HOMEWORK 5

Saikumar Yadugiri
9083079468, saikumar@cs.wisc.edu

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/saikumarysk/cs760_hw5`

1 Clustering

1.1 K-means Clustering (14 points)

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

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

   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.

   The main idea is that when $K = n$, we have that each $X_i \in X_1^n$ is a centroid and thus $J(\mu_1^K, f; X_1^n) = 0$. This means that $\mathcal{J}_K(X_1^n) = 0$. On the other hand, when $K = 1$, we have a single centroid and there is at least one point in X_1^n that is other than the centroid ($n > 1$). Hence, $J(\mu_1^K, f; X_1^n) > 0$. Thus $\mathcal{J}_K(X_1^n) > 0$. So, the function $\mathcal{J}_K(X_1^n)$ is not an increasing function. We need to show that it is non-increasing. The difference between $\mathcal{J}_K(X_1^n)$ and $\mathcal{J}_{K+1}(X_1^n)$ is the addition of another centroid and some change to some points' cluster label.

   We will randomly choose a point $\tilde{X} \in X_1^n$ to be the new centroid and this is the only point in that cluster. Hence, the new $J(\mu_1^{K+1}, f; X_1^n)$ will be at most as much as $J(\mu_1^K, f; X_1^n)$ as we are subtracting the value for $\tilde{x}$ from $J(\mu_1^K, f; X_1^n)$. Hence, the minimum value (by choosing another mapping f) can only decrease. Hence, $\mathcal{J}_K(X_1^n) \geq \mathcal{J}_{K+1}(X_1^n)$.

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.

   Note that f maps between n datapoints and k clusters. Hence, there are only finite(k^n) number of cluster assignments. As in each iteration of the k-means clustering algorithm, we decrease the clustering distance, we will eventually converge to the optimal one in finite number of steps when J doesn't change. Hence, we will converge to the optimal value in finite number of steps and hence the algorithm halts after finite number of steps.

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.

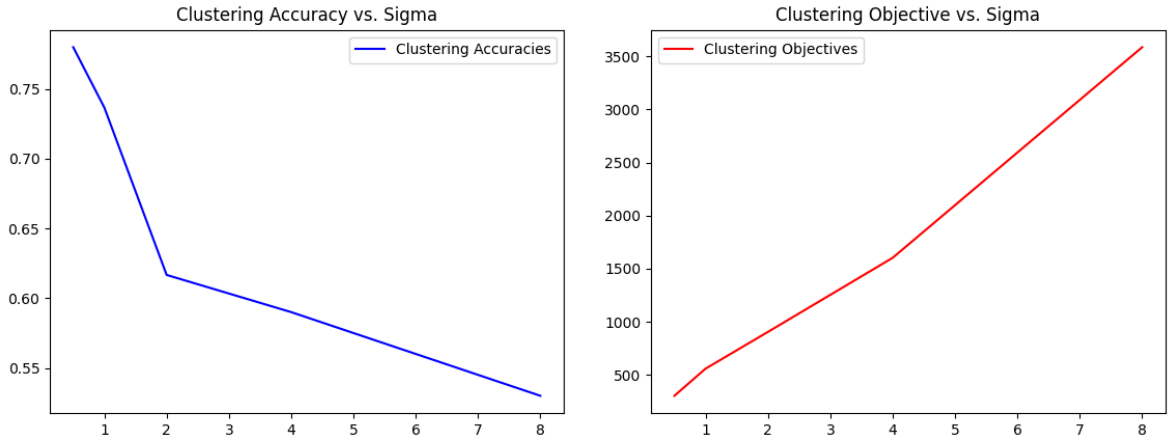
Guidelines:

- 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. These are the values I received for the evaluation of my K-Means algorithm. The results are tabulated in table 1 and figure 1. My Gaussian Mixture Model implementation was not working properly. But the implementation is present in GitHub,

σ	**Clustering Accuracy**	**Clustering Objective**
0.5	0.78	303.029643910935
1	0.7366666666666667	558.9918988336558
2	0.6166666666666667	904.9963963305394
4	0.59	1603.2153274028226
8	0.53	3587.3489027017526

Table 1: Clustering Metrics for K-Means Algorithm



(a) Clustering Accuracy

(b) Clustering Objective

Figure 1: Clustering Metrics for K-Means Algorithm

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 $X = [x_1^\top; \ldots; x_n^\top] \in \mathbb{R}^{n \times D}$ where $x_i \in \mathbb{R}^D$. We wish to find a d ($< D$) dimensional subspace $A = [a_1, \ldots, a_d] \in \mathbb{R}^{D \times d}$, 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.

