**Back-propagation Algorithm**

Back propagation is a systematic method in which training of multilayer artificial neural network is carried out.It is built on high dimensional mathematical foundation. Though back propagation has its own limitations,it can be used for the wide range of practical problem. And the result are successfully demonstrated its power. Here, Input, hidden and output layer are conneted to each other to get proper training and testing neural network.

* Input layer:- use linear transfer function.(g=tanØ=1)
* Hidden and output layer:- use sigmoidal or squashed s-function

Sigmoidal function is given as-

**Effect of tuning parameters of the backpropagation nerural network:**

For efficient learning and designing of a stable network,it is necessary to select tuning parameters such as –

* Momentum factor
* Learning coefficient
* Sigmoidal gain
* Threshold value

Momentum coefficients(α):

The goal of momentum factor is to raise the speed of learning without leading oscillations.It is used to improve the rate of convergence. Its value should be positive but less than 1.typical value lie between the range[0.5 0.9]. the most suitable factor is found α=0.9 . it is used to reduce train time .It helps to enhance the network.

Learning Coefficient(η):

It can not be negative since it will cause the change of weight vector postions.η should be positive. If η=2.network is unstable and if η>1, the weight vector overshoot from its ideal positions and oscillate. Thus the range of learning coefficient must lie between [0 1]. The best value for η is 0.6 which gives less error.

The empirical formula is given by-

Where, N is the numbers of pattern

M is number of different pattern type

**Backpropogation algorithm**

Let us consider the neural network in which input layers having ‘*l*’ nodes,hidden layers having ‘*m*’ nodes and output layers having ‘*n*’ nodes. The number of neurons in the hidden layer must lie between [0 21]. The basic loop structure for back-propagation algorithm is as below:

*Initialize the weights of given embedded system;*

*Define tunning parameter for embedded system;*

*Repeat*

*For each train pattern*

*Train the pattern;*

*Select desired value ;*

*End;*

*Do this*

*Until the error is low ;*

*check it with tolerance of given system;*

**Illustration on train and testing of different embedded system:**

Now, we are making the embedded system in which training and testing is done. Once the process convergences, we can stored the final weights on the file . Consider the following testing tables. In this system we are taking three inputs and one output. Since the range of input lie between [0 255]. We need to normalize the input. There is no need to normalize the output since our system’s output is in the form of 1 and 0.

**Normalization :**

Normalization is done in back-propagation because it is proved that neural networks works better in the range 0-1 of input and output values. Two tyes of Normalization:-

1. Max-min normalization
2. Decimal sacling

We performed max-min normalization .For given system A, we denoted the max\_value and min\_value. The new range of New\_max\_value and new\_min\_value of V is scaled.

Here, In our embedded system A has range[0 255] and V has range[-1 1].

Formula is as.

Table 1:Consider the following testing sets of embedded system.

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Input 1 | Input 2 | Input 3 |
| 1. | 0 | 6 | 131 |
| 2. | 0 | 6 | 195 |
| 3. | 64 | 128 | 131 |
| 4. | 128 | 128 | 194 |
| 5. | 0 | 128 | 194 |
| 6. | 32 | 128 | 193 |
| 7. | 64 | 128 | 195 |
| 8. | 0 | 6 | 195 |
| 9. | 160 | 128 | 67 |
| 10. | 0 | 28 | 66 |

Calculations:

With the data of the first testing set,

I1=0; I2=6 ;I3=131, o=1

Normalization of input:

Using the formula.

Range of A=[0 255];

Range of A’=[-1 1];

for Input *I2=v=*6 ;

*v’*= -0.9529

for Input *I3=v=*131

*v’*= 0.0274;

[\\diagram](file:///\\diagram) for the system………………….

Step 1:After normalization, Initialize the Inputs and normalized form. After using linear transformation, output of input layer is same as input values. Since g=tanØ=1.

Step 2: Assume the neurons in the hidden layer to lie between 0 to 21.For given embedded system,λ is taken as 1 and threshold values equal to zero.[*V*] is the weights of synapses connecting input neurons and hidden neurons and [*w*] is the weight of synapses connecting between hidden neuron and output neuron.

Initialize the weights to small random values in the range [-1 1] as

Step 3:Find Hidden layer Input.

