CS 182: Introduction to Machine Learning, Fall 2021
Homework 4

(Due on Tuesday, Nov. 23 at 11:59pm (CST))

Notice:

- Please submit your assignments via Gradescope. The entry code is KYJ626.
- Please make sure you select your answer to the corresponding question when submitting your assignments.
- Each person has a total of five days to be late without penalty for all the homeworks. Each late delivery less than one day will be counted as one day.

1. [20 points] *(Decision Trees)*
   Please build a classification tree based on the entropy impurity measure to classify the sample of colonies in the following table.

Index	Edge	Color	Size	Texture	Type
1	Rounded	White	Small	Rough	Bacteria
2	Rounded	White	Small	Smooth	Bacteria
3	Jagged	White	Small	Rough	Fungus
4	Villiform	Yellow	Small	Rough	Fungus
5	Villiform	Green	Big	Rough	Fungus
6	Villiform	Green	Big	Smooth	Bacteria
7	Jagged	Green	Big	Smooth	Fungus
8	Rounded	Yellow	Small	Rough	Bacteria
9	Rounded	Green	Big	Rough	Fungus
10	Villiform	Yellow	Big	Rough	Fungus
11	Rounded	Yellow	Big	Smooth	Fungus
12	Jagged	Yellow	Small	Smooth	Fungus
13	Jagged	White	Big	Rough	Fungus
14	Villiform	Yellow	Small	Smooth	Bacteria

Solution:

$$\text{Entropy}(S) = -\frac{5}{14}\log\frac{5}{14} - \frac{9}{14}\log\frac{9}{14} = 0.94$$

$$\text{Gain}(S,\text{Edge}) = 0.94 - \frac{5}{14}\left(-\frac{3}{5}\log\frac{3}{5} - \frac{2}{5}\log\frac{2}{5}\right) + \frac{5}{14}\left(-\frac{3}{5}\log\frac{3}{5} - \frac{2}{5}\log\frac{2}{5}\right) = 0.25$$

$$\text{Gain}(S,\text{Color}) = 0.94 - \frac{2}{7} + \frac{3}{7}\left(-\frac{1}{3}\log\frac{1}{3} - \frac{2}{3}\log\frac{2}{3}\right) + \frac{2}{7}\left(-\frac{1}{4}\log\frac{1}{4} - \frac{3}{4}\log\frac{3}{4}\right) = 0.03$$

$$\text{Gain}(S,\text{Size}) = 0.94 - \frac{1}{2}\left(-\frac{3}{7}\log\frac{3}{7} - \frac{4}{7}\log\frac{4}{7}\right) + \frac{1}{2}\left(-\frac{1}{7}\log\frac{1}{7} - \frac{6}{7}\log\frac{6}{7}\right) = 0.15$$

$$\text{Gain}(S,\text{Texture}) = 0.94 - \frac{4}{7}\left(-\frac{1}{4}\log\frac{1}{4} - \frac{3}{4}\log\frac{3}{4}\right) + \frac{3}{7} = 0.05$$

Therefore, we choose Edge ad the first feature.

$$\text{Entropy}(S, \text{Edge} = \text{Round}) = -\frac{3}{5}\log\frac{3}{5} - \frac{2}{5}\log\frac{2}{5} = 0.97$$
$$\text{Gain}(S, \text{Edge} = \text{Round}, \text{Color}) = 0.97 - \frac{2}{5} = 0.57$$
$$\text{Gain}(S, \text{Edge} = \text{Round}, \text{Size}) = 0.97$$
$$\text{Gain}(S, \text{Edge} = \text{Round}, \text{Texture}) = 0.97 - \frac{3}{5}\left(-\frac{1}{3}\log\frac{1}{3} - \frac{2}{3}\log\frac{2}{3}\right) - \frac{2}{5} = 0.02$$

Therefore, we choose Size for Edge = Round.

