CSCI 5521: Introduction to Machine Learning (Spring 2024)[1]

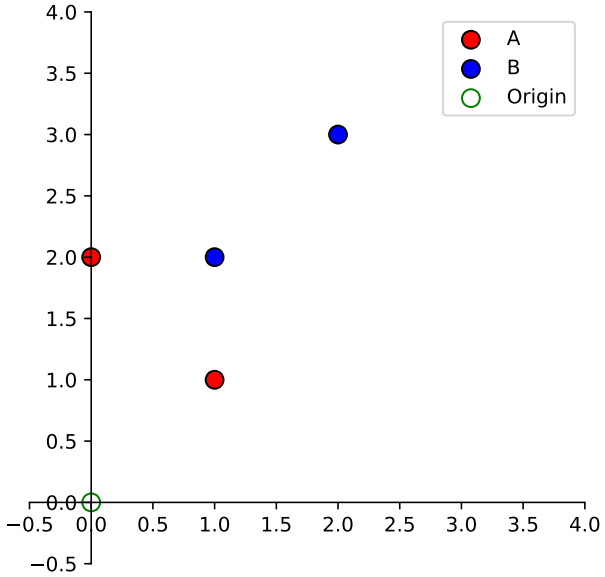
Homework 2

Due date: Feb 28, 2024 11:59pm

1. (**30 points**) Consider the following 2 sets of points in the plane:

$A: \begin{pmatrix}0\\2\end{pmatrix}, \begin{pmatrix}1\\1\end{pmatrix}$

$B: \begin{pmatrix}1\\2\end{pmatrix}, \begin{pmatrix}2\\3\end{pmatrix}$



   (a) What is the first principal component $\boldsymbol{w}_1$ (use the **unbiased** estimation of covariance)? Draw the first principal component direction $\boldsymbol{w}_1$ on the plot, anchored at the origin.

   (b) Consider a more general case (not specific to the aforementioned samples): PCA performs linear dimensionality reduction with $\mathbf{z}^t = \mathbf{W}^T\mathbf{x}^t$, where $\mathbf{x}^t \in \mathbb{R}^D$ is the original data for the t-th sample, $\mathbf{z}^t \in \mathbb{R}^d$ is the low-dimensional projection ($d < D$), $\mathbf{W} \in \mathbb{R}^{D\times d}$ is the PCA projection matrix (each column is a principal component). Professor HighLowHigh claims that we can reconstruct the original data with $\mathbf{v}^t = \mathbf{W}\mathbf{z}^t$, so that $\forall_t\ \mathbf{v}^t = \mathbf{x}^t$. Is the claim correct? Explain your answer with necessary details (you can use formulations if it helps explain).

   (c) Compute the Within-Class Scatter matrix $\boldsymbol{S_W}$ and Between-Class Scatter matrix $\boldsymbol{S_B}$.

   (d) What is the Fisher projection direction $\boldsymbol{w}$ found by the Fisher Linear Discriminant Analysis(LDA)? Normalize $\boldsymbol{w}$ to have unit length, that is $||\boldsymbol{w}||_2 = 1$, and draw such $\boldsymbol{w}$ on the plot, anchored at the origin.

[1]Instructor: Catherine Qi Zhao. TA: Xianyu Chen, Amarachi Nzeukwu, James Yang, Yifeng Zhang. Email: csci5521.s2024@gmail.com

Note: You need to show all your calculations in order to get full credits.

2. (**30 points**) Given the following data points in 1D: $x_1 = 4, x_2 = 6, x_3 = 7, x_4 = 8, x_5 = 10, x_6 = 12, x_7 = 14, x_8 = 16$, perform k-means clustering algorithm for $K = 2$.

   (a) Start from initial cluster centers $c_1 = 3$, $c_2 = 12$. Show your steps for all iterations: (1) the cluster assignments $y_1, \cdots, y_9$; (2) the updated cluster centers at the end of that iteration.

   (b) How many iterations does it take for k-means algorithm to converge (*i.e.,* number of iterations includes all iterations you perform to find convergence)? What is the reconstruction error (*i.e.*, distortion measure J, equation 9.1 of the Bishop's textbook) at the end of that iteration?

   (c) Repeat the above steps with initial cluster centers $c_1 = 8, c_2 = 14$.

   (d) How many iterations does it take for k-means algorithm to converge in this case? What is the reconstruction error at the end of that iteration?

   (e) Comparing (a) with (c), which solution is better? Why?