   Note that as

   $$X = \begin{bmatrix} x_1^\top \\ \vdots \\ x_n^\top \end{bmatrix}, X^\top X = \begin{bmatrix} x_1 & \ldots & x_n \end{bmatrix} \begin{bmatrix} x_1^\top \\ \vdots \\ x_n^\top \end{bmatrix} = \sum_{i=1}^n x_i x_i^\top$$

   We can re-write

   $$\frac{1}{n}\sum_{i=1}^n (a_1^\top x_i)^2 = \frac{1}{n}\sum_{i=1}^n a_1^\top x_i x_i^\top a_1 = a_1^\top \frac{X^\top X}{n} a_1$$

   If $X = U\Sigma V^\top$ (standard SVD form), $X^\top X = V\Sigma^T U^\top U \Sigma V^\top = V(\Sigma^\top \Sigma)V^\top$. Here, $\Sigma^\top\Sigma$ is a diagonal *square* matrix. where each value is square of the singular values of X. Hence, $X^\top X = V(\Sigma^\top\Sigma)V^\top$ is the eigendecomposition of $X^\top X$ and each column of V is an eigenvector of $X^\top X$. Since, the singular values in Σ are in the non-increasing order, the first column of V, the first right singular vector of X is the eignvector for which $X^\top X$ has the highest eigenvalue.

   We can prove a stronger result for convenience where for any non-zero vector a_1, the value $\frac{a_1^\top X^\top X a_1}{a_1^\top a_1}$ is maximized when a_1 is an eigenvector of $X^\top X$ which maximizes the eigenvalue. This is done simply by taking the gradient for the quantity and setting it to zero.

   $$\frac{\nabla}{\nabla a_1}\left(\frac{a_1^\top X^\top X a_1}{a_1^\top a_1}\right) = 0 \iff a_1^\top a_1 (X^\top X) a_1 = (a_1^\top X^\top X a_1) a_1$$

   Hence, the first right singular vector for X maximizes the desired value.

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

   This question is a re-wording of the *Rayleigh Quotient Theorem*. My proof is going to be similar to a compact version of the proof of this theorem from Dan Spielman's book. Consider $M = X^\top X$. Then we can prove this result using induction. The base case when $k = 0$ is similar to the proof of the above result. Now, for $\tilde{X}$ defined as $\tilde{X} = [\tilde{x}_1^\top; \ldots; \tilde{x}_n^\top]$, we can see that $\frac{1}{n}\sum_{i=1}^n (a_{k+1}^\top \tilde{x}_i)^2 = a_{k+1}^\top \frac{\tilde{X}^\top \tilde{X}}{n} a_{k+1}$. Now, $\tilde{X} = X - X A_k A_k^\top$. Then,

$$
\begin{aligned}
\tilde{X}^\top \tilde{X} &= (I - A_k A_k^\top) X^\top X (I - A_k A_k^\top) \\
&= (I - A_k A_k^\top) V \Sigma^\top \Sigma V^\top (I - A_k A_k^\top) \\
&= \left(V - A_k \begin{bmatrix} a_1^\top \\ \vdots \\ a_k^\top \end{bmatrix} \begin{bmatrix} a_1^\top & \dots & a_D^\top \end{bmatrix} \right) \Sigma^\top \Sigma \left(V^\top - \begin{bmatrix} a_1^\top \\ \vdots \\ a_D^\top \end{bmatrix} \begin{bmatrix} a_1 & \dots & a_k \end{bmatrix} A_k^\top \right) \\
&= (V - A_k \begin{bmatrix} I_k & 0 \end{bmatrix}) \Sigma^\top \Sigma (V^\top - \begin{bmatrix} I_k \\ 0 \end{bmatrix} A_k^\top) \\
&= \begin{bmatrix} 0 & \dots & 0 & a_{k+1} & \dots & a_D \end{bmatrix} \Sigma^\top \Sigma \begin{bmatrix} 0 \\ \vdots \\ 0 \\ a_{k+1}^\top \\ \vdots \\ a_D^\top \end{bmatrix}
\end{aligned}
$$

Hence, the maximizing value is the $(k+1)$-th right singular vector of X.

