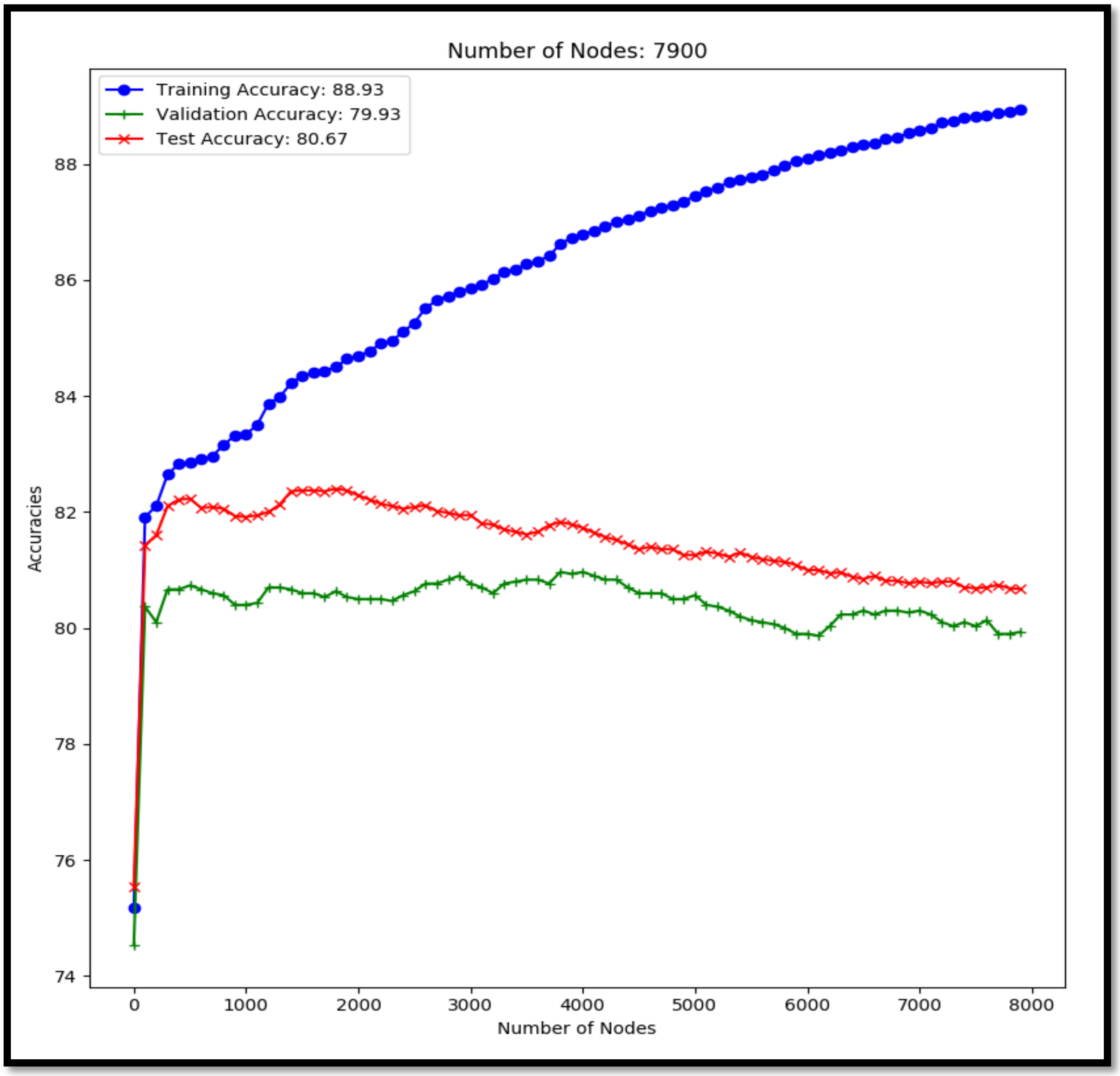
ML Assignment 3 Report

Harish Chandra Thuwal
2017MCS2074

Decision Trees and Random Forest

a) Decision Tree

- Each numerical attribute is converted to Boolean attribute before creating the tree.
- Best attribute selection criteria – Information Gain.



It is evident from the plot that as the number of nodes increase although the training accuracy continue to increase but the validation and test accuracy start to decrease. That is as we increase the number of nodes the model starts to overfit the training data and its performance decreases on test/validation data.

Another important point is the fact that with just a single node that predicts the majority class as ouput the accuracy obtained is around 75 percent.

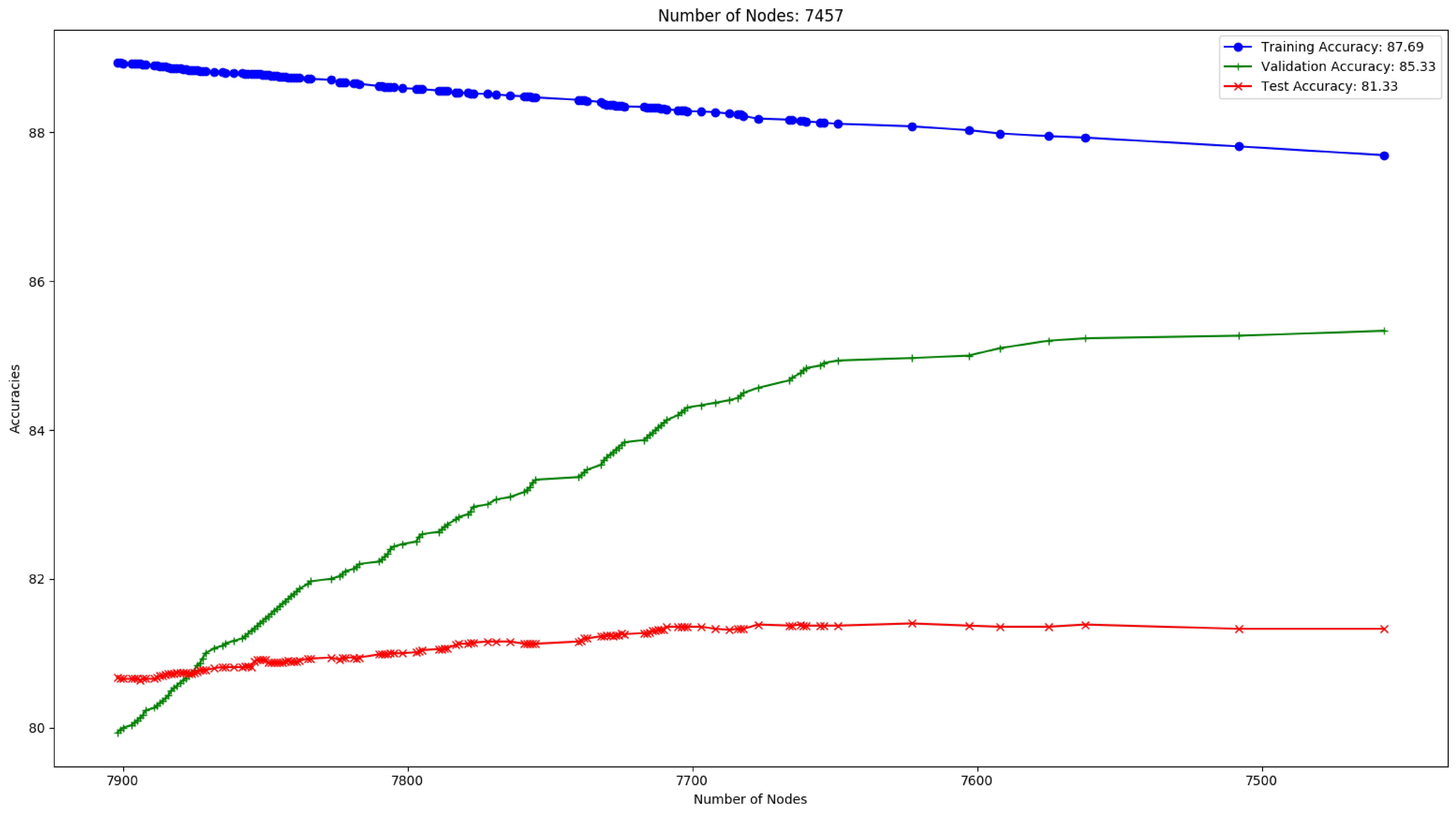
The final tree obtained had a height 11 and number of nodes 7902. The accuracies by this tree are:

Data Set	Accuracy
Training	88.93
Validation	79.93
Test	80.67

b) Post Pruning the decision tree

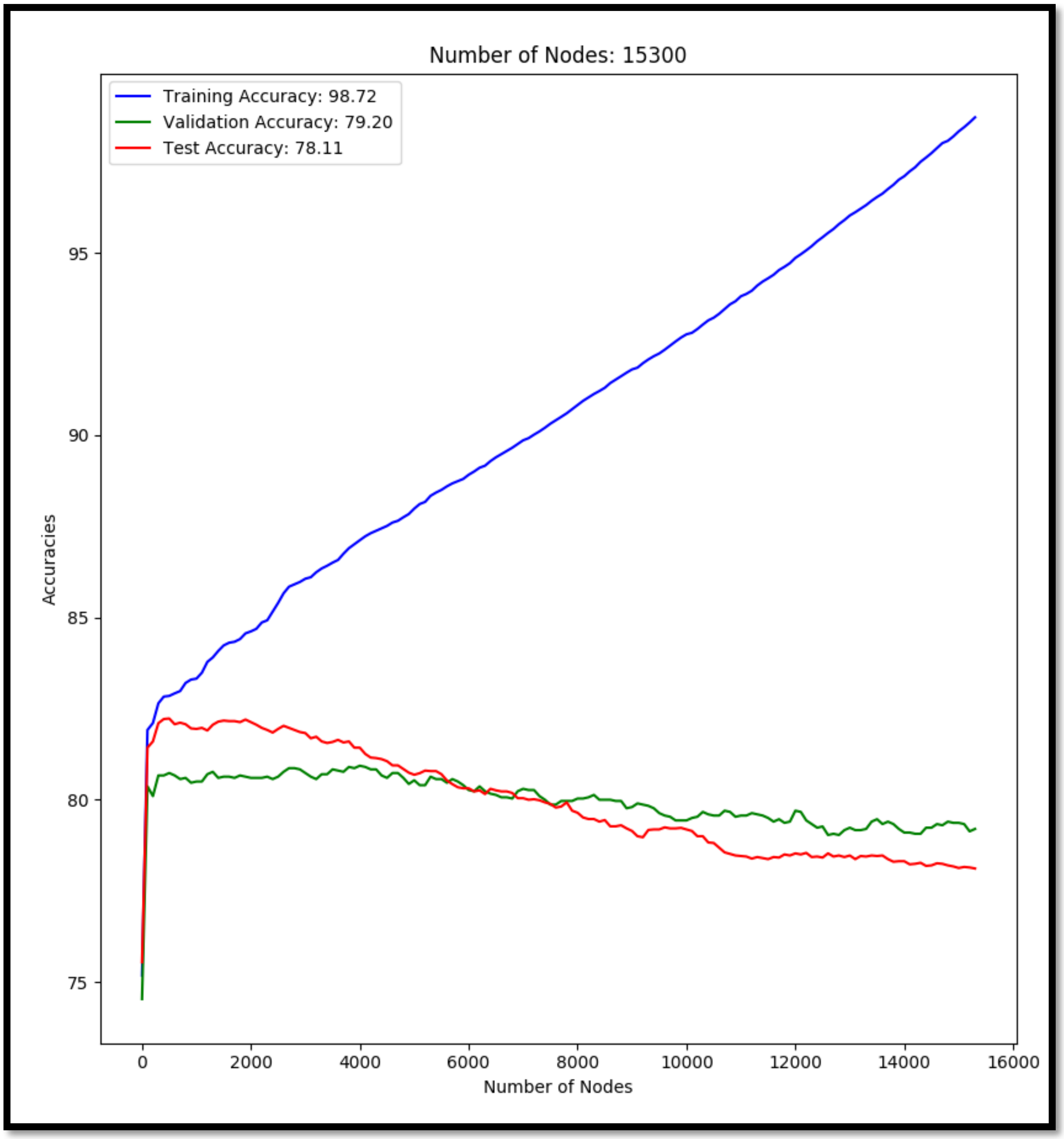
We start removing the nodes starting from the leafs in a bottom up manner. A node is removed if by removing this node (along with its entire subtree) the validation accuracy increases. At each stage when a node is removed the accuracy on the all three dataset is calculated and plotted as follows:

It is evident from the upcoming plot that as we remove nodes the validation accuracy increases from 79.93 % to 81.33% while the training accuracy decrease to 87.69%. The number of nodes decrease from 7902 to 7457.


Number of Nodes: 7457
Training Accuracy: 87.69
Validation Accuracy: 85.33
Test Accuracy: 81.33
Accuracies
88
86
84
82
80
7900
7800
7700
7600
7500
Number of Nodes

c) Dynamically calculate median (not beforehand)

In this setting the median is not calculated beforehand but rather calculated dynamically of only the data that is coming to a particular node.



It is clear from the above plot that how brilliantly this setting overfits the data!! The training accuracy reached 98.72% whereas the validation and test accuracy reduced to 79.20 and 78.11 respectively.

The decision tree created by this approach had 15365 nodes and a height 18.

Numerical Attributes (with their respective thresholds) that were split multiple number of times on a path are:

Numerical Attribute	Num	Thresholds
Age	8	39.0, 47.0, 51.0, 54.0, 57.0, 60.0, 65.0, 62.0
Fnlwgt	6	188965.5, 127768.0, 159244.0, 142914.0, 136824.0, 129172.0
Hpw	4	50.0, 57.5, 65.0, 70.0
Capg	2	7688.0, 20051.0
Canpl	1	1485.0

d) Scikit Learn Decision Tree

Using the scikit learn GridSearch tried different parameter settings for scikit learn's decision tree classifier. Numerical attributes were not converted to boolean values while giving the data to the decision tree classifier. Various Parameter tried are:

Attribute	Values
criterion	Gini, entropy
max_depth	None, 10, 20, 30
min_samples_split	2, 200, 400, 800, 1000
min_samples_leaf	1, 200, 400, 600, 800, 1000

GridSearch tries all possible combinations of the above parameter.

