Selected Topics in Neural Networks(IT-481)

Assignment Report

Name- Keya Desai
ID- 201501012

INDEX

1.Artificial Neural Network:

Artificial Neural Networks mimic the biological neural networks for computation. ANNs are composed of multiple nodes, which imitate biological neurons of human brain. The neurons are connected by links and they interact with each other. The nodes can take input data and perform simple operations on the data. The result of these operations is passed to other neurons. The output at each node is called its activation or node value.

There are two types of learning algorithms for neural networks:

1) Batch-wise learning algorithms - MLP, RBF

In this type of learning method, training set is already provided and the algorithm reviews the same training set multiple times to learn.

2) Sequential learning algorithms - RAN, MRAN, EMRAN, SRAN, Meta-cognitive learning

In this type of learning, each training data appears sequentially and only once. Training set is not known a priori.

2.Multi-Layer Perceptron (MLP) Learning Algorithm:

It is a feed-forward type of ANN. The hidden units are called the neurons which have a linear or nonlinear activation function to map inputs to outputs.
It uses backpropagation algorithm which is an error-minimizing algorithm to tune the weights to generate the desired mapping.

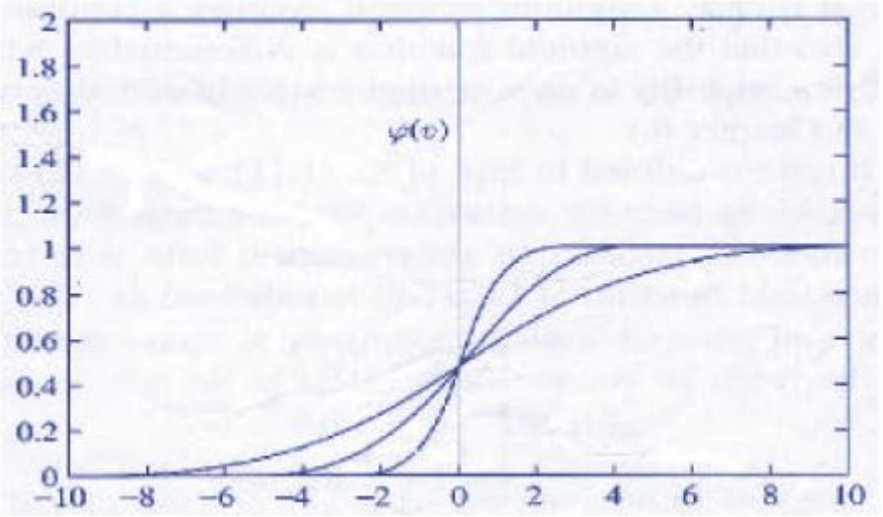
A.Learning Approaches for MLP:

Activation Functions : Activation function maps the input parameters(features) to ouput parameter(hidden layer neuron).
The activation function should be continuous and its derivative should exist at each point. This is essential when updating the parameters through gradient descent.
There are many different Activation functions such as Gaussian Symmetric Activation Function, Sigmoid Activation Function, Elliot Symmetric Activation Function etc.

We have used the following two Activation functions:

Unipolar Sigmoidal : $Ø = 1/(1+e^{-av})$
Bipolar Sigmoidal : $Ø = (2/(1+e^{-av}))-1$

In the sigmoidal functions, slope parameter ‘a’ controls the shape.



Loss Functions

- **Least Square (LS) :**

$$\sum_{i=1}^{i=n} \left(Y_{observed,i} - Y_{calculated,i} \right)^2$$

It computes the mean square difference of the predicted and the actual output for all data samples and is used as a performance measure for approximation as well as classification problems.

- **Modified Least Square (MLS) :**

MLS = 0 ; y(target,i)*y(calculated,i)>0
= LS; otherwise

Besides these,

- **Fourth Power error function**(FPE) which penalises the error heavily and
- **Cross Entropy error function** can also be implemented.

B. Parameters to be varied:

- Architecture: Number of layers and number of hidden neurons in each layer
- Learning Rate(alpha)
- Number of Epochs : The number of times the training set is revised

Approach to find optimal parameters for RBF and MLP Approximation and Classification Problems:

To find the optimal orientation of the parameters in both MLP and RBF, we started off by changing the number of hidden neurons since that affects the outputs the most. These were changed till no substantial increase was recorded. Once, we knew the number of hidden neurons, our next step was finding lambda for the problem. Here, small changes do not affect much so we stuck to 10 times or 1/10 times values. We had to use 1/100 time values in some cases when 0.01 lambda value was too large for the data set and was making it diverge. Finally, the number of epochs required were calculated by noting changes in errors in the validation output. Larger value of epochs lead to increase in overfitting and thus the error.

