REPORT

AASHKA KAPADIA

Function Approximation -

Methods used -

1. MLP approximation using different hidden layer activation functions -

MLP is used to find appropriate weights of input layer and output layer using gradient descent by minimising *least square error*.

Activation Functions : 1. Sigmoid and bi-sigmoidal activation function in hidden layer.

2. No activation in output layer.

Loss function: Least square error function.

Parameters to tune : 1. Number of hidden neurons.

2. Learning rate

3. Epox

Alogorithm :

S1. Normalising the data if needed and initializing the weights of both input and output layers.

S2. Going through all samples one by one and finding hidden layer output using sigmoidal or bi-sigmoidal activation function.

S3. Finding predicted output for samples and then finding error in prediction. This error is used for updating weights of input and output neurons. After going through all samples once weights are updated.

S4. Repeating step 2 and step 3 until given number of epoch.

S5. Validating and testing the network.

2. RBF with pseudo inverse using centres from k-means clustering -

Radial basis function is similar to MLP, only the difference is their is a hidden layer with gaussian activation function and no input weights.

Activation Function : 1. Gaussian function in hidden layer.

2. No activation in output layer.

Parameters to tune : Number of clusters.

Alogorithm :

S1. Normalising data if needed.

S2. Initialising centres through K-means clustering. Initialising sigma as maximum distance between any two centres divided by square root of total number of centres.

S3. Making phi matrix using gaussian activation function giving the distance between centres and data points as parameter.

S4. Finding weight matrix of output layer using pseudo inverse of phi and actual output. Using this weights finding the predicted output.

S5. Validating and testing the network.

3. RBF - gradient descent by minimising least square error -

Activation Function : 1. Gaussian function in hidden layer.

2. No activation in output layer.

Parameters to tune : 1. Number of clusters.

2. Learning rate of centres, weights and spread.

3. Epochs.

Algorithm :

S1. Normalising data if needed.

S2. Randomly initialising cluster centres and initialising sigma as maximum distance between any two centres divided by square root of total number of centres.

S3. Making phi matrix using gaussian activation function giving the distance between centres and data points as parameter.

S4. Finding the predicted output using phi matrix and weight matrix. Finding the error and using this errror to updated sigma, centre and weight matrix.

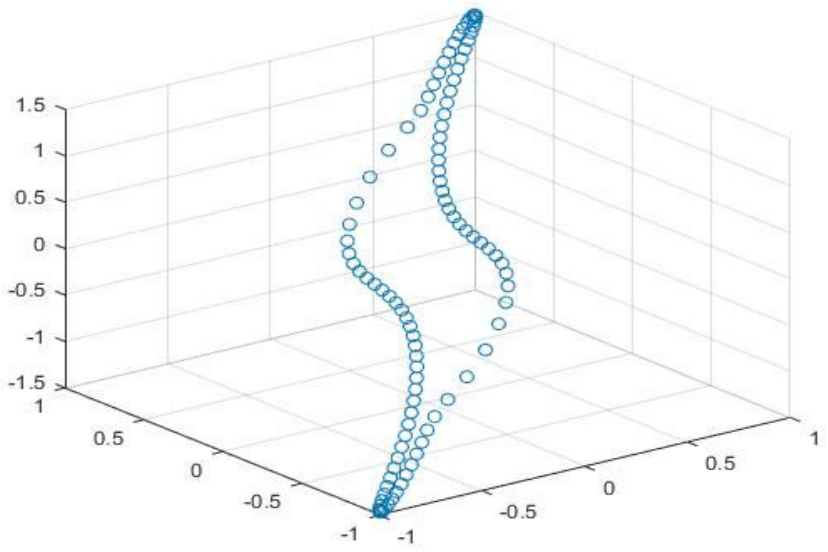
S5. Repeating step 3 and step 4 until given number of epoch.

S6. Validating and testing the network.

A. 1 System Identification -

1.1 Knowing data -

System identification is a approximation problem in which 50,000 samples are provided to train the network. Data provided is a normalised data with two input features ranging from [-1, 1] and output ranging from [-1.466, 1.466]. So, here normalisation of data is not needed. Below is the scatter plot of the SI data with features on x and y axis and output on z-axis.



For cross validation, 60% training data as training set and remaining 40% for testing.

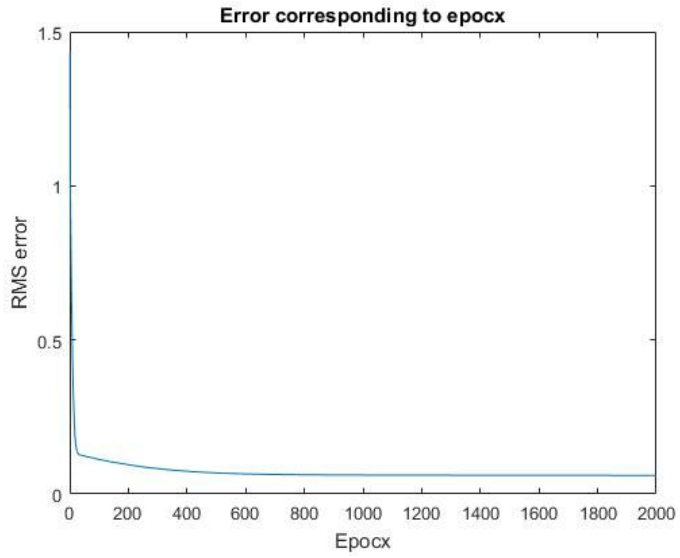
1.2 Results :

<table>
<tr><th>Methods</th><th># Hidden neurons</th><th>Learning rate</th><th>Epox</th><th>Training (RMSE)</th><th>Testing (RMSE)</th></tr>
<tr><td>MLP - Sigmoidal</td><td>11</td><td>1.e-5</td><td>2000</td><td>0.060263</td><td>0.060669</td></tr>
<tr><td>MLP - Bi-sigmoidal</td><td>11</td><td>1e-5</td><td>1100</td><td>0.056231</td><td>0.056484</td></tr>
<tr><td>RBF - pseudo inverse</td><td>15</td><td>-</td><td>-</td><td>0.0043344</td><td>0.0043184</td></tr>
<tr><td>RBF - gradient descent</td><td>15</td><td><table><tr><th>For weights</th><th>For centres</th><th>For spread</th></tr><tr><td>1.e-6</td><td>1.e-8</td><td>1.e-8</td></tr></table></td><td>300
2000</td><td>0.16182
0.12885</td><td>0.93338
0.93338</td></tr>
</table>

