DDA3020 Homework 3

Due date: Dec 08, 2024

Instructions

- The **deadline** is 23:59, Dec 08, 2024.
- The weight of this assignment in the final grade is 20%.
- **Electronic submission**: Turn in solutions electronically via Blackboard. Be sure to submit your homework as one pdf file plus two python scripts. Please name your solution files as "DDA3020HW3_studentID_name.pdf", "HW3_yourID_Q1.ipynb" and "HW3_yourID_Q2.ipynb". (.py files also acceptable)
- Note that **late submissions** will result in discounted scores: 0-24 hours $\rightarrow$ 80%, 24-120 hours $\rightarrow$ 50%, 120 or more hours $\rightarrow$ 0%.
- Answer the questions in English. Otherwise, you'll lose half of the points.
- Collaboration policy: You need to solve all questions independently and collaboration between students is **NOT** allowed.

1 Written Problems (50 points)

1.1. K-Means, 25 points

1.1.1.(5 points) Consider the 2 datasets A and B. Each dataset is classified into k clusters, with centers marked X and cluster membership represented by different colors in the figure. For each dataset, exactly one clustering was generated by k-means with Euclidean distance. Select the image with clusters generated by k-means.

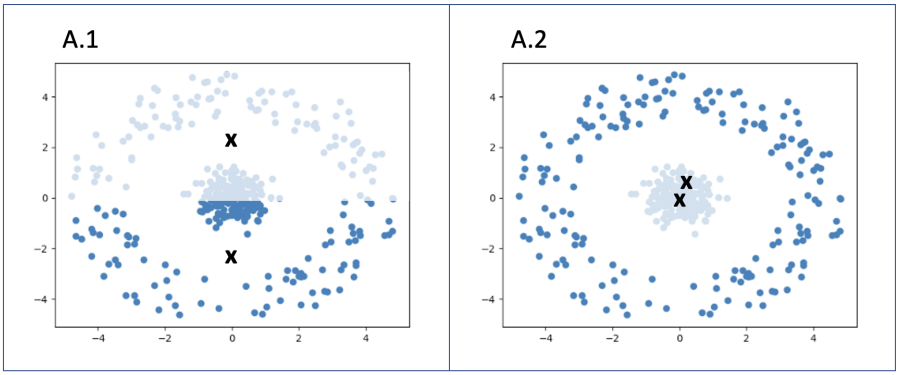


Figure 1: **Select one: A.1 or A.2**

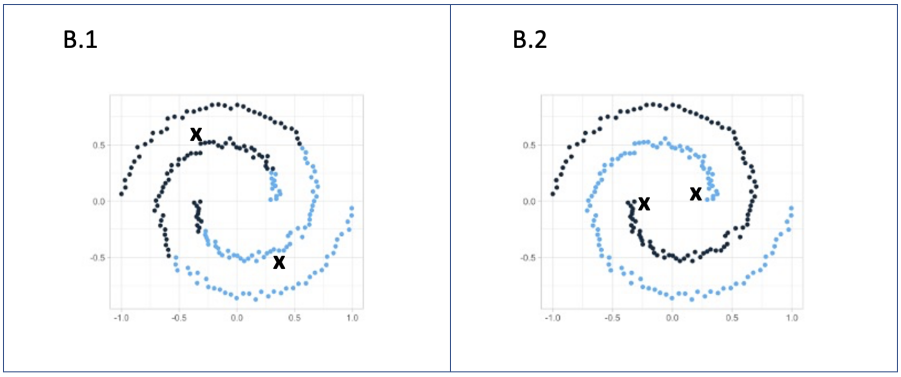


Figure 2: **Select one: B.1 or B.2**

1.1.2.(10 points) Consider a dataset $\mathcal{D}$ with 5 points as shown below. Perform a k-means clustering on this dataset with $k = 2$ using the Euclidean distance as the distance function. Remember that in the k-means algorithm, one iteration consists of following two steps: first, we assign each data point to its nearest cluster center; second, we recompute each center as the average of the data points assigned to it. Initially, the 2 cluster centers are chosen randomly as $\mu_0 = (5.3, 3.5)$, $\mu_1 = (5.1, 4.2)$. Parts (a) through (d) refer only to the first iteration of k-means clustering performed on

$\mathcal{D}$.

$$\mathcal{D} = \begin{bmatrix} 5.5 & 3.1 \\ 5.1 & 4.8 \\ 6.6 & 3.0 \\ 5.5 & 4.6 \\ 6.8 & 3.8 \end{bmatrix}$$

(a)Which of the following points will be the new center for cluster 0?

