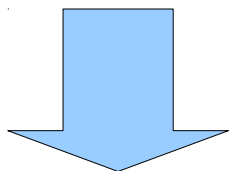


Reti Neurali

Modelli ad ispirazione biologica

Le reti neurali si ispirano al modo in cui i neuroni agiscono ed interagiscono

NB: i neuroni artificiali non sono modelli fedeli dei neuroni biologici, ne catturano solo alcuni principi



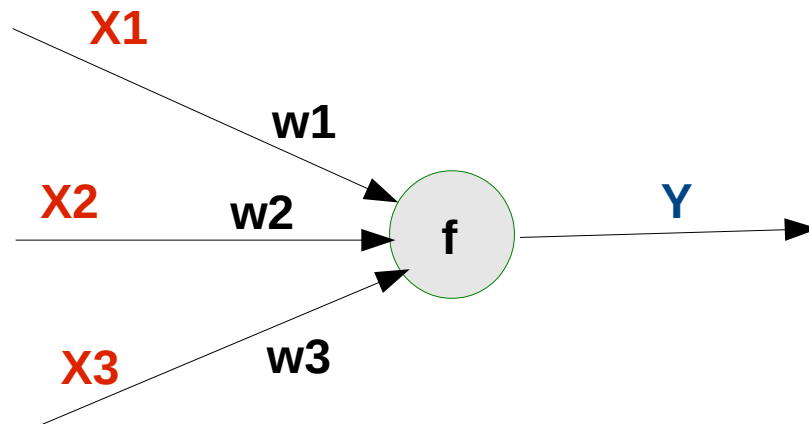
Perceptron (Rosenblatt)

B-machine di Turing

(http://www.alanturing.net/turing_archive/pages/Reference%20Articles/connectionism/Turing%27santicipation.html)

Fra i primi modelli proposti

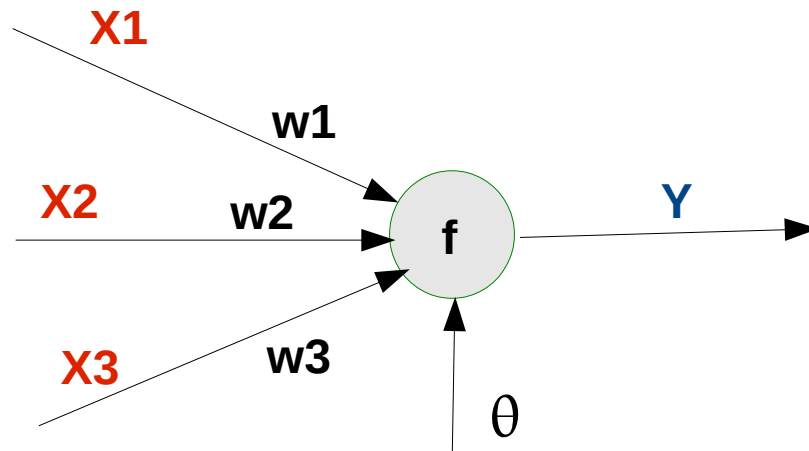
Perceptron



Un perceptron è un elemento computazionale, dotato di una piccola memoria in grado di calcolare una **funzione di attivazione** in esso strettamente codificata:

$$Y = f(\text{net})$$

tale funzione è calcolata su una composizione dei valori in ingresso, opportunamente pesati ...



$$net = \sum_{i=1}^n w_i X_i$$

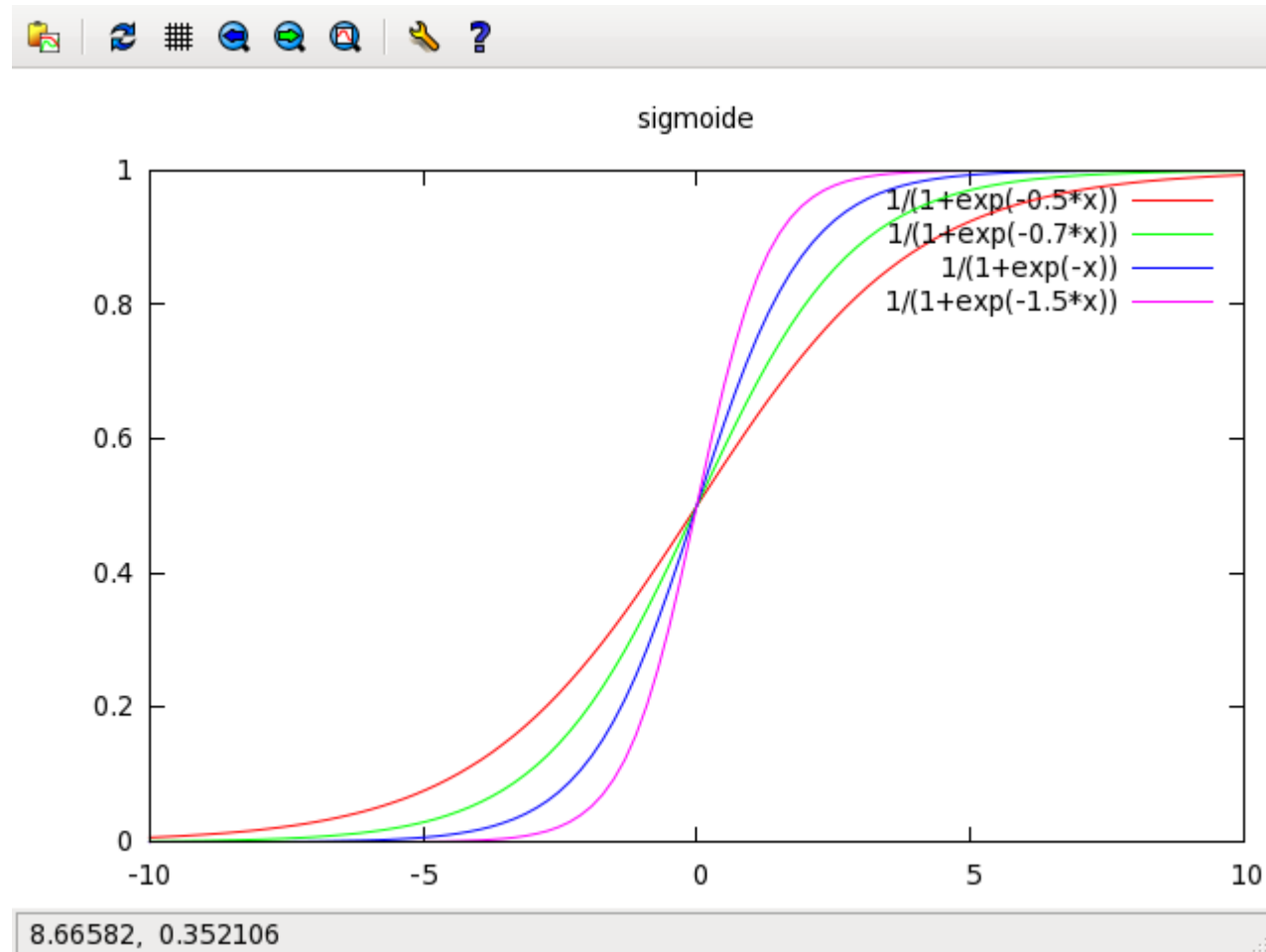
Combinazione lineare

$$Y = f(net) = \frac{1}{1 + e^{-\alpha * (net - \theta)}}$$

Funzione a gradino, spesso una sigmoide

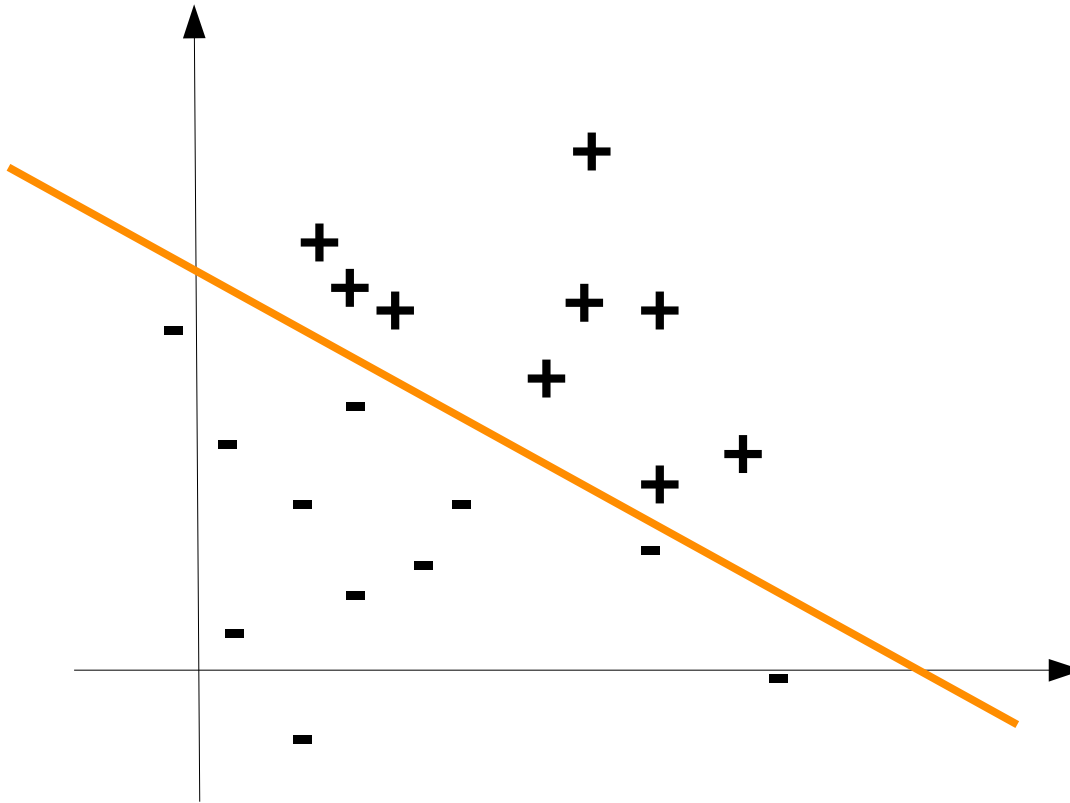
θ = soglia o bias

sigmoide



Nota: al crescere del parametro α la sigmoide tende alla funzione gradino

Cosa codifica un perceptron?

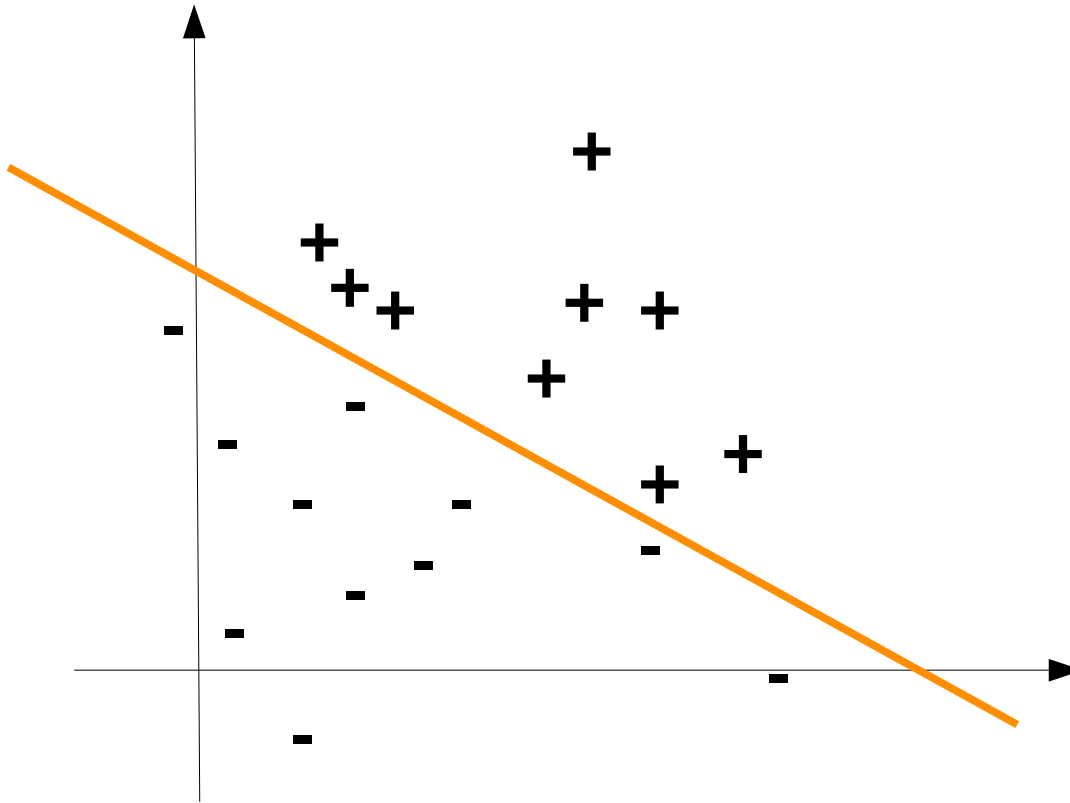


Un perceptron codifica un test lineare

Ciò che cade al di sopra dell'iperpiano codificato dai suoi pesi fa attivare il neurone, ciò che è sotto non fa attivare il neurone

Classificazione: caso particolare, sopra all'iperpiano ciò che è riconosciuto come appartenente alla classe obiettivo, sotto all'iperpiano le istanze negative

Cosa codifica un perceptron?



