Assignment 5

Step1-K-Nearest Neighbors

The program runs with the following command line

python orient.py train-data.txt test-data.txt nearest 11

The output for this instruction is

Accuracy: 70.6256627784%

Confusion_Matrix

168 19 37 15

22 168 11 23

42 23 162 9

20 38 18 168

The Output file generated is **nearest_output.txt** (uploaded in github)

Similarly for k=40

k=40

The output obtain is as below:

Accuracy: 70.3075291622%

Confusion_Matrix

170 15 35 19

20 163 14 27

39 21 159 17

22 32 19 171

Step of k-nearest neighbor algorithm:

1. First, the training data is stored in the form of list with its vectors. Thus, converting each training data in the form of a list of each list.
2. For the test data, heuristic function is applied by using Euclidean distance. Also, nearest k attribute is selected which is entered by the user.
3. Then the class label are selected which are assigned to the test data.
4. Finally, accuracy is calculated using total number of correct observations and test cases.

The method selection ‘nearest’ performs this algorithm. But the average time taken by the code for execution is 25-30 minutes because it is comparing each test data with each training data. The approximate accuracy obtained is around 71% when checked for higher k values.

Improvement in k-nearest neighbors algorithm

As mentioned above, the code takes long time for execution and so we tried to reduce this time to 15 minutes approx. This improvement was done at few parameters like the intensity value was used instead for RGB values. Thus, comparison happens for only 64 dimension vector. This resulted into reduce time, but the accuracy also reduced by 5%. The following is the output obtained from method nearest_modified:

Confusion_Matrix

Accuracy: 62.5662778367%

158 20 40 21

23 141 14 46

52 25 140 19

23 53 17 151

Step 3-Neural Networks

The program runs with the following command line

python orient.py train-data.txt test-data.txt nnet 30

The output for this instruction is

Weight Initialized for Neural Network

Input Data read

Train Accuracy: 69.5153613154

Matrix:

6312	974	1188	770
712	6306	1182	1044
1028	615	6459	1142
905	1003	709	6627

Test Accuracy: 72.0042417815

Matrix:

196	15	21	12
27	161	21	27
34	21	160	9
35	29	13	162

Matrix displays each row and column for 270 180 90 0 respectively.

The Outputfile generated is **nnet_output.txt**(uploaded in github)

Steps for Neural Networks:

1. Initially weight of the input data is calculated coming to the hidden node and applied sigmoid function to it.
2. At the output layer, sum of the weight at the hidden layer and output layer is calculated taking 4 output nodes.
3. Lastly, error in the neuron is looked upon by function Error=Output(1-Output)(Target-Output)
4. Weights are updated by calculating it with error and initial weights.
5. These weight are normalized by dividing the input data with 255
6. Lastly, back propagation from the output layer is applied and through the weights in the hidden layer errors are obtained which are further calculated using error equation.

Here, to obtain optimal results weights are changed ranging -0.1 to 0.1.

Step 4- Best Neural Network Algorithm

The program runs with the following command line

python orient.py train-data.txt test-data.txt best 30

The output for this instruction is

Weight Initialized for Neural Network Best algorithm

Input Data read

Train Accuracy: 73.7045651233

Output_Matrix:

6759	829	1087	569
539	6851	751	1103
1079	590	6814	761
708	1190	517	6829

Test Accuracy: 73.3828207847

Output_Matrix:

183	15	29	17
16	165	22	33
31	15	165	13
12	31	17	179

Matrix each row and column corresponds to 270 180 90 0 respectively

Output file= **best_output.txt**(uploaded in github)

Similarly, the second output

Weight Initialized for Neural Network Best algorithm

Input Data read

Train Accuracy: 73.5341843358

Output_Matrix:

6730	848	1088	578
555	6830	749	1110
1058	586	6812	788
661	1217	548	6818

Test Accuracy: 73.4888653234

Output_Matrix:

188	15	23	18
18	170	17	31
34	23	156	11
16	32	12	179

Matrix each row and column corresponds to 270 180 90 0 respectively

Implementation

For the best algorithm individual weights of the hidden layer and output layer are stored in files weight_layer1.txt and weight_layer2.txt.

We have taken learning rate as 0.6 and hidden layer nodes as 30.

Here, the accuracy obtained is around 74%, so it is advised to work with the best Neural Network classifier.

The following is the graph showing the accuracy of the algorithm at different values of training and test data.