COL774 - Assignment 3

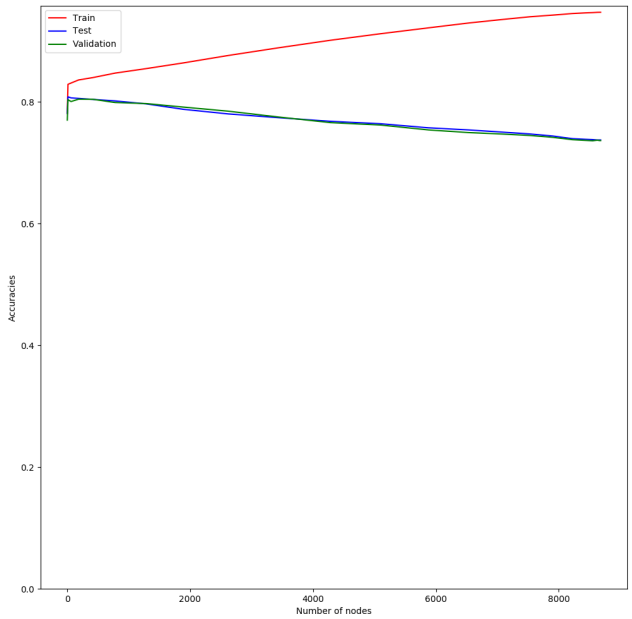
Ankit Solanki
2016CS50401

April 8, 2019

1 Part-A: Decision Trees (and Random Forests)

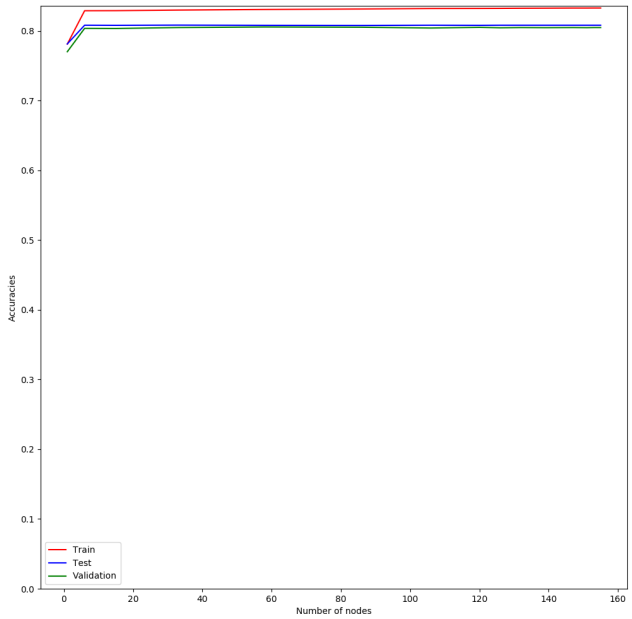
1.1 Part-a

```
[#] Part-A

[>] DECISION TREE BUILT:
[>] Root-Node:
[*] Label: None
[*] Number of children: 11
[>] Label Counts: Counter({'0': 14057, '1': 3943})
[>] Splitting Feature: X6
[>] Splitting Feature Value: None

[*] Accuracy Train: 0.9471666666666667
[*] Accuracy Test: 0.7368333333333333
[*] Accuracy Validation: 0.7366666666666667
[*] Time taken: 132.7152693271637
```



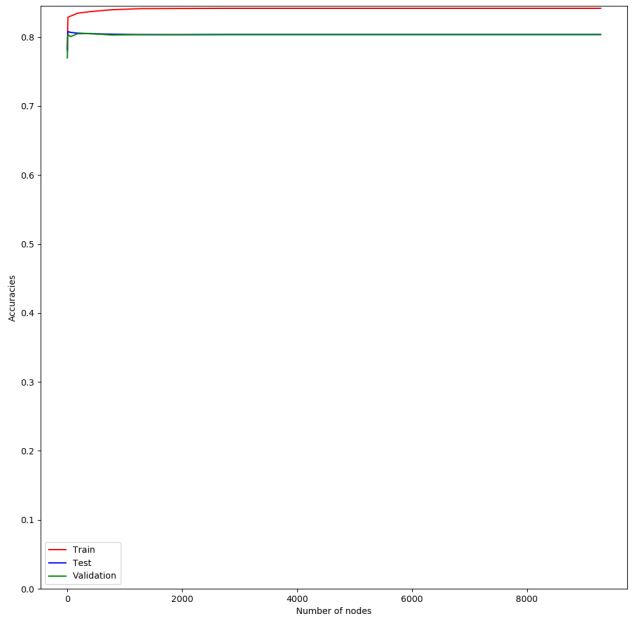
1.2 Part-b

```
[#] Part-B: Post-Pruning part

[>] DECISION TREE BUILT:
[-] Pruning

[*] Accuracy Train: 0.8326666666666667
[*] Accuracy Test: 0.808
[*] Accuracy Validation: 0.8066666666666666
```



1.3 Part-c

```
[#] Part-C | Pre-processing whilst building the tree

[>] DECISION TREE BUILT:
[>] Root-Node:
[*] Label: None
[*] Number of children: 11
[>] Label Counts: Counter({'0': 14057, '1': 3943})
[>] Splitting Feature: X6
[>] Splitting Feature Value: None

[*] Accuracy Train: 0.8418333333333333
[*] Accuracy Test: 0.8038333333333333
[*] Accuracy Validation: 0.8038333333333333
[*] Time taken: 176.9152867794037
```



1.4 Part-d

```
[#] Part-D | Using DecisionTreeClassifier
[*] Accuracy Train: 0.8308333333333333
[*] Accuracy Test: 0.8108333333333333
[*] Accuracy Validation: 0.8046666666666666

[*] Best Set of Parameters:
    [>] min_samples_split = 1000
    [>] min_samples_leaf = 10
    [>] max_depth = 5
```

- As max_depth increase, val-set accuracy decreases, due to overfitting, and thus not learning well to generalize. And after a while, it becomes constant, as logical depth saturates.
- min_samples_leaf: val-set accuracy keep on gradually increasing and then become constant, which shows the tree is overfitting and is in need of pruning.
- min_samples_split: val-set accuracy increases gradually and then become constant: It removes the effect of overfitting as it increases.

1.5 Part-e

```
[#] Part-E | Using One-hot encoding in categorical data
```

```
[*] Accuracy Train: 0.83
[*] Accuracy Test: 0.8085
[*] Accuracy Validation: 0.7981666666666667
```

1.6 Part-f

```
[#] Part-F | Random Forests Using One-hot encoding in categorical data
[*] Accuracy Train: 0.9984444444444445
[*] Accuracy Test: 0.8015
[*] Accuracy Validation: 0.8043333333333333

[*] Best Set of Parameters:
    [>] min_samples_split = 1000
    [>] min_samples_leaf = 10
    [>] max_depth = 5
```

- val-set accuracy increases with n_estimators, as the more trees there are, more accuracy; but the trade off ends because further increasing trees take too much time, but the change in val accuracy is statistically very minimal.
- Dataset is not very big, so bootstrap=True increases accuracy till the second decimal
- With max_features, for me, it is dependent on the data. It is hard to compare when the change is so little, but I saw that around 50% of total features, gives a better accuracy.

1.7 Comparisons

Since in every section, we have to comment on our findings and do comparison, I would do it all it in here.

