MACHINE LEARNING

Programming Assignment 2,
Submission: December 15th 2014

Classification using
Logistic Regression
and Neural Networks

By

Bala Seshadri Sura

balasesh@buffalo.edu

50097470

This project is divided into 2 parts and has been explained as below. The necessary files have been saved and submitted in the zip file and the necessary analysis has been done in this report.

Logistic Regression:

Logistic regression, or logit regression, or logit mode, is a type of probabilistic statistical classification model. It is also used to predict a binary response from a binary predictor, used for predicting the outcome of a categorical dependent variable (i.e., a class label) based on one or more predictor variables (features). That is, it is used in estimating the parameters of a qualitative response model. The probabilities describing the possible outcomes of a single trial are modeled, as a function of the explanatory (predictor) variables, using a logistic function. Frequently (and subsequently in this article) "logistic regression" is used to refer specifically to the problem in which the dependent variable is binary—that is, the number of available categories is two—while problems with more than two categories are referred to as multinomial logistic regression or, if the multiple categories are ordered, as ordered logistic regression.

Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables, which are usually (but not necessarily) continuous, by using probability scores as the predicted values of the dependent variable. As such it treats the same set of problems as does probate regression using similar techniques; the former assumed logistic function, the latter standard normal distribution function.

Logistic regression can be seen as a special case of generalized linear model and thus analogous to linear regression. The model of logistic regression, however, is based on quite different assumptions (regarding the relationship between dependent and independent variables) from those of linear regression. In particular the key differences of these two models can be seen in the following two features of the former. Firstly, the conditional mean P (y|x) subjects to Bernoulli distribution rather than Gauss distribution because logistic regression is a classifier. Secondly, the linear combination of the inputs $w^T x \in R$ is restricted to [0, 1] through the logistic distribution function because logistic regression predicts the probability of the instance being positive.

In this coding assignment we are performing a multi class classification problem where 10 classes of data has been provided and classified. Initially we are extracting the data from the given data files and concatenating them into one matrix of size 19978*512 and then we are adding a column of bias in the beginning of the matrix to make I a size of 19978*513. Then we are creating a target matrix of size 19978*10 and each data section and the corresponding column is set to 1.

In the **training phase**, we are creating a **weight matrix** of size 513*10 using the random function and by trial and error I have set my eta value to 0.0001. I am running my iterations to 500 so that the training can run for 500 iterations. I am using the probabilistic determinant model where the data labels are predicted on the basis of the probability. This is calculated by taking the exponential of the value of (X*W) where X is the data and W is the weight, then we calculate the sum of all the class probabilities and divide the individual with the sum of probable and calculate the value of Y. Then we calculate the Error Gradient value using the eta value and the weights to update the weights and use them in the next iteration. Error is calculated the cost function using the negative likelihood and the data is divided to by the number of features and the sign is changed from negative to positive. The errors are stored in an array and used for plotting for analysis purpose. In this function we are taking the **data(X), target matrix (T)** as the **input** and sending the **weight (W), error (E, Error)** as the **output** to the main project code so as to send the respective data to the test phase.

In the **test phase**, we are inputting the weights (W) and the test data (X) we are doing the same functions as in the train phase but we are only running it once with the previously calculated weights. Then we are taking the maximum of the indexes of the features and saving them into a label matrix (Index). This is sent to the main program and the data is copied and saved to the **classes_lr.txt** file.

The error in the training phase was observed to be **0.066386** which is **93.4**% accuracy and a **6.6**% error. The labels file has been created and saved into the zip folder.

Neural Network:

This method follows **feed forward**, where the output of one layer is feed as the input to the next layer. In between there are hidden layers where the classification takes place. Before feeding the system with training data, the system abstracted with initial weights. Classification of unknowns are done based on the features of the previous set of trained data set. During the initial phase of training, the system will be feed with inputs with correct output. The system records in the training data one at a time, then compares the resulting outputs against the desired outputs. Errors are is feed as the deviation and then use that to recalculate values (weights) this is known as back propagation, causing the system to adjust the weights for application to the next record to be processed. During the training of a network the previous step is processed many times as the output weights are refined again and again. This is done so that the error factors are used to adjust the weights in the hidden layers so that the next time the output values will be closer to the "correct" values.