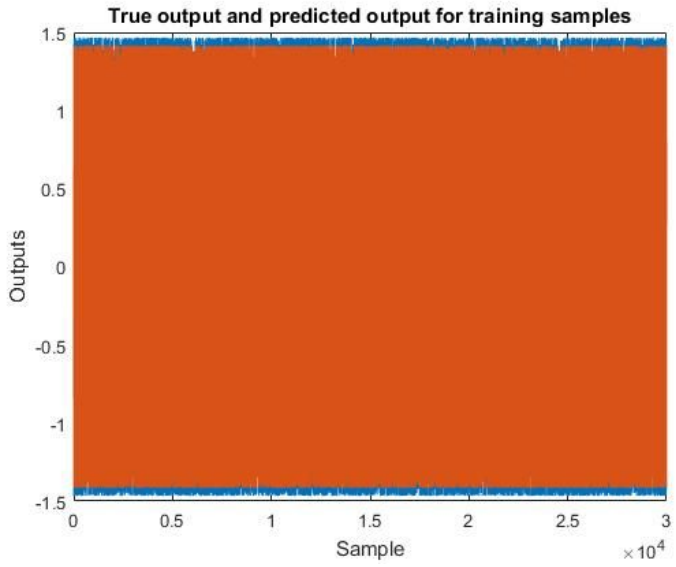
1.3 Graphs and observations -

1. MLP using sigmoidal activation function -

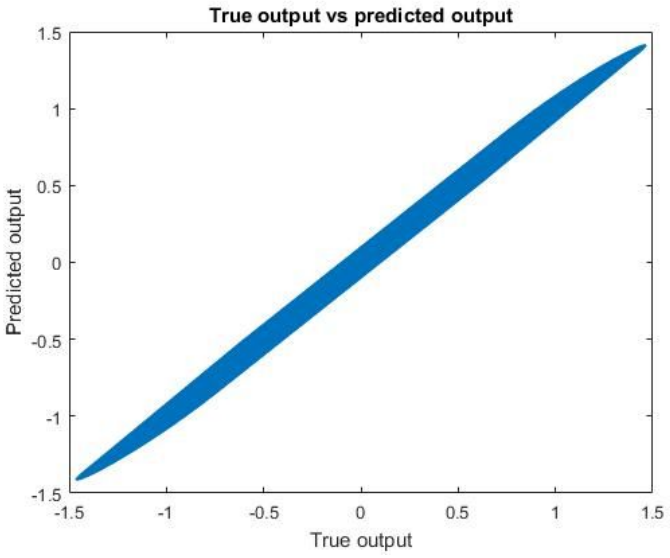
Training error corresponding to epoch-



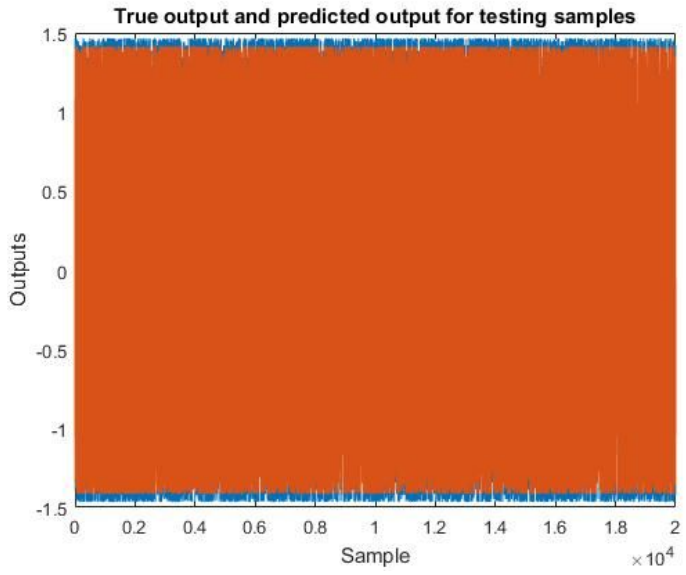
Training set -



Training set -



Testing set -



Adjusting parameters -

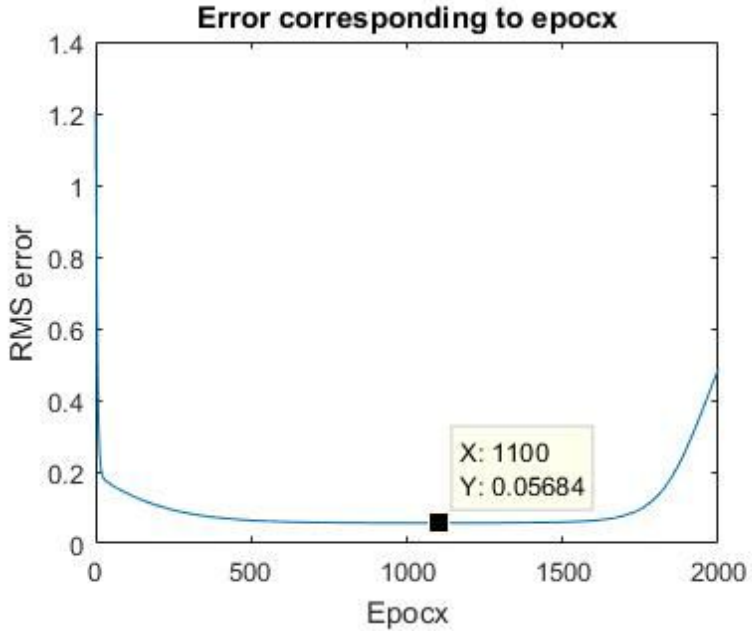
1. Epoch - From the first graph it can be seen that after some thousands iterations error is decreasing very slowly or its almost constant. So, seeing this trend 2000 epox is chosen for this.

2. Learning rate - Any learning rate greater than 1e-5 is leads to increase in error after each epox. And learning rate smaller than this leads to slow descent of the error. So, 1e-5 best suits for this data.

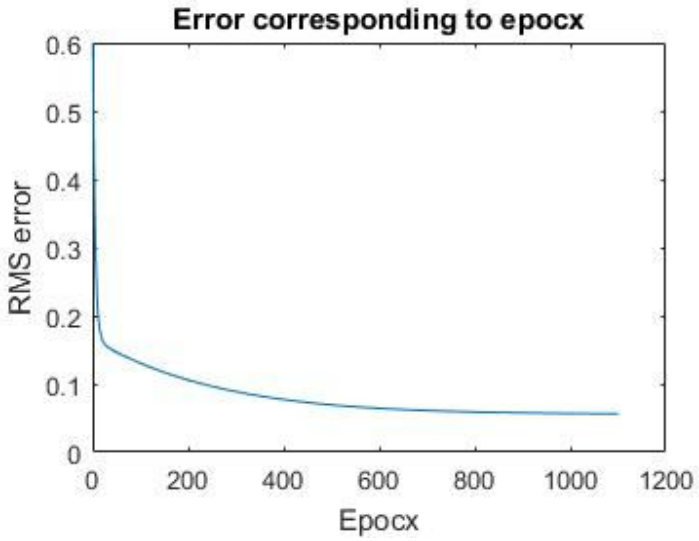
3. Hidden neurons - The general trend in hidden neuron is that more the number of hidden neurons more better is the result. Here 11 is chosen by seeing error decreasing trend using 11 hidden neurons and its neighbouring.

Thus, seeing graphs shows that MLP using sigmoidal activation function is able to approximate data well.

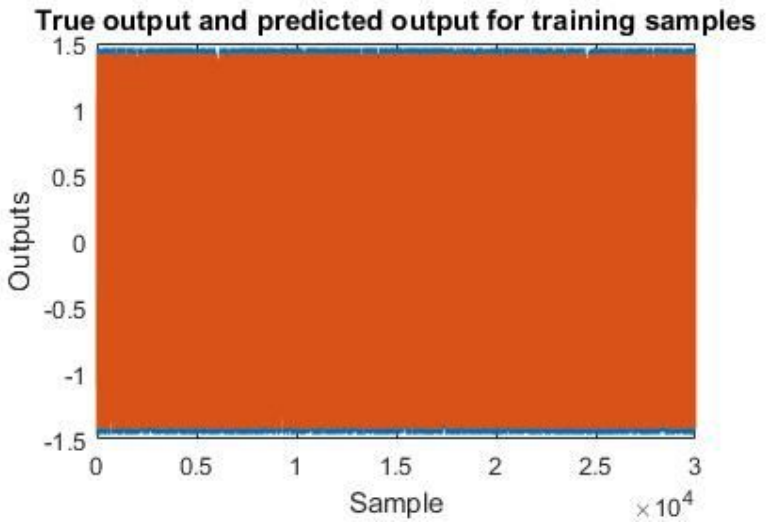
2. MLP using bi - sigmoidal activation function -



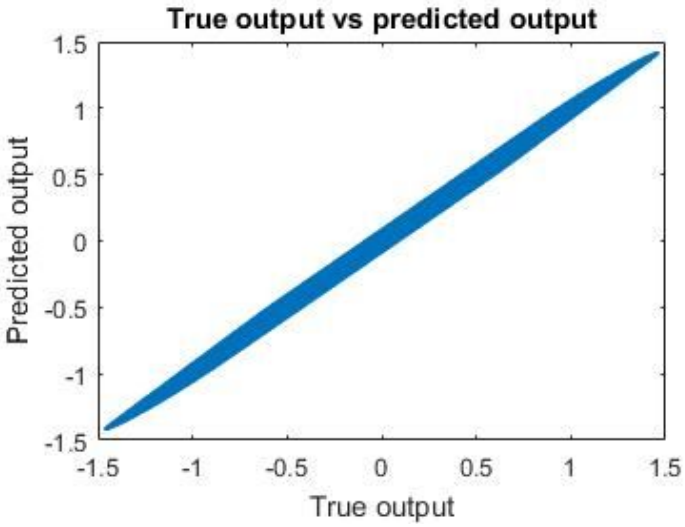
Training error corresponding to epoch - 1100



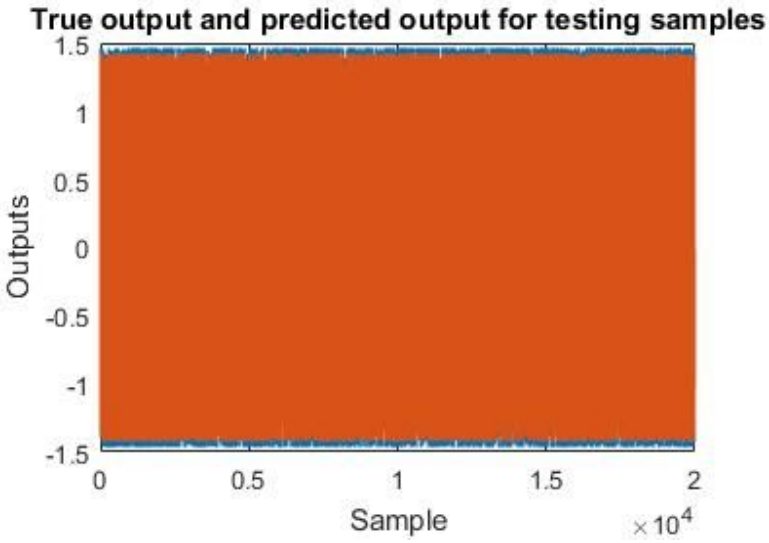
Training set -



Training set -



Testing set -



Adjusting parameters -

1. Epoch - From the first graph it can be seen that after some thousands iterations error is at its lowest and then it is again increasing. So, seeing this trend 1100 epox is chosen for this. The graph below it shows that error is considerably decreasing till end using 1100 epox.

2. Learning rate - Any learning rate greater than 1e-5 is leads to increase in error after each epox. And learning rate smaller than this leads to slow descent of the error. So, 1e-5 best suits for this data.