- In the part-a, the tree is built without any prevention from overfitting, and it is not surprising to see that it overfits. The train accuracy is around 94.7%, but the validation set accuracy is 73.6%. If only the root node is used to classify the tree, it would be giving an accuracy of 78.09&. Since, our model performs even below this, it definitely is doing something wrong.
- In the part-b, we use pruning. I, first, traverse my tree in breadth first. By which, I mean, I form a list in which a node with depth d comes after all node with depth d-1, and the nodes with depth d but which would have fallen in the left of it whilst traversing from left to right. I, then, move from leaf to root, whilst checking if it would have been pruned, then would it be giving an accuracy more or same on validation set? Upon receiving a positive reply, I thereby prune it in my successive iterations, and it gives me better accuracy.

- And when I pre-process tree whilst building, or should I say, I follow the method: At any given internal node of the tree, a numerical attribute is considered for a two way split by calculating the median attribute value from the data instances coming to that node, and then computing the information gain if the data was split based on whether the numerical value of the attribute is greater than the median or not. This time, without pruning, I am getting an accuracy of 80.38% which is way better than part-a. Perhaps I would have been getting more if pruned.
- Decision Tree method of scikit-learn, gives me an accuracy of 80.46%, which is almost same as I am getting in part-c, the difference is super small.
- Using one-hot encoding drops my accuracy by a 0.03% which is too low to consider, but still it shows that too many features are not necessary useful.
- Random forest method of scikit-learn with the same one-hot encoding performs 99.8% on training set, which is extraordinary but gives an 80.43% on validation set.

2 Part-B: Neural Networks

2.1 Binary Classification

2.1.1 Part-a

Implemented as instructed.

2.1.2 Part-b

Implemented as instructed.

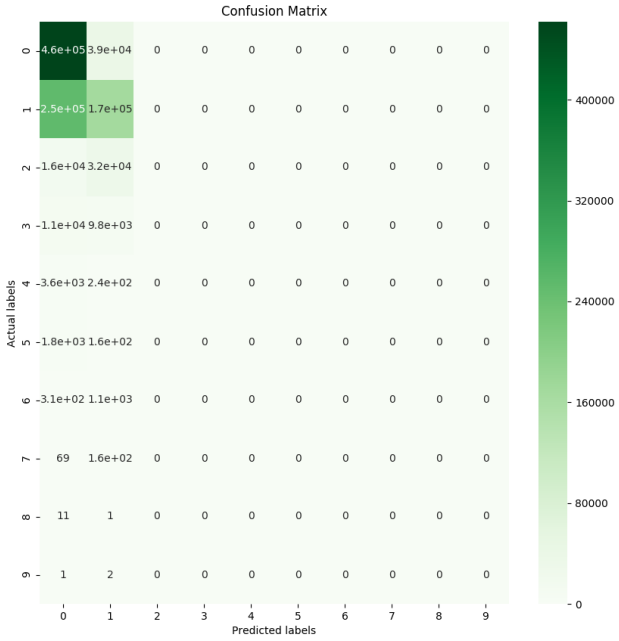
2.1.3 Part-c

NOTE: Since, there are too many graphs to attach: I'm gonna attach a part of them, and the rest plotted graph can be found in the attached code–in the organized manner–as the name of the part.

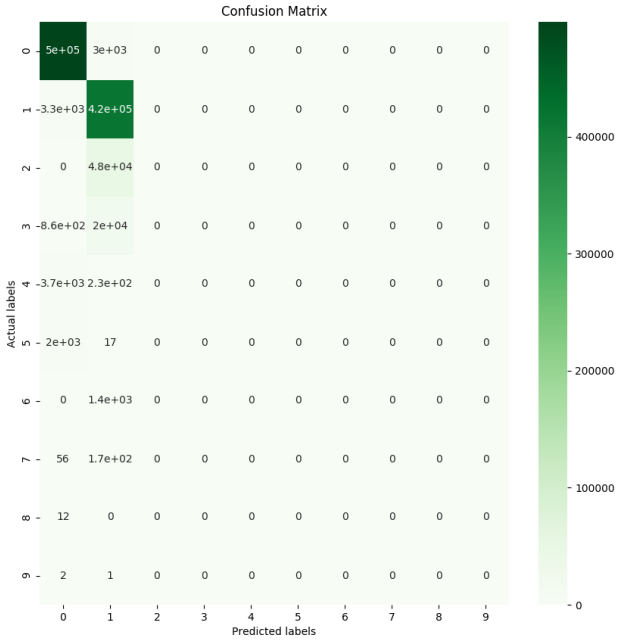
```
[#]------------PART-C--------------[#]

[>] Neural Network:
[>] Shape:  [85, 5, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100

[*] Time taken: 48.37870812416077
[*] Train Accuracy: 0.6444622151139544
[*] Test Accuracy: 0.6312296312296313

[>] Neural Network:
```

```
[>] Shape:  [85, 10, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100
[.] Max Epoch:  1000

[*] Time taken: 58.64585995674133
[*] Train Accuracy: 0.7350259896041583
[*] Test Accuracy: 0.6985906985906986

[>] Neural Network:
[>] Shape:  [85, 15, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100

[*] Time taken: 25.76823353767395
[*] Train Accuracy: 0.8285885645741703
[*] Test Accuracy: 0.8015318015318015

[>] Neural Network:
[>] Shape:  [85, 20, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100

[*] Time taken: 20.203080654144287
[*] Train Accuracy: 0.8854458216713315
[*] Test Accuracy: 0.8614788614788614

[>] Neural Network:
[>] Shape:  [85, 25, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100

[*] Time taken: 52.13519358634949
[*] Train Accuracy: 0.9233106757297082
[*] Test Accuracy: 0.9173789173789174
```