Un perceptron codifica un test lineare

Ciò che cade al di sopra dell'iperpiano codificato dai suoi pesi fa attivare il neurone, ciò che è sotto non fa attivare il neurone


I **pesi** sulle connessioni in ingresso definiscono la posizione e la pendenza dell'iperpiano nello spazio in cui sono definiti gli input. I pesi caratterizzano i neuroni e costituiscono la conoscenza del neurone.

Caratteristiche del perceptron

- Adatto a svolgere compiti di tipo numerico
- Consente di risolvere problemi separabili linearmente
- Conoscenza data dai pesi
- I pesi sono persistenti
- Apprendimento da esempi, supervisionato
- **Imparare = individuare la posizione corretta dell'iperpiano nello spazio**

quale segnale può essere usato per guidare l'apprendimento?

Apprendimento

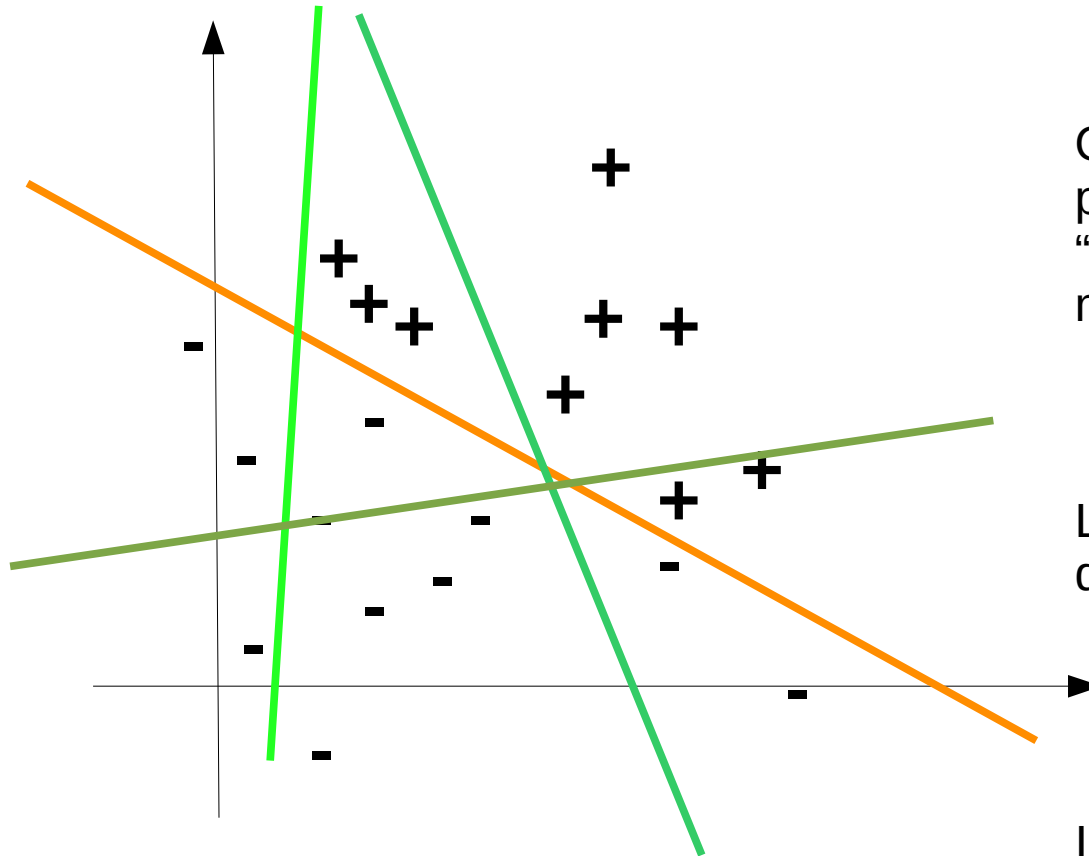
$$w_j^{(k+1)} = w_j^k + \alpha * (d - o) x_j$$


Poiché il problema affrontato è numerico possiamo sfruttare l'errore inteso come differenza fra valore desiderato e ottenuto

Il peso sulla connessione fra l'input j-mo e il neurone, all'iterazione (k+1)-ma viene aggiornato sommandogli l'errore moltiplicato per un fattore di scalamento α e per il valore della componente j-ma dell'input in questione

Novikoff ha dimostrato che quando il problema è linearmente separabile l'apprendimento converge alla soluzione, quando non è linearmente separabile invece non converge

Convergenza alla soluzione



Ogni esempio "tira" l'iperpiano per fare in modo che il suo "caso" venga trattato correttamente dal perceptron

L'iperpiano percorre lo spazio delle istanze

Il fattore di scalamento evita la rincorsa all'ultimo esempio e forza una generalizzazione

Limiti dei perceptron

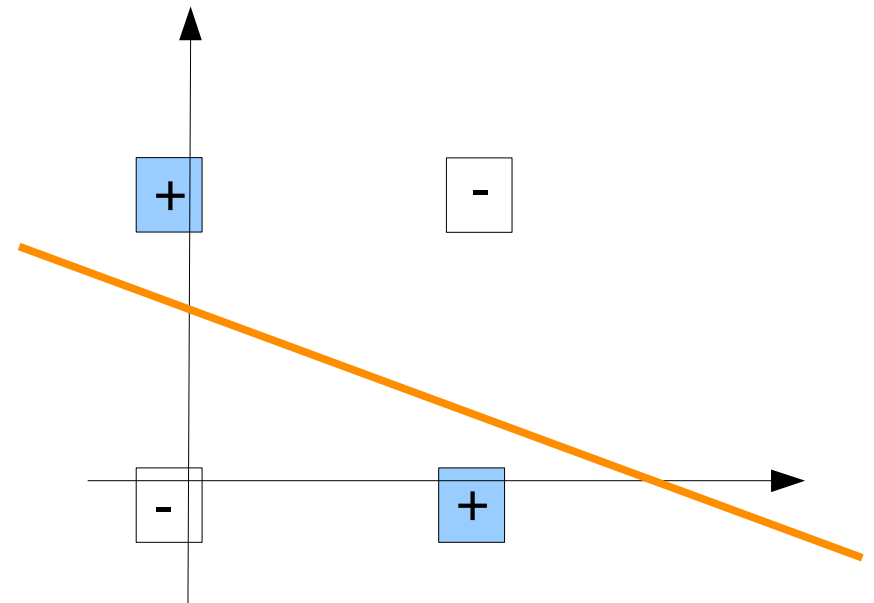
I limiti del perceptron furono evidenziati tramite un esempio semplicissimo: l'or esclusivo (XOR)

A1	A2	XOR
0	0	0
1	0	1
0	1	1
1	1	0



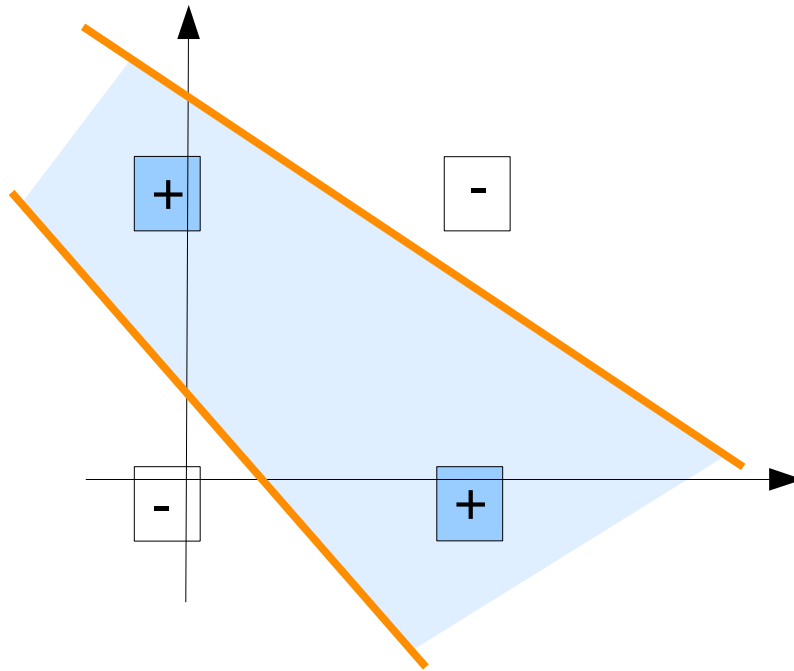
Valori degli
attributi

Output desiderato



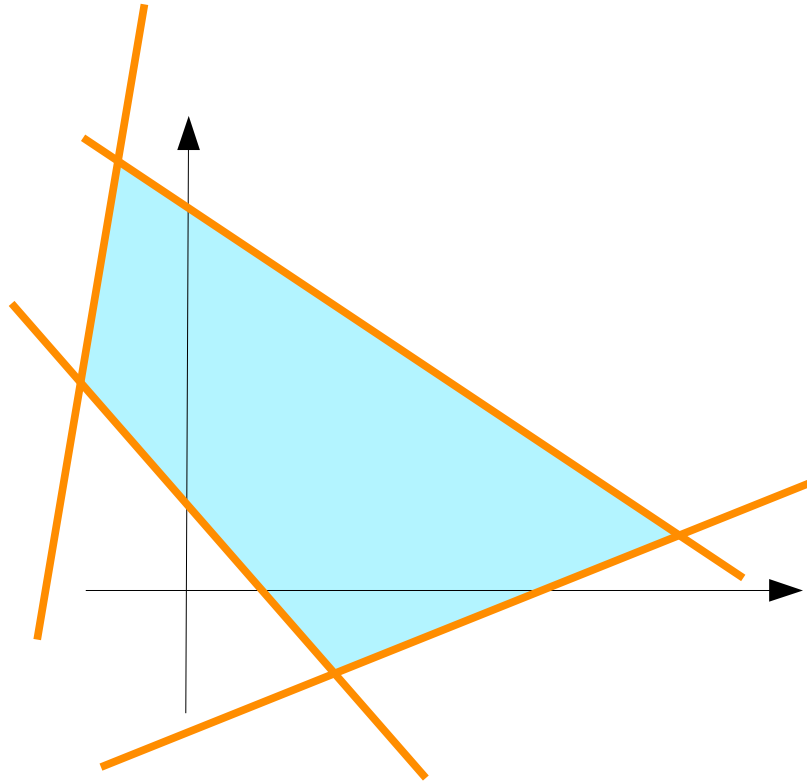
Interpreto le coppie come coordinate di punti e il risultato dello XOR come il fatto che il punto debba stare sopra o sotto all'iperpiano (non c'è verso)

Intuizione ...



