**Name: Rajdeep Pinge**

**ID: 201401103**

**Work:** RBF with Pseudo Inverse, Random Initialization of Centres

**Extra:** 1. RBF with Pseudo Inverse, Initialization of Centres using k-means clustering

     2. MLP classification with Fourth Power Error

     3. MLP classification with LS and momentum implementation

**Activation Function:**1: RBF kernel in the first layer

2: No activation in the last layer

**Parameters to tune:**

    Number of Hidden Neurons (K).

**Algorithm:**

Initially, we have to fix the number of neurons in the hidden layer. This has been done by iterating over the number of neurons and checking the accuracy of the results. The detailed analysis has been done at the end of the report.

Step:1 Choose Random Centres: Take some random samples from the input samples and assign them directly as the centres.

Step:2 Find the maximum distance among centres and initialize spread of the RBF (sigma) as **max\_dist / square\_root(K)** where K is the number of neurons.

Step:3 Build G matrix which stores the function values of all inputs for all the centres.

With the RBF being **phi(x) = exp ( -||x – mu||^2 / 2(sigma)^2 )**

Step:4 Find weights by using the pseudo inverse of G matrix and actual values of samples.

**Weight = pinv(G) \* actual classes**

For testing purpose,

Step:5 build the G matrix for the testing data

Step:6 Use the weights determined in step 4 and the G matrix to find the answer.

**Yout = G \* Weight**

Step:7 Calculate all the accuracies.

**Observations:**

For Set 5

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Train | Train | Train | Test | Test | Test |
| Dataset | Number of RBF center neurons | Overall Accuracy (ηo) | Average Accuracy (ηa) | Geometric Mean Accuracy (ηg) | Overall Accuracy (ηo) | Average Accuracy (ηa) | Geometric Mean Accuracy (ηg) |
| AE | 22 | 100 | 100 | 100 | 97.81022 | 98.3046 | 98.28997 |
| Iris | 12 | 97.77778 | 97.77778 | 97.72648 | 98.09524 | 98.09524 | 98.08605 |
| ION | 44 | 91.66667 | 91.66667 | 91.49816 | 83.26693 | 80.83161 | 80.37181 |
| Liver | 20 | 73.49398 | 73.49398 | 73.45446 | 66.89655 | 66.39137 | 66.29968 |
| PIMA | 12 | 73.87097 | 73.87097 | 73.27624 | 74.76494 | 74.97137 | 74.76494 |
| VC | 153 | 83.79254 | 83.7702 | 82.93661 | 73.45972 | 73.99127 | 70.36973 |
| Wine | 20 | 100 | 100 | 100 | 95.76271 | 95.99404 | 95.98394 |

(Detailed for all sets in a separate excel sheet)

The number of RBF center neurons is determined by plotting the graph of obtained accuracy vs the number of neurons done in the following graphs and analysis section.

**Analysis:**

I have performed cross validation by splitting the data 90:10. That is, 90% of the samples have been taken for training while 10% samples are taken for validation.

Number of hidden neurons: The thumb rule was to take the minimum neurons to be at least (number of input features + number of classes) Too less neurons increased the error while too much of them let to overtraining.

Since the centres are determined randomly, and the spread depends on it, the main factors of the algorithm are chosen randomly. Hence the answers vary by 5-10%. But largely the trend remains same.

For certain smaller (ae, Iris, Wine) data sets, the algorithm works very well. While for data sets having large number of features (ION, VC, Liver), it performs poorly. Also, if the data samples are close by and the features have high complexity, they only cover a certain range of sample space. Hence the accuracy decreases. This might be the case in the Liver and PIMA data sets.

**Limitations and Possible Solutions**:

There are certain limitations to this algorithm. One is the randomness of the starting data as mentioned earlier. One possible improvement for this may be by initializing the centres using k-means clustering algorithm. Though, k-means gives only a local optimum and itself is vulnerable because of it dependence on initial values. Therefore it may not improve the algorithm significantly.

The centres and the spread of the RBF remain constant throughout. There is no way of updating them based on the predictions results. This can be improved. Other implementations like the gradient descent, update the centre and spread of the RBF after each iteration. This may give better accuracy.