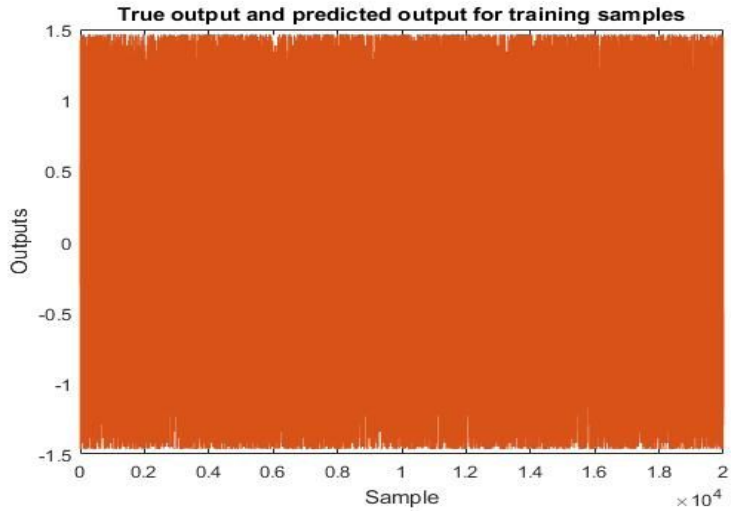
3. Hidden neurons - The general trend in hidden neuron is that more the number of hidden neurons more better is the result. Here 11 is chosen by seeing error decreasing trend using 11 hidden neurons and its neighbouring.

Here testing error is very small and there is very less difference in training and testing error, which shows it is trained well using 60% of training dataset.

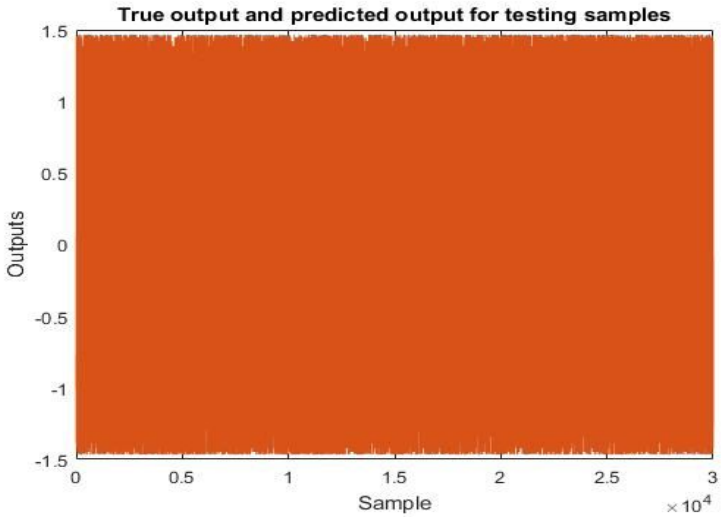
Thus, seeing graphs shows that MLP using bi sigmoidal activation function is able to approximate better than MLP using sigmoidal activation function.

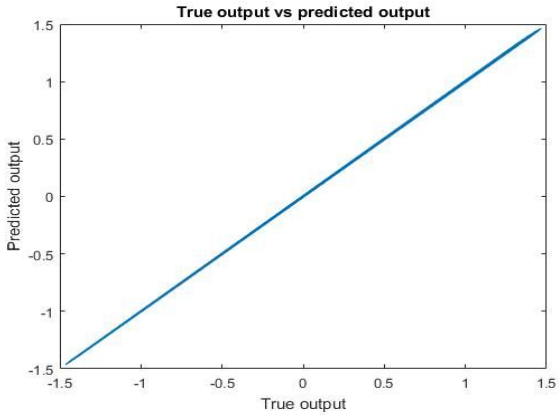
3. RBF with pseudo inverse using k-means clustering -

Training error corresponding to epoch-



Training set -



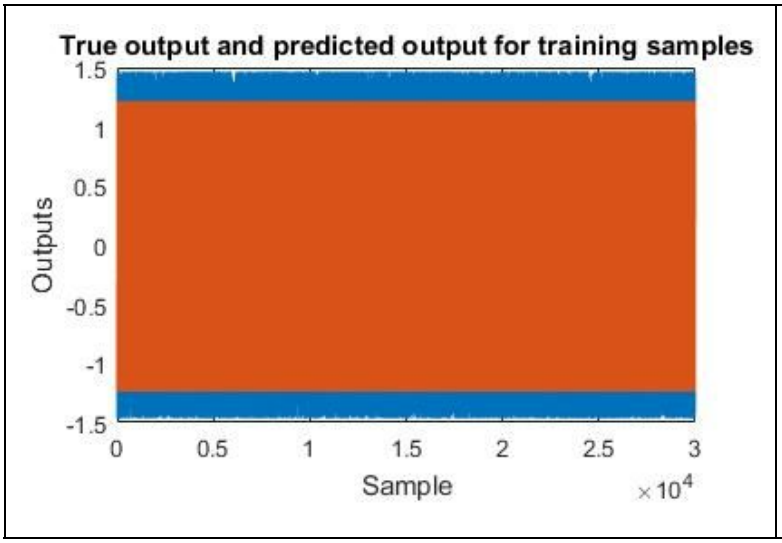


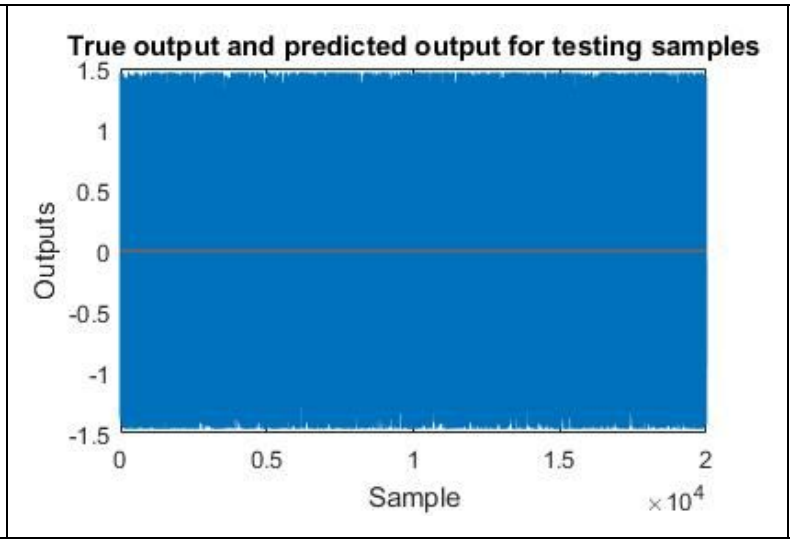
Here only number of clusters needs to be adjusted, which is decided by seeing the error decreasing trend using 15 and its neighboring hidden neurons. It is observed that increasing the number of hidden neurons leads to increase in efficiency slowly. So, a trade off is done and 15 is chosen.

From the above graphs it can be seen that true and predicted output exactly overlaps each other, only red part is visible for both training and testing set. Graph of true output and predicted output is a perfect straight line with angle 45 degree, which again shows true output and predicted output is perfectly similar.

RBF with pseudo inverse is giving best result in all the methods with a very small training and testing error.

4. RBF - gradient descent by minimising least square error -





Adjusting parameters -

1. Epoch - Table shows the training and testing error using 300 and 2000 epox, which shows that even on increasing epoch by 100 times their is very slow descent in training error, while testing error is almost constant.

2. Learning rate - Learning rate for centres, sigma and weights are adjusted by seeing the error decreasing trend with epoch. Although error is decreasing very slowly after certain number of epochs.

3. Hidden neurons - The general trend in hidden neuron is that more the number of hidden neurons more better is the result. Here 15 is chosen by seeing error decreasing trend using 11 hidden neurons and its neighbouring.

In graph, red and blue part can be easily seen differently in training data, shows RBF with gradient descent is not able to do training well and thus not able to give good approximation result of testing data.

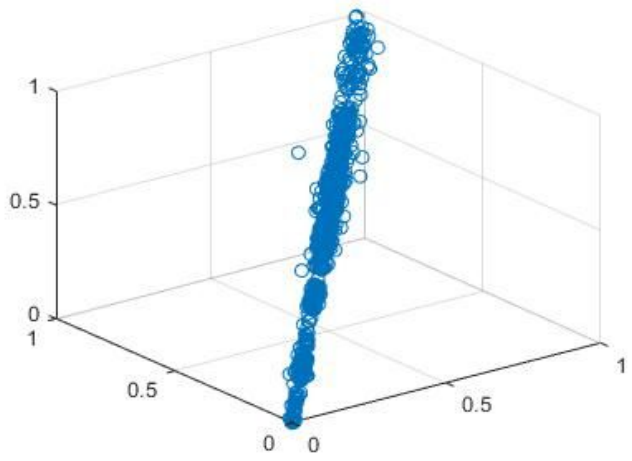
1.4 Conclusion :

From all above methods it can be concluded that RBI with pseudo inverse using centres from k-means is best suitable for training and approximating this data.

A. 2 Finance Approximation -

2.1 Knowing data -

Finance data is a future prediction problem in which 525 samples are provided to train the network. Data provided is not a normalised data with two input features and output ranging from [100.01, 741.7900]. So, here normalising data is necessary. Below is the scatter plot of the finance data after normalisation it between [0, 1], with features on x and y axis and output on z-axis.