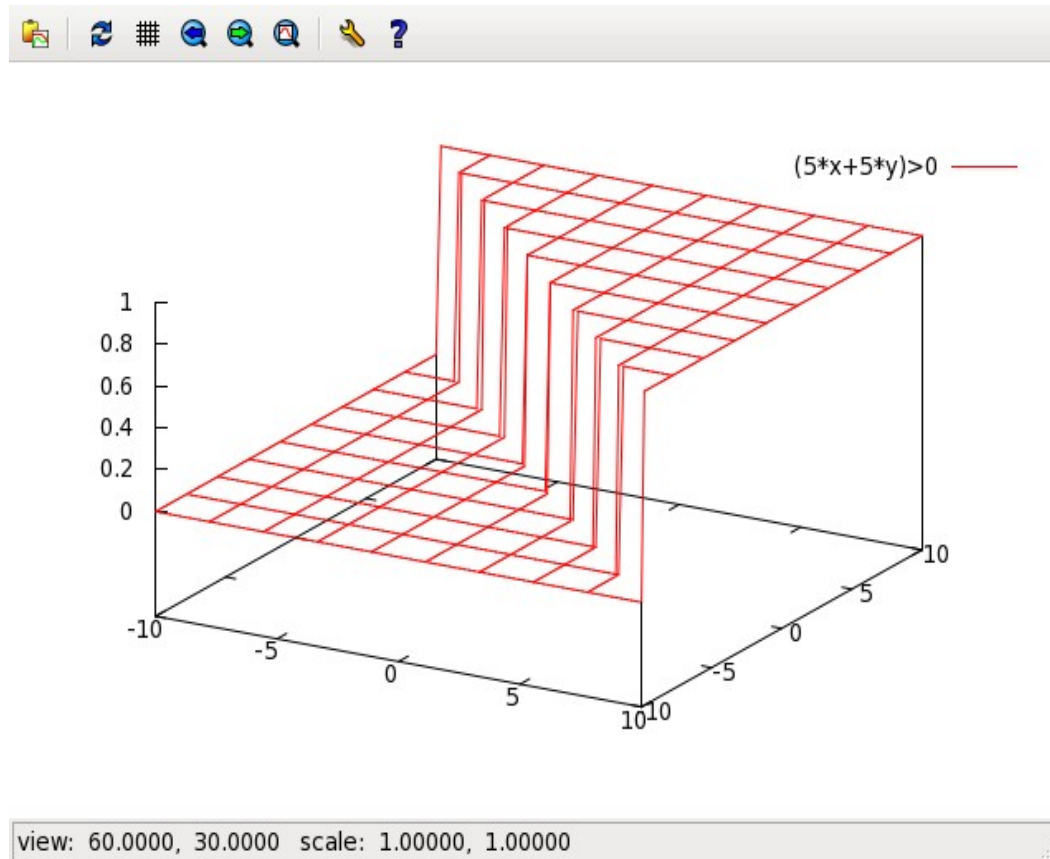
Un'iperpiano da solo non ce la fa ma se ne avessi a disposizione 2 e avessi la capacità di combinare le loro risposte?

Intuizione ...



Se ne avessi a disposizione tanti potrei addirittura circoscrivere delle regioni chiuse!

Come decide il perceptron?



Si parte dalla
combinazione lineare
degli input pesati (net)

La retta tracciata è disegnata
da $w_1 * x_1 + w_2 * x_2 = 0$

$$x_1 = \frac{-w_2}{w_1} * x_2$$

Il perceptron si attiva per i
punti che stanno sopra alla
retta cioè tali che

$$x_1 > \frac{-w_2}{w_1} * x_2$$

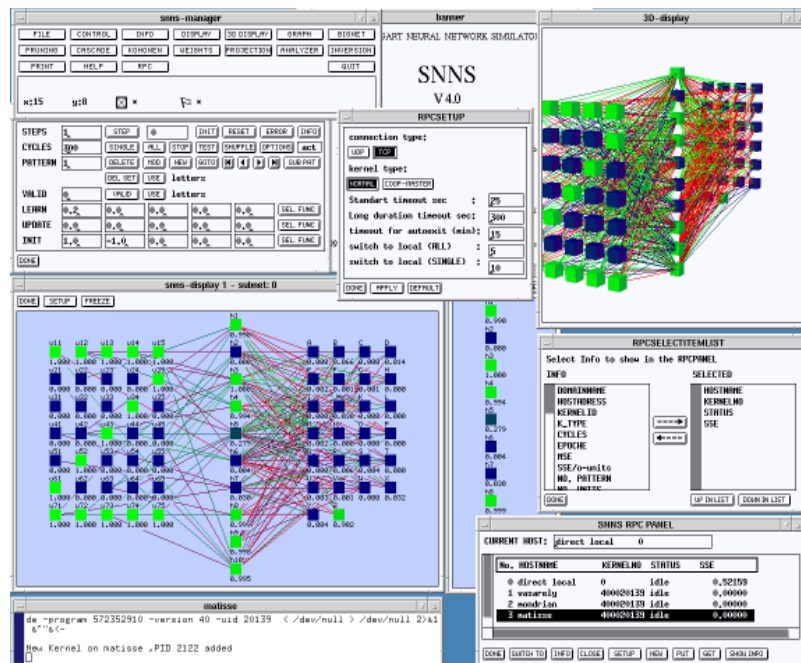
e cioè:

$$w_1 * x_1 + w_2 * x_2 > 0$$

Reti neurali

Una rete neurale è un approssimatore universale di funzioni, avente natura distribuita. È costituita da un insieme di neuroni, collegati fra di loro secondo una qualche topologia, che dipende dal modello di rete realizzato. I neuroni possono implementare funzioni diverse. Possono essere software oppure hardware.

<http://cortex.cs.nuim.ie/tools/spikeNNS/index.html>



SNNS: Stuttgart Neural Network simulator



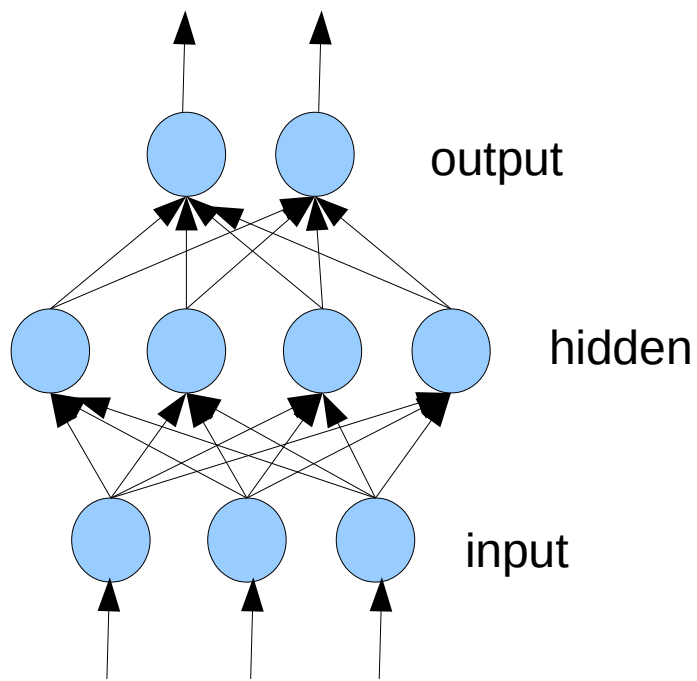
<http://steim.org/projectblog/?p=114>

Topologia di una rete neurale

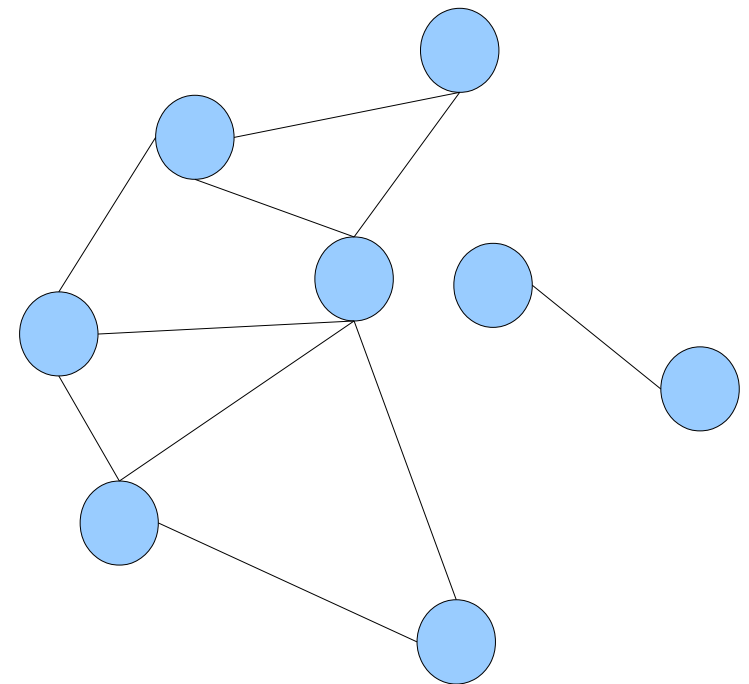
È il modo in cui sono organizzati i neuroni (o nodi) e le loro connessioni

Esempi

A strati (layered)

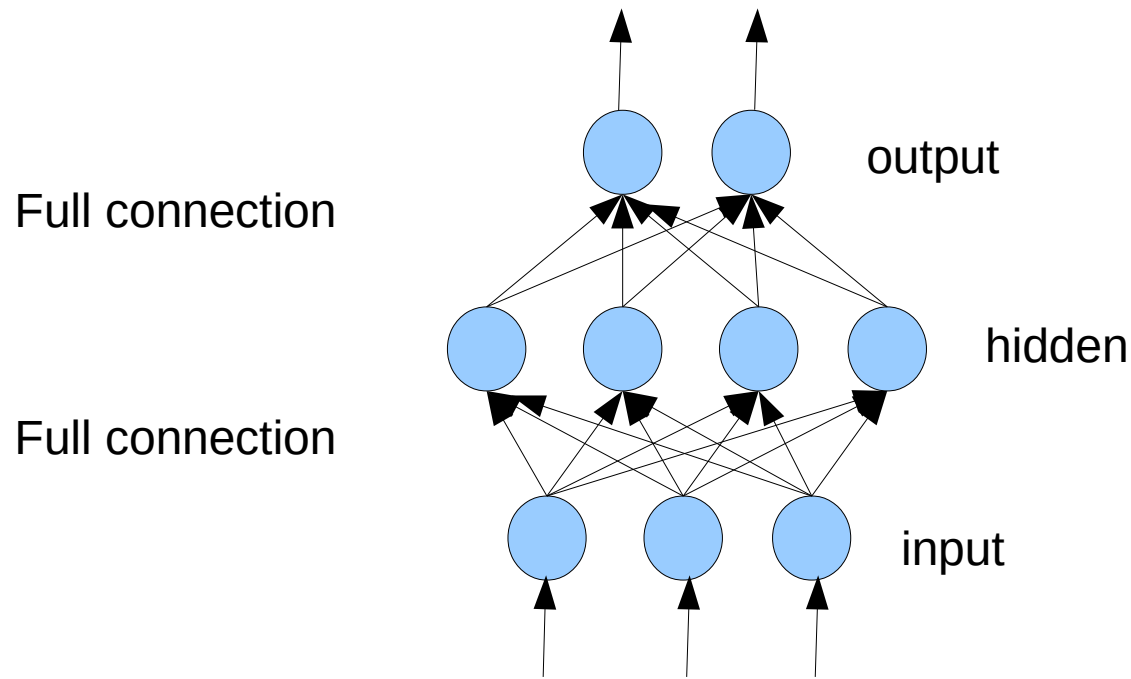


A vicinato



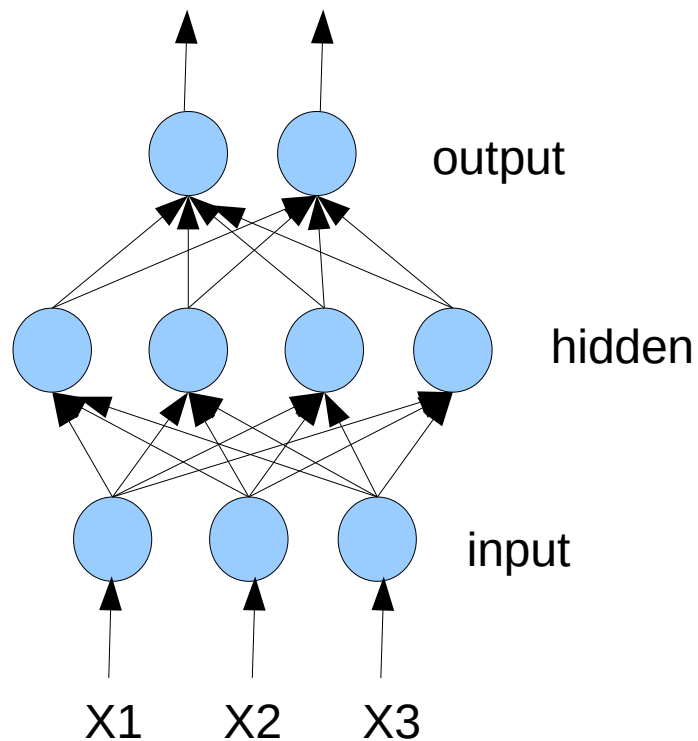
Multi-layer perceptron

Il multi-layer perceptron (MLP) è un modello di NN con topologia a strati, Feed-forward (flusso di calcolo in una sola direzione). Di solito i neuroni di input implementano la funzione identità. I neuroni hidden sono perceptron, che usano la sigmoide (o altre funzioni tipo a scalino, derivabili), i neuroni di output combinano i risultati dei neuroni hidden. Si possono avere più layer hidden



