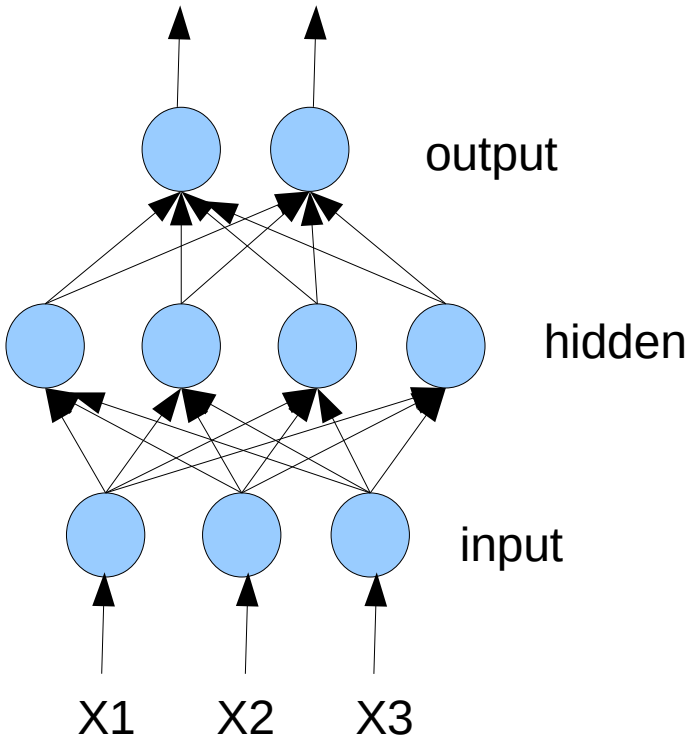
MLP e classificazione

Codifica delle classi

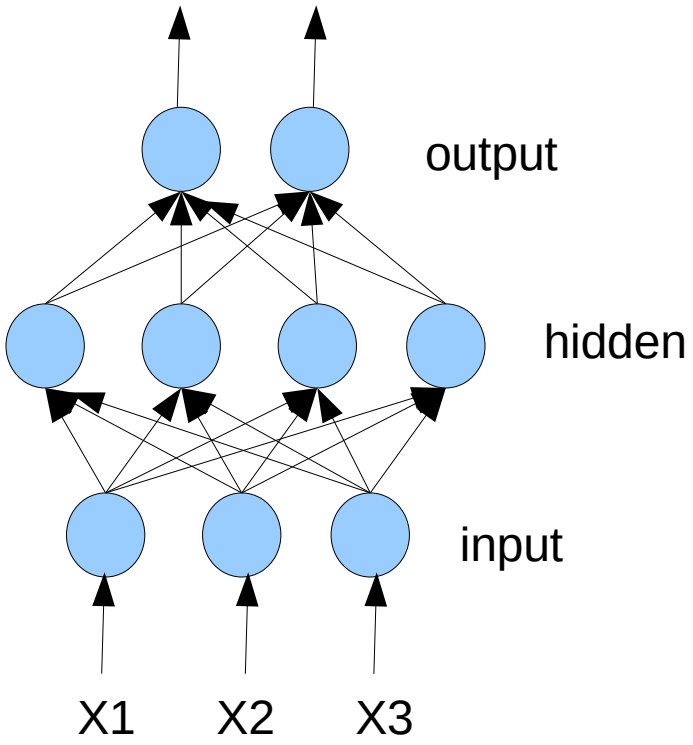


Valori dei diversi attributi che definiscono un'istanza

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

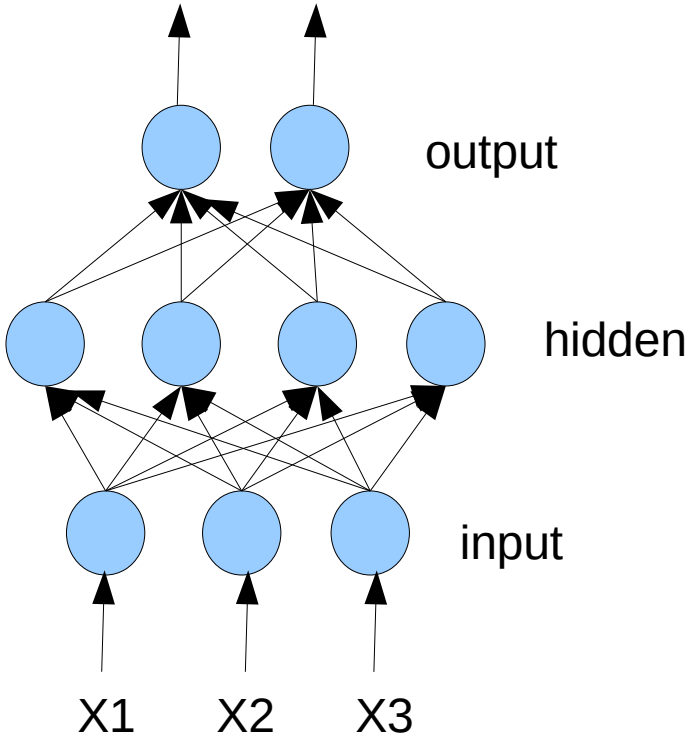


Valori dei diversi attributi che definiscono un'istanza