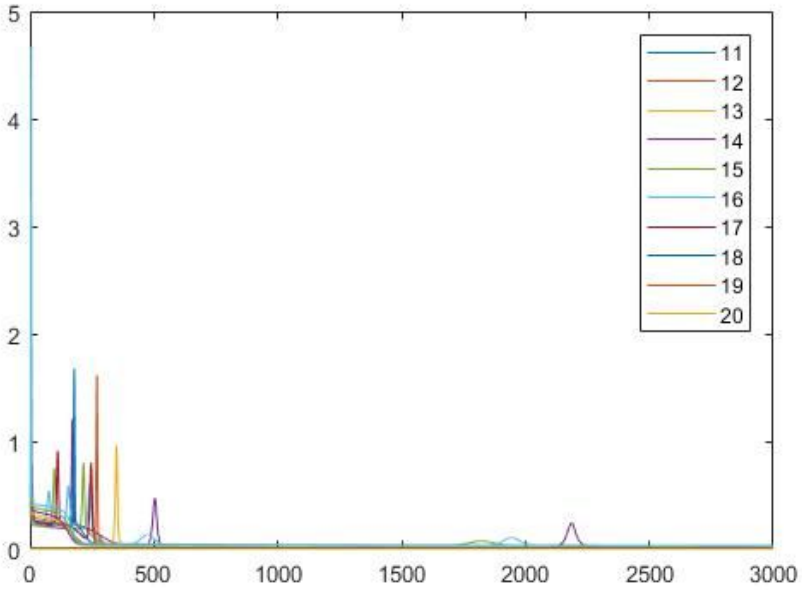
For cross validation, 60% training data as training set and remaining 40% for testing.

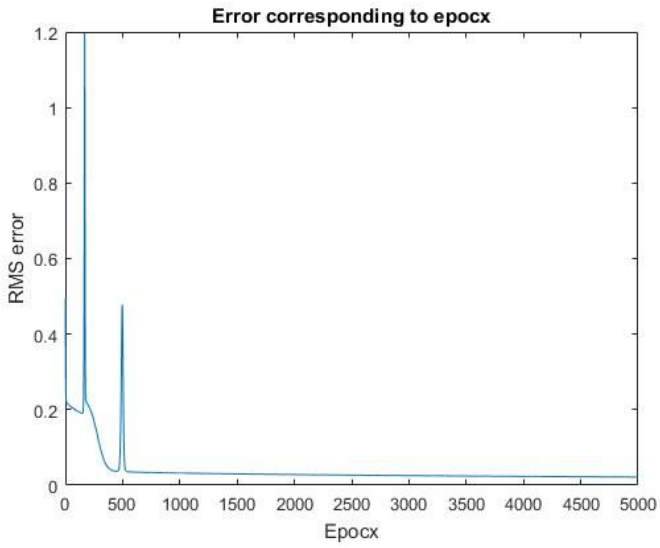
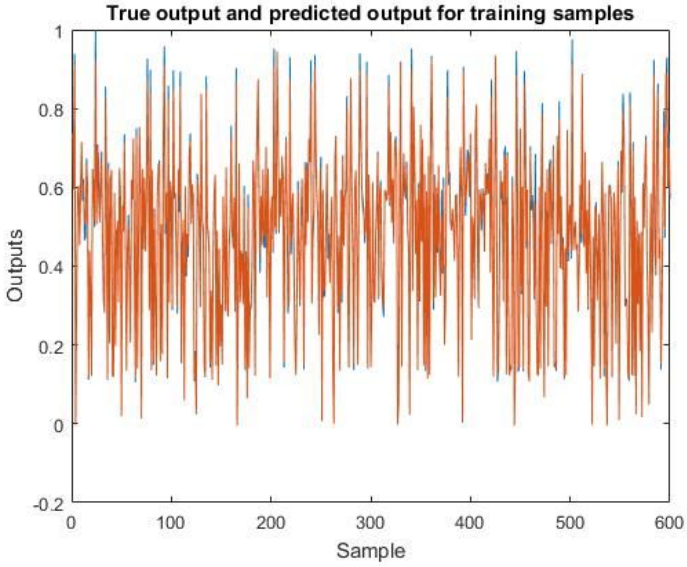
2.2 Results :

Methods	# Hidden neurons	Learning rate	Epox	Training (RMSE)	Testing (RMSE)
MLP using sigmoidal function	11	1.e-3	5000	0.02153	0.020783
MLP using bi -sigmoidal	20	1.e-3	5000	.021804	0.022046
RBF - pseudo inverse	244 16	-	-	0.011025 0.015741	0.026436 0.016174
RBF - gradient descent	19	<table><tr><th>For weights</th><th>For centres</th><th>For spread</th></tr><tr><td>1.e-6</td><td>1.e-8</td><td>1.e-8</td></tr></table>	5000	0.031567	0.52507

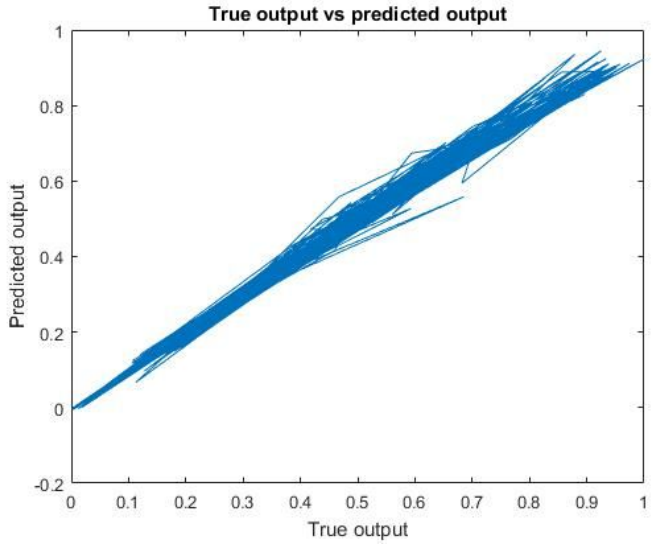
2.3 Graphs and observations :

1. MLP using sigmoidal activation function -

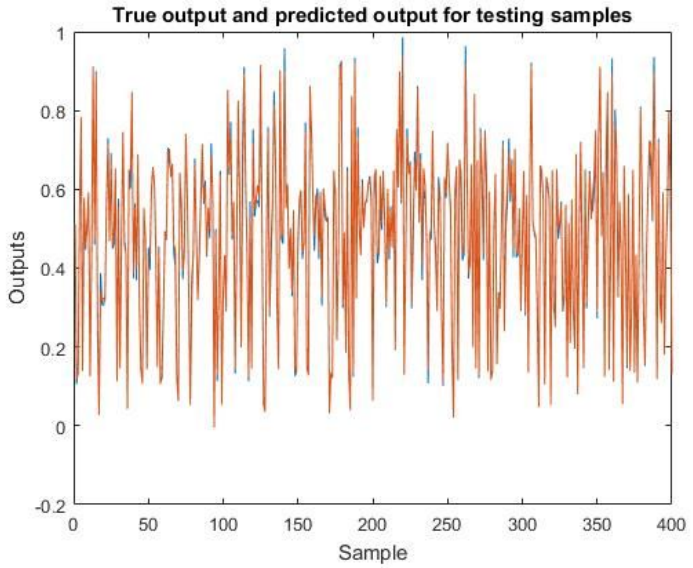


Training error corresponding to epoch-	Training set -
	

Training set -



Testing set -



Adjusting parameters -

1. Epoch - From the first graph it can be seen that after some iterations, error decreases very slowly so a large number of epox is required for good output. Here the data set is small thus large number of epox is used to get less training and testing error.

2. Learning rate - Learning rate greater than 1e-3 is leads to increase in error after each epox. And learning rate smaller than this leads to slow descent of the error. So, 1e-3 best suits for this data.

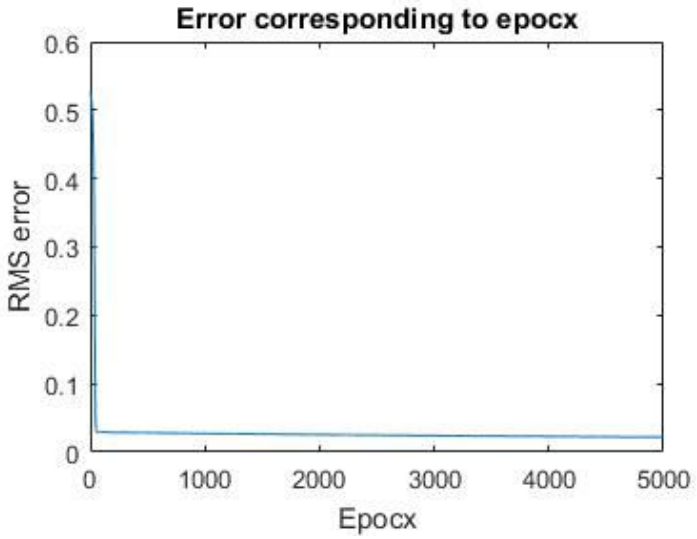
3. Hidden neurons - The general trend in hidden neuron is that more the number of hidden neurons more better is the result. Here 11 is chosen by seeing error decreasing trend using 11 hidden neurons and its neighbouring.

The graph shows that both training and testing is done perfectly, red and blue lines can be seen overlapped in both the graphs. Difference between training and testing error is also very small and the graph between true and predicted output is spread across the straight line with 45 degree, shows proper training of network using 60% of the dataset.