$$\text{Entropy}(S, \text{Edge} = \text{Villiform}) = -\frac{3}{5}\log\frac{3}{5} - \frac{2}{5}\log\frac{2}{5} = 0.97$$
$$\text{Gain}(S, \text{Edge} = \text{Villiform}, \text{Color}) = 0.97 - \frac{3}{5}\left(-\frac{1}{3}\log\frac{1}{3} - \frac{2}{3}\log\frac{2}{3}\right) - \frac{2}{5} = 0.02$$
$$\text{Gain}(S, \text{Edge} = \text{Villiform}, \text{Size}) = 0.97 - \frac{3}{5}\left(-\frac{1}{3}\log\frac{1}{3} - \frac{2}{3}\log\frac{2}{3}\right) - \frac{2}{5} = 0.02$$
$$\text{Gain}(S, \text{Edge} = \text{Villiform}, \text{Texture}) = 0.97$$

Therefore, we choose Size for Edge = Villiform.
In conclusion the decision tree is shown in Figure 1.

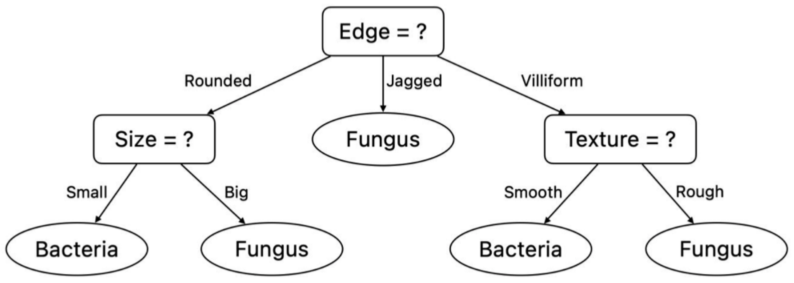


Figure 1: 1

2. [20 points] *(Nonparametric Density Estimation)*
Let the i.i.d. sample $\mathcal{X} = \{x_i\}_{i=1}^n$ be drawn from some unknown probability density $p(x)$ with $x \in [0,1]$. We can obtain the histogram estimator $\hat{p}(x)$ for $p(x)$ with bin width specified as h. We define the loss of estimation error in the L^2 space:

$$L(h) = \int_0^1 \left((\hat{p}(x) - p(x)\right)^2 \mathsf{d}x = \int_0^1 \hat{p}^2(x)\mathsf{d}x - 2\int_0^1 \hat{p}(x)p(x)\mathsf{d}x + \int_0^1 p^2(x)\mathsf{d}x.$$

Considering the last term $\int p^2(x)\mathsf{d}x$ is uncorrelated with $\hat{p}(x)$ and replacing the integral with the average, we get

$$L'(h) = \int_0^1 \hat{p}^2(x)\mathsf{d}x - \frac{2}{n}\sum_{i=1}^n \hat{p}(x_i).$$

Please derive

(a) the expression of $\hat{p}(x)$;

(b) the expression of $L'(h)$ based on the histogram estimator $\hat{p}(x)$;

(c) the h that minimizes $L'(h)$.

Solution:

(a) The expression of $\hat{p}(x)$ is

$$\hat{p}(x) = \frac{\#\left\{x^{(\ell)} \text{ in the same bin as } x\right\}}{nh}.$$

(b) Denote $m = \frac{1}{h}$ and Z_j as the number of x_i in the j bin. Since

$$\frac{1}{n}\sum_{i=1}^n \hat{p}(x_i) = \frac{1}{n}\sum_{j=1}^m \sum_{i \in B_j} \hat{p}(x_i) = \frac{1}{n}\sum_{j=1}^m Z_j \hat{p}(x_i) = \frac{1}{n}\sum_{j=1}^m \frac{Z_j^2}{nh},$$

we have

$$\begin{aligned} L'(h) &= \int_0^1 \hat{p}^2(x)\mathsf{d}x - \frac{2}{n}\sum_{i=1}^n \hat{p}(x_i) \\ &= \sum_{j=1}^m \int_{(j-1)h}^{jh} \hat{p}^2(x)\mathsf{d}x - \frac{2}{n}\sum_{j=1}^m \frac{Z_j^2}{nh} \\ &= \sum_{j=1}^m \int_{(j-1)h}^{jh} (\frac{Z_j}{nh})^2\mathsf{d}x - \frac{2}{n}\sum_{j=1}^m \frac{Z_j^2}{nh} \\ &= -\frac{1}{n^2h}\sum_{j=1}^m Z_j^2. \end{aligned}$$

