

HOMWORK 5

Rishideep Reddy Rallabandi
9084949099

Instructions: Use this latex file as a template to develop your homework. Submit your homework on time as a single pdf file. Please wrap your code and upload to a public GitHub repo, then attach the link below the instructions so that we can access it. Answers to the questions that are not within the pdf are not accepted. This includes external links or answers attached to the code implementation. Late submissions may not be accepted. You can choose any programming language (i.e. python, R, or MATLAB). Please check Piazza for updates about the homework. It is ok to share the experiments results and compare them with each other.

Github link: <https://github.com/Rishideep08/CS760-HW5>

1 Clustering

1.1 K-means Clustering (14 points)

1. **(6 Points)** Given n observations $X_1^n = \{X_1, \dots, X_n\}$, $X_i \in \mathcal{X}$, the K-means objective is to find $k (< n)$ centres $\mu_1^k = \{\mu_1, \dots, \mu_k\}$, and a rule $f: \mathcal{X} \rightarrow \{1, \dots, K\}$ so as to minimize the objective

$$J(\mu_1^K, f; X_1^n) = \sum_{i=1}^n \sum_{k=1}^K \mathbb{1}(f(X_i) = k) \|X_i - \mu_k\|^2 \quad (1)$$

Let $\mathcal{J}_K(X_1^n) = \min_{\mu_1^K, f} J(\mu_1^K, f; X_1^n)$. Prove that $\mathcal{J}_K(X_1^n)$ is a non-increasing function of K .

Let's consider a scenario where we have k centers that converge to μ_k with a loss function $J(\mu_K, f; X_n)$. Now, if we add one more center, two situations may arise:

1. If there's no data in the new cluster, the loss remains the same, i.e., $J(\mu_K, f; X_n)$.
2. If there are data points from the previous clusters in the new cluster, the overall loss function for $J(\mu_{K+1}^{\text{new}}, f_{\text{new}}; X_n)$ with the new center μ_{K+1}^{new} and new assignments f_{new} can be expressed as:

$$J(\mu_{K+1}^{\text{new}}, f_{\text{new}}; X_n) = J(\mu_K, f; X_n) + \sum_{i=1}^n \sum_{k=1}^{K+1} \mathbb{1}(f_{\text{new}}(X_i) = k) \|X_i - \mu_{K+1}^{\text{new}}\|^2 - \sum_{i=1}^n \sum_{k=1}^K \mathbb{1}(f(X_i) = k) \|X_i - \mu_k\|^2$$

In this case, the term $\sum_{i=1}^n \sum_{k=1}^{K+1} \mathbb{1}(f_{\text{new}}(X_i) = k) \|X_i - \mu_{K+1}^{\text{new}}\|^2 - \sum_{i=1}^n \sum_{k=1}^K \mathbb{1}(f(X_i) = k) \|X_i - \mu_k\|^2$ is always non-positive. This is because, as the points move to their nearest center, this term decreases or remains zero. i.e few points which moved to the new cluster will help in decreasing the loss of the previous clusters and the new cluster loss is less than that's why they move to that point.

Combining both scenarios, we can conclude that the loss is non-increasing.

2. **(8 Points)** Consider the K-means (Lloyd's) clustering algorithm we studied in class. We terminate the algorithm when there are no changes to the objective. Show that the algorithm terminates in a finite number of steps.

Assume we have k centers and n data points, resulting in k^n possible combinations. Initially, we establish a cluster setup, and after each iteration, a new cluster arrangement emerges. Two scenarios arise:

1. If the new clustering matches the old one, we can terminate and consider it final.
2. If the new clustering differs, the updated configuration incurs lower costs, motivating the shift.

The reason for the decreasing loss value in each iteration is that we move to the mean of assigned points, reducing the Euclidean distance.

To explore the possibility of cycles greater than size 1, it is improbable. At each step, we transition to the next set with lower loss, preventing the algorithm from entering a loop.

1.2 Experiment (20 Points)

In this question, we will evaluate K-means clustering and GMM on a simple 2 dimensional problem. First, create a two-dimensional synthetic dataset of 300 points by sampling 100 points each from the three Gaussian distributions shown below:

$$P_a = \mathcal{N}\left(\begin{bmatrix} -1 \\ -1 \end{bmatrix}, \sigma \begin{bmatrix} 2 & 0.5 \\ 0.5 & 1 \end{bmatrix}\right), \quad P_b = \mathcal{N}\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix}, \sigma \begin{bmatrix} 1 & -0.5 \\ -0.5 & 2 \end{bmatrix}\right), \quad P_c = \mathcal{N}\left(\begin{bmatrix} 0 \\ 1 \end{bmatrix}, \sigma \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}\right)$$

Here, σ is a parameter we will change to produce different datasets.

