- modifica della forza tra le connessioni
- aggiunta di nuove unità (algoritmi costruttivi)
- rimozione di unità (Optimal Brain Surgen - OBS)

Richiami di reti neurali: apprendimento

Come esempio di apprendimento consideriamo la legge di Hebb. Hebb afferma che se due unità connesse tra loro mediante contatto sinaptico sono entrambe “fortemente” attive allora la connessione sinaptica viene rafforzata.

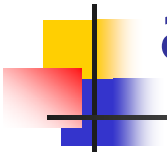


$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}$$

dove

$\Delta w_{ij} = \eta a_i o_j \; con \; 0 < \eta \leq 1$ *tasso di apprendimento*

Richiami di reti neurali: apprendimento



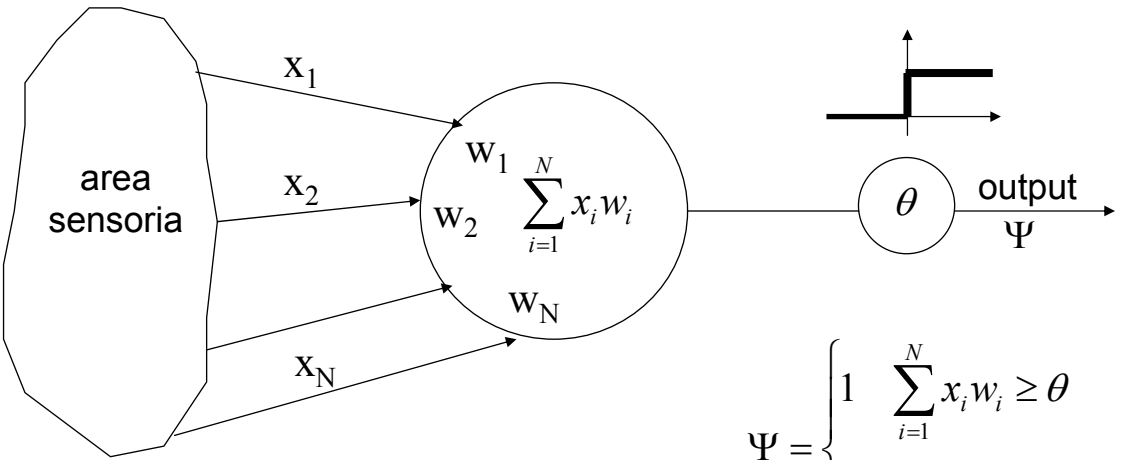
Un sistema che apprende, in questo contesto, è un sistema che “prende” decisioni su un problema in base all’esperienza accumulata.

In questo caso, un sistema siffatto **non** contiene un modello di ragionamento di un esperto umano, o di un sistema esperto secondo regole del tipo IF/THEN/ELSE di un motore inferenziale, l’accento è posto a livello percettivo e sensorio e non cognitivo.

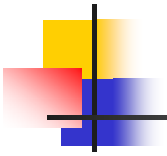
Il “comportamento” delle strutture nervose modellate vuole modellare una conoscenza di tipo percettivo-sensoriale. Questa conoscenza viene generata a partire dalle caratteristiche intrinseche o dalla natura degli stimoli sensoriali a cui la rete è sottoposta. La rete si adatta quindi ad un modello generale dei campioni del problema che gli abbiamo posto. Questo fenomeno è detto *generalizzazione*: applicare la conoscenza acquisita a casi nuovi.

Il Perceptron

Vediamo adesso come è costituita una rete formata da unità tipo perceptron. Lo schema del peceptron è



$$\Psi = \begin{cases} 1 & \sum_{i=1}^{N} x_i w_i \geq \theta \\ 0 & \sum_{i=1}^{N} x_i w_i < \theta \end{cases}$$



Il Perceptron

Una rete con unità siffatte è in grado di riconoscere, o meglio *classificare*, immagini; tuttavia non è in grado di risolvere il problema dello XOR.

Prima di mostrare questo fatto si presenta la procedura di convergenza del perceptron.

Per il perceptron è stato dimostrato che:

se una categoria può essere classificata da un perceptron, allora la procedura di convergenza termina in un numero finito di passi.

Il Perceptron

PROCEDURA DI APPRENDIMENTO

1. Inizializzazione casuale dei pesi e della soglia del nodo d'uscita;
2. Si presenta il pattern d'ingresso $x_i(t)$ e l'uscita desiderata $d(t)$;
3. Si calcola l'uscita $y(t)$

$$y(t) = f_h\left(\sum_{i=1}^{N} x_i(t) w_i(t) - \theta\right)$$

f_h è la funzione a gradino di Heavside

4. Modifica dei pesi
   se $y(t) <> d(t)$ allora $w_i(t+1) = w_i(t) + \eta[d(t) - y(t)]x_i(t)$

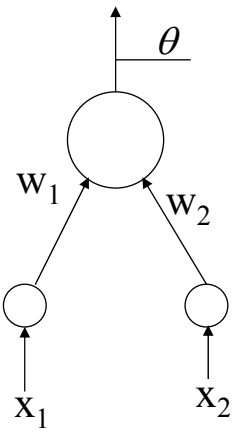
   con $0 < \eta <= 1$ tasso di apprendimento

torna a passo 2

Il Perceptron

Esempio: apprendere la funzione OR

X_1	X_2	OR
0	0	0
0	1	1
1	0	1
1	1	1



partiamo dalla condizione iniziale dei pesi

$w_1=0.1,\ w_2=0.2,\ \theta=0.5$

$t=1,\ x_1=0,\ x_2=0$

$y(1)=f_h(0x0.1+0x0.2-0.5)=$
$f_h(-0.5)=0 \Rightarrow OK$

$t=2,\ x_1=0,\ x_2=1$

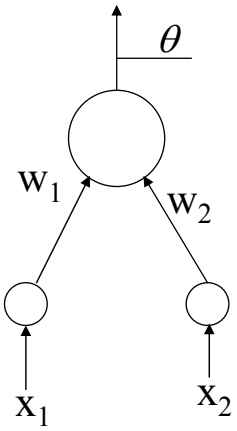
$y(2)=f_h(0x0.1+1x0.2-0.5)=f_h(-0.3)=0$
$\Rightarrow NO \Rightarrow$ *correzione dei pesi*

$w_{1NEW}=w_{1OLD}+[1-0]x0=0.1 \quad w_{2NEW}=w_{2OLD}+[1-0]x1=1.2$

..., ricalcolando y(2) $y(2)=f_h(0x0.1+1x1.2-0.5)=f_h(0.7)=1$
$\Rightarrow OK\ !$

Il Perceptron

IL problema dello XOR:
una rete siffatta, non riesce a risolvere il problema dello XOR

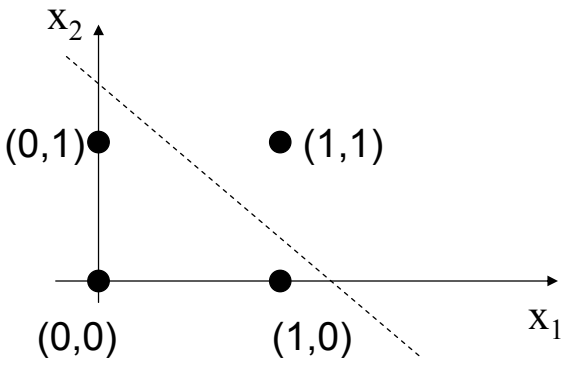


X_1	X_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

le uscite dello XOR sono le *d(t)* desiderate. Il perceptron non riesce a dare risposte corrette nel caso di (0,0) oppure (1,1). Come si vede dalla tabella dello XOR le possibili uscite, e quindi classi, sono solo due: 0 e 1. In pratica e come si avessimo una retta che divide il piano in due semipiani