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 $X = [x_1^\top; \dots x_n^\top] \in \mathbb{R}^{n \times D}$. 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

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

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).

$$
\begin{aligned}
X - Z_2 A_2^\top - \mathbf{1} b_2^\top &= X - (Z_1 M^\top + \mathbf{1} p^\top)(A_1 M^{-1})^\top - \mathbf{1}(b_1 - A_1 M^{-1} p)^\top \\
&= X - Z_1 M^\top (M^{-1})^\top A_1^\top - \mathbf{1} p^\top (M^{-1})^\top A_1^\top - \mathbf{1} b_1^\top + \mathbf{1}(A_1 M^{-1} p)^\top \\
&= X - Z_1 (M^{-1} M)^\top A_1^\top - \mathbf{1} p^\top (M^{-1})^\top A_1^\top - \mathbf{1} b_1^\top + \mathbf{1} p^\top (M^{-1})^\top A_1^\top \\
&= X - Z_1 A_1^\top - \mathbf{1} b_1^\top \\
\text{Hence, } J(A_1, b_1, Z_1) &= J(A_2, b_2, Z_2).
\end{aligned}
$$

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,

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

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

2. **(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.

   Let's derive optimal b. We can re-write $J(A, b, Z)$ as

$$\begin{aligned} J(A,b,Z) &= \frac{1}{n}\sum_{i=1}^{n}(x_i - Az_i - b)^\top (x_i - Az_i - b) \\ &= \frac{1}{n}\sum_{i=1}^{n}(x_1^\top - z_i^\top A^\top - b^\top)(x_i - Az_i - b) \\ &= \frac{1}{n}\sum_{i=1}^{n}(-x_i^\top b + z_i^\top A^\top b - b^\top x_i + b^\top A z_i + b^\top b) + (\text{constant w.r.to b}) \\ \implies \frac{\nabla J}{\nabla b} &= \frac{1}{n}\sum_{i=1}^{n}(-x_i^\top + z_i^\top A^\top - x_i^\top + z_i^\top A_i^\top + 2b) \end{aligned}$$

   We obtain the minimizing value of b by setting the gradient to 0. This is because clearly the second gradient derivative is 2 which is greater than 0.

$$\begin{aligned} &\frac{\nabla J}{\nabla b} = 0 \\ \implies &\sum_{i=1}^{n}(-x_i^\top + z_i^\top A^\top + b) = 0 \\ \implies &b = \frac{\sum_{i=1}^{n}(x_i^\top - z_i^\top A^\top)}{n} \\ \implies &b = \frac{1}{n}\sum_{i=1}^{n} x_i^\top = \frac{1}{n} X^\top \mathbf{1}_n \end{aligned}$$

   The last step is because we have $Z^\top \mathbf{1}_n = 0$ (we will derive this here). Now, we obtain A and Z using b as derived above. As we need $X = ZA^\top + \mathbf{1}_n b^\top$, we need to optimize $ZA^\top = X - \mathbf{1}_n b^\top$. We need that $Z^\top Z = nI_d$. So, Z is nearly an orthogonal matrix. One good candidate for that would be the left singular matrix of the SVD of $X - \mathbf{1}_n b^\top$. That is, consider $X - \mathbf{1}_n b^\top = U\Sigma V^\top$ (SVD). Using the hint, we want to restrict this to d-approximation by considering the first d singular values and hence the $d \times d$ top-left sub-matrix of Σ. Let's denote this by $\Sigma^{(d)}$. Then choose the first d columns of U and V as these will be the only singular vectors that matter. Let's call these $U^{(d)}$ and $V^{(d)}$. Then, we have $ZA^\top = U^{(d)}\Sigma^{(d)}(V^{(d)})^\top$. Now, we can use $Z = \sqrt{n}U^{(d)}$ and correspondingly $A = \frac{1}{\sqrt{n}}V^{(d)}\Sigma^{(d)}$. Now, we also need that $Z^\top \mathbf{1}_n = 0$. Then, $AZ^\top \mathbf{1}_n = V^{(d)}\Sigma^{(d)}(U^{(d)})^\top \mathbf{1}_n = 0$. Now, $V^{(d)}$ is a full-column rank matrix as $d < n$, $d < D$, and columns of $V^{(d)}$ are orthogonal and hence linearly independent. $\Sigma^{(d)}$ is also a diagonal matrix and hence $V^{(d)}\Sigma^{(d)}$ is a full column-rank matrix. Hence, left multiplication to a full-column rank matrix is not zero as a left-inverse exists. Hence, it should be that $(U^{(d)})^\top \mathbf{1}_n = 0$ and hence, $Z^\top \mathbf{1}_n = 0$.

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 shoud be the d–dimensional representation z_i.)

   As this is an affine transformation, we have that $x_* = Az_* + b$. Hence, $Az_* = x_* - b$. However, A might be a full column-rank and thus right inverse might not exist. In this case, we can use $A^\top$ to make LHS have a square matrix and then take the inverse. That is, $A^\top Az_* = A^\top(x_* - b)$. Then, $z_* = (A^\top A)^{-1}A^\top(x_* - b)$. As $A^\top A$ is a symmetric matrix, it might have a pseudo-inverse which can be used.