First implement K-means clustering and the expectation maximization algorithm for GMMs. Execute both methods on five synthetic datasets, generated as shown above with $\sigma \in \{0.5, 1, 2, 4, 8\}$. Finally, evaluate both methods on (i) the clustering objective (1) and (ii) the clustering accuracy. For each of the two criteria, plot the value achieved by each method against σ .

Guidelines:

- Both algorithms are only guaranteed to find only a local optimum so we recommend trying multiple restarts and picking the one with the lowest objective value (This is (1) for K-means and the negative log likelihood for GMMs). You may also experiment with a smart initialization strategy (such as kmeans++).
- To plot the clustering accuracy, you may treat the ‘label’ of points generated from distribution P_u as u , where $u \in \{a, b, c\}$. Assume that the cluster id i returned by a method is $i \in \{1, 2, 3\}$. Since clustering is an unsupervised learning problem, you should obtain the best possible mapping from $\{1, 2, 3\}$ to $\{a, b, c\}$ to compute the clustering objective. One way to do this is to compare the clustering centers returned by the method (centroids for K-means, means for GMMs) and map them to the distribution with the closest mean.

Points break down: 7 points each for implementation of each method, 6 points for reporting of evaluation metrics.

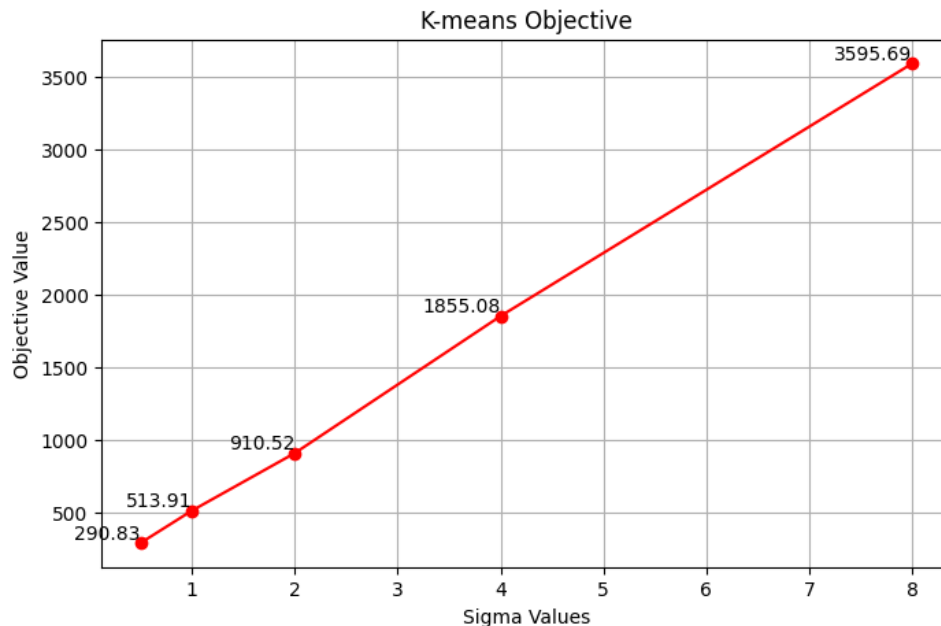


Figure 1: kmeans obj

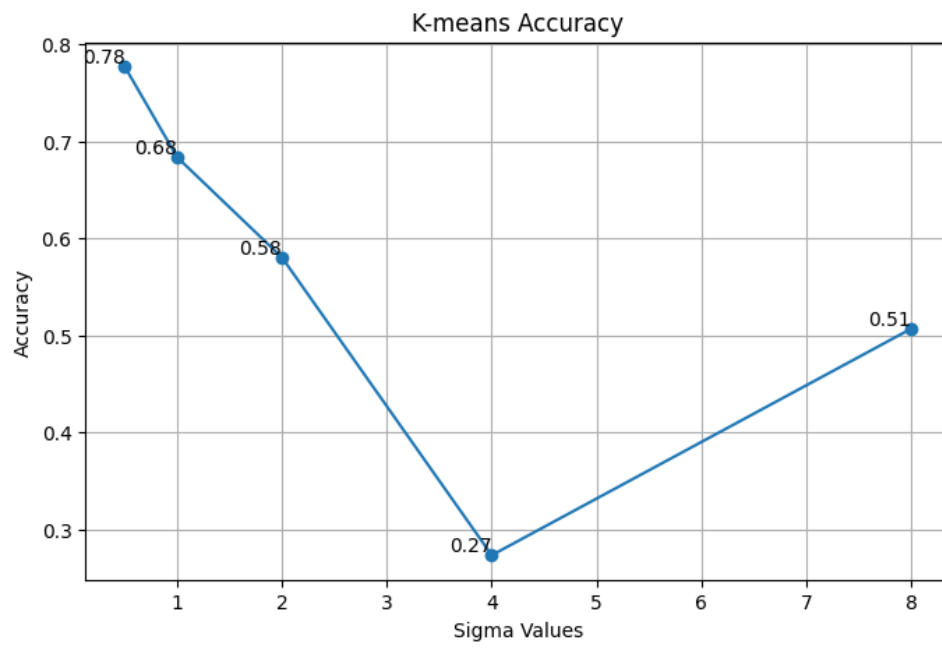


Figure 2: kmeans acc

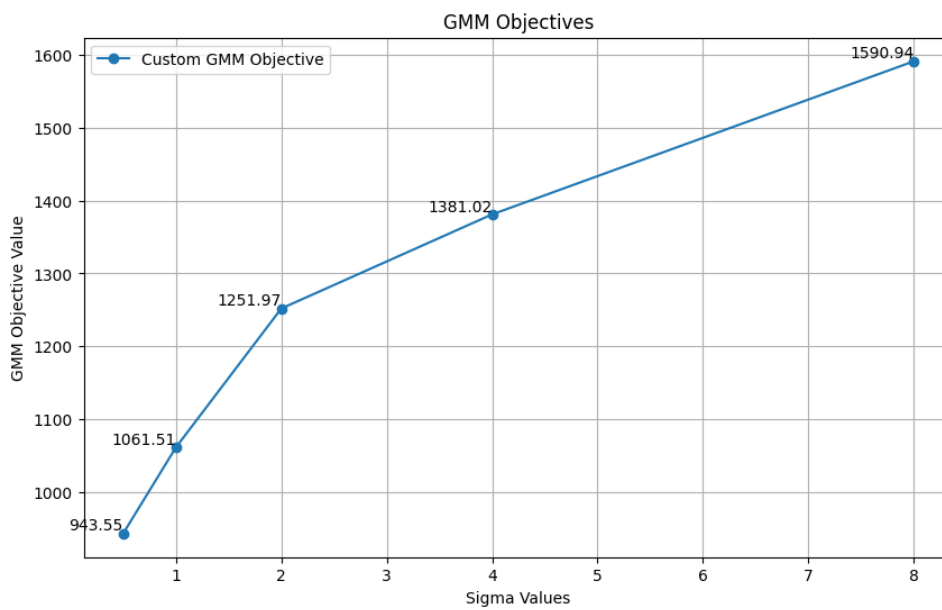


Figure 3: gmm obj

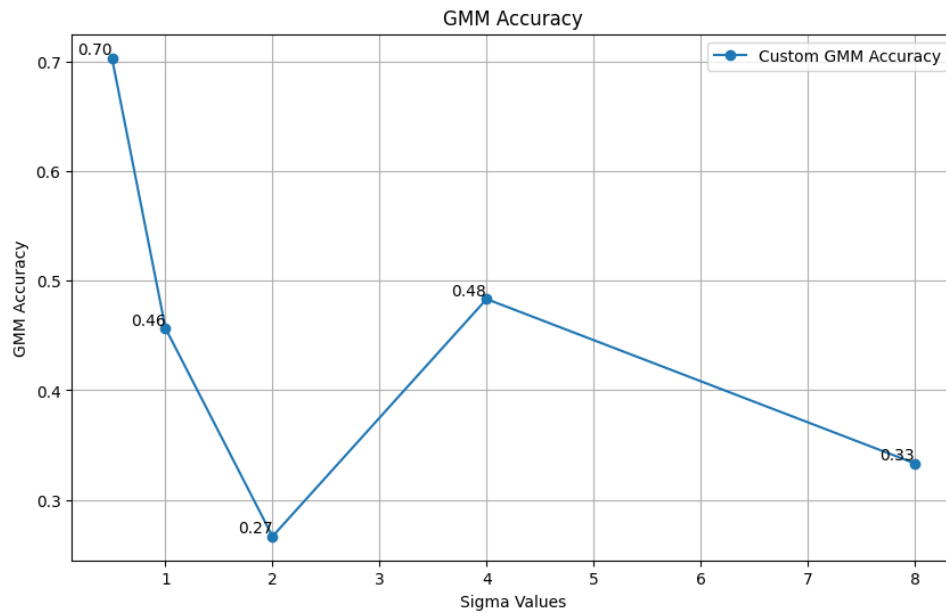


Figure 4: gmm acc

2 Linear Dimensionality Reduction

2.1 Principal Components Analysis (10 points)

Principal Components Analysis (PCA) is a popular method for linear dimensionality reduction. PCA attempts to find a lower dimensional subspace such that when you project the data onto the subspace as much of the information is preserved. Say we have data $X = [x_1^\top; \dots; x_n^\top] \in \mathbb{R}^{n \times D}$ where $x_i \in \mathbb{R}^D$. We wish to find a d ($< D$) dimensional subspace $A = [a_1, \dots, a_d] \in \mathbb{R}^{D \times d}$, such that $a_i \in \mathbb{R}^D$ and $A^\top A = I_d$, so as to maximize $\frac{1}{n} \sum_{i=1}^n \|A^\top x_i\|^2$.

1. **(4 Points)** Suppose we wish to find the first direction a_1 (such that $a_1^\top a_1 = 1$) to maximize $\frac{1}{n} \sum_i (a_1^\top x_i)^2$. Show that a_1 is the first right singular vector of X .

