CSE 591 Project 1

Due October 4th, Friday, 11:50PM via email to the TAs.

Submit a short report, your code, and any required output files in a single zipped file as an email attachment to the TAs.

Note: There is no specific requirement on the format or length of the report. Just explain clearly what you did for the project and include any required outputs, plots/images and anything else you feel important to show. Also, the report needs to briefly describe how to run your code for verifying your results. Do NOT include your code in the report; the code should be in separate files.

If you have any question regarding this assignment, please post it on the Blackboard under the forum thread created for this project. Please check if others have asked a similar question before posting your question.

Choose only one of the following options.

Option 1. Bayesian Decision Theory for Classification

Option 1 is based on two data files named Training.txt and Testing.txt, both being plain text files. This is a 2-dimensional, 3-class problem. Each of the files contains 1000 rows. Each row represents a 2-D feature point and its label: if we denote a feature vector by $\mathbf{x} = (x_1, x_2)$, the first two values in each row give a sample of (x_1, x_2), and the third number in the row is the class label (1, 2, or 3) of the sample. For example, the second row of Training.txt is

```
-1.619039   1.898733   3
```

which represents a feature vector (`-1.619039, 1.898733`) from Class 3.

We further assume that the underlying class conditional densities are all 2-D normal densities. That is, $p(\mathbf{x}|\omega_i)$ for $i=1,2,3$ have the form of the multivariate normal density given in Slide 28 of Notes02 (although different classes may have different means and/or covariance matrices).

Write computer programs to do the following:

1. (15 points) Using the data from Training.txt only, find the maximum likelihood estimation for the means and covariance matrices for the class conditional densities for the three classes respectively. Include the results in your report.

2. (45 points) Assuming the prior probabilities of the three classes are equal (i.e., $P(\omega_1)= P(\omega_2)= P(\omega_3) =1/3$), use the results from 1 to do minimum-error classification. In this experiment, you will need to intentionally withhold the labels given in the data file, and only take the feature values as input to your classifier, which should generate a class label. At that moment, you can compare the label generated by your code with the given label (the given label is called "groundtruth"). If these two are the same, the classification is correct; Otherwise, there is an error.

2.1 Classify all feature points from Training.txt, and compute the error rate (# of errors/1000).

2.2. Classify all feature points from Testing.txt, and computer the error rate.

In both 2.1 and 2.2, include the obtained error rates in your report, and record your classification results by saving the labels generated from your code into a 4th column of two new files, ClassifyTrainingEqualP.txt and ClassifyTestingEqualP.txt, in which the first three columns are just copied from the original files Training.txt and Testing.txt respectively. For example, each row in ClassifyTrainingEqualP.txt now contains the following four numbers:

x_1 x_2 groundtruth_label label_from_your_code

3. (40 points) Now, assume that the prior probabilities of the three classes are unknown and must be estimated using the maximum likelihood estimation approach based on Training.txt only. First estimate $P(\omega_1)$, $P(\omega_2)$ and $P(\omega_3)$ and include them in your report.

With these priors and the class-conditionals from 2, re-do the classification:

3.1 Classify all feature points from Training.txt, and compute the error rate (# of errors/1000).

3.2. Classify all feature points from Testing.txt, and computer the error rate.

In both 3.1 and 3.2, include the obtained error rates in your report, and record your classification results by saving the labels generated from your code into a 4th column of two new files, ClassifyTrainingMLEP.txt and ClassifyTestingMLEP.txt, in which the first three columns are just copied from the original files Training.txt and Testing.txt respectively, similar to what you did in 2.

Option 2. Principal Component Analysis for Images

In this option, you will be working with a given image dataset of many 64×64 face images (from the supplied file imgFolder.zip). The images are given as PNG files. You need to figure out a way to read an image into the computer's memory for processing. In Matlab, this may be achieved by im = imread('filename.png').

Part 1: Computation of PCA bases (40 points)

1. Write code to read in all the given images and perform PCA using all the data. For a given image of 64x64 pixels, you will need to "vectorize" it to form a 4096-dimensional vector by concatenating all columns. All the given images lie in this 4096-dimensional space, and your code will perform PCA with the given data in this space so that you may figure out a lower-dimensional subspace for further processing tasks.

2. Display the first 10 principal components as images, and include them in your report. (Hint: these are the top 10 Eigen vectors from the PCA algorithm, which are all 4096-dimensional vectors, and you can "de-vectorize" them by putting the elements into 64x64 images, reversing what you did in "vectorizing" the images. You may also need to properly scale the obtained "images" in order to display them well, as the values from PCA may not lie in a proper range for direct visualization.)

Part 2: Study how the principal components approximate the data (40 points)

Now let's assume that we keep only the top K principal components, and K varies from 1 to 100. The top K principal components $\{\mathbf{u}_i, i=1,\ldots,K\}$, ordered by their corresponding Eigen values in descending order, define a PCA projection/transform given by the matrix $\mathbf{U}=[\ \mathbf{u}_1, \ldots, \mathbf{u}_K]$ (where $[\mathbf{u}_1 \ldots, \mathbf{u}_K]$ is a matrix with $\mathbf{u}_i$ as columns, $i=1,2,\ldots,K$). For a column vector $\mathbf{x}$ representing a given image, its PCA transform is computed as $\mathbf{c} = \mathbf{U}^t\mathbf{x}$, where $\mathbf{c}$ is the new K-dimensional projection of the original vector $\mathbf{x}$ onto the subspace spanned by$\{\mathbf{u}_i, i=1,\ldots,K\}$, and the supscript t in $\mathbf{U}^t$ denotes matrix transposition.

With a lower-dimensional representation $\mathbf{c}$ for $\mathbf{x}$, we may estimate $\mathbf{x}$ by $\hat{\mathbf{x}} = \mathbf{U}\mathbf{c}$. This is only an approximation and we can compute the error as $e(\mathbf{x}) = \| \mathbf{x} - \hat{\mathbf{x}} \|^2/4096$, where $\| \cdot \|$ is the l_2-norm (and thus $e(\mathbf{x})$ corresponds to the mean squared error of the approximation for $\mathbf{x}$). If you compute this error $e(\mathbf{x})$ for each image $\mathbf{x}$ in the dataset and then average them, you have the averaged error for the entire dataset. Note: depending on the value of K, this averaged error may vary. Plot this averaged error as a function of K, for K=1,2,…,100. Further, also plot the variance of the averaged errors as a function of K, for K=1,2,…,100. Include both plots in your report.

Part 3: Study how image scaling may impact PCA (20 points)

In this part, you will first need to scale all the images down to 32x32. (Hint: you may use imresize in Matlab. It is a good idea to use a proper interpolation scheme, such as the bilinear interpolation method during downscaling an image.)

A. With the 32x32 images, first re-do the same tasks in Part 1 and Part 2 and get corresponding 10 images (as in Part 1, Step 1), and the averaged error plots (as in Part 2). (Note: In computing the averaged error in Part 2, the normalizing factor in e(**x**) should be changed to 32x32=1024; do NOT use 4096.)

B. Now let's take a seemingly intuitive "short-cut": we do not want to bother to re-do PCA with all the 32x32 images. Instead, we simply down-scale by the same factor the principal components we obtained in Parts 1&2. That is, we rearrange the principal components into 64x64 images, and then down-scale them into 32x32 images (with the same process we downscaled all the images), and then vectorize the images again to get 1024-dimensional vectors. We then use so-obtained vectors as if they were the true principal components for the 32x32 dataset, and re-do Part 2 to generate the error plots. Include your plots in your report. Compare these plots with those from Step A above.