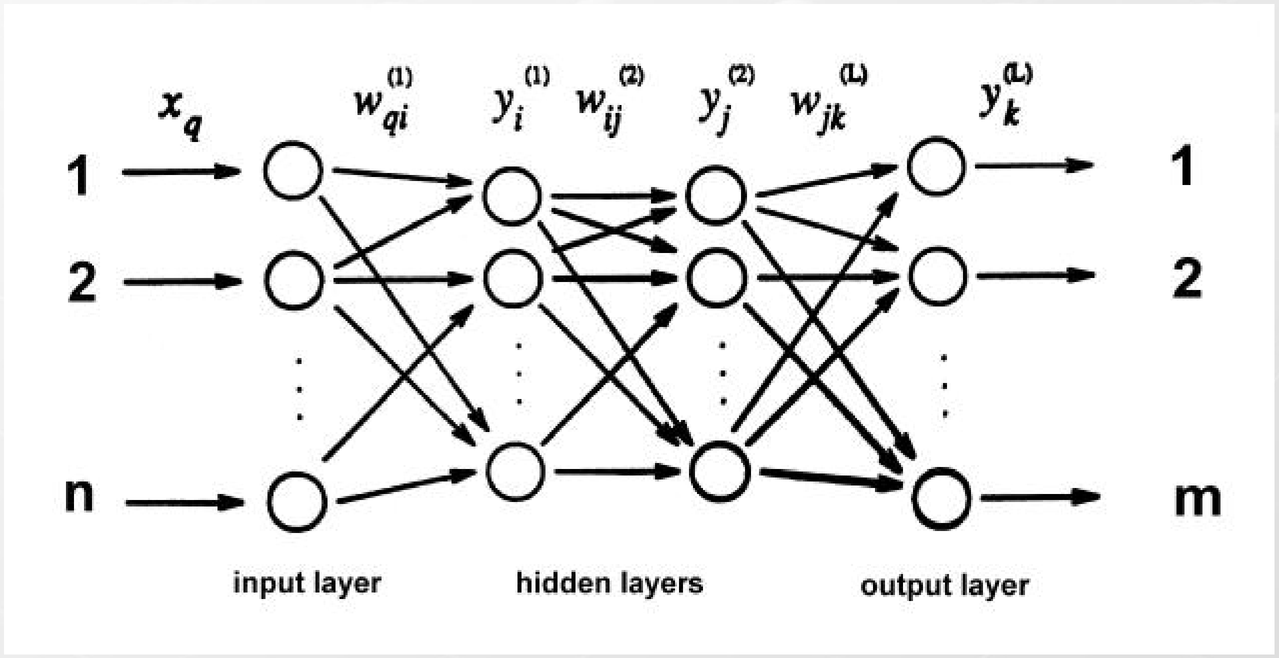
Backpropagation Learning Algorithm

- An algorithm for learning the weights in the network, given a training set of input-output pairs $\{ x^{\mu}, o^{\mu} \}$
- The algorithm is based on gradient descent method.
- Architecture



$w_{ij}^{(l)}$: Weight on connection between the i^{th} unit in layer (l-1) to j^{th} unit in layer l

Supervised Learning

Supervised learning algorithms require the presence of a “teacher” who provides the right answers to the input questions.

Technically, this means that we need a *training set* of the form

$$L = \left\{ \left(\bar{x}^{1}, \bar{y}^{1} \right), \quad ..... \quad \left(\bar{x}^{p}, \bar{y}^{p} \right) \right\}$$

where :

- $\bar{x}^{h} \quad \left(h = 1 \mathbf{K}\; p \right)$ is the network input vector
- $\bar{y}^{h} \quad \left(h = 1 \mathbf{K}\; p \right)$ is the network output vector

Supervised Learning

The learning (or training) phase consists of determining a configuration of weights in such a way that the network output be as close as possible to the desired output, for all examples in the training set.

Formally, this amounts to minimizing the following *error function* :

$$
\begin{aligned}
E &= \frac{1}{2}\sum_{h=1}^{p} \left\| \overline{out}_h - \overline{y}_h \right\|_2^2 \\
&= \frac{1}{2}\sum_{h}\sum_{k} \left(out_k^h - y_k^h \right)
\end{aligned}
$$

where $\overline{out}_h$ is the output provided by the network when given $\overline{x}^h$ as input.