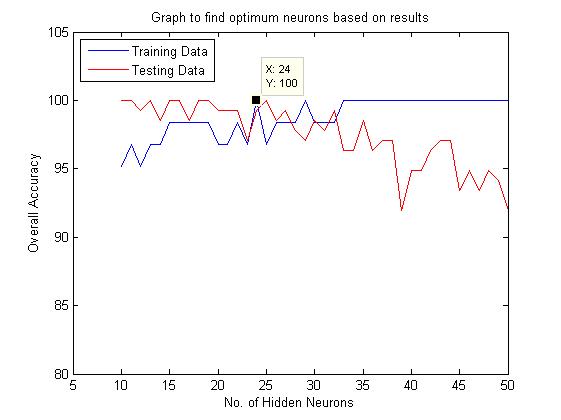
Coupling RBF gradient descent with k-means clustering for initializing centres and spread may give much improved results.

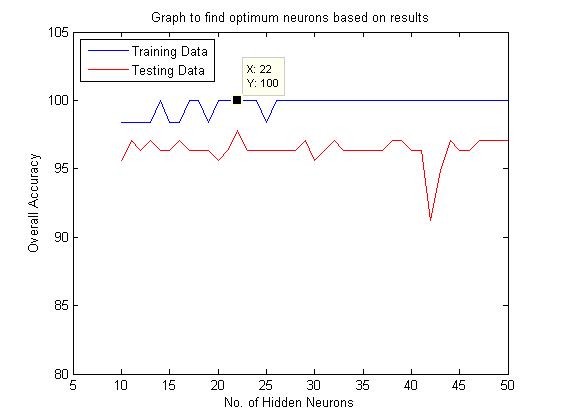
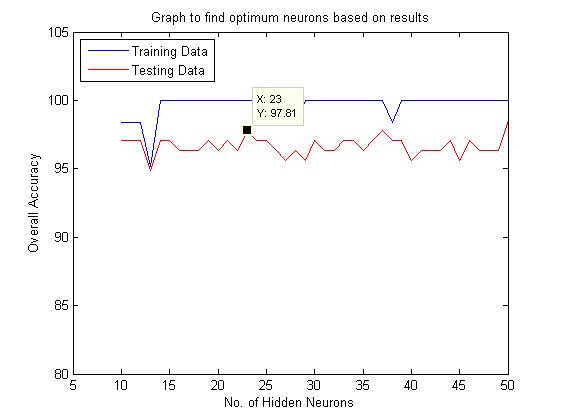
**Graphs and Analysis:**

**Data Set: ae**

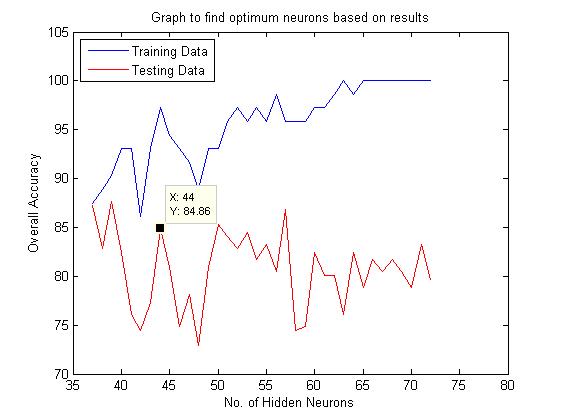
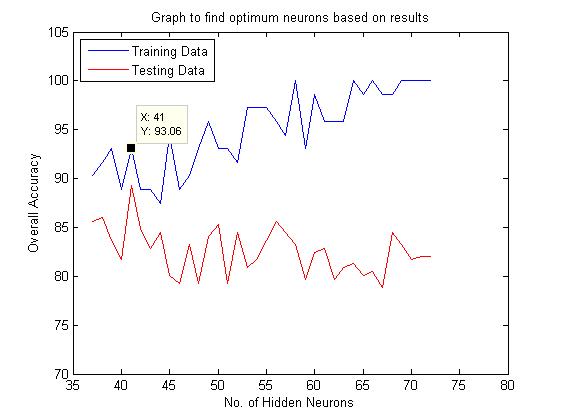
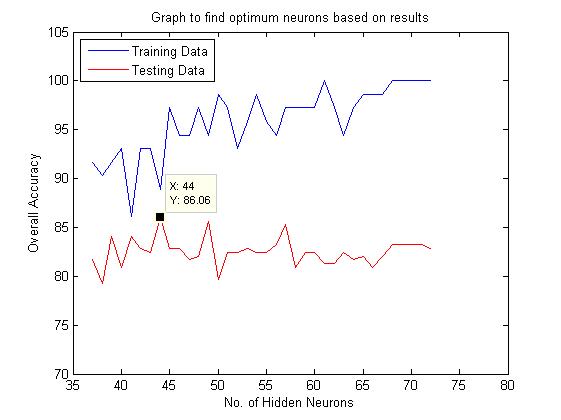
The three graphs below, are the accuracy vs number of neurons graphs for data samples from different sets.