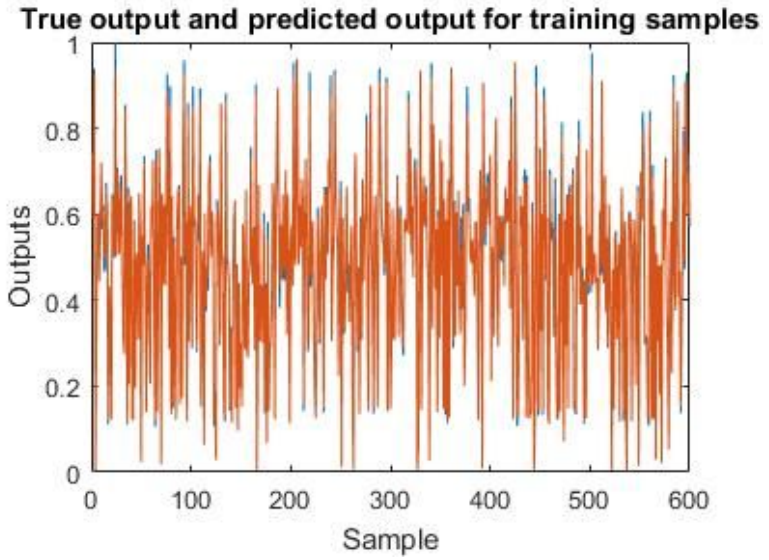
Thus, seeing graphs shows that MLP using sigmoidal activation function is sucessful to approximate finance data.

2. MLP using bi - sigmoidal activation function -

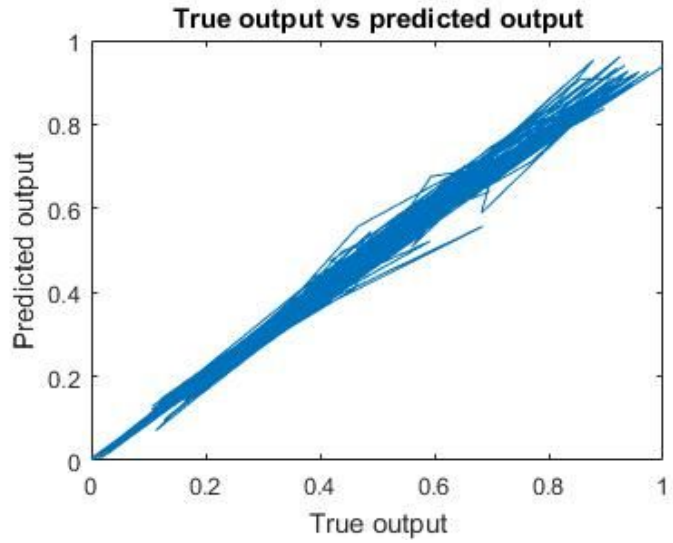
Training error corresponding to epoch-



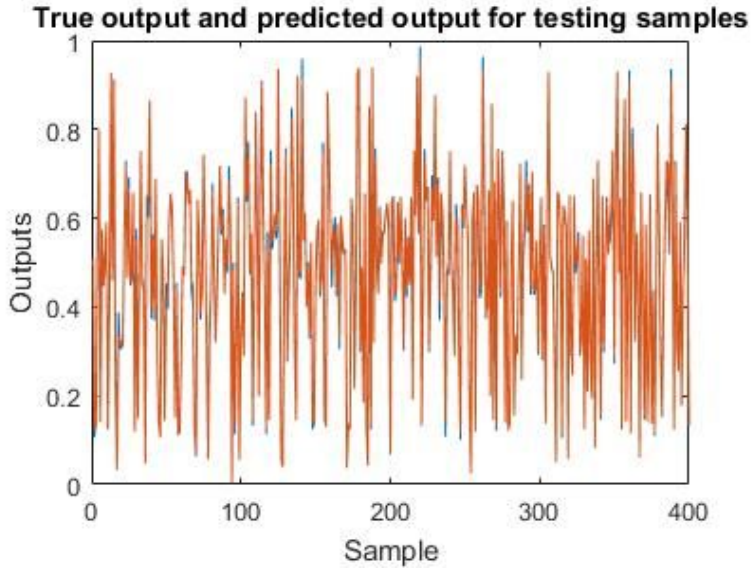
Training set -



Training set -



Testing set -



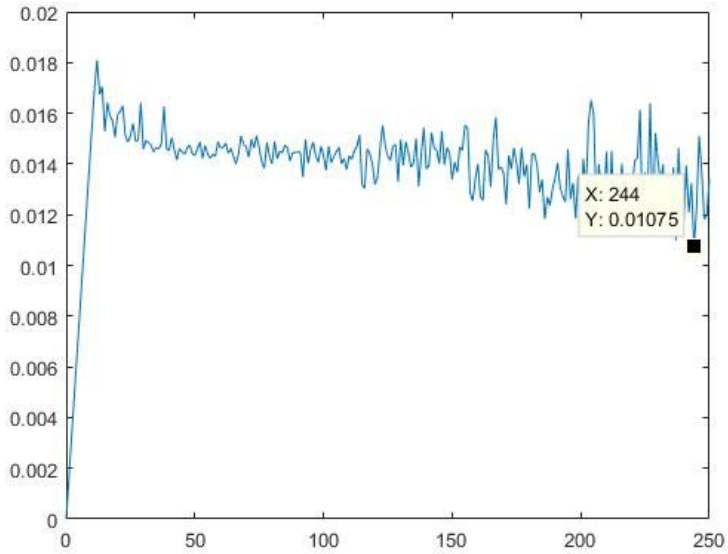
Selecting epoch, learning rate and hidden neurons is done similar to the above MLP using sigmoidal activation function.

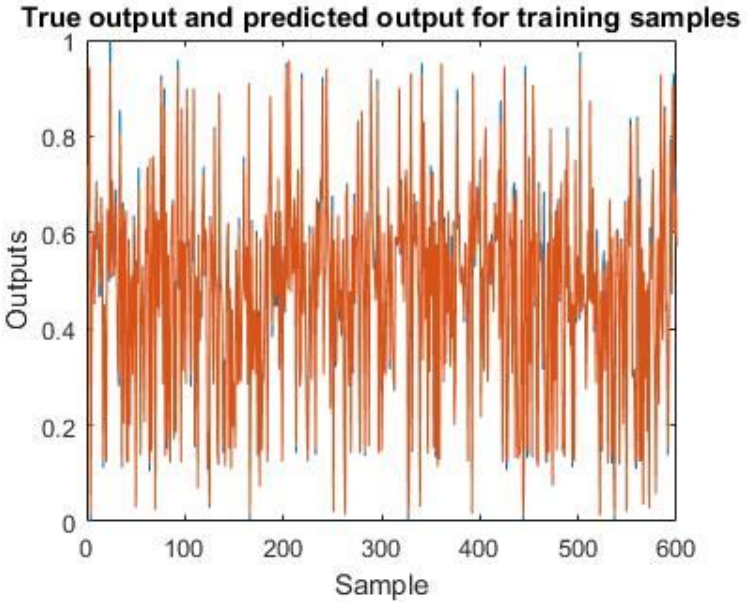
Results are also somewhat similar to above method. Here training error is slightly greater than previous model which can seen from the graph 3, with true o/p vs predicted o/p. Here the spread around straight line with 45 degree is more than previous model graph.

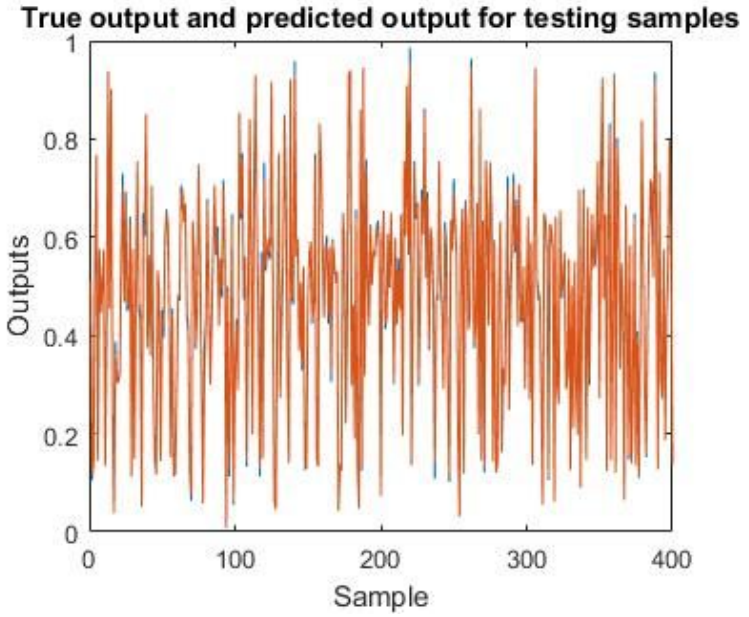
Thus, MLP with bi-sigmoidal is able to approximate well but little worse than MLP using sigmoidal activation function.

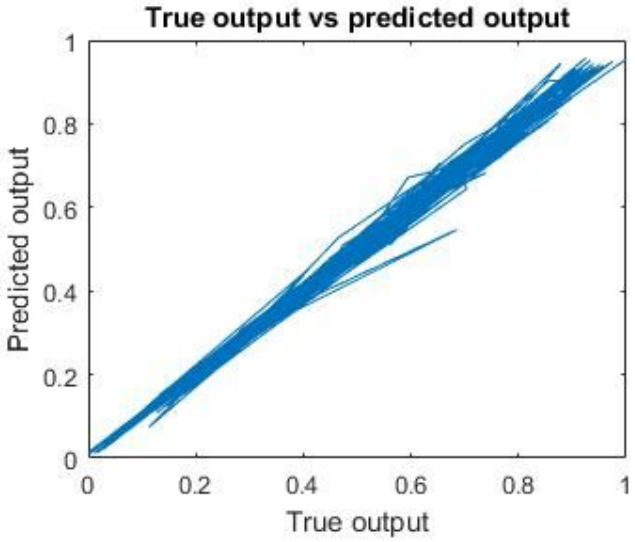
3. RBF with pseudo inverse using k-means clustering -

Number of hidden neurons on x-axis and error on y-axis. Looking to the graph output is noted for 16 and 244 hidden neurons.





