Lab work Neural Network

Bishal Chaulagain **Sushant Gautam**

Implementation of Back propagation Algorithm
WAP to implement back propagation algorithm and analyze the performance of your NN model using any one type of performance analysis method.

INTRODUCTION:
This program was written to use of back propagation neural networks (BPNN) for the identification of iris plants on the basis of the following measurements: sepal length, sepal width, petal length, and petal width.

Implementation Steps:

Data set construction:
The first of the data sets is the training set, which is used for the actual training of the network, and for the determination of the networks recall ability. The second data set is the testing data set, which is not used in the training process, and is used to test the networks level of generalization. This is done through the analysis of the accuracy achieved through testing against this set.

Normalization:
Normalization of input data is used for ranging of values within an acceptable scope, and region. There are many mechanisms towards this end Some of them are column, row, sigmoid, and column constrained sigmoid normalization.

Back propagation neural network (BPNN)
BPNN use a supervised learning mechanism, and are constructed from simple computational units referred to as neurons. Neurons are connected by weighted links that allow for communication of values. When a neuron’s signal is transmitted, it is transmitted along all of the links that diverge from it.

The algorithm towards the training of the network is as follows:
Step 1: Build a network with chosen number of input hidden and output units.
Step 2: Initialize all the weights to low random values.
Step 3: Randomly choose a single training pair
Step 4: Copy the input pattern to the input layer.
Step 5: Cycle the network so that the activation from the inputs generates the activation in the hidden and output layer.
Step 6: Calculate the error derivative between the output activation and the final output
Step 7: Apply the method of back propagation to the summed product of the weights and errors in the output layer in order to calculate the error in the hidden unit
Step 8: Update the weights attached at the each unit according to the error in that unit the output from the unit below it and the learning parameter until the error is sufficiently low.

These 8 steps of the Back propagation algorithm can be further simplified into following 4 Steps:

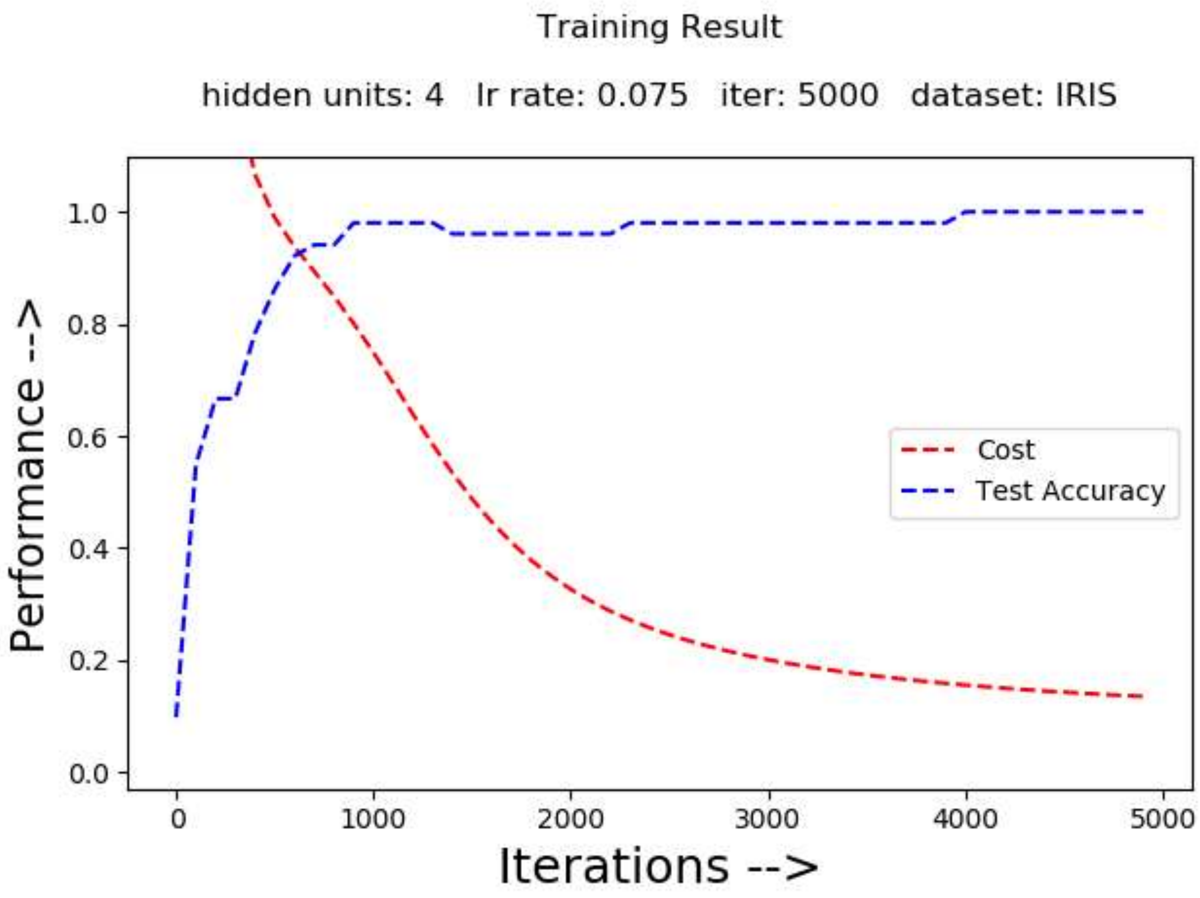
Step 1: Initialization
The weights and threshold values for the network are assigned values that are uniformly distributed over a small range.
Step 2: Activation
It is at this point that input values from a training set are presented to the networks input layer neurons, and the expected output values that are declared within the set qualified. The networks hidden layer neurons then calculate their outputs.
Step 3: Update weights
This is the step in which the weights of the BPNN are updated through the process of propagating backwards the errors related to the output neuron results.
Step 4: Iteration
Increment the value of P by 1, and return to the second step of the process. This iterative process is conditional upon a terminating condition, if the terminating condition is realized, the training is complete, and the algorithm terminates.

