Assignment Report

Neural Network Tests:

Hidden Nodes	Passes	Learning Rates	Number of Examples Used	Train Accur1acy	Test Accuracy
97	200	0.91	100	10587	
97	400	0.91	100	15633	
97	450	0.91	100	15199	
98	450	0.91	100	15735	
98	550	0.91	100	21248	
98	700	0.915	100	22531	
98	850	0.91	100	23292	
97	2650	0.95	100	25141	
97	3050	0.99	100	25161	
97	4050	0.89	100	25173	
97	4050	0.91	600		625
97	5050	0.91	600		624
97	2050	0.91	600		630
97	1050	0.91	600		632
97	1050	0.91	2000		615
97	1250	0.91	2000		614
97	250	0.91	5000		628
97	200	0.91	7000		631
97	200	0.91	8000		633
97	150	0.91	10000		633
97	50	0.91	10000		631
97	35	0.91	20000		632
97	35	0.91	30000		633
97	2	0.91	All		576
97	3	0.91	All		602
50	9	0.99	All		677
99	9	0.99	All		678
76	100	0.99	16976		697
75	100	0.99	16976		698

For testing parameters, I started increasing the training examples (for speed and error checking purposes) on 98/97 hidden nodes with passes up to 5050. Once we achieved a commendable training fit (97/4050), we started evaluating various parameters on the test set. The number of testing examples that were correctly classified is given in the table above. In the end, after testing for several parameters and combinations, we achieved a best error rate of 74.019% on the test set.

We haven't tried any new activation functions. We tried to stick to the basics and keep it simple. Therefore, we took the sigmoid activation functions for both the layers (hidden and output). We used a stochastic gradient descent (first tried on traditional gradient descent, but it was taken very long time for obvious reasons), and we re-iterated the data for some 'n' passes. Once we achieved a decent fit, we again tried increasing and decreasing the passes (+/- 2). In the end, we settled for a fit of 75% on the train set, with 86 nodes in the hidden layer passed 9 times over and over again. After this, we reached the accuracy above.

The learning rate was tricky. We started choosing like 0.001 and 0.01 since those were most preferred in machine learning. But these didn't go well for our technique. Therefore, we started to increase it. At a point, a learning rate of 2.99 gave a pretty decent error rate. But, after going through various other values, even though the value was still higher, we adjusted it to 0.92.
The best neural network model was done with 86 hidden nodes, iterated wholly 9 times, with a learning rate of 0.99.

The best accuracy was achieved when we excluded 20000 examples from the training set, and uses 100 passes. Only for this method, the program will run for 100 passes through the network. For all the others, the dataset is passed through the network on 9 times rather than 100. This method gave us the best accuracy on the test set of 74.019%. 698 examples out of 943 were correctly classified by the neural network.

Misclassified Example:

True 90 Predicted 180

From the misclassified example, we can see that when the true label is 90, it predicts it as 180. But it you notice carefully, the total composition of the image is not strictly altered when rotated by an angle of 90. I think this could be reason for wrong classification, because pixels at particular points could have been different.

Correctly Classified Example:

True 0 Predicted 0 Change if rotated to 9

This was correctly classified because if it were rotated, the cliff and the background would have been inverted, which would cause a huge change in pixel distributions.

Nearest Neighbors:

The parameters choosing mostly on k, and some other exclusions. Since this was taking time, for each iteration about 5 seconds, we limited the number of tests on this one. Once we hit the 70% barrier, we tried a few more values, and ended it.

K – Value	Training Examples Taken	Pixels	Accuracy
30	All	All	70.4
3	All	All	68.3
10	All	All	71.1
50	All	All	70.4
50	26976	All	71.5
10	26976	Green	58.6
10	14976	Red	68.4
30	14976	Red	71.5 (Best)

These are some of the variations that were considered, and choosing particular pixels also plays an important part. After some tests (obviously less than neural networks), we have decided to use the final model, that used only 14976 (20000 removed) as the training data. This variation might be because if there are more training examples, some of the distances may be ‘hallucinated’, like in neural networks, if we try to encode on too many passes, the network can hallucinate (or) produce false patterns that were not to be in data set.

Adaboost –

The parameters chosen were number of decision stumps. The more the number of stumps the more accurate the algorithm predicts he orientation of image.

The accuracy may vary from each run as the points selected for the comparison were randomly generated. The accuracy might converge if used large number of decision stumps for example greater than 10,000 so that the random generation will be fully distributed.

# Decision Stumps	Training examples taken	Testing examples taken	Pixel point	Accuracy
5	All	All	Two random points were chosen	38%
10	All	All	Two random points were chosen	43.2%
50	All	All	Two random points were chosen	48.2%
100	All	All	Two random points were chosen	54.43%
300	All	All	Two random points were chosen	54.2%
500	All	All	Two random points were chosen	55.1%