    As can be observed, the accuracy of testing data goes down or remains constant. In the first graph it goes down. Therefore we must take the K(number of hidden neurons as small as possible such that we get maximum test data accuracy and then maximum training data accuracy. This can only be achieved around K = 22-24, observing the results of all the three graphs.



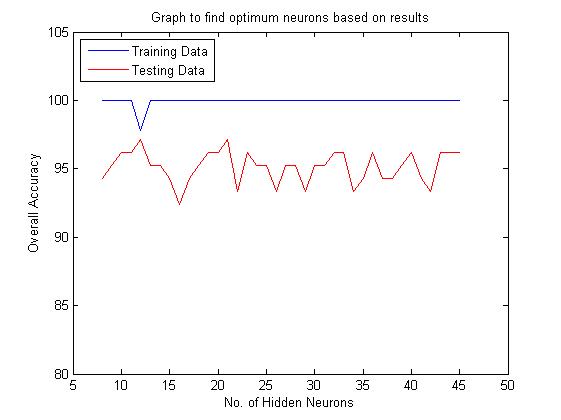
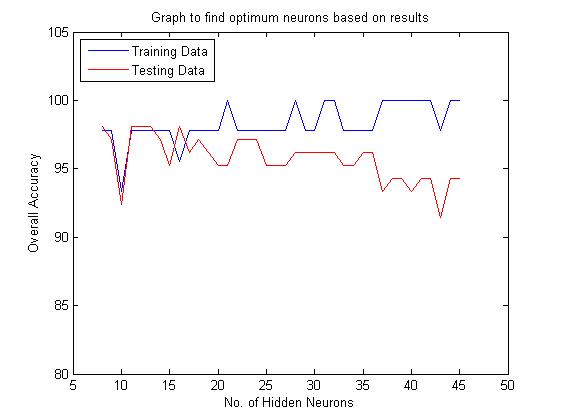


**Data Set: ION**

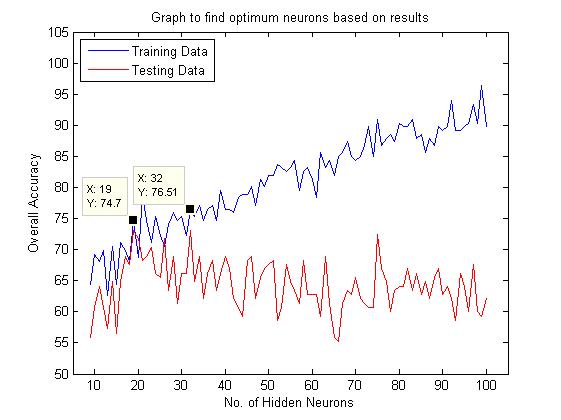
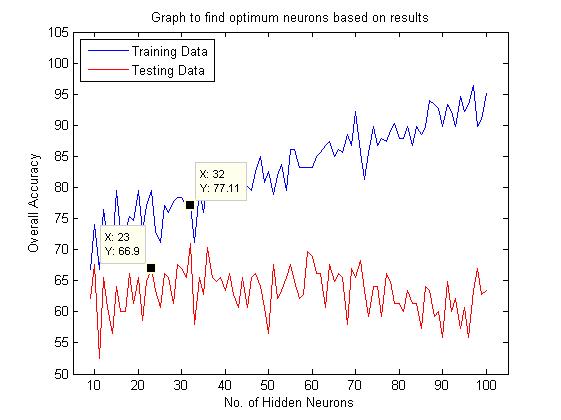
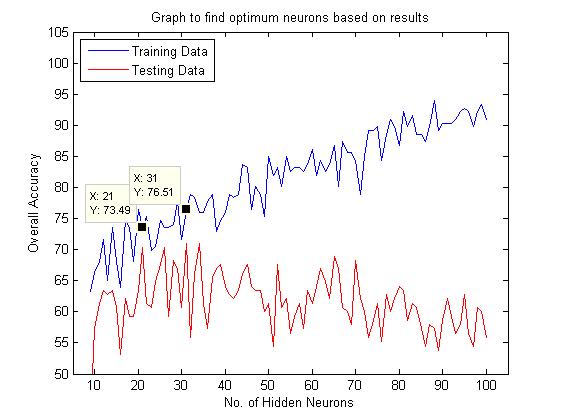
For this data set, the accuracy of the testing data goes down as the number of neurons increase, this must be because of overtraining problem. Therefore, to avoid this, we must take the K in the range 41-46.