(c) Because $\sum_{j=1}^m Z_j^2 \geq \sum_{j=1}^m Z_j = n$ and $\sum_{j=1}^m Z_j^2 \leq \sum_{j=1}^m Z_j n = n^2$, we have

$$-\frac{1}{h} \leq L'(h) \leq -\frac{1}{nh}.$$

Therefore, when $h \to 0$, $L'(h) \to -\infty$.

3. [30 points] *(Programming Problem: EM for GMMs)*

   In this problem you will practice with EM algorithms for fitting GMMs, and the notebook EM.ipynb will walk you through implementing it.

   (*Note:* You need to export the notebook as a PDF file and then upload it onto Gradescope.)

EM

December 31, 2021

0.1 Fitting Gaussian Mixture Models (GMMs) with EM Algorithm

Given $x_1, \ldots, x_n$ that are generated from the following distribution

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(x \mid \mu_k, \Sigma_k\right),$$

where $\pi_1, \ldots, \pi_K$ are the mixing coefficients satisfy

$$\sum_{k=1}^{K} \pi_k = 1 \text{ and } \pi_k \geq 0, \forall k = 1, \ldots, K.$$

Our interest is to fit the underlying GMM model from the n observations.

We consider a latent variable model where a hidden variable z is introduced, which leads to

$$p(x) = \sum_{k=1}^{K} p(z = k) p(x \mid z = k)$$

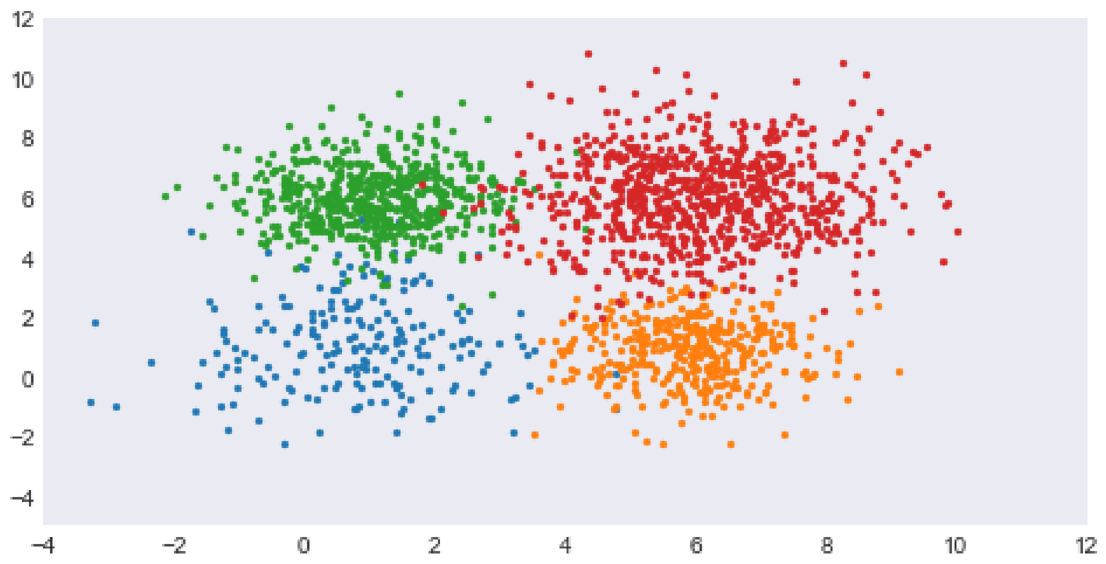
with $p(z = k) = \pi_k$ and $p(x \mid z = k) = \mathcal{N}\left(x \mid \mu_k, \Sigma_k\right)$. Then the MLE problem can be solved by the EM algorithm, which alternatively optimizes z and μ_k, Σ_k.