- (5.7 , 4.1)
- (5.6 , 4.8)
- (6.3 , 3.3)
- (6.7 , 3.4)

(b)Which of the following points will be the new center for cluster 1?

- (6.1 , 3.8)
- (5.5 , 4.6)
- (5.4 , 4.7)
- (5.3 , 4.7)

(c)How many points will belong to cluster 0, using the new centers?

(d)How many points will belong to cluster 1, using the new centers?

1.1.3.(5 points) Recall that in k-means clustering we attempt to find k cluster centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$ such that the total distance between each point and the nearest cluster center is minimized. We thus solve

$$\underset{\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_k}{\operatorname{argmin}} \sum_{i=1}^{N} \min_{j\in\{1,\ldots,k\}} \|\mathbf{x}^{(i)} - \boldsymbol{\mu}_j\|_2^2$$

where n is the number of data points. Instead of holding the number of clusters k fixed, your friend John tries to also minimize the objective over k, solving

$$\underset{k}{\operatorname{argmin}} \underset{\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_k}{\operatorname{argmin}} \sum_{i=1}^{N} \min_{j\in\{1,\ldots,k\}} \|\mathbf{x}^{(i)} - \boldsymbol{\mu}_j\|_2^2$$

You found this idea to be a bad one.

(a)What is the minimum possible value of the objective function when minimizing over k?

(b)What is a value of k for which we achieve the minimum possible value of the objective function when $N = 100$?

1.1.4.(5 points) Consider the following brute-force algorithm for minimizing the k-means objective: Iterate through each possible assignment of the points to k clusters, $\mathbf{z} = [z^{(1)}, \ldots, z^{(N)}]$. For each assignment $\mathbf{z} \in \{1, \ldots, k\}^N$, you evaluate the following objective function:

$$J(\mathbf{z}) = \underset{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k}{\operatorname{argmin}} \sum_{i=1}^{N} ||\mathbf{x}^{(i)} - \boldsymbol{\mu}_{z^{(i)}}||_2^2$$

At the end, you pick the assignment $\mathbf{z}$ that had lowest $J(\mathbf{z})$.

(a)Suppose we have N points and k clusters. For how many possible assignments $\mathbf{z}$ does the brute force algorithm have to evaluate $J(\mathbf{z})$?

(b)Suppose $N = 1000$, $k = 10$, and it takes us 0.01 seconds to evaluate $J(\mathbf{z})$ for a single assignment $\mathbf{z}$. How many seconds will the brute force algorithm take to check all assignments?

1.2. PCA (Principal Component Analysis), 25 points

(a). There are two derivations for the optimization problem of PCA in our lecture. The one (1) is to maximize the variance of the reconstructions and the other (2) is to minimize the the reconstruction error.

$$\max_{\mathbf{U}, \mathbf{U}^T\mathbf{U}=\mathbf{I}} \quad \frac{1}{N} \sum_{n=1}^{N} \|\tilde{\mathbf{x}}^{(n)} - \tilde{\boldsymbol{\mu}}\|^2, \tag{1}$$

$$\min_{\mathbf{U}, \mathbf{U}^T\mathbf{U}=\mathbf{I}} \quad \frac{1}{N} \sum_{n=1}^{N} \|\mathbf{x}^{(n)} - \tilde{\mathbf{x}}^{(n)}\|^2, \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^d$ denotes original data sample, $\tilde{\mathbf{x}} = \boldsymbol{\mu} + \mathbf{U}\mathbf{z}$ denotes the reconstruction of $\mathbf{x}$ and $\mathbf{U} \in \mathbb{R}^{d \times k}$, in which $\mathbf{z} = \mathbf{U}^T(\mathbf{x} - \boldsymbol{\mu})$ and $\boldsymbol{\mu} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}^{(n)}$, $\tilde{\boldsymbol{\mu}} = \frac{1}{N}\sum_{n=1}^{N} \tilde{\mathbf{x}}^{(n)}$ denote the mean of the data and reconstructions, respectively.

