Monday, 9 April 2018

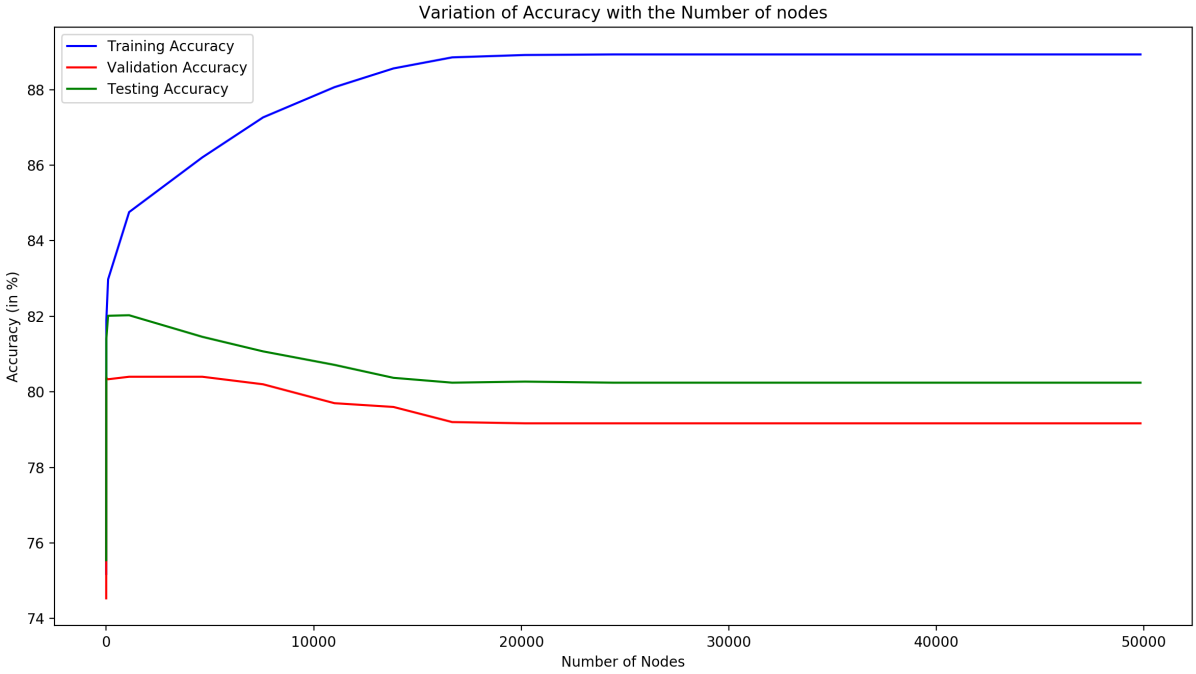
MACHINE LEARNING : ASSSIGNMENT 3

Decision Trees & Random Forests

DECISION TREES (PART 1)

1(a)

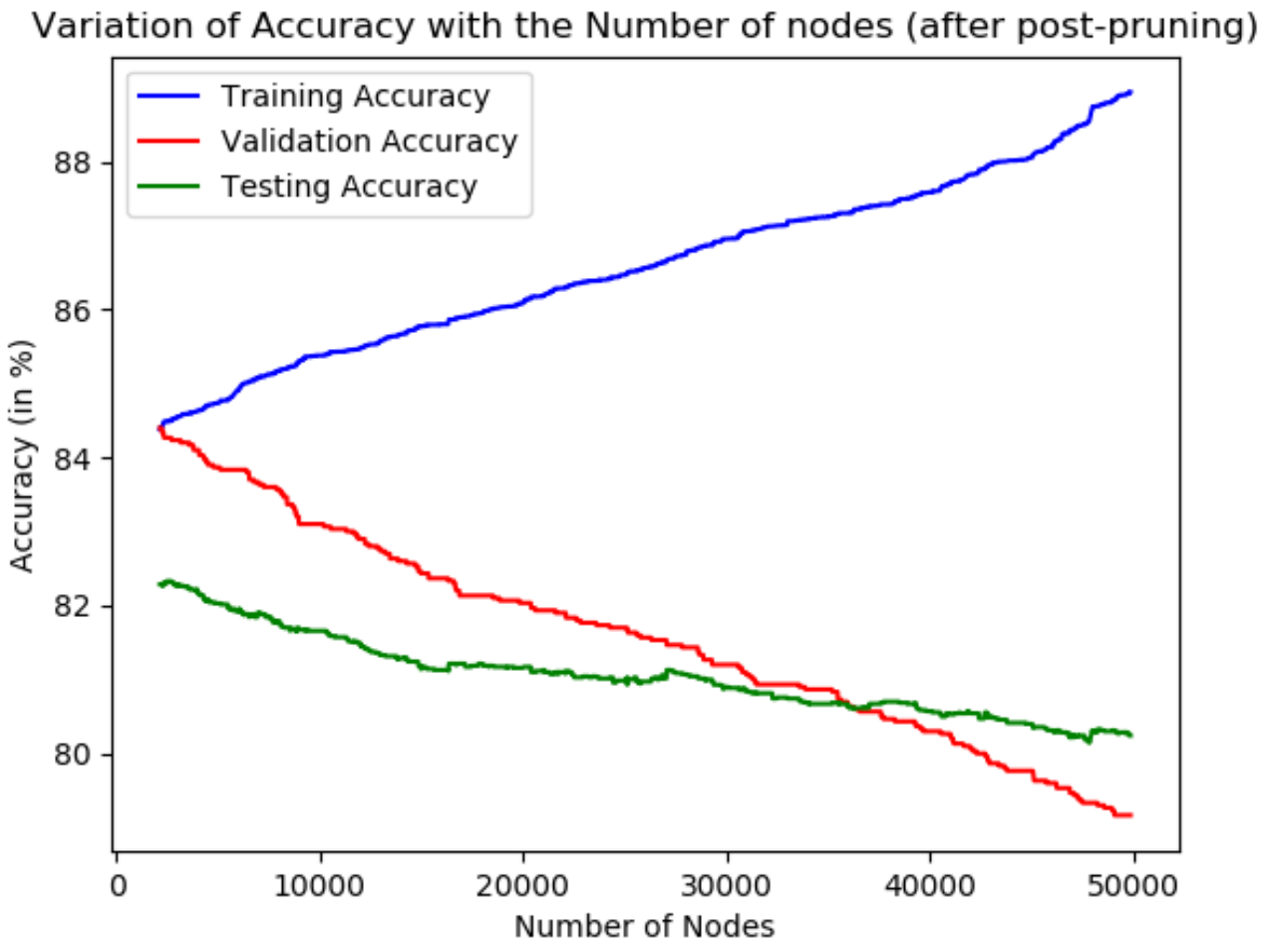
- In this preprocessing step of this part, the numerical attributes were binarised to the binary discrete values 0/1 using the *read_data.py* script available to us.
- The sizes of sets available to us are Train: 27000 , Validation: 3000 , Test: 7000.
- Results:
    - NUMBER OF NODES IN THE FULLY GROWN TREE : 49830
    - TRAINING SET ACCURACY : 88.93%
    - TESTING SET ACCURACY : 80.24%
    - VALIDATION SET ACCURACY : 79.17%
- Conclusions & Observations:
    - As the number of nodes in the tree increases, the tree begins to fit more and more onto the training examples leading to a condition called overfitting. As a result, we can see that the training accuracy increases with the number of nodes, and saturates towards the end in the decision tree.
    - Initially when the number of node increases, accuracy over the validation and testing set increases with it but as soon as the number of nodes increases beyond a certain limit, the tree begins to overfit on the training examples leading to a fall in the accuracy over the testing and validation set.



