**Assignment 2**

Zezhou Li

10405265

In this assignment, I am going to use Matlab to train the Iris.csv data and get the backpropagation neural network and then implement the neural network to the test data and get the classification results, finally compare them with the actual results to see the accuracy.

In Matlab, there is a toolbox which contains neural network tools and function so that we can simply set some desired parameters for the function we use and then implement the training data and result to train the neural network. The parameter of neural network is easy to be set differently so we can try many different parameters such as number of layers or number of iterations.

First, import the data from the given file, since the format of the file is ‘.*csv’*, thus I use an open source function to import them in Matlab without importing by hand, everything I will do will be automatically done by the program, thus it will be useful in the future.

Since the import function is an open source function, I am not going to show it here, I will put it into the attached file.

Here is the code importing the data:

|  |
| --- |
| TrainDataFile = 'Iris.csv';  TestDataFile = 'test.csv';  Data = csvimport(TrainDataFile);  Data = Data(2:end,:);  Data2 = csvimport(TestDataFile);  Data2 = Data2(2:end,:); |

After importing the files, I have to separate the data into two sets which are training data and result. The following codes realize this target:

|  |
| --- |
| TrainData = cell2mat(Data(:,2:end-1));  ClassData = Data(:,end);  TestData = cell2mat(Data2(:,2:end-1));  ResultData = Data2(:,end); |

Now we have separated data sets. However, here the class ClassData and ResultData are still in string type, I should assign them with different numbers representing different classes. Thus, I am going to use two iterations to do it:

|  |
| --- |
| for i = 1:size(ClassData,1)  if strcmp(ClassData{i},'Iris-setosa')  ClassData{i} = 1;  else if strcmp(ClassData{i},'Iris-versicolor')  ClassData{i} = 2;  else  ClassData{i} = 3;  end  end  end    for i = 1:size(ResultData,1)  if strcmp(ResultData{i},'Iris-setosa')  ResultData{i} = 1;  else if strcmp(ResultData{i},'Iris-versicolor')  ResultData{i} = 2;  else  ResultData{i} = 3;  end  end  end    ClassData = cell2mat(ClassData);    ResultData = cell2mat(ResultData); |

Finally, we have all the dataset already transformed.

Next step is to normalize the data, there is a function in Matlab called premnmx. Codes are shown as the following:

|  |
| --- |
| [input,minI,maxI] = premnmx(TrainData'); |

After normalizing the training data, we can then construct the output matrix, here is the output matric:

|  |
| --- |
| s = length(ClassData);  output = zeros(s, 3);    for i = 1:s  output(i, ClassData(i)) = 1;  end |

Finish constructing output matrix, we can then create the backpropagation neural network. Here we need 10 hidden nodes and 3 output nodes and also we will use sigmoid function as the threshold unit. We set learning rate as 0.01, convergence error as 0.01, iteration number as 10. The codes are as following:

|  |
| --- |
| net = newff(minmax(input),[10 3], {'logsig' 'purelin'},'traingdx');    net.trainparam.show = 5;  net.trainparam.epochs = 10;  net.trainparam.goal = 0.01;  net.trainParam.lr = 0.01; |

We have now finished setting the parameters for our neural network, we can then train the neural network:

|  |
| --- |
| net = train(net,input,output'); |

So now we have the neural network. Then test it with our test data. Here I extract the 20% of the data from each class. The remaining 80% is used for training.

|  |
| --- |
| testInput = tramnmx(TestData',minI,maxI);    Y = sim(net,testInput); |

Each column in Y contains three outputs. We now have to find the maximum of each column, that is the final classification result of the test. After that we can compare them with the true result we know from the given data:

|  |
| --- |
| [s1,s2] = size(Y);  hitNum = 0;  for i = 1:s2  [m,Index] = max(Y(:,i));  if(Index == ResultData(i))  hitNum = hitNum + 1;  end  end    sprintf('Accuracy is %3.3f%%',100 \* hitNum / s2 ) |

Now we get the accuracy is: 30%.

This accuracy is very low than what we need, we need to improve it. As I have shown in the above, we set the number of iteration as 10. The iteration is not enough, so we enlarge the iteration number now to 50.

|  |
| --- |
| net.trainparam.show = 10;  net.trainparam.epochs = 50; |

Then again we run the test, we get the accuracy as: 63.333%.

It is higher than before, then we try to set it to 100, the accuracy now becomes: 100.000%

Stop here, we get 100% accuracy.

Then try different hidden neurons, from 10 to 100, we can see that if the number of hidden neurons increases, the running time increases.