A simple MLP

Tiefei-Zhang
tfzhang@zju.edu.cn

1. Introduction

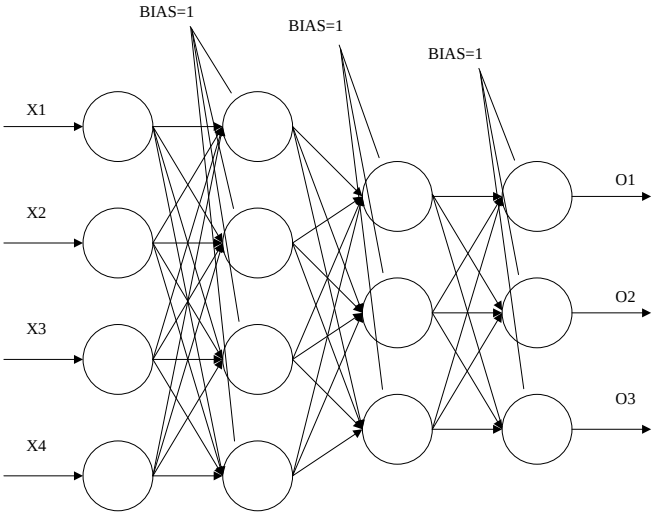
This is a simple multi-layer perceptron. It is used to classify types of plants based on four attributes. It uses the simple BP(Back Propagation) learning strategy.

2. The Data Set Used

The adopted data set is Iris Data Set, one of the most popular data set. Assume the input is four vectors, which denotes the attribute of the plant. The output is three vectors with the value “1” or “0” to denote which class the plant belongs to. For example, the [1, 0, 0] means the plant belongs to class-Iris Setosa, and [0, 1, 0] the second class-Virginica, the same is true with [0, 0, 1].

3. The Neuron Network Architecture:

The Neuron Network adopted is a 4-4-3-3 network, which means the network consists of one input layer with four units, two hidden layers with 4 and 3 unites respectively, and finally the output layer with three units.



4. The Learning Rule Adopted

3.1 The activation function:

$$f(net) = \frac{1}{1+\exp(-1*sum)}$$

3.2 The Error of output layer:

$$Error = \sum_{i \in M} \frac{1}{2} \times E_i^{\,2},\ for\ M, the\ units\ of\ output\ layer$$

$$E_i = 1 \times o_i \times (1 - o_i) \times (d_i - o_i)$$

3.3 The weight adjustment for single layer:

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial (net_k)} \bullet \frac{\partial (net_k)}{\partial w_{kj}}$$

$$\frac{\partial (net_k)}{\partial w_{kj}} = y_j \qquad (1)$$

$$E(net_k) = E[o_k(net_k)]$$

$$\frac{\partial E}{\partial (net_k)} = \frac{\partial E}{\partial o_k} \bullet \frac{\partial o_k}{\partial (net_k)}$$

$$\frac{\partial E}{\partial o_k} = -(d_k - o_k) \qquad (2)$$

$$\frac{\partial o_k}{\partial (net_k)} = f'(net_k) \qquad (3)$$

$$from (1),(2),(3):$$

$$\frac{\partial E}{\partial w_{kj}} = -(d_k - o_k)\ f'(net_k) y_j$$

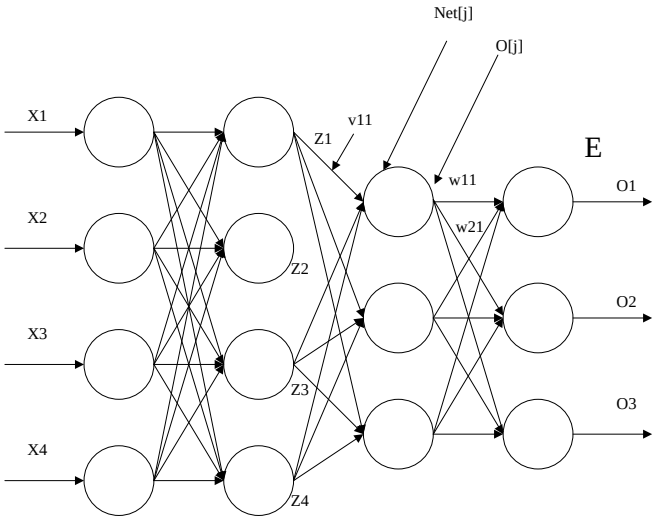
$$In\ simple\ form:$$

$$\frac{\partial E}{\partial w_{kj}} = -\delta_{ok} y_j,\ \delta_{ok} = (d_k - o_k)\ f'(net_k)$$