Il Perceptron

Se mappiamo sul piano le variabili dello XOR otteniamo



che non esiste una retta la quali separi nella stessa classe le coppie (0,0) e (1,1). Questo perché guardando bene la funzione del perceptron

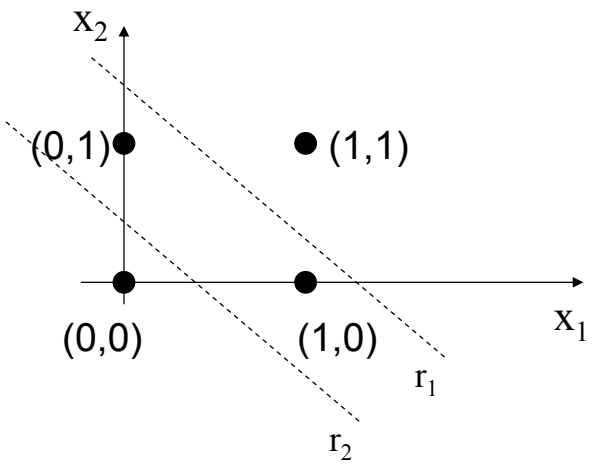
$$\theta = w_1 x_1 + w_2 x_2$$

Ci si accorge che *non è altro che l'espressione di una retta* !!!!

Poichè non potrà mai esistere una retta che risolva il problema dello XOR (è infatti un problema *non linearmente separabile) ne segue che il perceptron non potrà mai apprendere questa funzione.*

Il Perceptron multistrato

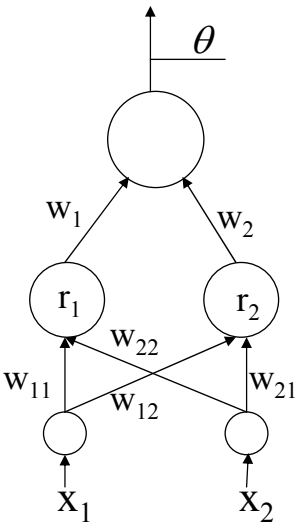
Abbiamo visto come un perceptron non riesce a risolvere il problema dello XOR in quanto opera come un discriminante lineare. La soluzione si avrebbe utilizzando un'altra retta che divida ulteriormente il piano, come in figura.

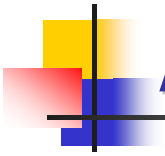


In questo modo, le coppie (1,1) e (0,0) possono essere classificate correttamente una volta riferitosi alla retta giusta. Ad esempio (1,1) è nel semipiano positivo individuato dalla retta *r1*. Il problema è quindi di individuare, o meglio apprendere l'eq. della retta che separa la classe cercata e quindi apprendere a che piano appartiene la coppia desiderata.

Il Perceptron multistrato

Ricordandosi che l'unità d'uscita del perceptron rappresentava l'eq. di una retta si può pensare di avere *prima* dell'unità d'uscita due unità *nascoste* (*hidden*) che rappresentino le rette volute. Infatti, in questo modo il peceptron, modificato, introducendo il terzo strato riesce a risolvere il problema dello XOR.

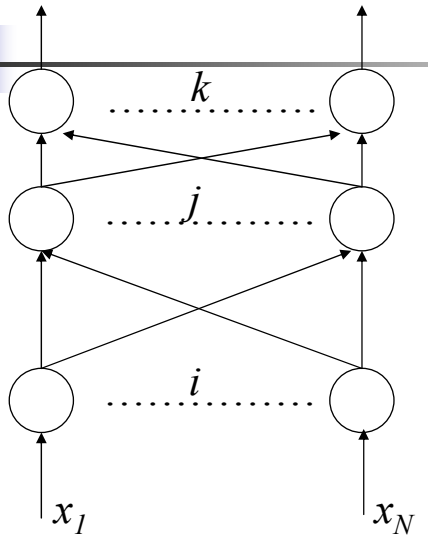




Algoritmo Back-Propogation(BP)

Nelle reti associative a due strati si ha una mappatura diretta tra patterns d'ingresso e d'uscita. Se la funzione da apprendere è "semplice" (linearmente separabile) allora le reti in questione riescono a computare (apprendere) la funzione obiettivo. In caso che la funzione sia più "complicata" (problema non linearmente separabile) è necessario introdurre il terzo strato in modo tale che la rete crei una rappresentazione interna del problema in modo da poterlo risolvere. L'algoritmo di BP, e sue varianti, è quello più diffuso per realizzare l'apprendimento in queste tipologia di reti, anche se soffre di alcuni limiti che vedremo nel proseguo, tuttavia il suo l'utilizzo permette di risolvere con successo una diversa casistica di problemi .

Algoritmo BP



M unità di output, *1<=k<=M*

L unità interne, *1<=j<=L*

N unità d'ingresso, *1<=i<=N*

La rete è totalmente connessa: ogni unità dello strato precedente proietta connessioni su ogni unità dello strato superiore.

Algoritmo BP

Si definiscono le seguenti quantità:

$$net^h_{p_j} = \sum_{i=1}^{N} w^h_{ji} x_{p_i}$$

net calcolato per la *j-esima* unità hidden nascosta dopo l'applicazione del pattern ***p*** all'ingresso

$$i_{p_j} = f^h_j(net^h_{p_j})$$

uscita calcolata dalla *j-esima* unità hidden

$$net^o_{p_k} = \sum_{j=1}^{L} w^o_{kj} i_{p_j}$$

net calcolato per la *k-esima* unità d'uscita dopo l'applicazione del pattern ***p*** all'ingresso

$$o_{p_k} = f^o_k(net^o_{p_k})$$

uscita calcolata dalla *k-esima* unità d'uscita

Algoritmo BP

$\delta_{p_k} = (y_{p_k} - o_{p_k})$

errore commesso dalla *k-esima* unità d'uscita. y_{pk} è l'uscita desiderata

$E_p = \frac{1}{2}\sum_{k=1}^{M} \delta^2{}_{p_k}$

errore quadratico medio totale commesso dallo strato d'uscita per il pattern ***p***

$E = \sum_{p=1}^{P} E_p$

errore totale su tutti i patterns del training-set

vogliamo attuare una tecnica per ridurre l'errore E mediante aggiornamento dei pesi. Si attua una discesa di gradiente derivando E_p rispetto ai pesi che lo hanno generato

$$\nabla E_p = \left[\frac{\partial E_p}{\partial w^o_{k1}}, \ldots, \frac{\partial E_p}{\partial w^o_{kL}} \right]$$