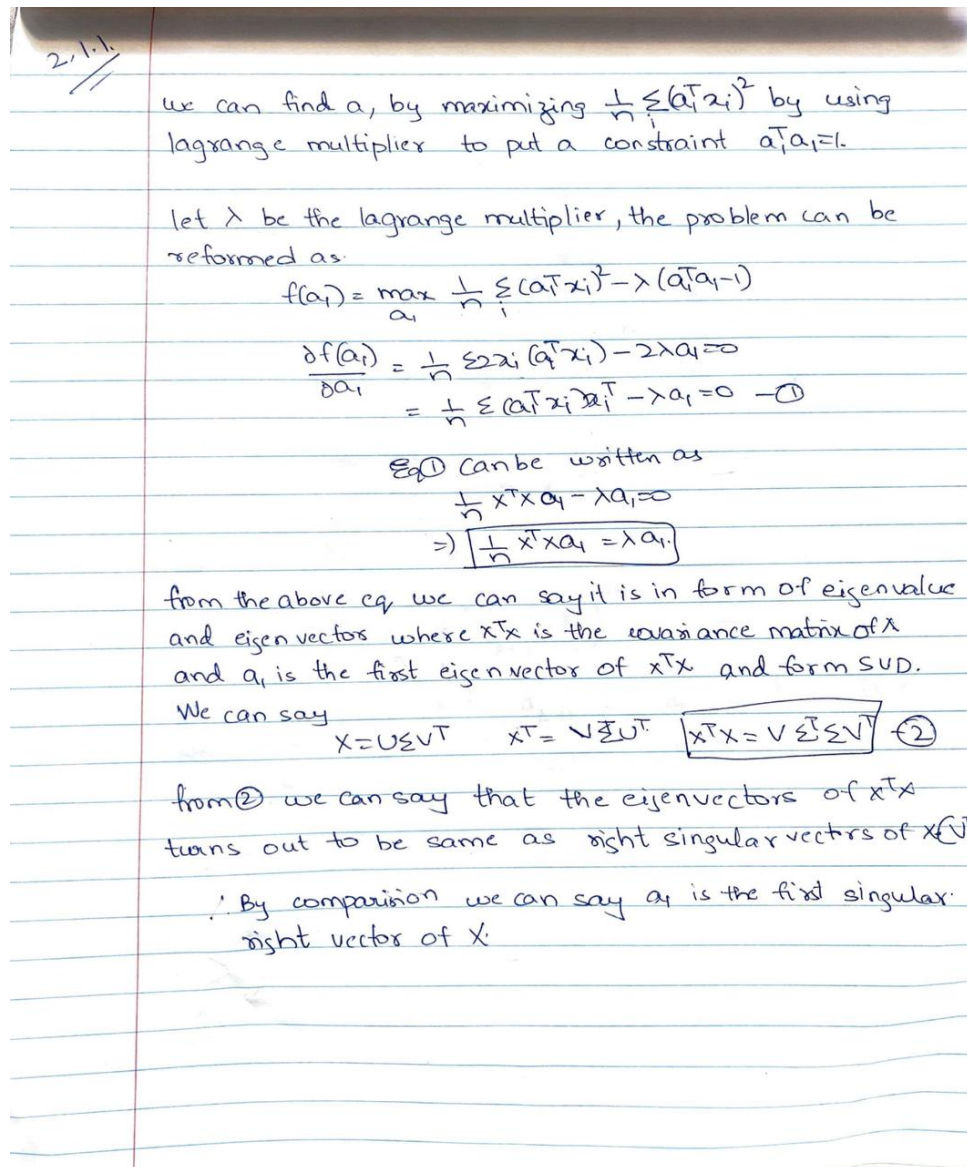


Figure 5: 2.1.1

2. (6 Points) Given a_1, \dots, a_k , let $A_k = [a_1, \dots, a_k]$ and $\tilde{x}_i = x_i - A_k A_k^T x_i$. We wish to find a_{k+1} , to maximize $\frac{1}{n} \sum_i (a_{k+1}^T \tilde{x}_i)^2$. Show that a_{k+1} is the $(k+1)^{th}$ right singular vector of X .

2.1.2

we are given that $\tilde{x}_i = x_i - A_k A_k^T x_i$

$$\tilde{x}_i = x_i - [a_1 a_2 \dots a_k] \begin{bmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_k^T \end{bmatrix} x_i$$

$$= x_i - \sum_{j=1}^k a_j a_j^T x_i$$

from maximization of $J = \frac{1}{n} \sum_{i=1}^n (a_{k+1}^T \tilde{x}_i)^2$

i.e. $J = \frac{1}{n} \sum_{i=1}^n [a_{k+1}^T (x_i - \sum_{j=1}^k a_j a_j^T x_i)]^2$

$$= \frac{1}{n} \sum_{i=1}^n (a_{k+1}^T x_i - \sum_{j=1}^k a_{k+1}^T a_j a_j^T x_i)^2$$

as we know $A^T A = I_d$ $\therefore a_i^T a_j = 1$ only if $i=j$

$$\Rightarrow J = \frac{1}{n} \sum_{i=1}^n (a_{k+1}^T x_i)^2$$

from the proof from above problem we can say a_{k+1} is the right singular vector of X and orthogonal to (a_i) $i \neq k+1$.

$\therefore a_{k+1}$ is the $(k+1)^{th}$ right singular vector of X .

Figure 6: 2.1.2

2.2 Dimensionality reduction via optimization (22 points)

We will now motivate the dimensionality reduction problem from a slightly different perspective. The resulting algorithm has many similarities to PCA. We will refer to method as DRO.