Backpropagation: It is the backward propagation of errors in neural network. To train a neural network to perform some task, we must adjust the weights of each unit in such a way that the error between the desired output and the actual output is reduced. This process requires that the neural network compute the error derivative of the weights (EW). The backpropagation is done in two steps.

Propagation: Training data is provided to neural network, generated an output for the sample is compared with desired output of the sample. Data from input to output propagates in forward direction. After the comparison of the outputs we get a difference that must be adjusted. This is feedback to the hidden layer to compute output with less deviation from the desired output. Here the data propagates backward.

Error calculation: Error backpropagation step requires computing the derivative of error function with respect to the weigh. The below diagram show how a neural network architecture looks. It contains 3 main distinct layer for each activity.

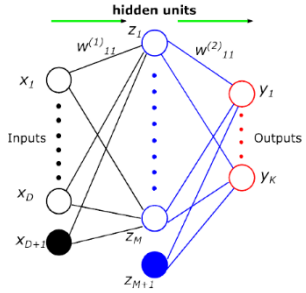


Figure 1: Neural network

At input layer:
Train data sets are provided using which the system trains itself, the input data is obtained and stored in the form of records.
Other inputs for validation are also feed to this layer. The data are stored and passed to next layer called the hidden layer.

At hidden layer:
Each layer can apply any function you want to the previous layer to produce an output (usually a linear transformation followed by a squashing nonlinearity).
The hidden layer's job is to transform the inputs into something that the output layer can use.
A random weight is assigned to each feature, the functions that happens in this layer is unknown and it is hidden. The layer does the computation of weights for each features and also does the

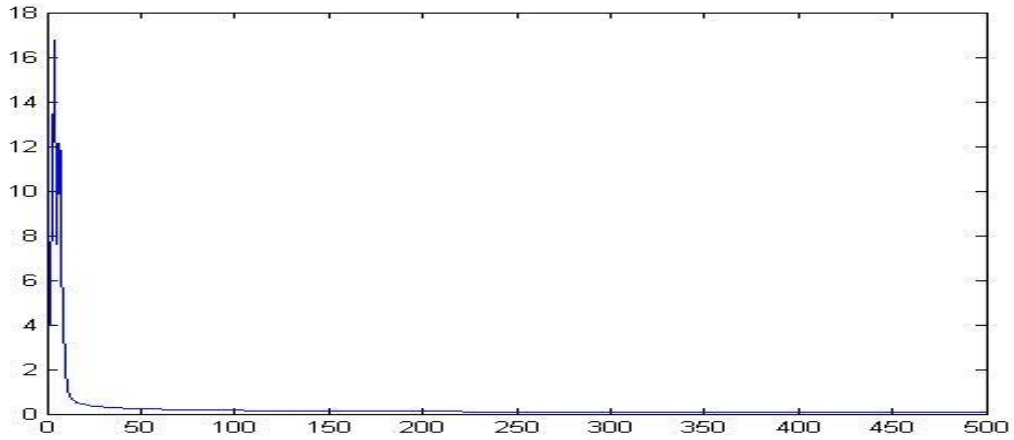
In this **training phase** we are taking the same data set created for the logistic regression part and passing the data to the train_nn function where the **input** is the same **dataset (X) and the target (T).** Now we create 2 weight matrix w1 (513*20) and w2 (10, 20) within the range of -0.5 to 0.5 so that the weight don't vary too much. Then we calculate the ***aj, zj***, and ***yk*** as in the neural network derivation and use the ***tanh*** function as the activation function. Now similarly as we calculated the probability in the logistic regression section. We calculate the error gradient using the gradient descent method and I have set the eta value to 0.0001 using trial and error. We then update the values of w1 and w2. We then calculate the updated w1 and w2. We calculate the error in the same way as the logistic regression section. The process runs for 500 iterations and the weights w1 and w2 is sent to test phase. The **output** elements for this function is **w1, w2, and the error data**.

In the test phase, we are taking the weights w1 and w2 and we are calculating the ***aj,zj*** and ***yk*** values, same as in the training phase but here we are just doing it in 1 iteration. Then we are taking the maximum of the indexes of the features and saving them into a label matrix (Index). This is sent to the main program and the data is copied and saved to the **classes_nn.txt** file.