In the following, you will implement the EM algortihm to fit a GMM with a toy example.

First let us generate n datapoints from a underlying GMM model.

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Ellipse
from scipy.stats import multivariate_normal
plt.style.use('seaborn-dark')

n = 2000
K = 4
# Data Generation
X1 = np.random.multivariate_normal([1,1], np.diag([2,2]), 200)
X2 = np.random.multivariate_normal([6,1], np.diag([1,1]), 400)
X3 = np.random.multivariate_normal([1,6], np.diag([1,1]), 600)
X4 = np.random.multivariate_normal([6,6], np.diag([2,2]), 800)
X = np.vstack((X1, X2, X3,X4))
# Data Visualization
```

```
plt.figure(figsize=(8,4))
plt.axis([-4, 12, -5, 12])
plt.scatter(X1[:, 0], X1[:, 1], s=5)
plt.scatter(X2[:, 0], X2[:, 1], s=5)
plt.scatter(X3[:, 0], X3[:, 1], s=5)
plt.scatter(X4[:, 0], X4[:, 1], s=5)
plt.show()
```



With the n observations $x_1, \dots, x_n$, we then implement the EM algorithm to fit GMM. You need to implement the EM function and the likelihood function.

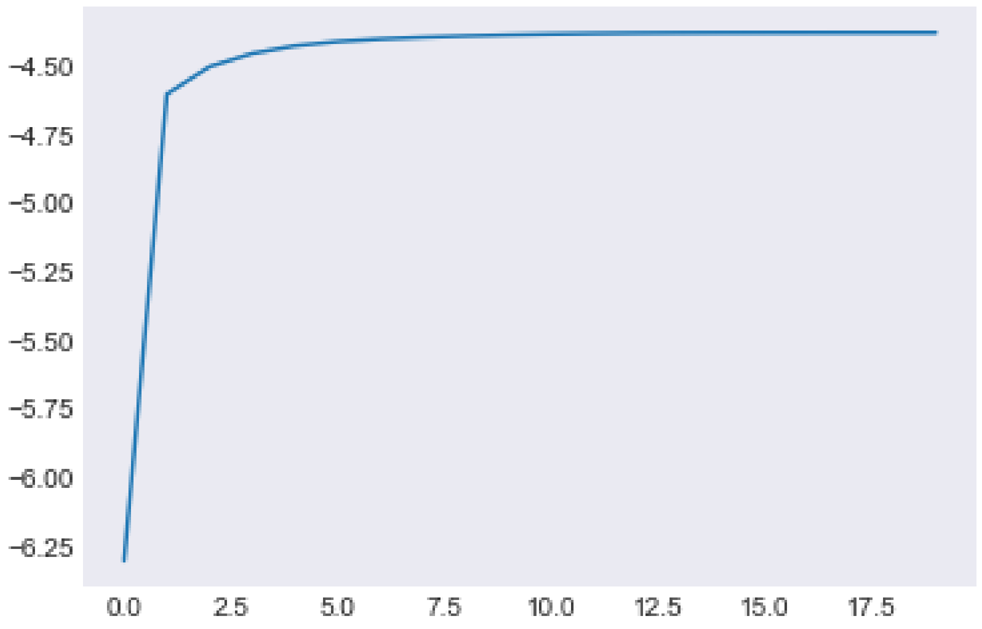
Note that you only need to implement two functions, that is, EM and likelihood. Do not modify any other existing codes or import any other packages, otherwise you will not get full points.

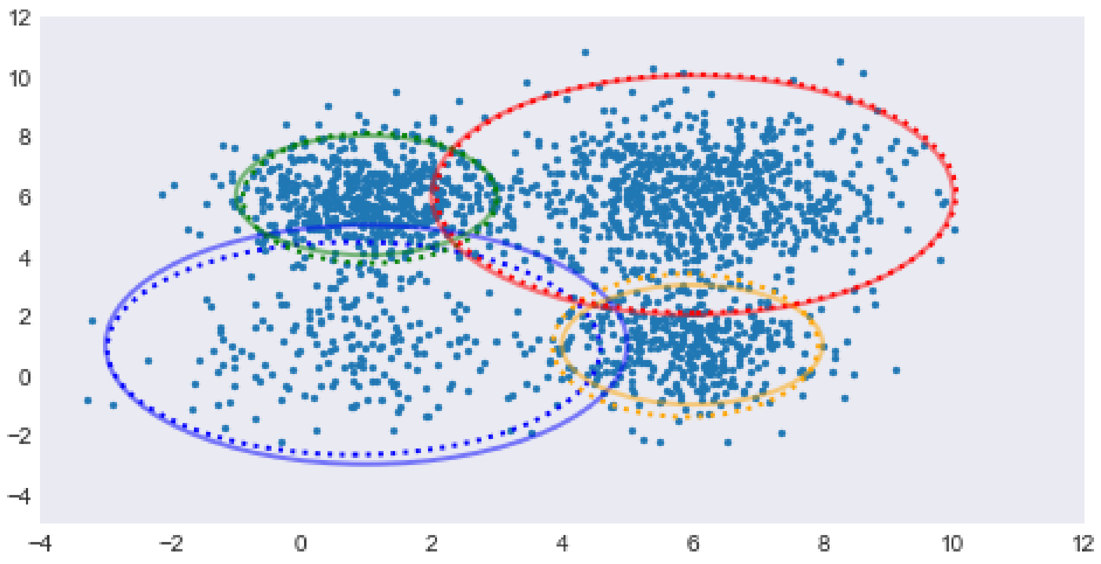
[2]:
```
# Initializations
z = np.ones((n, K)) / K
Pi = [1 / K] * 4
Mu = [[1,1], [7,2], [2, 7],[4,4]]
Var = [[1,1],[1,1],[1,1],[1,1]]
log_likelihood = []