As before, you are given data $\{x_i\}_{i=1}^n$, where $x_i \in \mathbb{R}^D$. Let $X = [x_1^T; \dots x_n^T] \in \mathbb{R}^{n \times D}$. We suspect that the data actually lies approximately in a d dimensional affine subspace. Here $d < D$ and $d < n$. Our goal, as in PCA, is to use this dataset to find a d dimensional representation z for each $x \in \mathbb{R}^D$. (We will assume that the span of the

data has dimension larger than d , but our method should work whether $n > D$ or $n < D$.)

Let $z_i \in \mathbb{R}^d$ be the lower dimensional representation for x_i and let $Z = [z_1^\top; \dots; z_n^\top] \in \mathbb{R}^{n \times d}$. We wish to find parameters $A \in \mathbb{R}^{D \times d}$, $b \in \mathbb{R}^D$ and the lower dimensional representation $Z \in \mathbb{R}^{n \times d}$ so as to minimize

$$J(A, b, Z) = \frac{1}{n} \sum_{i=1}^n \|x_i - Az_i - b\|^2 = \|X - ZA^\top - \mathbf{1}b^\top\|_F^2. \quad (2)$$

Here, $\|A\|_F^2 = \sum_{i,j} A_{ij}^2$ is the Frobenius norm of a matrix.

1. **(3 Points)** Let $M \in \mathbb{R}^{d \times d}$ be an arbitrary invertible matrix and $p \in \mathbb{R}^d$ be an arbitrary vector. Denote, $A_2 = A_1 M^{-1}$, $b_2 = b_1 - A_1 M^{-1} p$ and $Z_2 = Z_1 M^\top + \mathbf{1} p^\top$. Show that both (A_1, b_1, Z_1) and (A_2, b_2, Z_2) achieve the same objective value J (2).

Therefore, in order to make the problem determined, we need to impose some constraint on Z . We will assume that the z_i 's have zero mean and identity covariance. That is,

$$\bar{Z} = \frac{1}{n} \sum_{i=1}^n z_i = \frac{1}{n} Z^\top \mathbf{1}_n = 0, \quad S = \frac{1}{n} \sum_{i=1}^n z_i z_i^\top = \frac{1}{n} Z^\top Z = I_d$$

Here, $\mathbf{1}_d = [1, 1, \dots, 1]^\top \in \mathbb{R}^d$ and I_d is the $d \times d$ identity matrix.

