## Analysis - Assignment 4

## aarigela-singh74-snehchem

# Nearest neighbor

To train, run below command:

**python orient.py train train-data.txt nearest**

To test, run below command:

**python orient.py test test-data.txt nearest**

Nearest neighbor is a special case of k nearest neighbor where k=1.

We analyzed the accuracy for various values of k instead of just taking the nearest neighbor, we found that setting the k=25 gives the maximum accuracy which is 71.04% but for k=1, we get the accuracy as 67.23%.

|  |  |
| --- | --- |
| k | Accuracy |
| 1 | 67.23 |
| 5 | 69.14 |
| 10 | 70.3 |
| 15 | 70.20 |
| 20 | 70.5 |
| 25 | 71.04 |
| 30 | 70.20 |
| 35 | 70.41 |
| 40 | 70.2 |

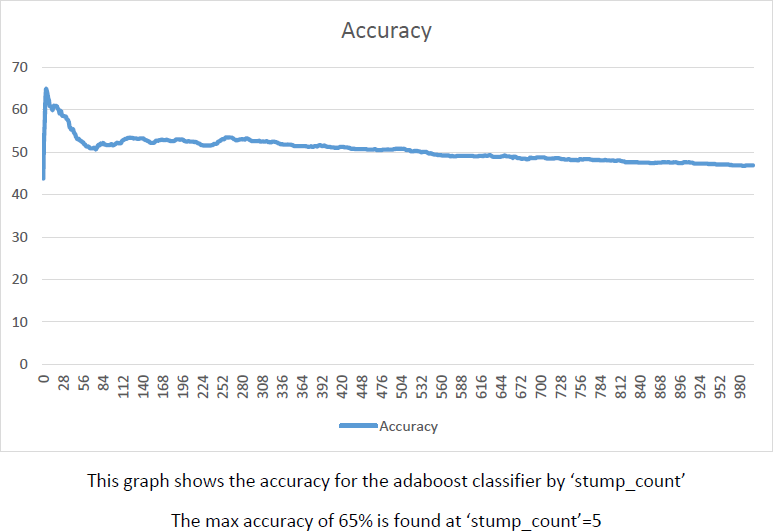
# Adaboost

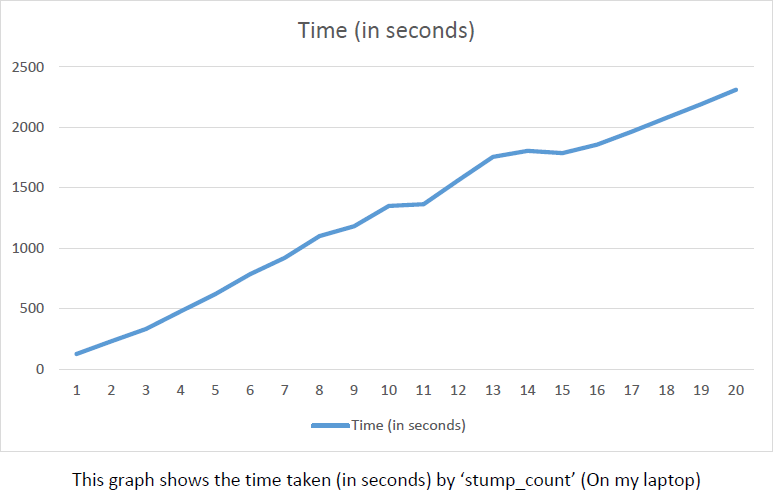
**Run python orient.py train train-data.txt adaboost\_model.pickle adaboost [stump\_count]**

**Run python orient.py test test-data.txt adaboost\_model.pickle adaboost [stump\_count]**

We ran the adaboost method with different stump counts and found that initially the classifier is weak, then as we add more stumps the accuracy increases, but after a certain value (in our case 5) it began to decrease as the model began to overfit.

The plot of percentage accuracy vs stump count is shown below:



The plot of Time taken (in seconds) vs stump count is shown below:

# Neural network classification

Design Decisions:

The neural network built here is a fully connected feed forward neural network with one hidden layer. As it was my first ever implementation of neural networks I decided to a network with a single hidden layer and sigmoid as the activation function for the hidden and output layers.

I initially started out with traditional gradient descent since it seemed simpler to do matrix operations on the whole data at once and update the weights in each iteration but that took quite some time to converge. Hence, I switched to stochastic gradient descent where I iterate over each example and update the weights immediately by calculating the error for that sample.

Also, since the training data has around 37K examples, it reaches optimal values in a single iteration.

I tried multiple permutations and combinations for the parameters – learning rate, number of nodes in hidden layer

For the complete training data –

|  |  |  |  |
| --- | --- | --- | --- |
| number of nodes in hidden layer | learning rate | Accuracy | Training Time |
| 35 | 0.5 | 72.32 | 174s |
|  | 0.4 | 72.53 | 161s |
|  | 0.0375 | 71.04 | 161s |
|  | 0.015 | 70.62 | 161s |
| 5 | 0.6 | 70.41 | 28s |
|  | 0.2 | 71.47 | 25s |
|  | 0.0475 | 72.11 | 26s |

Our best model gives an accuracy of 72.53% on the test data, parameter values highlighted above

Even with around 2/3 of the training data we get the same accuracy for 35 hidden nodes and learning rate of 0.4, but the training time is drops to 61sec. Hence this the best model we are using.

# 4. Analysis on Partial Data

If we run our models on partial data we see that some of the images are misclassified for example this image below:

the image with id:10351347465 was misclassified by Neural Network in the whole data and also by Adaboost but not by nearest neighbors, but after running on partial data Adaboost and Nearest neighbor could not classify the image correctly but neural network could do it. If we train the models on some other set of data and then again run them we see that this image classification was similar as that on whole. So we can say that changing the training data can lead to misclassification or correct classification of some images, but again if we change the training data it may lead to erratic results.

Also, it seemed that the neural network made mistakes with (0,180) and (90,270) output values.

# 

# 5. Best

Neural network is our best model and our implementation is much efficient and does not take much time to train.

It is working at an accuracy of 72.5% higher than other classifiers. In general after trying above analysis on different sets of data Neural Network could better generalize the model and performed better than others and after that K nearest neighbor performed better than adaboost, hence we can say that adaboost tend to overfit the data. Hence we see that Neural network was one of our best model.