The best set of parameters obtained are:

criterion="entropy", max_depth=10, min_samples_leaf=1, min_samples_split=2

In the best parameter setting the accuracies obtained are:

Data Set	Accuracy
Training	86.459
Validation	84.966
Test	84.80

The accuracies obtained by the scikit learns decision tree are better than my implementation of the pruned decision tree but not by much. (2-3 %).

e) Random Forest.

A Similar grid search is performed on the following parameters to find the best parameter configuration for the random forest classifer:

While performing grid search combination of following attribute values were tried. Also, other attributes like *max_depth*, *min_samples_leaf* and *min_samples_split* were kept as obtained in the previous part.

Attribute	Values
bootstrap	True, False
max_features	2, 5, 8, 10, 12, 14, auto, log2, None
n_estimators	20, 25, 30, 40, 50, 60

The best set of parameters obtained are:

max_features=12, n_estimators=20, bootstrap=True

The accuracies in the best parameter setting are:

Data Set	Accuracy
Training	87.048
Validation	85.133
Test	85.485

Random forest of 20 estimators though increased the accuracy over test set as compared to the accuracy of a single tree, but the improvement was not very large (mere 0.6 %)

Neural Network

a) Description of the Created Neural Network Architecture

Class Layer:

- *Number of units in this layer*
- *Inputs to this layer*
- *Outputs of this layer*
- *Thetas: weights associated with each unit*
- *Gradients: gradients associated with each unit*

Class Neural Network:

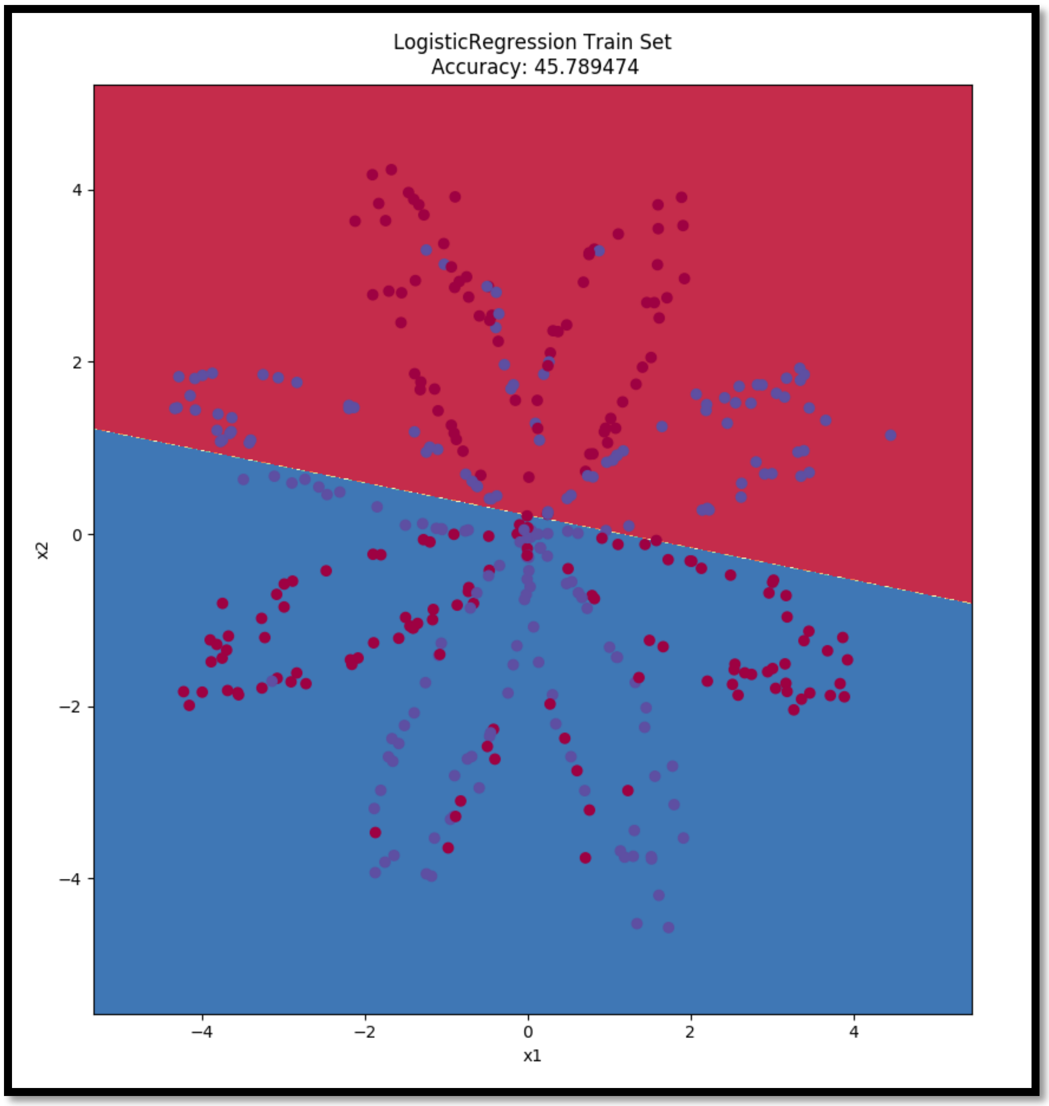
- *List of layers*
- *Function: forward pass: to update outputs of each layer*
- *Function: backward pass: to update gradients by back propagation*
- *Function: update_thetas: to update thetas of each unit in each layer*
- *Function: train – implemented SGD that calls above three functions in the above order. Unitill convergence*