**Data Set: Iris**

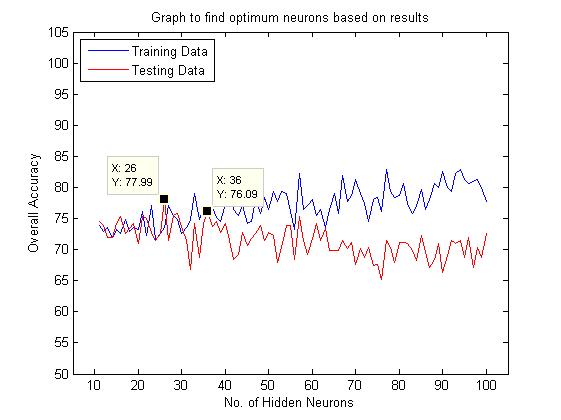
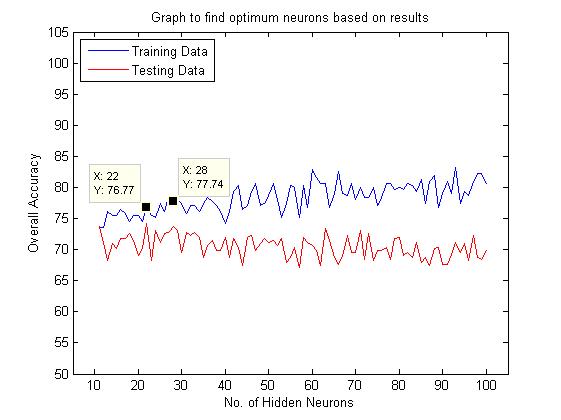
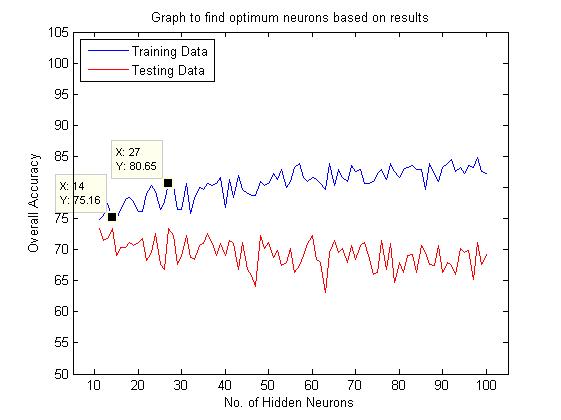
Here also the accuracy of testing data goes down eventually. Therefore we must take 10-12 neurons.



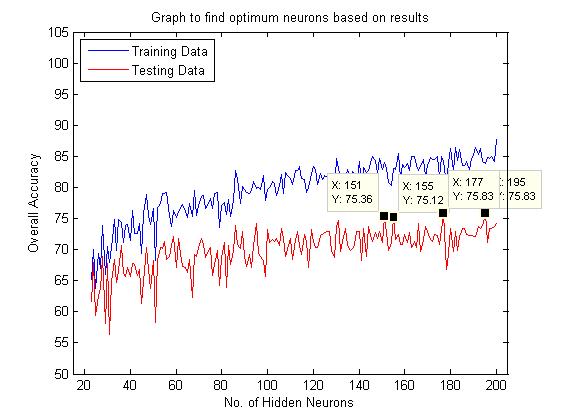
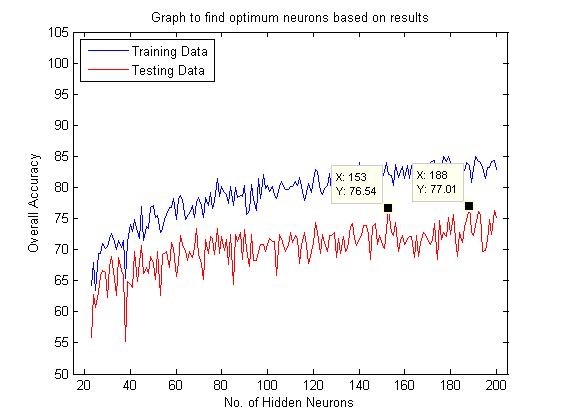
**Data Set: Liver**

In this data set also, the general trend can be observed that as the accuracy of the training data increases, the accuracy of the testing data goes down. The optimum solution can be obtained if neurons are in the range of 20-30.

**Data Set: PIMA**

Optimum solution in the range 20-30 neurons.

**Data Set: VC**

These graphs are very different from the earlier ones. Here the accuracy of both the data samples is increasing. The graphs saturate towards the end, indicating the optimal solution. Therefore, neurons in the range 150-200 will give best solution.

**Data Set: Wine**

This set is straightforward with the smallest neurons being in the rage of 20-22.