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

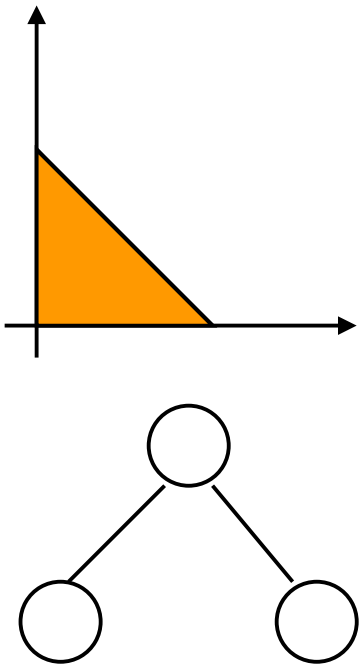
Alternativa

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

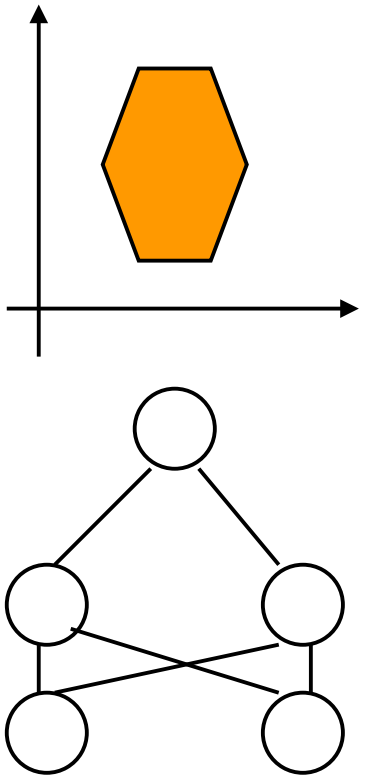
Esempio

Output rete	*Classe*
1 0	→ Classe A
0 1	→ Classe B

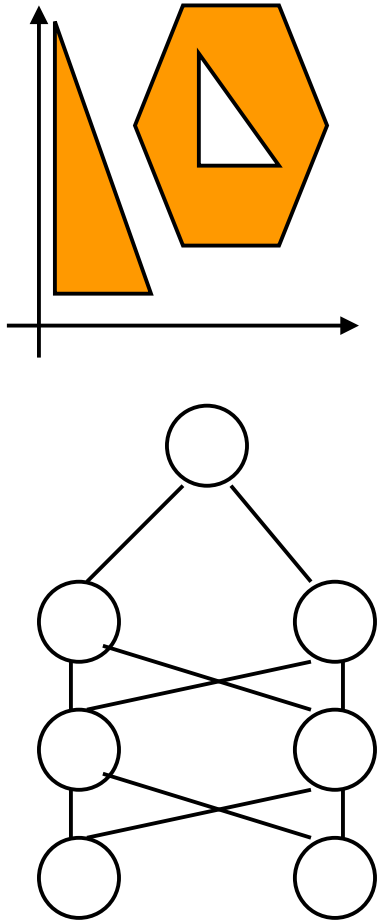
Compito degli hidden layer



Il primo layer