Regression and Classification on IRIS Dataset

Experiment 1: Logistic Regression (classification)

```
number_of_hidden_units = 4
learning_rate = 0.075
num_iterations = 5000
regression = False
Cost Function Used:
```

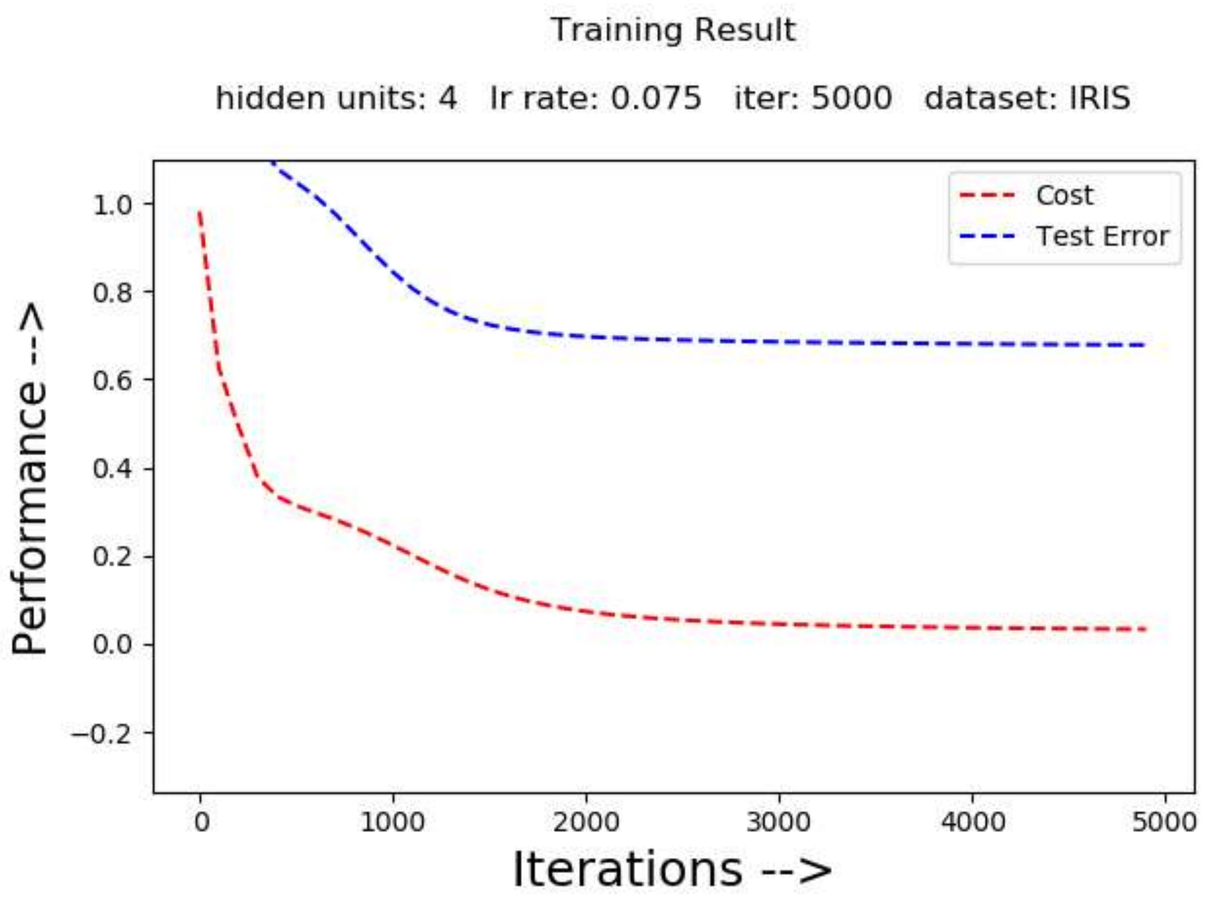
$L = Y * \log(Y\_pred) + (1 - Y) * \log(1 - Y\_pred)$



Experiment 2: Regression

```
number_of_hidden_units = 4
learning_rate = 0.075
num_iterations = 5000
```

