Spring 2020
CSCE 666 Pattern Analysis
Homework #2

Due date: 02/24/2020

In recognition of the Texas A&M University policies of academic integrity, I certify that I have neither given nor received dishonest aid in this homework assignment.

Name: _____ *Signature:* _____

PLEASE FOLLOW THESE GUIDELINES:

1. *Download the compressed file 'hw2.zip' from the course web page*
2. *Submit your solutions as a report, with each problem being a separate section of the report –you may use this assignment as a template*
3. *Please show your work and discuss your findings. This ensures full credit if your results are correct, and allows me to give you partial credit otherwise*
4. *Sign and return this page with your finished assignment.*
5. *Submit your code as a ZIP file through ecampus, each problem on a separate subfolder*

Problem 1 (25%)

(1) Use a Gaussian kernel to generate a kernel density estimate (KDE) of the univariate dataset *hw2p1_data.csv*. Compute the estimate at a set of points evenly spaced over the range of the data. For comparison purposes, plot the KDE on the same graph as the normalized histogram of the data. Experiment with the value of the bandwidth h to obtain an appropriate smoothing effect. Show results for a "small", a "large" and your "best" bandwidth.

> *NOTE: You are expected to implement <u>your own</u> KDE code.*
> *HINT: To save trees, use the command 'subplot' to display the three plots ("small", "large" and "best") on the same window.*

(2) Next we will use the leave-one-out method outlined in Lecture 7 to estimate the optimum bandwidth.

(a) Split the dataset into two subsets: a training set containing all but the n^{th} example, and a validation set containing only the n^{th} example.
(b) Build a KDE using <u>only</u> the training set, and estimate the log-likelihood of the validation sample (i.e., the logarithm of the KDE at the validation sample)
(c) Repeat steps (a-b) for every example in the dataset, and compute the average log likelihood. Store this value.
(d) Repeat steps (a-c) for 100 different bandwidths, logarithmically spaced between $h_0/100$ and $100h_0$ as follows:

$$h = \left\{\frac{h_0}{100}, \alpha^1\frac{h_0}{100}, \alpha^2\frac{h_0}{100}, \alpha^3\frac{h_0}{100} \ldots \alpha^{98}\frac{h_0}{100}, 100h_0\right\} \; with\; h_0 = 0.9AN^{-1/5}, A = \min\left(\sigma_x, \frac{iqr(x)}{1.34}\right)$$

(e) Generate a plot of the average log-likelihood in (c) versus the bandwidth.
(f) Select the bandwidth which yields the highest log-likelihood, and
    1. Generate a KDE with this bandwidth, and with the plug-in estimate h_o,
    2. Plot the two KDE on the same graph as the normalized histogram, and
(g) <u>Discuss your results</u>.

Problem 2 (10%)

Load the binary file *hw2p2_data.csv* into MATLAB. This will create a matrix x, which consists of high-dimensional feature vectors arranged by rows:

(a) Generate a scatter plot of the first three dimensions. Rotate the reference frame (command *rotate3d*) and try to find structure in the data by changing the viewpoint. Plot the best orientation that you can find. Can you identify any structure?
(b) Compute the Principal Components of the data and generate a plot of the eigenvalues, sorted in decreasing order. How many eigenvalues are responsible for most of the variance in the data?
(c) Generate a scatter plot of the first three PCA projections (the ones with largest eigenvalues). Rotate the reference frame and try to find structure in the data by changing the viewpoint. Plot the best such orientation. Can you identify any structure? Where else in the data may the structure be?
(d) Discuss your findings.

NOTE: You are expected to implement your own PCA code.