traccia dei confini



Il secondo layer
costruisce delle forme

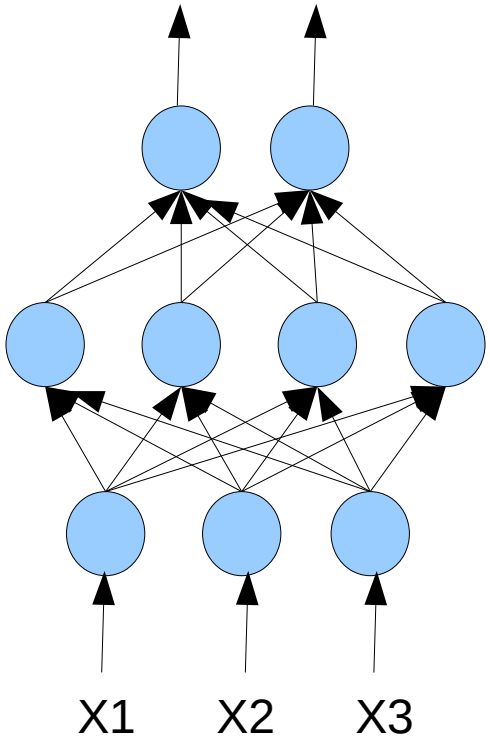


Il terzo layer
crea forme qualsivoglia
complesse

(immagine: Andrew Philippides, 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.



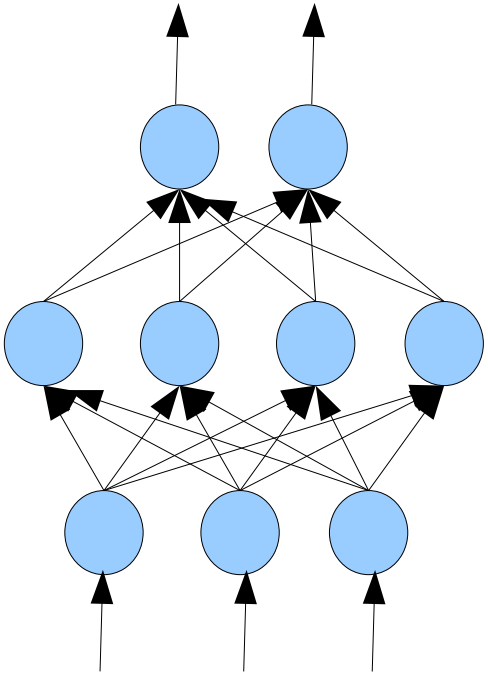
Dove viene memorizzata la **conoscenza** acquisita dalla rete neurale?