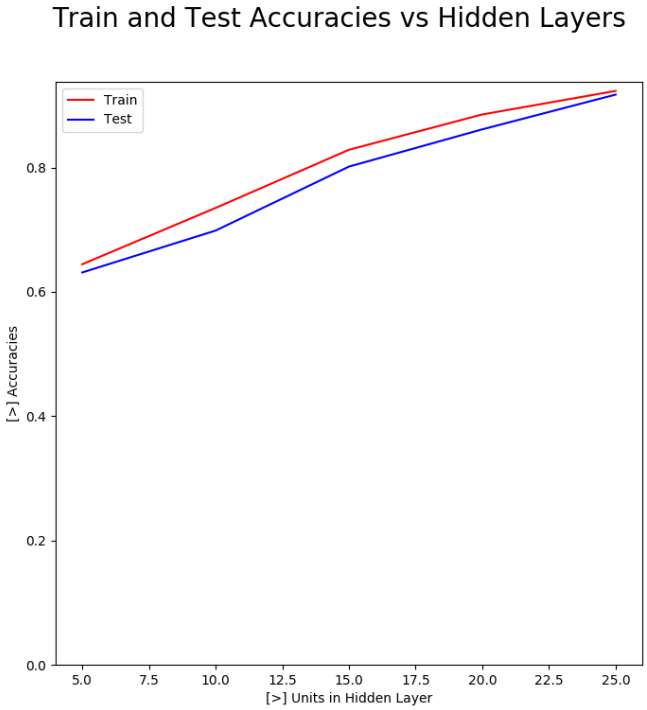
Confusion Matrix | Hidden Units: 5



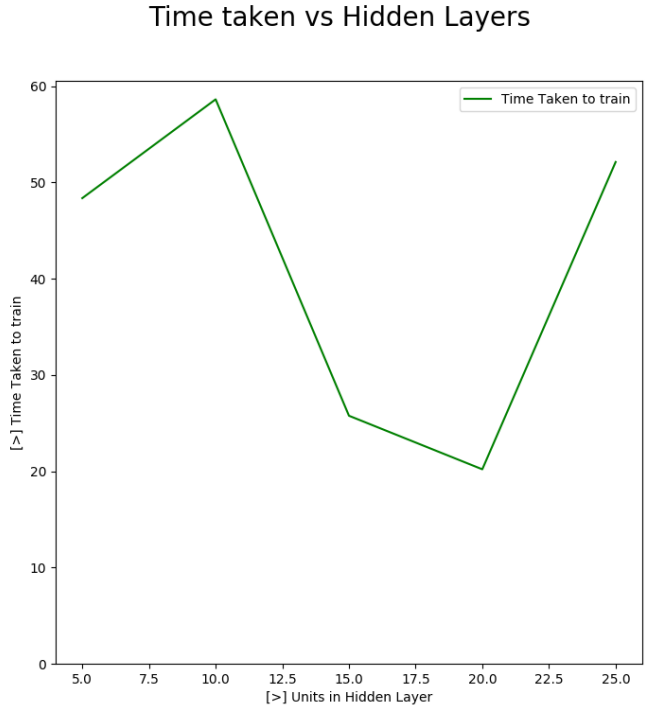
Confusion Matrix | Hidden Units: 25



Accuracies vs Hidden units | Hidden Units: 5, 10, 15, 20, 25



Time taken vs Hidden units | Hidden Units: 5, 10, 15, 20, 25



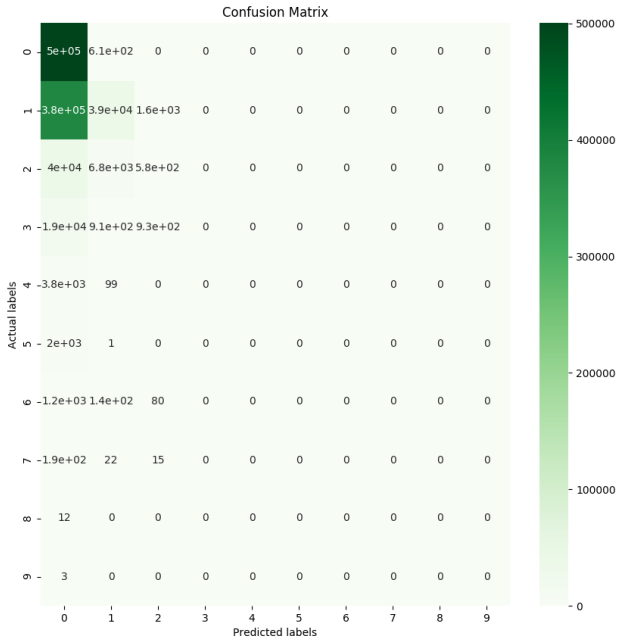
2.1.4 Part-d

```
[#]------------PART-D--------------[#]
```

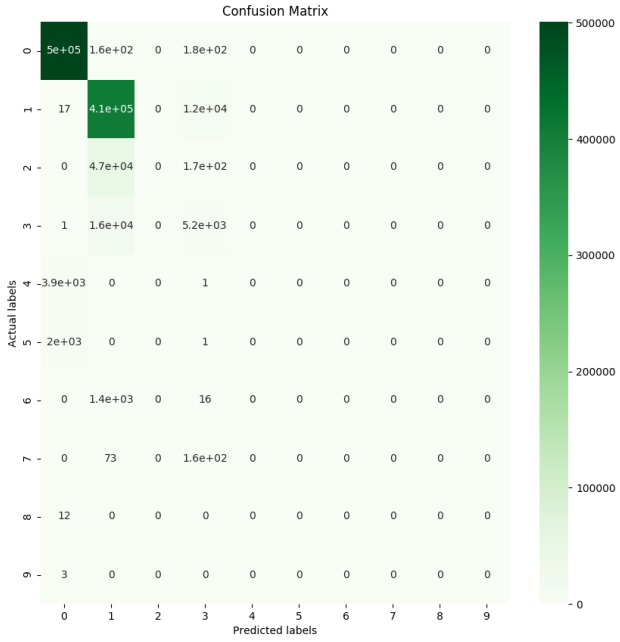
```
[>] Neural Network:
[>] Shape:  [85, 5, 5, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100

[*] Time taken: 48.90382671356201
[*] Train Accuracy: 0.5450219912035186
[*] Test Accuracy: 0.5401375401375401

[>] Neural Network:
[>] Shape:  [85, 10, 10, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100

[*] Time taken: 87.5962553024292
[*] Train Accuracy: 0.8817273090763694
[*] Test Accuracy: 0.8724718724718725

[>] Neural Network:
[>] Shape:  [85, 15, 15, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100

[*] Time taken: 138.35107016563416
[*] Train Accuracy: 0.9341063574570172
[*] Test Accuracy: 0.9161779161779162

[>] Neural Network:
[>] Shape:  [85, 20, 20, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100

[*] Time taken: 121.30362486839294
[*] Train Accuracy: 0.9232706917233107
[*] Test Accuracy: 0.9228179228179229

[>] Neural Network:
[>] Shape:  [85, 25, 25, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100
[.] Max Epoch:  1000

[*] Time taken: 74.93778967857361
[*] Train Accuracy: 0.9716513394642143
```

```
[*] Test Accuracy: 0.9676829676829677
```