C. Classification Problem using MLP

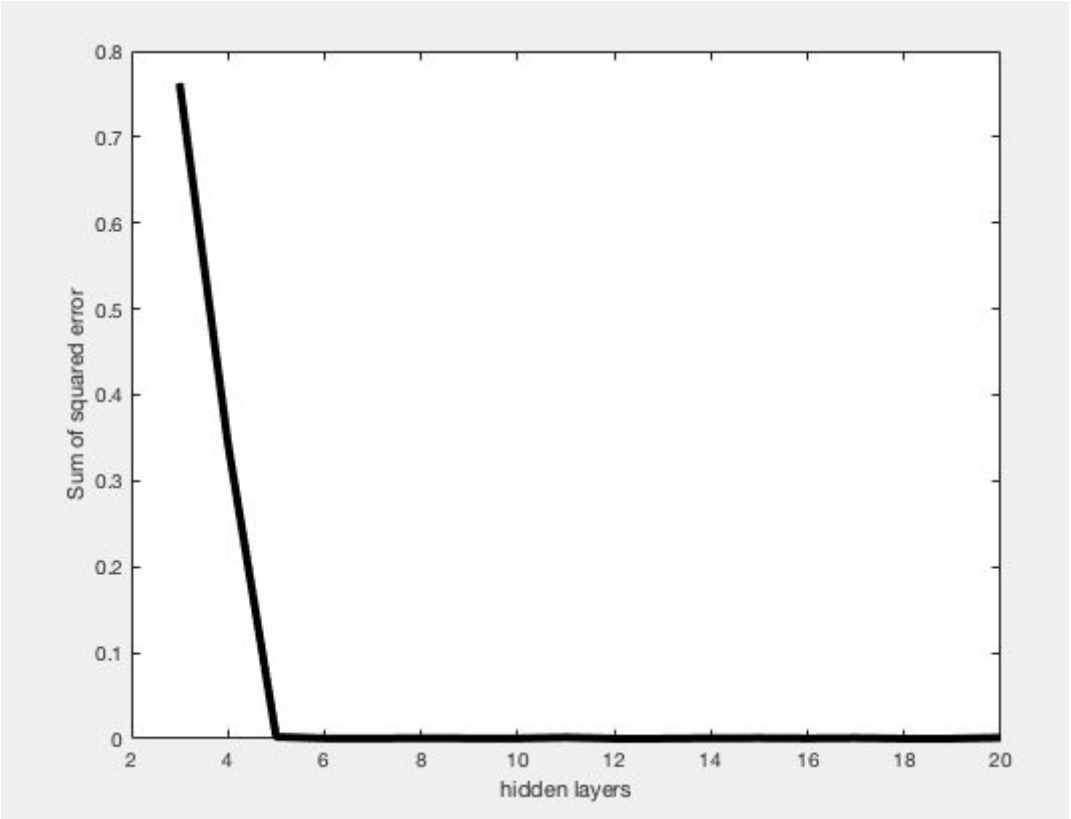
- **Effect of different Error measure**

In the classification problem, if the output is outside the range [-1,+1], it means it is more confident about the element being the part of that class or not. So, it should not be an error. If we use the LS error, it will give an error if our output is outside the range [-1,1] which is not really an error. This behaviour of the hinge loss error function in MLS, makes it a better performance measure for our model.

Data Set	MLP with LS					MLP with MLS				
	epoch	Hidden layers	Overall Accuracy	Average Accuracy	Geometric Accuracy	Epoch	Hidden layers	Overall Accuracy	Average Accuracy	Geometric Accuracy
ACTREC3D_27	1000	20	100	100	100	1000	20	100	100	100
BERK	4000	20	77.7778	86.3636	82.7753	4000	20	88.8889	89.3939	84.9687

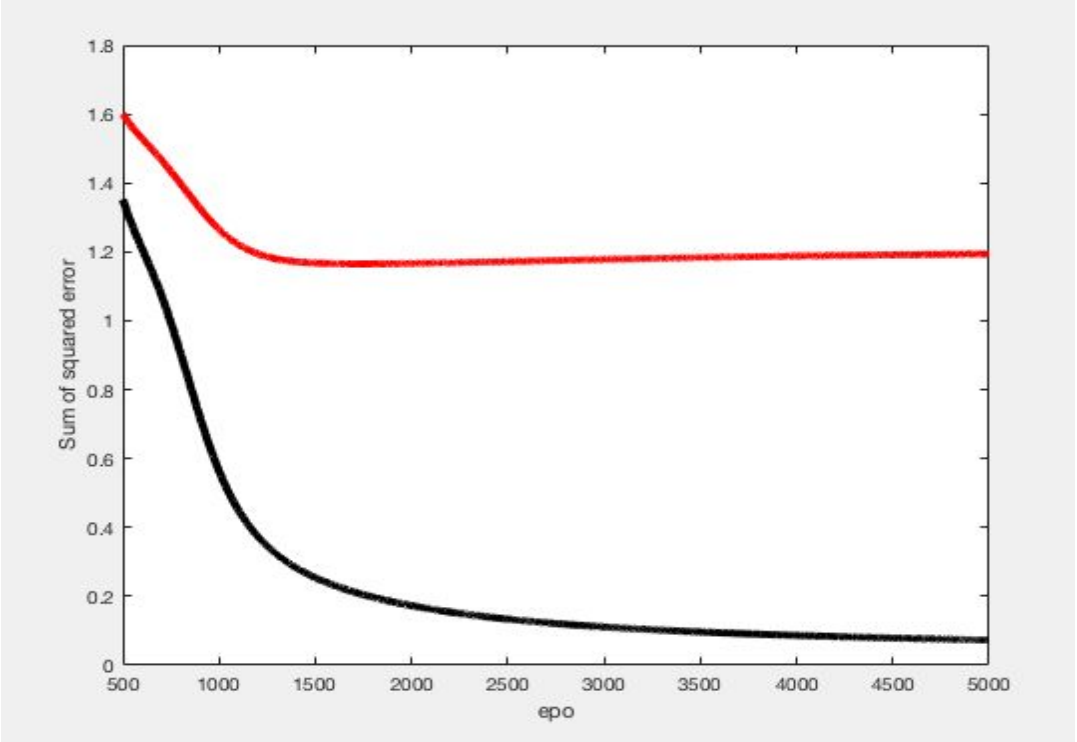
- **Effect of changing the Neural Structure**