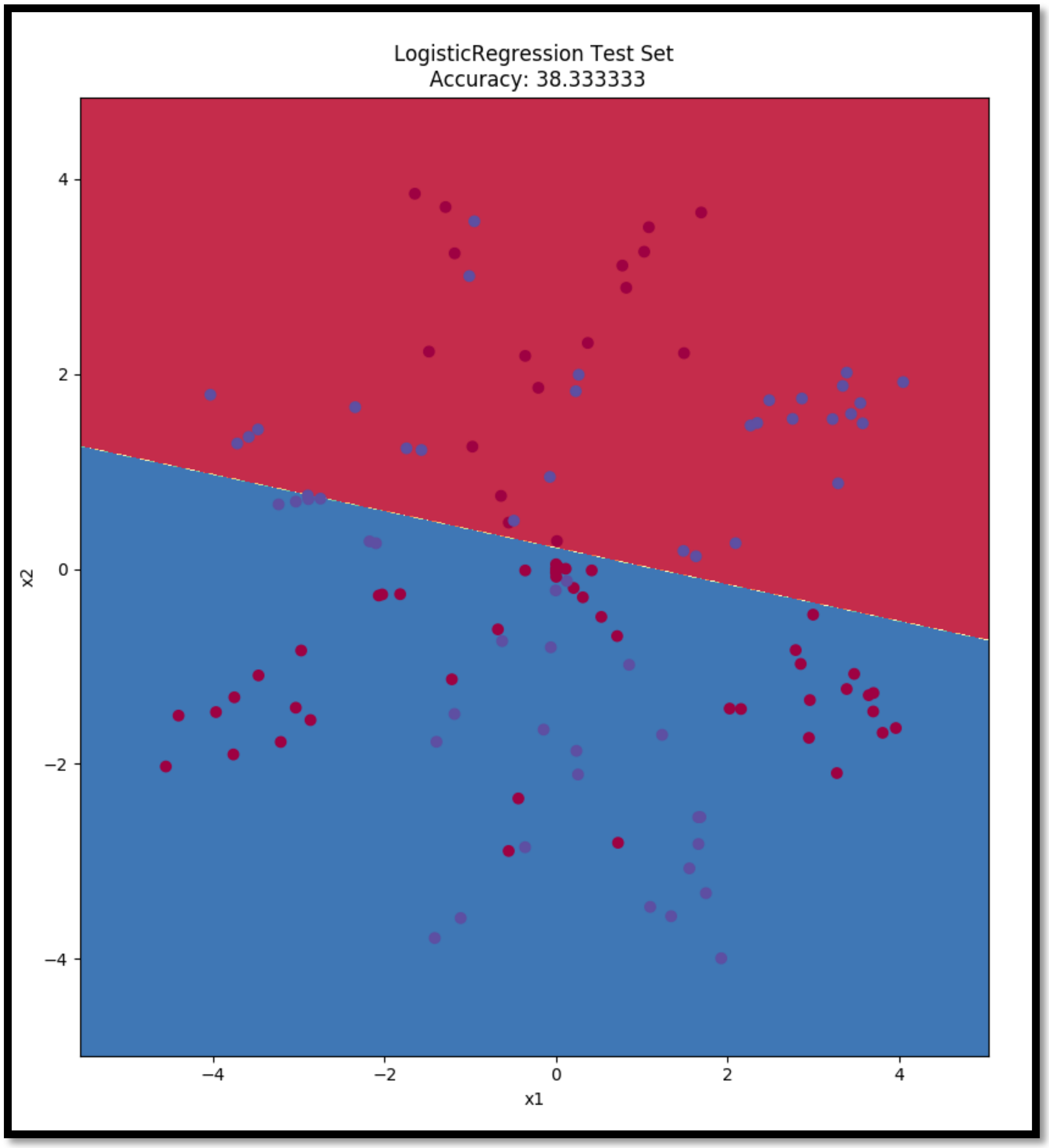
Stopping Criteria: max_epochs, change in training error < threshold
Bothe threshold and max_epochs are passed as parameter to the train function.

b) Visualizing Decision Boundary (Toy Data)

1) Logistic Regression

Data Set	Accuracy
Training	45.8
Test	38.33


LogisticRegression Train Set
Accuracy: 45.789474
x1
x2
−4
−2
0
2
4
4
2
0
−2
−4



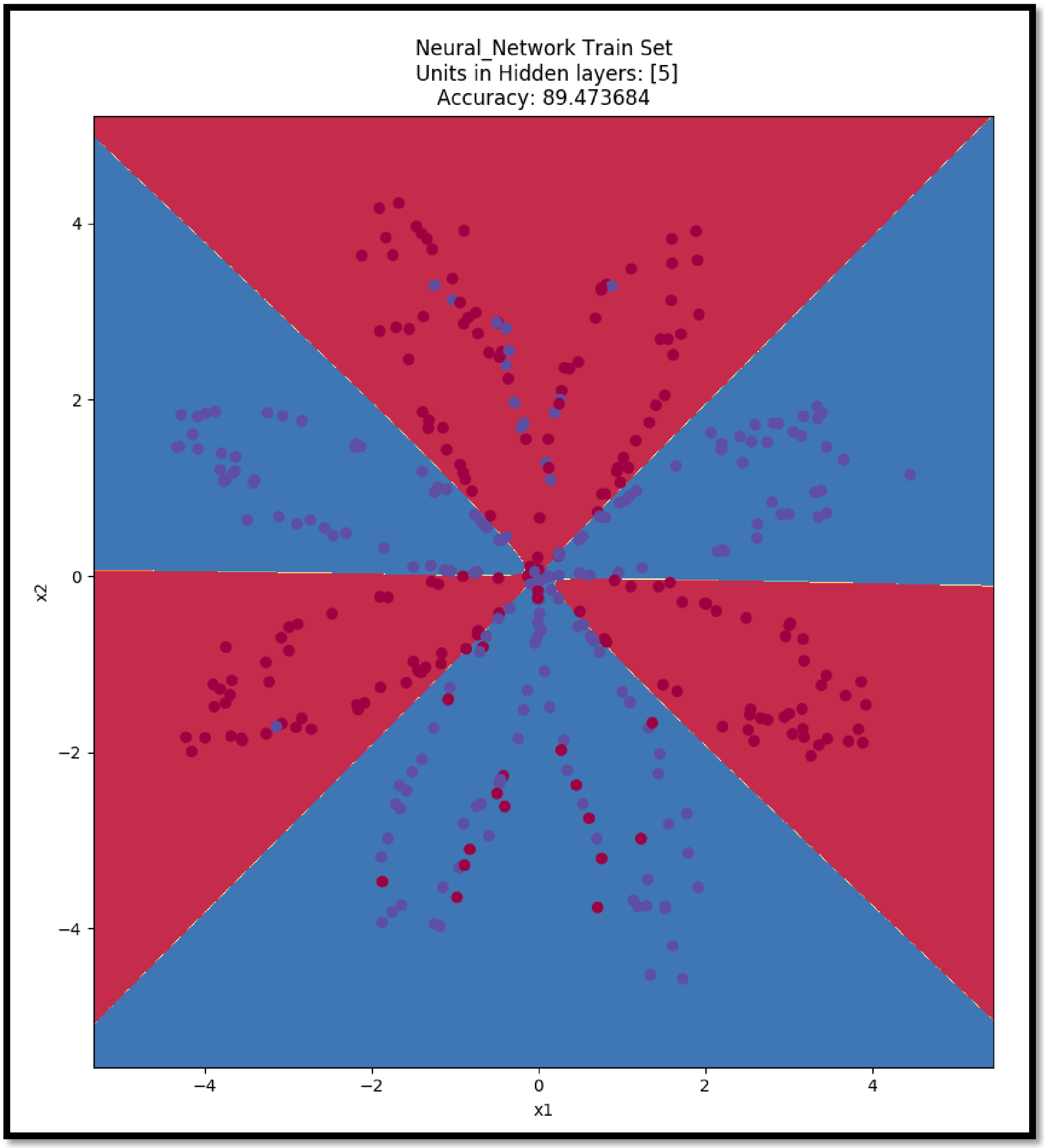
Since the logistic regression is a linear classifier and the data is inherently non-linear, it is unable to classify the data and the accuracy obtained is a low 38%.