Algoritmo BP

Per mantenere la trattazione più semplice, si calcolano separatamente le componenti di ∇E_p

$$E_p = \frac{1}{2}\sum_{k=1}^{M} \delta^2{}_{p\,k} = \frac{1}{2}\sum_{k=1}^{M} \left(y_{p_k} - o_{p_k}\right)^2 = \frac{1}{2}\sum_{k=1}^{M} \left(y_{p_k} - f_k^o(net_{p_k}^o)\right)^2$$

fissato *k* e *j*

$$\frac{\partial E_p}{\partial w_{kj}^o} = \frac{\partial\left(\frac{1}{2}(y_{p_k} - f_k^o(net_{p_k}^o))^2\right)}{\partial w_{kj}^o} = (y_{p_k} - o_{p_k})\frac{\partial(y_{p_k} - f_k^o(net_{p_k}^o))}{\partial(net_{p_k}^o)}\frac{\partial(net_{p_k}^o)}{\partial w_{kj}^o}$$

$$= -(y_{p_k} - o_{p_k})(f_k^o(net_{p_k}^o))^I \frac{\partial(net_{p_k}^o)}{\partial w_{kj}^o} = -(y_{p_k} - o_{p_k})(f_k^o(net_{p_k}^o))^I \frac{\partial(w_{kj}^o i_{p_j})}{\partial w_{kj}^o} =$$

$$= -(y_{p_k} - o_{p_k})(f_k^o(net_{p_k}^o))^I i_{p_j}$$

Algoritmo BP

Scegliendo opportunamente la $f_k^o(\circ)$ come:

a) funzione identità

$$f_k^o(net_{p_k}^o) = net_{p_k}^o \Rightarrow (f_k^o(net_{p_k}^o))^I = 1$$

b) funzione sigmoide

$$f_k^o(net_{p_k}^o) = \frac{1}{1+e^{-net_{pk}^o}} \Rightarrow (f_k^o(net_{p_k}^o))^I = f_k^o(\circ)(1-f_k^o(\circ))$$

si ottiene la legge della variazione dei pesi per lo strato d'uscita come riportato di seguito

Algoritmo BP

caso a)

$$\Delta w_{kj}^{o} = \eta (y_{p_k} - o_{p_k}) i_{p_j}$$

η tasso di apprendimento

caso b)

$$\Delta w_{kj}^{o} = \eta (y_{p_k} - o_{p_k}) o_{p_k} (1 - o_{p_k}) i_{p_j}$$

osservando la forma di queste relazioni si nota che sono in una forma più generale della variazione dei pesi introdotta da Hebb (regola Δ). Questo è un risultato importante:

La regola Δ generalizzata implementa la tecnica di discesa del gradiente

Algoritmo BP

AGGIORNAMENTO PESI STRATO NASCOSTO

Le formule introdotte fin qui hanno permesso di aggiornare i pesi dello strato d'uscita in virtù di una comparazione tra l'uscita calcolata e l'uscita desiderata. Il problema nasce quando si deve valutare l'errore commesso dallo strato nascosto.

Infatti per tali unità si può conoscere il valore di output attuale (basta calcolarla) ma non si ha modo di conoscere quale deve essere l'uscita desiderata.

Tuttavia, in maniera intuitiva, si comprende che in qualche modo l'errore E_p deve essere collegato con i valori d'uscita dello strato nascosto

Algoritmo BP

riprendendo

$$E_p = \frac{1}{2}\sum_{k=1}^{M} \delta^2{}_{p_k} = \frac{1}{2}\sum_{k=1}^{M} \left(y_{p_k} - o_{p_k}\right)^2 = \frac{1}{2}\sum_{k=1}^{M} \left(y_{p_k} - f_k^o(net_{p_k}^o)\right)^2$$

fissato *i* e *j*

$$\frac{\partial E_p}{\partial w_{ji}^h} = -\sum_k (y_{p_k} - f_k^o(net_{p_k}^o)) \frac{\partial (f_k^o(net_{p_k}^o))}{\partial w_{ji}^h} =$$

$$= -\sum_k (y_{p_k} - o_{p_k})) \frac{\partial (f_k^o(net_{p_k}^o))}{\partial (net_{p_k}^o)} \frac{\partial (net_{p_k}^o)}{\partial i_{p_j}} \frac{\partial i_{p_j}}{\partial (net_{p_j}^h)} \frac{\partial (net_{p_j}^h)}{\partial w_{ji}^h}$$

sviluppando ogni termine

Algoritmo BP

$$\frac{\partial(f_k^o(net_{p_k}^o))}{\partial(net_{p_k}^o)} = (f_k^o(net_{p_k}^o))^I$$

$$\frac{\partial(net_{p_k}^o)}{\partial i_{p_j}} = \frac{\partial(\sum_{j=1}^{L} w_{kj}^o i_{p_j})}{\partial i_{p_j}} = w_{kj}^o$$

$$\frac{\partial i_{p_j}}{\partial(net_{p_j}^h)} = \frac{\partial(f_j^h(net_{p_j}^h))}{\partial(net_{p_j}^h)} = (f_j^h(net_{p_j}^h))^I$$

$$\frac{\partial(net_{p_j}^h)}{\partial w_{ji}^h} = \frac{\partial(\sum_{i=1}^{N} w_{ji}^h x_{p_i})}{\partial w_{ji}^h} = x_{p_i}$$

riassumendo ...

Algoritmo BP

$$\frac{\partial E_p}{\partial w_{ji}^h} = -\sum_k (y_{p_k} - o_{p_k}))(f_k^o(net_{p_k}^o))^I w_{kj}^o (f_j^h(net_{p_j}^h))^I x_{p_i}$$

quindi ...

$$\Delta w_{ji}^h = \eta \sum_k (y_{p_k} - o_{p_k}))(f_k^o(net_{p_k}^o))^I w_{kj}^o (f_j^h(net_{p_j}^h))^I x_{p_i}$$

si noti che l'aggiornamento del peso di una connessione interna dipende dal contributo di tutte le unità d'uscita. Inoltre w_{kj}^o è il valore del peso della connessione *j->k* prima dell'aggiornamento.

Algoritmo BP

I passi dell'algoritmo

1. Applicare il vettore d'ingresso x_p per il pattern p
2. Calcolare $net^h_{p_j} = \sum_{i=1}^{N} w^h_{ji} x_{p_i} \quad i_{p_j} = f^h_j(net^h_{p_j})$
3. Calcolare per ogni unità d'output $net^o_{p_k} = \sum_{j=1}^{L} w^o_{kj} i_{p_j} \quad o_{p_k} = f^o_k(net^o_{p_k})$
4. Calcolare l'errore commesso da ogni unità d'output

$$\delta^o_{p_k} = (y_{p_k} - o_{p_k})(f^o_k(net^o_{p_k}))^I$$

5. Calcolare l'errore commesso da ogni unità nascosta

$$\delta^h_{p_j} = \sum_k \delta^o_{p_k} w^o_{kj} (f^h_j(net^h_{p_j}))^I$$

Algoritmo BP

6. Aggiornamento dei pesi dello strato d'uscita

$$w_{kj}^{o}(t+1) = w_{kj}^{o}(t) + \eta \delta_{p_k}^{o} i_{p_j}$$

7. Aggiornamento dei pesi dello strato nascosto

$$w_{ji}^{h}(t+1) = w_{ji}^{h}(t) + \eta \delta_{p_j}^{h} x_{p_i}$$

8. Calcolare

$$E_p = \frac{1}{2} \sum_{k=1}^{M} \left(y_{p_k} - o_{p_k} \right)^2$$

9. Se E_p è accettabile (sotto una soglia predefinita) allora si può proseguire l'addestramento presentando altri patterns, altrimenti si reitera dal punto 1 mantenendo lo stesso pattern.