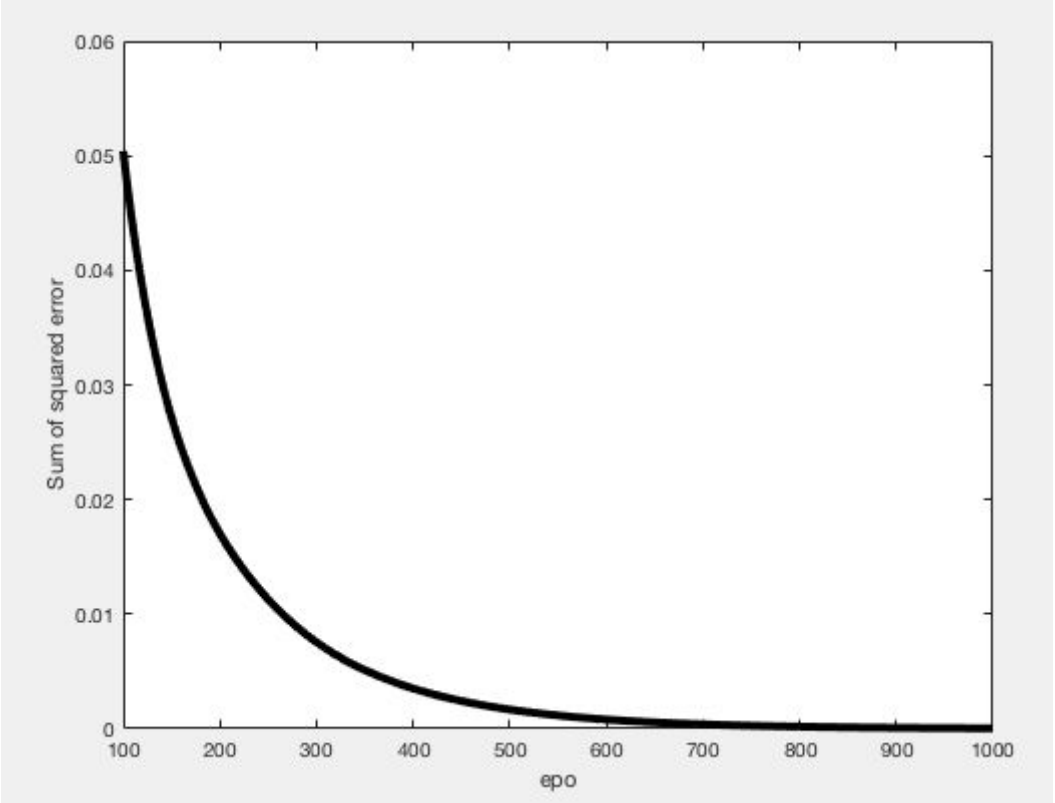
On increasing the number of hidden neurons, the accuracy increases initially but then saturates. If we have too many neurons compared to the input, it will lead to overfitting. So, the training accuracy will increase but the testing accuracy will decrease.



- **Effect of number of epochs**

On increasing the number of epochs, the error decreases initially and then saturates. And if we have too many epochs, it leads to overtraining.





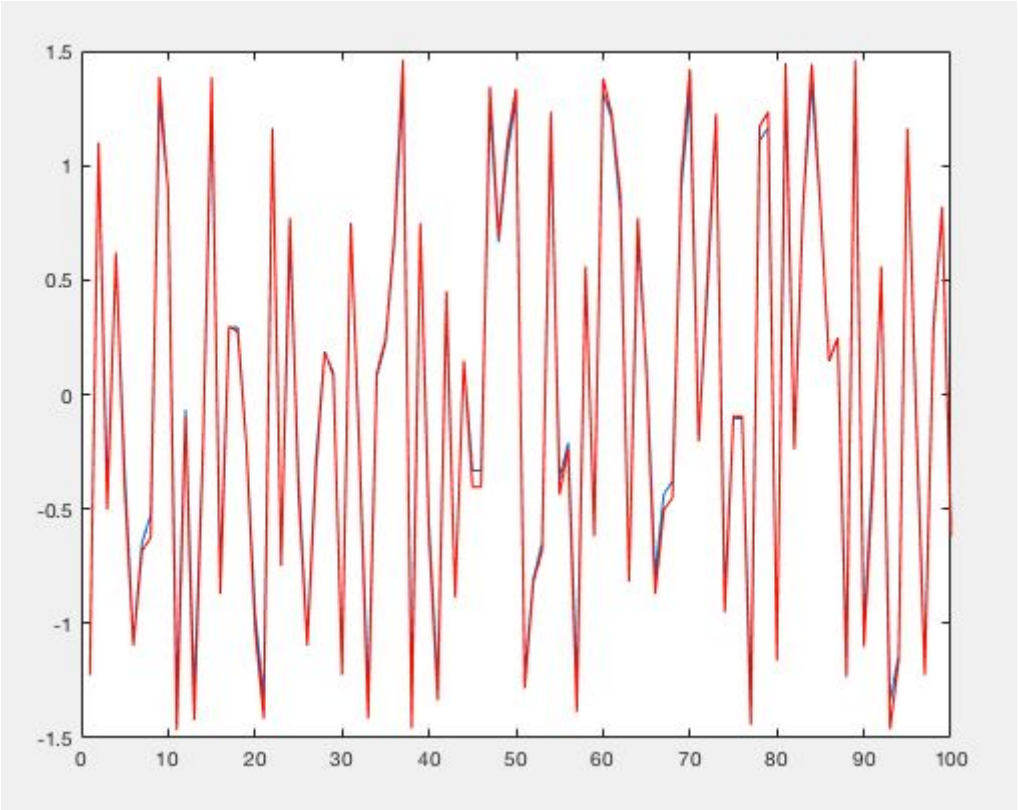
D.Approximation Problem using MLP

- **Effect of changing Activation Function:**

Unipolar Sigmoidal activation function gives better results as compared to the Bipolar Sigmoidal activation function.