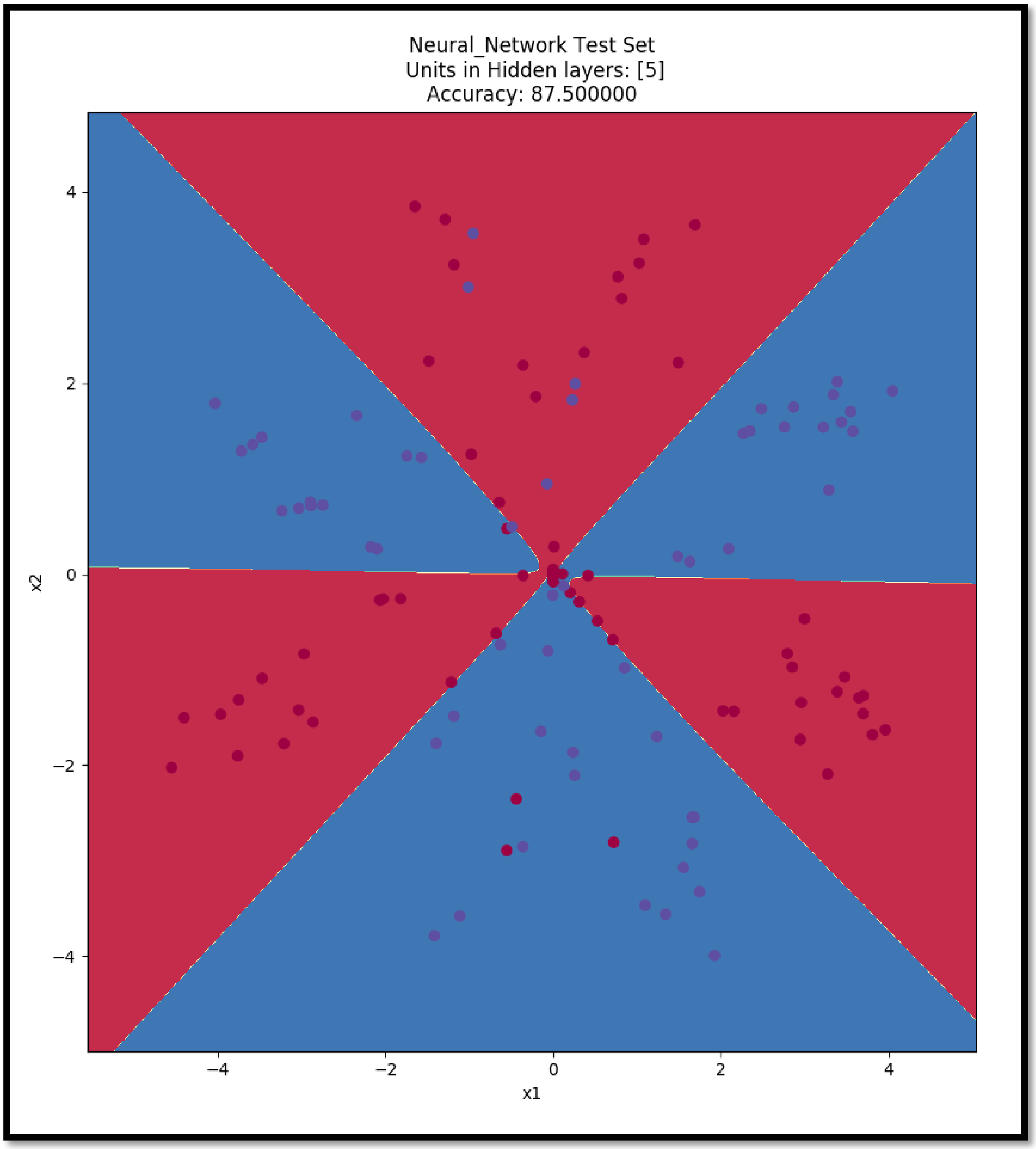
2) Single Hidden Layer with 5 neurons

Data Set	Accuracy
Training	89.473
Test	87.5

Stopping Criteria: max_epochs:5000, threshold: 1e-6

Converged in 4512 epochs





The neural network classifies the data way more accurately (87%) as compared to the logistic regression (38%).

This is because the data is inherently nonlinear and logistic regression is a linear classifier and hence could not classify the data correctly. Neural net on the other hand are nonlinear because of the nonlinear activation functions and hence learned the non-linear nature of the data quite brilliantly as evident from the decision boundaries.

3) Varying the number of units in the hidden layer

Stopping Criteria: max_epochs:10000, change in error < threshold: 1e-6

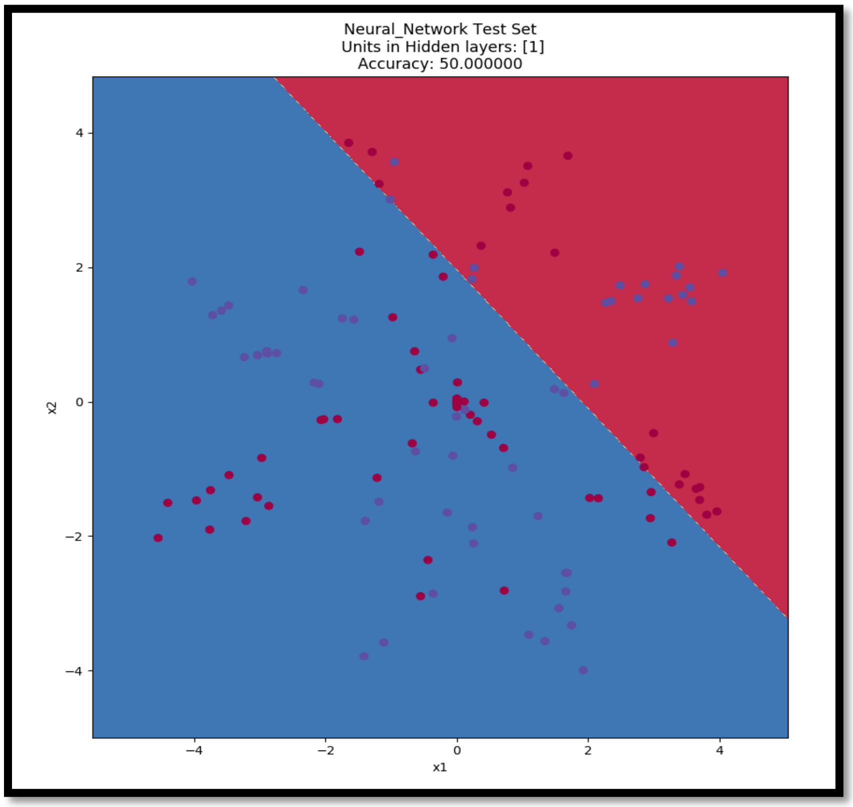
Num Units	Training Accuracy	Testing Accuracy
1	54.47368	50.0000
2	58.15789	55.8333
3	75.26315	76.6667
5	89.47300	87.5000
10	90.26315	85.8333
20	90.78947	84.1667
40	89.47368	83.3333