Algoritmo BP

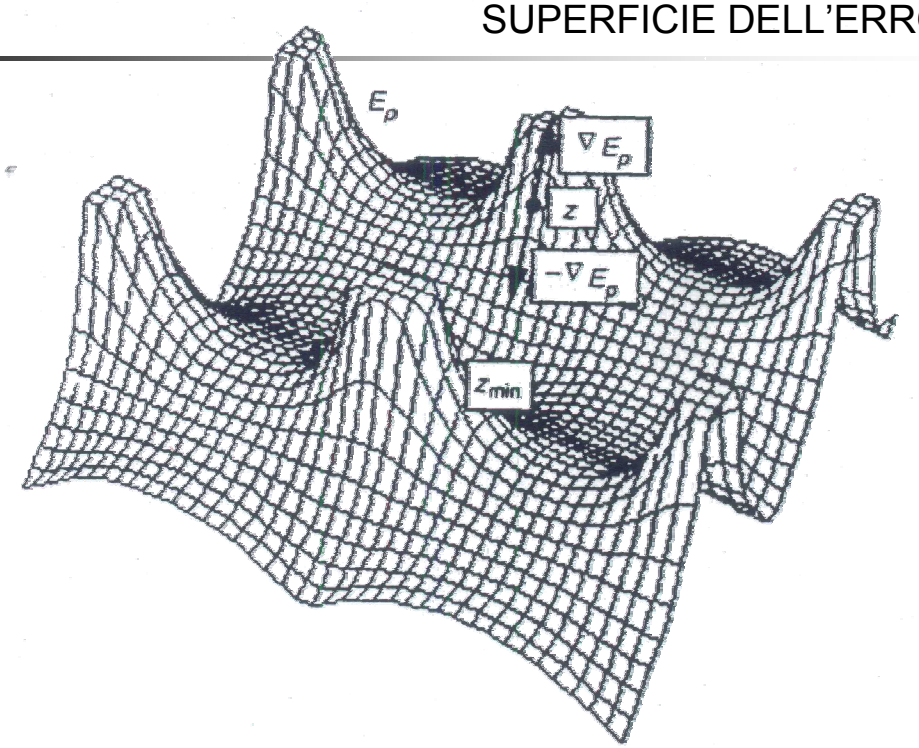
CONSIDERAZIONI

1. *E' necessario stabilire a priori il numero delle unità nascoste*: un *sovradimensionamento* delle unità oltre a apportare un aggravio computazionale comporta una cattiva generalizzazione. Al contrario, un *sottodimensionamento* del numero delle unità può far si di non riuscire a raggiungere la soglia di accettabilità dell'errore. La determinazione del "giusto" numero delle unità è lasciata alla "sensibilità" del progettista riguardo la natura del problema.

2. Nelle reti a due strati con funzioni d'attivazione e di uscita lineari la superficie dell'errore risulta essere un paraboloide con un unico punto di minimo. Con l'introduzione del terzo strato e con funzioni semi-lineari, la superficie dell'errore assume la forma di una curva con diversi picchi e minimi locali

Algoritmo BP

SUPERFICIE DELL'ERRORE



Algoritmo BP

Quello che accade e che, a partire dalla configurazione iniziale dei pesi, la tecnica di discesa del gradiente potrebbe dirigere l'apprendimento (il vettore peso) verso un minimo locale a cui potrebbe non corrispondere un errore totale accettabile. Per risolvere questo problema, si potrebbe cercare di "scuotere" la superficie in modo da far saltare fuori da un minimo locale il vettore peso. In formule, ad esempio per lo strato d'uscita

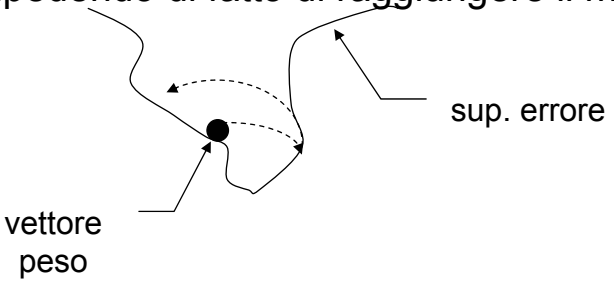
$$w^o_{kj}(t+1) = \alpha w^o_{kj}(t) + \eta \delta^o_{p_k} i_{p_j}$$

dove α (tipico valore 0.9) è detto *momento.*

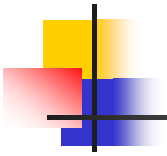
3. *Lentezza dell'apprendimento:* in parte dovuta alla complessità dei calcoli ed in parte alla possibilità dell'instaurarsi di cicli per l'aggiornamento dei pesi tra lo strato d'uscita e quello nascosto in caso che l'errore non scenda sotto la soglia d'accettabilità. Per velocizzare l'apprendimento si può lavorare sul valore del tasso d'apprendimento.

Algoritmo BP

Grandi valori, circa 1, potrebbero tuttavia innescare fenomeni di oscillazione impedendo di fatto di raggiungere il minimo.



Il raggiungimento della soglia di accettabilità dell'errore è principalmente legato, tuttavia, alla configurazione iniziale dei pesi in quanto questi determinano la "forma" della superficie dell'errore. Per cercare di ottenere superfici "lisce" con pochi minimi locali sono state approntate diverse tecniche tra le quali le più promettenti per risultati sono da attribuirsi agli *algoritmi genetici.*



Algoritmo BP

I pesi della rete sono considerati come una “popolazione” che può evolvere nel tempo secondo tecniche evolutive attraverso *riproduzione* (ricombinazione dei geni) *e mutazione*. Gli algoritmi genetici implementano operatori di valutazione di “adattamento all’ambiente” (*fitness*) che di volta in volta selezionano i nuovi pesi generati che meglio si adattano alle caratteristiche del problema in termini di riduzione dell’errore.

Le mappe di Kohonen - SOFM

Le mappe di Kohonen sono modelli auto-organizzanti di reti neurali dette SOFM (Self Organizing Feature Map) in quanto le unità tendono a disporsi autonomamente, nello spazio dei pesi, in maniera di preservare la disposizione “spaziale” degli stimoli di input.

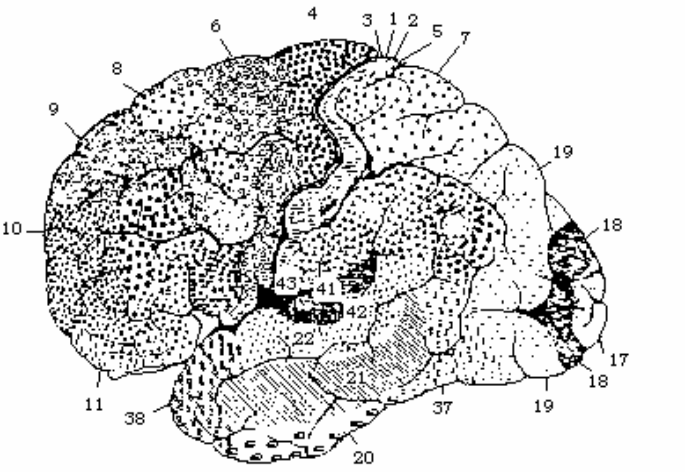
L’idea di Kohonen prende spunto dall’osservazione che a livello della corteccia cerebrale appaiono delle popolazioni di neuroni, organizzati topograficamente, in modo da formare mappe citoarchitettoniche ben definite, in accordo a differenti modalità sensorie (omuncolo senso-motorio).

Ad esempio, i diversi input sensori (motori, somatosensoriali, visivi, uditivi, ecc.) sono mappati nelle corrispondenti aree cerebrali in maniera ordinata, come mostra la figura seguente

Le mappe di Kohonen - SOFM

Aree	*Denominazione*
4, 6	area motoria (area 4), area premotoria (area 6).
8	campi visivi frontali
3,1, 2	corteccia somatosensoriale
17,1, 19	corteccia visiva
41, 42	corteccia uditiva

Alcune aree della corteccia



Un esempio: nella corteccia uditiva esiste la mappa *tonotopica*, in cui l'ordine spaziale delle risposte delle cellule corrisponde al picco o frequenza acustica dei toni avvertiti. La mappa delle frequenze acustiche sulla corteccia uditiva è perfettamente ordinata, in scala logaritimica, rispetto alle frequenze.

