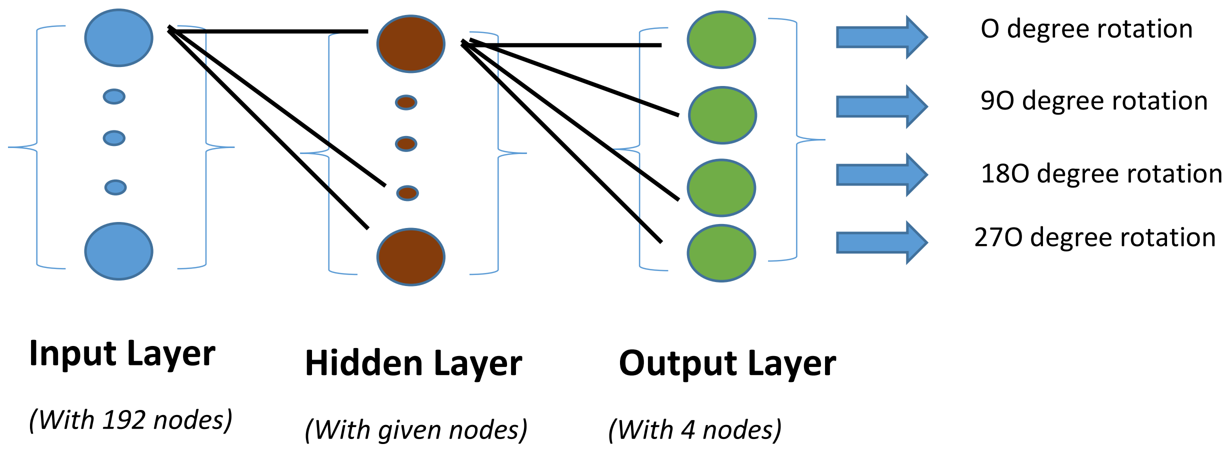
Assignment 6: Machine learning

Analysis and improvement

Neural networks:

Total number of layers: 3



Activation functions used: Sigmoid, Tanh

Analysis of results obtained:

Hidden Nodes	Activation Function	Training Iterations	Learning Rate	Time Required (Seconds)	Back propagation method	Training Data set size	Accuracy
20	Sigmoid	3	0.1	312	Stochastic	100%	73.81
20	Sigmoid	2	0.2	157	Stochastic	100%	71.58
20	Sigmoid	1	0.1	76	Stochastic	100%	70.81
384	Sigmoid	5	0.5	1600	Stochastic	100%	71.56
20	Sigmoid	3	0.5	309	Stochastic	100%	71.37
20	Tanh	1	0.1	200	Stochastic	100%	69.56
20	Tanh	15	0.1	2500	Stochastic	100%	73.2
20	Sigmoid	5	0.1	1300	Batch	100%	65.23
20	Sigmoid	1	0.1	35	Stochastic	50%	70.31
20	Sigmoid	3	0.1	105	Stochastic	50%	72.22
20	Sigmoid	10	0.1	379	Stochastic	50%	73.38
20	Sigmoid	10	0.1	318	Stochastic	25%	70.38

Recommendation:

Classifier: Neural network

Gradient Descent function to use: stochastic

Weight initialization: Normal distribution between -0.01 to 0.01

Hidden Nodes # 20

Learning Rate: 0.1

Inferences:

1. Lower learning rate gives better accuracy with more iterations
2. Tanh performs better with more iterations
3. Network gives best performance with 20 hidden nodes, with more nodes time increases and accuracy remains constant
4. Overall performance largely depends on how well weights gets initialized if training iterations are very small
5. For half training data the accuracy is almost similar and even increases with number of iterations. That shows the data is **not over fitted**.

About the best algorithm:

The best algorithm is neural network with model trained with recommended configuration from above.

It gives accuracy of 73.81 over current test data set.

K-Nearest Neighbor

We have used two approaches to solve the image classification using K-Nearest Neighbor problem.