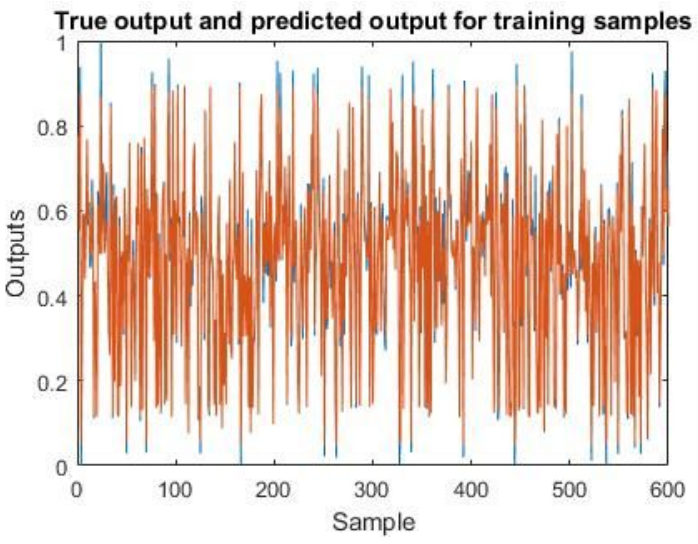


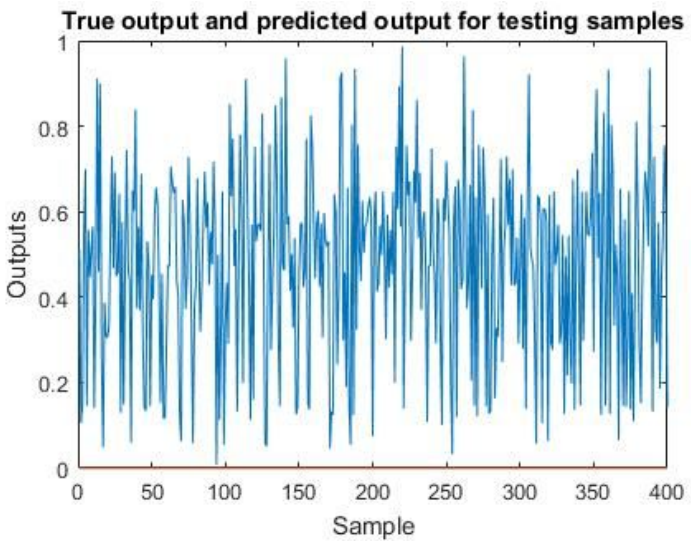


Here only number of clusters needs to be adjusted, which is decided by seeing the error decreasing trend. It is observed that increasing the number of hidden neurons leads to decrease in training error little but that leads to increase in the testing error. So, 16 is chosen with almost same training and testing error.

From the above graphs it can be seen that true and predicted output exactly overlaps each other, only red part is visible for both training and testing set. RBF with pseudo inverse is giving best result in all the methods with a very small training and testing error.

4. RBF - gradient descent by minimising least square error -





Adjusting parameters -

1. Epoch - From the first graph it can be seen that after some iterations, error decreases very slowly so a large number of epox is required for good output. Here the data set is small thus large number of epox is used to get less training and testing error..

2. Learning rate - Learning rate for centres, sigma and weights are adjusted by seeing the error decreasing trend with epoch. Although error is decreasing very slowly after certain number of epochs.

3. Hidden neurons - The general trend in hidden neuron is that more the number of hidden neurons more better is the result. Here 19 is chosen by seeing error decreasing trend using 19 hidden neurons and its neighbouring.

In graph, red and blue part can be easily seen differently in training data, shows RBF with gradient descent is not able to do training well and thus not able to give good approximation result of testing data due to which testing error is very high.

2.4 Conclusion :

From all above methods it can be concluded that RBI with pseudo inverse using centres from k-means is best suitable for training and approximating finance data.

A. Classification -

Methods used -

1. MLP classification using different error functions -

Activation Functions :	1. Sigmoid activation function in hidden layer.
	2. No activation in output layer.
Loss function:	1. Least square error function.
	2. Modified least square error.
Parameters to tune :	1. Number of hidden neurons.
	2. Learning rate.
	3. Epox.

Alogorithm :

S1. Normalising the data if needed and initializing the weights of both input and output layers.

S2. Going through all samples one by one and finding hidden layer output using sigmoidal activation function.

S3. Finding predicted output for samples and then finding error(least square error or modified least square error) in prediction of class. This error is used for updating weights of input and output neurons. After going through all samples once weights are updated.

S4. Repeating step 2 and step 3 until given number of epoch.

S5. Validating and testing the network by making confussion matrix, finding overall, average and geometric mean accuracy.

S6. Repeating this for different parameters to get optimum result.

2. RBF with pseudo inverse using centres from k-means clustering -

Radial basis function is similar to MLP, only the difference is their is a hidden layer with gaussian activation function and no input weights.

Activation Function :	1. Gaussian function in hidden layer.
	2. No activation in output layer.
Parameters to tune :	Number of clusters.

Alogorithm :

S1. Normalising data if needed.

S2. Initialising centres through K-means clustering. Initialising sigma as maximum distance between any two centres divided by square root of total number of centres.

S3. Making phi matrix using gaussian activation function giving the distance between centres and data points as parameter.

S4. Finding weight matrix of output layer using pseudo inverse of phi and actual output. Using this weights finding the predicted output.

S5. Validating and testing the network by making confusion matrix, finding overall, average and geometric mean accuracy.

S6. Repeating this for different parameters to get optimum result.

B. 1 VAL Classification -

1.1 Knowing data -

Val is a classification problem with 2432 data samples classified into 8 clusters. Data provided is a normalised data. So, here normalisation of data is not needed.

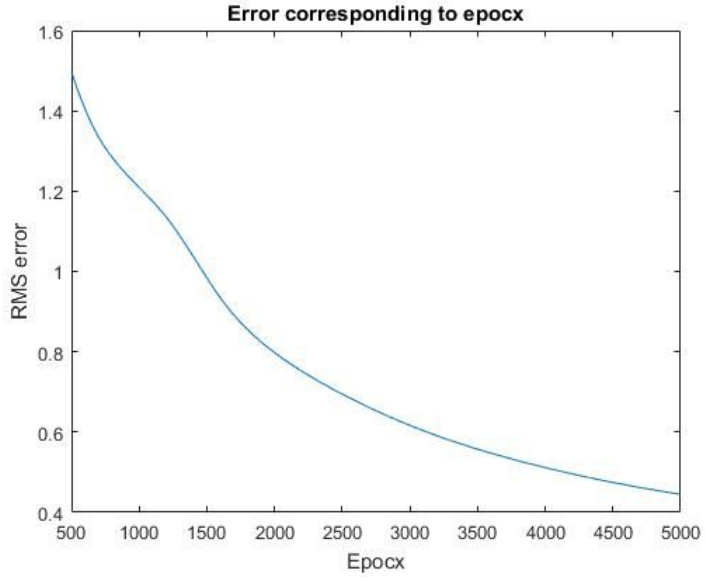
For cross validation, 60% training data as training set and remaining 40% for testing.

1.2 Results :

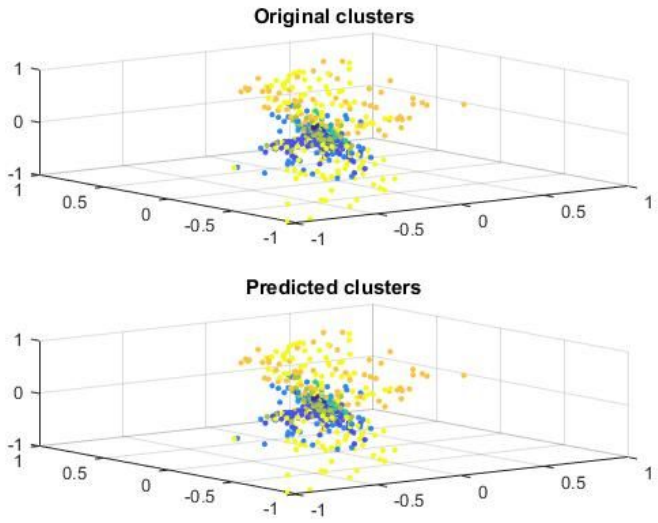
Methods	# Hidden neurons	Learning rates	Epochs	Training			Testing		
				Total	Average	Geo	Total	Average	Geo
MLP - LS	200	1e-4	5000	0.9952	0.9896	0.9581	0.9743	0.9523	0.8136
MLP - MLS	200	1e-4	5000	1	1	1	0.9969	0.9978	0.9969
RBF - pseudo inverse	174	-	-	0.9911	0.9938	0.9752	0.9897	0.9905	0.9620

1.3 Graphs and observations -

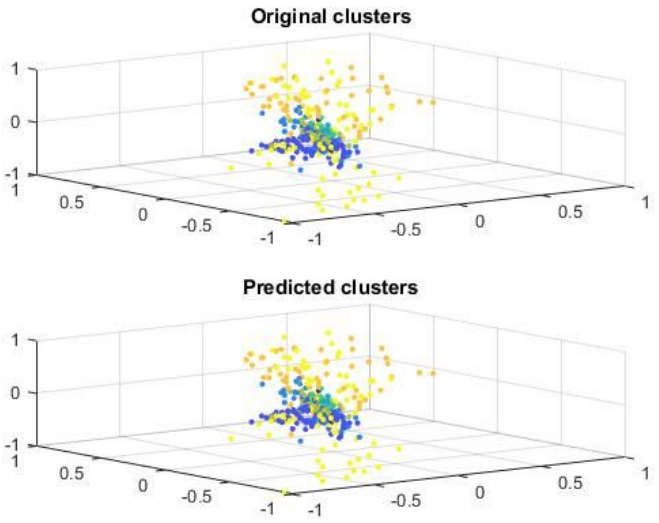
1. MLP with Least square error -



Training set -



Testing set -



Adjusting parameters -

1. Epoch - From the first graph it can be seen that after some iterations, error decreases very slowly so a large number of epox is required for good output. Here the data set is not very large thus large number of epox is used to get good training and testing efficiency.