2.2.1.

we are given $M \in \mathbb{R}^{d \times d}$ $P \in \mathbb{R}^{d \times d}$

and $A_2 = A_1 M^T$, $b_2 = b_1 - A_1 M^T P$, $z_2 = z_1 M^T + P^T$

$$J(A_1, b_1, z_1) = \frac{1}{n} \sum_{i=1}^n \|x_i - A_1 z_i - b_1\|^2$$

$$= \|X - Z A_1^T - b_1^T\|_F^2 \quad \text{--- (1)}$$

$$J(A_2, b_2, z_2) = \|X - Z_2 A_2^T - b_2^T\|_F^2$$

$$= \|X - (z_1 M^T + P^T) A_1^T M - (b_1 - A_1 M^T P)^T\|_F^2$$

$$\Rightarrow \|X - z_1 M^T (M^T A_1^T M) - P^T (M^T A_1^T M) - b_1^T + A_1 M^T P^T\|_F^2$$

$$\Rightarrow \|X - z_1 (M^T M)^T A_1^T - b_1^T\|_F^2$$

$$\Rightarrow \|X - z_1 A_1^T - b_1^T\|_F^2 \quad \text{--- (2)}$$

from (1) and (2) we can say $J(A_1, b_1, z_1) = J(A_2, b_2, z_2)$

Figure 7: 2.2.1

2. (16 Points) Outline a procedure to solve the above problem. Specify how you would obtain A, Z, b which minimize the objective and satisfy the constraints.

Hint: The rank k approximation of a matrix in Frobenius norm is obtained by taking its SVD and then zeroing out all but the first k singular values.

2.2.2

$$J = \frac{1}{n} \sum_{i=1}^n (\|x_i - Az_i - b\|^2)$$

$$\frac{\partial J}{\partial b} = \frac{1}{n} \sum_{i=1}^n 2(x_i - Az_i - b) = 0$$

$$\Rightarrow b = \frac{1}{n} \sum_{i=1}^n (x_i - Az_i)$$

→ let's consider b to be the mean of x_i
 i.e. $b = \frac{1}{n} \sum_{i=1}^n x_i$

It puts a constraint on $\frac{1}{n} \sum_{i=1}^n z_i = 0$

→ Consider a matrix $R = X - b$ and its singular value decomposition $R = U \Sigma V^T$

→ $R = Z A^T = X - b$. Rankd approximation gives $U_d \Sigma_d V_d^T$. So we can approximate

$$Z = \sqrt{n} U_d \quad A^T = \frac{1}{\sqrt{n}} \Sigma_d V_d^T$$

As we know $V \Sigma U^T \mathbf{1} = 0 \Rightarrow U_d^T \mathbf{1} = 0 \Rightarrow Z^T \mathbf{1} = \frac{1}{n} \sum_{i=1}^n x_i = 0$

So our assumption to consider b has set the z_i to zero

Figure 8: 2.2.2

3. (3 Points) You are given a point x_* in the original D dimensional space. State the rule to obtain the d dimensional representation z_* for this new point. (If x_* is some original point x_i from the D -dimensional space, it should be the d -dimensional representation z_i .)