$$if\ \ f'(net_k) = \frac{1}{1+\exp(-net)},\ f'(net_k) = o_k \times (1 - o_k)$$

$$\Delta w_{kj} = -\eta \frac{\partial E}{\partial w_{kj}} = \eta \delta_{ok} y_j, \delta_{ok} = (d_k - o_k) \times o_k \times (1 - o_k)$$

$$w'_{kj} = w_{kj} + \Delta w_{kj}$$



3.4 The weight adjustment for hidden layer:

$The\ same\ formular\ \eta\delta_{ok}y_j\ can\ be\ used$:

$\eta\delta_{yj}z_i$

$$\delta_{yj}=\frac{\partial E}{\partial(net_j)}=\frac{\partial E}{\partial(o_j)}\bullet\frac{\partial o_j}{\partial(net_j)}$$

$$\frac{\partial o_j}{\partial(net_j)}=f_j^{'}(net_j)$$

$$\frac{\partial E}{\partial(o_j)}=\frac{\partial}{\partial y_j}(\frac{1}{2}\sum_{k=1}^{K}\{d_k-f[net_k(y)]\}^2),\ \mathrm{K=3}$$

$$\frac{\partial E}{\partial(o_j)}=-\sum_{k=1}^{K}(d_k-o_k)\frac{\partial}{\partial y_j}\{f[net_k(y)]\}$$

$$\frac{\partial}{\partial y_j}\{f[net_k(y)]\}=f_k^{'}(net_k)\frac{\partial(net_k)}{\partial y_j}=f_k^{'}(net_k)w_{kj}$$

$$\frac{\partial E}{\partial(o_j)}=-\sum_{k=1}^{K}(d_k-o_k)f_k^{'}(net_k)w_{kj}$$

$$\eta\delta_{yj}z_j=\eta f_j^{'}(net_j)z_i\sum_{k=1}^{K}(d_k-o_k)f_k^{'}(net_k)w_{kj}$$

$$\delta_{yj}=f_j^{'}(net_j)\sum_{k=1}^{K}(d_k-o_k)f_k^{'}(net_k)w_{kj}$$

$Compare\ to\ the\ output\ layer, which\ E_i=o_i\times(1-o_i)\times(d_i-o_i)$

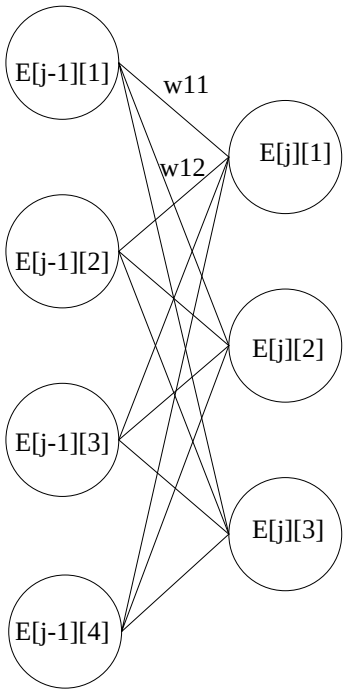
$So\ the\ error\ of\ node\ j:\sum_{k=1}^{K}E_i\times w_{kj}$

3.5 The relationship of error between two subsequent layers:

From above, we can obtain the relationship of error between two subsequent layers:

$$E[j-1] = E[j] \times w[j][j-1]$$

That is: each error for the unit of the lower layer is related to the error of all upper layer units. For example, E[j-1][1]=E[j][1]*w[1][1]+E[j][2]*w[2][1]+E[j][3]*w[3][1];



5. About the source file:

The main source file is mybpn.c, and it is compiled successfully under ubuntu9.04 with gcc-4.3.3. The compile command is as follows: gcc mybpn.c –lm. The “-lm” should be attached, otherwise link errors will occur.

6. The Result:

After the testing data set trained for 200 times, the outcome becomes stable. The result is as follows:

	Training Accuracy (When trained with 150 items and 200 training times)	Testing Accuracy (When trained and tested with same number 75, besides with 200 training times
Iris	(150-4)/150=97.33%	(75-5)/75=93.33%