ML ASSIGNMENT 3

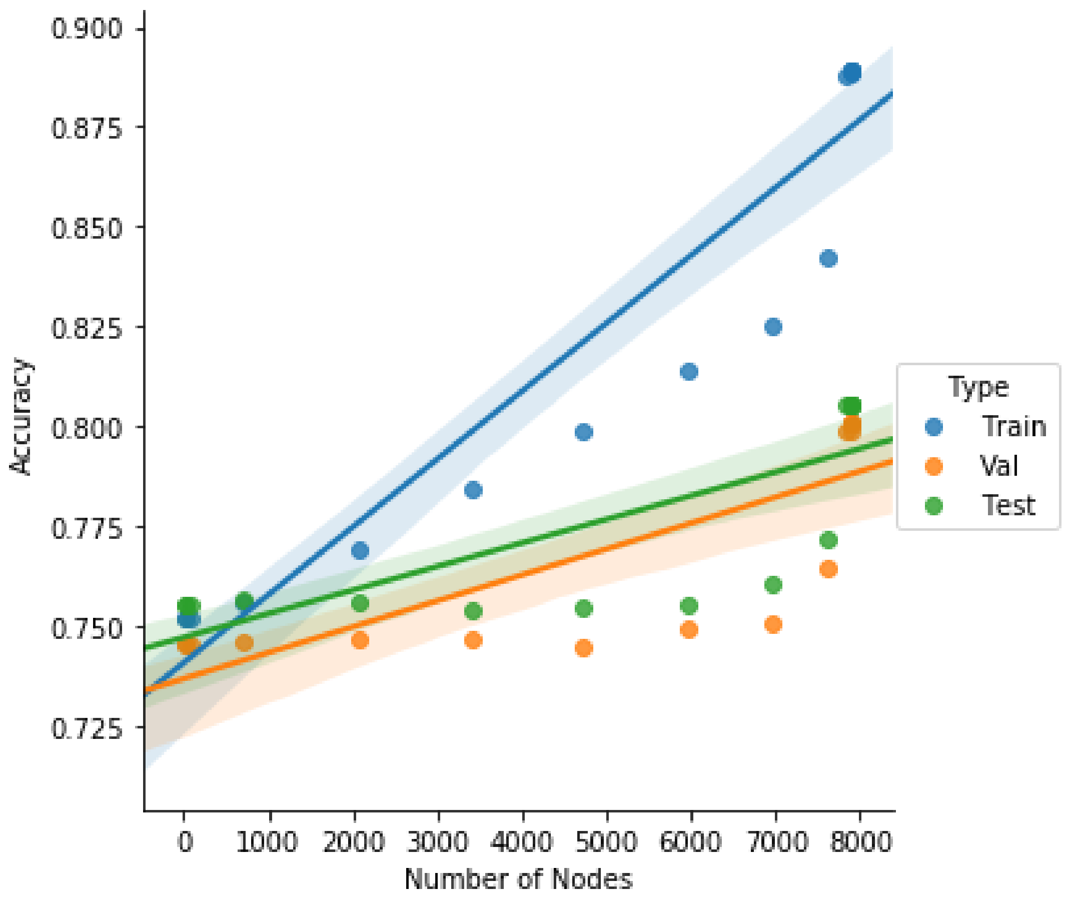
MADHUR SINGHAL

2015CS10235

PART 1 : DECISION TREE

SUBPART A)