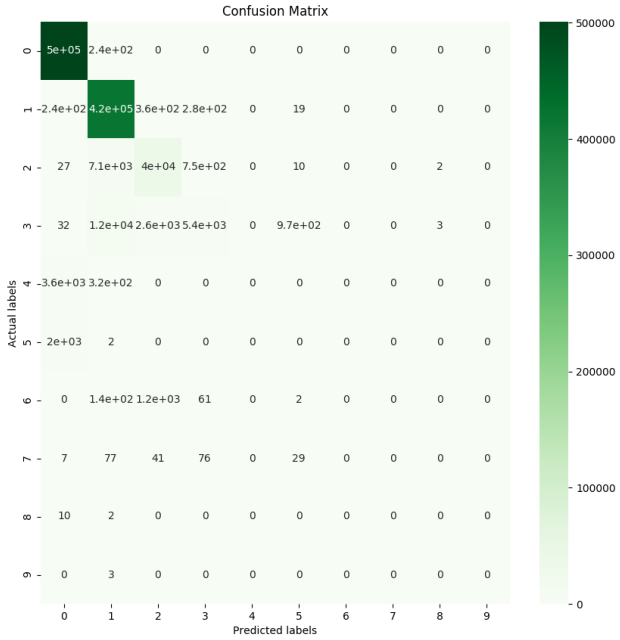
Confusion Matrix | Hidden Units: 5



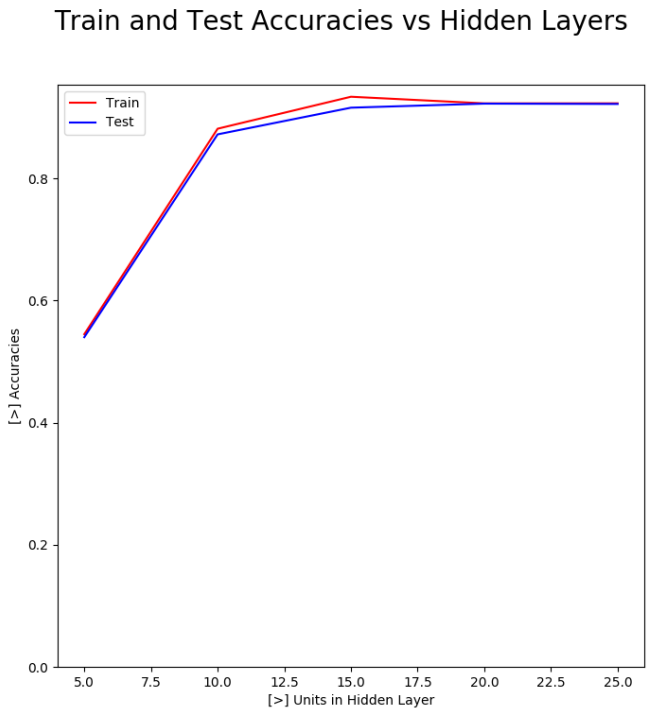
Confusion Matrix | Hidden Units: 15



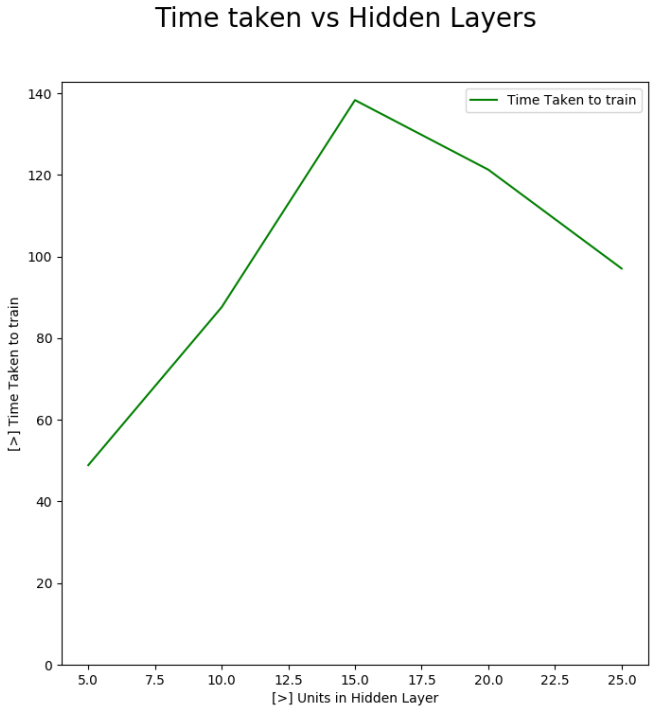
Confusion Matrix | Hidden Units: 25



Accuracies vs Hidden units | Hidden Units: [5,5] [10,10] [15,15] [20,20] [25,25]



Time taken vs Hidden units | Hidden Units: [5,5] [10,10] [15,15] [20,20] [25,25]

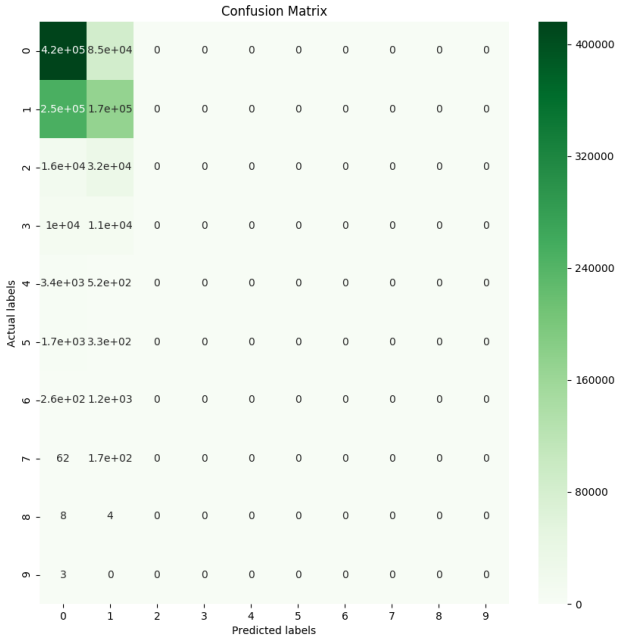


2.1.5 Part-e

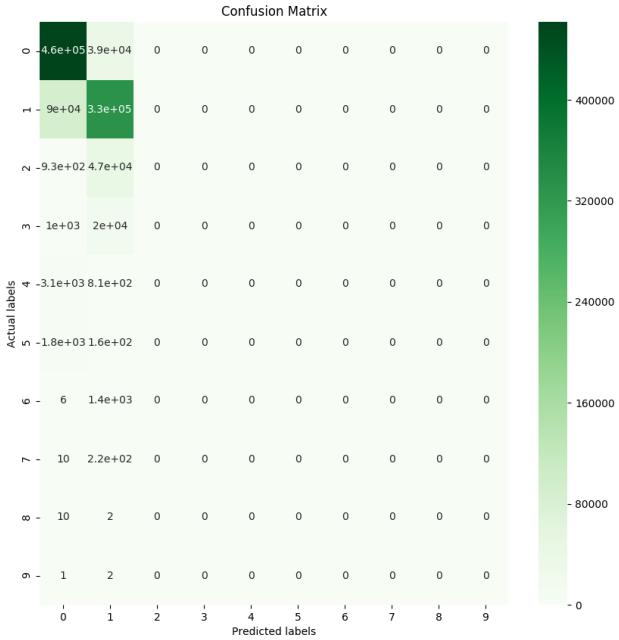
```
USING ADAPTIVE LEARNING RATE
[#]------------PART-E--C------------[#]
[>] Neural Network:
[>] Shape:  [85, 5, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100

[*] Time taken: 11.63818359375
[*] Train Accuracy: 0.6130347860855657
[*] Test Accuracy: 0.5873925873925874

[>] Neural Network:
[>] Shape:  [85, 10, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100

[*] Time taken: 11.937899112701416
[*] Train Accuracy: 0.8313474610155938
[*] Test Accuracy: 0.8211328211328212

[>] Neural Network:
[>] Shape:  [85, 15, 10]
```

```
[.] Learning Rate:  0.1
[.] Batch Size:  100

[*] Time taken: 17.90938663482666
[*] Train Accuracy: 0.831827269092363
[*] Test Accuracy: 0.794063794063794

[>] Neural Network:
[>] Shape:  [85, 20, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100

[*] Time taken: 16.634249687194824
[*] Train Accuracy: 0.9230307876849261
[*] Test Accuracy: 0.9194039194039194

[>] Neural Network:
[>] Shape:  [85, 25, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100

[*] Time taken: 16.012179374694824
[*] Train Accuracy: 0.9232307077169132
[*] Test Accuracy: 0.9220709220709221
```

