Assignment 6 --Orient.py Analysis

Note: The Neural networks backpropagation program is scheduled to run 800 iterations currently. Above 800 iterations, memory error is coming because of overflow. But, at 800 iterations, the error do not converge and output is not very accurate. To check for higher accuracy on smaller training data, please increase the 'lnMax' value at line no 174 in program to 10000 or 20000.

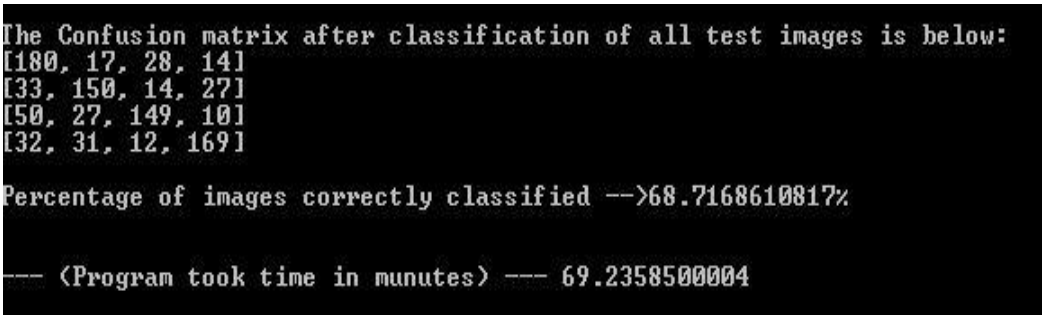
K-nearest neighbors:

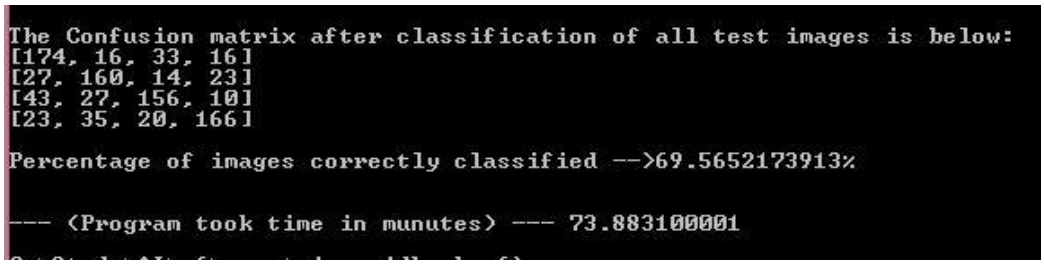
1) For K-nearest Neighbors, we are calculating "Euclidean distance" between each test image with all the training images and taking specified k minimums amongst them which are called the k-neighbors of that test image.

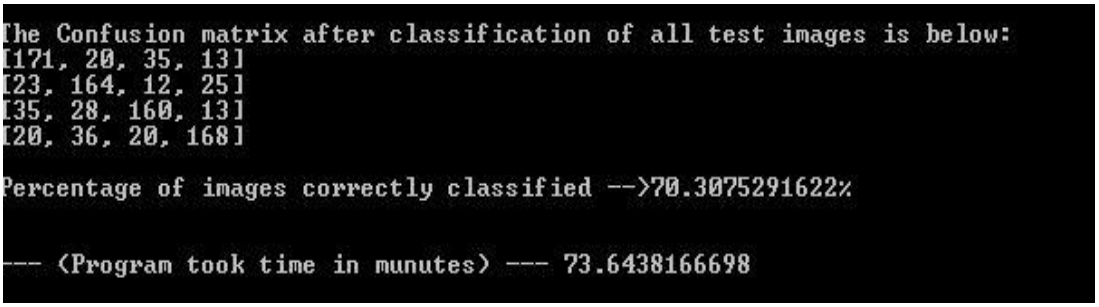
Below are some analysis results changing the parameters:

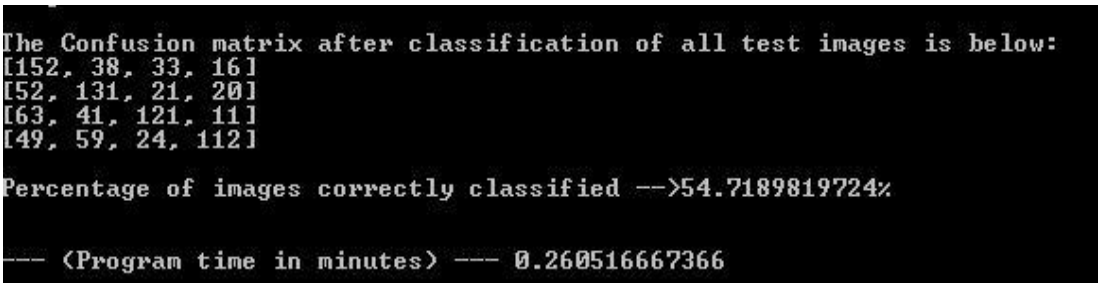
K	Training input size	Test input size	Accuracy (%)	Response time (min)
3	36976 (for full train-data.txt and test-data.txt)	943	68.71	69.23
5	36976	943	69.56	73.88
10	36976	943	70.30	73.64
5	500	943	54.71	0.26

Some of the sample screenshots of accuracy, running time and confusion matrix above: (**In respective order from top**):









Inferences:

a) The results become more accurate with the increasing size of k, evident from row 1, 2, 3 in table but can become more prone to overfitting.
b) Results become less accurate, obviously with less training data. Less knowledge corresponds to less deep learning.

Neural Network with Backpropagation:

The activation function used here is ***1/(1+e^(-x)).***

Below are some analysis results changing the parameters:

Iterations	Number of Hidden Layers	Training data size	Test data size	Accuracy	Response time(mins)
20000	2	101	23	43.47%	0.886
30000	5	101	943	50.47%	1.45
50000	10	101	943	50.68%	2.54
800	3	500	943	31.60%	3.22
800	2	36976	943	25.02%	9.64

Screenshots of above in order of the tuples:

```
The Confusion matrix after classification of all test images is below:
[7, 0, 1, 0]
[4, 0, 0, 1]
[3, 1, 2, 0]
[2, 1, 0, 1]

 Percentage of correct prediction of orientation of test images--> 43.4782608696
%

--- (Program time in minutes) --- 0.886400000254
```

```
The Confusion matrix after classification of all test images is below:
[224, 1, 13, 1]
[144, 72, 4, 4]
[124, 6, 106, 0]
[156, 6, 8, 74]

 Percentage of correct prediction of orientation of test images--> 50.4772004242
%

--- (Program time in minutes) --- 1.45863333146
```

```
The Confusion matrix after classification of all test images is below:
[68, 160, 4, 7]
[3, 211, 2, 8]
[3, 149, 82, 2]
[2, 123, 2, 117]

 Percentage of correct prediction of orientation of test images--> 50.6892895016
%

--- (Program time in minutes) --- 2.54298333327
```

```
The Confusion matrix after classification of all test images is below:
[236, 0, 3, 0]
[223, 0, 1, 0]
[174, 0, 62, 0]
[244, 0, 0, 0]

 Percentage of correct prediction of orientation of test images--> 31.6012725345
%
```

```
The Confusion matrix after classification of all test images is below:
[0, 0, 239, 0]
[0, 0, 224, 0]
[0, 0, 236, 0]
[0, 0, 244, 0]

 Percentage of correct prediction of orientation of test images--> 25.0265111347
%

--- (Program time in minutes) --- 9.64764999946
```

Inferences:

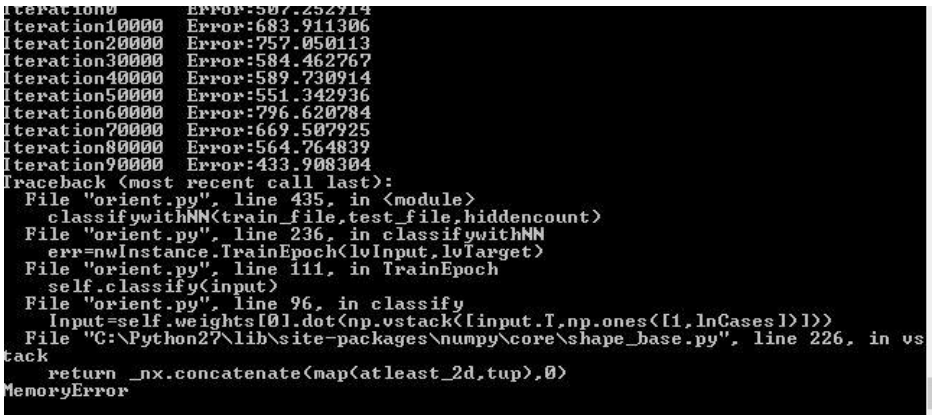
1) We can see, with increasing number of hidden layers count, the accuracy is increasing, evident from rows 1, 2, 3 of above table.
2) With increasing number of iterations, the result has improved, evident from 1, 2.

3) While we have decreased the iterations, though increasing the training input size, the accuracy drops drastically.

So, for neural networks, more number of hidden layers are recommended though it can lead to 'overfitting' to training data. Also, more the number of iterations of backpropagation, the errors will converge more towards lesser value and hence better deep learning.

Problem faced:

Memory Error: If we give iterations for backpropagation more than 10000 for the full train-data.txt and test-data.txt, then the program gives memory overflow exception because of consuming too much memory through List. So, we couldn't test the program with iterations more than 800.



We'll recommend KNN classifier to a potential client, KNN algorithm with k around 10 with Euclidean distance as evaluating distance function.

Best Algorithm:

For the best algorithm, we have used our KNN network with number of neighbors=10, for two reasons:

1) Looking at result, we feel that though KNN can be less accurate than NN, but KNN is more stable and gives consistent output through varied parameters. Also, sometimes neural networks just cannot learn because of so many evaluation parameters involved like iterations, activation function, number of hidden layers, scaling etc.
2) Since, we are able to test the program with neural networks with iterations<800 only because of memory overflow exception. But, for that less number of iterations, error do not converge to a lesser value. Hence we cannot get the fair accuracy.

Result of Best algorithm:

```
The Confusion matrix after classification of all test images is below:
[171, 20, 35, 13]
[23, 164, 12, 25]
[35, 28, 160, 13]
[20, 36, 20, 168]

Percentage of images correctly classified -->70.3075291622%


--- (Program took time in munutes) --- 73.6438166698
```

Sample Images which were classified correctly:

test/10008707066.jpg 0

test/10099910984.jpg 0

test/10107730656.jpg 180

test/10164298814.jpg 0

test/10304005245.jpg 90

test/10313218445.jpg 90

test/10795283303.jpg 180

test/11185093106.jpg 180

Sample Images classified incorrectly:

test/10161556064.jpg 90 original: 270

test/10351347465.jpg 180 original: 270

test/10352491496.jpg 180 original: 90

test/10484444553.jpg 0 original: 180

test/112713406.jpg 180 original: 0

Inference:

One pattern we can observe here is the algorithm generally incorrectly classifies those images which are exactly inverted (opposite in orientation).

Like 180-0, 90-270

Assignment references and sources:

1) https://www.youtube.com/watch?v=XqRUHEeiyCs&list=PLRyu4ecIE9tibdzuhJr94uQeKnOFkkbq6

2) http://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/

3) http://www.bogotobogo.com/python/python_Neural_Networks_Backpropagation_for_XOR_using_one_hidden_layer.php

Also, Discussed the assignment with classmate **Deepika Bajpai** (**dbajpai**) on Read data functions, nearest neighbor implementations.