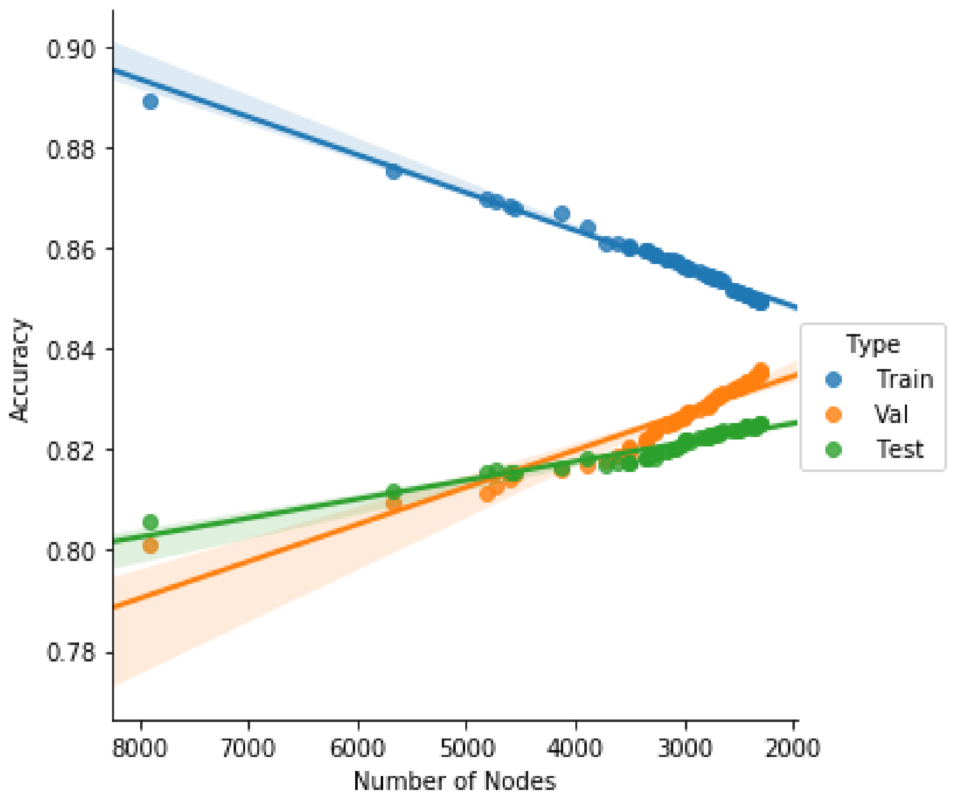
In this part we implement the descision tree as per the given specifications. Nodes were split whenever information gain was positive. Some other tricks were required for special cases. Following is a plot of accuracies with number of nodes in a tree grown level by level.



It's clear from the graph that all accuracies increase with number of nodes added. The train accuracy is higher than the other two which indicates some slight overfitting to the train data.

SUBPART B)

I implemented pruning as specified. Following is the plot obtained.