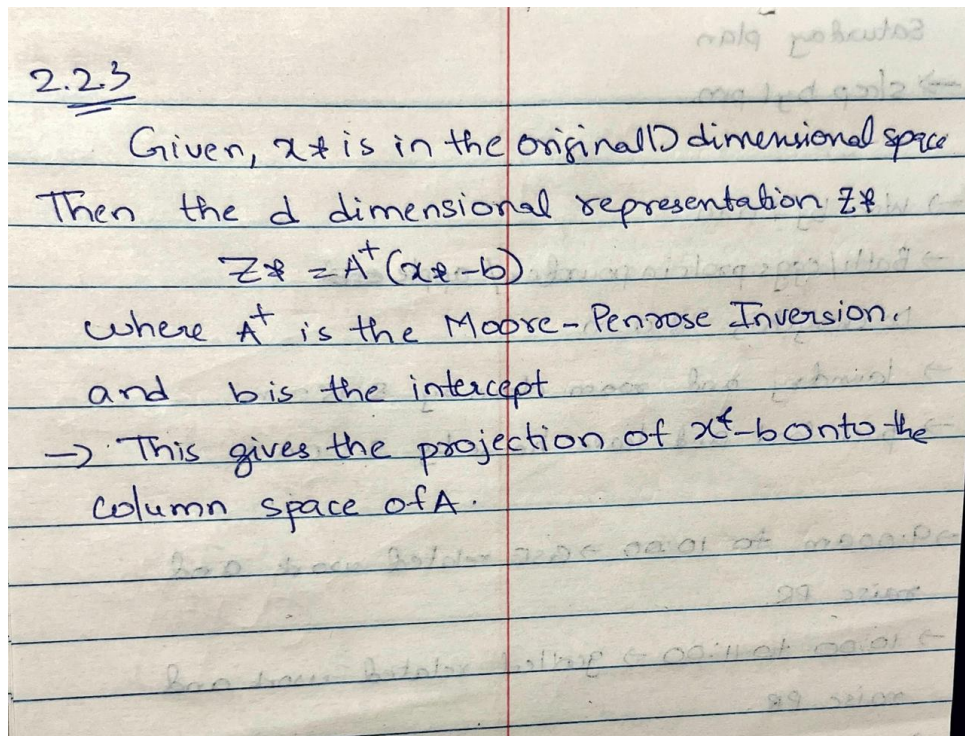


Figure 9: 2.2.3

2.3 Experiment (34 points)

Here we will compare the above three methods on two data sets.

- We will implement three variants of PCA:
 1. "buggy PCA": PCA applied directly on the matrix X .
 2. "demeaned PCA": We subtract the mean along each dimension before applying PCA.
 3. "normalized PCA": Before applying PCA, we subtract the mean and scale each dimension so that the sample mean and standard deviation along each dimension is 0 and 1 respectively.
- One way to study how well the low dimensional representation Z captures the linear structure in our data is to project Z back to D dimensions and look at the reconstruction error. For PCA, if we mapped it to d dimensions via $z = Vx$ then the reconstruction is $V^T z$. For the preprocessed versions, we first do this and then reverse the preprocessing steps as well. For DRO we just compute $Az + b$. We will compare all methods by the reconstruction error on the datasets.
- Please implement code for the methods: Buggy PCA (just take the SVD of X), Demeaned PCA, Normalized PCA, DRO. In all cases your function should take in an $n \times d$ data matrix and d as an argument. It should return the d dimensional representations, the estimated parameters, and the reconstructions of these representations in D dimensions.
- You are given two datasets: A two Dimensional dataset with 50 points `data2D.csv` and a thousand dimensional dataset with 500 points `data1000D.csv`.
- For the 2D dataset use $d = 1$. For the 1000D dataset, you need to choose d . For this, observe the singular values in DRO and see if there is a clear "knee point" in the spectrum. Attach any figures/ Statistics you computed to justify your choice.
- For the 2D dataset you need to attach the a plot comparing the original points with the reconstructed points for all 4 methods. For both datasets you should also report the reconstruction errors, that is the squared sum of differences $\sum_{i=1}^n \|x_i - r(z_i)\|^2$, where x_i 's are the original points and $r(z_i)$ are the D dimensional points reconstructed from the d dimensional representation z_i .

- **Questions:** After you have completed the experiments, please answer the following questions.
 1. Look at the results for Buggy PCA. The reconstruction error is bad and the reconstructed points don't seem to well represent the original points. Why is this?
Hint: Which subspace is Buggy PCA trying to project the points onto?
 When we perform Singular Value Decomposition (SVD) without centering the dataset, you're essentially identifying the optimal linear subspace rather than an affine one. In this scenario, the first principal component extends from the origin towards the data, and subsequent principal components are orthogonal to it.
 2. The error criterion we are using is the average squared error between the original points and the reconstructed points. In both examples DRO and demeaned PCA achieves the lowest error among all methods. Is this surprising? Why?
 No. It is not surprising because the reconstruction method is almost the same in both.
- Point allocation:
 - Implementation of the three PCA methods: **(6 Points)**
 - Implementation of DRO: **(6 points)**
 - Plots showing original points and reconstructed points for 2D dataset for each one of the 4 methods: **(10 points)**

