ECON526: Problem Set 5

Jesse Perla

Name: _____

Student ID: _____

Instructions

- Ensure you modify the field above with your **name and student number above immediately**
- Modify directly and do not rename the file as Canvas will automatically append your name to it.
- Submit both the `.ipynb` and an exported `html` version of this file.
- See here for instructions.
- Do not change the filenames. Canvas automatically appends your name/student number to submitted files.

Setup

Feel free to use the following packages (and we have added a few convenience imports)

```
import numpy as np
import matplotlib.pyplot as plt
import scipy
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from numpy.linalg import norm
```

```
from scipy.linalg import inv, solve, eig, eigvalsh, cholesky
np.random.seed(77)
```

Question 1

Take the following code from our PCA examples in class:

```
N = 50 # number of observations
L, M = 2, 3 # number of latent factors and observed features
Z = np.random.randn(N, L) # latent factors
F = np.array([[1.0, 0.05], # X_1 = Z_1 + 0.05 Z_2
              [2.0, 0.0], # X_2 = 2 Z_1
              [3.0, 0.1]]) # X_3 = 3 Z_1 + 0.1 Z_2
X = Z @ F.T + 0.1 * np.random.randn(N, M) # added noise
pca = PCA(n_components=2) # one less, and correctly specified
Z_hat = pca.fit_transform(X) # transformed by dropping last factor
# Sign is arbitrary, and the components can mix the latent factors
print(f"Correlation of Z_1 to Z_hat_1 = {np.corrcoef(Z.T, Z_hat.T)[0,2]}")
print(f"Correlation of Z_2 to Z_hat_2 = {np.corrcoef(Z.T, Z_hat.T)[1,3]}")
```

```
Correlation of Z_1 to Z_hat_1 = 0.999329900857125
Correlation of Z_2 to Z_hat_2 = -0.479077909271834
```

Question 1.1

We said that it was doing a poor job at separating the 2nd latent factor from the noise. Show code to demonstrate this and then explain why you think it is happening.

```
# modify here
```

Answer:

(double click to edit your answer)

Question 1.2

Choose a new loading matrix `F_new` and build `X_new = Z @ F_new.T + 0.1 * np.random.randn(N, M)`, keeping `Z`, the noise scale, and the other parameters as they are, so that the data more clearly has 2 latent factors as specified. Leave the original `F` and `X` alone, since later questions use them. Hint: get better separation between the 2nd and 3rd singular values.

```
# modify here
```

Answer:

(double click to edit your answer)

Question 1.3

This example uses the same F as the original.

```
X_tiny = Z @ F.T + 1e-16 * np.random.randn(N, M)
pca_tiny = PCA(n_components=3).fit(X_tiny)
print(f"singular values: {pca_tiny.singular_values_}")
print(f"explained variance ratio: {pca_tiny.explained_variance_ratio_}")
```

```
singular values: [24.83798261  0.39483058  0.        ]
explained variance ratio: [9.99747373e-01 2.52626671e-04 0.00000000e+00]
```

Explain the key differences of these results compared to the baseline.

Answer:

(double click to edit your answer)

Question 2

PCA with L components is a linear auto-encoder: with the data demeaned, the encoder is $z = W^\top x$ and the decoder is $\hat{x} = Wz$ with $W = V_{1:L}$. In scikit-learn, `pca.transform` is the encoder and `pca.inverse_transform` is the decoder; both handle the demeaning for you.

Question 2.1

Using `X` from Question 1, fit `PCA(n_components=L)` for $L = 1, 2, 3$. For each, reconstruct $\hat{X}$ by encoding and decoding and compute the reconstruction error $\sum_{n=1}^{N} \|\hat{x}_n - x_n\|_2^2$. Compare each error to the sum of the squared singular values that were dropped, $\sum_{m>L} \sigma_m^2$, from a `PCA(n_components=3)` fit.

```
# modify here
```

Question 2.2

Explain the relationship you found. Why does the error reach zero (up to roundoff) at $L = 3$, and what does the comparison between $L = 1$ and $L = 2$ say about how many factors to keep?

Answer:

(double click to edit your answer)

Question 3

Question 3.1

Take a new mean μ and covariance matrix Σ. You want to find a matrix A such that $Y = \mu + AX$ has that mean and covariance for $X \sim \mathcal{N}(0, I)$, as in the probability lecture.

