COL 774: Assignment 4

Due Date: 11:50 pm, Thursday April 21. Total Points: 42 (for the first two questions)

Notes:

- This assignment has three questions. Two of the questions are described below. The details of the third question will be shared with you separately.
- You should submit all your code (including any pre-processing scripts written by you) and any graphs that you might plot.
- Do not submit the datasets. Do not submit any code that we have provided to you for processing.
- Include a **single write-up (pdf) file** which includes a brief description for each question explaining what you did. Include any observations and/or plots required by the question in this single write-up file.
- You should use MATLAB/Python for your implementation.
- Your code should have appropriate documentation for readability.
- You will be graded based on what you have submitted as well as your ability to explain your code.
- Refer to the course website for assignment submission instructions.
- This assignment is supposed to be done individually. You should carry out all the implementation by yourself.
- We plan to run Moss on the submissions. We will also include submissions from previous years since some of the questions may be repeated. Any cheating will result in a zero on the assignment, a penalty of -10 points and possibly much stricter penalties (including a **fail grade** and/or a **DISCO**).

1. **(22 points) K-means**
   In this problem, you will be working with a dataset for recognizing human activities and postural transitions based on attributes collected through a smartphone. The dataset comes with a set of 10411 examples, with each example being represented by a set of 561 attributes (attr.txt) and a class label in the set $\{1, \cdots, 6\}$ (label.txt). To know more about the dataset, click here.

   (a) **(8 points)** Implement the K-means ($K = 6$) clustering algorithm (as discussed in class) until convergence to discover the clusters in the data. Start from an initial random assignment of the cluster means and run it until convergence. Report your findings. Note that for this part (and the next one), you do not need to use the class labels.

   (b) As discussed in class, one of the ways to characterize the quality of clusters obtained is to calculate the sum of squares of distance of each of the data points $x^{(i)}$ from the mean of the cluster it is assigned to i.e. the quantity $J(c, \mu) = \Sigma_i (x^{(i)} - \mu_{c^{(i)}})^2$ where $x^{(i)}$ denotes the ith data point, $c^{(i)}$ is the index of the cluster that $x^{(i)}$ belongs to and $\mu_{c^{(i)}}$ denotes the mean of the cluster with index $c^{(i)}$. $J(c, \mu)$ essentially captures how close the points within each cluster are (or equivalently, how far points across different clusters are). Presumably, smaller the value of $J(c, \mu)$, better the clustering is.

   - (**3 points**) Run your K-means algorithm with 10 different random initializations. In each case, compute the final value of J. What do you observe?
   - (**3 points**) Pick the initialization that gave you the best J value in previous part. Plot the quantity J as we vary the number of iterations from 1 to 60. Comment on your findings.

(c) **(4 points)** Since we have the true labels of the data in this example, we can plot the accuracy of clustering as we run the K-means algorithm. For each cluster obtained at a given step of K-means, assign to the cluster the label which occurs most frequently in the examples in the cluster. The remaining examples are treated as misclassified. Plot the accuracy (ratio of the number of correctly classified examples to the total number of examples) as we vary the number of iterations from 1 to 60. Comment on your observations. How does this graph compare to the one you drew in part(b) above?

(d) (**4 points**:) Use the scikit-learn library of Python to learn a multi-class linear SVM classifier (use the default one-vs-one setting). Use the default value of C parameter. Perform 10 fold cross-validiation over the data and report your cross-validation accuracies. How do these compare with the accuracies that you obtained using K-Means in part(c) above (at convergence)? Comment on your observations.

2. **(20 points) Principal Component Analysis**

~~In this problem, you are given a database of grayscale face images for the task of face recognition (each pixel value belongs to the set $\{0, 1, 2, \ldots 255\}$.) You will apply PCA on the given data which can be seen as a pre-processing step for the recognition task. You should not normalize the data since each pixel value comes from the same scale (0-255).~~

In this problem, you are given two different databases of face images for the task of face recognition with details as follows.

- Dataset 1 [1]: This is a dataset of 1288 colored face images with 7 different classes. Each image in this dataset is 50×37 pixels. You should convert the images in the dataset to grayscale before any further experimentation.
- Dataset 2 [2]: This is a dataset of 400 grayscale images with 10 different classes Each image in this dataset is ~~64×64~~ 92×112 pixels.

You will apply PCA on the given datasets and then use the projected data for the task of classfication. You should not normalize the data since each pixel value comes from the same scale (0-255). Perform the following experiments for each of the two datasets (separately for each dataset).

(a) **(3 points)** Calculate the average face for ~~the dataset~~ each of the datasets and display it.

(b) **(6 points)** Implement the PCA algorithm using the SVD formulation discussed in class. You should not call any function to perform PCA directly but rather do it explicitly using SVD. You should be able to perform SVD in Matlab/Python using existing libraries. Note that since data is not normalized to zero mean, you might have to change the co-variance matrix calculation accordingly. Calculate the principal components corresponding to top 50 eigenvalues. You can store the principal components in a single file as a sequence of numbers (one number for each pixel). ~~a sequence of $361 (19 \times 19)$ numbers.~~

(c) **(3 points)** Display the eigenfaces corresponding the top five principal components. You might have to scale the pixel values so that maximum value touches 255.

(d) **(2 points)** For each face in the databsae, project it on to the top 50 dimensions obtained using PCA. Store these projections in a separate file.

(e) **(4 points)** Since this is a face recognition task, the dataset provided to you also comes with class labels (i.e, the label of the person associated with each face). Use the scikit-learn library of Python to learn a multi-class linear SVM classifier using the original set of attributes (i.e., grayscale values) using the default one-vs-one setting. Perform 10-fold cross-validation. What accuracy do you obtain? Now, learn a separate classifier using the projected set of attributes. Again, perform 10 fold cross-validation. How do the accuracies obtained in each step compare with each other? What are the times taken to do the classification (i.e., total time for cross-validation) in each case? Comment on your observations.

(f) **(2 points)** Write a small program which takes as input a face id in the database, reads its projection computed in part above, and displays the projected image. Try it for a few different faces. How different is the projected image from the original image? Comment on your observations.

(g) **(Extra Fun: No Credits!)** Download a few (at least 5) non-face images from the web. Convert them to grayscale and the size same as those in the face image database. Project these images on the 50 dimensions obtained from PCA. How do the projected images look like? Comment on your observations.

3. **Online ML Competition:** Details will be communicated separately.

[1] Source: http://vis-www.cs.umass.edu/lfw/
[2] Source: https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html