=

Step 4:Find hidden layer output using sigmoidal function.

=

Step 5:Computation of inputs to the output layer by multiplying corresponding weights of synapses as

Step 6: Evaluate the output layer units using sigmoidal function.

Step 7: Calculate the error for the output=1

Error=

Step 8:Let us adjust the weight

=

Step 9: Assume η=0.6 and α=0.9

=0.9

Step 10:

Step 11:

Step12:

Step13:

=

Step 14:

Step 15:With the updated weights [V] and [W] ,error is calculated again and next training set is taken and the error will be adjusted.Now our error=0.1389 which is greater than our system’s tolerance so we are updating weights to get appropriate output with minimum error.

Step 16:The Updated hidden inputs and output

Step 17:The new Input and Output to the output layer neuron with updated weights.

0.6200

Step 18:The error with updated weights is as

=

Now ,this error is minimum so this weights can be used for further calculations. We calculated this error when our embedded system’s output is one. Now ,we required the error when output is zero. The calculations are as follows. From Step 1 to step 6 all calculations are same.so following steps are calculated for system’s output is zero.

Step 19: Error when O=0;

Error=

This error is minimum than our desired to tolerance. Further for our sake of satisfaction we are updating weights to get minimum error than this.

=0.9

=

0.5321

Error=

Here, error with updated weights is higher than previous one so There is no need for updated weight for output=0; For observation we again updated the weights and we got error =0.1420. New Error is again greater than previous one.

**Table 2: Observation for given Inputs:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Inputs | Outputs | Iteration | Remark | Error value | observation |
| I1=0;  I2=-0.9529;  I3=0.0274 | 1 | 3 |  | 0.1389 | More updated weights give you less error |
| 0.0722 |
| STOP | 0.0623 |
| 0 | 5 | STOP | 0.11181 | No need for updated weights |
|  | 0.1415 |
| 0.1420 |
| 0.1427 |
| 0.1500 |

Graph 1: Iteration versus error rate

Follow the steps and calculate for remaining inputs.The following table gives all the result.After the observation,the values in training pattern and testing pattern are found near about same. So we have taken some values from training pattern and some value from testing pattern.We also observed that our output is either one or zero.

**Table 3:Training and Testing Data for embedded System**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sr. no. | Inputs | | | | Actual Output | Iteration | Calculated Outputs | Error | Observation |
| I1 | I2 | I3 | |
| 1. | 0 | 6 | 131 | | 1 | 3 | 0.4729 | 0.1389 | Updated weights gives more accuracy. Error can be minimized upto 0 after increasing the no. of iteration. |
| 0.6200 | 0.0722 |
| 0.7043 | 0.04371 |
| 2. | 0 | 6 | 195 | | 1 | 4 | 0.4685 | 0.1412 |
| 0.4987 | 0.1256 |
| 0.5012 | 0.1244 |
| 0.5150 | 0.1176 |
| 3. | 64 | 128 | 131 | | 1 | 1 | 1.1665 | 0.01386 |
| 4. | 128 | 128 | 194 | | 1 | 1 | 0.45800 | 0.14688 |
| 5. | 0 | 128 | 194 | | 1 | 3 | 0.4226 | 0.16666 |
| 0.4520 | 0.1501 |
| 0.508481 | 0.1207 |
| 6. | 0 | 6 | 131 | | 0 | 5 | 0.4729 | 0.11181 | Accuracy is obtained in first iteration. So no more weights updating is required specially when output=0 |
| 0.5321 | 0.1415 |
| 0.5329 | 0.14199 |
| 0.5356 | 0.1434 |
| 0.6001 | 0.18006 |
| 7. | 0 | 6 | 195 | | 0 | 2 | 0.4685 | 0.1097 |
| 0.4987 | 0.1243 |
| 8. | 64 | 128 | 131 | | 0 | 1 | 1.1665 | 0.68036 | This is the worst case for given embedded system. |
| 9. | 128 | 128 | 194 | | 0 | 2 | 0.45800 | 0.104882 | Accuracy is poor. |
| 0.5001 | 0.1250 |
| 10. | 0 | 128 | | 194 | 0 | 2 | 0.457996 | 0.104836 |
| 0.5023 | 0.1261 |