Problem 3 (10%)

Generate 250 examples for each of the three densities:

$$\mu_1 = [5 \quad 0 \quad 5]^T \quad \mu_2 = [4 \quad 6 \quad 7]^T \quad \mu_3 = [6 \quad 2 \quad 4]^T$$

$$\Sigma_1 = \begin{bmatrix} 5 & -4 & -2 \\ & 4 & 0 \\ & & 5 \end{bmatrix} \quad \Sigma_2 = \begin{bmatrix} 3 & 0 & 0 \\ & 3 & 0 \\ & & 3 \end{bmatrix} \quad \Sigma_3 = \begin{bmatrix} 6 & 5 & 6 \\ & 6 & 7 \\ & & 9 \end{bmatrix}$$

To simulate noisy channels, augment the feature vector with 48 additional dimensions, each one from a Gaussian distribution $N(\mu = 1; \sigma = 6)$ regardless of class.

(a) Compute the first two principal components of the data, and generate a scatter plot of all the examples. Use the *text* function to plot examples by their class label. Use the *axis* function to adjust the limits of the plot.
(b) Compute the Fisher's LDA projections of the data, and generate a scatter plot of all the examples. You may use the *tamu_lda.m* implementation included in *hw2.zip*, any other existing implementation, or implement it on your own.
(c) Discuss your results.

HINT: Use the MATLAB command 'mvnrnd' to generate data from a multivariate Gaussian pdf.

NOTE: $\sigma = 4$ is the standard deviation, NOT the variance.

Problem 4 (10%)

Implement a Quadratic classifier for Problem 3

(a) Split the data into training and test sets by randomly selecting 25% of the examples from each class for the test set.
(b) Classify the test data in the original high-dimensional space.
(c) Repeat the above steps several times. What is the average classification rate?

(d) Compute the PCA projections using ONLY the training data
(e) Project both training and test data using the first three PCA eigenvectors
(f) Classify the test data in a 2-dimensional PCA subspace.
(g) Repeat the above steps several times. What is the average classification rate?

(h) Compute the LDA projections using ONLY the training data
(i) Project both training and test data using the LDA eigenvalues
(j) Classify the test data in the LDA subspace.
(k) Repeat the above steps several times. What is the average classification rate?
(l) Discuss your results

Problem 5 (10%)

Repeat Problem 4, but this time implementing a KNN classifier. Experiment with the value of k. Discuss your results and compared them with those in the previous exercise.

Problem 6 (35%)

In the previous problems you had the opportunity to develop, debug and evaluate two classifiers using synthetic data generated on your own. Things get a little more interesting now, because we are going to use a blind test to evaluate the performance of your implementations.

The file *hw2.zip* contains the following data files:

- *hw2p6_x1.csv* : training set (row) vectors
- *hw2p6_c1.csv*: training set labels
- *hw2p6_x2.csv* : validation set (row) vectors
- *hw2p6_c2.csv*: validation set labels

You are to:

(a) Perform dimensionality reduction to visualize the structure of the data. How many eigenvalues should be used for PCA and LDA? Which technique does a better job at unfolding the structure of the data? Why?
(b) Use your training set as in Problems 4 and 5 to determine which classification approach works best.
(c) Discuss your findings. Can you reconcile the results in part (b) with the structure of the data and what you know about the two classifiers?
(d) Prepare a MATLAB program (please call it *hw2p6.m*) that takes as an argument the name of a CSV file containing test examples (arranged as rows vectors, as in *hw2p6_x2.csv*) and generates a class label for each test example. Once you submit your code, I will run it on a separate test set (say, *hw2p6_x3.csv*) containing my own test data. Your grade will be based on the performance of your classifier on my test set, which will contain a very large number of examples so I can approximate the true error rate. Needless to say, all datasets will be generated from the same distribution.

NOTES:

- Please submit a single ZIP file (**your_last_name.zip**).
- Make sure your code can handle ANY file name for the test set, and ANY number of examples in the test set.

- Make sure your code works. Are all required files included? You will receive no credit if the program returns with an error. I will run your code on *unix.cs.tamu.edu*, so you should test it there before submitting.
- To facilitate grading, your program should create a CSV file (please call it *test_labels.csv*) containing the predicted class labels arranged as a COLUMN VECTOR, one label for each of the examples in the test set (i.e., *hw2p6_x3.csv*). These are the class predictions that I will compare against the true class labels of my test set, which I have kept aside.