Monday, 9 April 2018

1(b)

- After preprocessing the tree, we did a post order traversal of the tree and started pruning the nodes in a bottom-up manner. The issue is after each pruning operation, we need to check the accuracy over the validation set which consists of 3000 examples making the pruning operation slightly costlier than it already was.
- So, to overcome this issue, after fully growing the tree, simply distribute your validation set examples label over each node in one traversal of the tree. Now, we simply need to update the label of this count as the node is pruned to get the accuracy in a much faster way. This method works because while pruning the node in the tree we only need to update the labels on a very small part of the tree while the majority of the labels would remain unaffected by this pruning operation. This speeds up the pruning operation in a very significant way.
- Results:
    - Depth of the decision tree = 14
    - NUMBER OF NODES IN THE TREE : 2163
    - TRAINING SET ACCURACY : 84.38%
    - TESTING SET ACCURACY : 82.29%
    - VALIDATION SET ACCURACY : 84.4%



- Observations:
    - Initially when the pruning procedure started, the tree had 49830 nodes. When the post order procedure to prune the nodes started, the training accuracy started decreasing and decreasing showing that the specific patterns in data learnt from the particular training of the model are getting eliminated, and more general patterns are still intact and well learnt in the tree.
    - A node has been pruned if the pruning operation leads to a non-positive decrease in the validation set accuracy.
    - As we are pruning the data, we observe that the accuracy over the validation set increases(which makes sense since we are pruning only when there is an increase in accuracy), and the accuracy over the test set increases as well which confirms our claim that the earlier fully grown tree wasn't able to generalise well to new examples, and this pruning has removed those specific traits from the model which were very specific to those examples. Hence, we finally had very less number of nodes (2163), and the best accuracy over the test set as per the post-pruning procedure.

1(c)

- Depth of the decision tree = 19
- Attributes were indexed from 1 to 14 (there are 14 attributes in all, out of which 6 are numerical attributes and can take a range of values, while the others are discrete).

Results:

- NUMBER OF NODES IN THE TREE : 23599
- TRAINING SET ACCURACY : 99.78%
- TESTING SET ACCURACY : 77.84%
- VALIDATION SET ACCURACY : 78.7%

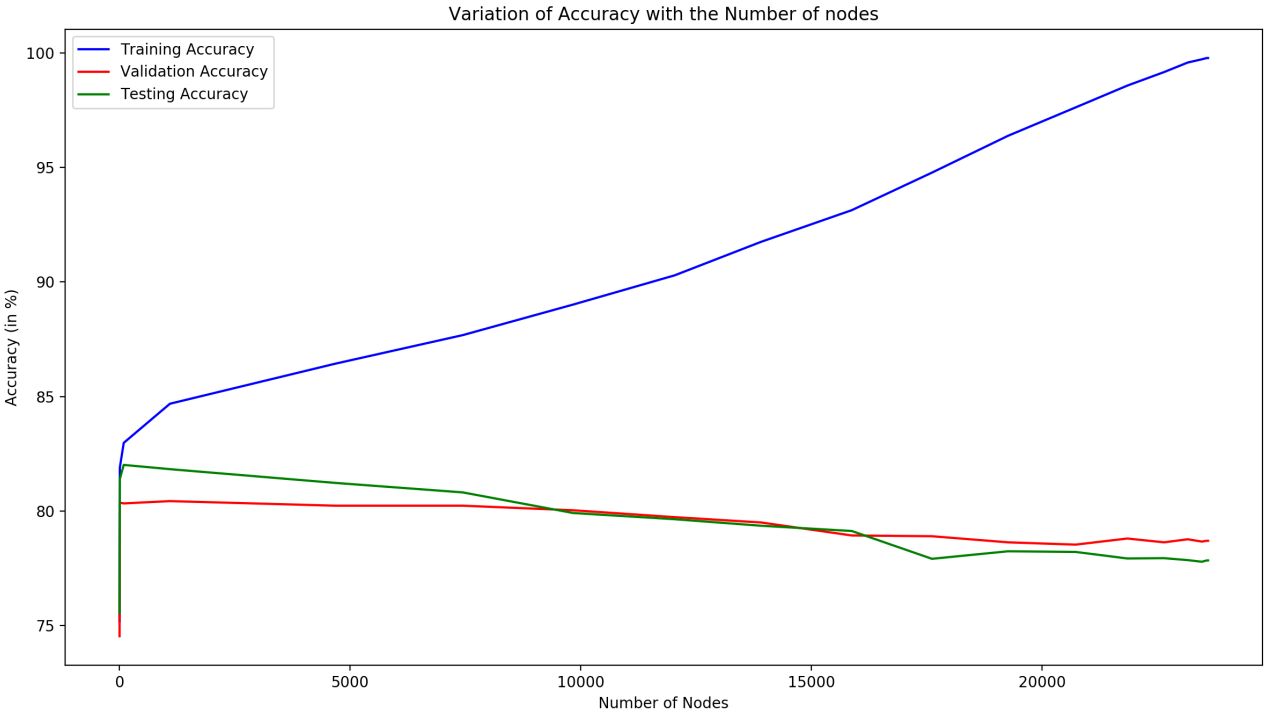
Observations:

- The following chart shows the possible numerical attributes at which the split has taken maximum number of times in each path from root down to leaf. Corresponding thresholds are nothing but the values of the medians on the path at which the maximum number of splits have happened.

Numerical Attribute	Name of the attribute	Maximum times repeated	Thresholds
1	Age	8	62.0, 65.0, 60, 57, 54.0, 51.0, 47, 39
3	Fnlwgt	6	129172, 136824, 142914.0, 159244.0, 127768, 188965.5

5	Education Number	0	-
11	Capital Gain	2	20051, 7688
12	Capital Loss	1	1485
13	Hour per Week	4	70, 65, 57.5, 50

- In this part as we grow the tree, the training accuracy increases as the tree starts overfitting the training set and as a result, the validation and test set accuracy falls. The reason why the decision tree overfits to such an extent is primarily because of the numerical attributes which can take a number of values and correspondingly setting the median value at each node on what the split has to be done. As a reason, all these



medians (or thresholds) are decided by the training set, and hence the classification over the training set increases drastically (> 99%).

- Since the medians are highly training set dependent, the model is so not likely to perform that good on test and validation set. As we can see from the graph, the test/ validation accuracy falls as the model tries to absorb the patterns in the training set, and we can see that there is a declining curve with a negative slope showing a corresponding decrease in accuracies.
- Part(a) results are better than Part(c) although the training set accuracies are much higher in part(c). We concluded this from the observation that the model in part(a)

performs better (on test and validation) than the model learnt in part(c) which is highly dependent on the training set (as it overfits on it).

1(d)

Parameters : (max_depth, min_samples_leaf, min_samples_split)

Case 1: When the attributes were binarised initially: A script was run varying over a number of values for the three main features in this part. Best accuracies over the validation set as evaluated from the script are as follows:

max_depth : 14
min_samples_leaf: 6
min_samples_split : 4
Training Set Accuracy : 84.78%
Validation Set Accuracy : 82.67%
Testing Set Accuracy : 82.34%

max_depth : 14
min_samples_leaf: 6
min_samples_split : 2
Training Set Accuracy : 84.78%
Validation Set Accuracy : 82.67%
Testing Set Accuracy : 82.24%

Optimal Parameters : (14, 6, 2) OR (14, 6, 4).

Case 2: When the attributes weren't binarised initially: A script was run varying over a number of values for the three main features in this part. Best accuracies over the validation set as evaluated from the script are as follows:

max_depth : 12
min_samples_leaf: 3
min_samples_split : 8
Training Set Accuracy : 87.06%
Validation Set Accuracy : 85.4%
Testing Set Accuracy : 84.5%

Optimal Parameters : (12, 3, 8).

PART B	PART D (BEST PARAMETERS)
Training Set Accuracy : 84.38% Validation Set Accuracy : 82.29% Testing Set Accuracy : 84.4%	Training Set Accuracy : 87.06% Validation Set Accuracy : 85.4% Testing Set Accuracy : 84.5%

Comparison : As we can see that the both the models have the same performance over the testing set. However, we can still see a noticeable difference in the accuracies over the training set and the validation set. Sklearn model performs better than the pruned decision tree model on the validation set, and gives a better accuracy on the training set as well. But the main point to note is that our post pruned implementation of the decision tree from the first principles performs at par with the optimised and implemented version of the decision tree classifier in the sklearn library.

Monday, 9 April 2018

1(e)

Parameters : (n_estimators, bootstrap, max_features)

Case 1: When the attributes were binarised initially: A script was run varying over a number of values for the three main features in this part. Best accuracies over the validation set as evaluated from the script are as follows:

n_estimators : 8
bootstrap : True
max_features: log2
Training Set Accuracy : 88.32%
Validation Set Accuracy : 81.1%
Testing Set Accuracy : 81.66%

n_estimators : 4
bootstrap : True
max_features: 5
Training Set Accuracy : 87.62%
Validation Set Accuracy : 81.3%
Testing Set Accuracy : 81.01%

n_estimators : 7
bootstrap : True
max_features: None
Training Set Accuracy : 88.25%
Validation Set Accuracy : 80.93%
Testing Set Accuracy : 80.79%

Optimal Parameters : (8, True, log2) OR (4, True, 5) OR (7, True, None).

Case 2: When the attributes weren't binarised initially: A script was run varying over a number of values for the three main features in this part. Best accuracies over the validation set as evaluated from the script are as follows:

n_estimators : 19
bootstrap : True
max_features: log2
Training Set Accuracy : 99.70%
Validation Set Accuracy : 85.56%
Testing Set Accuracy : 84.78%

n_estimators : 28
bootstrap : True
max_features: None
Training Set Accuracy : 99.78%
Validation Set Accuracy : 85.3%
Testing Set Accuracy : 84.77%

n_estimators : 16
bootstrap : True
max_features: 8
Training Set Accuracy : 99.48%
Validation Set Accuracy : 84.43%
Testing Set Accuracy : 84.73%

Optimal Parameters : **(19, True, log2)** OR (19,True, None) OR (16, True, 8).