MLP e classificazione

Codifica delle classi

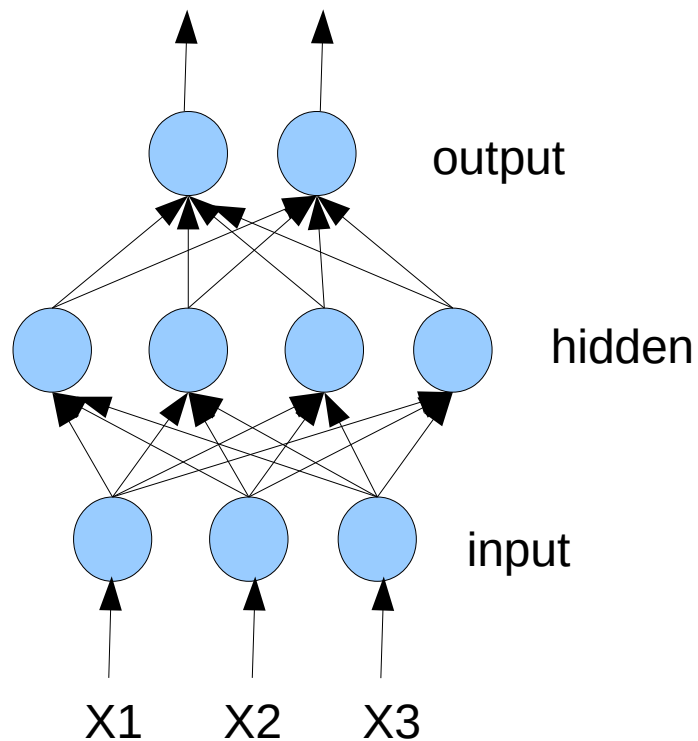


**Valori dei diversi attributi
che definiscono un'istanza**

Se il problema è riconoscere le istanze della classe X, basta un neurone di output. Se si attiva si ha il riconoscimento.

MLP e classificazione

Codifica delle classi

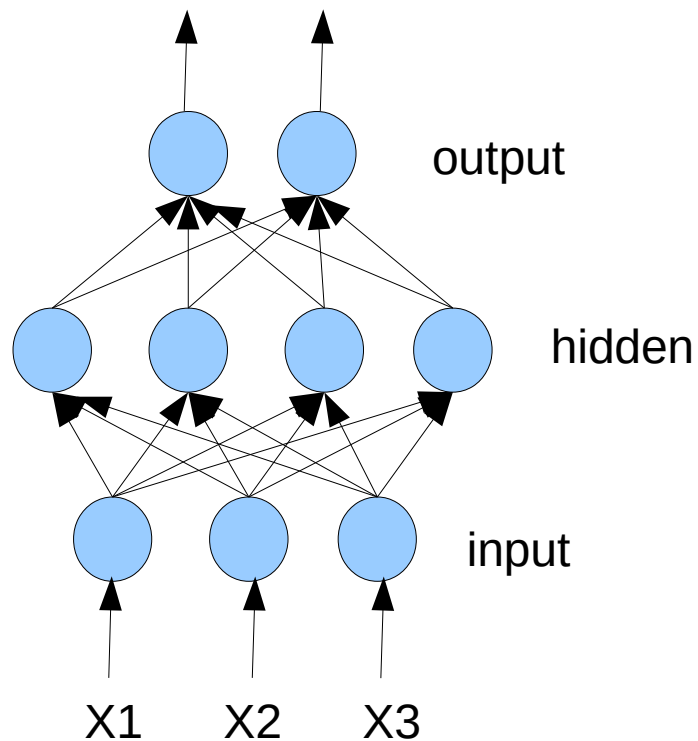


**Valori dei diversi attributi
che definiscono un'istanza**

Se il problema è distinguere le istanze di classe X da quelle di classe Y, basta un neurone di output. Si associa l'attivazione alla classe X

MLP e classificazione

Codifica delle classi



**Valori dei diversi attributi
che definiscono un'istanza**

Se il problema è distinguere fra 3 classi
occorrono almeno 2 neuroni: il problema
è infatti rappresentare tutti i numeri fino a
3 in modo binario

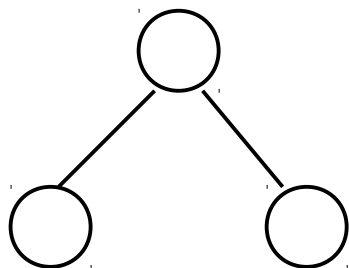
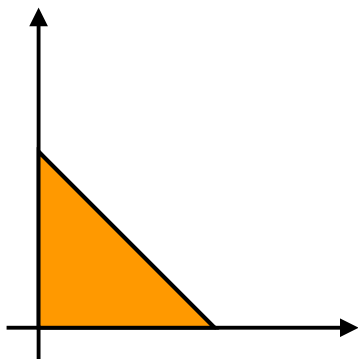
Alternativa

Spesso però si usa un neurone di output
per ogni classe. Il neurone corrispondente
alla classe giusta deve attivarsi, gli altri
rimanere a zero

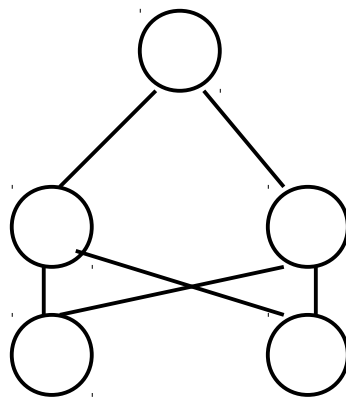
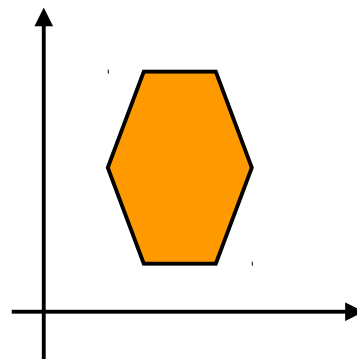
Esempio

<i>Output rete</i>	<i>Classe</i>
1 0	→ Classe A
0 1	→ Classe B

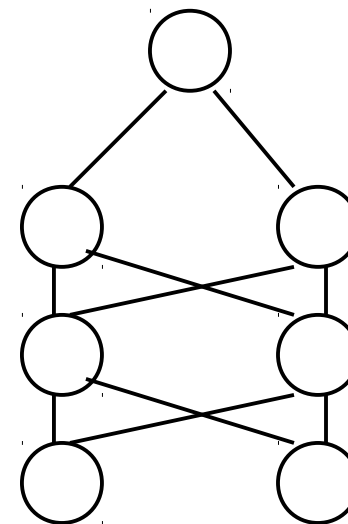
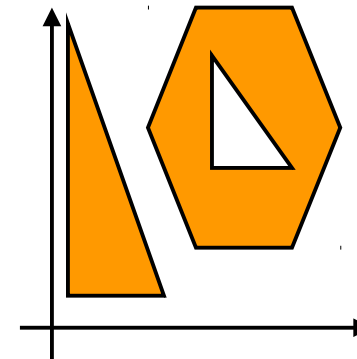
Compito degli hidden layer



Il primo layer
traccia dei confini



Il secondo layer
costruisce delle forme

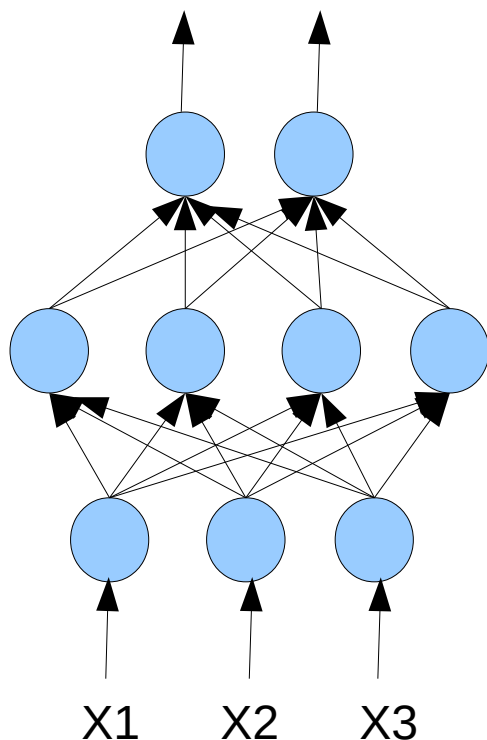


Il terzo layer
crea forme qualsivoglia
complesse

(immagine: Andrew Philpides, Univ. of Sussex)

Cosa può imparare un MLP?

I MLP a 3 layer i cui neuroni hanno come funzione di attivazione una sigmoide sono approssimatori universali di funzioni ... a patto di poter utilizzare un qualsivoglia numero di neuroni.



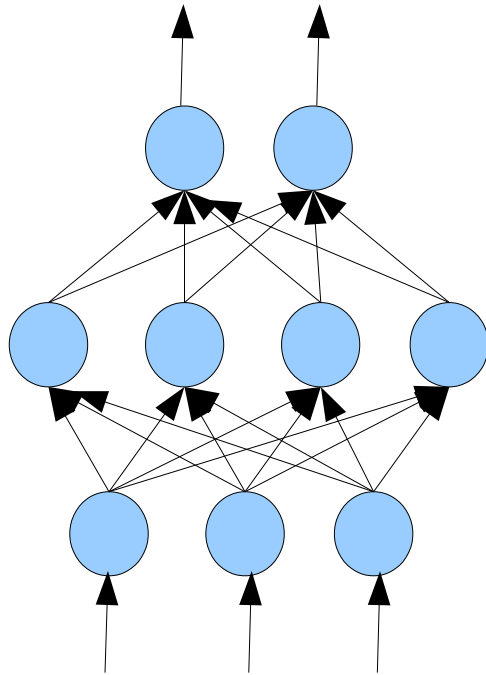
Dove viene memorizzata la **conoscenza** acquisita dalla rete neurale?

Nella matrice dei **pesi** che definiscono la forza delle connessioni

Apprendimento

- Una rete neurale di tipo **MLP** impara in **modo supervisionato** inducendo la matrice dei pesi a partire da un insieme di esempi (etichettati nel caso della classificazione)
- Per ogni istanza vengono effettuate:
 - **Passata in avanti (forward)**: l'**istanza** viene sottoposta all'MLP che la elabora e produce un **risultato**;
 - **Passata all'indietro (backward)**: l'**errore** viene utilizzato per modificare i **pesi** partendo dalle connessioni più vicine ai neuroni di output e procedendo a ritroso
- Il learning set viene elaborato dalla rete molte volte, ogni passata dell'intero learning set è detta **epoca di apprendimento**

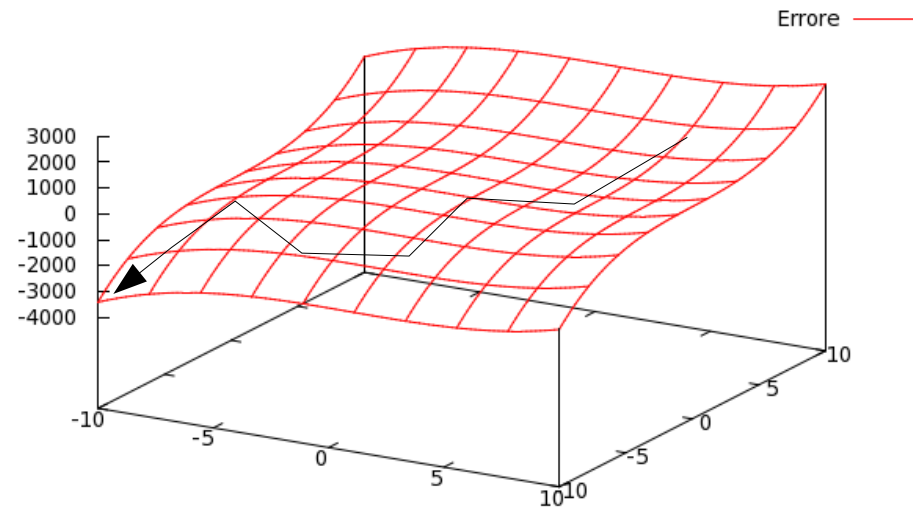
Discesa del gradiente



La formula della
funzione errore?