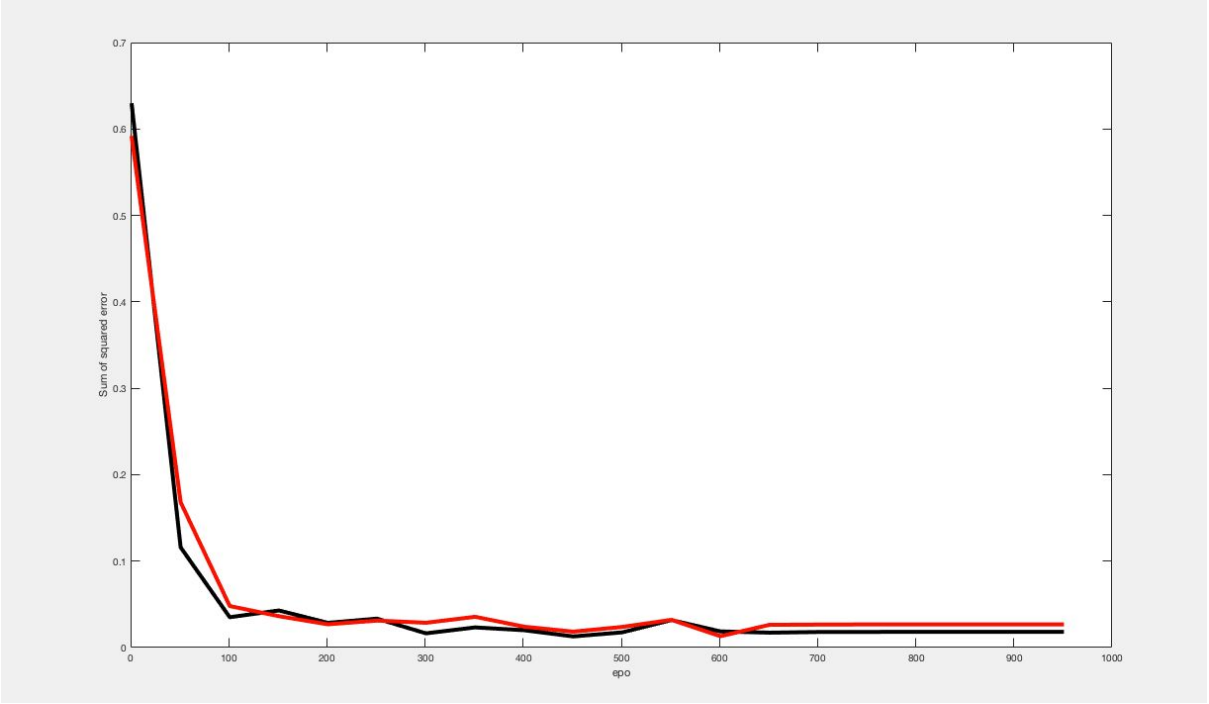
SI data set				
Model	Epoch	Hidden Neurons	Training RMSE	Validation RMSE
Uni	100	8	0.0581	0.0581
Bi	100	20	0.8897	0.9234

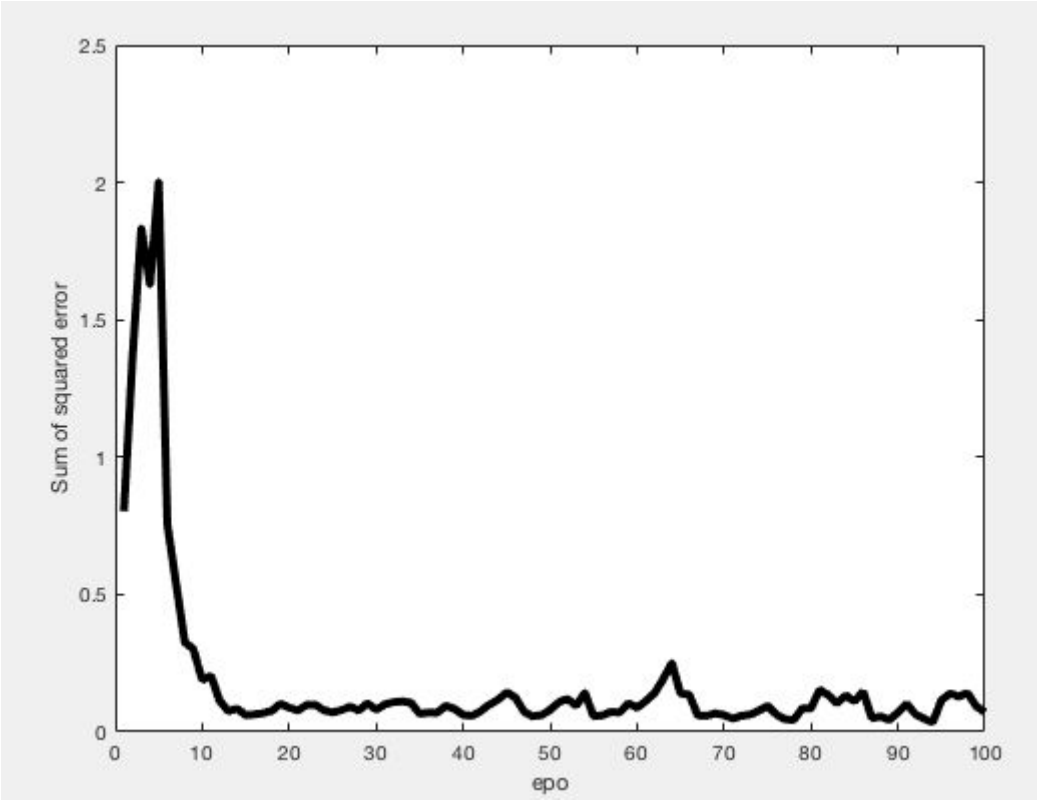
Finance data set				
Model	Epoch	Hidden Neurons	Training RMSE	Validation RMSE
Uni	100	10	142.7238	141.8229
Bi	750	7	142.7238	141.8229