# EM Algorithm
def EM(X, z, Pi, Mu, Var):
    # *****START OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****
    pdfs = np.zeros(((n, K)))
    for i in range(K):
        pdfs[:, i] = Pi[i] * multivariate_normal.pdf(X, Mu[i], np.diag(Var[i]))
    z = pdfs / pdfs.sum(axis=1).reshape(-1, 1)
    Pi = z.sum(axis=0) / z.sum()
    for i in range(K):
        Mu[i] = np.average(X, axis=0, weights=z[:, i])
        Var[i] = np.average((X - Mu[i]) ** 2, axis=0, weights=z[:, i])
    # *****END OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****
    return z, Pi, Mu, Var

# Calculate the log-likelihood
def likelihood(X, Pi, Mu, Var):
    # *****START OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****
    n_points, n_clusters = len(X), len(Pi)
    pdfs = np.zeros(((n_points, n_clusters)))
    for i in range(n_clusters):
        pdfs[:, i] = Pi[i] * multivariate_normal.pdf(X, Mu[i], np.diag(Var[i]))
    log_likelihood =  np.mean(np.log(pdfs.sum(axis=1)))
    # *****END OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****
    return log_likelihood

# Run EM algorithm for 20 iterates
for i in range(20):
    log_likelihood.append(likelihood(X, Pi, Mu, Var))
    z, Pi, Mu, Var = EM(X, z, Pi, Mu, Var)
    print('log-likehood:%.3f'%log_likelihood[-1])
```

```
log-likehood:-6.303
log-likehood:-4.605
log-likehood:-4.504
log-likehood:-4.456
log-likehood:-4.429
log-likehood:-4.414
log-likehood:-4.404
log-likehood:-4.398
log-likehood:-4.393
log-likehood:-4.389
log-likehood:-4.386
log-likehood:-4.384
log-likehood:-4.383
log-likehood:-4.382
log-likehood:-4.382
log-likehood:-4.381
log-likehood:-4.381
log-likehood:-4.381
log-likehood:-4.381
log-likehood:-4.381
```

Run the following codes to visualize your results. If you implementation is right, then the log-likelihood should be monotonically increasing and your GMM model (dashed curve) should be very close to the underlying one (solid curve).