Monday, 9 April 2018

PART B	PART C	PART E (RANDOM FOREST)
Training : 84.38% Validation : 82.29% Testing : 84.4%	Training : 99.78% Validation : 78.7% Testing : 77.84%	Training : 99.70% Validation : 85.56% Testing : 84.78%

Comparison : From the accuracies observed, it is clear that the performance of Model E is better than the Model B which is better than Model C . (E > B > C)
In Random Forest Classifier, since we are using bootstrap samples so the model will have low variance and will be stable in nature against the noisy data points because it is essentially sampling with replacement each time. Although, the number of features in each example is 14, but we will restrict ourself to log2(max_number_features) in order to determine the best accuracies as can be seen experimentally. More features means more ways of splitting the node on more impactful features which in a way increases the performance of the model. Number of estimators is essentially the number of trees in the forest which in this case gives us the best result when it is 19. Random Forest Classifier performs the best among all the models and comes out to be a very strong model in this case. And the worst performing model is the one in which the binarisation has been done at each node, and which sets thresholds as per the training set and sucks in performance as a result.

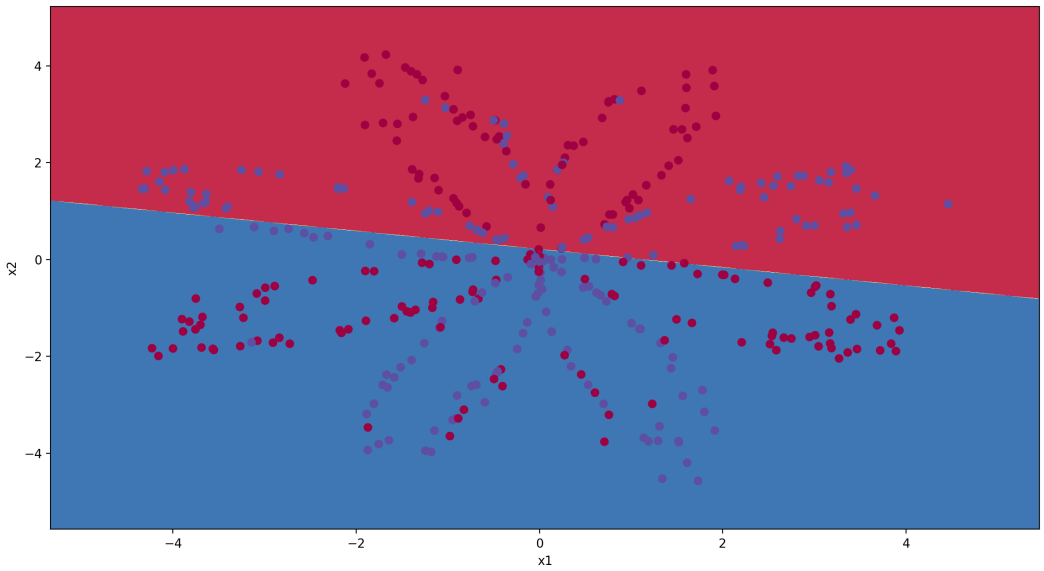
Monday, 9 April 2018

NEURAL NETWORKS (PART 2)

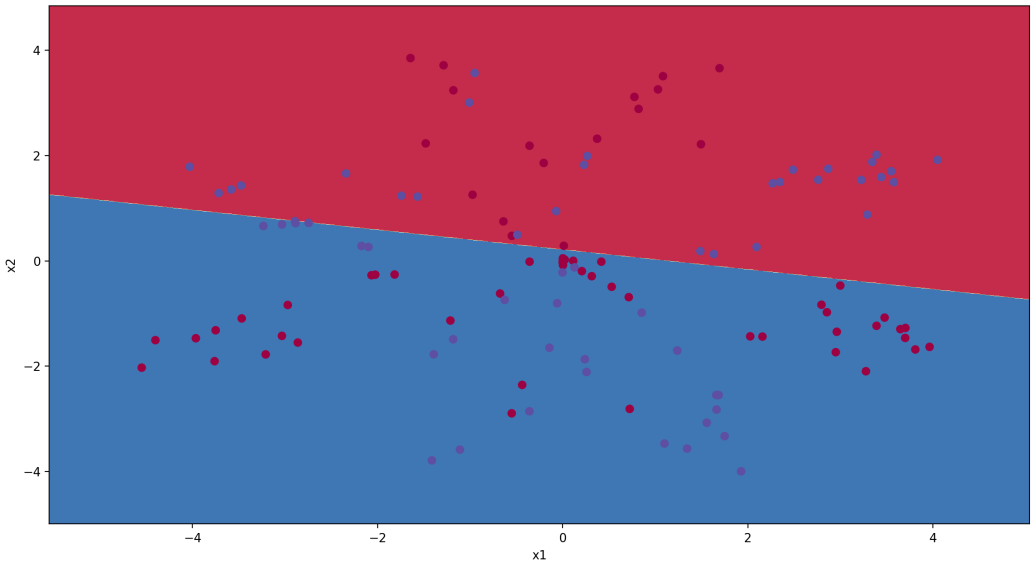
2(b)(i)

LOGISTIC REGRESSION LEARNER : This gives a linear decision boundary which in this case works very poor and the same can be concluded from the accuracies below.