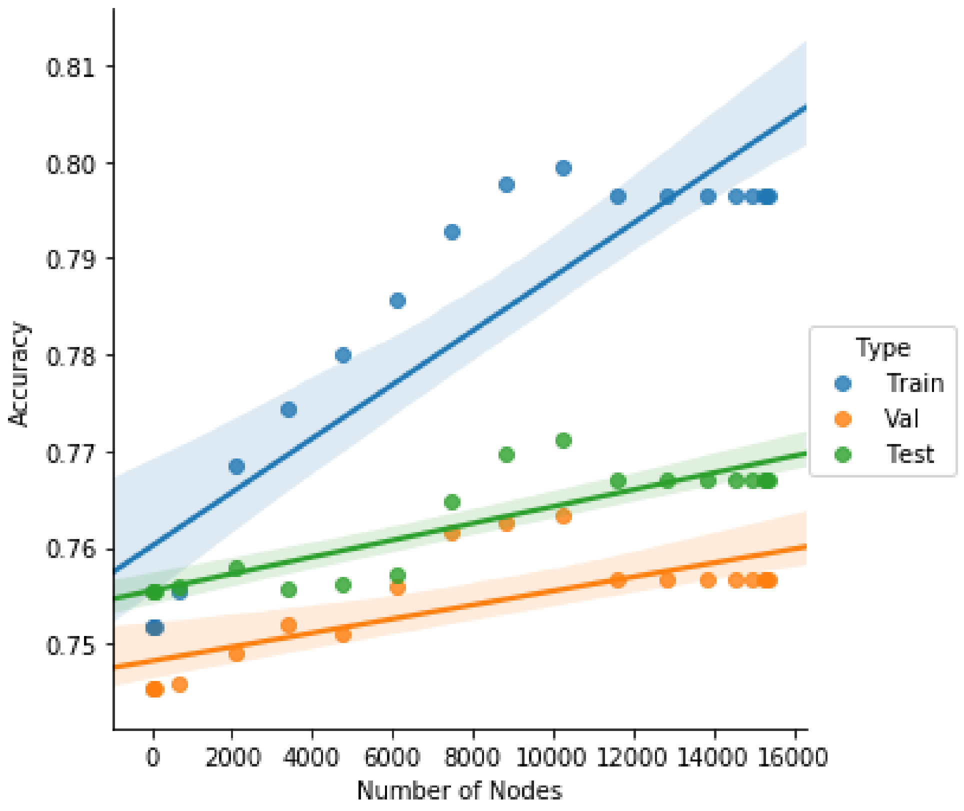


It can be easily seen that pruning improves the validation and test accuracies while reducing the training avccuracy as it should. We go from a 7898 node tree with about 80% test accuracy to a 2308 node tree with 83.5 % validation accuracy and 82.5% test accuracy.

SUBPART C)

```
[(0, [62.0, 65.0, 60.0, 57.0, 54.0, 51.0, 47.0, 39.0], 8),
 (2, [129172.0, 136824.0, 142914.0, 159244.0, 127768.0, 188965.5], 6),
 (10, [20051.0, 7688.0], 2),
 (12, [70.0, 65.0, 57.5, 50.0], 4)]
```

The above denote the numerical feature indices where multiple splitting occured and the medians that were calculated for the split thresholds. They are “age”, “fnlwgt”,”capg” and “hpw”. The below is the plot obtained.



This is worse than the first part in all three accuracies. This indicates that splitting based on the dynamic median yields worse results presumably because in some lower level node when information gain is being calculated, the dynamic median causes us to vastly overstate the information gain for a numerical attribute by splitting even numerically close output values to different classes.

SUBPART D)

I perform a grid search over the parameter values and find the following to be the optimal values.

```
{'max_depth': 10, 'min_samples_leaf': 10, 'min_samples_split': 7}
```

Train Accuracy – 86.4%

Val Accuracy – 85.2%

Test Accuracy – 84.7%

In all of these, both too low and too high values give lower accuracies than the shown ones. Having a minimum number for splitting anddesignating as leaf is seen to be a good thing in general.

SUBPART E)

For the random Forest the following parameter values were found to be optimal.

```
{'max_depth': 30,
 'max_features': 9,
 'min_samples_leaf': 6,
 'min_samples_split': 5,
 'n_estimators': 100}
```

Train Accuracy – 91.1%

Val Accuracy – 85.5%

Test Accuracy – 85.4%

Bootstrapping is seen to increase accuracy. Keeping max features to a moderate value is the best as too low results in very bad trees and two high leads to less diversity among the trees. Number of estimators should be kept at a high value.

The accuracies obtained are about 2 percentage points higher than the ones we got in pruning experiment and 6 percentage points higher than in dynamic median case. This is mostly because of the min_splits and min_leaf parameters and partly because of having multiple estimators.