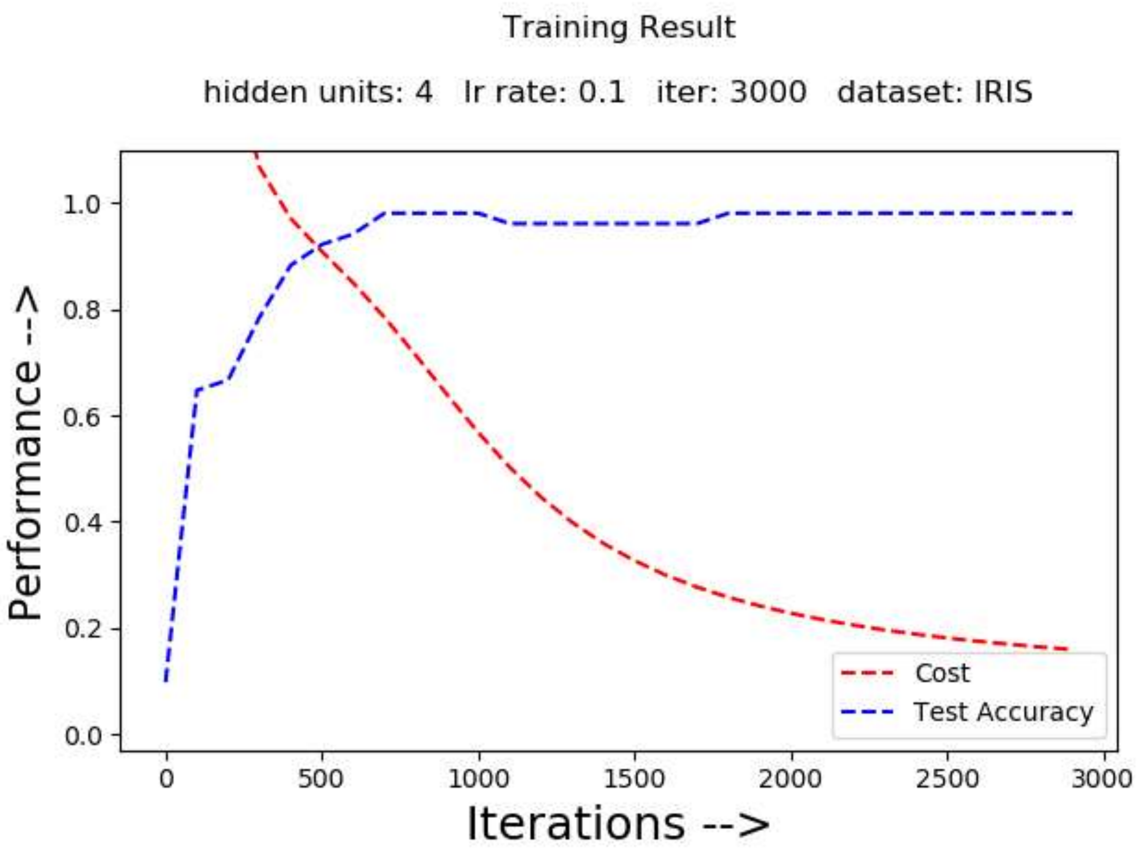
Cost Function: RMSE

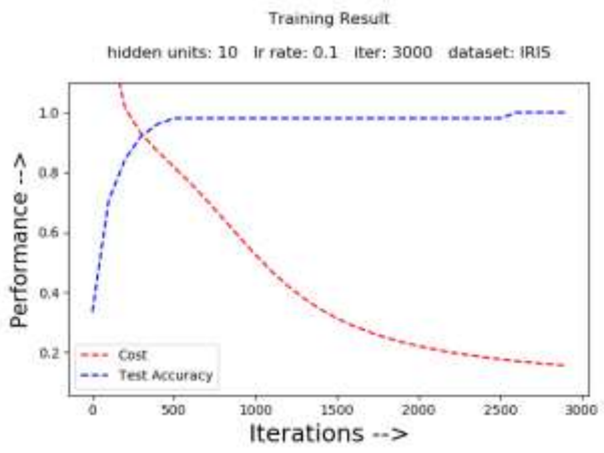


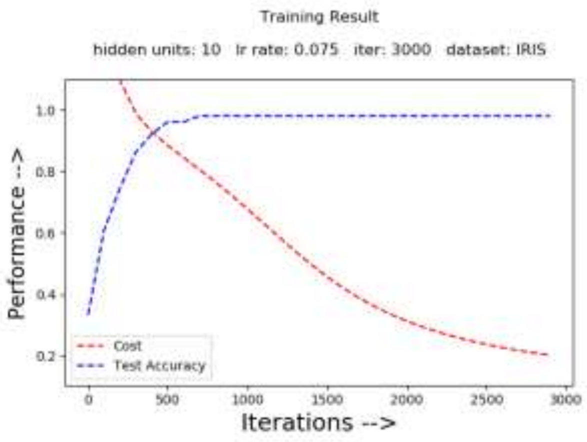
Experiment 2: Logistic Regression varying params

```
number_of_hidden_units = 4
learning_rate = 0.1  #chaning learning rate
num_iterations = 3000
regression = False
Cost Function Used:
```

$$L = Y * \log(Y\_pred) + (1 - Y) * \log(1 - Y\_pred)$$







Conclusion:

We successfully implemented backpropagation algorithm on Python on the given data set. We implemented logistic regression and regression algorithms and tested the output varying various parameters.

Increasing hidden units to 10 did have some effect on learning of algorithm and same did the learning rate. But having lots of hidden units made the model complex and the learning was not faster as expected. We concluded that optimal hidden units should be chosen to trade between complexity and training.