Assignment 5 – Report

1) KNN

Data pre-processing:-

Normalization did not improve accuracy but took lot of time to run,so no preprocessing done.

Below table has runtime and accuracy for different values of K.

K	Accuracy	Runtime (mins)
1	67.23	20.54
50	71.26	25.41
100	70.30	40.51
150	70.30	41.10
200	7.26	40.40

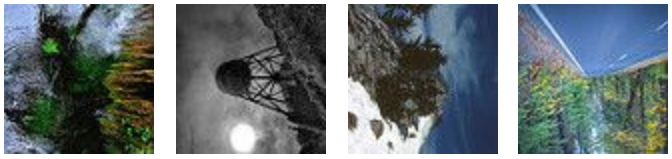
Below table has runtime, accuracy for varying amount of probability of choosing each record in test.

Probability	K	Accuracy	Runtime (mins)
0.25	200	68.92	10.34
0.5	200	70.83	20.55
1	200	70.26	40.40

Please note that there was a drastic increase in runtime just before submission time(could be because lot of load on burrow. So I had to rerun and take runtimes for every K.

Sample Images classified correctly:

Sample Images classified incorrectly:



Observations: k > 25 works giving less error I would recommend working with K = 200; keeping in mind that time to run KNN with K as 200 takes time similar to k as 1.

2) Adaboost

For each decision stump, best pair of features are chosen from 100 pairs to reduce computation time. One vs all classifiers are implemented for multi class prediction among classes of 0, 90, 180, 270. In each decision stump, best attribute is determined based on the accuracy, number of images are correctly classified.

Number of stumps	Accuracy (%)	Running Time (sec)
1	61	43.55
2	65.1	
3	65.3	142.5
5	70	267.96
7	72.8	383.77
9	73.49	477.92
11	74.12	647.81
17	74.12	969.85

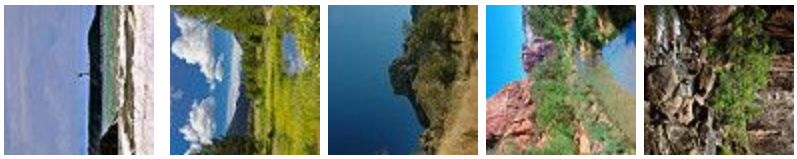
It is observed that, in each run with different number of stumps, most of the 270 orientation images are failed to be predicted correctly.
In most circumstances, the 270 orientation is predicted as 180.

Percent of train data set	Number of train rows processes	Running Time(sec)	Accuracy(%)

50	18488	325	73.9
80	29580	630	73.7
100	36976	647.81	74.12

Sample Images classified correctly:

Sample Images classified incorrectly:



3) Neural Network

A three layer (one hidden layer) feed forward neural network featuring **Stochastic Gradient Descent** is implemented in the code submitted.

Data pre-processing: The data is normalized by dividing each data point by 255 followed by subtraction with the mean and finally dividing by the standard deviation. It was clearly observed that without this preprocessing step, **the model was not able to perform** (was giving random accuracy) under any circumstances.

Parameters experimented with:

Step sizes: 0.1, 0.01, 0.0001
Activation functions: Sigmoid, tanh
No. of Hidden layer neurons: 10, 200, 600
Epochs: 1, 20, 50

Observations:

It was observed that the number of epochs did not enhance the accuracy by a noticeable margin. Hence, the following results are shown for one epoch.

Activation Function	Step size	Number of Hidden Layer Neurons	Running Time(sec)	Accuracy(%)

Sigmoid	0.1	10	14.72200012	67.126193
		200	51.89700007	65.00530223
		600	186.4519999	30.54082715
	0.01	10	14.66200018	64.05090138
		200	53.99899983	**71.26193001**
		600	225.901	68.83775186
	0.0001	10	14.50999999	44.22057264
		200	54.28299999	58.53658537
		600	185.635	61.82396607

Accuracies obtained with Sigmoid activation function

Activation Function	Step size	Number of Hidden Layer Neurons	Running Time(sec)	Accuracy(%)
tanh	0.1	10	13.59500003	56.81018028
		200	42.16799998	38.38812301
		600	154.8970001	50.26511135
	0.01	10	13.94499993	63.30858961
		200	42.95900011	64.58112407
		600	167.5339999	62.88441145
	0.0001	10	13.44700003	32.34358431
		200	43.27499986	43.90243902
		600	158.342	35.41887593

Accuracies obtained with tanh activation function

- Sigmoid activation function works better than tanh
- The error increases for very low or very high number of hidden layer neurons
- 0.01 was found to be the most effective in terms of accuracy
- The running time increases with the increase in the number of hidden layer neurons
- The accuracy for each run varies by approximately $\pm$ 5%.

The code was also run with random splits of the train set of varying percentages. Some interesting observations were made as follows:

Percent of train data set	Number of train rows processes	Running Time(sec)	Accuracy(%)
0.5	184	9.460000038	43.69034995
2	739	9.905999899	60.65747614
10	3697	13.41300011	65.42948038
20	7395	18.37800002	68.82290562
30	11092	22.63999987	69.67126193
50	18488	30.99000001	69.88335101
80	29580	44.70600009	69.95917285
100	36976	63.12800002	72.11028632

It can be seen that with just 20% of randomly selected training data, almost 69% accuracy is achieved which is very near to the accuracy obtained training on the entire train set. Moreover, there is a huge difference between the time taken to train with 20% data and 100% data.

Conclusion:

As it can be observed from the above tables that for a three layer feed forward neural network implementing Stochastic Gradient Descent, the following configuration is recommended:

Activation function: **Sigmoid**
Step size: **0.01**
No. of Hidden layer neurons: **200**
Accuracy obtained: **71.26193001%**

Sample Images classified correctly:




Sample Images classified incorrectly:




Best classifier:

Among all of the above classifiers, Adaboost with decision stumps performed better on the given training and test data sets. Accuracy of 74.12 % is achieved for the 11 stumps.