TRAINING SET ACCURACY : 45.79%
TESTING SET ACCURACY : 38.33%



DECISION BOUNDARY IN THE TOY TRAINING SET



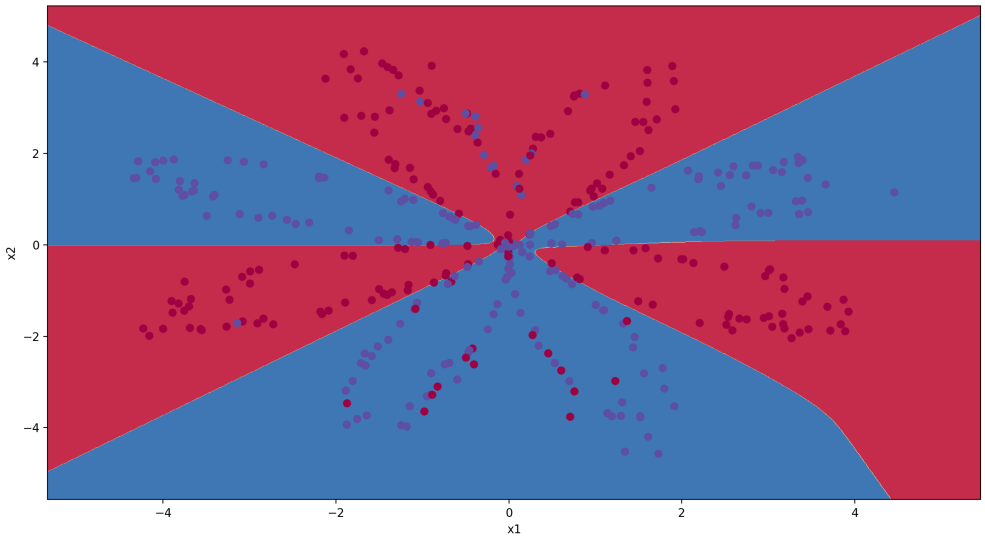
DECISION BOUNDARY IN THE TOY TESTING SET

Monday, 9 April 2018

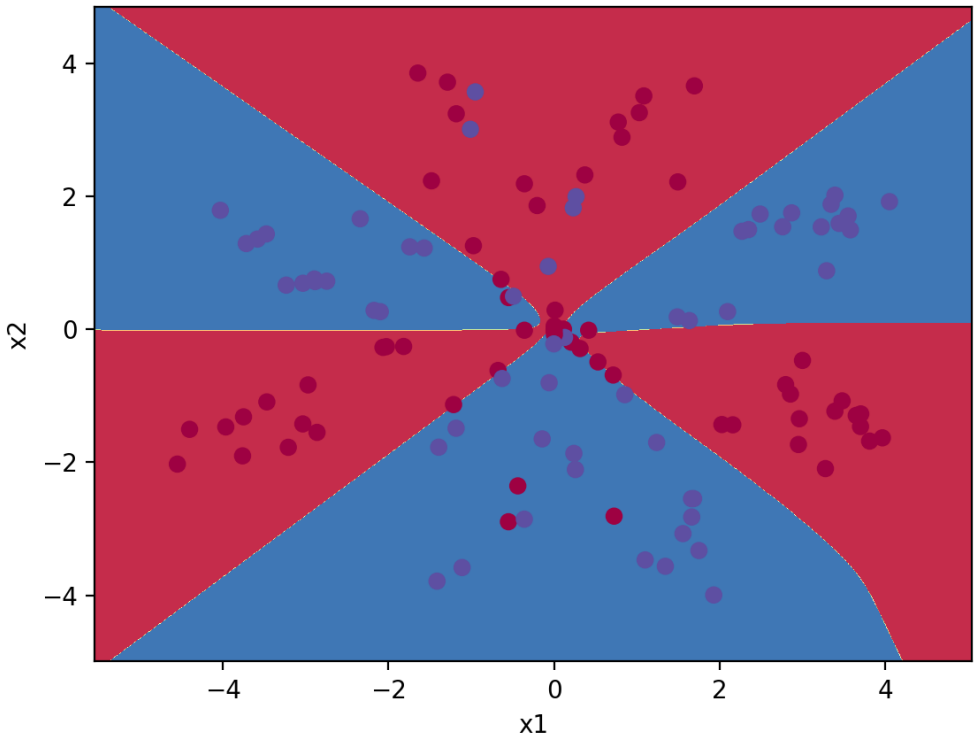
2(b)(ii) NN with Single Hidden Layer having 5 units

Training Set Accuracy : 89.74%

Testing Set Accuracy: 87.5%



DECISION BOUNDARY IN THE TOY TRAINING SET



DECISION BOUNDARY IN THE TOY TESTING SET

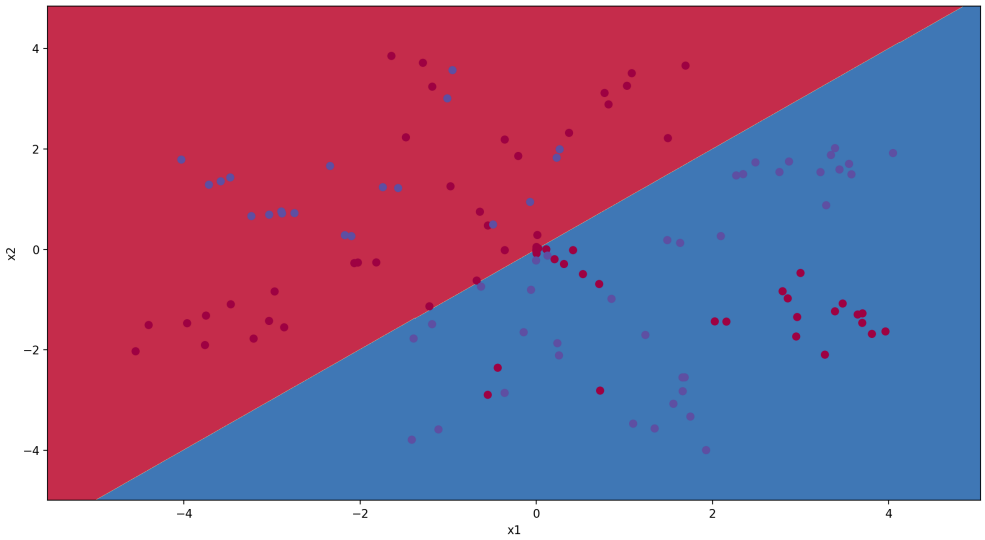
Monday, 9 April 2018

Comparison : In the given scenario, NN is able to realise the patterns in the data much well, and hence gives a much clearer decision boundary. On, the other hand, the linear logistic regression classifier only gives a linear decision boundary which as we can see isn't a good model to realise the data we have in this case. The sigmoidal activation function sitting at the end of each unit adds non-linearity to the data, and gives us non-linear boundary (as can be seen) to fit the patterns in the underlying data well.