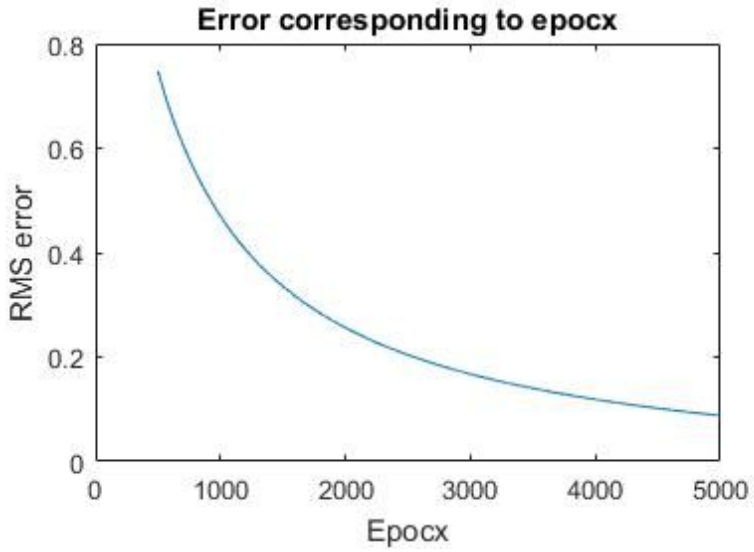
2. Learning rate - Learning rate greater than 1e-4 is leads to increase in error after each epox. And learning rate smaller than this leads to slow descent of the error. So, 1e-4 best suits for this data which leads to decrease in least mean error till 0.4448.

3. Hidden neurons - The general trend in hidden neuron is that more the number of hidden neurons more better is the result. Here 200 is chosen by seeing error decreasing trend using 200 hidden neurons and its neighbouring.

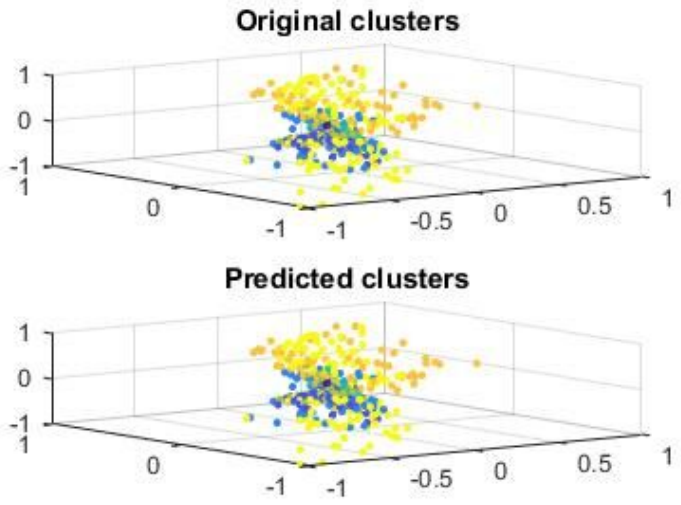
The graph shows that both training and testing is done perfectly, similarity in the clusters in training and testing set can be seen. Training and testing efficiency is close to each other, shows proper training of network using 60% of the dataset.

Thus, seeing graphs shows that MLP using least square error as loss function is successful to classify data set.

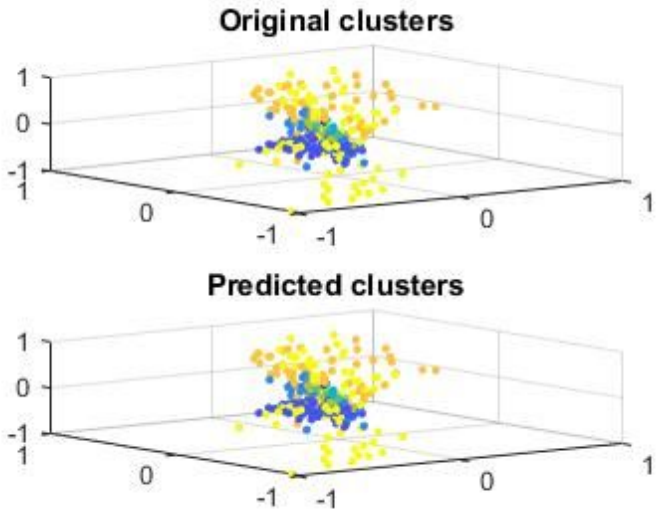
2. MLP with Modified least square error function :



Training set-



Testing set-

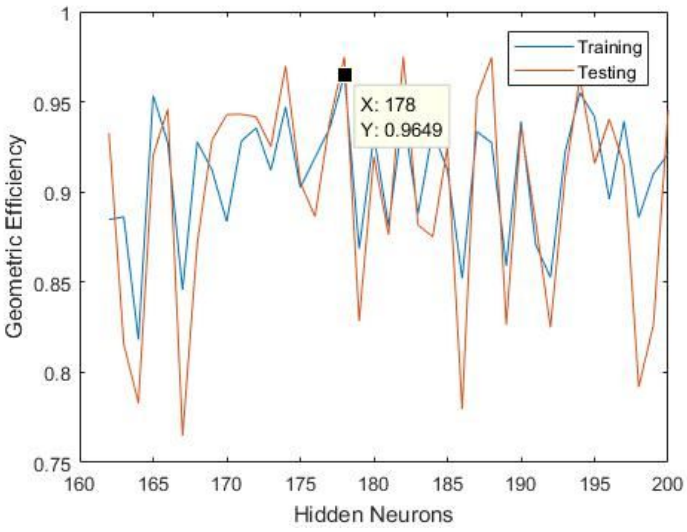


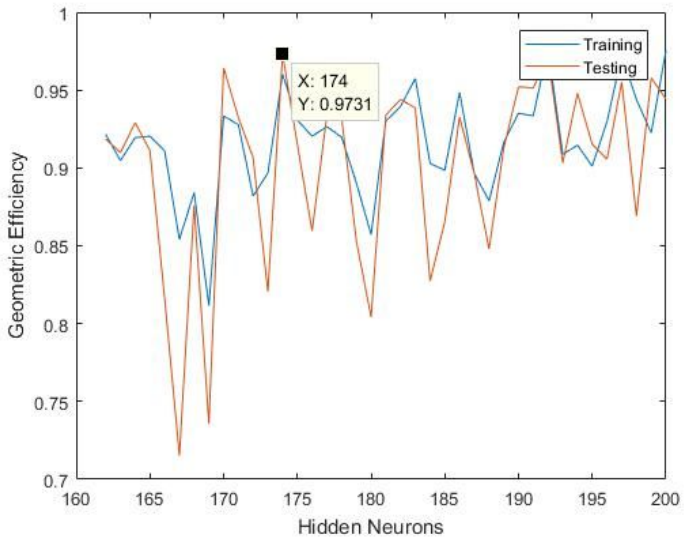
Selecting epoch, learning rate and hidden neurons is done similar to the above MLP using least square error function.

Results are better than above method. Here training efficiencies are 100%, showing proper training of the training set data, can be seen from graph. And testing efficiencies are also near to hundred.

Thus, MLP with modified least square error function is able to do classification of data set perfectly.

3. RBF pseudo inverse using centres from k-means :





Here only number of clusters needs to be adjusted, which is decided by seeing the error decreasing trend. It is observed that increasing the number of hidden neurons leads to increase in training efficiency. So, 174 is chosen as trade off having good geometric mean and overall efficiency.

The results from this are better than MLP with LS error function.

1.4 Conclusion -

From all above methods it can be concluded that MLP with modified least square error function is best suitable for training and classification of this data.

B. 2 MHAD Classification

2.1 Knowing data -

BERK is a classification problem with only 99 data samples classified into 11 clusters. Data provided is a normalised data. So, here normalisation of data is not needed. But here number of features are lot more than training data samples so, PCA reduction is used to get good training and testing efficiency.

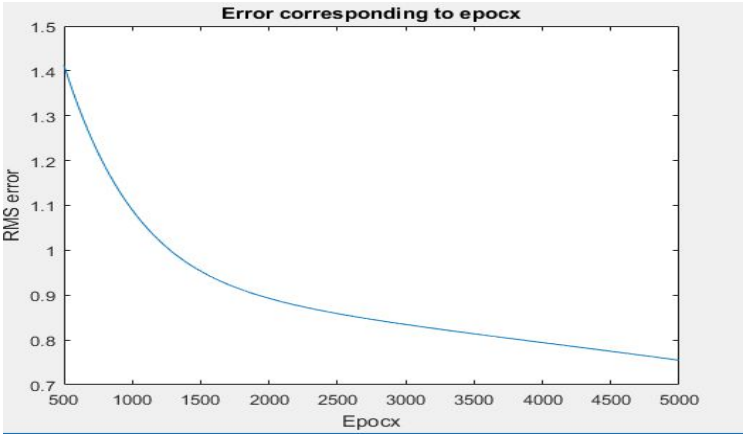
For cross validation, 90% training data as training set and remaining 10% and some samples from training is used for testing.