- **Effect of number of epochs**

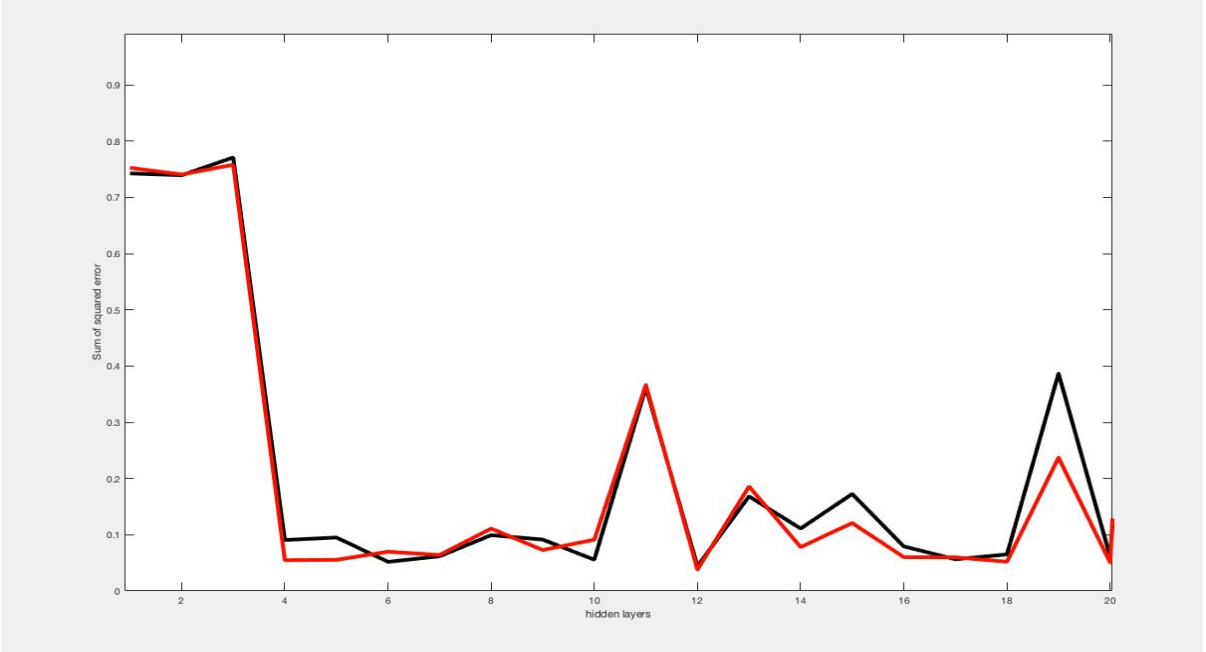
On increasing the number of epochs, the error decreases initially. And if we have too many epochs, it leads to overtraining.





- **Effect of changing the Neural Structure**

On increasing the number of hidden neurons, the error decreases initially but then saturates. If we have too many neurons compared to the input, it will lead to overfitting. When overfitting occurs, our model tries to imitate (or memorize) the training set rather than generalizing (learning) for any given data. So, the training accuracy will increase but the testing accuracy will decrease.



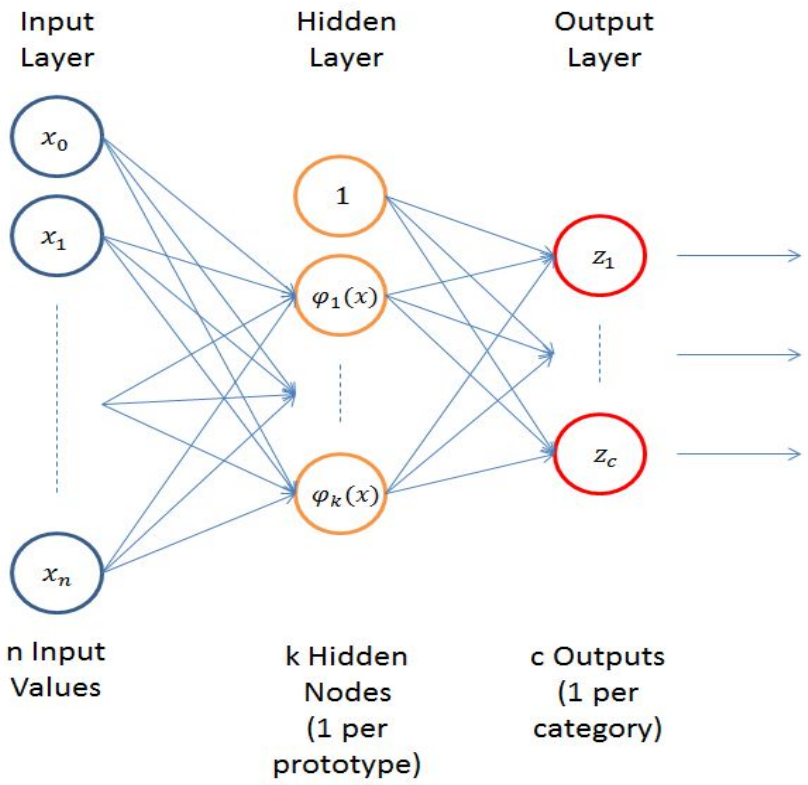
- **Update Rule for parameters, input weights and output weights**

We can update the parameters i.e. input and output weights either ***epoch wise*** or ***sample wise***. In the epoch wise update, we keep on adding error for each training example and after each epoch, update the weights. Whereas in the sample wise update, we update weights at every sample for every epoch. If the data set is very large, it is better to use the sample wise update as epoch wise update will lead to a large change in weights and may not result into convergence of the error function.

Finally, to check the accuracy of code on the dataset, we have calculated the root mean square errors by comparing the predicted output with target values.

3. Radial Basis Function (RBF) Learning Algorithm:

RBF is a model that can be used for implementing artificial neural network model. RBF uses gaussian functions to find the approximate function that can be used for prediction.



A.Learning Approaches of RBF

Initialization of the centres

- **Clustering algorithm**

  The data points for centers can be initialized using the unsupervised learning K-means algorithm. By using this algorithm, we will be able to obtain centers where most of its neighbouring element belong to the same class. Hence, the performance of RBF can significantly improve using this method for initialization of the centers.

- **Random initialization**

  We can randomly initialize the centres and use gradient descent to find the optimum centres for classification problems.

Initialization of the spread (width)

The initialization of the spread for the gaussian distribution for each center can be done by looking at the maximum distance between the centers and the number of centers.

Different Loss function

Least square (LS) error

Modified least square (MLS) error

Learning approach of RBF used in implementation:

1. We initialized the 'K' centres using data points which are randomly selected from the dataset. The initialization of the spread is done using random function generator. In order to converge to these parameters, we implemented the gradient descent algorithm.
2. We initialized the K centres from the data points, then initialized the weights and bias using pseudo inverse of the G matrix shown below. We then updated the weights, centres and spread using gradient descent to find their optimum values.