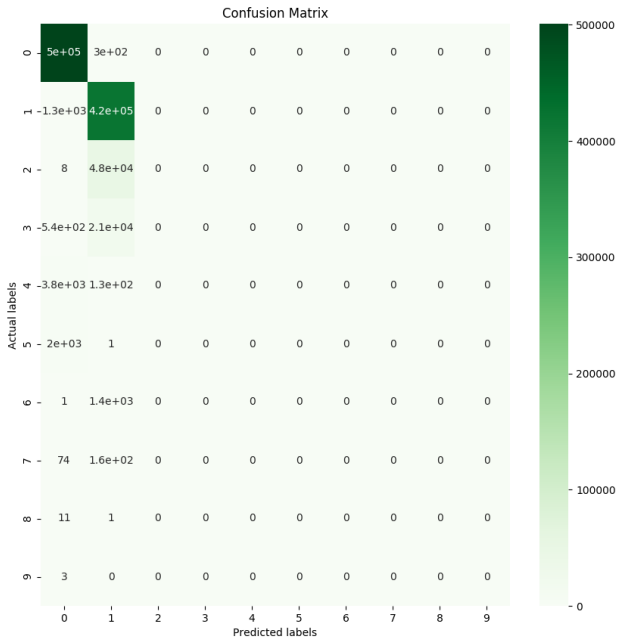
Confusion Matrix | Hidden Units: 5



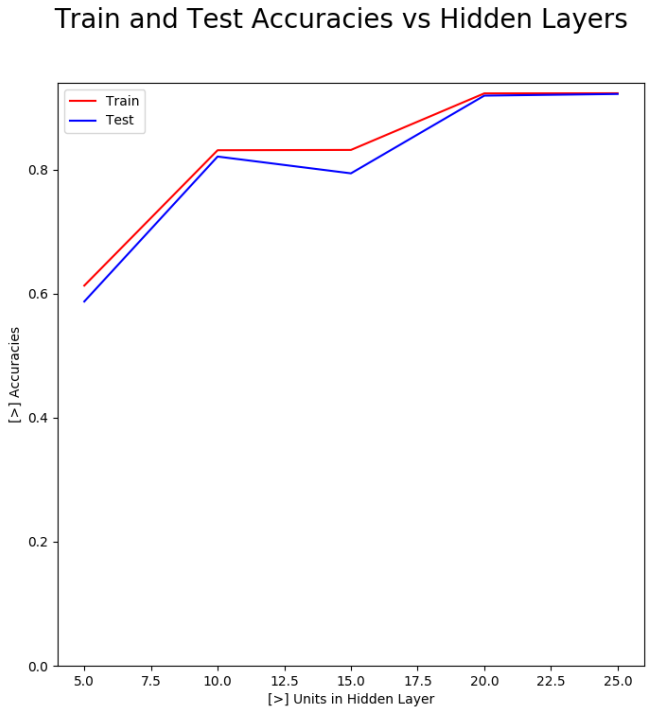
Confusion Matrix | Hidden Units: 15



Confusion Matrix | Hidden Units: 25

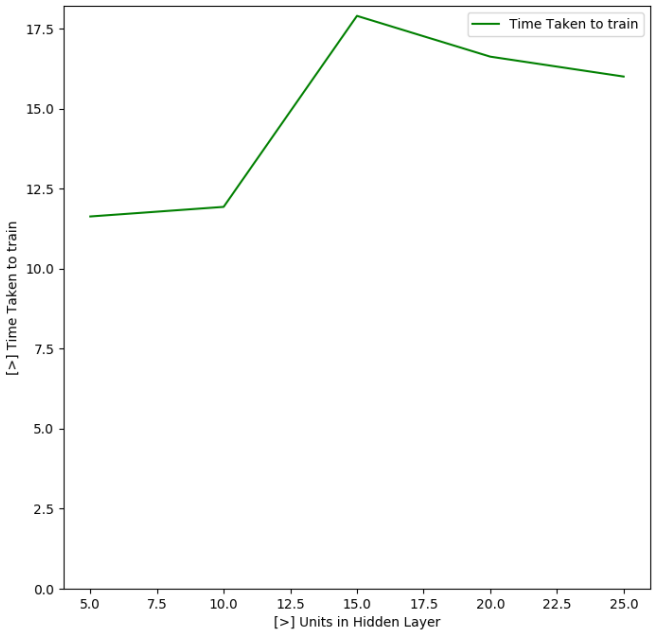


Accuracies vs Hidden units | Hidden Units: 5, 10, 15, 20, 25



Time taken vs Hidden units | Hidden Units: 5, 10, 15, 20, 25

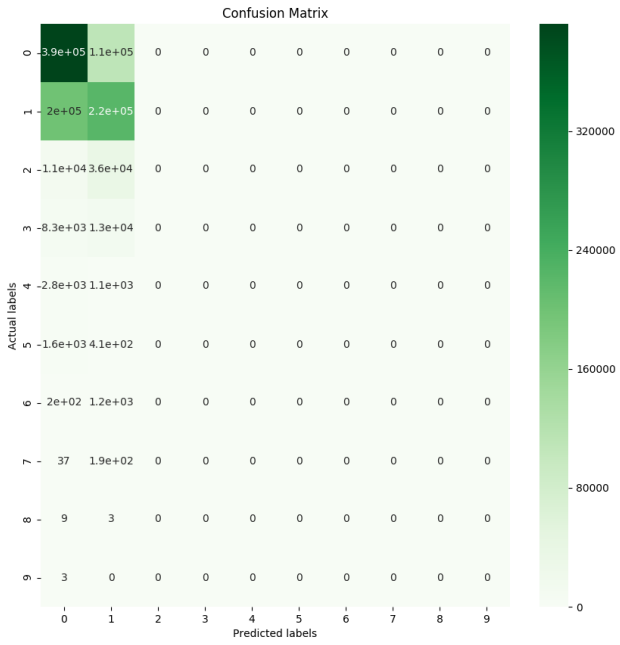
Time taken vs Hidden Layers



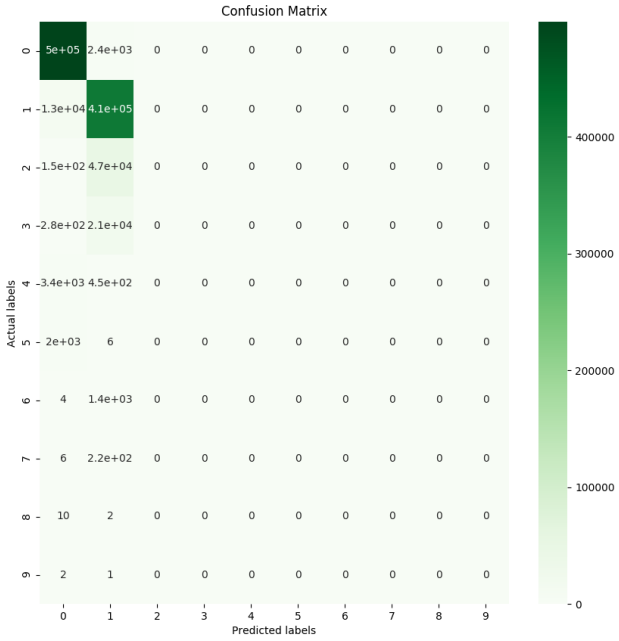

```
[#]------------PART-E--D------------[#]
[>] Neural Network:
[>] Shape:  [85, 5, 5, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100

[*] Time taken: 14.218446493148804
[*] Train Accuracy: 0.6347860855657737
[*] Test Accuracy: 0.6146356146356147

[>] Neural Network:
[>] Shape:  [85, 10, 10, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100

[*] Time taken: 25.27064800262451
[*] Train Accuracy: 0.8573370651739304
[*] Test Accuracy: 0.843965843965844

[>] Neural Network:
[>] Shape:  [85, 15, 15, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100
```

```
[*] Time taken: 26.252105236053467
[*] Train Accuracy: 0.9165933626549381
[*] Test Accuracy: 0.9086539086539086

[>] Neural Network:
[>] Shape:  [85, 20, 20, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100

[*] Time taken: 14.38525104522705
[*] Train Accuracy: 0.9474610155937625
[*] Test Accuracy: 0.9440919440919441

[>] Neural Network:
[>] Shape:  [85, 25, 25, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100

[*] Time taken: 15.839985847473145
[*] Train Accuracy: 0.9233106757297082
[*] Test Accuracy: 0.922934922934923
```