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**

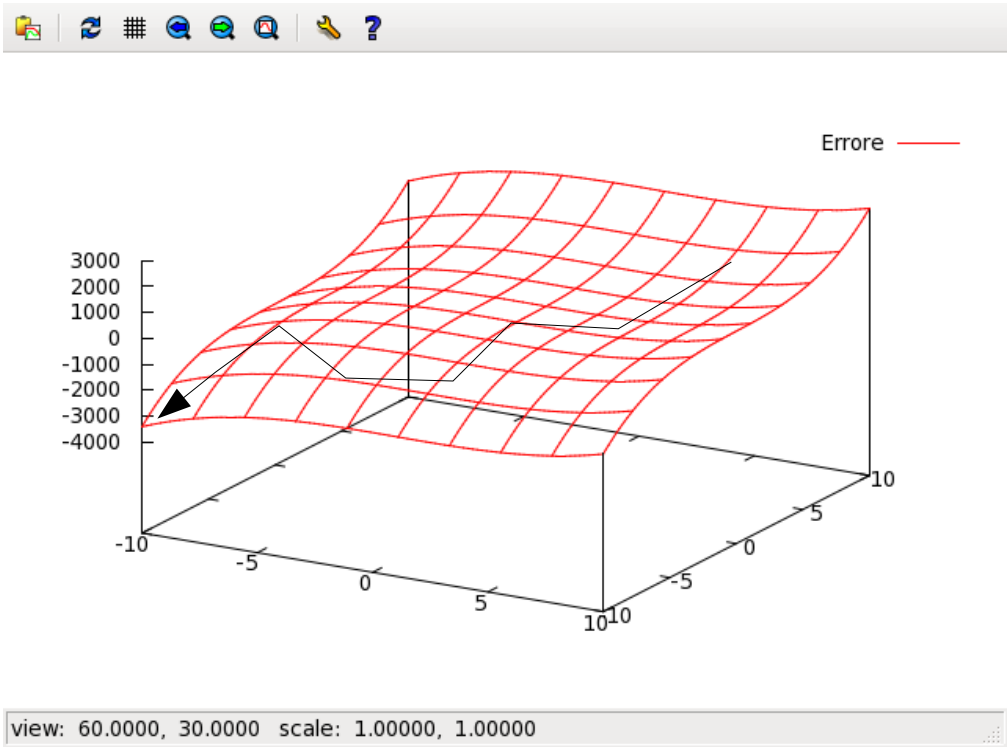
Discesa del gradiente



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



Discesa del gradiente

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

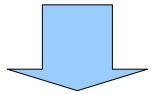
Ogni peso w_{ji} (su una connessione dal neurone *j-mo* al neurone *i-mo*) viene modificato sottraendo la derivata parziale dell'errore rispetto al peso stesso

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

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

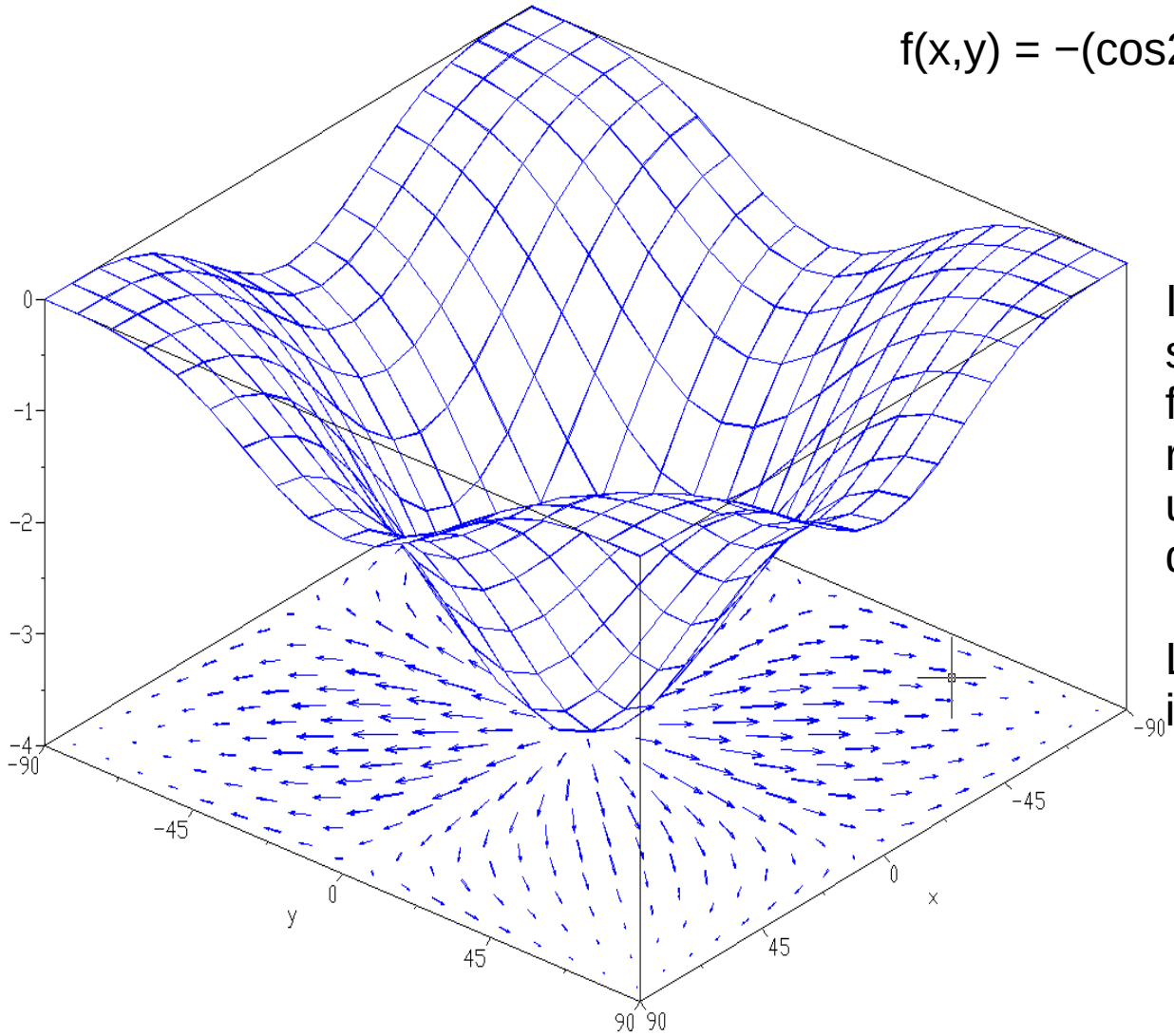
Funzione rispetto alle variabili su cui è calcolata



