Pseudocode for MLP Backprop Learning

Constants:

```
N_In  = 100      # of input nodes
N_Hid = 10       # of hidden nodes
N_Out = 2        # of output nodes
Eta   = 0.2      learning rate parameter
Exemp = 20       # of training examples
```

Data Structures:

Input[N_In]
Hidden[N_Hid]
Output[N_Out]
IH_Weights[N_In,N_Hid]
HO_Weights[N_Hid,N_Out]
Stim[Exemp,N_In]
Resp[Exemp,N_Out]

Some Tools:

Random_Vector(Vector,N)
Update_Weights(Weights, Error, X, Y)
Squash(Activation)

$$\Delta w_{jk} = \eta \delta_k h_j$$

$$o_k = \frac{1}{1 + \exp^{-net_k}}$$

Initializing random weights:

```
do i = 1, N_In
  Random_Vector(IH_Weights[i,:], N_Hid)
enddo

do i = 1, N_Hid
  Random_Vector(HO_Weights[i,:], N_Out)
enddo
```

Presenting a training exemplar and determining output:

Input = Stim[Rand,:]

! Determining hidden layer activation
do j = 1, N_Hid
  do i = 1, N_In
    Hidden[j] = Hidden[j] + Input[i] * IH_Weight[i,j] $h_j = \sum_{i=1}^{I} i_i w_{ji}$
  enddo
enddo

Sqaush(Hidden)

! Determining output layer activation
do k = 1, N_Out
  do j = 1, N_Hid
    Ouput[k] = Output[k] + Hidden[j] * HO_Weight[j,k]
  enddo
enddo

Sqaush(Output)

Determining Error for Output Layer:

do k = 1, N_Out
    Delta[k] = Resp[Rand,:] – Output[k]
enddo

Update_Weights(HO_Weight, Delta, N_Out, N_Hid)

Determining Error for Hidden Layer:

do j = 1, N_Hid
  do k = 1, N_Out
    Out_Error = Out_Error + HO_Weight[j,k]*Delta[k] $\sum_{k=1}^{K} w_{kj}\delta_k$
  enddo
 Delta[j] = Hidden[j]*(1-Hidden[j])*Out_Error $\delta_j = h_j(1-h_j)\sum_{k=1}^{K} w_{kj}\delta_k$
enddo

Update_Weights(IH_Weight, Delta, N_Hid, N_In)

….AND THEN DO IT AGAIN UNTIL YOU REACH A CRITERION