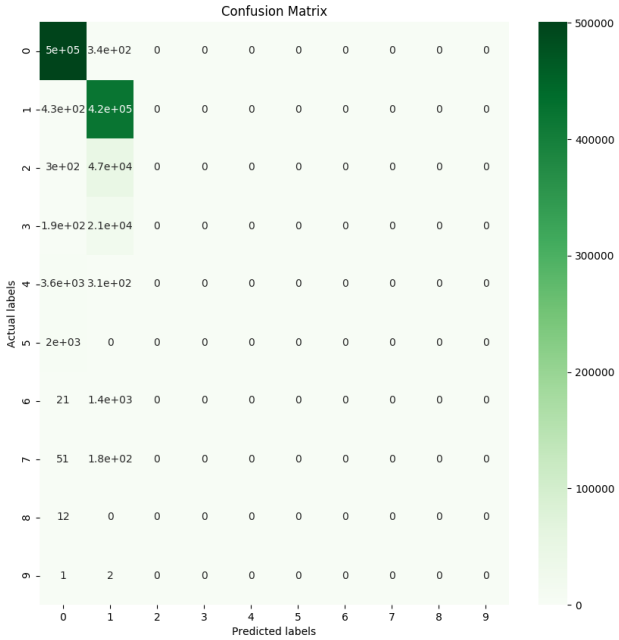
Confusion Matrix | Hidden Units: 5



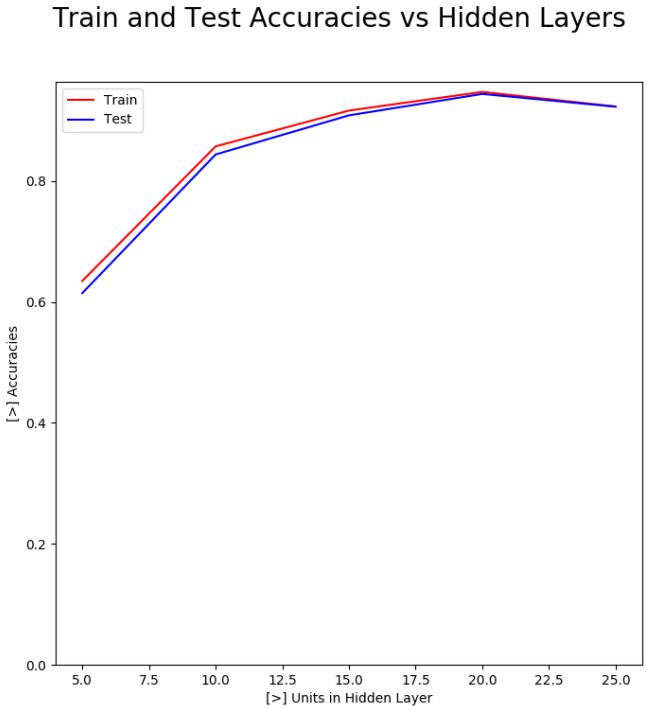
Confusion Matrix | Hidden Units: 15



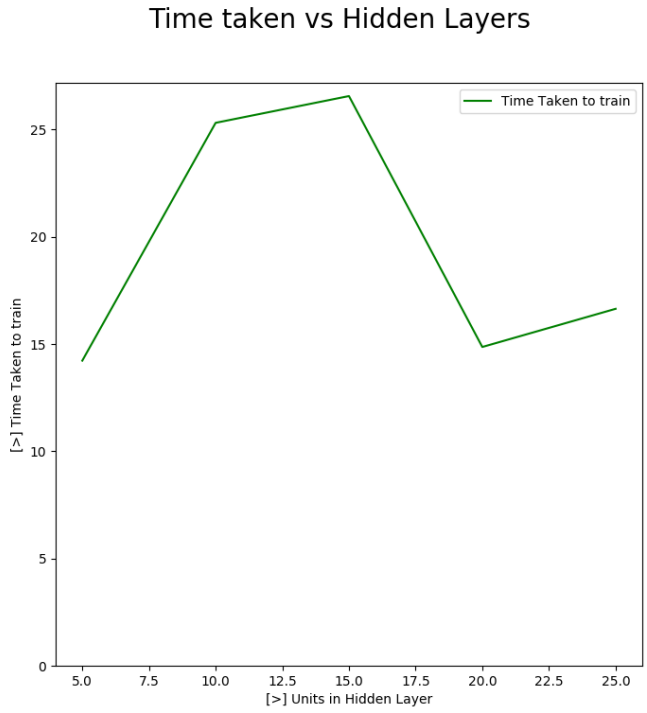
Confusion Matrix | Hidden Units: 25



Accuracies vs Hidden units | Hidden Units: [5,5] [10,10] [15,15] [20,20] [25,25]



Time taken vs Hidden units | Hidden Units: [5,5] [10,10] [15,15] [20,20] [25,25]