2(b)(iii)

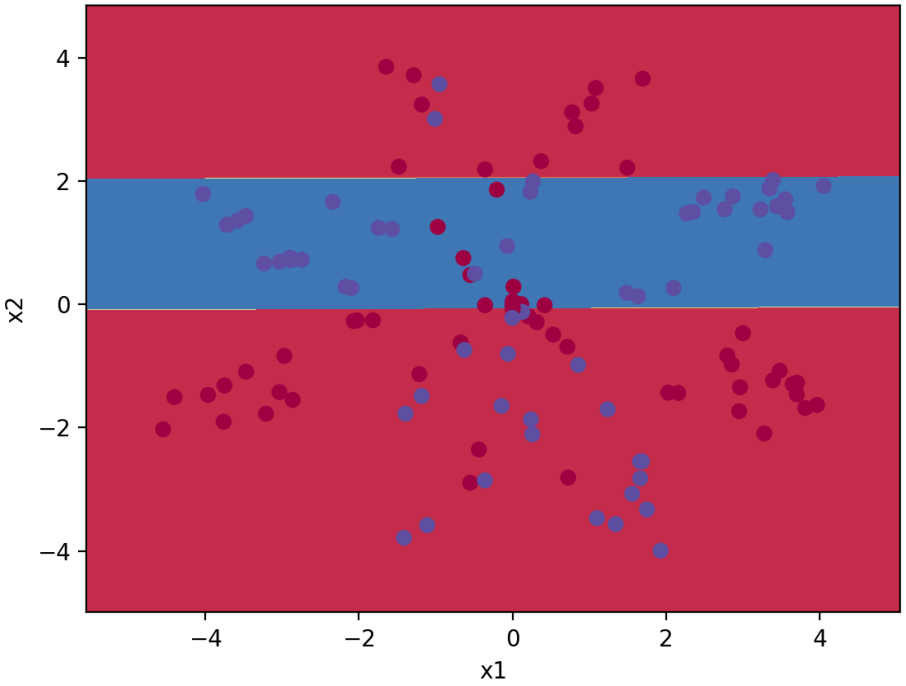
Units (hidden layer)	Training Accuracy(%)	Testing Accuracy(%)	Learning Rate
1	65.79	60.83	0.263
2	68.16	70.0	0.158
3	89.47	86.67	0.105
10	89.47	86.67	0.079
20	90.53	85.0	0.0526
40	90.26	81.67	0.0263

VISUALISATION OF THE GRAPHS:

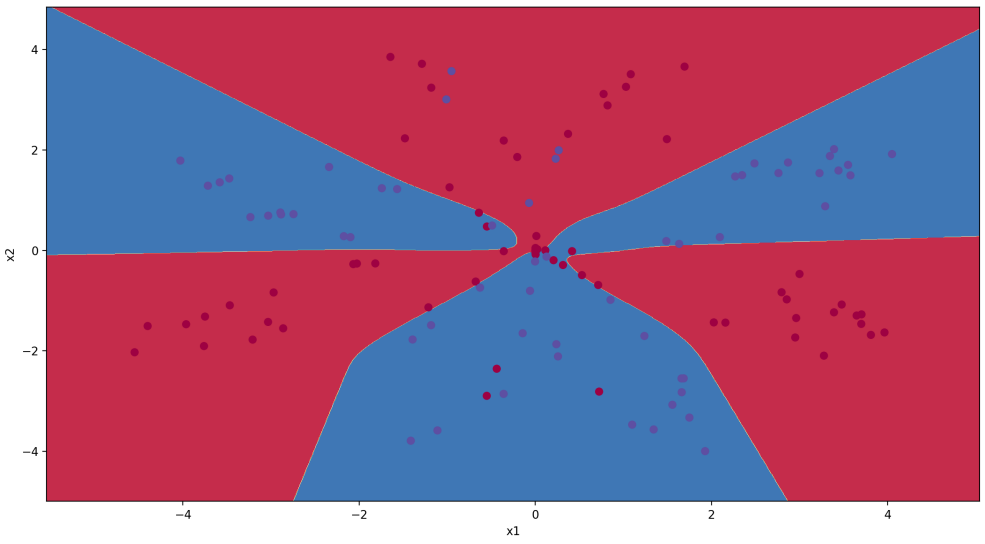


DECISION BOUNDARY WITH 1 UNIT IN THE HIDDEN LAYER FOR TEST DATA

Monday, 9 April 2018

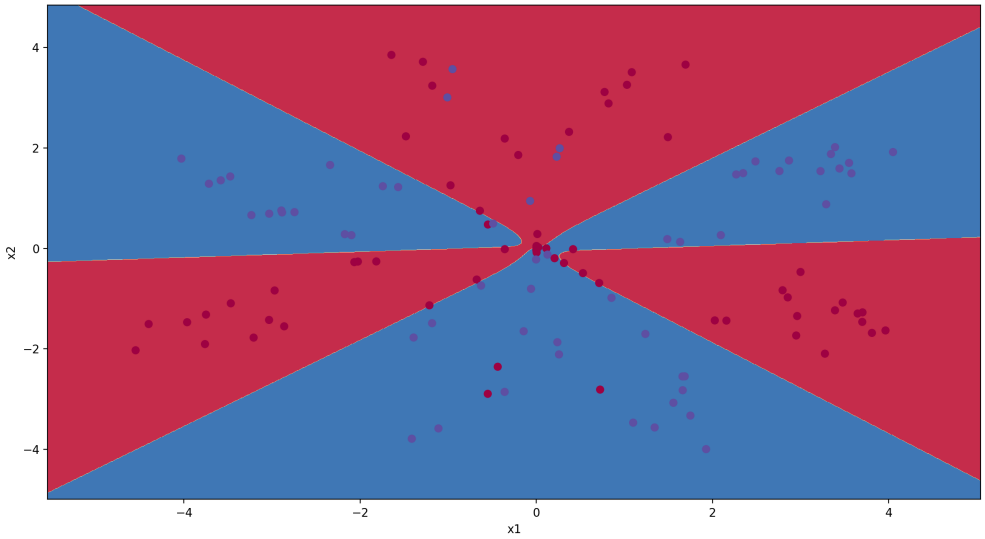


DECISION BOUNDARY WITH 2 UNIT IN THE HIDDEN LAYER FOR TEST DATA

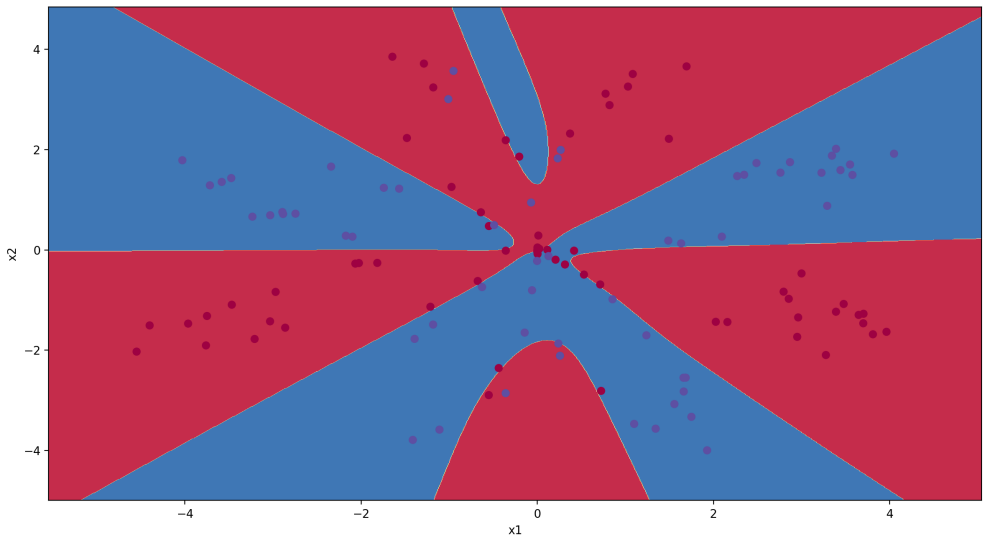


DECISION BOUNDARY WITH 3 UNIT IN THE HIDDEN LAYER FOR TEST DATA

Monday, 9 April 2018

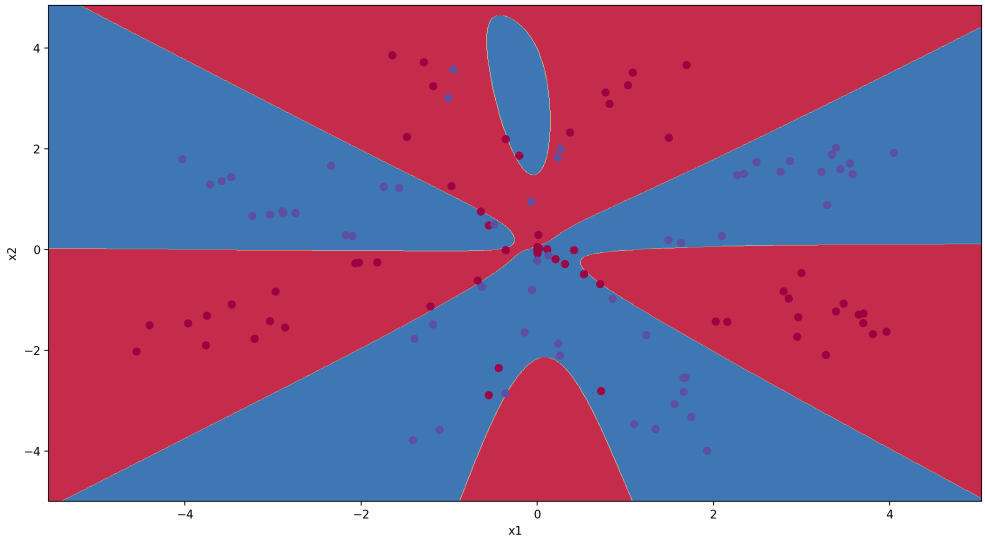


DECISION BOUNDARY WITH 10 UNIT IN THE HIDDEN LAYER FOR TEST DATA



DECISION BOUNDARY WITH 20 UNIT IN THE HIDDEN LAYER FOR TEST DATA

Monday, 9 April 2018



DECISION BOUNDARY WITH 40 UNIT IN THE HIDDEN LAYER FOR TEST DATA

OPTIMAL VALUE OF THE NUMBER OF UNITS = 3 OR 10
REASON: As we can see from the value of accuracies obtained by training different models depending on the number of units in the hidden layer, the testing set accuracies increases initially, and reaches a maximum around 10(or 3), and then again starts decreasing. This observation tells us that as the number of units increases, the complexity of the model increases and it starts learning complex patterns lying within the data. As a result, the accuracy improves. Now when the number of units starts increasing above a certain point, the model starts learning very specific complexities associated with the training data itself and hence overfitting occurs. And as can be seen, the training set accuracy ALWAYS increases with the number of units. And because of overfitting after a certain point, the test accuracy starts falling. Hence, we achieve a maximum at about 10 or 3.