Le mappe di Kohonen - SOFM

Queste mappe *non sono interamente* predeterminate geneticamente: piuttosto sono organizzate durante i primi tempi dello sviluppo del sistema nervoso. Come questo avvenga non è stato ancora chiaramente appurato ma sono state postulate diverse ipotesi:

1. La struttura target (post-sinapitca) possiede degli indirizzi (ad esempio, segnali chimici) che sono attivamente cercati dagli assoni che si sviluppano verso l'interno.
2. La struttura, all'inizio (struttura target senza alcuna informazione) si auto organizza con l'uso di regole di apprendimento e interazioni del sistema.
3. Gli assoni, man mano che si sviluppano, mantengono fisicamente le relazioni di vicinanza, arrivando così alla struttura target già topograficamente arrangiati.
4. Gli assoni si sviluppano in una sequenza temporale topograficamente organizzata e si connettono alla struttura target che viene generata nel confronto temporale.

Le mappe di Kohonen - SOFM

Ognuna di queste ipotesi ha dati che le avvalorano ed anzi, si può supporre che siano coinvolti, in questo processo, meccanismi multipli. Le SOFM mostrano di possedere le proprietà appena descritte.

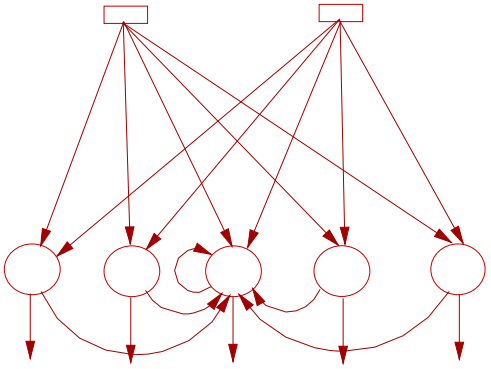
Questo risulta possibile grazie all'algoritmo di apprendimento di tipo *competitivo* con il quale vengono addestrate.

Le unità della mappa competono tra loro in modo che una sola, o un solo gruppo, risulti massimamente attiva in seguito all'applicazione di un particolare stimolo.

L'unità vincente, così determinata, è del tipo "winner-take-all": solo questa, o il gruppo a cui appartiene, infatti, modifica le proprie connessioni sinaptiche per rispondere più adeguatamente allo stimolo, che le ha permesso di vincere la competizione, una volta che questi venga nuovamente presentato.

Le mappe di Kohonen - SOFM

Un modo per indurre una competizione del tipo “winner-take-all” è quello di introdurre delle connessioni di feedback laterale tra le unità.



l’input viene applicato e una unità vince la competizione

sono mostrate solo le connessioni laterali per l’unità centrale che ha vinto la competizione.

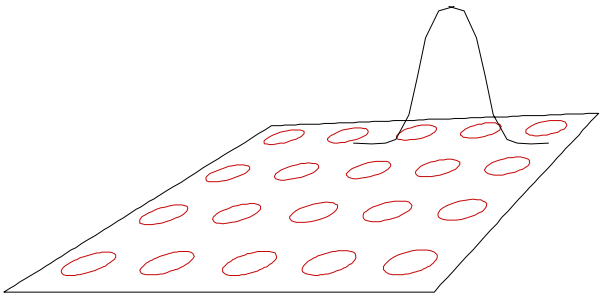
Seguendo motivazioni supportate dalla biologia, l’unità che vince la competizione su un particolare stimolo induce un feedback laterale alle altre unità; la forza di questo feedback è proporzionale alla distanza laterale fra l’unità vincente e le altre.

Le mappe di Kohonen - SOFM

In seguito a queste considerazioni possiamo focalizzare due importanti caratteristiche delle SOFM:

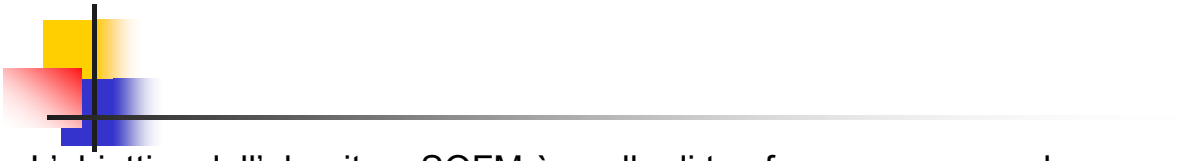
1. la rete tende a concentrare la sua attività in cluster locali chiamati “zone di attività” (*activity bubbles*);

2. la posizione delle “bolle” d’attività sono determinate dalla natura degli stimoli in input; infatti la bolla ha il picco sulla cellula che meglio risponde allo stimolo applicato.



zona d’attività per una mappa disposte su un piano.

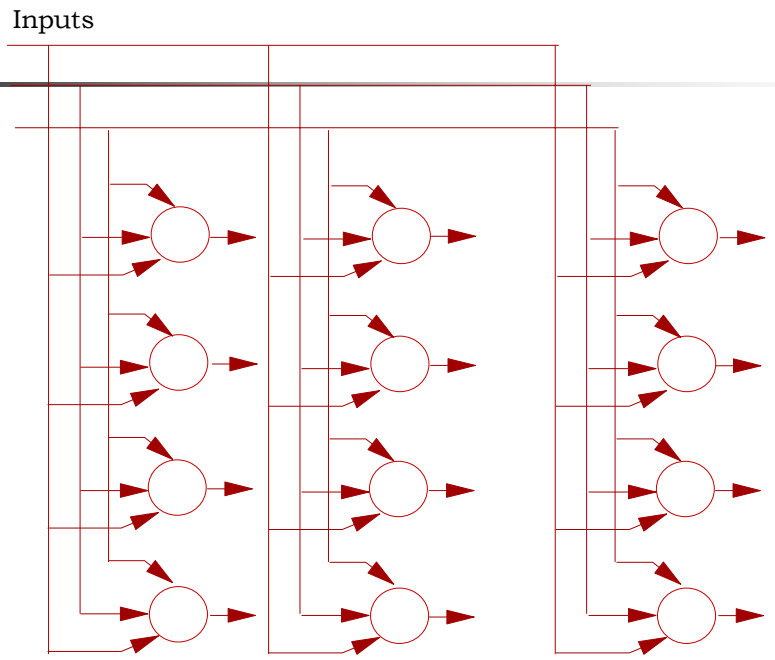
Le mappe di Kohonen - SOFM



L'obiettivo dell'algoritmo SOFM è quello di trasformare un segnale entrante, di dimensione arbitraria, nell'attivazione di una singola unità disposta su una mappa (mono, bidimensionale, ecc.) ed eseguire questa trasformazione in maniera adattiva e topologicamente ordinata. Gli elementi principali di una rete neurale addestrata con questo algoritmo sono:

- Un reticolo, comunemente, ad una o due dimensioni di unità che calcolino semplici funzioni discriminanti degli stimoli in ingresso
- Un meccanismo che selezioni l'unità con il valore di discriminazione più alto
- Un meccanismo per attivare simultaneamente il neurone selezionato e i suoi "vicini"
- Un processo adattivo che permetta ai neuroni attivati di accrescere il loro valore della funzione di discriminazione in relazione ai segnali in input.