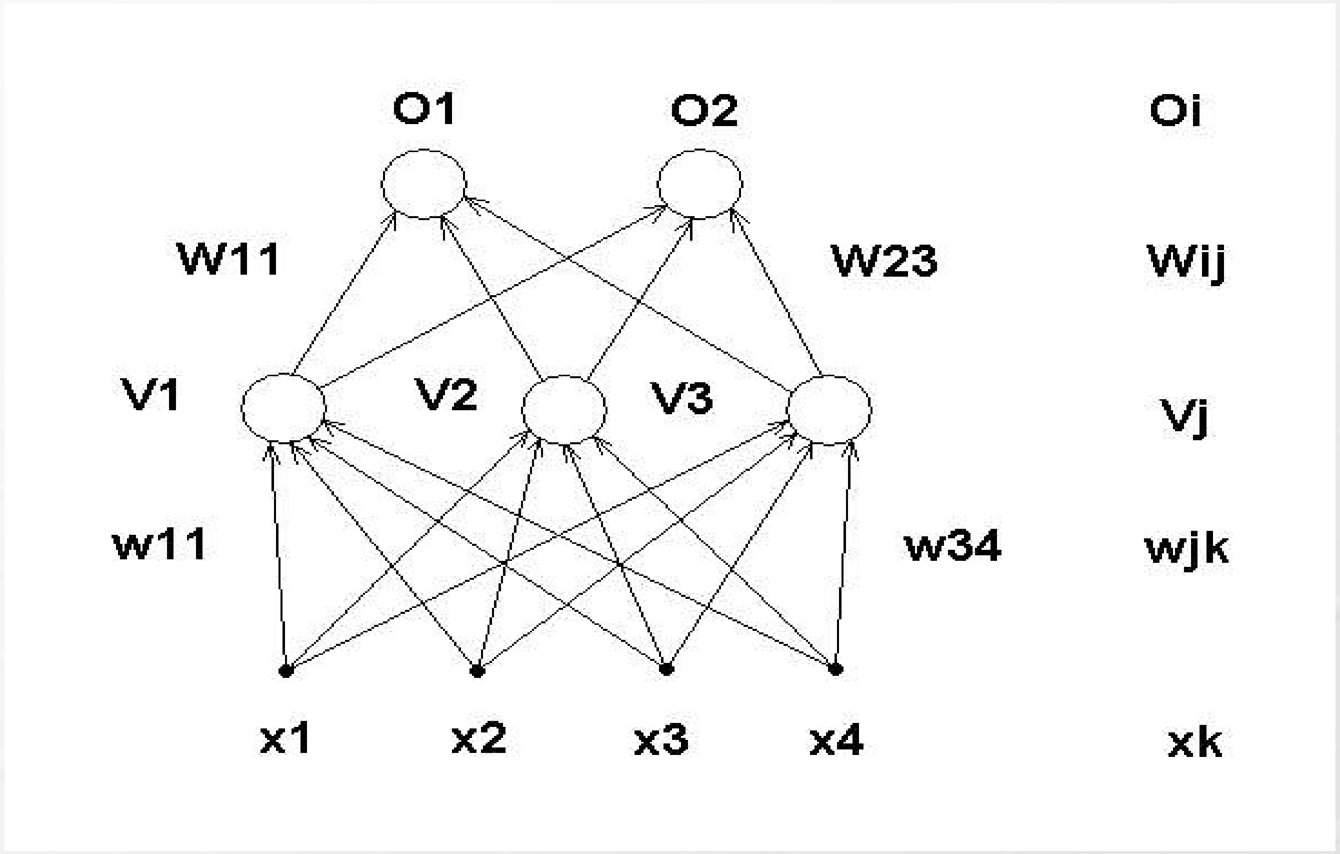
Back - Propagation

To minimize the error function *E* we can use the classic gradient – descent algorithm.

To compute the partial derivates $\partial E / \partial w_{ij}$, we use the *error back propagation* algorithm.

It consists of two stages:

- *Forward pass* : the input to the network is propagated layer after layer in forward direction
- *Backward pass* : the “ error ” made by the network is propagated backward, and weights are updated properly



Dato il pattern *μ*, l’unità nascosta *j* riceve un input netto dato da

$$h_j^m = \sum_k w_{jk} \, x_k^m$$

e produce come output :

$$V_j^m = g\left(h_j^m\right) = g\left(\sum_k w_{jk} \, x_k^m\right)$$

Back-Prop : Updating Hidden-to-Output Weights

$$\begin{aligned}
\Delta W_{ij} &= -h \frac{\partial E}{\partial W_{ij}} \\
&= -h \frac{\partial}{\partial W_{ij}} \left[\frac{1}{2} \sum_m \sum_k \left(y_k^m - O_k^m \right)^2 \right] \\
&= h \sum_m \sum_k \left(y_k^m - O_k^m \right) \frac{\partial O_k^m}{\partial W_{ij}} \\
&= h \sum_m \left(y_i^m - O_i^m \right) \frac{\partial O_i^m}{\partial W_{ij}} \\
&= h \sum_m \left(y_i^m - O_i^m \right) g'\left(h_i^m \right) V_j^m \\
&= h \sum_m d_i^m V_j^m
\end{aligned}$$

where: $d_i^m = \left(y_i^m - O_i^m \right) g'\left(h_i^m \right)$

Back-Prop : Updating Input-to-Hidden Weights (I)

