ECE 443 (Fall 2022) – Programming Exercise #2 (75 points)

Last updated: October 9, 2022

Rationale and learning expectations: We have discussed the concepts of feature engineering as well as feature learning using *Principal Component Analysis* (PCA) in the class. The purpose of this exercise is to reinforce many of the concepts discussed in relation to these two topics. This exercise is also meant to reinforce many of the linear algebra concepts that were discussed in relation to our discussion of the PCA problem. Students attempting this exercise are expected to understand the basics of feature engineering and the fundamental concepts (including the ones related to linear algebra) underlying the PCA problem at the conclusion of this activity.

General Instruction: All parts of this exercise must be done within a Notebook, with text answers (and other discussion) provided in text cells and commented code provided in code cells. Please refer to the solution template for Exercise #2 as a template for your own solution. In particular:

- Replace any text in the text cells enclosed within square brackets (e.g., `[Your answer to 1.1a goes in this cell]`) with your own text using Markdown and/or LaTeX .
- Replace any text in the code cells enclosed within pairs of three hash symbols (e.g., `### Your code for 1.1a goes in this cell ###`) with your own code.
- Unless expressly permitted in the template, **do not** edit parts of the template that are not within square brackets / three hash symbols and **do not** change the cell structure of the template.
- Make sure the submitted notebook is fully executed (i.e., submit the notebook only after a full run of all cells).

Restrictions: You are free to use `numpy`, `pandas`, `scipy.stats`, `matplotlib`, `mpl_toolkits.mplot3d`, `seaborn`, and `IPython.display` packages within your code. Unless explicitly permitted by the instructor, you are not allowed to use any other libraries, packages, or modules within your submission.

Notebook Preamble: I suggest importing the different libraries / packages / modules in the preamble as follows (but you are allowed to use any other names of your liking):

```
import numpy as np
import pandas as pd
from scipy import stats as sps
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from IPython.display import display, Latex
```

1 Feature Engineering for *Environmental Sensor Telemetry Data*

In this part, we will focus on 'feature engineering' (*aka*, hand-crafted features) for the *Environmental Sensor Telemetry Data*. This dataset corresponds to *time-series* data collected using three identical, custom-built, breadboard-based sensor arrays mounted on three Raspberry Pi devices. This dataset was created with the hope that temporal fluctuations in the sensor data of each device might enable machine learning algorithms to determine when a person is near one of the devices. You can read further about this dataset at Kaggle using the link provided below. The dataset has been organized and structured as a `csv` file within Kaggle, which is being provided to you as part of this exercise.

- **Kaggle dataset link:** `https://bit.ly/environmental-sensor-data`
- **Dataset** `csv` **filename:** `iot_telemetry_dataset.csv`

1.1. Load the dataset from the `csv` file and carry out basic exploration of data by addressing the following questions.

(a) (3 points) Based on your reading of the dataset from Kaggle, is this a supervised machine learning task or an unsupervised machine learning task? Justify your answer as much as possible in a text cell.

(b) (1 point) How many data samples are there in the dataset? *Note:* All such questions must be answered programmatically in one or more code cells.

(c) (1 point) How many samples are associated with the device having MAC address `00:0f:00:70:91:0a`?

(d) (1 point) How many samples are associated with the device having MAC address `1c:bf:ce:15:ec:4d`?

(e) (1 point) How many samples are associated with the device having MAC address `b8:27:eb:bf:9d:51`?

1.2. We now turn our attention to preprocessing of the dataset, which includes one-hot encoding of categorical variables and standardization of non-categorical variables that don't represent time. Note that no further preprocessing is needed for this data since the dataset does not have any missing entries.[1]

(a) (3 points) Provide two *Grouped Bar Charts*, with grouping using the three devices, for the means and variances associated with `co`, `humidity`, `lpg`, `smoke`, and `temp` variables. Comment on any observations that you can make from these charts in a text cell.

(b) (6 points) *Standardize* the dataset by making the data associated with `co`, `humidity`, `lpg`, `smoke`, and `temp` variables zero mean and unit variance. Such standardization, however, must be done separately for data associated with each device. This is an important lesson for practical purposes, as data samples associated with different devices cannot be thought of as having the same mean and variance.

(c) (3 points) One-hot encode the categorical variables of `device`, `light`, and `motion`. In this exercise (and subsequent exercises), you are allowed to use `pandas.get_dummies()` method for this purpose.

(d) (1 point) Print the first 20 samples of the preprocessed data (e.g., using the `pandas.DataFrame.head()` method).

(e) (1 point) Why do you think the `ts` variable in the dataset has not been touched during preprocessing? Comment as much as you can in a text cell.