Le mappe di Kohonen - SOFM



Una SOFM bidimensionale. Notare che tutte le unità ricevono afferenze da tutte le componenti dell'input.

Le mappe di Kohonen - SOFM

input $\mathbf{x} = [x_1, x_2, \ldots, x_p]$ *p* è il num. delle componenti del vettore d'ingresso

$\mathbf{w}_j = [w_{j1}, w_{j2}, \ldots, w_{jp}]$ vettore dei pesi $j = 1,2,\ldots,N$

N numero dei neuroni

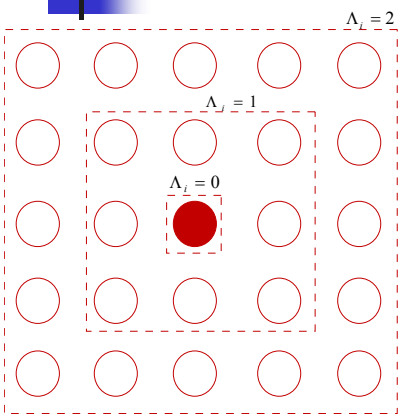
calcolo dell'unità winner:

$$i(\mathbf{x}) = \arg_j \min \left\| \mathbf{x} - \mathbf{w}_j \right\|, j = 1,2,\ldots,N$$

Si seleziona quell'unità a cui e associato il vettore peso che minimizza la norma euclidea con il vettore d'ingresso.

Le mappe di Kohonen - SOFM

$\Lambda_{i(x)}(n)$



L'unità in neretto è quella che ha vinto la competizione con il suo intorno al trascorrere del tempo.

Definisce l'ampiezza dell'intorno, al tempo discreto n, che esprime la vicinanza topologica attorno al winner. All'inizio risulta essere grande abbastanza in modo che tutte le unità della mappa risultino "vicine" a in seguito, con il trascorerre del tempo (al crescere di n), si riduce man mano fino ad assumere il valore unitario. Il comportamento è equivalente ad usare inizialmente un feedback laterale fortemente positivo e in seguito permettere un avanzamento di quello negativo in modo tale che l'avanzamento dell'apprendimento consista nel far si che una unità si specializzi a rappresentare un certo input in modo da evitare ambiguità classificatorie.

Le mappe di Kohonen - SOFM

Una volta che il pattern è stato applicato e si sia calcolato il winner si procede ad aggiustare i pesi delle unità che appartengono all'intorno corrente del neurone che ha vinto la competizione.

Per far ciò si definisce:

$$y_j = \begin{cases} 1 & \text{se neurone j è attivo } (j \in \Lambda_{i(x)}) \\ 0 & \text{se neurone j è inattivo } (j \notin \Lambda_{i(x)}) \end{cases}$$

e con

$$g(y_j) = \begin{cases} \alpha & \text{se } j \in \Lambda_{i(x)} \\ 0 & \text{se } j \notin \Lambda_{i(x)} \end{cases} \quad \text{con } \alpha \text{ cost. positiva}$$

Il valore d'attivazione delle unità che cadono nell'intorno. Adesso abbiamo bisogno di una espressione che ci permetta di variare i pesi dell'intorno in modo tale che il vettore peso si "sposti" verso l'input.

Le mappe di Kohonen - SOFM

$$\frac{d\mathbf{w}_j}{d\mathrm{t}} = \begin{cases} \eta\mathbf{x} - \alpha\mathbf{w}_j & \text{se } j \in \Lambda_{i(x)} \\ 0 & \text{se } j \notin \Lambda_{i(x)} \end{cases}$$

η tasso di apprendimento

senza perdita di generalità possiamo usare lo stesso fattore di scala

$$\frac{d\mathbf{w}_j}{d\mathrm{t}} = \begin{cases} \eta(\mathbf{x} - \mathbf{w}_j) & \text{se } j \in \Lambda_{i(x)} \\ 0 & \text{se } j \notin \Lambda_{i(x)} \end{cases}$$

passando al caso discreto è immediata la regola di variazione dei pesi

$$\mathbf{w}_j(n+1) = \begin{cases} \mathbf{w}_j(n) + \eta(n)\left[\mathbf{x} - \mathbf{w}_j(n)\right] & \text{se } j \in \Lambda_{i(x)}(n) \\ \mathbf{w}_j(n) & \text{se } j \notin \Lambda_{i(x)}(n) \end{cases}$$

si noti che η è divenuta una funzione del tempo.

Le mappe di Kohonen - SOFM

L'effetto della modifica nei vettori di peso indotta dall'ultima equazione è quello di muovere il vettore dei pesi dell'unità winner verso il vettore di input . Ripetendo questa operazione, su tutti i dati di training, i vettori dei pesi tenderanno a seguire la distribuzione dei vettori d'ingresso. L'algoritmo porta dunque ad una condizione di ordinamento topologico della features-map dello spazio di input *nel senso che i neuroni che sono adiacenti nel reticolo tendono ad avere pesi sinaptici simili*.

Alcune considerazioni: l'accuratezza della mappa è tanto maggiore quanto maggiore è il numero di iterazioni e dalla scelta della funzione di vicinanza e del parametro d'apprendimento. Per quanto riguarda la funzione, questa, può assumere anche forme esagonali o persino una forma gaussiana continua. In ogni caso, questa funzione inizialmente deve includere tutti i neuroni della rete e gradualmente restringersi con il tempo.

Le mappe di Kohonen - SOFM

Per ottenere ciò, si prevedono due fasi: la fase di *ordinamento* dove si permette che il raggio di vicinanza decresca linearmente con il tempo fino ad includere una coppia di neuroni. Mentre nella seconda fase, chiamata di *convergenza*, il raggio deve contenere solo i neuroni massimamente vicini, eventualmente 1 o solo il vincitore. Per quanto riguarda la scelta del parametro d'apprendimento, η, durante la fase di ordinamento, dovrebbe partire con un valore prossimo all'unità e successivamente decrescere monoticamente in funzione del tempo. Una accurata funzione del tempo non è importante: potrebbe essere lineare, esponenziale, o inversamente proporzionale con n. Ad esempio

$\eta(n) = 0.9x(1 - \frac{n}{\text{n° iterazioni}})$ nella fase di convergenza,

per un fine aggiustamento della mappa e quindi dovrebbe mantenersi su piccoli valori (per esempio sull'ordine di 0.2 o meno) per un lungo periodo di tempo.

Le mappe di Kohonen - SOFM

PROPRIETA' DELL'ALGORITMO

Efficiente processamento delle informazioni: al sistema nervoso viene richiesto di analizzare eventi complessi che compaiono in processi dinamici che lo coinvolgono. Questo richiede l'uso di strategie di processamento che permettano una rapida manipolazione di una grande quantità di dati. Le mappe computazionali svolgono questo compito efficacemente grazie alla capacità di processamento parallelo delle cellelule neurali.

Facilità di accesso alle informazioni: l'uso delle mappe computazionali semplifica lo schema di connettività richiesto per l'utilizzo delle informazioni, processate nelle mappe, da moduli cerebrali di ordine superiore. Ad esempio le "coordinate" del winner possono rappresentare le informazioni da passare ad un modulo superiore

Le mappe di Kohonen - SOFM

Riduzione dello spazio di dimensionalità: le mappe hanno la capacità di “compattare” un gran numero di differenti input sensori in una regione limitata. Questa “economicità” nella rappresentazione dei dati e delle loro interrelazioni è una caratteristica del cervello. Nell’atto del pensare e nei processi del subconscio vi è una generale tendenza a comprimere l’informazione favorendo rappresentazioni ridotte dei fatti più rilevanti, senza tuttavia la perdita della conoscenza delle loro interrelazioni.