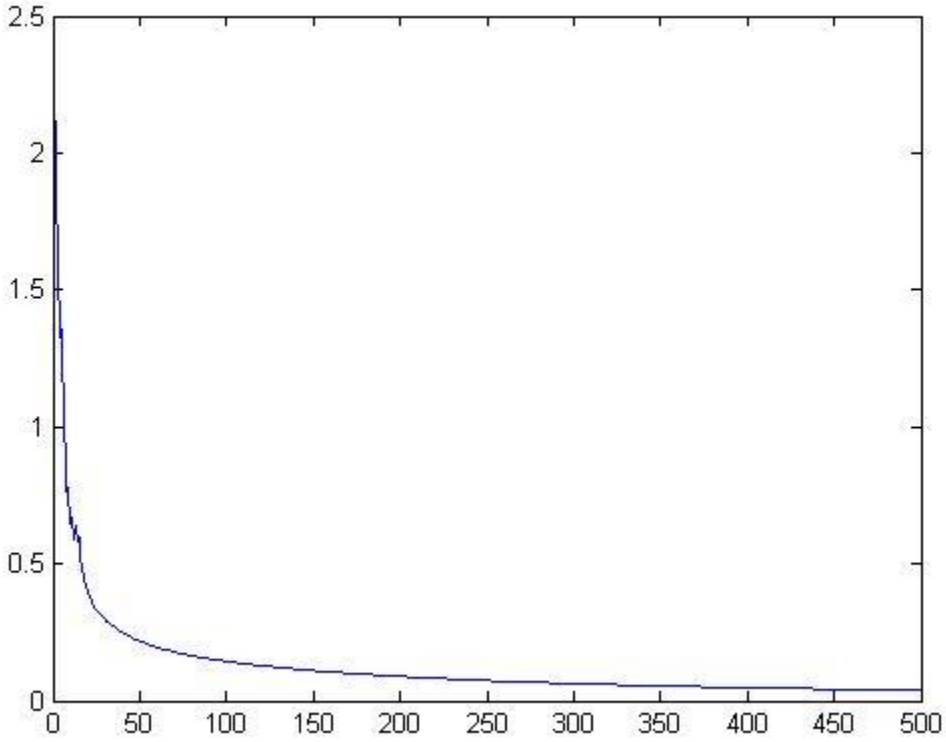
At the end the error I have recorded is: **0.035610.** So the error rate is **3.5%** which means the accuracy is **96.5%.**

This states that the neural network is better than logistic regression classification as the error is literally half of that of logistic regression. Please find the graphs for the below.

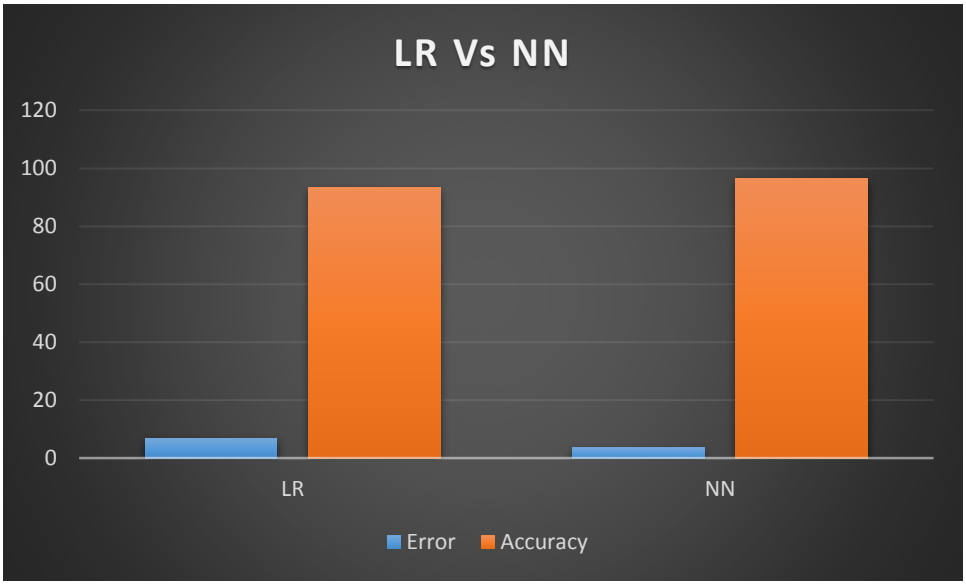
Error is Logistic Regression:



Error is Neural Network:

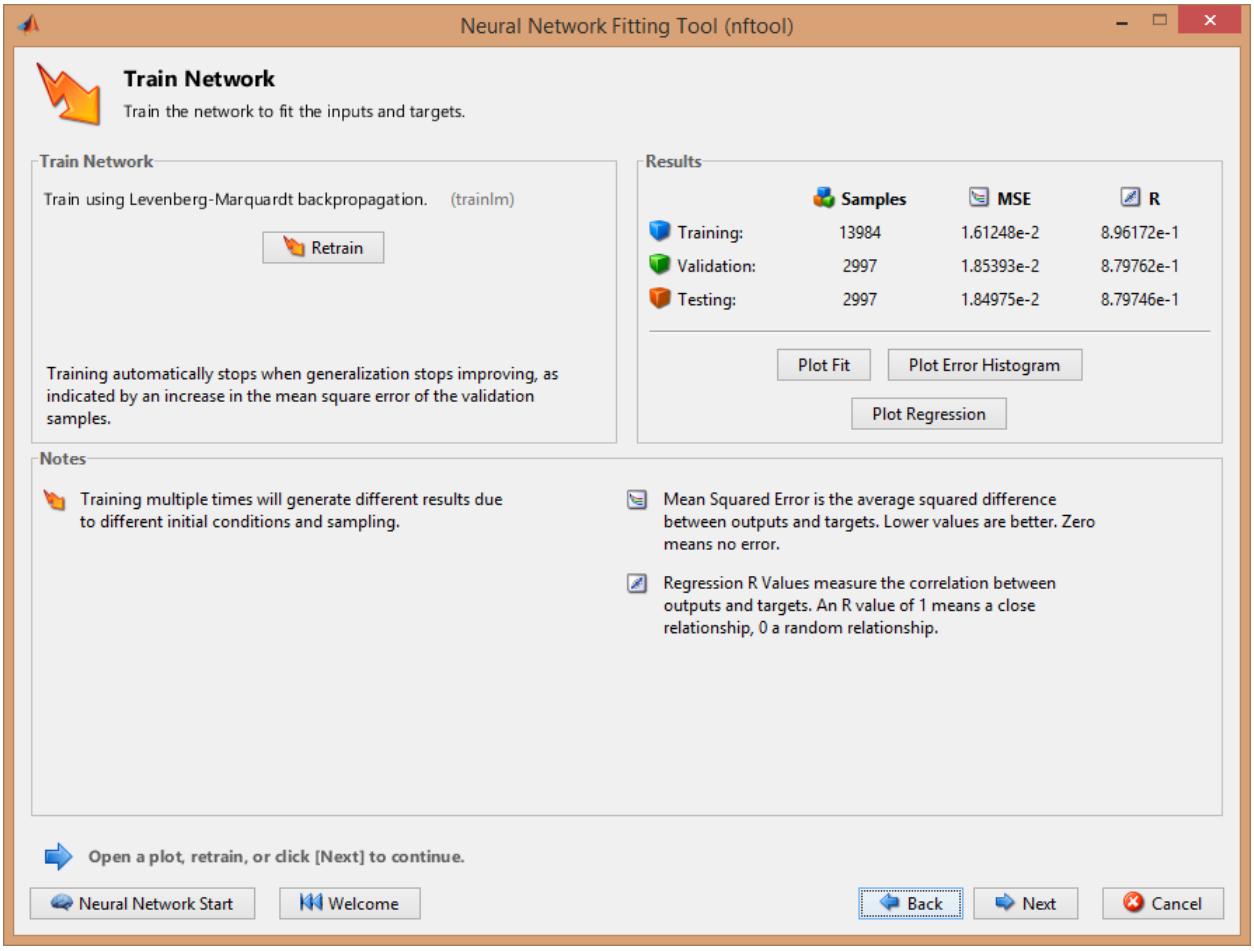


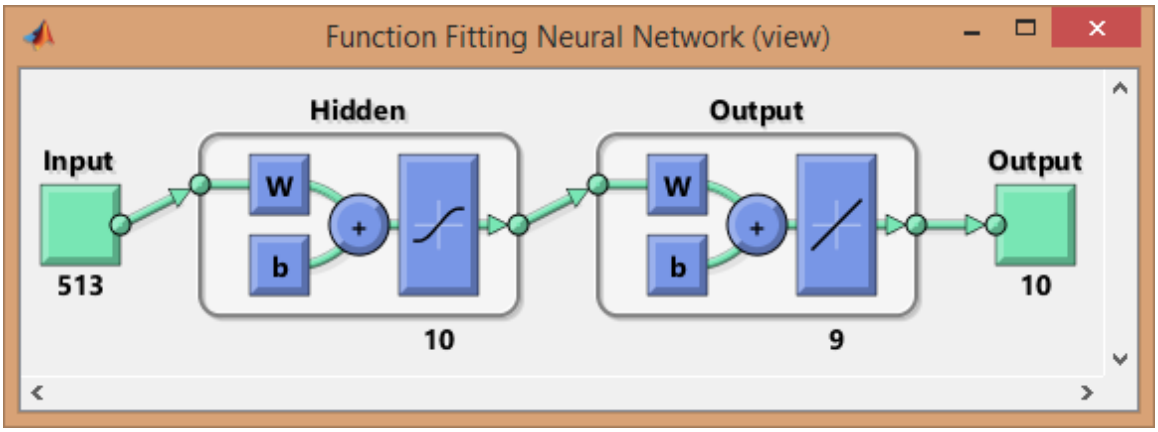
Logistic Regression Vs Neural Network.