1.3. (5 points) Map the `co`, `humidity`, `lpg`, `smoke`, and `temp` variables for each data sample into the following six independent features:

1. mean of the five independent variables (e.g., use `mean()` function in either `pandas` or `numpy`)
2. geometric mean of the five independent variables (e.g., use `gmean()` function in `scipy.stats`)
3. harmonic mean of the five independent variables (e.g., use `hmean()` function in `scipy.stats`)
4. variance of the five independent variables (e.g., use `var()` function in either `pandas` or `numpy`)
5. kurtosis of the five independent variables (e.g., use `kurtosis()` function in `scipy.stats`)
6. skewness of the five independent variables (e.g., use `skew()` function in `scipy.stats`)

Print the first 40 samples of the transformed dataset (e.g., using the `pandas.DataFrame.head()` method), which has the six features calculated from the five original independent variables.

Remark 1: One of the things you will notice is that there are some terminologies being used in descriptions of the functions in `scipy.stats` that might not be familiar to you. It is my hope that you can try to get a handle on these terminologies by digging into resources such as `Wikipedia`, `Stack Overflow`, `Google`, etc. Helping you become comfortable with the idea of lifelong self-learning is one of the goals of this course.

2 Feature Learning for *Synthetically Generated Data*

In this part of the exercise, we will focus on 'feature learning' using *Principal Component Analysis* (PCA). In order to grasp the basic concepts underlying PCA, we limit ourselves in this exercise to synthetically generated three-dimensional data samples (i.e., $p = 3$) that actually lie on a two-dimensional subspace (i.e., $k = 2$).

[1]Be aware that real-world data in most problems is never this nice!

2.1. In order to create synthetic data in $\mathbb{R}^3$ that lies on a two-dimensional subspace, we need basis vectors (i.e., a basis) for the two-dimensional subspace. You will generate such a basis (matrix) *randomly*, as follows.

(a) (2 points) Create a matrix $\mathbf{B} \in \mathbb{R}^{3\times 2}$ whose individual entries are drawn from a Gaussian distribution with mean 0 and variance 1 in an independent and identically distributed (iid) fashion. While this can be accomplished in a number of ways in Python, you might want to use `numpy.random.randn()` method for this purpose. Once generated, this matrix should not be changed for the rest of this exercise.

(b) (1 point) Matrices with iid Gaussian entries are always *full* rank, which makes the matrix $\mathbf{B}$ a basis matrix whose column space is a two-dimensional subspace in $\mathbb{R}^3$. Verify this by printing the rank of $\mathbf{B}$; it should be 2. *Note:* `numpy.linalg` package (`https://numpy.org/doc/stable/reference/routines.linalg.html`) is one of the best packages for most linear algebra operations in Python.

(c) (1 point) Note that the basis vectors in $\mathbf{B}$ are neither unit-norm, nor orthogonal to each other. Verify this by printing the norm of each vector in $\mathbf{B}$ as well as the inner product between the two vectors in $\mathbf{B}$.

(d) (1 point) Let $\mathbb{S}$ denote the subspace corresponding to the column space of the matrix $\mathbf{B}$. Generate and print three unique vectors that lie in the subspace $\mathbb{S}$.

2.2. We now turn our attention to generation of synthetic data. We will resort to a 'random' generation mechanism for this purpose. Specifically, each of our (unlabeled) data sample $\mathbf{x} \in \mathbb{R}^3$ is going to be generated as follows: $\mathbf{x} = \mathbf{B}\mathbf{v}$, where $\mathbf{v} \in \mathbb{R}^2$ is a random vector whose entries are iid Gaussian with mean 0 and variance 1. Note that we will have a different $\mathbf{v}$ for each new data sample (i.e., unlike $\mathbf{B}$, it is **not** fixed for each data sample).

(a) (2 points) Generate 200 data samples $\{\mathbf{x}_i\}_{i=1}^{200}$ using the aforementioned mathematical model.

(b) (1 point) Does each data sample $\mathbf{x}_i$ lie in the subspace $\mathbb{S}$? Justify your answer in a text cell.

(c) (3 points) Store the data samples into a *data matrix* $\mathbf{X} \in \mathbb{R}^{n\times p}$ such that each data sample is a row in this data matrix. What is n and p in this case? Print the dimensionality of $\mathbf{X}$ and confirm it matches your answer.

(d) (1 point) Since we can write $\mathbf{X}^{\mathrm{T}} = \mathbf{B}\mathbf{V}$, where $\mathbf{V} \in \mathbb{R}^{2\times 200}$ is a matrix whose columns are the vectors $\mathbf{v}_i$'s corresponding to data samples $\mathbf{x}_i$'s, the rank of $\mathbf{X}$ is 2 *(Can you see why? Perhaps refer to Wikipedia?)*. Verify this by printing the rank of $\mathbf{X}$.

2.3. Before turning our attention to calculation of PCA features for our data samples, we first investigate the relationship between eigenvectors of the scaled covariance matrix $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ and the right singular vectors of $\mathbf{X}$.