Prove the problem 1 and problem 2 are equivalent. In other words, you need to prove the following result:

$$\frac{1}{N} \sum_{n=1}^{N} \|\tilde{\mathbf{x}}^{(n)} - \tilde{\boldsymbol{\mu}}\|^2 + \frac{1}{N} \sum_{n=1}^{N} \|\mathbf{x}^{(n)} - \tilde{\mathbf{x}}^{(n)}\|^2 = \underbrace{\frac{1}{N} \sum_{n=1}^{N} \|\mathbf{x}^{(n)} - \boldsymbol{\mu}\|^2}_{\textbf{constant}}$$

(b). Let dataset $\mathbf{X} = \{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(n)}\}, \mathbf{x} \in \mathbb{R}^d$. Now, I want to project $\mathbf{X}$ into one-dimensional space using PCA that means I need to find a $\mathbf{w} \in \mathbf{R}^d$, s.t. $\mathbf{w}^T\mathbf{w} = 1$ to satisfy $z^{(i)} = \mathbf{w}^T(\mathbf{x}^{(i)} - \boldsymbol{\mu}) \in \mathbb{R}$, where $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}^{(i)}$ denotes the data mean. Prove the $\mathbf{w}$ is the eigenvector associated with the largest eigenvalue of covariance matrix $\boldsymbol{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}^{(i)} - \boldsymbol{\mu})(\mathbf{x}^{(i)} - \boldsymbol{\mu})^T$.

2 Programming (50 points)

2.1. PCA Coding, 25 points

Task Description You are asked to project data in the **iris** dataset into smaller subspaces, using PCA. It should be implemented **from scratch** in Python. Only numpy is allowed for calculation (Dataset loading and figure plotting are free to use other packages).

Datasets **iris** is exactly the one you have played with in SVM programming homework.

What you should do

1. **(1 points) Split training set and test set**. This task is to ensure all of you use the same dataset split. As with the task in SVM programming, you should first split the data into a training set and a test set. The training set should contain 70% of the samples, while the test set should include 30%. The number of samples from each category in both the training and test sets should reflect this 70-30 split; for each category, the first 70% of the samples will form the training set, and the remaining 30% will form the test set. Ensure that the split maintains the original order of the data. You should report instance ids in the split training set and test set. The output format is as follows:

   ```
   Q2.1.1 Split training set and test set:
   Training set:  xx

   Test set:  xx
   ```

   You should fill up xx in the template. You should write ids for each set in the same line with comma separated, e.g. `Training set:[1, 4, 19]`.

2. **(5 points) SVD decomposition**: Use the training set to calculate the mean vector and the covariance matrix, and perform SVD decomposition to derive the eigenvalues and eigenvectors. Following the slide (page 26 of Lecture 17), use a biased estimator for the covariance matrix (i.e. normalized by N rather than $N - 1$). The report format is as follows:

```
Q2.1.2 SVD decomposition:

Mean vector:  xx


Covariance matrix:  xx


Eigenvalues (vector):  xx


Eigenvectors (matrix):  xx
```

3. **(9 points) Project onto 1-dimensional subspace and reconstruct**: Based on the results from the previous problem, derive the projection matrix. Then, project the training set and the testing set onto the 1-dimensional subspace, and reconstruct them back to the original space, respectively. Calculate the variance (refer to Page 15 of Lecture 17) and the reconstruction loss (refer to Page 18 of Lecture 17). The report format is as follows:

```
Q2.1.3 Project onto 1-dimensional subspace and reconstruct:
Project matrix W: xx

shape of X_train_mapped:  xx, shape of X_train_reconstruct:  xx
variance_train:  xx
reconstruction_loss_train:  xx

shape of X_test_mapped:  xx, shape of X_test_reconstruct:  xx
variance_test:  xx
reconstruction_loss_test:  xx
```