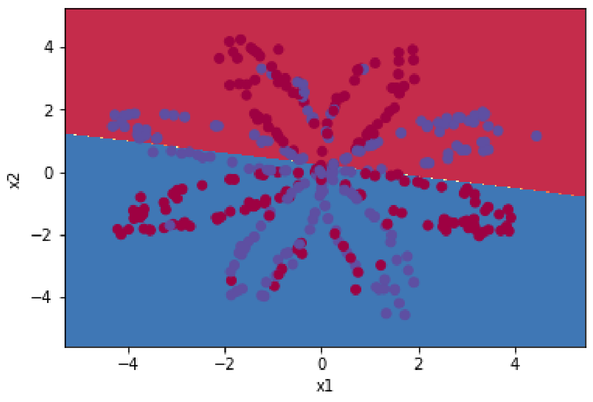
PART 2 : NEURAL NETWORK CLASSIFICATION

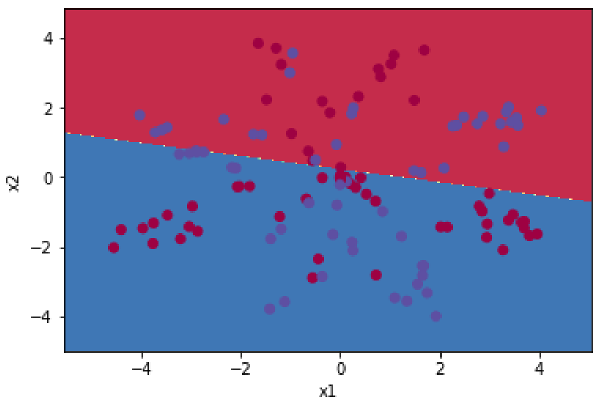
SUBPART A) IMPLEMENTATION

We implemented a neural network using standard techniques in numpy.

SUBPART B) TOY

Train Accuracy is 45.7% and test accuracy is 38.3%. Decision boundary with train and test points is hown below. Note that because of numerical problems our logistic is not able to converge to a better solution (even classisfying all as one class).

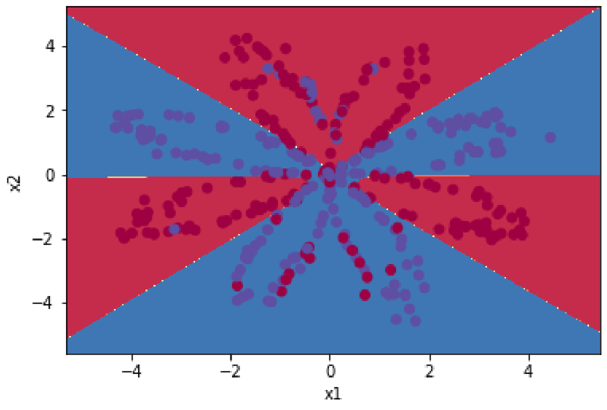


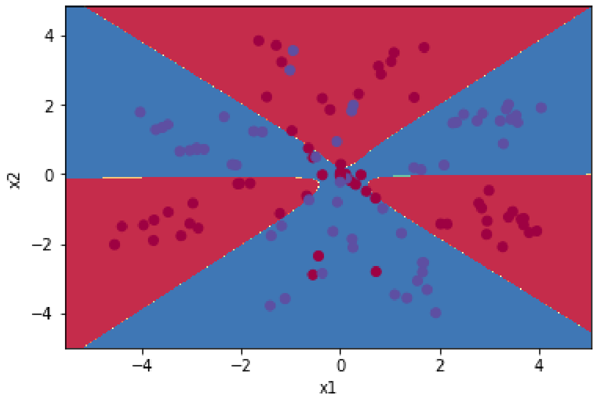


For a Nrural network with one hiddlen layer of size 5 the following are the accuracies and descision boundaries obained. (Run for 20000 epochs)