2.2 Results :

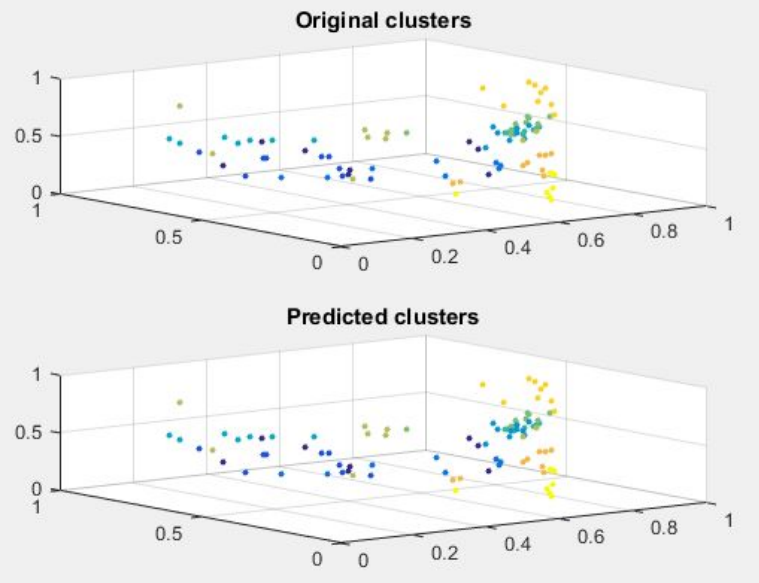
Methods	Reduced features	# Hidden neurons	Learning rates	Epochs	Training			Testing		
					Total	Geo.	Average	Total	Geo.	Average
MLP - LS	34	35	1e-3	5000	0.9785	0.8889	0.9798	0.9459	0.7303	0.9515
MLP - MLS	34	35	1e-3	5000	1	1	1	0.9756	0.8660	0.9773
RBF - pseudo inverse	34	35	-	-	0.9677	0.8315	0.9697	1	1	1

2.3 Graphs

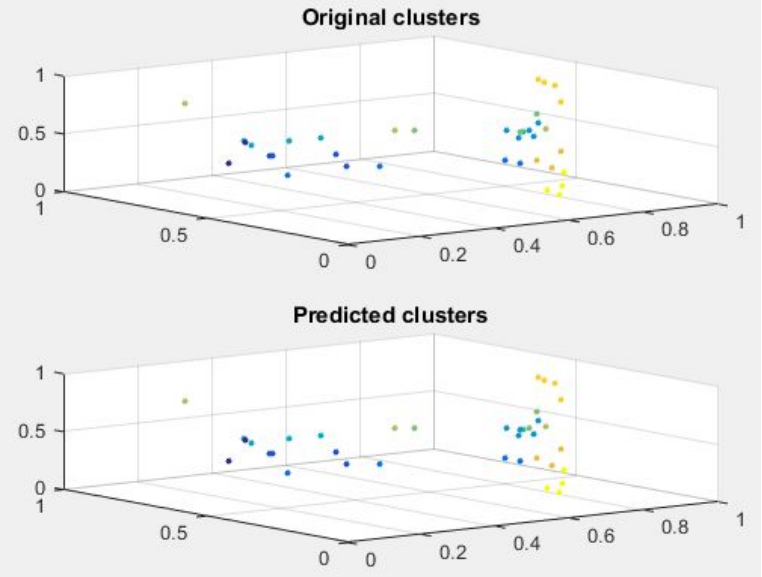
1. **MLP with LS error function -**



Training Set -



Testing Set-



Adjusting parameters -

1. Epoch - From the first graph it can be seen that after some iterations, error decreases very slowly so a large number of epox is required for good output. Here the data set is not very large thus large number of epox is used to get good training and testing efficiency.

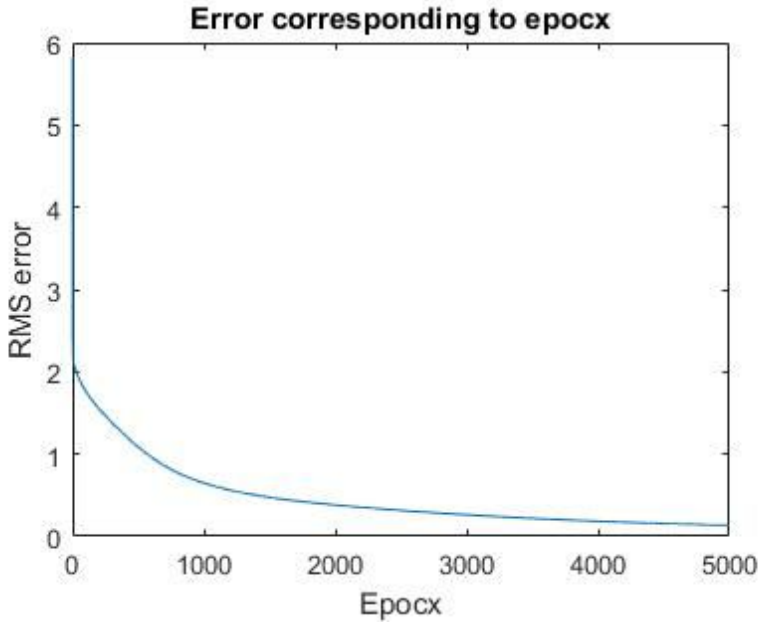
2. Learning rate - Learning rate greater than 1e-3 is leads to increase in error after each epox. And learning rate smaller than this leads to slow descent of the error. So, 1e-3 best suits for this data.

3. Hidden neurons and reduced features - The general trend in neurons and features is that more the number of hidden neurons and features selected more better is the result. Here 34 features is chosen by comparing efficiencies.

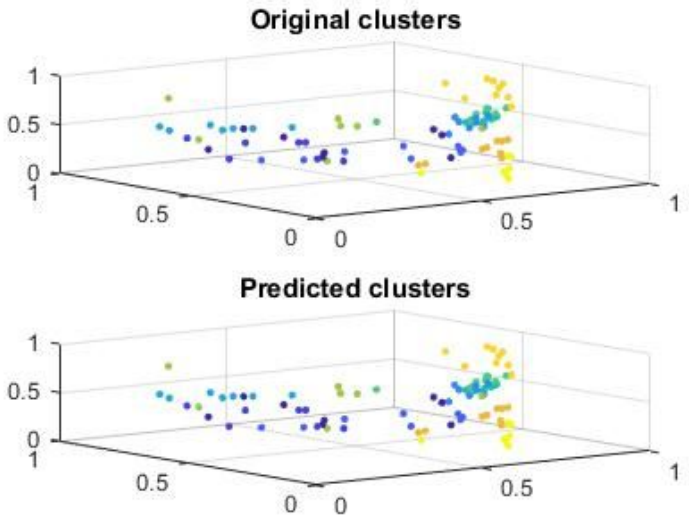
The graph shows that both training and testing is done good, similarity in the clusters in training and testing set can be seen. Training and testing efficiency is close to each other, shows proper training of network using 90% of the dataset.

Thus, seeing graphs shows that MLP using least square error as loss function is somewhat successful to classify data set.

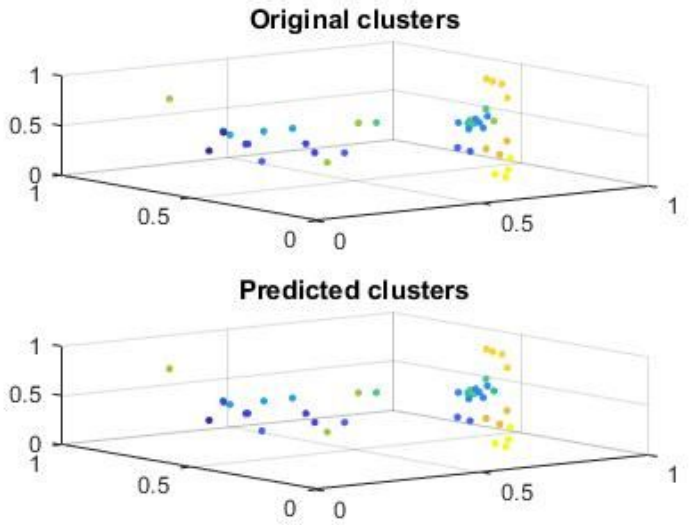
2. MLP with MLS error function -



Training Set -



Testing Set-



Selecting epoch, learning rate and hidden neurons is done similar to the above MLP using least square error function.

Results are better than above method. Here training efficiencies are 100%, showing proper training of the training set data, can be seen from graph. And testing efficiencies are also good.

Thus, MLP with modified least square error function is able to do classification of data set well.

3. RBF pseudo inverse using centres from k-means :

Here only number of clusters needs to be adjusted, which is decided by seeing the error decreasing trend. It is observed that increasing the number of hidden neurons leads to increase in training efficiency. So, 34 is chosen as trade off having good geometric mean and overall efficiency.

Its results are very oscillating, thus this model is not good for classification of this data.

2.4 Conclusion -

From all above methods it can be concluded that MLP with modified least square error function is best suitable for training and classification of this data.