1. Exhaustive K-Nearest Neighbor
2. K-Nearest Neighbor with K-Means clustering

1. Exhaustive K-Nearest Neighbor-

Algorithm –

1. This is traditional, naïve approach for K-Nearest Neighbor, where-in there is no learning in the training data set and it is used just to store the training examples.
2. Each of the test data is compared with every training data
3. K-closest neighbors based on distances of from the test data
4. Distances are calculated using
    a. Manhattan Distances
    b. Euclidean Distance
5. Majority voting is done amongst the K-closest neighbors
6. If there's a tie:
    a. Decrease the value of K

    b. Pick the Kth (decreased by 1) closest neighbors
    c. Do a majority vote again, if still a tie, go to step 6.a and repeat, if not a tie, return the winner

7. Assign the orientation of the training sample
8. Do for every test data

2. K-Nearest Neighbor with K-Means clustering

Algorithm –

1. We initially cluster the training data using unsupervised learning algorithm K-Means for a given number of clusters (fixed as 10) and for given number iterations (fixed as 10).
2. This generates centroids from each clusters
3. Each test example compares the distance from the centroid and finds L (fixed as 3) closest clusters centroid based on the distances of
    a. Manhattan
    b. Euclidian
4. For each of the closest centroids, compare the distance from training samples in the cluster
5. This reduces the number of comparisons with the training example,
6. As number of clusters are increased there's a better distribution in the training examplars and lesser comparison.
7. Pick K closest neighbors
8. Do steps similar to exhaustive - <re-iterating>
9. Majority voting is done amongst the K-closest neighbors
10. If there's a tie:
    a. Decrease the value of K
    b. Pick the Kth (decreased by 1) closest neighbors
    c. Do a majority vote again, if still a tie, go to step 6.a and repeat, if not a tie, return the winner

11. Assign the orientation of the training sample

Design decisions/Assumptions –

1. We picked Manhattan and Euclidian distances to calculate between the examplars
2. For K-Nearest with K-Means, we have used different iterations and number of clusters
3. To improve the running time, we used the following functions of Numpy
    a. Array
    b. Abs
    c. Sum
    d. Liang.norm -> Euclidian
4. We have not used for Numpy in the run.

Feature Space Reduction –

1. Instead of using all the features in the image, vector of 192 (8 X 8 X 3), we tried different formulas for reducing the feature space
2. Formula used –
    a. HSL –
        i. Formula

        brightness = sqrt(.299 R^2 + .587 G^2 + .114 B^2)

        ii. Link - http://alienryderflex.com/hsp.html
        iii. R, G, B value of each pixel, is used as brightness

    b. Luminance (Conversion to GreyScale) –
        i. Formula

        0.3 R + 0.59 G + 0.11 B

3. None of the reductions gave a good increase in accuracy
4. So we used the same number of features

K-Nearest Neighbor for 100% data

Type of Algorithm	K	Distance	Clusters	Iterations	Accuracy	Time (approx mins)
Exhaustive	5	Euclidian	NA	NA	67	40
Exhaustive	21	Euclidian	NA	NA	69	40
Exhaustive	70	Euclidian	NA	NA	71.04	40
Exhaustive	80	Euclidian	NA	NA	70	45
Exhaustive	21	Manhattan	NA	NA	70.7	20
Exhaustive	70	Manhattan	NA	NA	70.94	22
Exhaustive with K-Means	21	Manhattan	10	10	70.57	13
Exhaustive with K-Means	70	Manhattan	10	10	70.37	15
Exhaustive with K-Means	21	Euclidian	10	10	70.37	15
Exhaustive with K-Means	73	Euclidian	10	10	70.37	15
Exhaustive with K-Means	73	Euclidian	8	10	69.7	14
Exhaustive with K-Means	73	Euclidian	4	10	68.9	12

Knn for 50% data gave accuracy of 69.78%

Few Images classified wrong:

test/10196604813.jpg

test/10351347465.jpg

test/10352491496.jpg

test/10484444553.jpg

test/10577249185.jpg

test/10684428096.jpg

test/108172840.jpg

test/11057679623.jpg

test/112713406.jpg

test/11418669873.jpg

test/1160014608.jpg

test/12115177323.jpg

Few Images classified correct:

train/10008667845.jpg

train/10017728034.jpg

train/10034256894.jpg

train/4955645147.jpg

Similarity between incorrect images:

Most of the images I found had symmetric colors. The rgb values hence mostly look alike even after any rotation. That may result in incorrect classification of these images.

Future scope:

More training and error based convergence. Use local sensing hashing for knn.