Train Accuracy – 87.36%

Test accuracy – 81.66%

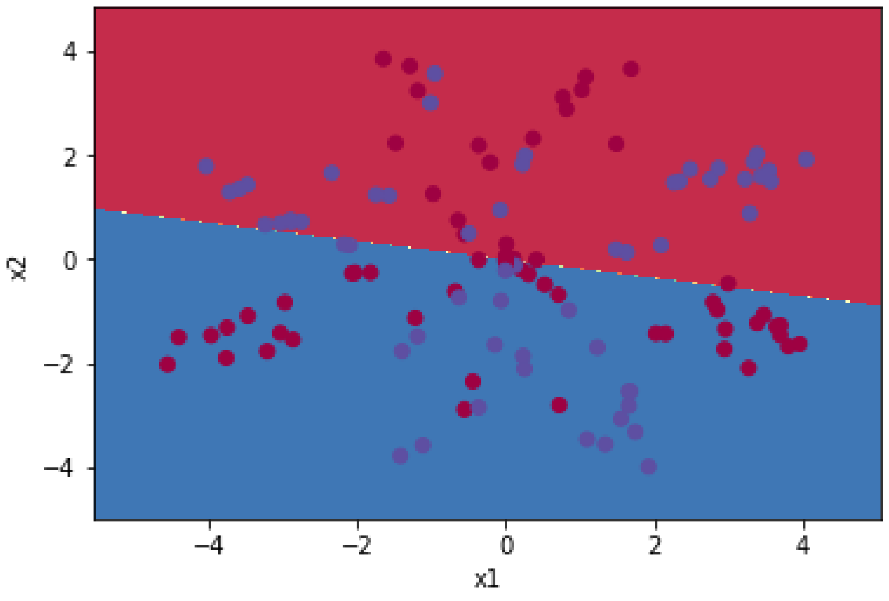




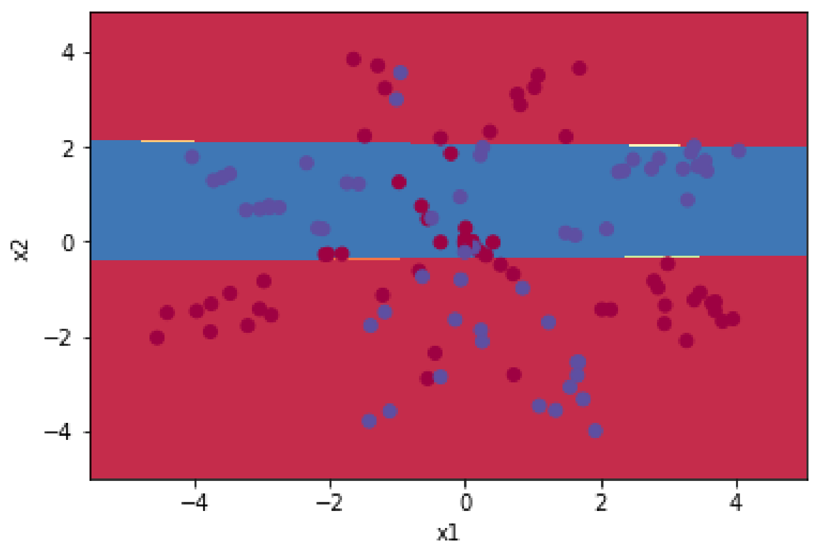
After this we change the number of hidden units and note the accuracies and the decision boundaries. The decision boundary is seen to split up into multiple parts on increasing the number of hidden units. This is expected since more hidden units means more combinations of features can be utilized by the network. The number of optimal units seems to be 20 since too large a number causes overfitting and too small does not have enough expressive power.