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

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

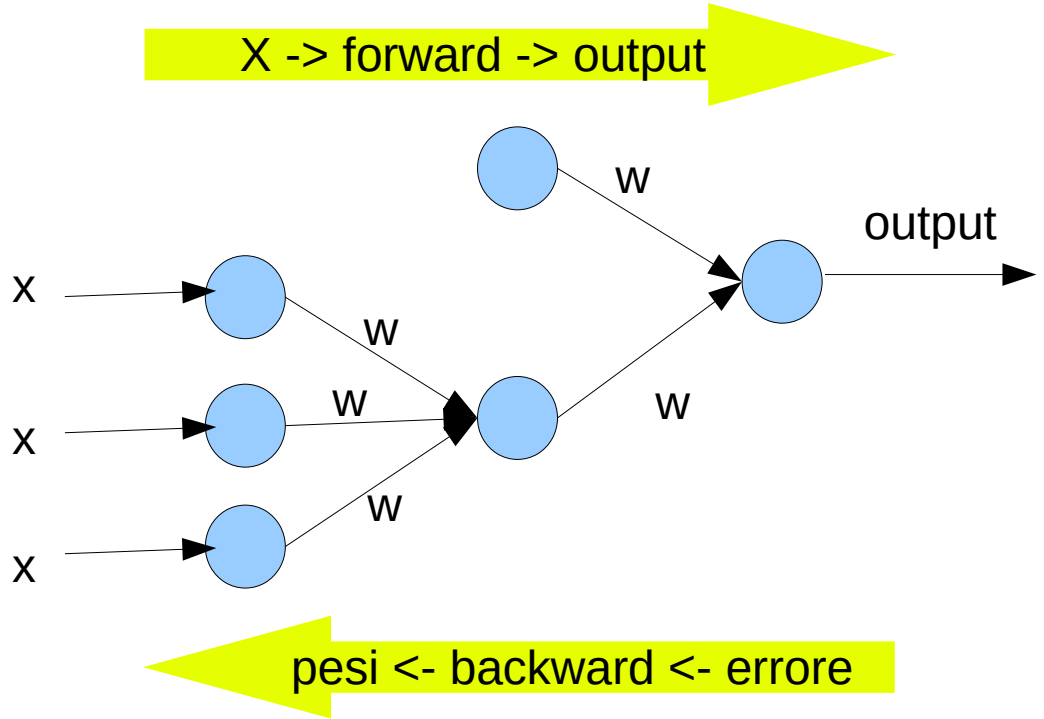
http://en.wikipedia.org/wiki/File:Gradient99.png