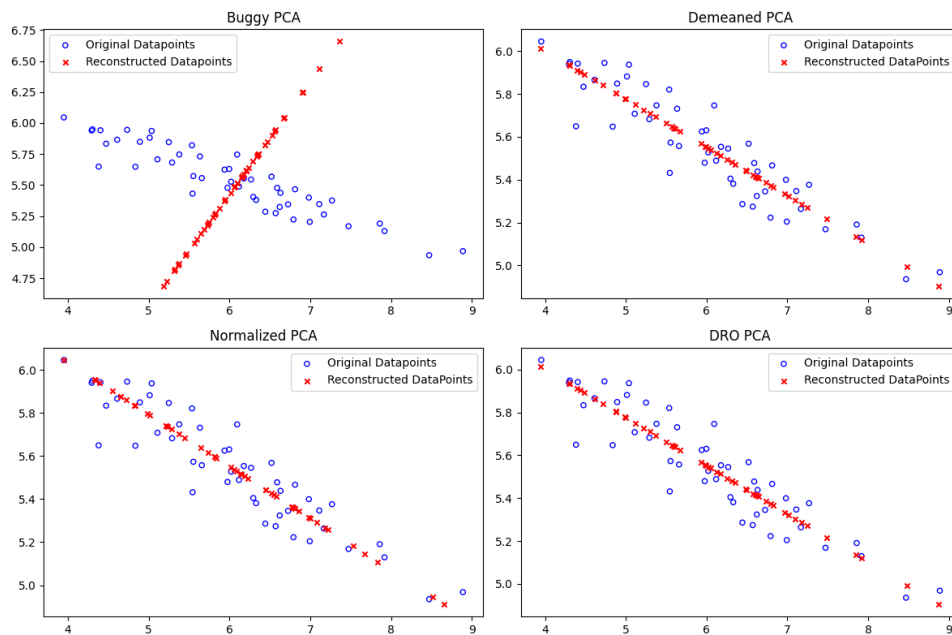


Figure 10: PCA

- Implementing reconstructions and reporting results for each one of the 4 methods for the 2 datasets: **(5 points)**

I have consider mean for error

Data Dimensionality	Buggy PCA	Demeaned PCA	Normalized PCA	DRO PCA
2D	0.4435	0.0050	0.0247	0.0050
1000D	0.8027	0.2730	0.2736	0.2730

Table 1: Reconstruction Errors for Different PCA Methods

- Choice of d for 1000D dataset and appropriate justification: **(3 Points)**

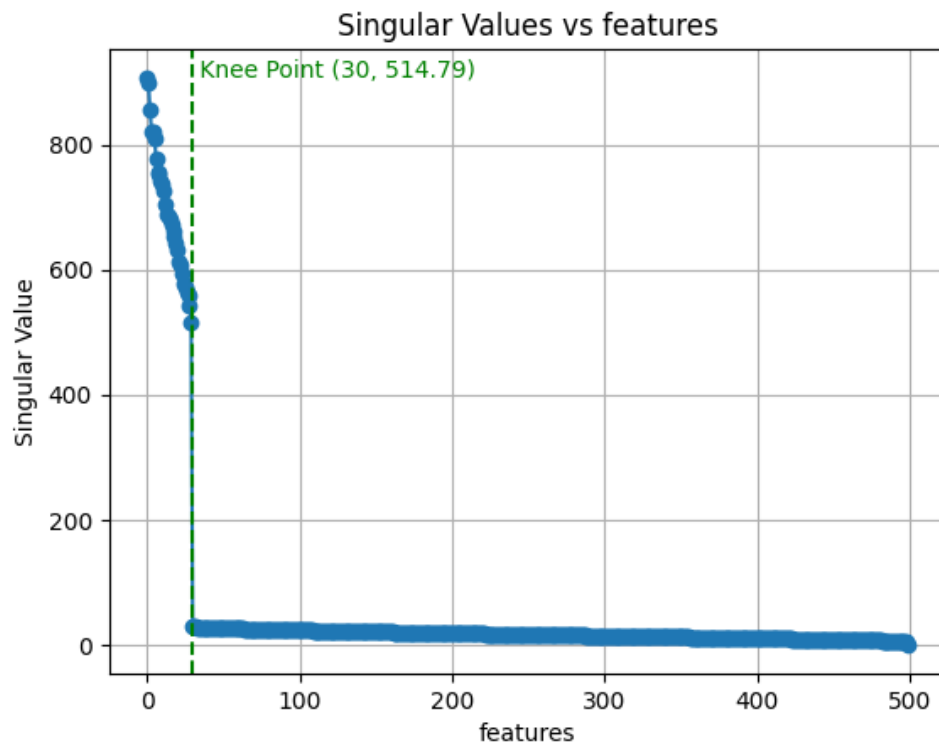


Figure 11: kneePoint

In the singular value plot 11, we observed a distinct drop at 30, indicating that the initial 30 values play a more significant role in contributing to the overall value, while the subsequent values have a lesser impact.

– Questions (4 Points)

Answer format:

The graph below is an example of how a plot of one of the algorithms for the 2D dataset may look like:

The blue circles are from the original dataset and the red crosses are the reconstructed points.

And this is how the reconstruction error may look like for Buggy PCA for the 2D dataset: 0.886903