Num Hidden Units	Test Accuracy
1	39%
3	65%
10	82%
20	82%
40	80%

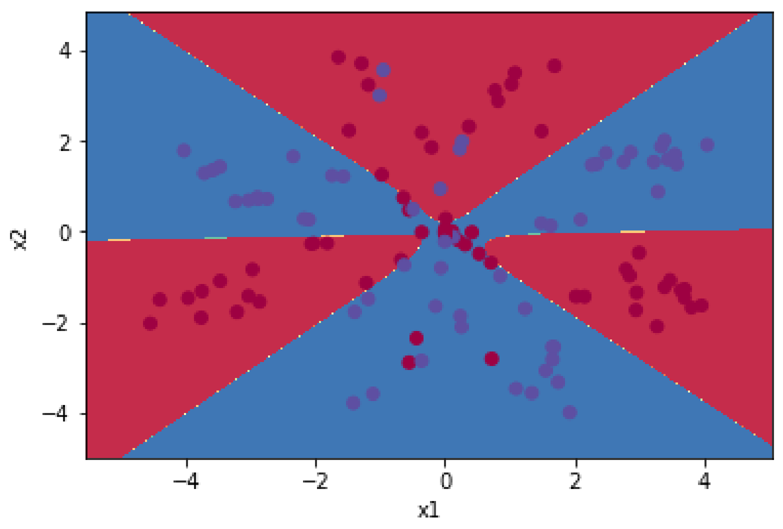
Hidden Units 1



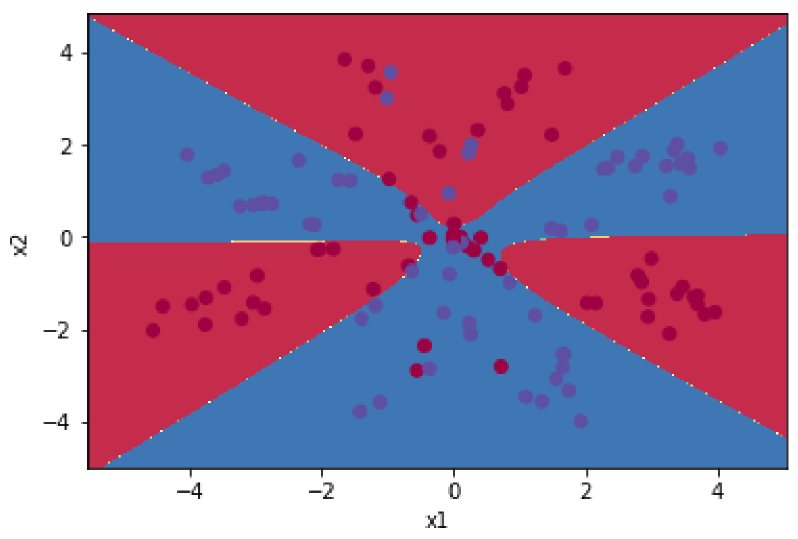
Hidden Units 2



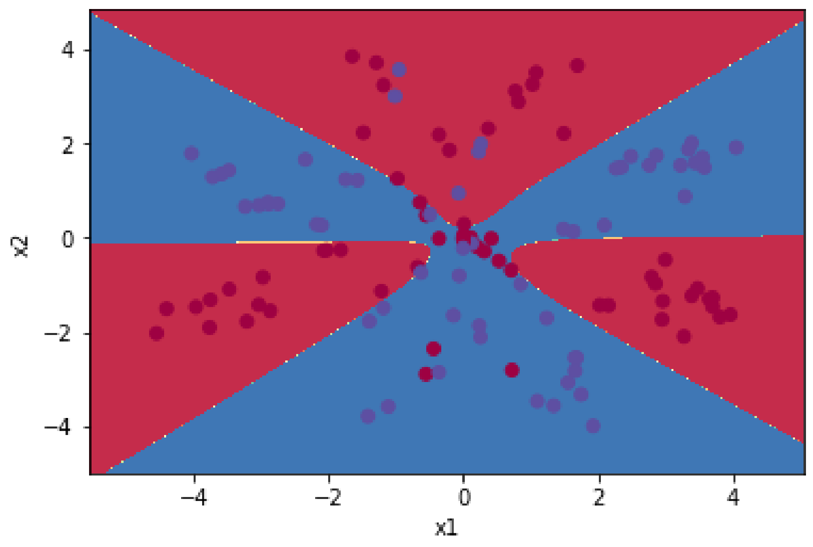
Hidden Units 3



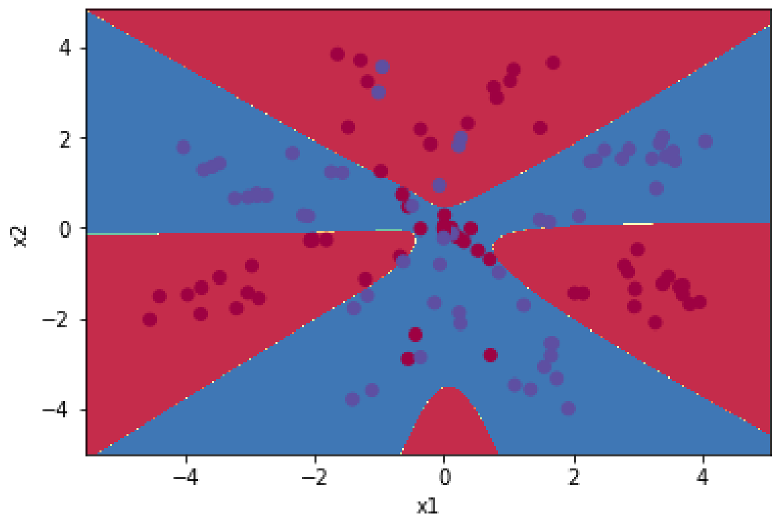
Hidden Units 10



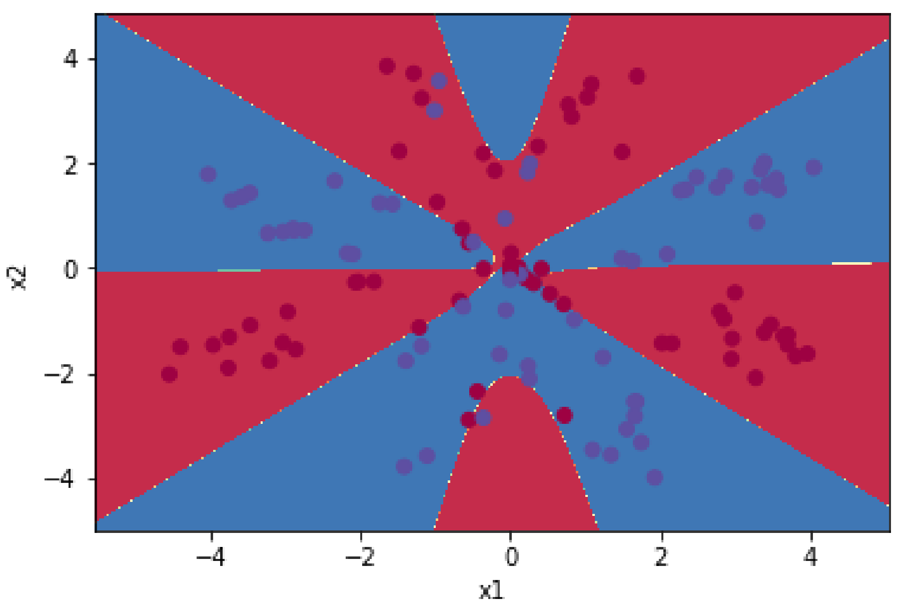
Hidden Units 20



Hidden Units 40



For the two hiddle layer case we use the two hidden layers of size 5 and 5. The test accuracy observed is 85% which is a bit higher than the previous cases (run for more epochs too). The decision boundary is shown below which also looks the same as the previous decision boundaries.



SUBPART C) MNIST

Firstly linear SVM accuracy was compared with a single preceptron.

SVM Train Accuracy - 100%	Perceptron Train Accuracy - 99.32%
SVM Test Accuracy - 98.5%	Perceptron Test Accuracy - 98.36%

After this we train a neural network with a single hidden layer of 100 nodes.

For this part our train accuracy was 99.8% and test accuracy was 98.7%. The test accuracy was higher than both of the above parts. This hints that a neural model is more expressive than a Linear SVM.

The stopping criterion chosen was that the total train accuracy should not increase over two consecutive epochs for us to stop learning which was seen to perform well in practice. Learning rate decay was also utilized.

After this RELU was uilitzed as activattions. That gave training accuracy of 99.9% and test accuracy off 98.9%. Thus there is an improvement. Despite this I observe that sometimes with RELU the Neural net does not converge at all. This is beacuse in RELU’s zero part derivative is zero and so a lot of neurons can die out and never produce any output because of bad random initialization.