```
[3]: plt.plot(range(20),log_likelihood)
     Mu_Star = [[1,1], [6,1], [1, 6],[6,6]]
     Var_Star = [[2,2], [1,1], [1, 1],[2,2]]
     colors = ['b', 'orange', 'g', 'r']
     plt.figure(figsize=(8,4))
     plt.axis([-4, 12, -5, 12])
     plt.scatter(X[:, 0], X[:, 1], s=5)
     ax = plt.gca()
     for i in range(K):
         plot_args = {'fc': 'None', 'lw': 2, 'edgecolor': colors[i], 'ls': ':'}
         ellipse = Ellipse(Mu[i], 4 * Var[i][0], 4 * Var[i][1], **plot_args)
         ax.add_patch(ellipse)
     for i in range(K):
         plot_args = {'fc': 'None', 'lw': 2, 'edgecolor': colors[i], 'alpha': 0.5}
         ellipse = Ellipse(Mu_Star[i], 4* Var_Star[i][0], 4 * Var_Star[i][1],␣
      ↪**plot_args)
         ax.add_patch(ellipse)
     plt.show()
```




12
10
8
6
4
2
0
−2
−4
−4
−2
0
2
4
6
8
10
12

```
[ ]:
```

4. [30 points] *(PCA as An Autoencoder)*
Principal component analysis (PCA) and autoencoders are popular tools for dimension reduction in machine learning. In this problem, we look into the relation between PCA and the linear autoencoders.

(a) Given a sample matrix $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_M] \in \mathbb{R}^{d\times M}$ where each column denotes a d-dimensional zero-mean sample. The goal of PCA is to find an orthogonal matrix (transformation) $\mathbf{W} \in \mathbb{R}^{d\times r}$ $(r \leq d)$ which is the solution to

$$\underset{\mathbf{W}}{\text{maximize}} \ \text{Tr}(\mathbf{W}^T\mathbf{X}\mathbf{X}^T\mathbf{W}) \quad \text{s.t.} \ \mathbf{W}^T\mathbf{W} = \mathbf{I}_r,$$

where $\text{Tr}(\cdot)$ denotes the trace and $\mathbf{I}_r$ is an $r \times r$ identity matrix. Show that

i. when $r = 1$, $\mathbf{W}$ is exactly the eigenvector of $\mathbf{X}\mathbf{X}^T$ corresponding to its largest eigenvalue.

ii. $\mathbf{W}$ is also the solution to

$$\underset{\mathbf{W}}{\text{minimize}} \ \|\mathbf{X} - \mathbf{W}\mathbf{W}^T\mathbf{X}\|_F^2 \quad \text{s.t.} \ \mathbf{W}^T\mathbf{W} = \mathbf{I}_r,$$

where $\|\cdot\|_F$ is the Frobenius norm, i.e., $\|\mathbf{P}\|_F = \sqrt{\text{Tr}(\mathbf{P}^T\mathbf{P})}$.

(b) Consider a linear autoencoder (as a neural network) with a single hidden layer structure:

$$\mathbf{H} = \mathbf{A}_1\mathbf{X} + \mathbf{b}_1\mathbf{1}^T$$
$$\hat{\mathbf{X}} = \mathbf{A}_2\mathbf{H} + \mathbf{b}_2\mathbf{1}^T,$$

where $\mathbf{A}_1 \in \mathbb{R}^{r\times d}$ $(\mathbf{A}_2 \in \mathbb{R}^{d\times r})$ and $\mathbf{b}_1 \in \mathbb{R}^{r\times 1}$ $(\mathbf{b}_2 \in \mathbb{R}^{d\times 1})$ are respectively the weight matrix and the bias vector of the layer in the encoder (decoder), and $\mathbf{1} = [1, ..., 1]^T \in \mathbb{R}^M$. One trains the linear autoencoder by minimizing the reconstruction error:

$$\{\mathbf{A}_1^\star, \mathbf{b}_1^\star, \mathbf{A}_2^\star, \mathbf{b}_2^\star\} = \underset{\mathbf{A}_1,\mathbf{b}_1,\mathbf{A}_2,\mathbf{b}_2}{\text{argmin}} \ \|\mathbf{X} - \hat{\mathbf{X}}\|_F^2.$$

Show that

i. $\mathbf{A}_2^\star$ can be solved from:

$$\underset{\mathbf{A}_2}{\text{minimize}} \ \|\mathbf{X} - \mathbf{A}_2\mathbf{A}_2^\dagger\mathbf{X}\|_F^2,$$

where $\mathbf{A}_2^\dagger = (\mathbf{A}_2^T\mathbf{A}_2)^{-1}\mathbf{A}_2^T$ is the Moore-Penrose pseudoinverse of $\mathbf{A}_2$. (*Hint:* use the derivative of Frobenius norm and the fact that $\mathbf{A}_2^\dagger = \underset{\mathbf{A}_1}{\text{argmin}}\|\mathbf{X} - \mathbf{A}_2\mathbf{A}_1\mathbf{X}\|_F^2$)

ii. the solution $\mathbf{W}$ from (a) can be taken as the same as $\mathbf{A}_2^\star$. (*Hint:* you may prove it by showing the equivalence of the column spaces spanned by these two matrices)

Solution:

(a) Proof:

i. When $r = 1$, it is easy to show that $\mathbf{W}^T\mathbf{X}\mathbf{X}^T\mathbf{W}$ and $\mathbf{W}^T\mathbf{W}$ are scalars. Therefore, one can reformulate the problem as follows:

$$\underset{\mathbf{W}}{\text{maximize}} \ \mathbf{W}^T\mathbf{X}\mathbf{X}^T\mathbf{W} \quad \text{s.t.} \ \mathbf{W}^T\mathbf{W} = 1.$$

The Lagrangian is

$$L(\mathbf{W}, \lambda) = \mathbf{W}^T\mathbf{X}\mathbf{X}^T\mathbf{W} + \lambda(1 - \mathbf{W}^T\mathbf{W}),$$

where λ is the Lagrangian multiplier. Then, consider the stationarity condition in KKT conditions

$$\frac{\partial L}{\partial \mathbf{W}} = 2\mathbf{X}\mathbf{X}^T\mathbf{W} - \lambda\mathbf{W} = 0,$$

which implies that $\mathbf{X}\mathbf{X}^T\mathbf{W} = \lambda\mathbf{W}$. It means that the optimal $\mathbf{W}$ must be an eigevector of $\mathbf{X}\mathbf{X}^T$. Substituting $\mathbf{X}\mathbf{X}^T\mathbf{W} = \lambda\mathbf{W}$ into the original objective, maximizing $\mathbf{W}^T\mathbf{X}\mathbf{X}^T\mathbf{W}$ is equivalent to maximizing λ. Hence, the optimal $\mathbf{W}$ is the eigenvector of $\mathbf{X}\mathbf{X}^T$ corresponding to its largest eigenvalue.

ii. Using the definition of Frobenius norm and the properties of trace, the following derivations hold:

$$\begin{aligned}
\|\mathbf{X}-\mathbf{W}\mathbf{W}^T\mathbf{X}\|_F^2 &= \mathrm{Tr}\left((\mathbf{X}-\mathbf{W}\mathbf{W}^T\mathbf{X})^T(\mathbf{X}-\mathbf{W}\mathbf{W}^T\mathbf{X})\right)\\
&= \mathrm{Tr}\left(\mathbf{X}^T\mathbf{X}-2\mathbf{X}^T\mathbf{W}\mathbf{W}^T\mathbf{X}+\mathbf{X}^T\mathbf{W}\mathbf{W}^T\mathbf{W}\mathbf{W}^T\mathbf{X}\right)\\