4. **(2 points) Project onto 2-dimensional subspace and reconstruct**: Similar to the previous problem, but project the data onto a 2-dimensional subspace. The report format is as follows:

```
Q2.1.4 Project onto 2-dimensional subspace and reconstruct:
Project matrix W: xx

shape of X_train_mapped:  xx, shape of X_train_reconstruct:  xx
variance_train:  xx
reconstruction_loss_train:  xx

shape of X_test_mapped:  xx, shape of X_test_reconstruct:  xx
variance_test:  xx
reconstruction_loss_test:  xx
```

5. **(2 points) Project onto 3-dimensional subspace and reconstruct**: Similar to the previous problem, but project the data onto a 3-dimensional subspace. The report format is as follows:

```
Q2.1.5 Project onto 3-dimensional subspace and reconstruct:
Project matrix W: xx

shape of X_train_mapped: xx, shape of X_train_reconstruct: xx
variance_train: xx
reconstruction_loss_train: xx

shape of X_test_mapped: xx, shape of X_test_reconstruct: xx
variance_test: xx
reconstruction_loss_test: xx
```

6. **(6 points) Plotting**: Plot the curves of variances and reconstruction losses as a function of the projection dimension. The report format is as follows:

```
Q2.1.6 Plotting:
Dimension - Variance:
{figure}
{description of relationship}

Dimension - Reconstruction loss:
{figure}
{description of relationship}
```

Notably, each figure should contain two lines, one for training set, the other for the testing.

2.2. (K-Means, Mixture Models and EM Algorithms, 25 points)

– Datasets: **Abalone** (from UCI Repository: https://archive.ics.uci.edu/dataset/1/abalone)

– Task: Predict the age of abalones using KNN and GMM.

– Reference: https://www.kaggle.com/code/charel/learn-by-example-expectation-maximization/notebook.

– Note:

- Please follow the instructions below and implement tasks.
- Ensure that your **code is executable**. Otherwise, points for the respective question will be deducted.

2.2.1 Data Pre-Processing (1pt * 7)

i) Load data into python and print information about the dataframe.

ii) Assign exact column names to dataframe (Column names can be found in .names file provided in Blackboard.)

iii) Check missing values.

iv) Perform one-hot encoding to 'sex'.

v) Perform min-max normalisation on numerical variables(exclude target variable 'Rings').

vi) Transform target variable: ('Rings'+ 1.5 = Age (yrs)) and remove 'Rings' in dataframe.

vii) Plot bar charts for numerical variables.

2.2.2 K-Means Clustering(using sklearn) (5pts)

i) Construct model with k = 5 (3pts)

ii) Compute and print silhouette coefficients. (2pts)

2.2.3 GMM and EM Algorithms (13pts)

i) Fit Gaussian Mixture Model with 2 Gaussians on ['Whole weights'] using EM Algorithm (choose number_of_iteration = 20).
Third party packages are not allowed for this task, you can refer to reference link above. (5pts)

ii) Plot fitted GMM. (1pt)

iii) Now with the help of sklearn, fit GMM with 2 Gaussians on ['Whole weights'] again and Plot fitted GMM. (3pts)

iv) Fit GMM with all predictors (sklearn is allowed to use). (2pts)

v) Compute and print silhouette coefficients for GMM modelled in iii) and compare it with your results 2.2.2. Briefly explain your preference. (2pts)

Q 1.1.1

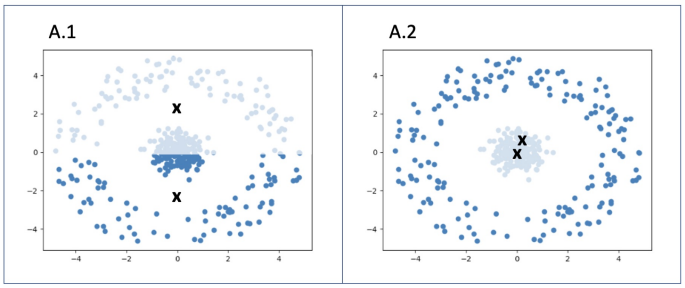


Figure 1: **Select one: A.1 or A.2**

Image ∈ images generated by k-means:

⇒ A.2

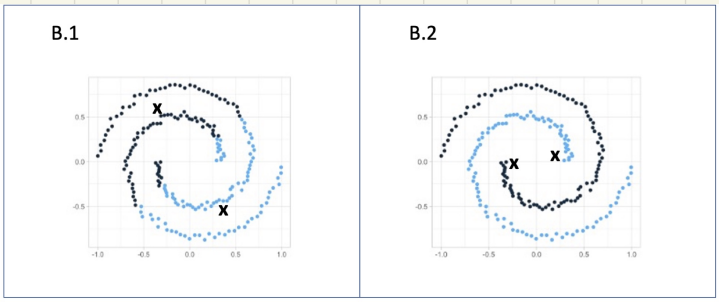


Figure 2: **Select one: B.1 or B.2**

⇒ B.2

Q 1.1.2

1.1.2.(10 points) Consider a dataset $\mathcal{D}$ with 5 points as shown below. Perform a k-means clustering on this dataset with $k = 2$ using the Euclidean distance as the distance function. Remember that in the k-means algorithm, one iteration consists of following two steps: first, we assign each data point to its nearest cluster center; second, we recompute each center as the average of the data points assigned to it. Initially, the 2 cluster centers are chosen randomly as $\mu_0 = (5.3, 3.5)$, $\mu_1 = (5.1, 4.2)$. Parts (a) through (d) refer only to the first iteration of k-means clustering performed on

$$\mathcal{D} = \begin{bmatrix} 5.5 & 3.1 \\ 5.1 & 4.8 \\ 6.6 & 3.0 \\ 5.5 & 4.6 \\ 6.8 & 3.8 \end{bmatrix}$$

$(5.5, 3.1)$: Dist to $\mu_0 = \sqrt{(5.5-5.3)^2+(3.1-3.5)^2} = \sqrt{0.2} \approx 0.44721$
Dist to $\mu_1 = \sqrt{(5.5-5.1)^2+(3.1-4.2)^2} = \sqrt{0.97} \approx 0.98489$ } to μ_0

$(5.1, 4.8)$: Dist to $\mu_0 = \sqrt{(5.1-5.3)^2+(4.8-3.5)^2} = \sqrt{1.73} \approx 1.3151$
Dist to $\mu_1 = \sqrt{(5.1-5.1)^2+(4.8-4.2)^2} = \sqrt{0.36} = 0.6$ } to μ_1

$(6.6, 3.0)$: Dist to $\mu_0 = \sqrt{(6.6-5.3)^2+(3.0-3.5)^2} = \sqrt{1.94} \approx 1.8928$
Dist to $\mu_1 = \sqrt{(6.6-5.1)^2+(3.0-4.2)^2} = \sqrt{3.69} \approx 1.9209$ } to μ_0

$(5.5, 4.6)$: Dist to $\mu_0 = \sqrt{(5.5-5.3)^2+(4.6-3.5)^2} = \sqrt{1.25} \approx 1.11803$
Dist to $\mu_1 = \sqrt{(5.5-5.1)^2+(4.6-4.2)^2} = \sqrt{0.32} \approx 0.56569$ } to μ_1

$(6.8, 3.8)$: Dist to $\mu_0 = \sqrt{(6.8-5.3)^2+(3.8-3.5)^2} = \sqrt{2.34} \approx 1.5297$
Dist to $\mu_1 = \sqrt{(6.8-5.1)^2+(3.8-4.2)^2} = \sqrt{3.05} \approx 1.7464$ } to μ_0

new center of $\mu_0 = \left[\frac{5.5+6.8+6.6}{3}, \frac{3.1+3.8+3.0}{3}\right] = (6.3, 3.3)$

new center of $\mu_1 = \left[\frac{5.1+5.5}{2}, \frac{4.8+4.6}{2}\right] = (5.3, 4.7)$

(a)Which of the following points will be the new center for cluster 0?