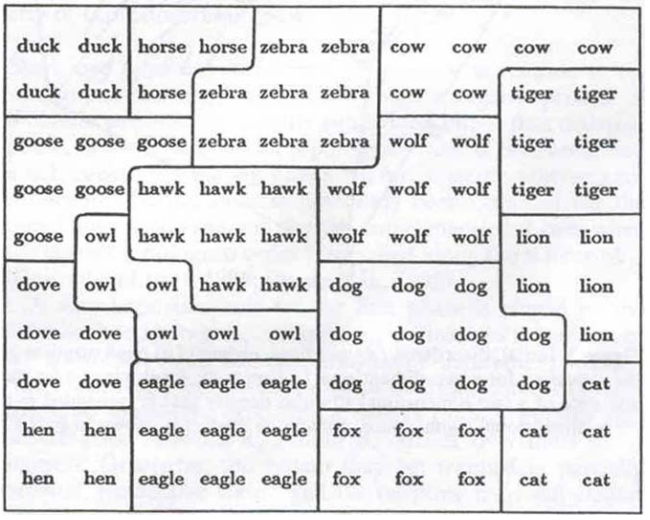
Rappresentazione comune di informazioni di natura diversa: un’altra caratteristica delle mappe è quella di fornire una rappresentazione comune dei risultati di differenti tipi di computazione effettuate; questo facilità la strategia di elaborazione di questi risultati da parte del sistema nervoso. Ad esempio, funzionalità sensorie di tipo diverso sono mappate su aree diverse con caratteristiche simili

Le mappe di Kohonen - SOFM

VISUALIZZAZIONE DELLE MAPPE

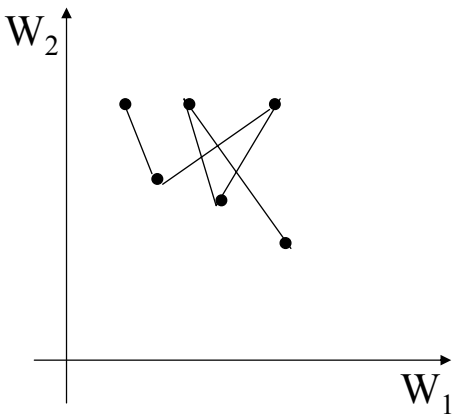
Esistono due metodi per visualizzare il comportamento delle mappe e rendersi conto effettivamente del tipo di elaborazione effettuato. Il primo consiste nell'identificare con un'etichetta i neuroni che vengono attivati, es:

Si supponga di processare dei vettori d'ingresso che codificano delle caratteristiche di diversi animali a noi note. Quello che accade è che sulla mappa si formano dei cluster di attivazione che raggruppano gli input in base alle loro caratteristiche.



Le mappe di Kohonen - SOFM

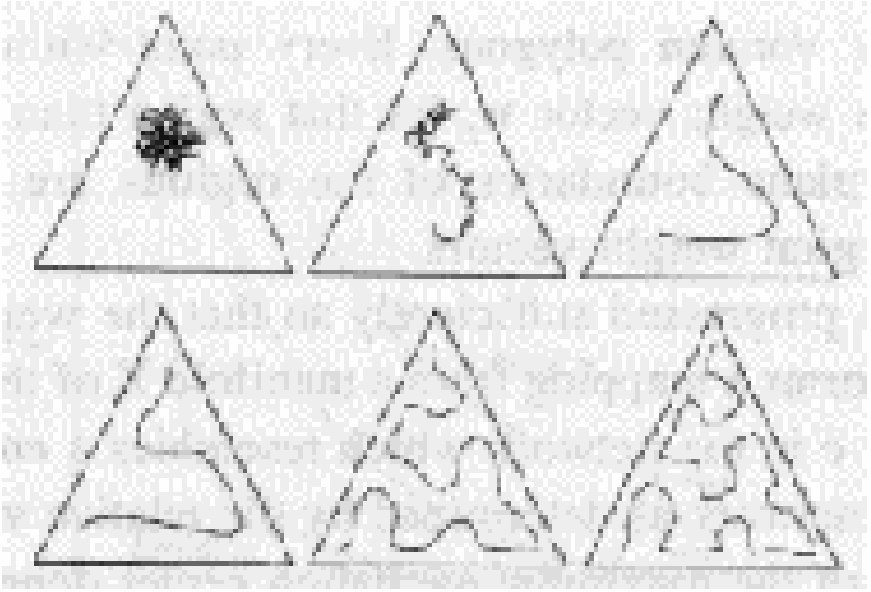
Il secondo metodo consiste nell'ottenere un grafico dell'evoluzione dell'ordinamento dei pesi. Si supponga di avere un una mappa monodimensionale e un vettore d'ingresso a due componenti e di aver inizializzato casualmente i pesi. Ad ogni unità è associato un vettore peso (w_1,w_2) che plottati sul piano W1 e W2 danno una rappresentazione geometrica della configurazione dei pesi attuale.



Si uniscano le coppie cosi ottenute medianti archi *in modo tale da avere un arco fra le unità vicine sulla mappa.* All'inizio questa disposizione appare disordinata in virtù della casualità dei pesi iniziali. Con il procedere delle fasi di apprendimento i pesi tendono a seguire la distribuzione della probabilità con cui gli input si presentano.

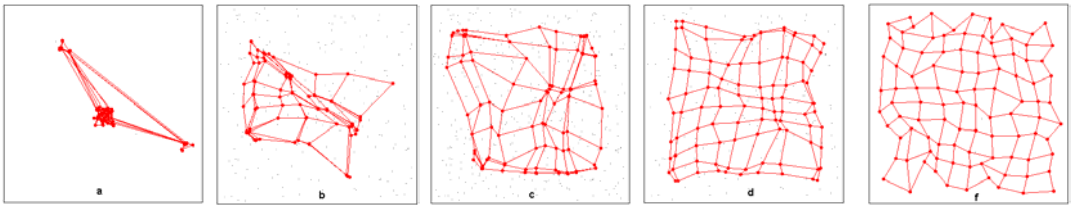
Le mappe di Kohonen - SOFM

Ad esempio se la distribuzione degli input seguisse una sorta di triangolo alla fine dell'apprendimento avremmo una distribuzione come in figura.