(a) Compute the *singular value decomposition* (SVD) of $\mathbf{X}$ and the *eigenvalue decomposition* (EVD) of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ and verify (by printing) that:

i. (2 points) The right singular vectors of $\mathbf{X}$ correspond to the eigenvectors of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$. *Hint:* Recall that eigenvalue decomposition does not necessarily list the eigenvalues in decreasing order. You would need to be aware of this fact to appropriately match the eigenvectors and singular vectors.

ii. (2 points) The eigenvalues of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ are square of the singular values of $\mathbf{X}$.

iii. (2 points) The *energy* in $\mathbf{X}$, defined by $\|\mathbf{X}\|_F^2$, is equal to sum of squares of the singular values of $\mathbf{X}$.

(b) Since the rank of $\mathbf{X}$ is 2, it means that the entire dataset spans only a two-dimensional subspace in $\mathbb{R}^3$. We now dig a bit deeper into this.

i. (2 points) Since rank of $\mathbf{X}$ is 2, we should ideally only have two nonzero singular values of $\mathbf{X}$. However, unless you are really lucky, you will see that none of your singular values are exactly zero. Comment on why that might be happening (and if you are the lucky one then run your code again and you will hopefully become unlucky :).

ii. (3 points) What do you think is the relationship between the right singular vectors of $\mathbf{X}$ corresponding to the two largest singular values and the subspace $\mathbb{S}$? Try to be as precise and mathematically rigorous as you can.

2.4. We finally turn our attention to PCA of the synthetic dataset, which is stored in matrix $\mathbf{X}$. Our focus in this problem is on computing of the PCA features for $k = 2$, computation of projected data (also termed *reconstructed* data), and the sum of squared errors (also termed *reconstruction error*, *approximation error*, *representation error*, or *PCA error*).

(a) (2 points) Since each data sample $\mathbf{x}_i$ lies in a three-dimensional space, we can have up to three principal components of this data. However, based on your knowledge of how the data was created (and subsequent discussion above), how many principal components should be enough to capture all variation in the data? Justify your answer as much as you can, especially in light of the discussion in class.

(b) While *mean centering* is an important preprocessing step for PCA, we do not necessarily need to carry out mean centering in this problem since the mean vector for this dataset will have very small entries. Indeed, if we let x_1, x_2, and x_3 denote the first, second, and third component of the random vector $\mathbf{x}$ then it follows that $\mathbb{E}[x_k] = 0, k = 1, 2, 3$.

   i. (3 points) Formally show that $\mathbb{E}[x_k] = 0, k = 1, 2, 3$, for our particular data generation method.

   ii. (2 points) Compute the (empirical) mean vector $\widehat{\boldsymbol{\mu}}$ from the data matrix $\mathbf{X}$ and verify by printing that its entries are indeed small.

(c) (3 points) Compute the top two principal component directions (loading vectors) $\mathbf{U} = \begin{bmatrix}\mathbf{u}_1 & \mathbf{u}_2\end{bmatrix}$ of this dataset and print them.

(d) (2 points) Compute feature vectors $\mathbf{x}_{i,\mathrm{f}}$ from data samples $\mathbf{x}_i$ by 'projecting' data onto the top two principal component directions of $\mathbf{X}$.

(e) (2 points) Reconstruct (approximate) the original data samples $\mathbf{x}_i$ from the PCA feature vectors $\mathbf{x}_{i,\mathrm{f}}$ by computing $\widehat{\mathbf{x}}_i = \mathbf{U}\mathbf{x}_{i,\mathrm{f}}$.

(f) (2 points) Ideally, since the data comes from a two-dimensional subspace, the *reconstruction error* (*aka*, the sum of squared errors)

$$\sum_{i=1}^{n} \|\widehat{\mathbf{x}}_i - \mathbf{x}_i\|_2^2 = \|\widehat{\mathbf{X}} - \mathbf{X}\|_F^2$$

should be zero. Verify (unless, again, you are super lucky) that this is, in fact, not the case. This error, however, is so small that it can be treated as zero for all practical purposes.

(g) (2 points) Now compute feature vectors $\mathbf{x}_{i,\mathrm{f}}$ from data samples $\mathbf{x}_i$ by projecting data onto *only* the top principal component direction of $\mathbf{X}$.

(h) (2 points) Reconstruct (approximate) the original data samples $\mathbf{x}_i$ from the PCA feature vectors $\mathbf{x}_{i,\mathrm{f}}$ by computing $\widehat{\mathbf{x}}_i = \mathbf{u}_1\mathbf{x}_{i,\mathrm{f}}$.

(i) (2 points) Compute the representation error $\|\widehat{\mathbf{X}} - \mathbf{X}\|_F^2$ and show that this error is equal to the square of the second-largest singular value of $\mathbf{X}$.

(j) (4 points) Using `mpl_toolkits.mplot3d`, display two 3D scatterplots corresponding to the original data samples $\mathbf{x}_i$ and the reconstructed data samples $\widehat{\mathbf{x}}_i$ corresponding to the top principal component. Comment on the shape of the scatterplot for the reconstructed samples and the mathematical reason for this shape.