- (5.7 , 4.1)
- (5.6 , 4.8)
- (6.3 , 3.3)
- (6.7 , 3.4)

Ans: (6.3, 3.3)

Points in μ_0:
1. (5.5, 3.1)
2. (6.6, 3.0)
3. (6.8, 3.8)

(b)Which of the following points will be the new center for cluster 1?

- (6.1 , 3.8)
- (5.5 , 4.6)
- (5.4 , 4.7)
- (5.3 , 4.7)

Ans: (5.3, 4.7)

Points in μ_1:
1. (5.1, 4.8)
2. (5.5, 4.6)

(c)How many points will belong to cluster 0, using the new centers? = 3

(d)How many points will belong to cluster 1, using the new centers? = 2

Q 1.1.3

1.1.3.(5 points) Recall that in k-means clustering we attempt to find k cluster centers $\mu_1, \ldots, \mu_k$ such that the total distance between each point and the nearest cluster center is minimized. We thus solve

$$\underset{\mu_1,\ldots,\mu_k}{\operatorname{argmin}} \sum_{i=1}^{N} \min_{j\in\{1,\ldots,k\}} \|\mathbf{x}^{(i)} - \mu_j\|_2^2$$

where n is the number of data points. Instead of holding the number of clusters k fixed, your friend John tries to also minimize the objective over k, solving

$$\underset{k}{\operatorname{argmin}} \underset{\mu_1,\ldots,\mu_k}{\operatorname{argmin}} \sum_{i=1}^{N} \min_{j\in\{1,\ldots,k\}} \|\mathbf{x}^{(i)} - \mu_j\|_2^2$$

You found this idea to be a bad one.

(a)What is the minimum possible value of the objective function when minimizing over k?

When minimizing over k, min possible objective function value is zero

i.e. $\underset{\mu_1 \ldots \mu_k}{\operatorname{argmin}} \sum_{i=1}^{N} \min_{j\in\{1,\ldots,k\}} \|x^{(i)} - \mu_j\|_2^2 = 0$ (every point becomes its own cluster)

(b)What is a value of k for which we achieve the minimum possible value of the objective function when $N = 100$?

When N=100, for objective function value = 0 implies that k = 100

Q1-2.

(a). There are two derivations for the optimization problem of PCA in our lecture. The one (1) is to maximize the variance of the reconstructions and the other (2) is to minimize the the reconstruction error.

$$\max_{\mathbf{U},\mathbf{U}^T\mathbf{U}=\mathbf{I}} \frac{1}{N}\sum_{n=1}^{N}\|\tilde{\mathbf{x}}^{(n)}-\tilde{\boldsymbol{\mu}}\|^2, \tag{1}$$

$$\min_{\mathbf{U},\mathbf{U}^T\mathbf{U}=\mathbf{I}} \frac{1}{N}\sum_{n=1}^{N}\|\mathbf{x}^{(n)}-\tilde{\mathbf{x}}^{(n)}\|^2, \tag{2}$$

where $\mathbf{x}\in\mathbb{R}^d$ denotes original data sample, $\tilde{\mathbf{x}}=\boldsymbol{\mu}+\mathbf{U}\mathbf{z}$ denotes the reconstruction of $\mathbf{x}$ and $\mathbf{U}\in\mathbb{R}^{d\times k}$, in which $\mathbf{z}=\mathbf{U}^T(\mathbf{x}-\boldsymbol{\mu})$ and $\boldsymbol{\mu}=\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}^{(n)}$, $\tilde{\boldsymbol{\mu}}=\frac{1}{N}\sum_{n=1}^{N}\tilde{\mathbf{x}}^{(n)}$ denote the mean of the data and reconstructions, respectively.

