Homework 3.2 (60 points)

Machine Learning

1 Neural Networks, Gradient Boosting and SVMs [30 points]

• For this part, you will use scikit learn.

• Download the MNIST dataset available at http://yann.lecun.com/exdb/mnist/. The dataset has a training set of 60,000 examples, and a test set of 10,000 examples where the digits have been centered inside 28x28 pixel images. You can also use scikit-learn to download and rescale the dataset using the following code:

• Use the SVM classifier in scikit learn and try different kernels and values of penalty parameter. Important: Depending on your computer hardware, you may have to carefully select the parameters (see the documentation on scikit learn for details) in order to speed up the computation. Report the error rate for at least 10 parameter settings that you tried (see how it is reported on http://yann.lecun.com/exdb/mnist/). Make sure to precisely describe the parameters used so that your results are reproducible.

• Use the MLPClassifier in scikit learn and try different architectures, gradient descent schemes, etc. Depending on your computer hardware, you may have to carefully select the parameters of MLPClassifier in order to speed up the computation. Report the error rate for at least 10 parameters that you tried. Make sure to precisely describe the parameters used so that your results are reproducible.

• Use the GradientBoostingClassifier in scikit learn and try different parameters (see the documentation for details). Again depending on your computer hardware, you may have to carefully select the parameters in order to speed up the computation. Report the error rate for at least 10 parameters that you tried. Make sure to precisely describe the parameters used so that your results are reproducible.

• What is the best error rate you were able to reach for each of the three classifiers?

For SVM the minimum error rate = 5.54% (SVC with rbf kernel of degree 0 and C 1)

For Neural Network the minimum error rate = 2.32% (MLPC with activation relu and solver lbfgs)

For Gradient Boosting Classifier the minimum error rate = 3.33% (GBC with 10000 estimators, 2 features, 2 max_depth)

So, the best error rate is given by Neural Network.

Please find the accuracy with exact hyper-parameters in Accuracy.xlsx & Result.txt files.

Some of the values may not match exactly because of the randomization involved in the internal implementation of MLPC and GBC classifiers.

*As the SVM with polynomial kernel is slow, I have used LinearSVM in place of polynomial SVM of degree 1 (LinearSVM is equivalent to polynomial SVM of degree 1 but it is much faster)

2 K-means clustering on images [30 points]

In this problem, you will use K-means clustering for image compression. We have provided you with two images.

• Display the images after data compression using K-means clustering for different values of K (2, 5, 10, 15, 20).

• What are the compression ratios for different values of K? Note that you have to repeat the experiment multiple times with different initializations and report the average as well as variance in the compression ratio.

The formula for Compression % = 100 – (new size/old size)*100

Image	k	Avg Compression %	Variance
Penguins.jpg	2	87.83495214	0
Penguins.jpg	5	86.09666575	6.91E-05
Penguins.jpg	10	86.0617226	16.76713628
Penguins.jpg	15	84.63672887	0.061676635
Penguins.jpg	20	84.84418932	0.020424714
Koala.jpg	2	76.36584613	0.000106022
Koala.jpg	5	74.68462446	1.70E-05
Koala.jpg	10	78.31045386	0.021999795
Koala.jpg	15	79.37807285	0.015191962
Koala.jpg	20	79.71943993	0.009717063

Please find the complete results in compression_perc.xlsx & compression.xlsx

*The average and variance values may differ slightly from the above table during testing due to randomization in initializing cluster centers. Due to different initializations the kmeans algorithm may reach different local minima.

* I have also implemented K-means++ algorithm for initializing cluster centers which results in marked improvement in final error (local minima) of k-means and takes less number of iterations to reach the local minima.

• Is there a tradeoff between image quality and degree of compression. What would be a good value of K for each of the two images?

Yes. As the compression ratio increases, a lesser number of colors are assigned to the pixels of the image. For example, for k=2 all the pixels have to choose between only two color combinations. As a result, the quality of the image is degraded. The degradation is more prominent in more colorful and vibrant images which become dull after compression.

In my view, as there are a lot of colors in Penguins.jpg, we have to choose a large value of K to preserve most of the colors. So, in the above table the value of K=20 should be appropriate. Please check the quality of the compressed image penguins20.jpg

In case of Koala.jpg, there are very less colors as compared to Penguins.jpg, so we can choose a lower value of K, achieve a good compression ratio and still preserve the image quality to a high degree. So, according to the above table the value of K=10 should be appropriate. Please check the quality of the compressed image koala10.jpg

We have provided you a java template “KMeans.java” which implements various image input/output operations. You have to implement the function kmeans in the template. If you want to use python, you will have to replicate the code in KMeans.java in python (will take roughly 10-15 minutes).

What to turn in for this homework:

In a single zip file:

- A PDF report containing your write up for part 1 and 2.
- Your source code for the kmeans algorithm.

Note that your program must compile and we should be able to replicate your results. Otherwise no credit will be given.