CHANGE IN THE DECISION BOUNDARY: As the number of units increases, the model gets more and more complex and starts segmenting some underlying hidden patterns in the data.
When there was just one unit, decision boundary was a linear separator leading to a very bad accuracy.
When the number of units are 2, decision boundary is more complex than the earlier and is now a two-way separator passing nearly through the centre since most of the points are clustered there. The accuracy is still not very good, since the petal-pattern cannot be captured by such a separator.
When the number of units are 3 or 10, decision boundary starts capturing some non-linear pattern in the shape of a petal and the accuracy becomes drastically good.
When the number of units are 20 or 40, decision boundary starts overfitting and starts segmenting non-linear data in the training set which may not be very well reported in the test data. Thus, we can see some enclosed figures which are very specific to training data only.

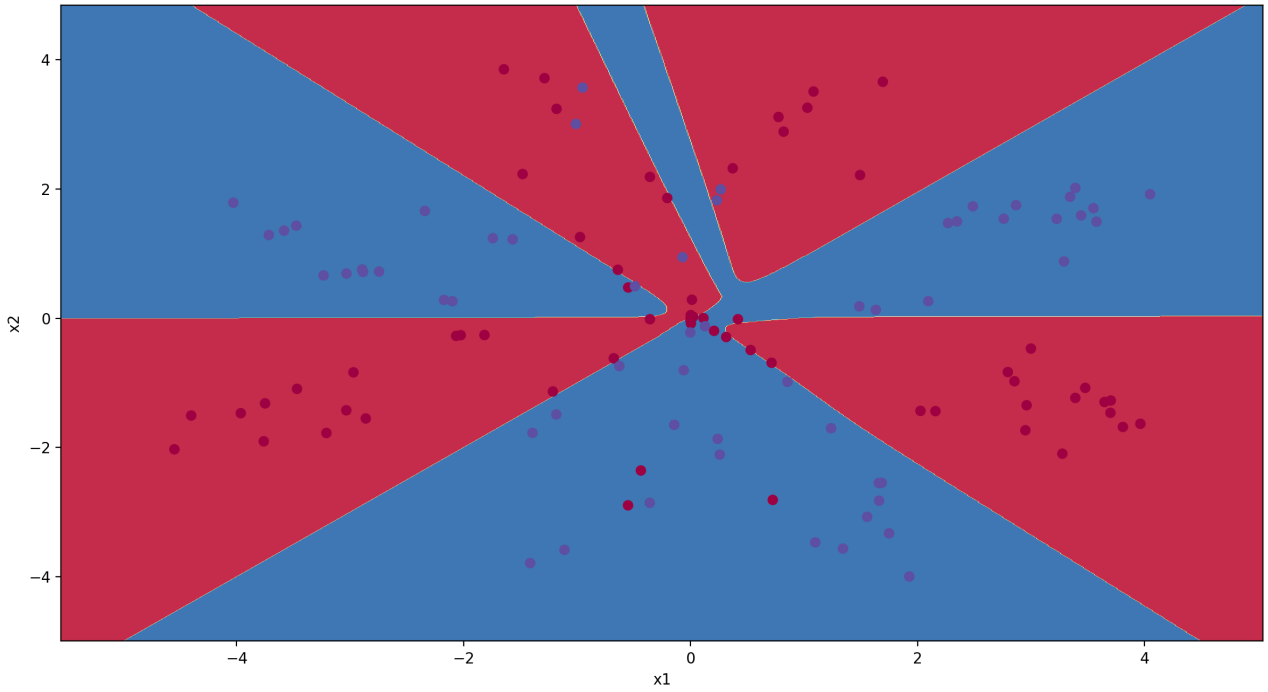
Monday, 9 April 2018

2(b)(iv)
Case when there are two hidden layers each having 5 units
Eta = 0.052

Training Set Accuracy : 89.74%
Testing Set Accuracy : 87.5%

PART (i)	PART (ii)	PART (iii) - 10 units	PART (iv)
Training : 45.79% Testing : 38.33%	Training : 89.74% Testing : 87.5%	Training : 89.47% Testing : 86.67%	Training : 89.74% Testing : 87.5%

Comparison: Here (iv) the accuracies are much similar to the accuracies obtained in the part(ii), and which is in fact the highest among all the models we obtained so far. Part(i) performs the worst since it fails to capture the non-linearity in the decision boundary. Part(iii) compares quite well with the Part(iv). It seems like that this is the best that could have been learned using a network structure like this.



Monday, 9 April 2018

2(c)(i). Working with MNIST.

LINEAR SVM-CLASSIFIER (C = 1)	SINGLE-PERCEPTRON MODEL
Training : 99.87% Testing : 98.83%	Training : 99.34% Testing : 99.08%

Both the models are equally capable and powerful enough to classify the testing dataset with almost 99% accuracy. It seems like that a single perceptron model and a linear classifier are both able to learn the decision boundary with a reasonable amount of error. We see that a single perceptron model performs slightly better than svm-classifer by a margin of mere 0.25%, but the difference is too small to make a reasonable difference.

(ii)

Stopping Criteria : If the difference in the validation set accuracy stops increasing (in a certain well-defined threshold margin around it) across successive epochs, then declare that the algorithm 'converged'.

Number of units in the hidden layer = 100
Training Accuracy : 99.47%
Testing Accuracy : 99.19%

Accuracy comes pretty much the same in this part as well denoting that the single hidden layer of 100 units is a strong classifier in this binary classification setting. However, number of iterations for convergence comes out to be much less than the previous parts. (~921 iterations)

(iii)

Pure Sigmoid Activation	ReLU Activation
Training : 99.14% Testing : 98.89%	Training : 98.62% Testing : 98.61%
Number of iterations = 160	Number of iterations = 140

As can be seen, ReLU converges faster than Sigmoid activation using the stopping criteria as defined above. Accuracy comes pretty much the same in both the cases i.e. ReLU and Pure Sigmoid. Apart from faster convergence, ReLU gives us sparsity in the representation, and gives us much simpler but powerful models with a reasonable accuracy in comparison to the deep complex functions learnt by using sigmoid activation at each unit. So, apart form giving good accuracies on the model, it is computationally efficient as well.