Da cosa dipende l'errore E di una rete neurale?
Dalla matrice dei pesi W

Come far imparare una rete neurale?
Facendole cercare la configurazione di pesi W che
minimizza l'errore E



view: 60.0000, 30.0000 scale: 1.00000, 1.00000

Discesa del gradiente

$$\Delta w_{ji} = -\lambda \frac{\partial E(\bar{w})}{\partial w_{ji}}$$

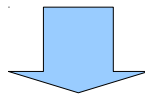
Ogni peso w_{ji} (su una connessione dal neurone j -mo al neurone i -mo) viene modificato sottraendo la derivata parziale dell'errore rispetto al peso stesso

Gradiente: misura che indica la direzione di massimo cambiamento di una funzione

Viene usata per trovare i massimi (nel nostro caso i minimi) di una curva

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n} \right)$$

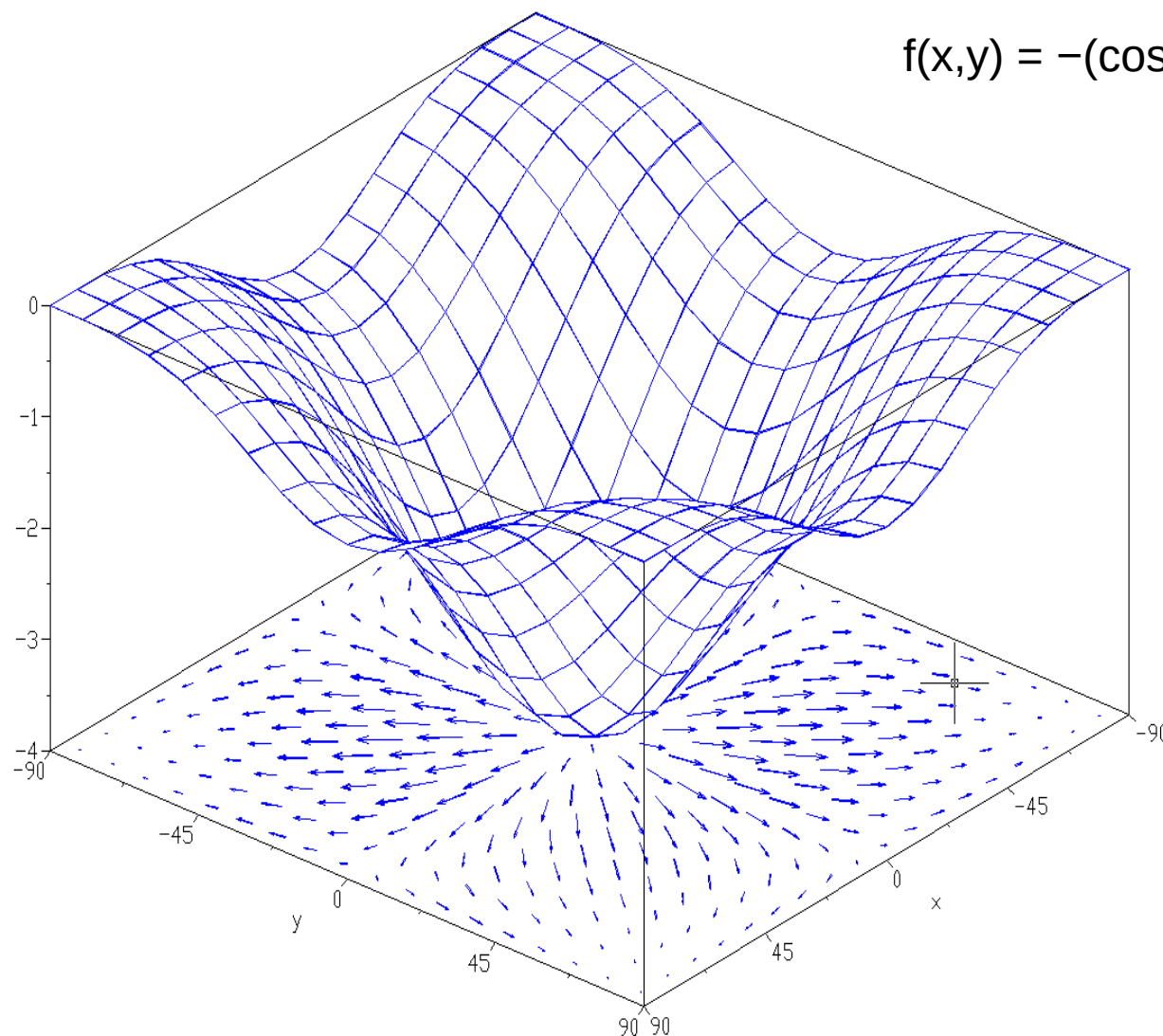
Funzione rispetto alle variabili su cui è calcolata



$$\nabla E = \left(\frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n} \right)$$

Errore rispetto ai pesi

Gradiente: esempio



$$f(x,y) = -(\cos 2x + \cos 2y)^2$$

Il gradiente è disegnato sul piano alla base della figura, in ogni punto viene rappresentato in forma di un vettore che indica la direzione del massimo.

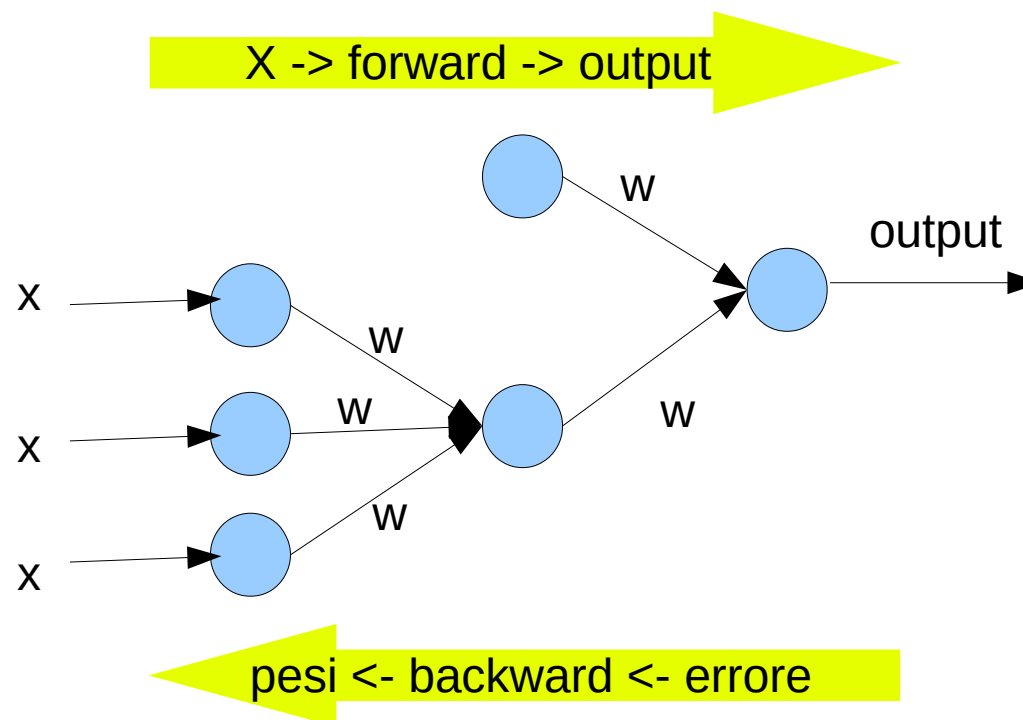
La lunghezza del vettore indica la rapidità di crescita

Discesa del gradiente

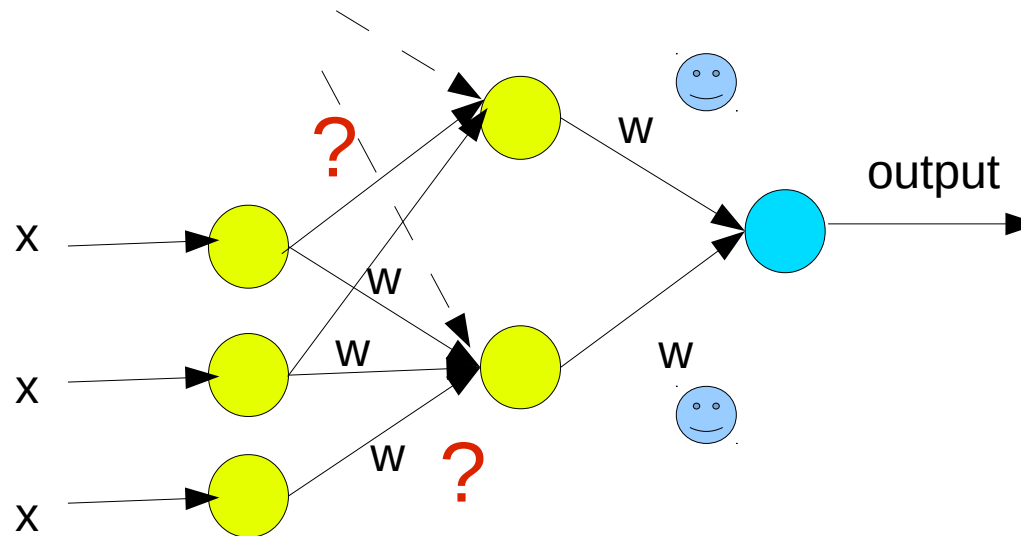
$$\Delta w_{ji} = -\lambda \frac{\partial E(\bar{w})}{\partial w_{ji}}$$

Ogni peso w_{ji} (su una connessione dal neurone j -mo al neurone i -mo) viene modificato sottraendo la derivata parziale dell'errore rispetto al peso stesso

È una **tecnica greedy**, che cerca un punto di minimo



Problema: distribuire il credito/biasimo



L'errore viene calcolato esclusivamente per i **neuroni del layer di output**, ma qual è la **formula dell'errore**? **Quale segnale** utilizzare per modificare i pesi fra tutti i neuroni degli **strati precedenti**?