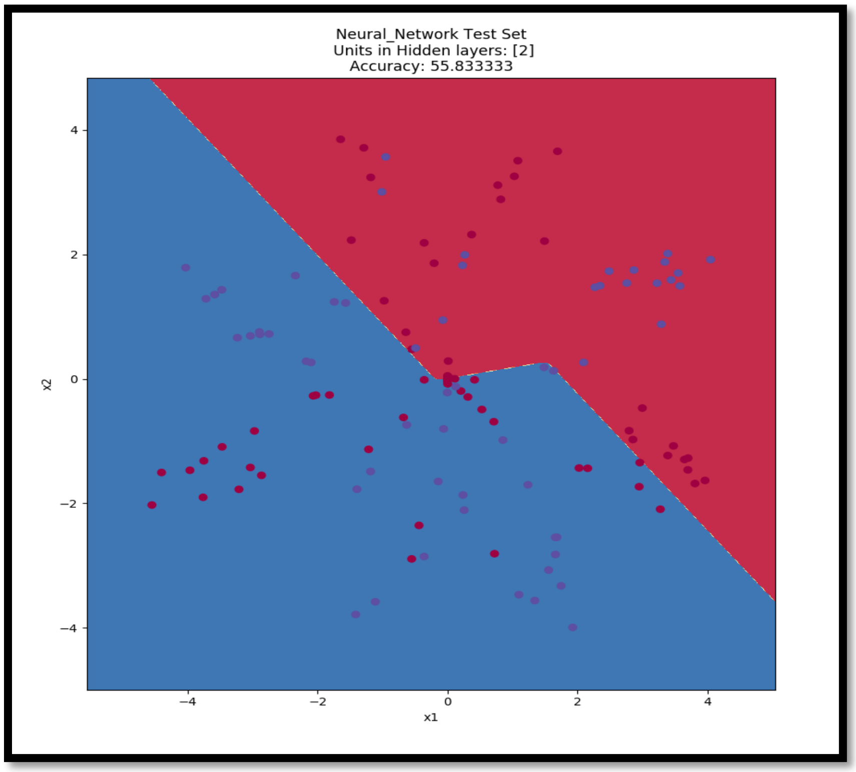
Initially while increasing the number of neurons from 1 to 5 both the training and testing accuracies increase. But on further increasing the number of neurons. The training accuracy increases but the testing accuracy start to decrease. This is indicative of the fact that as the number of neurons increase beyond a certain threshold the model starts to learn some extra non-important features or patterns from the training data which do not generalize to the unknow test data. Or in other words the model starts to overfit the the training data.

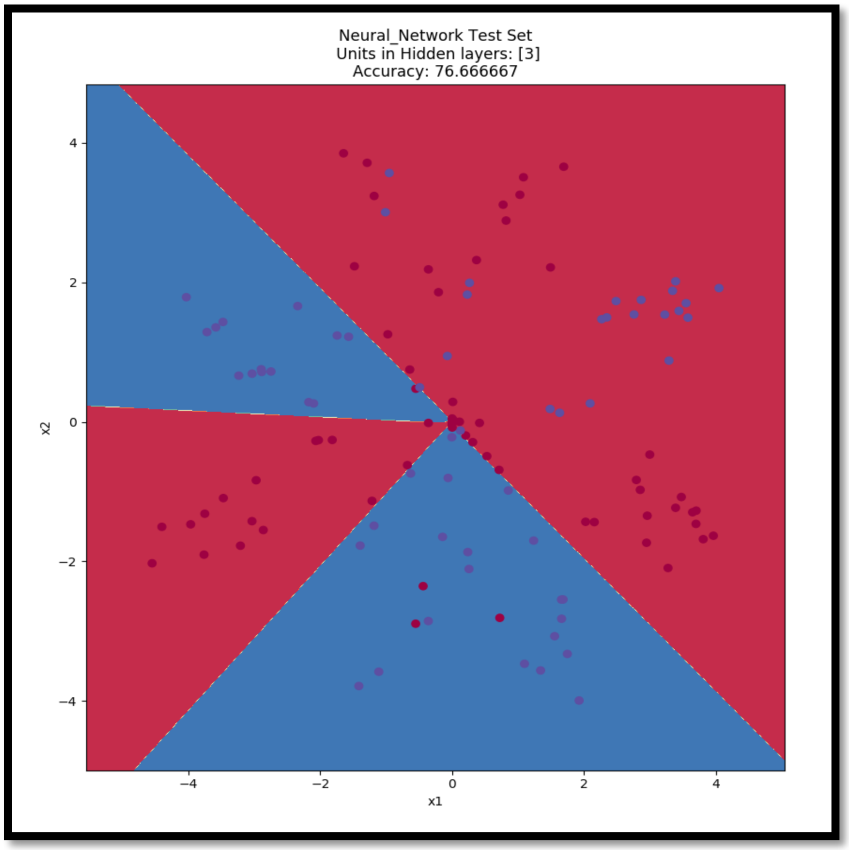
Based on the above observation optimum number of neurons seems to be 5 as it has the highest accuracy on the test set.

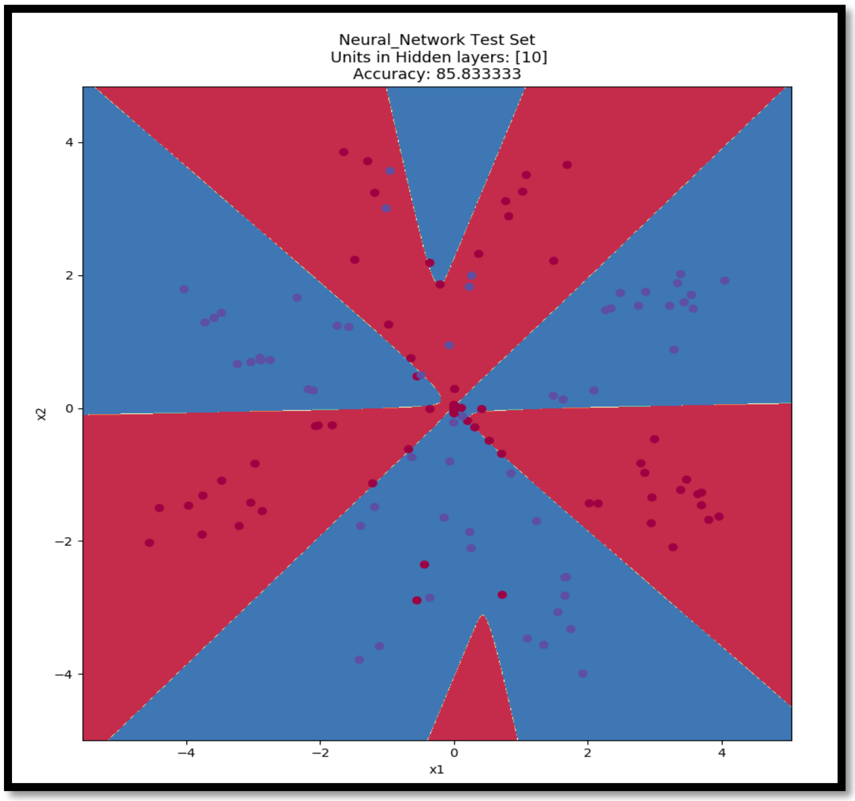
The decision boundary for 5 neurons is already shown for. Upcoming are the decision boundaries and the test data plots for the others.