Discesa del gradiente

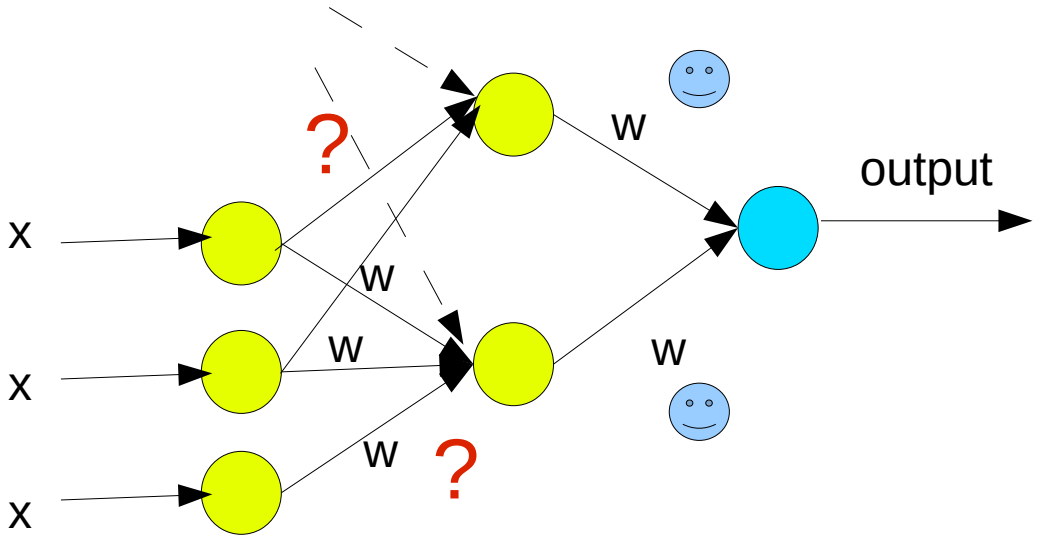
$$\Delta w_{ji} = -\lambda \frac{\partial E(\overline{w})}{\partial w_{ji}}$$

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**?

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

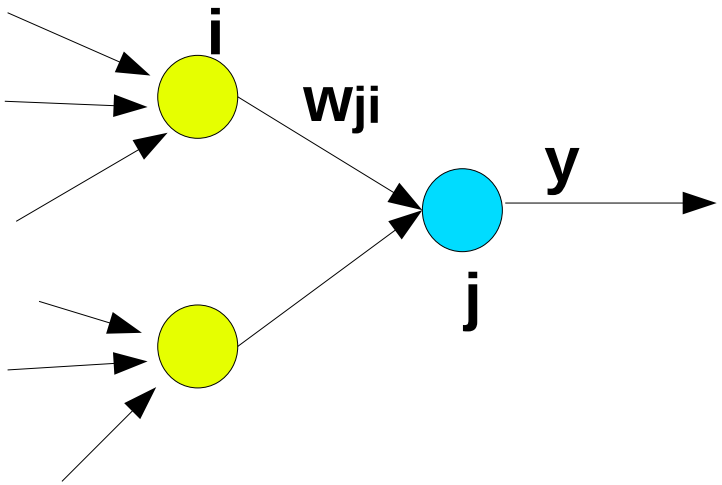
Nota: la forma analitica serve per calcolare le derivate

Errore globale dell'MLP

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

p = numero dei neuroni di output
t_i = valore desiderato per il nuerone di output i-mo
y_i = valore prodotto dal neurone di output i-mo

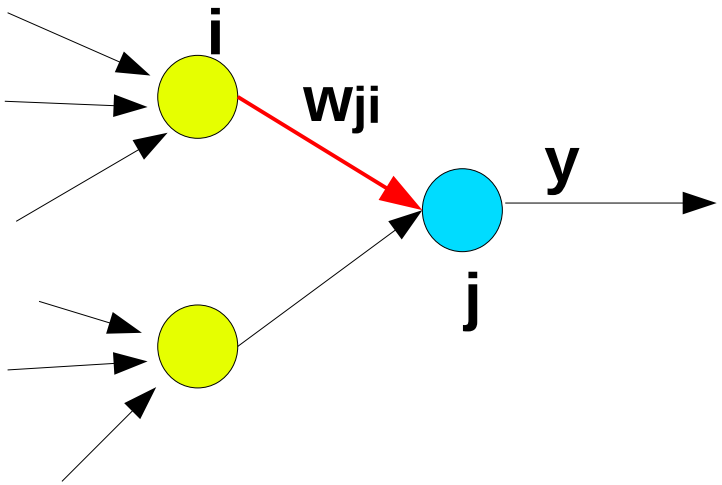
Backpropagation: neuroni di output



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
$\mathbf{x_{ji}}$ = il valore inviato dal neurone *i* al neurone *j* che ha contribuito a produrre *y*
$\mathbf{w_{ji}}$ = is 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:

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

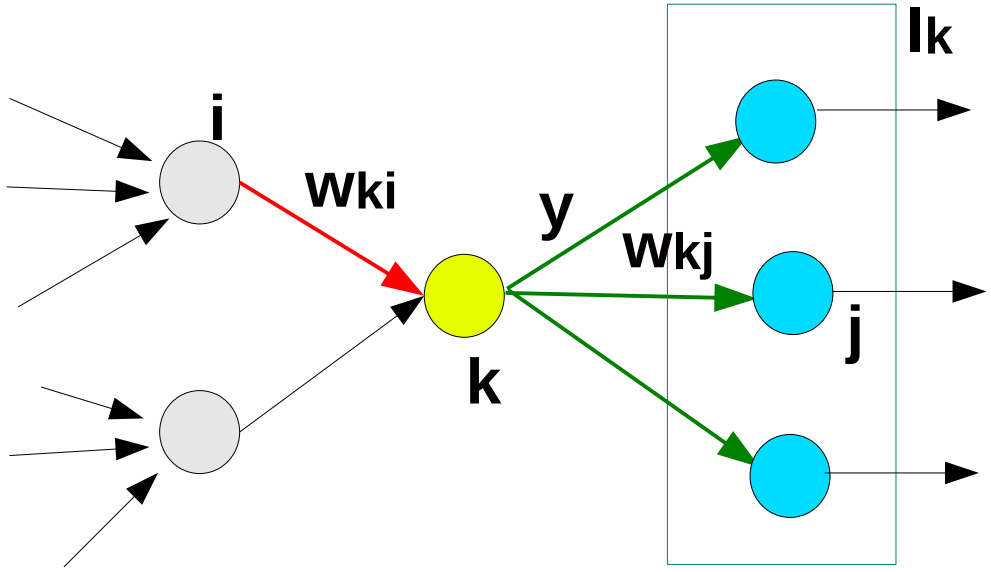
Variazione da applicare al peso

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

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 retropro-pagazione 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***

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

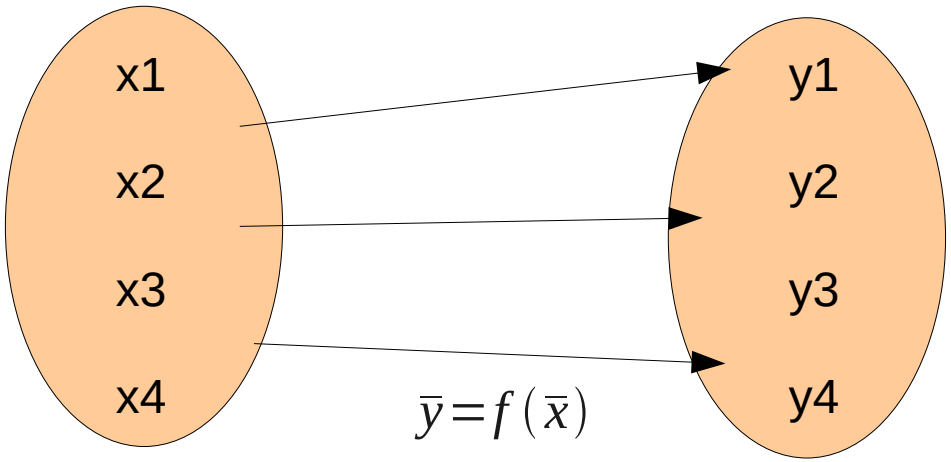
La forma analitica della funzione non è nota
occorre indurla da esempi

Learning set

Come per la classificazione l'apprendimento è supervisionato

	Vrb indipendenti			Vrb dipendente
	X1	**...**	**Xn**	**f**
i1	v11	...	v1n	y1
i2	v21	...	v2n	y2
i3	...			...
...				
	...		...	...
ik	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