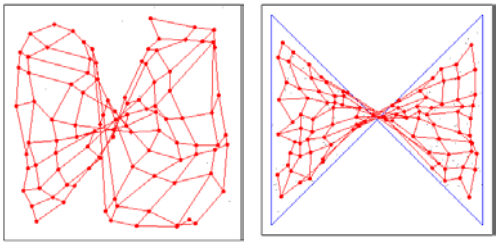


Le mappe di Kohonen - SOFM

Altri esempi:



distribuzione quadrata



distribuzione a “farfalla”

da questi esempi si evince che le mappe di Kohonen forniscono un efficace metodo per “estrarre” le caratteristiche salienti della distribuzione dei pattern dello spazio d’input

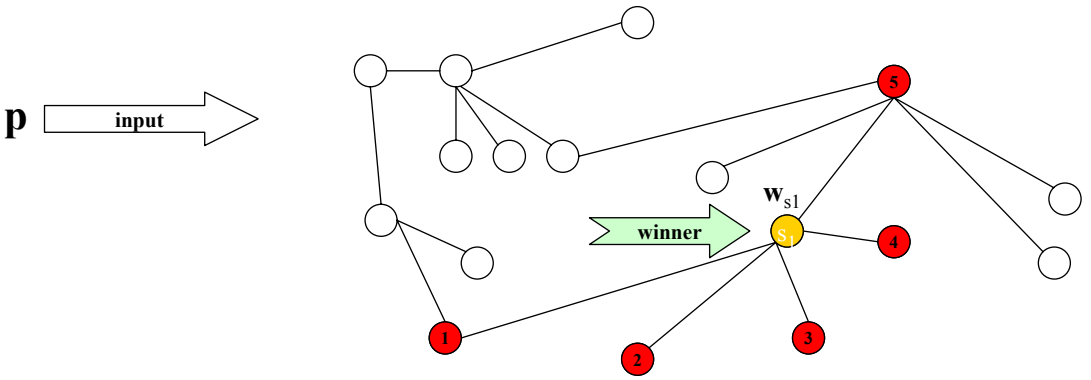
Growing Neural Gas

Bernd Fritzke, Growing Cell Structures - A Self-organizing Network for Unsupervised and Supervised Learning. ICSI TR-93-026, 1993. *Neural Networks* 7(9):1441-1460, 1994a

- Unsupervised learning paradigm
- Competitive learning methods (winner-takes-all)
- Generation of a topology-preserving mapping from the input space onto a topological structure of equal or lower dimension

Growing Neural Gas

- Competitive learning methods (winner-take-all)



$\mathbf{w}_i$ is the weight vector associated to the unit i

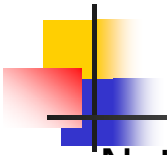
Set of direct topological neighbors of the winner unit (S_1):

$N_{s1} = \{ 1 \; 2 \; 3 \; 4 \; 5 \}$

Updating rules:

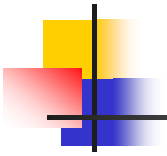
$$\mathbf{w}_{s1} = \mathbf{w}_{s1} + \varepsilon_b (\mathbf{p} - \mathbf{w}_{s1})$$

$$\mathbf{w}_i = \mathbf{w}_i + \varepsilon_n (\mathbf{p} - \mathbf{w}_i) \; (\forall i \in N_{s_1})$$



Growing Neural Gas

- Network topology is unconstrained
- Competitive Hebbian learning and edge aging are also used to generate the topology
- Uses growth mechanism (the network size does not need be predefined)
- The growth process can be interruped when a user defined performance criterion has been fulfilled



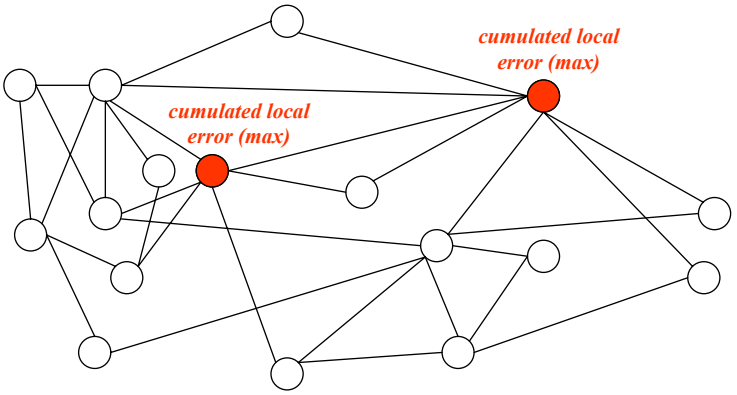
Growing techniques

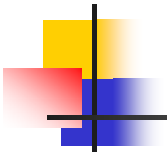
- Start with very few “random” units
- New units are inserted iteratively:
  - to determine where to insert new units, local error measures are performed during the adaptation process
  - one new unit is inserted near the unit which has accumulated the maximum error

Growing Neural Gas

- Number of units of the net is modifided during the process of self-organization
  - start with very few units
  - new units are inserted successively:

insert new cell



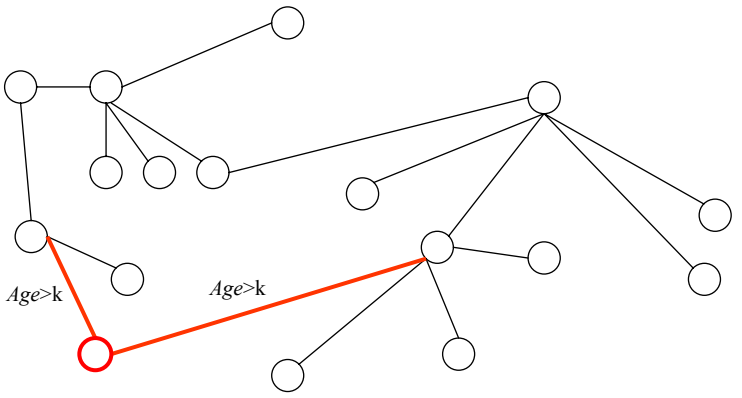


Growing techniques

- The “older” connections (edges, nodes) are suppressed:
  - edges with an “age” larger than a predetermined value are suppressed
  - if this results in nodes having no emanating edges, they are removed as well

Growing Neural Gas

- Number of units of the net is modifided during the process of self-organization
  - the "older" connection are eliminate:
    - remove edges with an age larger than a predetermined value (k)
    - if this results in nodes having no emanating edges, remove them as well.



Growing Neural Gas Algorithm

1. Initialize the set A to contain two units c_1 and c_2 with reference vectors chosen randomly according to $P(\mathbf{p})$
2. Initialize the connection set C to the empty set
3. Generate at random an input signal $\mathbf{p}$ according to $P(\mathbf{p})$
4. Determine the winner s_1 and the second-nearest unit s_2 by:

$$s_1 = \arg\min_{c \in A} \|\mathbf{p} - \mathbf{w}_c\|$$

$$s_2 = \arg\min_{c \in A \setminus \{s_1\}} \|\mathbf{p} - \mathbf{w}_c\|$$

Growing Neural Gas Algorithm

5. If a connection between s_1 and s_2 does not exist already, create it and set the age of the connection between s_1 and s_2 to zero ("refresh" the edge).
6. Add the squared distance between the input signal and the winner to a local error variable
$$\Delta E_{s_1} = \left\| \mathbf{p} - \mathbf{w}_{s_1} \right\|^2$$
7. Adapt the reference vectors of the winner and its direct topological neighbors N_{s_1} by fractions ε_b and ε_n, respectively, of the total distance to the input signal:
$$\Delta \mathbf{w}_{s1} = \varepsilon_b (\mathbf{p} - \mathbf{w}_{s1})$$
$$\Delta \mathbf{w}_i = \varepsilon_n (\mathbf{p} - \mathbf{w}_i) \quad (\forall i \in N_{s_1}) \qquad \text{where: } N_{s_1} = \{ i \in A \mid (s_1, i) \in C \}$$

Growing Neural Gas Algorithm


Visual GNG.exe

8. Increment the age of all edges emanating from s_1
9. Remove edges with an age larger than a_{max}. If this results in units having no more emanating edges, remove those units as well
10. If the number of input signals generated so far is an integer multiple of a parameter λ, insert a new unit
11. Decrease the error variables of all units
12. If a stopping criterion (e.g., net size or some performance measure) is not yet fulfilled continue with step 3

Richiami di reti neurali: applicazioni

- **Pattern Classification**
- **Clustering/categorization**
- **Function approximation**
- **Prediction/forecasting**
- **Optimization**
- **Content-addressable memory**
- **Control**