Take the matrix below, its eigenvalues, and the result of calling Cholesky. If `cholesky` raises an error on your machine, that is part of the answer. Explain why you probably should not rely on the Cholesky factor in this case.

```
mu = np.array([0, 1, 1])
Sigma = np.array([
    [26, 21, 6],
    [21, 17, 5],
    [6, 5, 2]
]) / 30.0
C = cholesky(Sigma, lower=True)
print(C)
print(f"Eigenvalues(Sigma) = {eigvalsh(Sigma)}")
```

```
[[9.30949336e-01 0.00000000e+00 0.00000000e+00]
 [7.51920618e-01 3.58057437e-02 0.00000000e+00]
 [2.14834462e-01 1.43222975e-01 5.75915904e-08]]
Eigenvalues(Sigma) = [-4.09003431e-17  1.95131000e-02  1.48048690e+00]
```

Answer:

(double click to edit your answer)

Question 3.2

Your friend suggests an SVD of the covariance matrix, which is always defined. Here are the results for $USV^\top = \Sigma$,

```
U, S, Vh = scipy.linalg.svd(Sigma) # returns V' not V
print(f"U =\n {U}")
print(f"S =\n {S}")
print(f"V' =\n {Vh}")
```

```
U =
 [[-0.76451208  0.26337697  0.58834841]
 [-0.61865349 -0.04339637 -0.78446454]
 [-0.18107771 -0.96371641  0.19611614]]
S =
 [1.48048690e+00 1.95131000e-02 2.83470951e-17]
V' =
 [[-0.76451208 -0.61865349 -0.18107771]
 [ 0.26337697 -0.04339637 -0.96371641]
 [ 0.58834841 -0.78446454  0.19611614]]
```

Your friend then shows you the following

```
U_truncated = U[:, :2] # only the first two columns
S_truncated = np.diag(np.sqrt(S[:2])) # first two singular values
A = U_truncated @ S_truncated
print(f"shape(A) = {A.shape}")
print(f"A =\n{A}")
print(f"||A A' - Sigma|| = {norm(A @ A.T - Sigma)}")
```

```
shape(A) = (3, 2)
A =
[[-0.93022207  0.03679094]
 [-0.75274824 -0.00606201]
 [-0.22032678 -0.13462087]]
||A A' - Sigma|| = 1.1664773918553167e-15
```

Explain why this is a great observation and how it might be useful for transforming the random variable $Y = \mu + A\hat{X}$ for some unit normal $\hat{X}$.

Answer:

(double click to edit your answer)

Question 3.3

Finally, your friend uses this to generate random values and compares the two approaches.

```
N_draws = 100000
Y_draws = np.random.multivariate_normal(mu, Sigma, size=N_draws)
X_hat_draws = np.random.multivariate_normal(np.zeros(2), np.eye(2),
                                            size=N_draws)
Y_draws_hat = mu + X_hat_draws @ A.T # using the A from the SVD
print(f"||mean(Y_draws) - mu|| = {norm(np.mean(Y_draws, axis=0) - mu)}")
print(f"||mean(Y_draws_hat) - mu|| = {norm(np.mean(Y_draws_hat, axis=0) - mu)}")
cov_draws = np.cov(Y_draws, rowvar=False)
cov_draws_hat = np.cov(Y_draws_hat, rowvar=False)
print(f"||cov(Y_draws) - Sigma|| = {norm(cov_draws - Sigma)}")
print(f"||cov(Y_draws_hat) - Sigma|| = {norm(cov_draws_hat - Sigma)}")
```

```
||mean(Y_draws) - mu|| = 0.0007320975941730382
||mean(Y_draws_hat) - mu|| = 0.0019852311518718893
||cov(Y_draws) - Sigma|| = 0.006362725673611749
||cov(Y_draws_hat) - Sigma|| = 0.008846999449467891
```

This worked, but your friend is confused: `Y_draws` came from a three-dimensional normal, while `Y_draws_hat` only used two dimensions of randomness. How can it deliver the same mean and covariance? Explain it to your friend.

Answer:

(double click to edit your answer)

Question 4

Find some economic or financial data on the web of your choice

1. Download it and load it into a pandas dataframe
2. Either run PCA or k-means clustering on the data
3. Briefly interpret the results in whatever way you wish

We are not giving you a list of possible data sources because we want you to explore and look for interesting sources of data.

The economic rationale for the dimension reduction or the clustering does not need to be especially tight, but do your best to explain why you ran the particular specification. Remember to standardize the features if their units are not comparable. A small table, one fit, and a short interpretation is enough; this is not meant to be a data-cleaning exercise. Feel free to use an AI to help with loading and cleaning the data; the choice of features, the specification, and the interpretation are yours.

For the data, post a link to where you found it, but there is no need to have it automatically downloaded in the notebook, and no need to submit it with your problem set.

```
# modify here
```

Answer:

(double click to add a brief interpretation or summary)