3. (**40 points**) In this programming exercise you will implement k-means and Principal Component Analysis algorithms:

   (a) You will first apply k-means algorithm ($K = 8$ and $K = 4$) to the provided dataset `Digits089.csv`. The dataset contains 3000 samples, where each sample has 784 features. The first column of the data file contains flags that are used for separating training and test samples, the second column includes class labels (*i.e.,* 0, 8, 9), and the rest of the columns store features. Your program should initialize the centers with a set of pre-selected samples (see the template for detailed initialization), iteratively update the center of each cluster based on the training samples, and record the reconstruction error after each iteration. The following table demonstrates the key steps and corresponding helper functions that you are going to implement:

Function	Desp.
initCenters(X, dataIndex)	store K centers (specified by dataIndex) in (K, D) array
computeDis(x, centers)	compute Euclidean distance between the data and centers
assignCen(distances)	assign the data to closest centers according to distances
updateCen(new_center)	update the center position based on data assignments
computeError(X, assign, centers)	compute the reconstruction error

Table 1: Helper Functions for K-Means

After reaching convergence, we can assign a class label to each cluster based on majority voting (*i.e.,* use the most dominant label in the cluster as its class label). To estimate the quality of clustering, you will use the clusters for classification of test samples. More specifically, you should determine the cluster for each test sample, and use the previously assigned class label of the selected cluster for prediction.

For different K, report the number of iterations for convergence and the classification accuracy on the test samples. Plot the history of reconstruction errors for both K. Are the plot shapes following what you expect? Compare the results (reconstruction errors and classification accuracy) of different K and briefly explain the reason. The code for plotting and computing classification accuracy is included in `hw2.py`, and you do not need to modify the file.

(b) Repeat the above, but use low-dimensional data obtained from PCA. Your PCA algorithm should reduce the original training samples to dimensions needed to capture $> 95\%$ of the variance. Note that you should use the means and projection matrix learned from training samples for projecting test samples. The following table demonstrates the key steps and corresponding helper functions that you are going to implement:

Function	Desp.
centerData(X)	center the data by subtracting their mean
computeE(centered_X)	compute eigenvectors/values from centered data
computeDim(eig_val, percent)	compute the number of eigenvectors to keep
project(X, w)	project the data to lower dimensions by projection matrix

Table 2: Steps and Functions for PCA

How many dimensions are necessary in this case? Does PCA help clustering? Explain. (Hint: Consider both the classification accuracy and the runtime of the algorithm.)

(c) Repeat (b), but using only the first principal component. **Are the results better? Explain.** Note you do not need to write any codes for this case.

We have provided the skeleton code `Mykmeans.py` and `MyPCA.py` for implementing the algorithms. They are written in a *scikit-learn* convention, where you have a *fit* function for model training and a *predict* function for generating predictions on given samples. To verify your implementation, call the main function `hw2.py`, which automatically generates plots for the reconstruction errors and computes the classification accuracy.

Submission

- **Things to submit:**

  1. hw2_sol.pdf: a document containing all your answers for the written questions (including answers and plots for problem 3).
  2. `Mykmeans.py`: a Python source file containing your implementation of the k-means algorithm with header `class Kmeans`. Use the skeleton file `Mykmeans.py` found with the data on the class web site, and fill in the missing parts. The `fit` function should take the training samples and training labels as inputs, initialize the centers, update the model parameters and record the reconstruction errors. It should also assign a class label to each cluster, and return the number of iterations for convergence as well as the history of reconstruction errors. The `predict` function should take the test samples as inputs, and predict their class labels based on clustering results. The `computeError` function should take the samples and cluster assignment (a vector specifying the cluster ID assigned to each sample) as inputs, and return the reconstruction error for the current assignment.
  3. `MyPCA.py`: a Python source file containing your implementation of the Principal Component Analysis algorithm with header `class PCA`. Given the original samples and an optional parameter *num_dim* as inputs, the `fit` function should determine the projection matrix based on the training samples, and return the low-dimensional projection as well as the dimension of features. If *num_dim* is specified during initialization of the class, it should project the samples to d dimensional data, where d=*num_dim*. Otherwise, it will use the dimensions that capture $> 95\%$ of the variance. The `predict` function should return the low-dimensional projection of test samples based on the projection matrix and means learned from the training samples.

- **Submit**: All material must be submitted electronically via Gradescope. **Note that there are two entries for the assignment, *i.e.,* Hw2-Written (for hw2_sol.pdf) and Hw2-Programming (for a zipped file containing the Python code), please submit your files accordingly.** We will grade the assignment with vanilla Python, and code submitted as iPython notebooks will not be graded.