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 $V^\top z$. 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 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.

  I chose $d = 30$. This evident from the knee present in the SVD values. The justification plot can be found in Figure 2. Note that this is a zoomed-in plot as the range of values is too high.
- For the $2D$ dataset you need to attach the a plot comparing the orignal 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.

  The required numbers are provided in table 2. The reconstruction plots can be found in figure 3. In all the figures, blue circles are the original data points and red x markers are reconstructed points.

Method	Reconstruction Error - 2D	Reconstruction Error - 1000D
Buggy	38.396359916101076	13163.110154859782
Demean	4.097135348936227	8259.74014907716
Norm	9.014756571421726	8268.5467161029
DRO	4.097135348936228	8259.74014907716

Table 2: Reconstruction Errors for Various Methods for data2D.csv and data1000D.csv

- **Questions:** After you have completed the experiments, please answer the following questions.

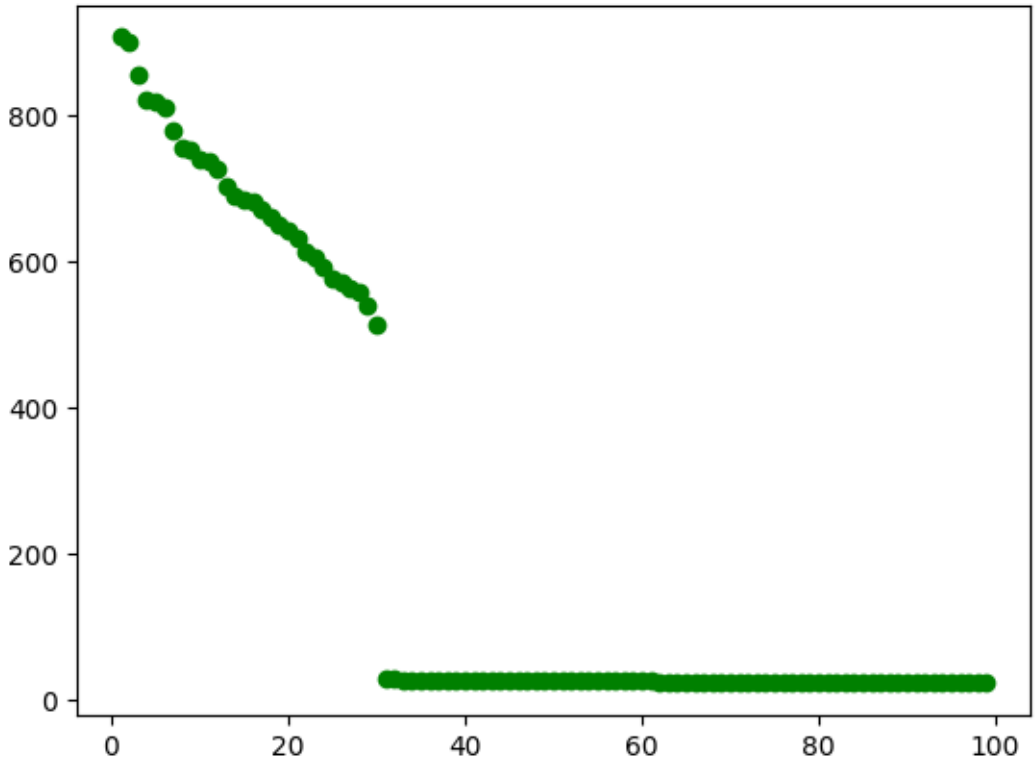
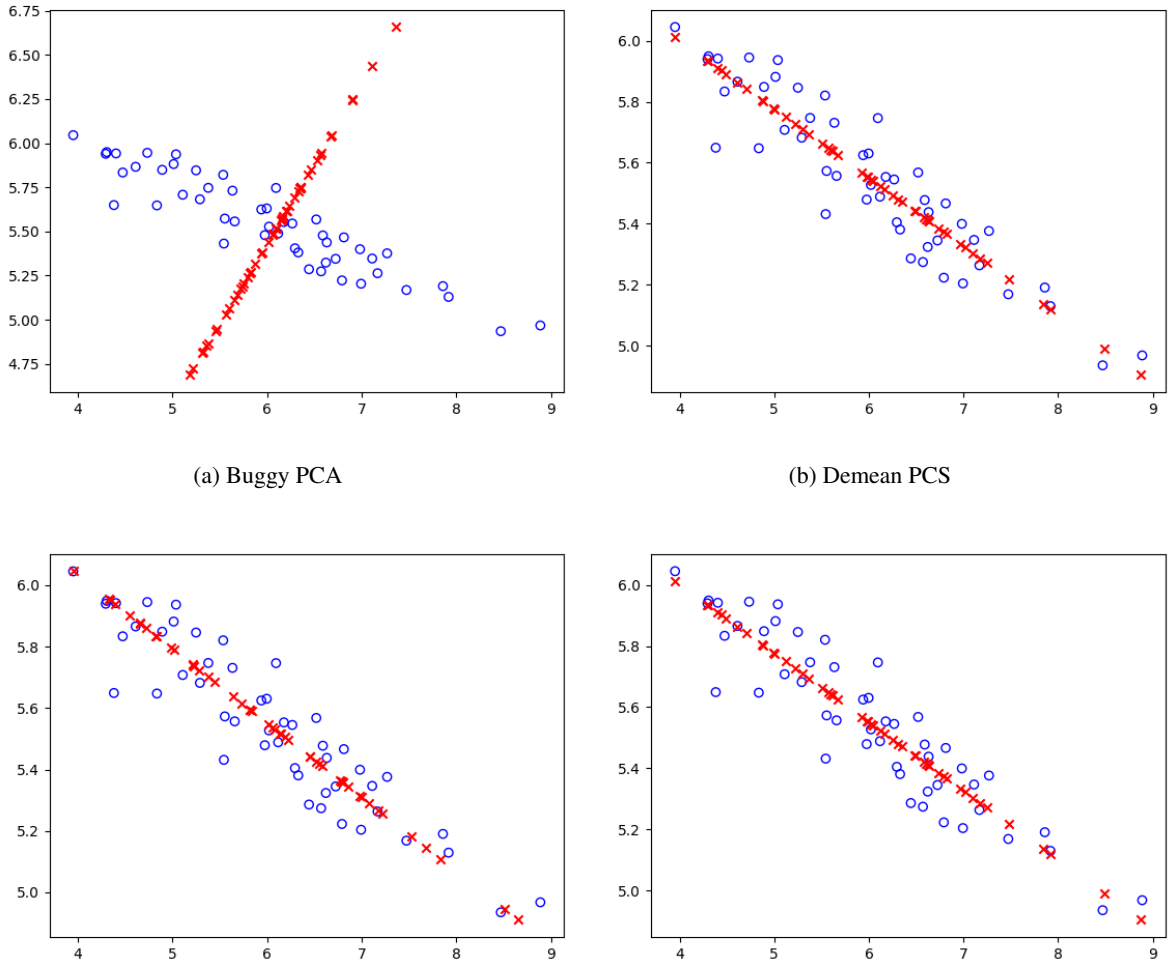