B. Approximation Problem using RBF

A continuous function can be represented as an estimate for the approximation problem. Theoretically, sum of different exponential functions can generate almost any continuous function. Since RBF is represented by sum of weighted exponential functions, it acts as a good estimate for approximating a function in this type of problems.

- **Effect of changing number of centers**

We observed that on increasing the number of centers, the model performance increases i.e. it tries to fit increasingly better on the training set. Initially, on increasing the number of centers there is a substantial increase in the accuracy on the validating set. But when the number of center neurons start overfitting the training set, then the accuracy of the validating set starts falling.

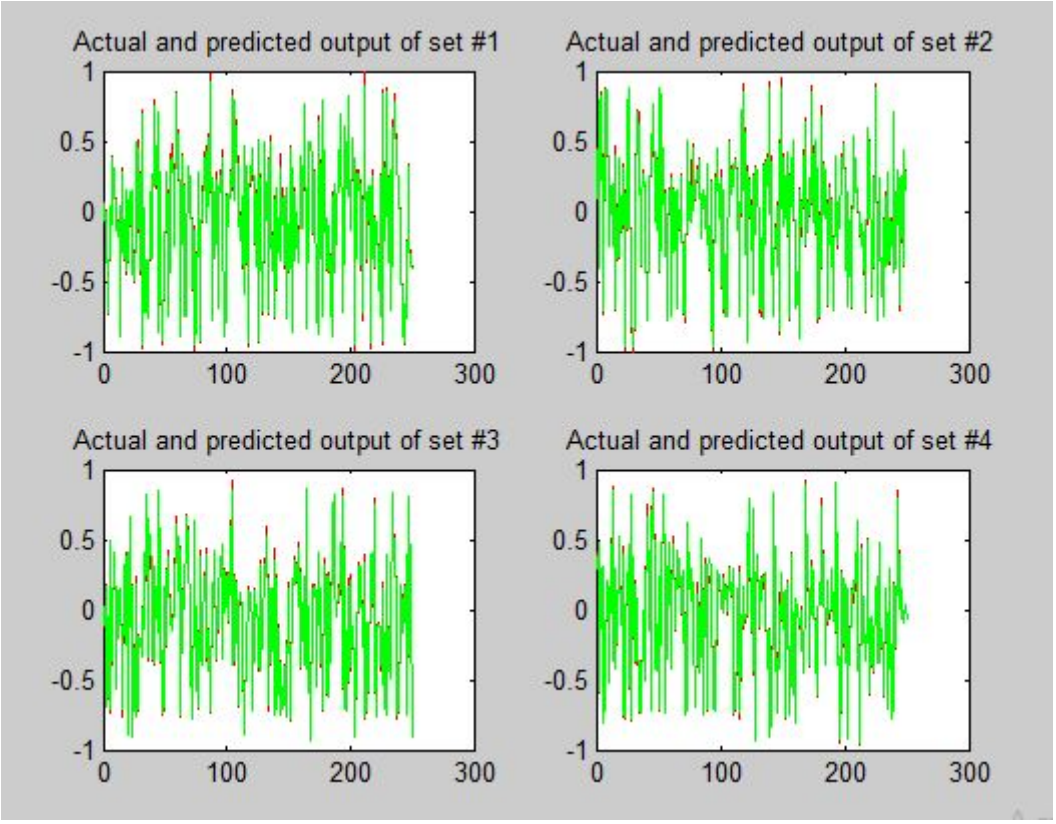
- **Effect of changing number of epochs**

We implemented the gradient descent algorithm to update various parameters like the spread, width and the weights at the end of each batch. The batch can be sample wise or it can be epoch-wise. We observed that on increasing the number of epochs for a specific value of learning rate, error decreases.

In gradient descent, the parameters are updated such that the training error is minimized to a local minima. After reaching the minima, error and accuracy are saturated. If we choose different initial conditions and learning rate, error and accuracy can be improved.

- **Observation**

Dataset	Training RMSE	Validation RMSE
Finance	0.1394	0.1589
SI	0.0037	0.0010



C. Classification Problem using RBF

According to the Cover theorem, a non-linear classification problem is more likely to be separable in a higher dimensional space than in lower dimensional space. Using this theorem as the basis, it can be said RBF can be used for different classification problems as it transfers the problem to a non-linear dimensional by using the gaussian function. Also, due to using Gaussian function as the activation function, RBF gives better posterior probabilities for the input data.

RBF as classifier:

Similar to MLP, RBF as classifier can be implemented with the output layer having neurons equivalent to the different class labels present in the dataset. RBF assigns that class label corresponding to the output neuron with maximum value.

RBF can perform well for classification problems where a lot of training data has missing class label values because it implements unsupervised clustering.

- **Effect of changing number of centres**

As we increase the number of centres, the performance of the model increases as it is able to reproduce the data behaviour due to more locally tuned exponential functions.

- **Effect of changing number of epochs**

On increasing the number of epochs, the accuracy increases initially. But later it leads to overtraining and decrease in the accuracy of the testing data.

GRAPH visualizing overfitting:

It can be seen in the graph below that the training error ***decreases*** with increase in the number of epoch which is represented by the blue line. Also,with increase in the number of epochs, the *validation error* initially almost ***decreases*** until 600 epochs and then starts ***increasing*** due to *overfitting* on the training set.

- **Effect of different Loss functions:**

Modified Least Square error Vs Least Square Error

Theoretically, the MLS loss function can capture accurately the target when output is predicted with more than 100% confidence. The error should not be difference between the actual and the predicted output in this scenario but should be zero. We observed that practically, MLS gives better results on classification datasets than LS loss function.

- **Observation**

Data Set	Overall Accuracy	Average Accuracy
ACTREC3D_27	27	30
BERK	42.42	42.42

4. Comparison between MLP and RBF for Approximation problem

	SI dataset		Finance dataset	
Model	Training RMSE	Validation RMSE	Training RMSE	Validation RMSE
MLP	0.0581	0.0581	142.7238	141.8229
RBF	0.0037	0.0010	0.1394	0.1589

- From the above table we can see that RBF approximation gives better result as compared to MLP approximation.