$N\tilde{\mu} = \sum_{n=1}^{N}\tilde{x}^{(n)} = N\mu$

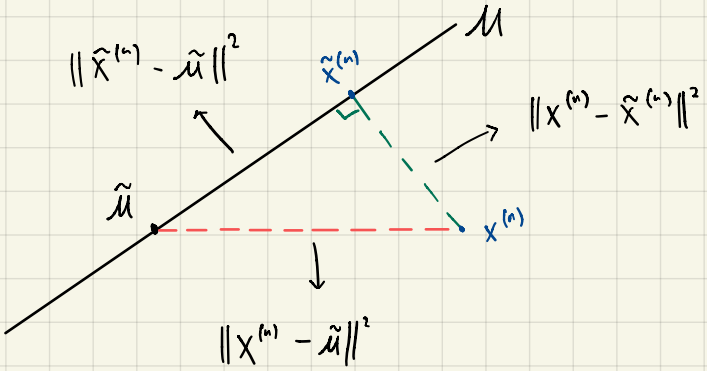
Prove the problem 1 and problem 2 are equivalent. In other words, you need to prove the following result:

$$\frac{1}{N}\sum_{n=1}^{N}\|\tilde{\mathbf{x}}^{(n)}-\tilde{\boldsymbol{\mu}}\|^2+\frac{1}{N}\sum_{n=1}^{N}\|\mathbf{x}^{(n)}-\tilde{\mathbf{x}}^{(n)}\|^2=\underbrace{\frac{1}{N}\sum_{n=1}^{N}\|\mathbf{x}^{(n)}-\boldsymbol{\mu}\|^2}_{\text{constant}}$$

Proof by Pythagoras' Theorem,

$x^{(n)}$: original data point
$\tilde{x}^{(n)}$: reconstructed data point

μ : mean of original data
$\tilde{\mu}$: mean of reconstructed data



We get the Eqn : $\|x^{(n)}-\tilde{\mu}\|^2 = \|\tilde{x}^{(n)}-\tilde{\mu}\|^2 + \|x^{(n)}-\tilde{x}^{(n)}\|^2$

→ take all data points (Sum) and divide by N

$$\Rightarrow \frac{1}{N}\sum_{n=1}^{N}\|\tilde{x}^{(n)}-\tilde{\mu}\|^2+\frac{1}{N}\sum_{n=1}^{N}\|x^{(n)}-\tilde{x}^{(n)}\|^2=\frac{1}{N}\sum_{n=1}^{N}\|x^{(n)}-\tilde{\mu}\|^2$$

Thus, Pythagoras' Theorem shows that total variance can be decomposed into reconstruction error and variance of the reconstructed data

(b). Let dataset $\mathbf{X} = \{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(n)}\}, \mathbf{x} \in \mathbb{R}^d$. Now, I want to project $\mathbf{X}$ into one-dimensional space using PCA that means I need to find a $\mathbf{w} \in \mathbf{R}^d$, s.t. $\mathbf{w}^T\mathbf{w} = 1$ to satisfy $z^{(i)} = \mathbf{w}^T(\mathbf{x}^{(i)} - \boldsymbol{\mu}) \in \mathbb{R}$, where $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}^{(i)}$ denotes the data mean. Prove the $\mathbf{w}$ is the eigenvector associated with the largest eigenvalue of covariance matrix $\boldsymbol{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}^{(i)} - \boldsymbol{\mu})(\mathbf{x}^{(i)} - \boldsymbol{\mu})^T$.

Using the PCA optimization problem: $\max_{U} \text{Trace}(U^T \Sigma U)$ st $U^T U = I$

$$\Rightarrow \max \sum_{k=1}^{K} u_k^T \Sigma u_k \quad \text{st} \quad u_k^T u_k = 1$$

We consider the single case of w_1:

$$\max \; w_1^T \Sigma w_1 \quad \text{st} \quad w_1^T w_1 = 1$$

Take the Lagrangian: $\mathcal{L}(w_1, \lambda_1) = w_1^T \Sigma w_1 - \lambda_1 (w_1^T w_1 - 1)$

1st order derivative: $\frac{\partial \mathcal{L}}{\partial w_1} = \Sigma w_1 - \lambda_1 w_1 = 0$

$$\Rightarrow \Sigma w_1 = \lambda_1 w_1$$

$\Rightarrow w_1$ is an eigenvector of Σ, with λ_1 as the corresponding eigenvalue

Since we are maximizing variance i.e. $\max w_1^T \Sigma w_1$, w_1 corresponds to the eigenvector of Σ with the largest eigenvalue λ_1 (Proven)