Figure 2: SVD Values From 2 to 99 For data1000D.csv

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? As we are trying to find A such that $x_i = Az_i$, we are always working on the linear domain and hence, we are projecting onto a line that passes through the origin. However, by viewing the dataset, it is clear that the approximation of datapoints don't pass through origin and hence we get a heavy reconstruction error.
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? In the de-meaned version, we remove the mean, do PCA and then for re-construction, add the mean back. In DRO, we do the similar thing as our affine term, b is the mean of the data. Hence, both of them have the same re-construction noise. Moreover, affiner transformation should "fit" the data perfectly as evident from visualizing the data. Hence, it is not surprising that both these operations have the least re-construction noise and both of them minimize the error. Normalized version fails here as we are disrupting the standard deviation of the data prior to the SVD.

- 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)**
  - Implementing reconstructions and reporting results for each one of the 4 methods for the 2 datasets: **(5 points)**
  - Choice of d for $1000D$ dataset and appropriate justification: **(3 Points)**



(a) Buggy PCA

(b) Demean PCS

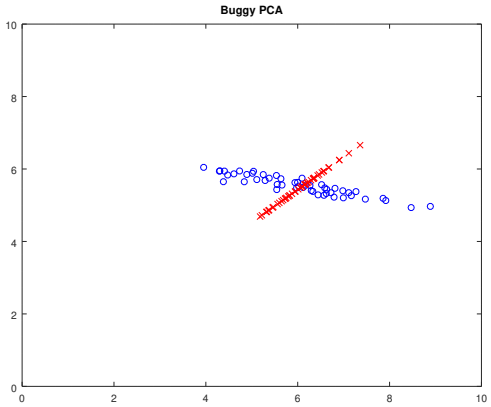
(c) Norm PCA

(d) DRO

Figure 3: Reconstructed and Original Datapoints for data2D.csv

– Questions **(4 Points)**

Answer format:
The graph bellow is in 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