5. Comparison between MLP and RBF for Classification problem

	BERK		ACTREC3D	
Model	Overall accuracy	Average Accuracy	Overall Accuracy	Average Accuracy
MLP	100	100	100	100
RBF	42.42	42.42	27	27

- In contrast to approximation problem, we got better results for MLP classification than RBF classification.

6. Evolving Neural Networks

The distinguishing features of sequential learning algorithms are:

1) all the training observations are *sequentially* (one-by-one) presented to the learning system rather than giving them all together before starting the algorithm;
2) at any time, *only one* training observation is seen and learned;
3) a training observation is *discarded* as soon as the learning procedure for that particular observation is completed;
4) the learning system has no *prior* knowledge as to how many total training observations will be presented.

Resource Allocation Network(RAN) uses sequential learning algorithm. A new hidden neuron is added if the input sample has some "novelty".
The novelty conditions for adding a new neuron being:
(i) if the current sample is far away from existing centres(hidden neurons)
(ii) the error between predicted & actual output is higher than a threshold.

If novelty conditions are not met, current sample is used to update the network parameters by the learning algorithm (Gradient descent using Least Mean Square function).
LMS based RAN oscillates, hence Extended Kalman Filter based RAN is used which gives better performance with less complexity.

One of the major drawbacks of RAN is lack of pruning. RAN and EKF-RAN only add neurons, but not discard irrelevant neurons. This may lead to a very large network size for complex real life applications.

MRAN (Minimal Resource Allocation Network) gives better performance as the algorithm employs pruning of non-performing neurons.

- The *growth strategy* for MRAN is same as in RAN.
- The *pruning strategy* is that, if the output of a hidden neuron is less than a threshold for M consecutive input samples, one can infer that the particular hidden unit is not required as it does not contribute in learning from incoming new samples. Such a neuron is removed from the network and learning resumes from the next input sample.

In MRAN, all parameters including hidden neuron's weights, widths and centers have to be updated in every step. This leads to size of matrices to be updated huge and the complexity of network increases. Also, computation time is more in this model. Hence, Extended MRAN

(**EMRAN**) evolved in which the parameters of only 'winner' neuron (neuron that is close to current sample - close in norm sense) are updated. The computation time of EMRAN is effectively less than MRAN.

Parameters in MRAN code:

par= [p0 , q0 , r0 , emin1 , emin2 , espmax , espmin , r , kp , nw , sw , del , emran , prune]

Threshold for adding neuron, threshold for deleting neurons, overlap factor (kp) , size of sliding window (sw) for adding and removing neurons, estimate of parameter for uncertainties, etc.

As seen from graphs, as the threshold for adding neurons is increased, the number of neurons in the network at the end of training all samples decreases.
This threshold should be chosen appropriately. If the threshold is kept very low, a neuron will be added in spite of not containing much new information. If the threshold is kept very high, sample containing new information will also not add a new neuron, leading to lack of accuracy.

Pruning factor also has significant impact on network size. Pruning factor is set to zero, implies pruning of non-performing network is not allowed and pruning factor set to one implies pruning is taken into consideration.

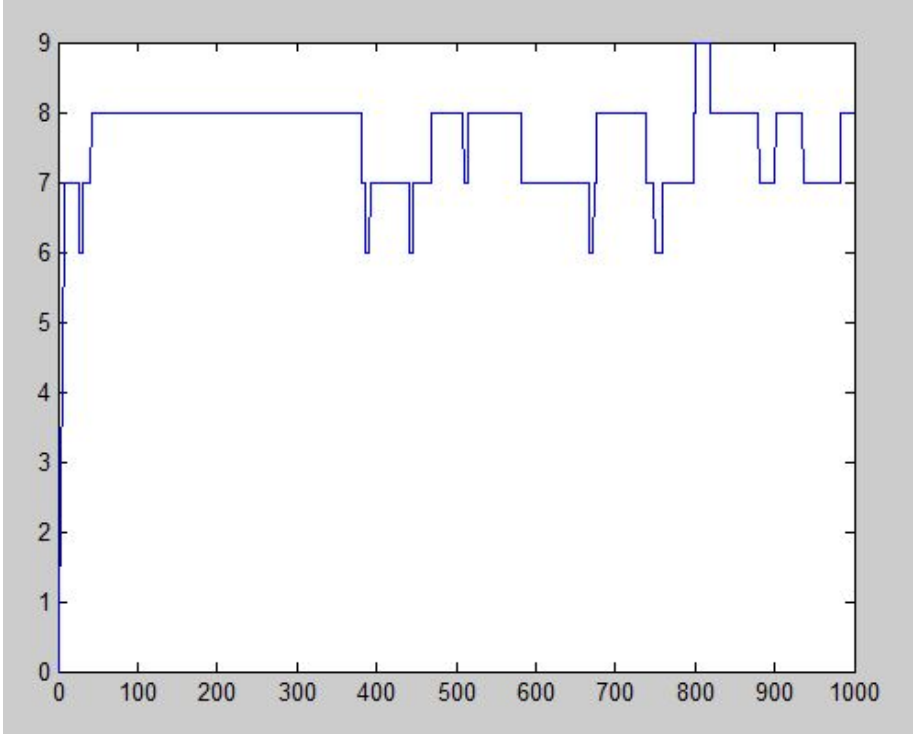
Graphs of mRAN:

X-axis: Number of samples
Y-axis: Number of hidden neurons

1. Fin data:

Ideal number of neurons = 8

Network Algorithm Parameters
par = [1.01 1.01 0.02 0.5 0.3 0.9 0.01 0.9 0.8 50 5 0.25 0 1];

