Lab-5&6
CSL2050 - Pattern Recognition and Machine Learning

NOTE:

1. This Programming Assignment is associated with your Lab-5 and Lab-6.
2. Perform all tasks in a single Google Colab file. Prepare a report regarding the steps followed while performing the given tasks. The report should not include excessive unscaled preprocessing plots or screenshots. A well-written report is important for getting higher points.
3. Try to modularize the code for readability wherever possible. Submit a zip with the Colab file [.ipynb] and report [.pdf] in the classroom. Name your files with your roll-number. (ex: B20CS003.ipynb, B20CS003.pdf, B20CS003.zip). **Not following the instructions may lead to zero points in the assignment.**
4. Please refer to the Academic Code of Honor for this course (ref: Lecture-1) before submission of this programming assignment.
5. **Maximum Points:** 150 Points
6. **Deadline:** March 2, 2024, 10:30 PM.
7. **Late Submission Policy:** Late submissions beyond the due date will incur a 10% penalty for each day. Plan the submission ahead, and do not wait until the last minute.

Problems:

1. **(Perceptron)**

   In this problem, your goal is to implement *Perceptron* to solve the linear classification problem. (Please refer to Lecture-11 and Blog shared in the Google Classroom).

   In the training input file, the first line will show the number of the input vectors. Following that number, each line is an input vector. Since the input vector is 4D and the label of each input is 0 or 1, each line will contain $4 + 1 = 5$ numbers. Here is an example:
   4
   0 6 5 0 1
   5 2 2 9 1
   3 5 0 0 0
   9 4 1 5 0

   The testing input file is similar, but labels are not provided. For example:
   3
   1 2 3 4
   2 2 3 3
   1 1 4 4

   For your convenience, all the numbers in both training and testing sets are integers. However, you should use double for the weights.

   That means the first data point in the testing set belongs to type "0", the second data point belongs to type "0", etc.

   You are supposed to develop two codes, namely train.py and test.py. The train.py takes train.txt as argument and saves the weights w_0, w_1, w_2, w_3, w_4. The test.py takes test.txt file and returns labels of each sample in a comma-separated form.
   $ train.py train.txt
   $ Training Over and Weights are saved
   $ test.py test.txt
   $ 0 0 1

   Task-0: Generate a synthetic 4-dimensional dataset:

   (a) Initial a linear function: $f(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4$.

   (b) Choose w_0, w_1, w_2, w_3,and w_4 randomly.

(c) Evaluate $f(\mathbf{x})$. If the result is greater than or equal to zero, then label it as a positive example, otherwise label it as a negative example.

(Deliverable: data.txt) **(5 points)**

Task-1: Write training code for the perceptron learning algorithm. (Do not forget to normalize the data both during training and testing). (Deliverable: train.py) **(10 points)**

Task-2: Write testing and evaluation code. Report accuracy. (Deliverable: test.py) **(10 points)**

Task-3: Report results with training using 20%, 50%, and 70% of synthetic data. (Deliverable: table in the report comparing results) **(5 points)** (This problem is prepared by modifying the following assignment: `https://cs.fit.edu/~dmitra/Cplus/2019Spr/assignment5-2.pdf`)

2. **(Eigenfaces)** In this problem, you will explore Eigenfaces – a dimensionality reduction technique based on PCA for face recognition.

   Reference code for Eigenfaces: `https://scikit-learn.org/stable/auto_examples/applications/plot_face_recognition.html`

   Dataset: We will use the Labelled Faces in the Wild (LFW) dataset. You can find the code to load it directly using the Sklearn library. (Check reference code above) or you can download the dataset from this link: `http://vis-www.cs.umass.edu/lfw/lfw-funneled.tgz`

   Task-1: **Data Preprocessing** (a) Load the LFW dataset using the Scikit-learn's fetch_lfw_people function. (b) Split the dataset into training and testing sets using an 80:20 split ratio. **(5 points)**

   Task-2: **Eigenfaces Implementation** (a) Implement Eigenfaces using Principal Component Analysis (PCA). Set an appropriate value for $n_{components}$ and explain your choice in the report (refer explained_variance_ratio_ attribute for PCA in the Scikit-learn documentation to justify your choice). **(15 points)**

   Task-3: **Model Training** (a) Choose a classifier for Eigenfaces (e.g., K-Nearest Neighbors) and train the classifier using the transformed training data. **(10 points)**

   Task-4: **Model Evaluation:** (a) Use the trained Eigenfaces classifier to make predictions on the Eigenfaces-transformed testing data. (b) Calculate and report accuracy. (c) Visualize a subset of Eigenfaces and report the observations.(Report on what type of test images the model is failing, and mention ways to improve the model) **(15 points)**

Task-5: **Experiment with different values of** $n_{components}$ in PCA and observe the impact on the performance metrics (accuracy). **(5 points)**

Note: complete each task thoroughly and document your findings in the lab report.

Rubrics:
Task completion with proper documentation and variable naming: 80 Points
Viva: 30 Points
Report: 40 Points

End of Paper