$$\Delta w_{ji} = -\lambda \frac{\partial E(\bar{w})}{\partial w_{ji}}$$

Nota: la forma analitica serve per calcolare le derivate

Errore globale dell'MLP

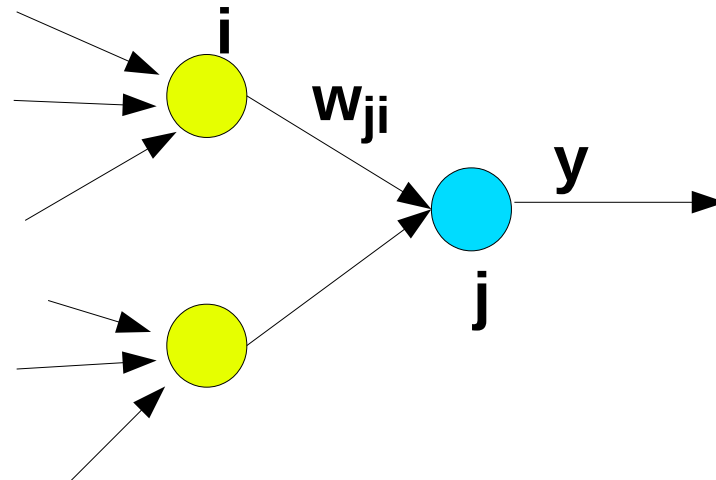
$$E = 1/2 \sum_{i=1}^p \|t_i - y_i\|^2$$

p = numero dei neuroni di output

t_i = valore desiderato per il neurone di output i -mo

y_i = valore prodotto dal neurone di output i -mo

Backpropagation: neuroni di output



Siano:

j = un neurone del layer di output

i = un neurone del layer precedente, connesso a j

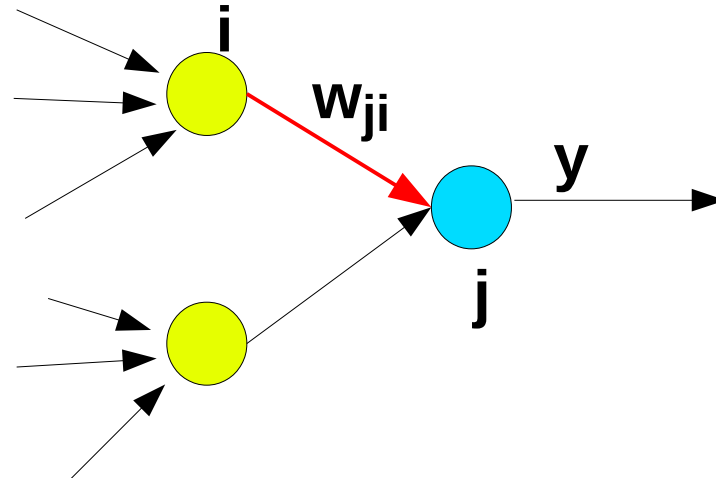
y = l'output prodotto da j per una certa combinazione di valori in ingresso

x_{ji} = il valore inviato dal neurone i al neurone j che ha contribuito a produrre y

w_{ji} = il peso corrente della connessione da i a j

t = il valore desiderato al posto di y (il target)

Backpropagation: neuroni di output



Delta rule generalizzata:

$$\Delta w_{ji} = \alpha * \delta^j * x_{ji}$$

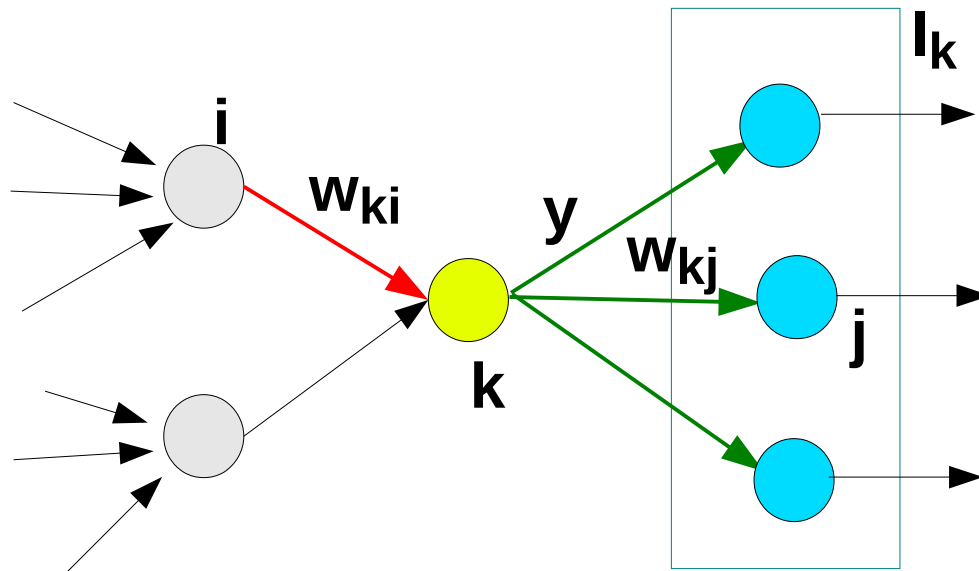
$$\delta^j = y_j * (1 - y_j) * (t_j - y_j)$$

Variazione da applicare al peso

Misura derivante dal calcolo dell'errore

Backpropagation: neuroni hidden

Idea: distribuire l'errore all'indietro fra le diverse connessioni in maniera proporzionale alla forza dei pesi correnti



Delta rule per i neuroni hidden:

$$\Delta w_{ki} = \alpha * \delta^k * x_{ki}$$

Variazione da applicare al peso

$$\delta^k = y * (1 - y) * \sum_{j \in I_k} \delta^j w_{kj}$$

Misura derivante dalla retropropagazione dell'errore

pesi <- backward <- errore

Non solo classificazione ...

Una rete neurale (non nec. di tipo MLP) è in grado di risolvere problemi di classificazione e **problemi di regressione**

Regressione: in statistica questo termine denota il problema di definire la relazione tra un insieme di variabili indipendenti e una variabile dipendente

Più in generale con questo termine si identifica il problema relativo a *costruire l'approssimazione di una funzione continua*

$$y = f(\bar{x})$$



La forma analitica della funzione non è nota
occorre indurla da esempi

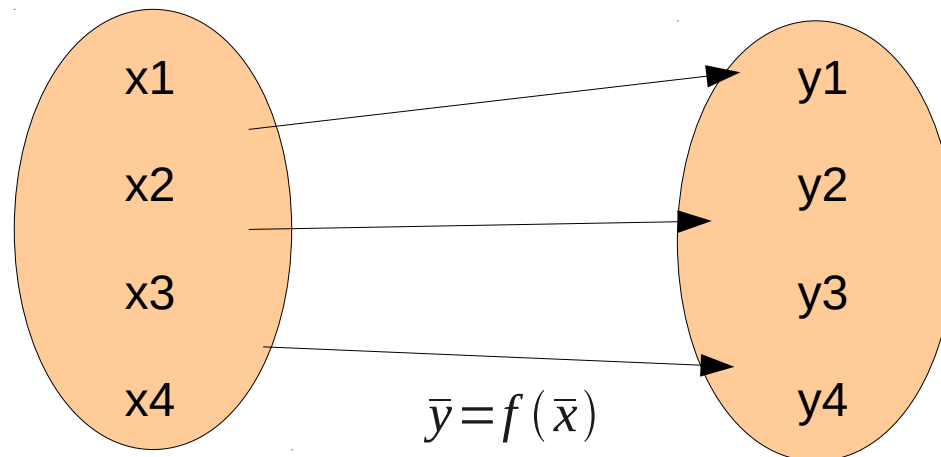
Learning set

Come per la classificazione l'apprendimento è supervisionato

	Vrb indipendenti			Vrb dipendente
	X1	...	Xn	f
<i>i1</i>	v11	...	v1n	y1
<i>i2</i>	v21	...	v2n	y2
<i>i3</i>
...				

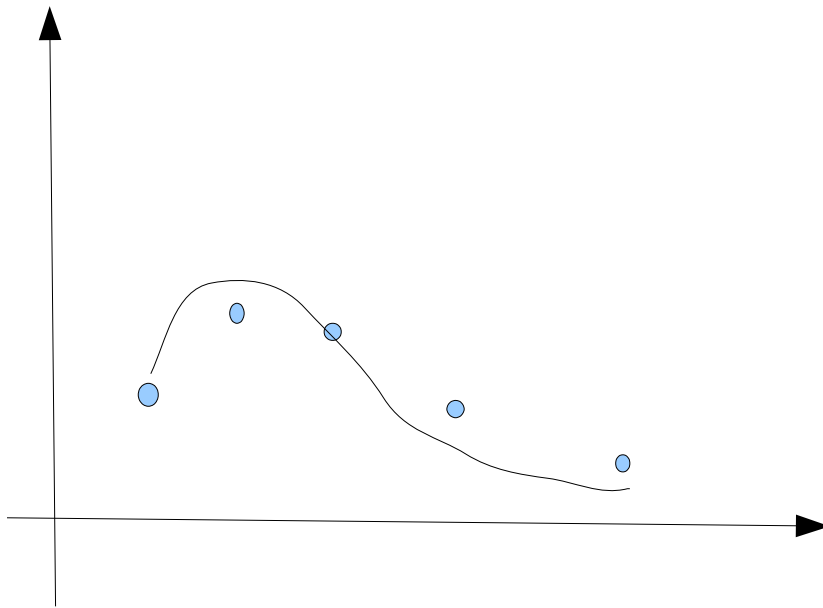
<i>ik</i>	vk1	...	vkn	yk

Funzioni a più valori

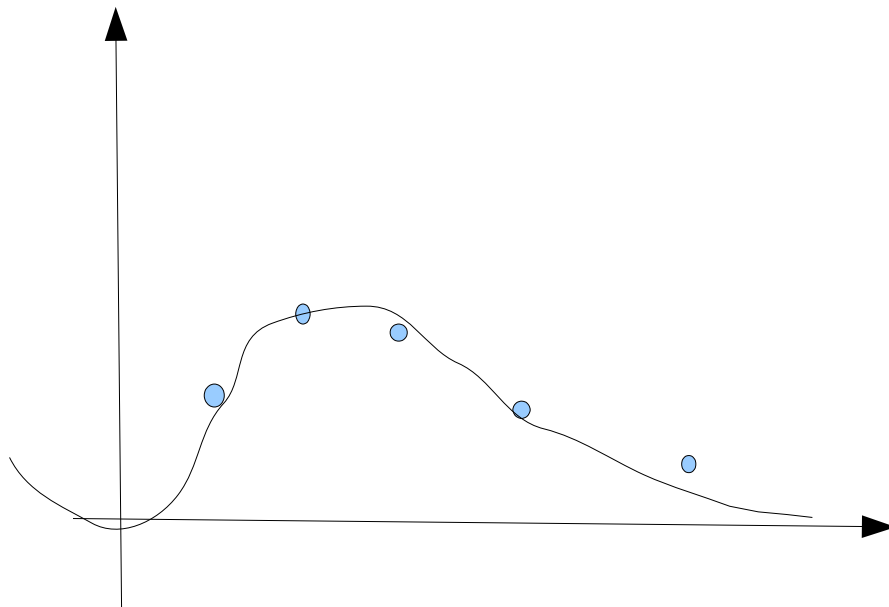


Una rete neurale può imparare funzioni a più valori, cioè funzioni che hanno più di una variabile dipendente basta prevedere un neurone di output per ogni valore prodotto dalla funzione

interpolazione/estrapolazione



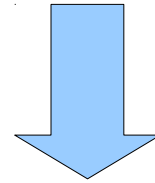
Approssimare la funzione obiettivo
fra i punti in cui il suo valore è noto



Approssimare la funzione obiettivo
al di fuori dei punti in cui il suo valore
è noto

Calcolo dell'errore

Quando si esegue un compito di regressione, non si può più utilizzare una matrice di confusione né ha senso parlare di precision e recall, TP e FP rate, accuratezza ed error rate