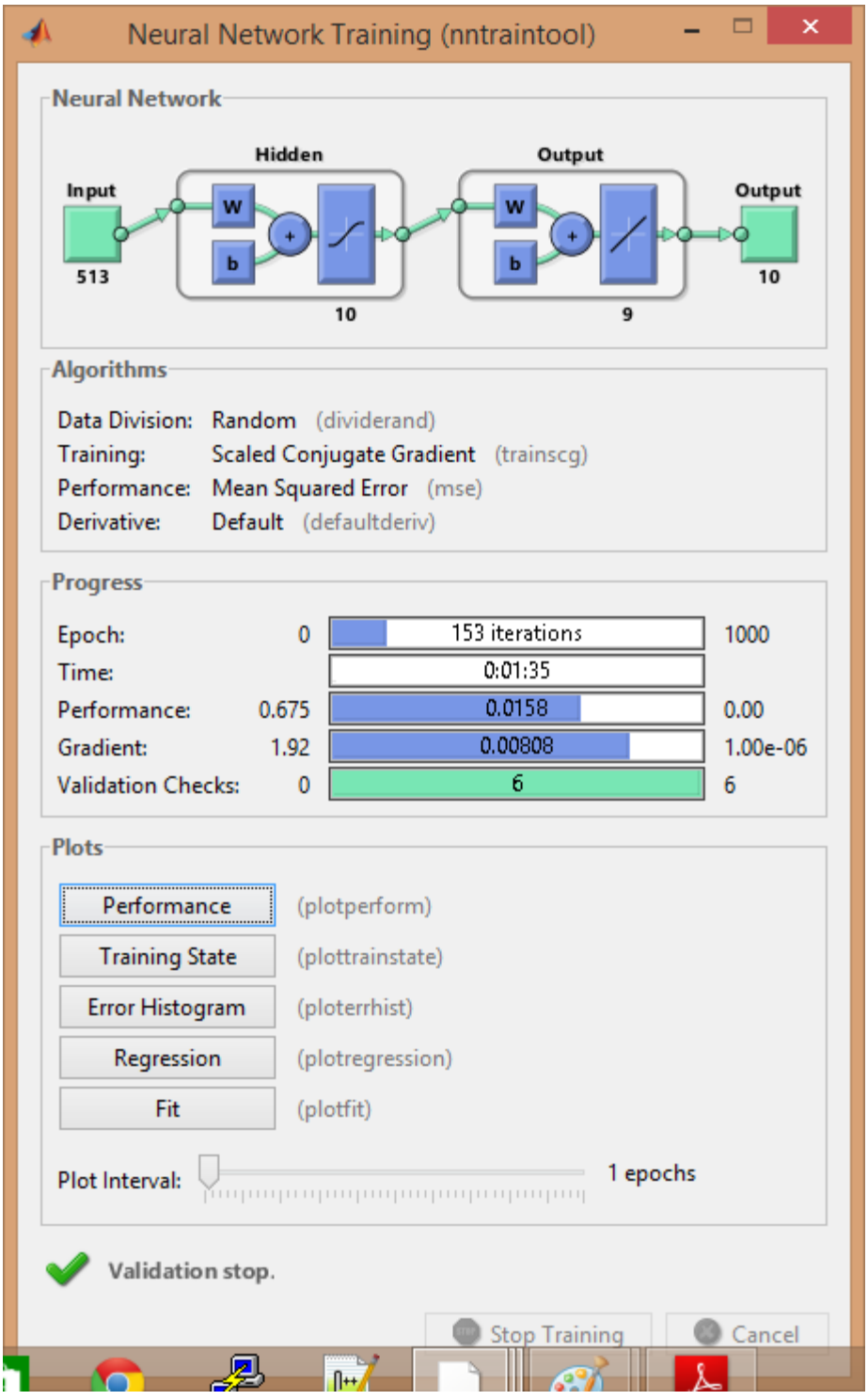


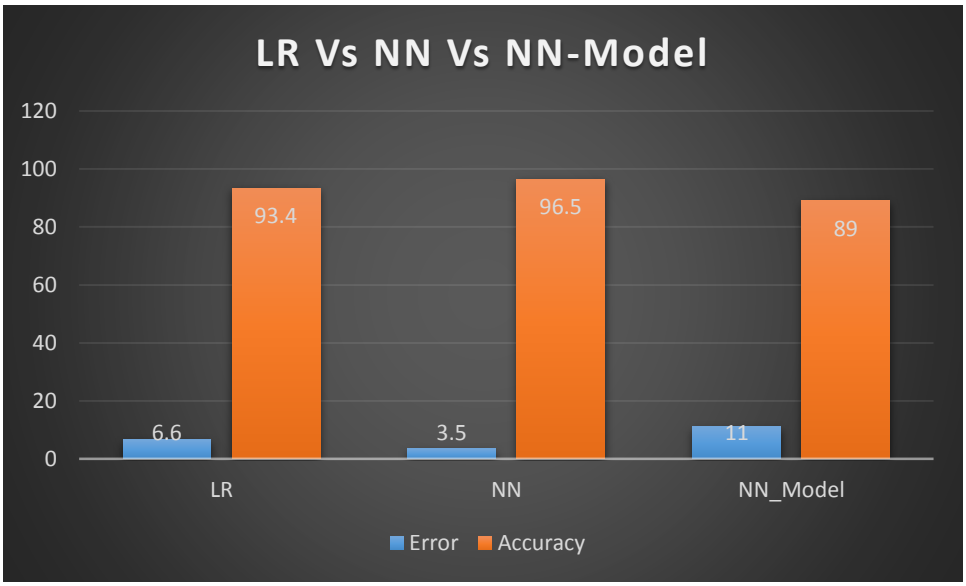
Using NN Model to compare to the current data.

I have used the nn_model to check the performance of the neural network. Please find the attached screen shots below. The Error rate for the nn_model is about 11 percent in the test phase and the accuracy is 89%. To run the code I have used the **nftool** command and generated the code. The **nn_model.m** file has been attached in the zip file.




Function Fitting Neural Network (view)
Input
513
Hidden
W
b
10
Output
W
b
9
Output
10


Neural Network Training (nntraintool)
Neural Network
Input
513
Hidden
W
b
10
Output
W
b
9
Output
10
Algorithms
Data Division: Random (dividerand)
Training: Scaled Conjugate Gradient (trainscg)
Performance: Mean Squared Error (mse)
Derivative: Default (defaultderiv)
Progress
Epoch: 0 153 iterations 1000
Time: 0:01:35
Performance: 0.675 0.0158 0.00
Gradient: 1.92 0.00808 1.00e-06
Validation Checks: 0 6 6
Plots
Performance (plotperform)
Training State (plottrainstate)
Error Histogram (ploterrhist)
Regression (plotregression)
Fit (plotfit)
Plot Interval: 1 epochs
Validation stop.
Stop Training
Cancel



References:

http://en.wikipedia.org/wiki/Logistic_regression
http://en.wikipedia.org/wiki/Artificial_neural_network
http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html
http://www.mathworks.com/help/matlab/line-plots.html
http://www.mathworks.com/help/matlab/ref/fprintf.html
http://www.mathworks.com/matlabcentral/answers/1085-inserting-a-column-in-a-matrix-without-deleting-any-column
http://www.mathworks.com/help/matlab/ref/exp.html