$$\Delta w_{jk} = -h \frac{\partial E}{\partial w_{jk}}$$

$$= h \sum_m \sum_i \left(y_i^m - O_i^m\right) \frac{\partial O_i^m}{\partial w_{jk}}$$

$$= h \sum_m \sum_i \left(y_i^m - O_i^m\right) g'\left(h_i^m\right) \frac{\partial h_i^m}{\partial w_{jk}}$$

$$\frac{\partial h_i^m}{\partial w_{jk}} = \sum_l W_{il} \frac{\partial V_l^m}{\partial w_{jk}}$$

$$= W_{ij} \frac{\partial V_j^m}{\partial w_{jk}}$$

$$= W_{ij} \frac{\partial g\left(h_j^m\right)}{\partial w_{jk}}$$

$$= W_{ij}\, g'\left(h_j^m\right) \frac{\partial h_j^m}{\partial w_{jk}}$$

Back-Prop : Updating Input-to-Hidden Weights (II)

$$\frac{\partial h_j^m}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}} \sum_m w_{jm} x_m^m$$

$$= x_k^m$$

Hence, we get :

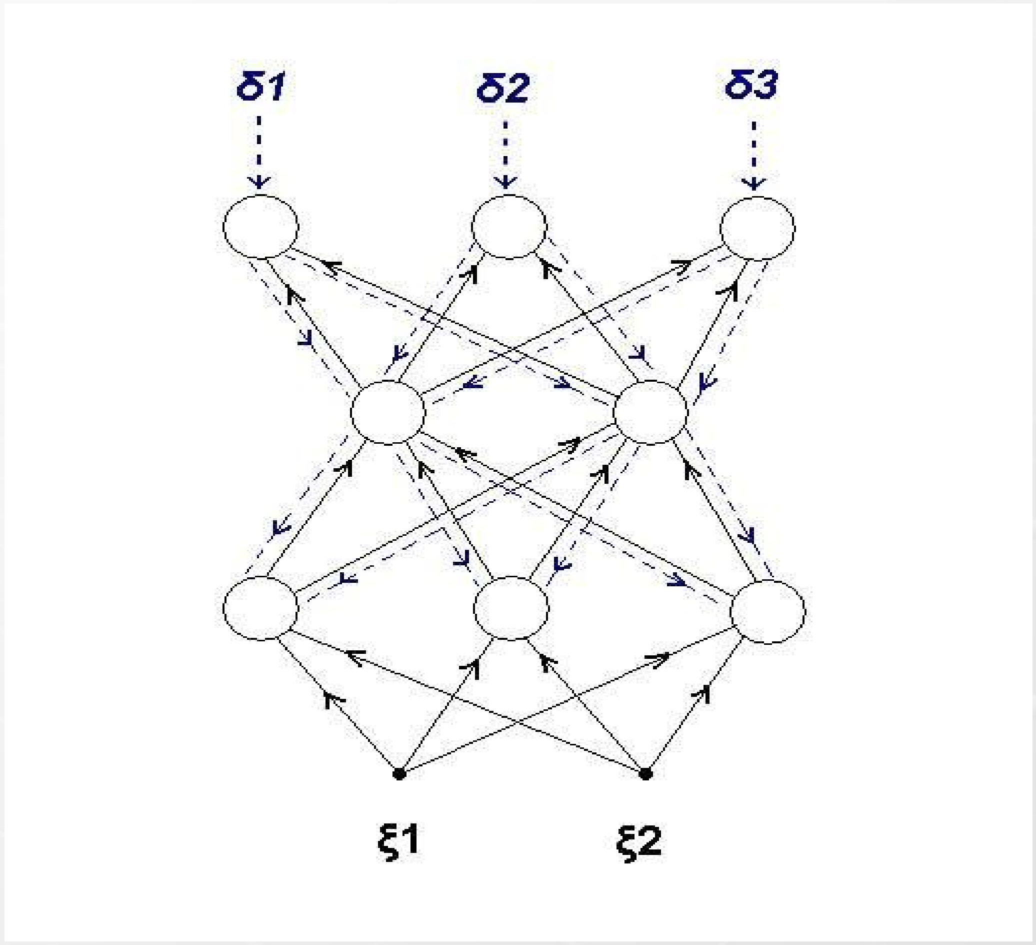
$$\Delta w_{jk} = h \sum_{m,i} \left(y_i^m - O_i^m\right) g'\left(h_i^m\right) W_{ij} \; g'\left(h_j^m\right)$$

$$= h \sum_{m,i} d_i^m \; W_{ij} \; g'\left(h_j^m\right) x_k^m$$

$$= h \sum_m \hat{d}_j^m \; x_k^m$$

where :

$$\hat{d}_j^m = g'\left(h_j^m\right) \sum_i d_i^m \; W_{ij}$$



Retropropagazione dell’errore :

- le linee nere indicano il segnale propagato in avanti
- Le linee blu indicano l’errore (i *δ*) propagato all’indietro