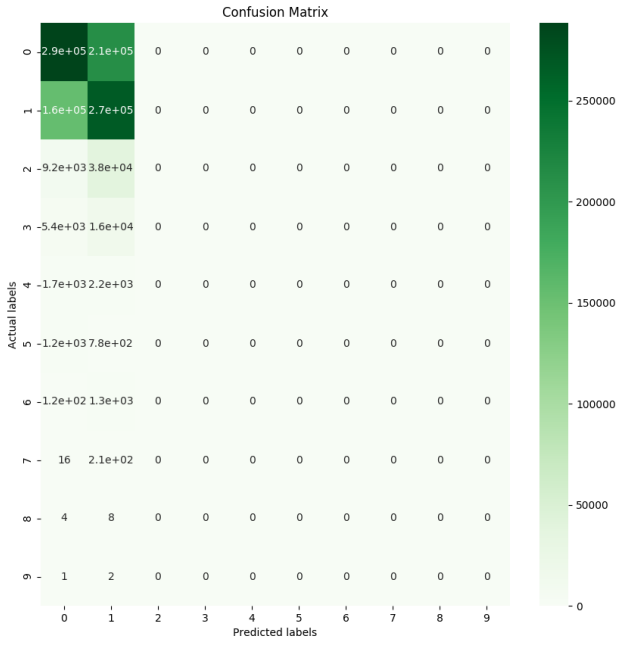
2.1.6 Part-f

USING ReLU ON HIDDEN LAYERS

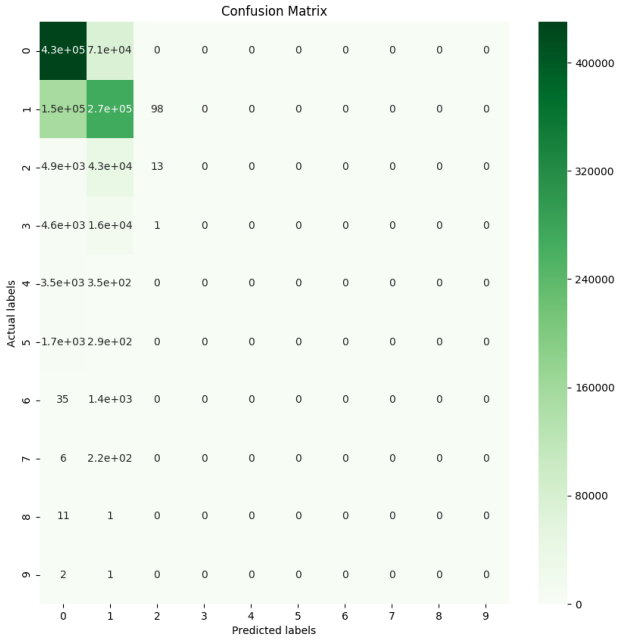
```
[#]------------PART-F--C------------[#]
[>] Neural Network:
[>] Shape:  [85, 5, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100
[.] Tolerance:  0.0001
[.] Activation Function for Hidden Layers:  relu

[*] Time taken: 6.734306573867798
[*] Train Accuracy: 0.5895241903238705
[*] Test Accuracy: 0.555895555895556

[>] Neural Network:
[>] Shape:  [85, 10, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100
[.] Tolerance:  0.0001
[.] Activation Function for Hidden Layers:  relu

[*] Time taken: 8.615839004516602
[*] Train Accuracy: 0.8159136345461815
[*] Test Accuracy: 0.8091678091678092

[>] Neural Network:
[>] Shape:  [85, 15, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100
[.] Tolerance:  0.0001
[.] Activation Function for Hidden Layers:  relu

[*] Time taken: 12.52137041091919
[*] Train Accuracy: 0.7371451419432227
[*] Test Accuracy: 0.6995586995586995

[>] Neural Network:
[>] Shape:  [85, 20, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100
[.] Tolerance:  0.0001
[.] Activation Function for Hidden Layers:  relu

[*] Time taken: 13.784131050109863
[*] Train Accuracy: 0.8988404638144742
[*] Test Accuracy: 0.8775078775078775
```

```
[>] Neural Network:
[>] Shape:  [85, 25, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100
[.] Tolerance:  0.0001
[.] Activation Function for Hidden Layers:  relu

[*] Time taken: 18.944247722625732
[*] Train Accuracy: 0.9145541783286686
[*] Test Accuracy: 0.8894628894628894
```

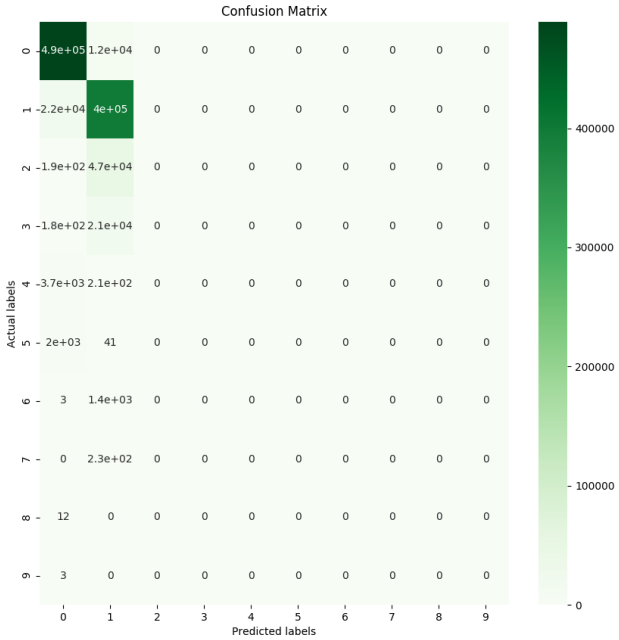
Confusion Matrix | Hidden Units: 5



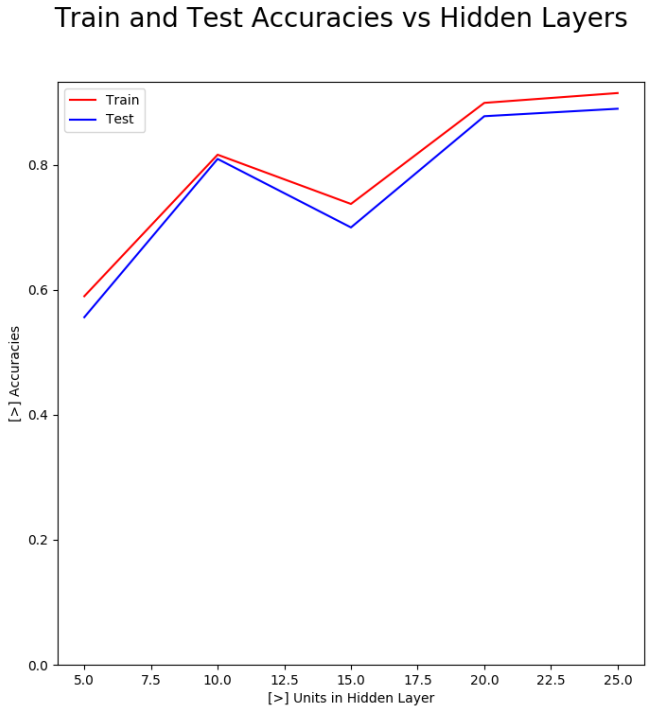
Confusion Matrix | Hidden Units: 15



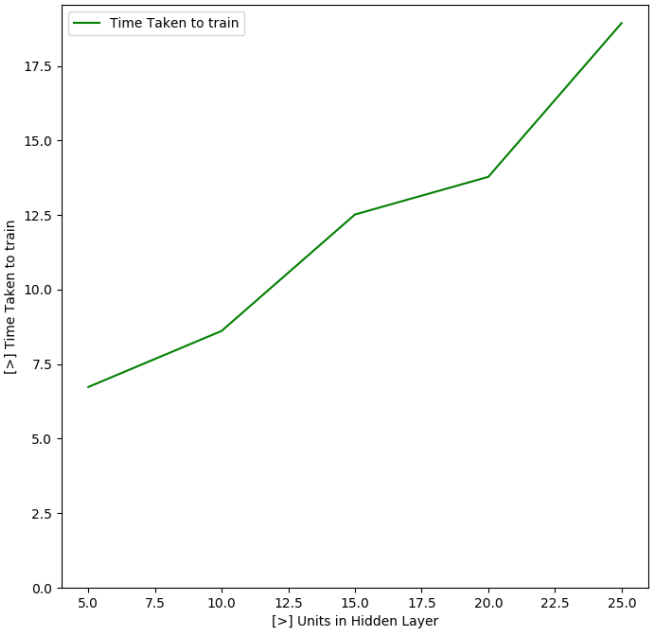
Confusion Matrix | Hidden Units: 25



Accuracies vs Hidden units | Hidden Units: 5, 10, 15, 20, 25



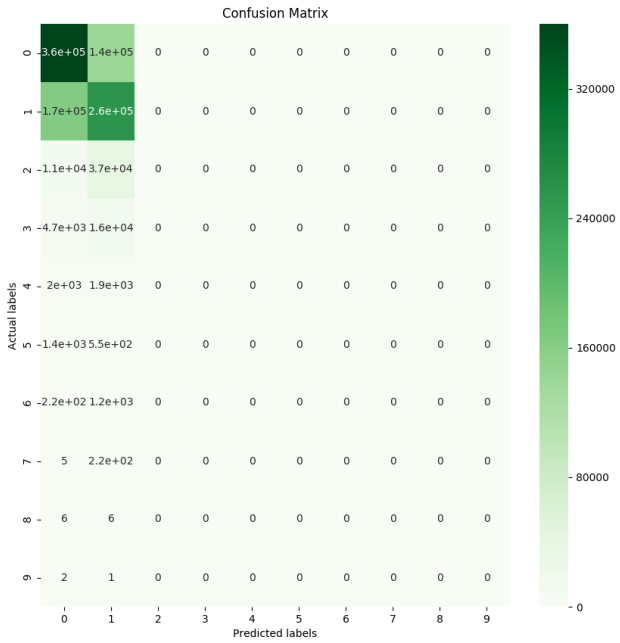
Time taken vs Hidden units | Hidden Units: 5, 10, 15, 20, 25