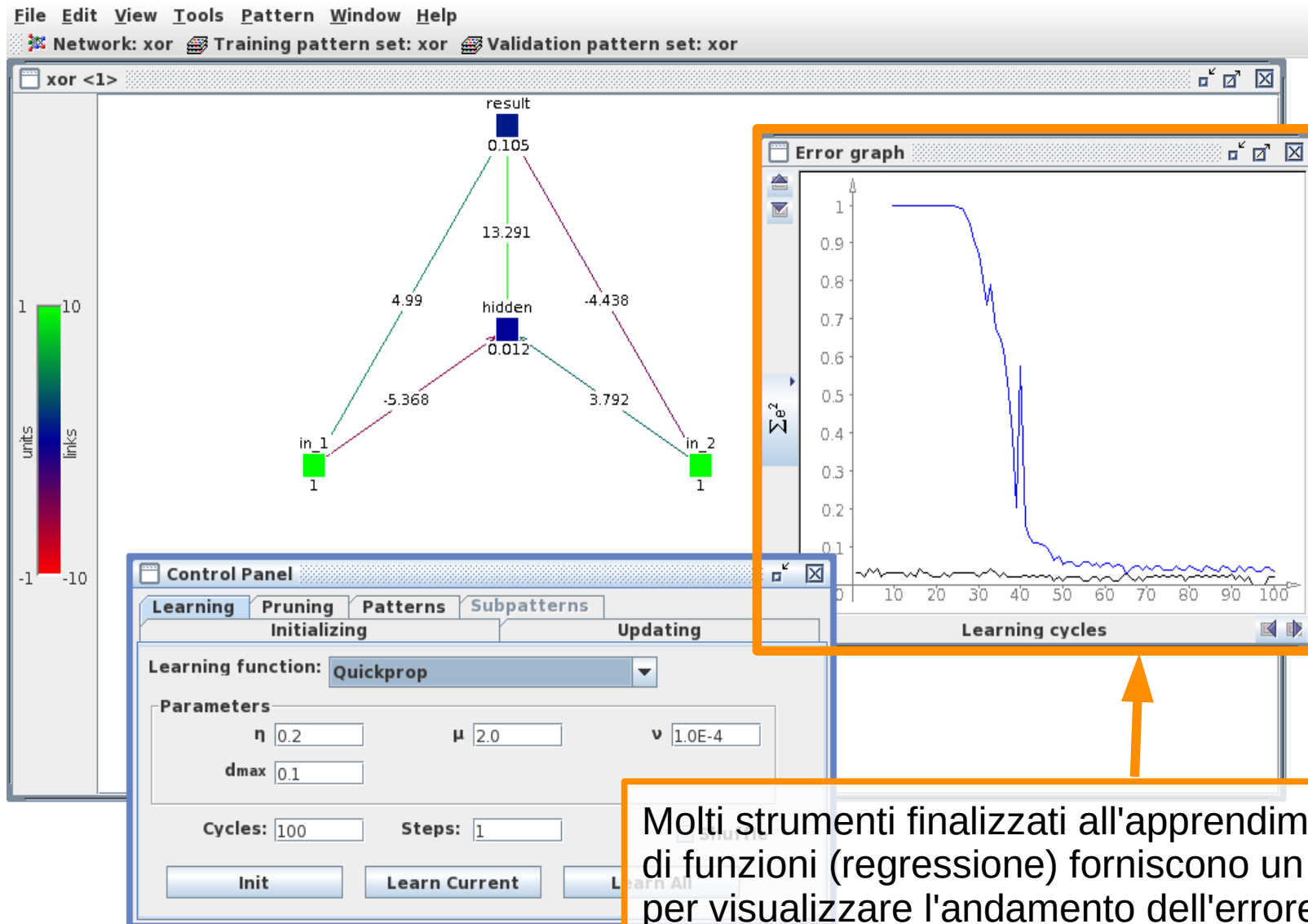
Errore Quadratico Medio -- Mean Squared Error (**MSE**)

$$MSE(y) = \frac{\sum_{i=1}^n (y_o - y_d)^2}{n}$$

Somma del quadrato degli errori compiuti sulle singole istanze, diviso il numero delle istanze su cui si è fatto il test

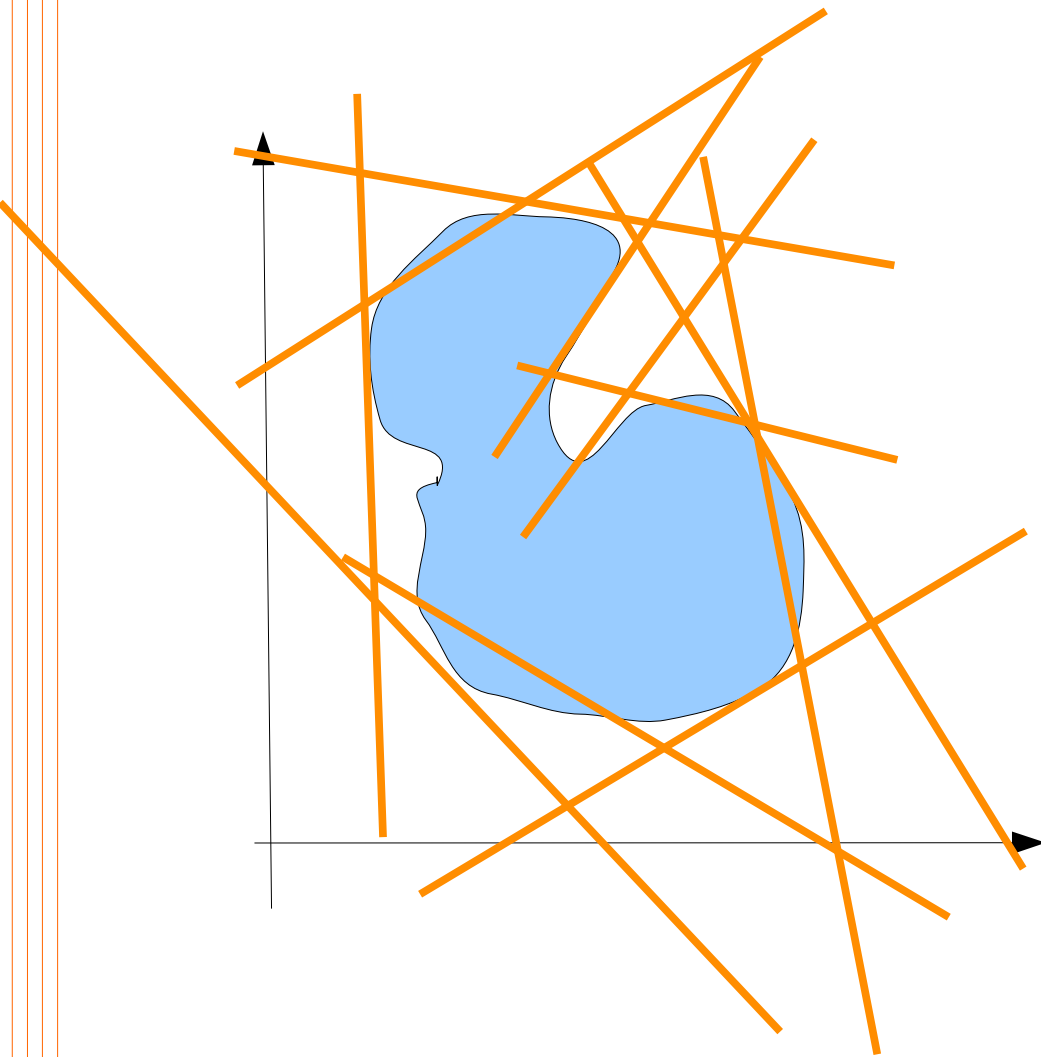
Perché i quadrati? Per evitare che a causa dei segni opposti certi errori si annullino

Visualizzazione dell'andamento dell'apprendimento



Molti strumenti finalizzati all'apprendimento di funzioni (regressione) forniscono un tool per visualizzare l'andamento dell'errore medio, per esempio di epoca in epoca

Sigmoidi

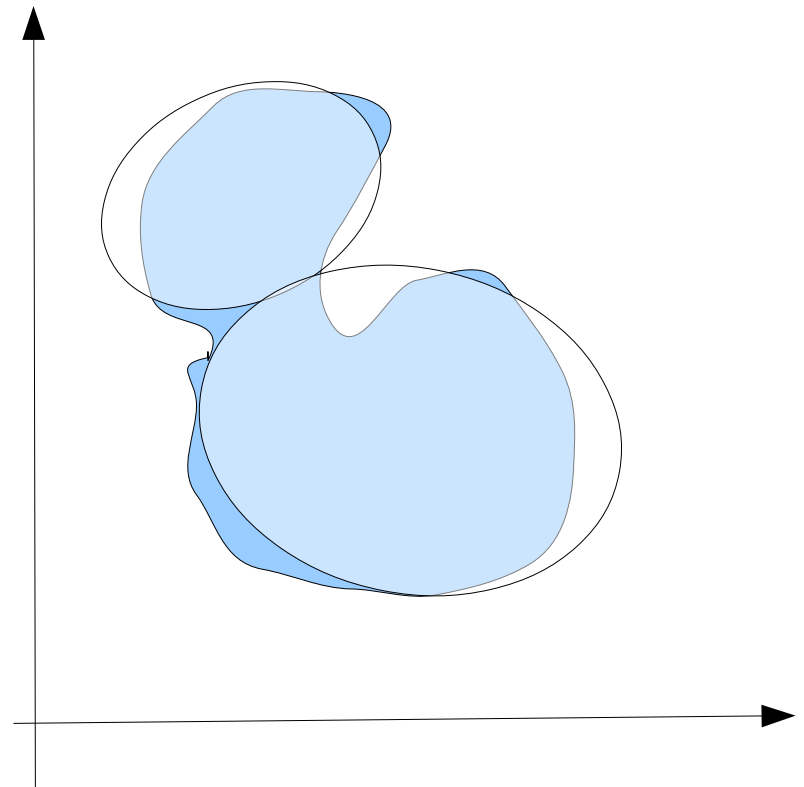


Solo sigmoidi?

E se usassi delle **funzioni radiali**?
Esempio: gaussiane

Per esplorare alcuni dei tantissimi modelli di reti neurali che sono stati proposti, potete scaricare e utilizzare JavaNNS (versione più recente dello Stuttgart Neural Network Simulator):

Vedere Moodle per i link



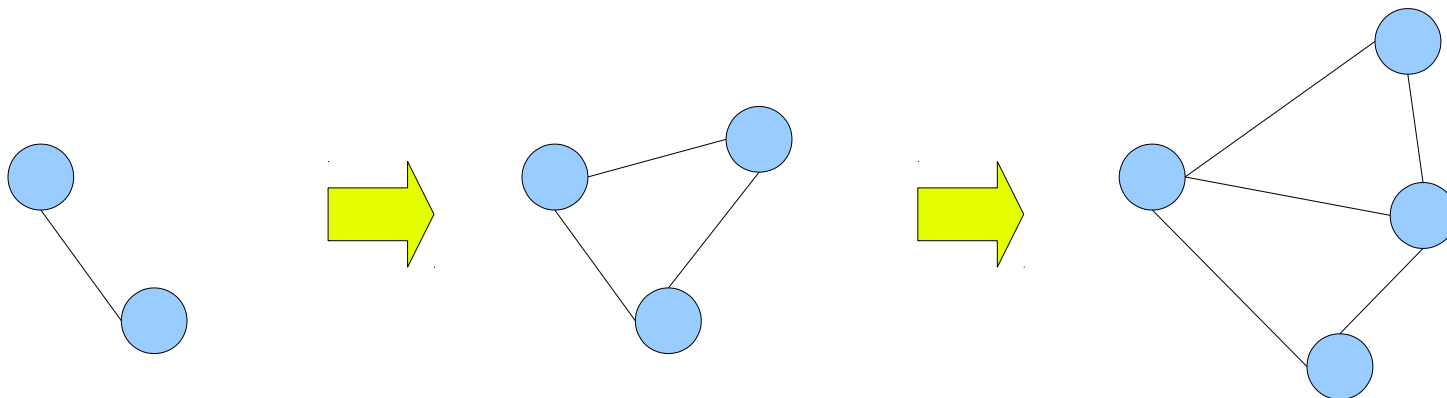
Solo reti a topologia fissa?