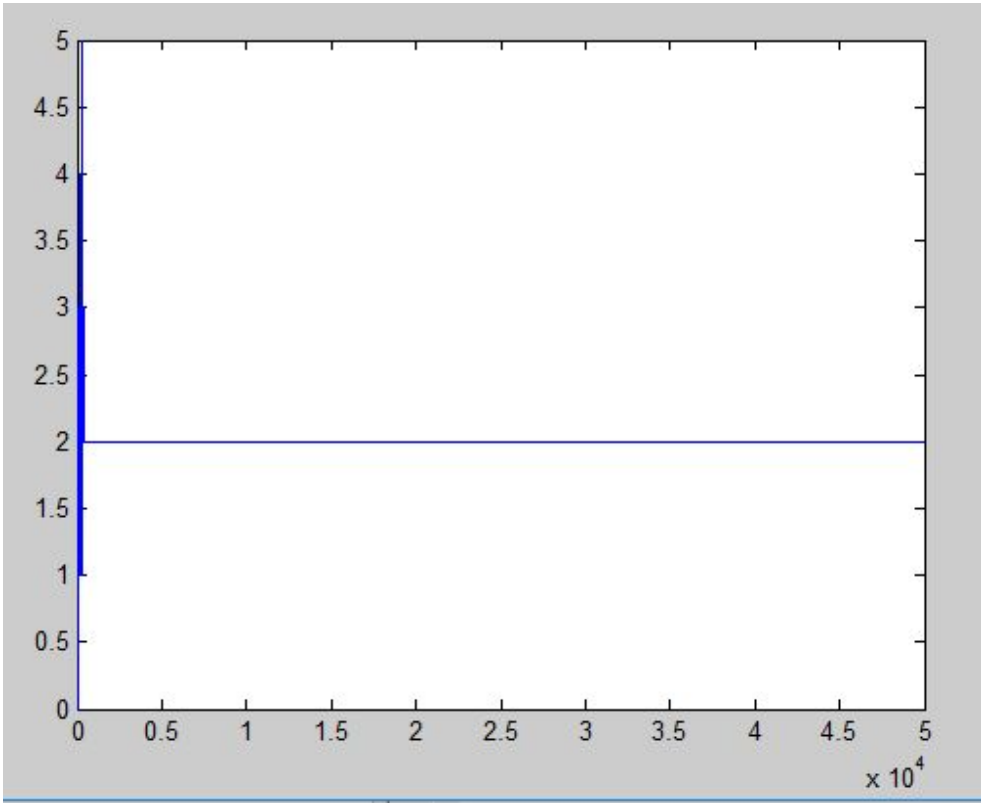


2. SI data set

Ideal number of neurons = 2

Network Algorithm Parameters:

par = [1.01 1.01 0.02 0.3 0.1 0.9 0.01 0.99 0.5 40 5 0.25 0 1]



APPENDIX:

a) MLP_MLS_actrec

```
Training error
   4.6243e-06


ng =

   100

Warning: Could not start Excel server for export.
XLSWRITE will attempt to write file in CSV format.
> In xlswrite (line 181)
  In MLP_Classifier_MLS_epo (line 149)
Overall Accurracy
   100

Average Accuracy
   100
```

Confusion Matrix:

83	0	0	0	0	0	0	0
0	93	0	0	0	0	0	0
0	0	34	0	0	0	0	0
0	0	0	12	0	0	0	0
0	0	0	0	25	0	0	0
0	0	0	0	0	101	0	0
0	0	0	0	0	0	70	0
0	0	0	0	0	0	0	70

b.MLP_LS_actrec

```
ng =

   100

Geometric Accuracy
   100

Warning: Could not start Excel server for export.
XLSWRITE will attempt to write file in CSV format.
> In xlswrite (line 181)
  In MLP_classifier_LS_epo (line 149)
Overall Accurracy
   100

Average Accuracy
   100
```

78	0	0	0	0	0	0	0
0	93	0	0	0	0	0	0
0	0	31	0	0	0	0	0
0	0	0	19	0	0	0	0
0	0	0	0	28	0	0	0
0	0	0	0	0	109	0	0
0	0	0	0	0	0	62	0
0	0	0	0	0	0	2	66

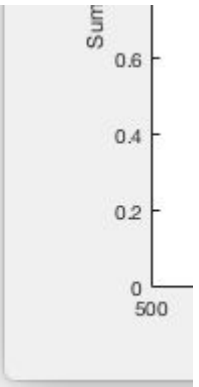
C. MLP_MLS_BERK

```
Training error
   1.9180e-09

Geometric Accuracy
   84.9687

Warning: Could not start Excel server for export.
XLSWRITE will attempt to write file in CSV format.
> In xlswrite (line 181)
  In MLP_Classifier_MLS_epo (line 152)
Overall Accurracy
   88.8889

Average Accuracy
   89.3939
```

1	1	0	0	0	0	0	0	0	0	0
0	3	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0
0	0	0	2	0	0	0	0	0	0	0
0	0	0	0	3	0	0	0	0	0	0
0	0	0	0	0	4	0	0	0	0	0
0	0	0	0	0	0	2	0	0	0	0
1	0	0	0	0	0	0	1	0	1	0
0	0	0	0	0	0	0	0	2	0	0
0	0	0	0	0	0	0	0	0	2	0
0	0	0	0	0	0	0	0	0	0	3

D. MLP_LS_BERK

```
Training error
    0.0259

Validation Error
    1.4636


ng =

   82.7753

Geometric Accuracy
   82.7753

Warning: Could not start Excel server for export.
XLSWRITE will attempt to write file in CSV format.
> In xlswrite (line 181)
  In MLP_classifier_LS_epo (line 150)
Overall Accurracy
   77.7778

Average Accuracy
   86.3636
```



1	0	0	0	0	0	0	0	0	0	0
2	3	1	0	0	0	0	0	0	0	0
0	0	2	0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	3	0	0	0	0	0
0	0	0	0	0	0	2	0	0	0	0
0	0	0	0	1	0	1	2	0	0	0
0	0	0	0	0	0	0	0	3	0	0
0	0	0	0	0	0	0	0	0	2	0
0	0	0	0	0	0	0	0	0	0	1