&= \mathrm{Tr}\left(\mathbf{X}^T\mathbf{X}-2\mathbf{X}^T\mathbf{W}\mathbf{W}^T\mathbf{X}+\mathbf{X}^T\mathbf{W}\mathbf{W}^T\mathbf{X}\right)\\
&= \mathrm{Tr}\left(\mathbf{X}^T\mathbf{X}-\mathbf{X}^T\mathbf{W}\mathbf{W}^T\mathbf{X}\right)\\
&= \mathrm{Tr}\left(\mathbf{X}^T\mathbf{X}\right)-\mathrm{Tr}\left(\mathbf{X}^T\mathbf{W}\mathbf{W}^T\mathbf{X}\right)\\
&= \mathrm{Tr}\left(\mathbf{X}^T\mathbf{X}\right)-\mathrm{Tr}\left(\mathbf{W}^T\mathbf{X}\mathbf{X}^T\mathbf{W}\right).
\end{aligned}$$

Note that, given $\mathbf{X}$, $\mathrm{Tr}\left(\mathbf{X}^T\mathbf{X}\right)$ is a constant that is irrelevant with $\mathbf{W}$. Hence, minimizing $\|\mathbf{X}-\mathbf{W}\mathbf{W}^T\mathbf{X}\|_F^2$ is equivalent to maximizing $\mathrm{Tr}\left(\mathbf{W}^T\mathbf{X}\mathbf{X}^T\mathbf{W}\right)$. It implies that the solutions of these two optimization problems are same.

(b) Proof:

i. Let $\frac{\partial\|\mathbf{X}-\hat{\mathbf{X}}\|_F^2}{\partial\mathbf{b}_2}=\mathbf{0}$, the following condition holds:

$$-2(\mathbf{X}-\mathbf{A}_2\mathbf{A}_1\mathbf{X}-\mathbf{A}_2\mathbf{b}_1\mathbf{1}^T-\mathbf{b}_2\mathbf{1}^T)\mathbf{1}=\mathbf{0}.$$

Since each $\mathbf{X}$ represents a zero-mean sample matrix, one has $\mathbf{X}\mathbf{1}=\mathbf{0}$. Substituting it into the condition above gives

$$\mathbf{b}_2=\mathbf{A}_2\mathbf{b}_1.$$

Therefore, one can simplify $\|\mathbf{X}-\hat{\mathbf{X}}\|_F^2$ as $\|\mathbf{X}-\mathbf{A}_2\mathbf{A}_1\mathbf{X}\|_F^2$. Then using the fact that $\mathbf{A}_2^\dagger=\underset{\mathbf{A}_1}{\mathrm{argmin}}\|\mathbf{X}-\mathbf{A}_2\mathbf{A}_1\mathbf{X}\|_F^2$ gives

$$\mathbf{A}_2^\star\in\underset{\mathbf{A}_2}{\mathrm{argmin}}\ \|\mathbf{X}-\mathbf{A}_2\mathbf{A}_2^\dagger\mathbf{X}\|_F^2.$$

ii. Denote $\mathbf{W}^\star$ as the optimal solution in (a)i. Since $\mathbf{W}^T\mathbf{W}=\mathbf{I}_r$, we have

$$\mathbf{W}^\star\in\underset{\mathbf{W}}{\mathrm{argmin}}\ \|\mathbf{X}-\mathbf{W}\mathbf{W}^T\mathbf{X}\|_F^2=\underset{\mathbf{W}}{\mathrm{argmin}}\ \|\mathbf{X}-\mathbf{W}(\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T\mathbf{X}\|_F^2.$$

And from (b)i, we know

$$\mathbf{A}_2^\star\in\underset{\mathbf{A}_2}{\mathrm{argmin}}\ \|\mathbf{X}-\mathbf{A}_2(\mathbf{A}_2^T\mathbf{A}_2)^{-1}\mathbf{A}_2^T\mathbf{X}\|_F^2.$$

The key difference between $\mathbf{W}^\star$ and $\mathbf{A}_2^\star$ is that $\mathbf{W}^\star$ is orthogonal while $\mathbf{A}_2^\star$ is not. But they share a same column space. Hence, we can set $\mathbf{A}_2^\star$ as orthogonal to make $\mathbf{W}^\star=\mathbf{A}_2^\star$.