Cascade-correlation

NO

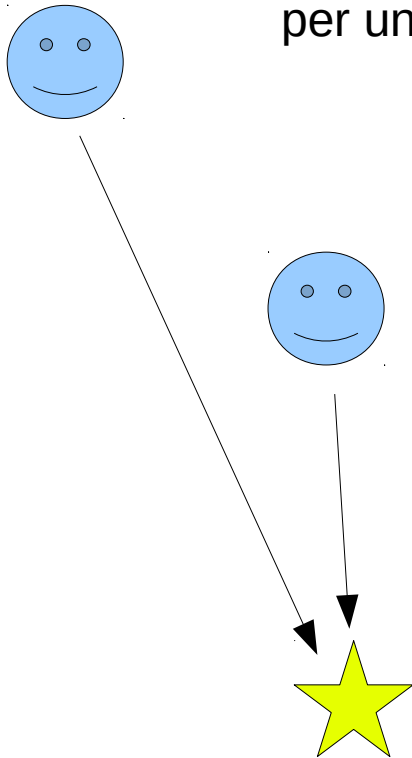
Growing-neural Gas

Self-organizing Maps



Apprendimento competitivo

Ogni neurone è interpretato come un **prototipo** e funge da referente per una *zona dello spazio degli input*

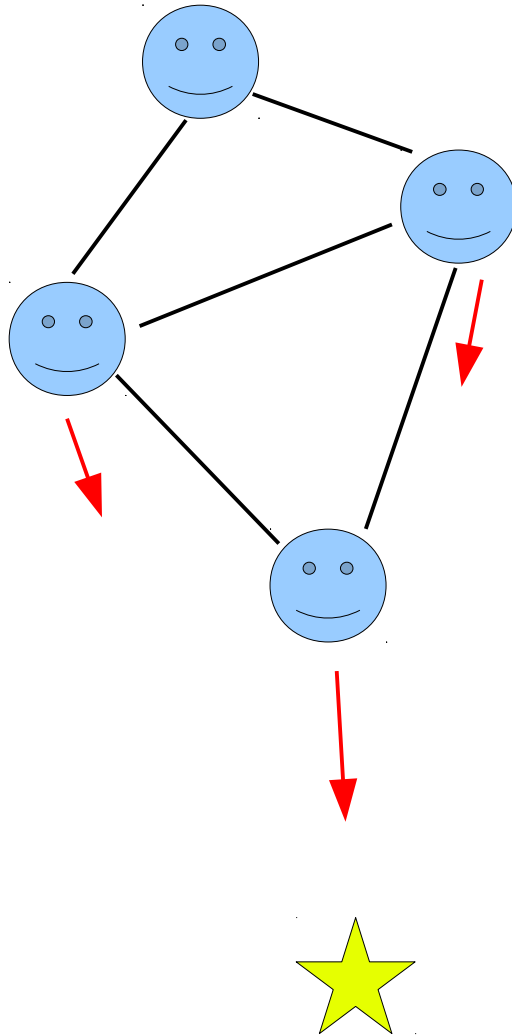


In presenza di un nuovo dato, viene calcolata la **distanza** del dato da tutti i neuroni (come in K-NN da tutti gli esempi memorizzati)

Il più vicino è il **vincitore**

Il **vincitore si sposta** un po' più vicino all'istanza considerata

Apprendimento competitivo



I nodi sono organizzati in una struttura a grafo detta **vicinato**

Non solo il vincitore si sposta verso l'istanza di una certa quantità, anche i neuroni appartenenti al suo **vicinato diretto** si spostano in quella direzione, di una percentuale minore

I neuroni coinvolti accumulano ciascuno un'**errore locale** che dipende dalla distanza del neurone dall'esempio

Neurone come prototipo rappresentativo

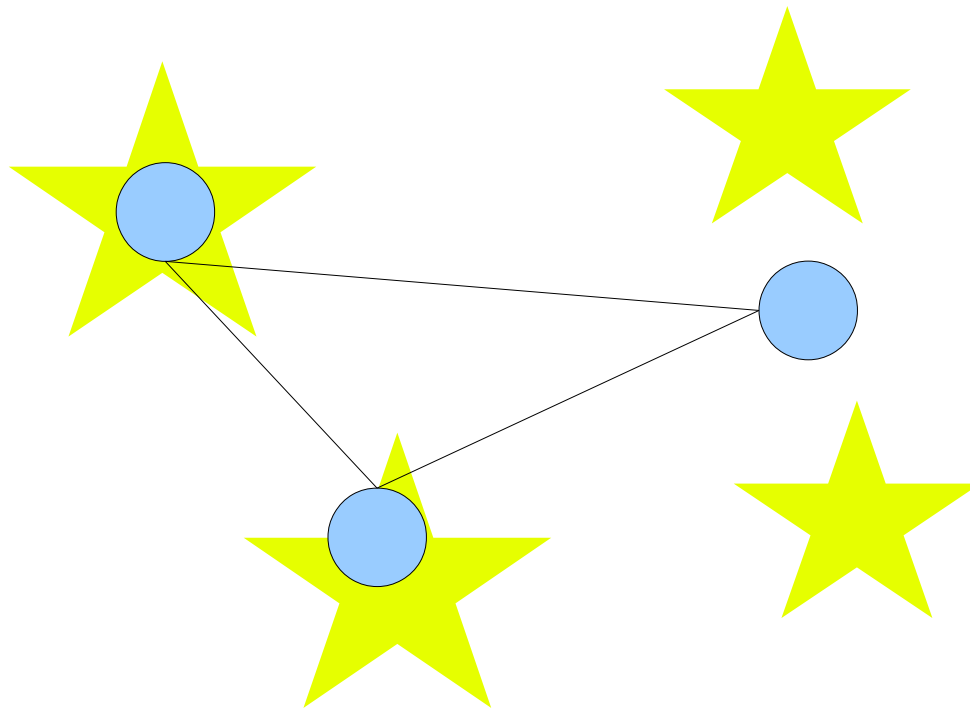
I neuroni hanno posizioni iniziali casuali, nel tempo tendono a spostarsi al centro di aree in cui sono presenti istanze: diventano prototipi rappresentativi di un insieme di istanze



NB: differenza da k-NN il neurone non corrisponde a un esempio del learning set

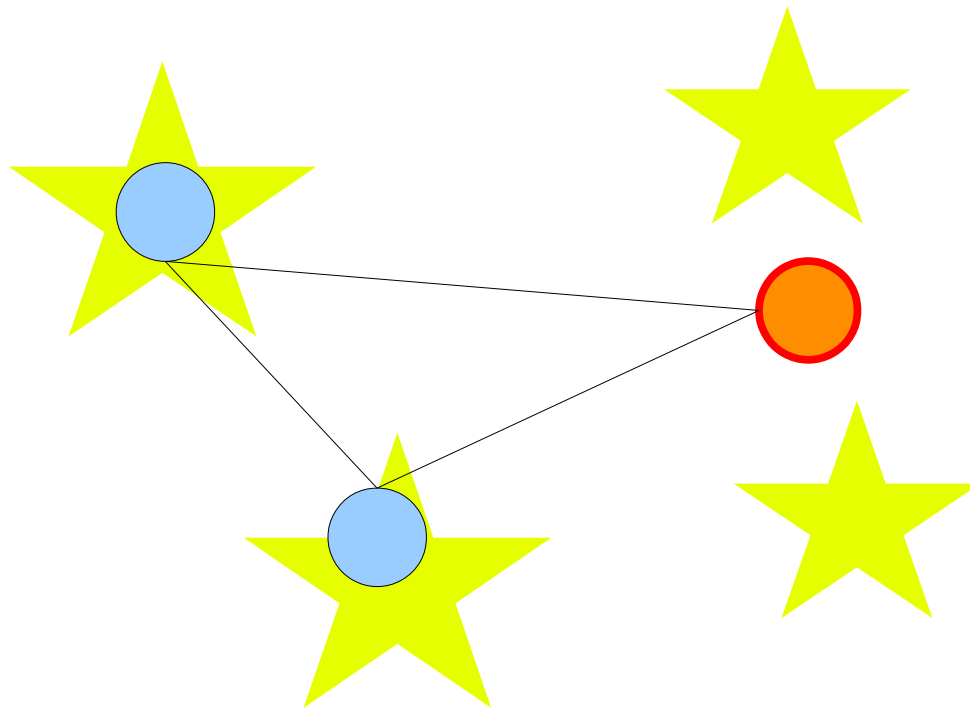
Questo tipo di reti svolge compiti di **clustering**

Dinamicità della topologia



Quale neurone avrà accumulato l'errore locale maggiore?

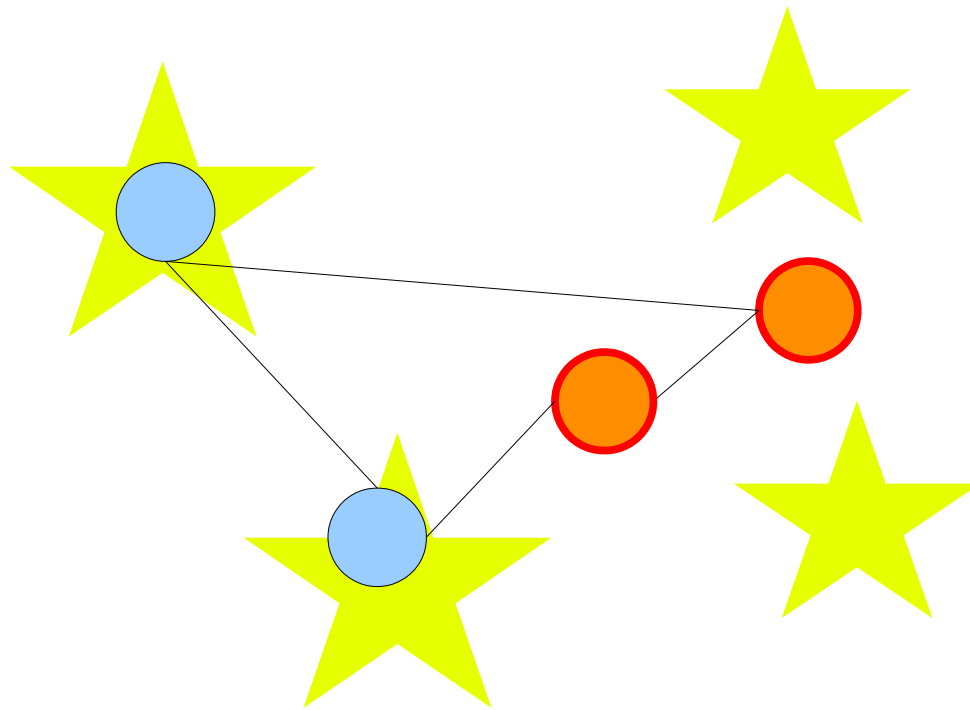
Dinamicità della topologia



Quale neurone avrà accumulato l'errore locale maggiore?

Vuol dire che quel neurone **da solo non basta** a catturare la regione di sua competenza bene quanto gli altri catturano la propria

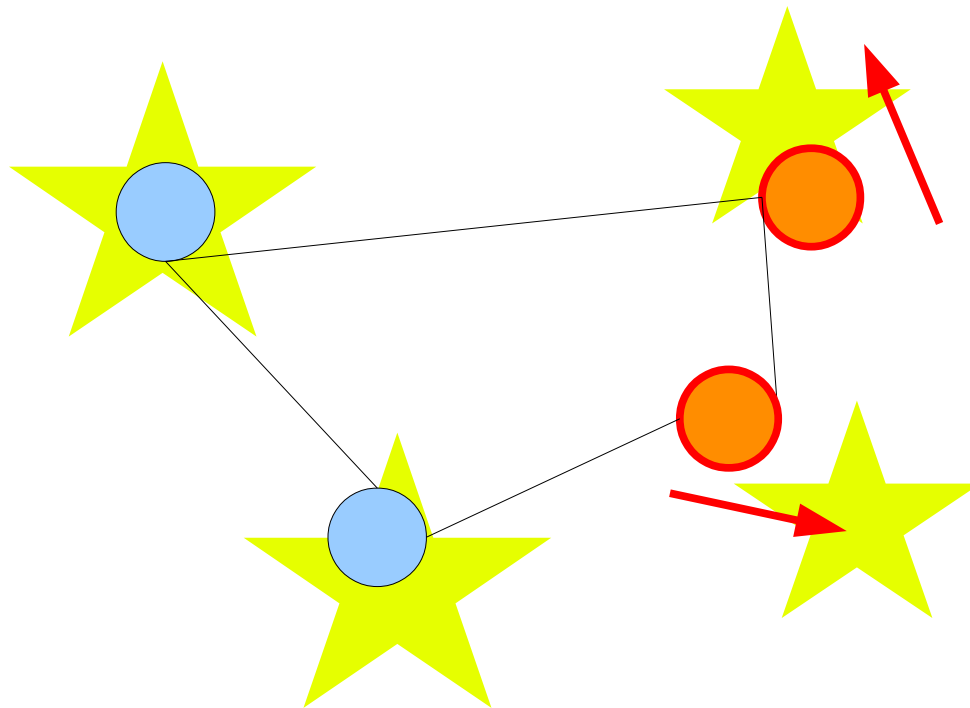
Dinamicità della topologia



Soluzione:

Inserire un nuovo neurone
lungo una delle sue connes-
sioni

Dinamicità della topologia



Soluzione:

Inserire un nuovo neurone lungo una delle sue connessioni

I due neuroni sono indipendenti e si spartiscono gli esempi della regione

Ragionando sulla base dello stesso principio è possibile decidere quando rimuovere connessioni e neuroni