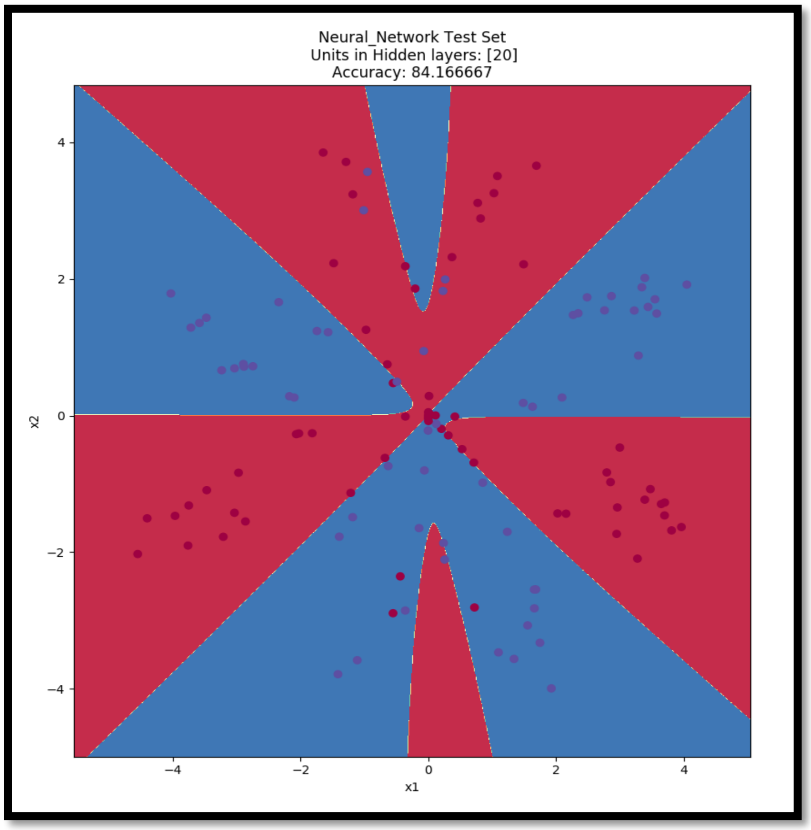
As we increase the number of neurons the decision boundary tends to divide the domain into multiple regions. In case of 1 neuron there is a single line separating the domain into two regions. Then with increase in number of neurons the number of regions that decision boundary creates increases to 4, 6 and 8 in case of 3, 5 and 10 neurons respectively. 6 seems to be the optimum number of regions. Further increase in number of domains causes overfitting.

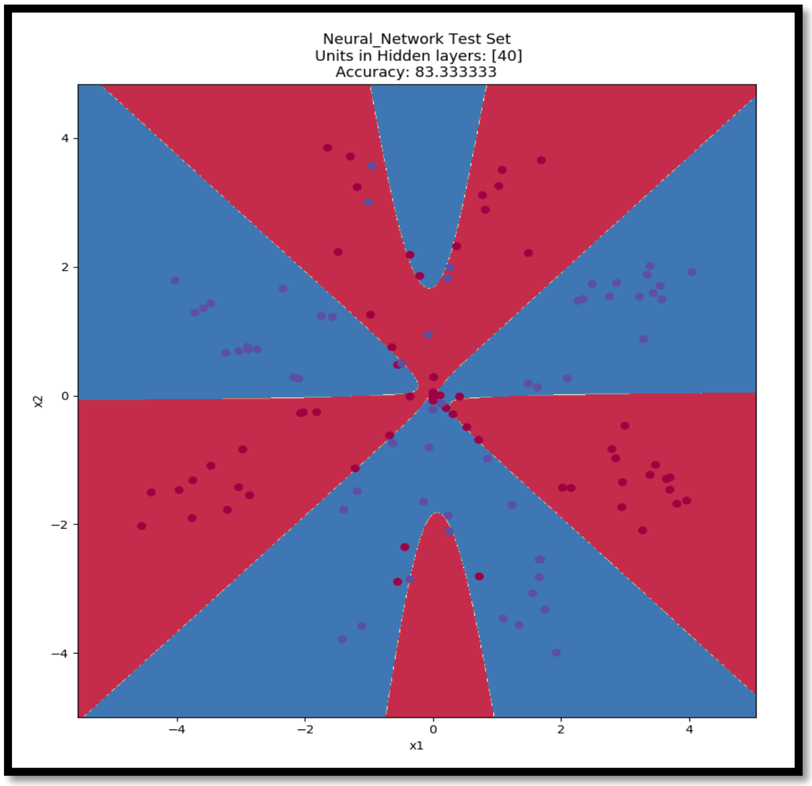

Neural_Network Test Set
Units in Hidden layers: [1]
Accuracy: 50.000000
x1
x2
−4
−2
0
2
4


Neural_Network Test Set
Units in Hidden layers: [2]
Accuracy: 55.833333
x1
x2
−4
−2
0
2
4


Neural_Network Test Set
Units in Hidden layers: [3]
Accuracy: 76.666667
x1
x2
−4
−2
0
2
4


Neural_Network Test Set
Units in Hidden layers: [10]
Accuracy: 85.833333
x1
x2
−4
−2
0
2
4


Neural_Network Test Set
Units in Hidden layers: [20]
Accuracy: 84.166667
4
2
0
−2
−4
x2
−4
−2
0
2
4
x1


Neural_Network Test Set
Units in Hidden layers: [40]
Accuracy: 83.333333
4
2
0
−2
−4
x2
−4
−2
0
2
4
x1

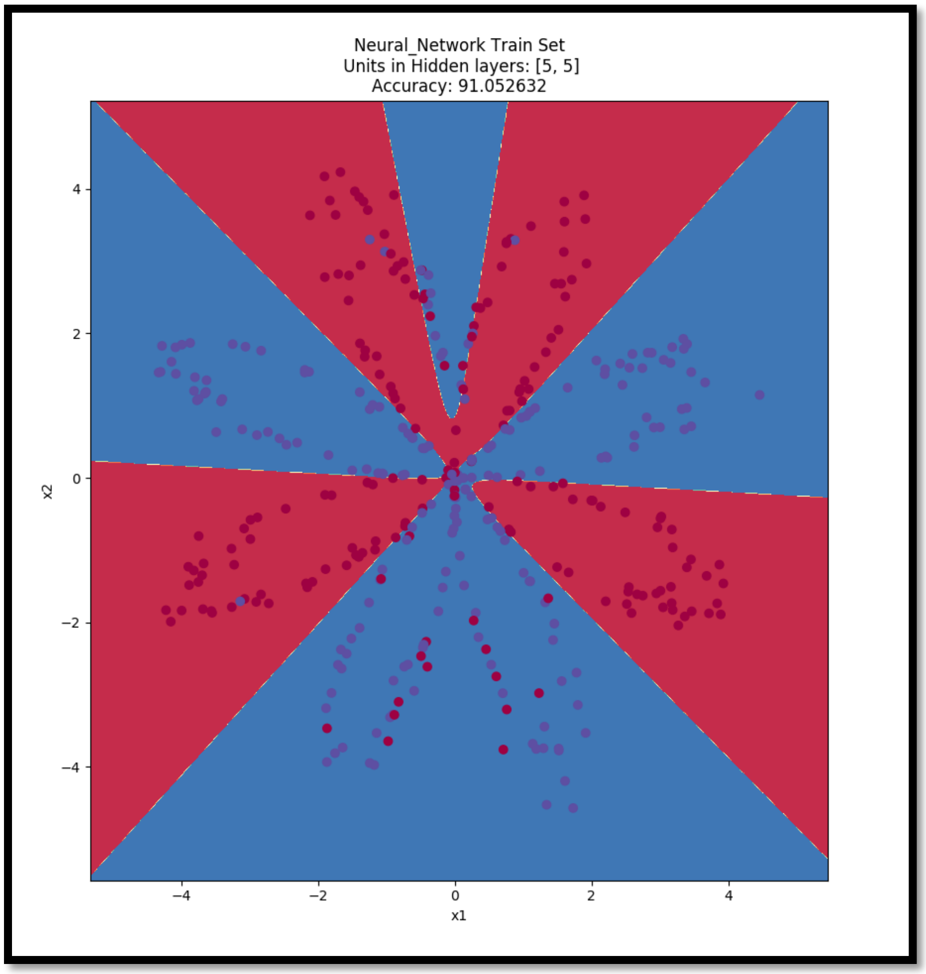
4) Network with two hidden layers of 5 – 5 units each

