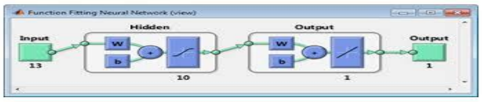


Neural Networks and Fuzzy Logic (630514)

Lecture 6

Multi-Layer Feedforward Neural Networks using matlab Part 2

Examples:

Example 1: (**fitting data**) Consider **humps** function in MATLAB. It is given by

```
y = 1 ./ ((x-.3).^2 + .01) + 1 ./ ((x-.9).^2 + .04) - 6;
```

Solution: **Matlab Code:**

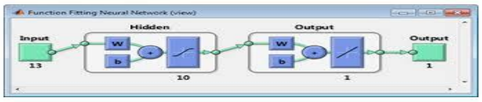
```
%Obtain the data:
x = 0:.05:2; y=humps(x);
P=x; T=y;
%Plot the data
plot (P, T, 'x')
grid; xlabel('time (s)'); ylabel('output'); title('humps function')
%DESIGN THE NETWORK
net = newff([0 2], [5,1], {'tansig','purelin'},'traingd');
% the first argument [0 2] defines the range of the input and initializes the network.
% the second argument the structure of the network, there are two layers.
% 5 is the number of the nodes in the first hidden layer,
% 1 is the number of nodes in the output layer,
% Next the activation functions in the layers are defined.
% in the first hidden layer there are 5 tansig functions.
% in the output layer there is 1 linear function.
% 'traingd' defines the basic teaching scheme – gradient method
% Define learning parameters
net.trainParam.show = 50; % The result is shown at every 50th iteration (epoch)
net.trainParam.lr = 0.05; % Learning rate used in some gradient schemes
net.trainParam.epochs =1000; % Max number of iterations
net.trainParam.goal = 1e-3; % Error tolerance; stopping criterion
%Train network
net1 = train(net, P, T); % Iterates gradient type of loop
% Resulting network is strored in net1
%Convergenceurve c is shown below.
% Simulate how good a result is achieved: Input is the same input vector P.
% Output is the output of the neural network, which should be compared with output data
a= sim(net1,P);
% Plot result and compare
plot (P, a-T, P,T); grid;
```

The fit is quite bad, to solve this problem:

- Change the size of the network (bigger. size).

```
net=newff([0 2], [20,1], {'tansig','purelin'},'traingd');
```

- Improve the training algorithm performance or even change the algorithm.



Try Levenberg-Marquardt – trainlm (more efficient training algorithm)

```
net=newff([0 2], [10,1], {'tansig','purelin'},'trainlm');
```

It is clear that L-M algorithm is significantly faster and preferable method to **back-propagation**.

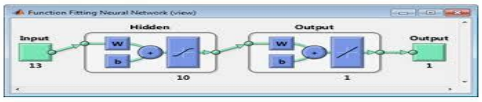
Try simulating with independent data.

```
x1=0:0.01:2; P1=x1;y1=humps(x1); T1=y1;
a1= sim (net1, P1);
plot (P1,a1, 'r', P1,T1, 'g', P,T, 'b', P1, T1-a1, 'y');
legend ('a1','T1','T','Error');
```

Example 2: Consider a surface described by $z = \cos(x)\sin(y)$ defined on a square $-2 \le x \le 2, -2 \le y \le 2$, plot the surface **z** as a function of **x** and **y** and design a neural network, which will fit the data. You should study different alternatives and test the final result by studying the fitting error.

Solution

```
%Generate data
x = -2:0.25:2; y = -2:0.25:2;
z = cos(x)'*sin(y);
%Draw the surface
mesh (x, y, z)
xlabel ('x axis'); ylabel ('y axis'); zlabel ('z axis');
title('surface z = cos(x)sin(y)');
gi=input('Strike any key ...');
%Store data in input matrix P and output vector T
P = [x; y]; T = z;
%Create and initialize the network
net=newff ([-2 2; -2 2], [25 17], {'tansig' 'purelin'},'trainlm');
%Apply Levenberg-Marquardt algorithm
%Define parameters
net.trainParam.show = 50;
net.trainParam.lr = 0.05;
net.trainParam.epochs = 300;
net.trainParam.goal = 1e-3;
%Train network
net1 = train (net, P, T);
gi=input('Strike any key ...');
%Plot how the error develops
%Simulate the response of the neural network and draw the surface
a= sim(net1,P);
mesh (x, y, a)
% Error surface
mesh (x, y, a-z)
xlabel ('x axis'); ylabel ('y axis'); zlabel ('Error'); title ('Error surface')
gi=input('Strike any key to continue......');
% Maximum fitting error
Maxfiterror = max (max (z-a));  %Maxfiterror = 0.1116
```



Example 3

- Here a perceptron is created with a 1-element input ranging from -10 to 10, and one neuron.

```
net = newp ([-10 10],1);
```

- Here the network is given a batch of inputs **P**. The error is calculated by subtracting the output **Y** from target **T.** Then the mean absolute error is calculated.

```
P = [-10 -5 0 5 10];
T = [0 0 1 1 1];
Y = sim (net, P)
E = T-Y;
perf = mae (E) ;
```

Example 4

- Here a two-layer feed-forward network is created with a **1**-element input ranging from **-10** to **10**, **four** hidden **tansig** neurons, and one **purelin** output neuron.

```
net = newff ([-10 10],[4  1],{'tansig','purelin'});
```

- Here the network is given a batch of inputs **P**. The error is calculated by subtracting the output **A** from target **T**. Then the mean squared error is calculated.

```
P = [-10 -5 0 5 10];
T= [0 0 1 1 1];
Y = sim(net,P)
E = T-Y
perf = mse(E)
```

Example 5:

```
load house_dataset;
inputs  = houseInputs;
targets = houseTargets;
net = newff (inputs, targets, 20);
net = train(net, inputs, targets);
outputs = sim(net, inputs)
%outputs1 = net (inputs);
errors = outputs - targets;
perf = perform(net, outputs, targets)
```