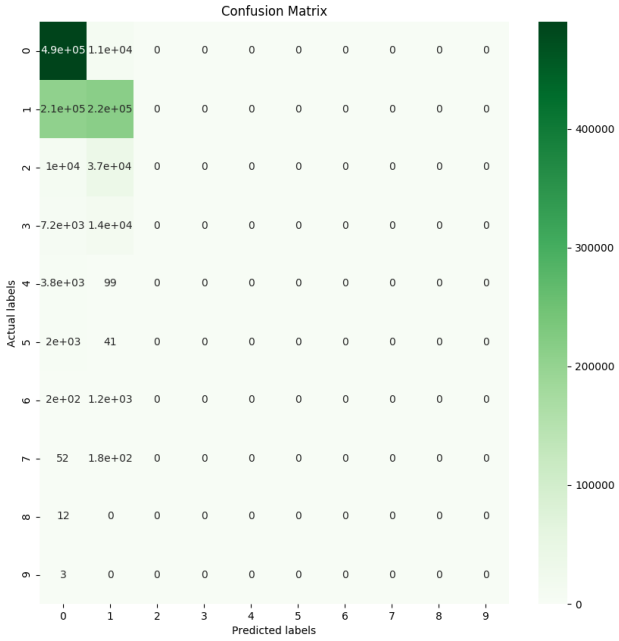
```
[#]------------PART-F--D------------[#]
[>] Neural Network:
[>] Shape:  [85, 5, 5, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100
[.] Tolerance:  0.0001
[.] Activation Function for Hidden Layers:  relu


[*] Time taken: 8.68295168876648
[*] Train Accuracy: 0.649140343862455
[*] Test Accuracy: 0.6175766175766175
[>] Neural Network:
[>] Shape:  [85, 10, 10, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100
[.] Tolerance:  0.0001
[.] Activation Function for Hidden Layers:  relu


[*] Time taken: 13.479251861572266
[*] Train Accuracy: 0.8699320271891243
[*] Test Accuracy: 0.8597778597778598

[>] Neural Network:
[>] Shape:  [85, 15, 15, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100
[.] Tolerance:  0.0001
[.] Activation Function for Hidden Layers:  relu

[*] Time taken: 52.231123208999634
[*] Train Accuracy: 0.7232706917233107
[*] Test Accuracy: 0.7052217052217052

[>] Neural Network:
[>] Shape:  [85, 20, 20, 10]
[.] Learning Rate:  0.1
[.] Batch Size:  100
[.] Tolerance:  0.0001
[.] Activation Function for Hidden Layers:  relu

[*] Time taken: 30.649317264556885
[*] Train Accuracy: 0.931547381047581
[*] Test Accuracy: 0.9179399179399179
```

```
[>] Neural Network:
[>] Shape:  [85, 25, 25, 10]
[.] Learning Rate:  0.01
[.] Batch Size:  100
[.] Tolerance:  0.0001
[.] Activation Function for Hidden Layers:  relu

[*] Time taken: 22.03646492958069
[*] Train Accuracy: 0.9824470211915234
[*] Test Accuracy: 0.9637469637469638
```

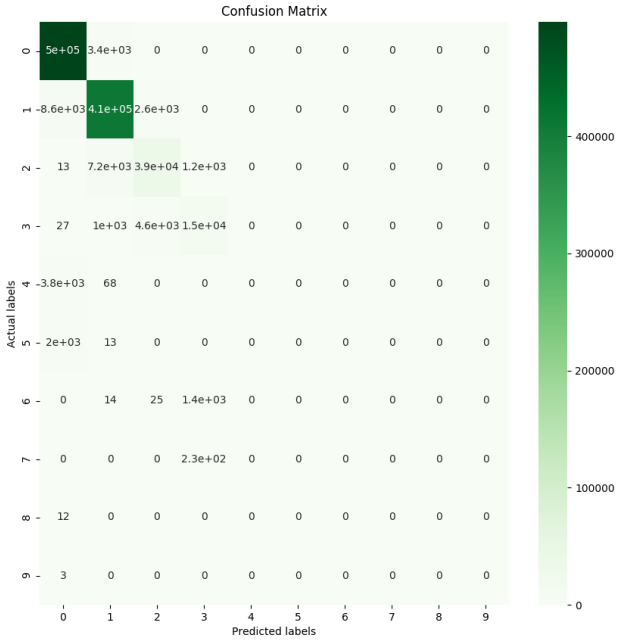
Confusion Matrix | Hidden Units: 5



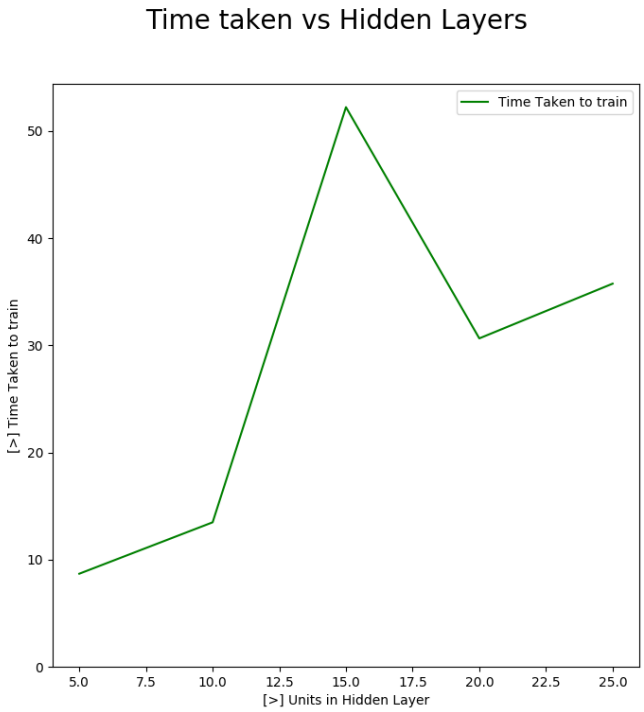
Confusion Matrix | Hidden Units: 15



Confusion Matrix | Hidden Units: 25



Time taken vs Hidden units | Hidden Units: [5,5] [10,10] [15,15] [20,20] [25,25]



2.2 Comparisons

Since, in each part from c-f we have to do comparisons, I'll do all the compares here (in order):

- For both part c and d, the logic is same: As we increase the hidden layer, the neural network trains better, thus increasing the accuracy, and hence better performing confusion matrix. This applies to the number of hidden layers too.
- It should also be kept in mind that, if the data-set is not very large, increasing the hidden layers or hidden units can cause overfitting, or in other words training too well. This is always a problem, and implementation specific.
- As the loss decreases, the learning rate decreases, this causes the model using adaptive learning to saturate at the end, as can be seen in the draw graphs. The weights converge to a value, because after a while learning rate is so small, weights doesn't change anymore.
- This causes it to converges, sometimes, before it actually should. Causing a little bit decrease in accuracy, and thus confusion matrix not so generalized.
- ReLU trains the model very fast in comparison to sigmoid, as it had to do a significantly smaller number of calculations, to be fair, it doesn't even do calculation, only comparisons.
- The confusion matrix is spread out, and the model performs around 96% on test data, which is a good accuracy.