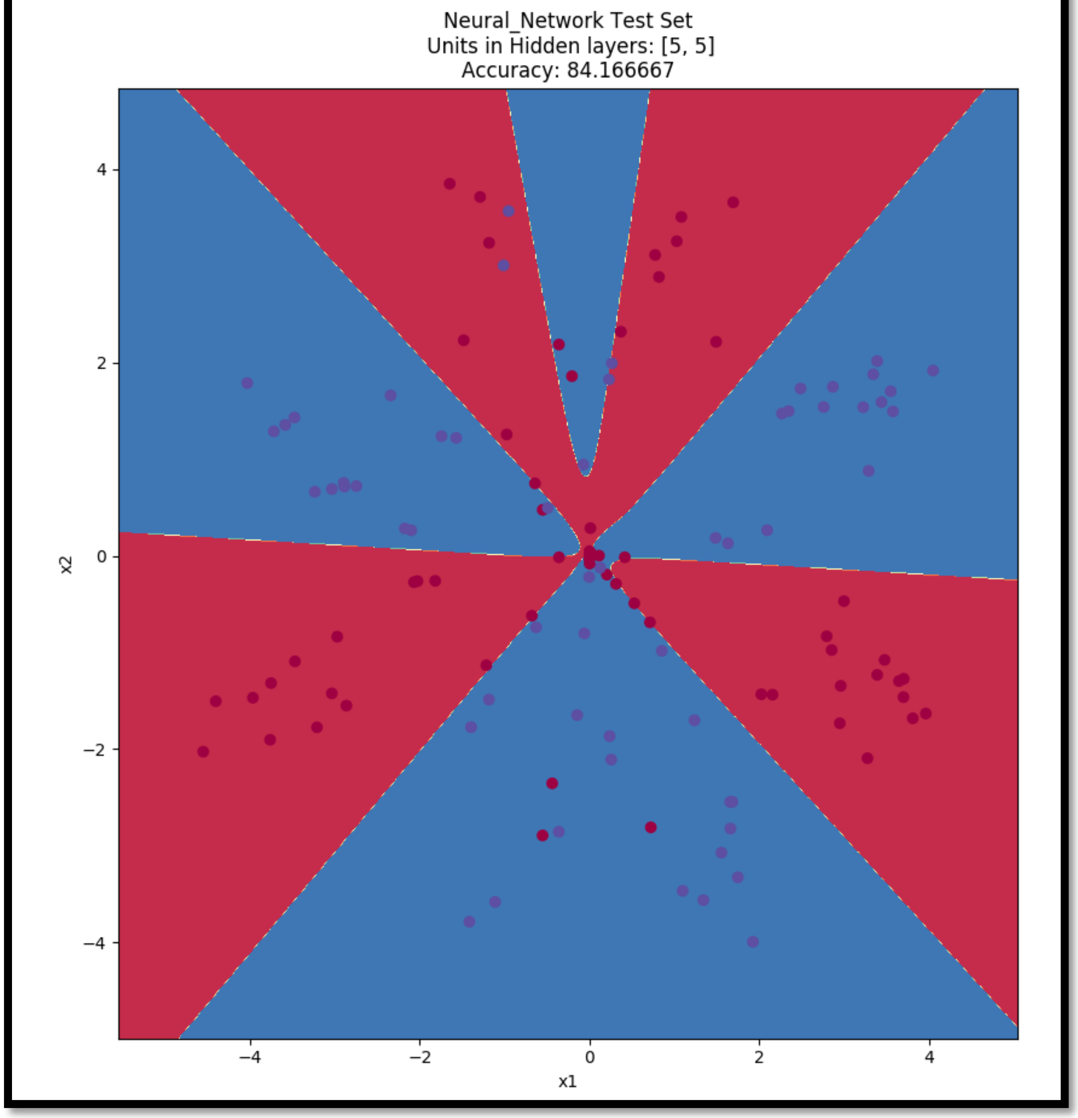
Learning Rate: 0.1
change in error < threshold = 1e-10

Data Set	Accuracy
Training	91.0526%
Test	84.1667%

Accuracy obtained by this model is similar to that obtained by a single layer with 20 units. Certainly better than everyon else above except 5 and 10 units models.

Decision boundaries obtained for train and test data are as follows:




Neural_Network Test Set
Units in Hidden layers: [5, 5]
Accuracy: 84.166667
x1
x2
−4
−2
0
2
4
−4
−2
0
2
4

c) MNIST Handwritten Digits Recognition – 6 & 8

1) libsvm v/s a single perceptron

- SVM Linear Kernel, C = 1

Data Set	Accuracy
Training	100% (10000/10000)
Test	98.4722% (3545/3600)

Since linear kernel is able to give such high accuracy on the test set. This shows the data is inherently linear and thus a single perceptron should be able to give good results on the same set.

- Single Perceptron: Batch Size = 100,
  Learning Rate = 0.001 * (1 / sqrt(iteration))
  max_epochs = 250
  change in error < threshold = 1e-3

Data Set	Accuracy
Training	99.92%
Test	98.91%

A single perceptron outperforms the linear SVM by 0.5%. on Test Data

2) 100 units in hidden layer

Batch Size = 100,
Learning Rate = 0.1 * (1 / sqrt(iteration))
max_epochs = 250
change in error < threshold = 1e-3

Data Set	Accuracy
Training	99.82%
Test	99.1389%

The accuracy obtained after 250 epochs by this version was slightly better as compared to the single perceptron (0.229 %). However, since this model had a whole lot of parameter to be learned (gradients to be calculated), each epoch took more time than it took for the single perceptron case.

3) ReLU instead of sigmoid in hidden layer.

Batch Size = 100,
Learning Rate = 0.01 * (1 / sqrt(iteration))
max_epochs = 250
change in error < threshold = 1e-3

Data Set	Accuracy
Training	99.82%
Test	99.027%

There wasn't any significant difference in the final results if we use ReLU. However, it converged in 112 epochs. Each epoch took less time and network tend to converge quickly to lower errors as compared to sigmoid. This may be because of the fact that it is simpler and faster to compute and does not saturate. Also, its gradient is always 1 for anything greater than 0 which reduces the likelihood of gradient to vanish.