HOMEWORK 5

Ivan Hu
9081641624

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.

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.

   Let $K < K'$. Suppose μ_1^K, f minimize the objective $J(\mu_1^K, f; X_1^n)$; in other words, $\mathcal{J}_K(X_1^n) = J(\mu_1^K, f; X_1^n)$. From these parameters, we construct $\mu_1^{K'}, f'$ as follows: for $1 \leq k \leq K'$, let $\mu_1^{K'}$ be a collection of K' centers, where the first K centers are identical to the centers of μ_1^K and the remaining centers are arbitrary, and let f' be defined by $f'(X_i) = f(X_i)$.

   Notice that $J(\mu_1^{K'}, f'; X_1^n) = J(\mu_1^K, f; X_1^n) = \mathcal{J}_K(X_1^n)$, so $\mathcal{J}_{K'}(X_1^n) \leq \mathcal{J}_K(X_1^n)$.

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.

   Consider the set of centers chosen at each step. After the first step, there are only finitely many possible selections of centers, as each center must be the centroid of a subset of points. This means that eventually the same set of centers is selected twice. As the objective function is strictly increasing as the number of steps increases, this means that every step will past a certain point will return the same set of centers.

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.

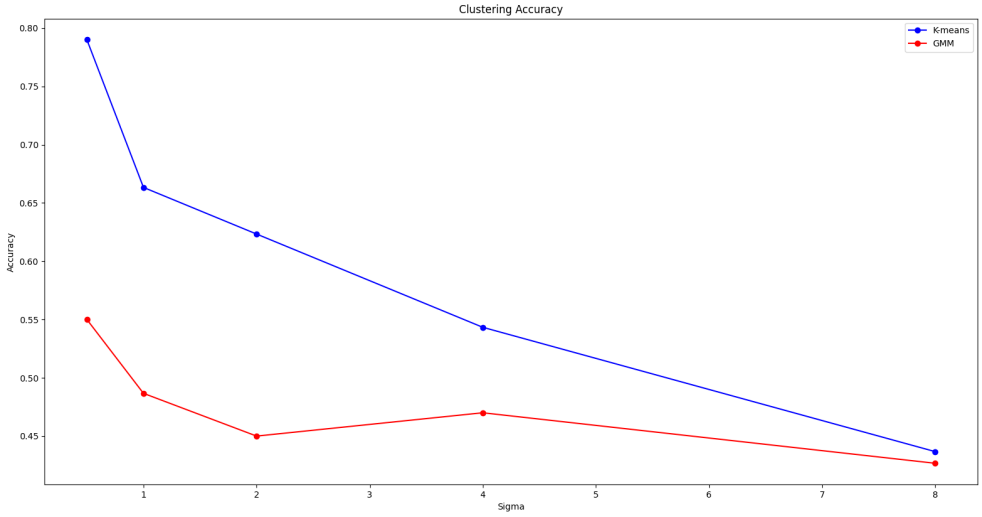
First implement K-means clustering and the expectation maximization algorithm for GMMs. Execute both methods on five synthetic datasets, generated as shown above with $\sigma \in \{0.5, 1, 2, 4, 8\}$. Finally, evaluate both methods on *(i)* the clustering objective (1) and *(ii)* the clustering accuracy. 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.



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.

   Let $a_1, \ldots, a_D$ be the (orthonormal) eigenvectors of $X^\top X$ with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_D$. These exist because $X^\top X \in \mathbb{R}^{D \times D}$ is symmetric, and these correspond to the right singular vectors of X. In particular, a_1 is the first right singular vector of X.

   Let $a \in \mathbb{R}^d$. Notice that $\sum_i (a^\top x_i)^2 = \|a^\top X^\top\|_2^2 = a^\top X^\top X a$. We can decompose $a = c_1 a_1 + \ldots c_D a_D$. Maximizing $\sum_i (a^\top x_i)^2$ given $a^\top a = 1$ is equivalent to maximizing $c_1^2 \lambda_1 + \cdots + c_D^2 \lambda_D$ given $c_1^2 + \ldots c_D^2 = 1$. This occurs when $a = a_1$.

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.

   We can assume inductively that $a_1, \ldots, a_k$ are the first k orthonormal eigenvectors of $X^\top X$ listed above. Notice that the matrix $A_k A_k^\top$ sends $a_1, \ldots, a_k$ to themselves, while it sends $a_{k+1}, \ldots, a_D$ to 0. Therefore, the matrix $I - A_k A_k^\top$ sends $a_1, \ldots, a_k$ to 0, while it sends $a_{k+1}, \ldots, a_D$ to themselves.

Let $a \in \mathbb{R}^D$. We can write

$$\sum_i (a^\top (x_i - A_k A_k^\top x_i)) = a^\top (I - A_k A_k^\top)^\top X^\top X (I - A_k A_k^\top) a.$$

Decomposing $a = c_1 a_1 + \ldots c_D a_D$, the above expression is equal to $c_{k+1}^2 \lambda_{k+1} + \cdots + c_D^2 \lambda_D$. Maximizing this given $c_1^2 + \cdots + c_D^2 = 1$ is achieved when $a = a_{k+1}$.

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 $X = [x_1^\top; \ldots x_n^\top] \in \mathbb{R}^{n \times D}$. 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 $Z = [z_1^\top; \ldots; z_n^\top] \in \mathbb{R}^{n \times d}$. 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).

   We can calculate that

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

   So (A_1, b_1, Z_1) and (A_2, b_2, Z_2) achieve the same objective value.

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, $\mathbf{1}_d = [1, 1 \ldots, 1]^\top \in \mathbb{R}^d$ 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.

   We use the following procedure:

   - Let $\overline{x} = \frac{1}{n} \sum_{i=1}^n x_i$. Define $Y = [x_1^\top - \overline{x}^\top; \ldots; x_n^\top - \overline{x}^\top]$.
   - Compute the singular value decomposition $Y = U \Sigma V^\top$, where U is an orthogonal $n \times n$ matrix, Σ is a diagonal $n \times D$ matrix with nonnegative entries, and V is an orthogonal $D \times D$ matrix.
   - Sort the diagonal elements of Σ in decreasing order, and sort the rows of U and V accordingly. Let the first d columns of U, Σ, V, be U_1, Σ_1, V_1, respectively. We note that $U_1 \in \mathbb{R}^{n \times d}$, $\Sigma_1 \in \mathbb{R}^{d \times d}$, $V_1 \in \mathbb{R}^{D \times d}$.
   - Output $Z = \sqrt{n} \cdot U_1$, $A^\top = \frac{1}{\sqrt{n}} \Sigma_1 V_1^\top$, and $b = \overline{x}$.

By low rank approximation (Eckart-Young), $ZA^\top = U_1\Sigma_1 V_1^\top$ is a rank d matrix that achieves the minimum value of $\|Y - ZA^\top\|_F^2$ over all possible rank d matrices. Shifting all rows by $\overline{x}^\top$, we have that $ZA^\top + \mathbf{1}b^\top$ is a rank d matrix that achieves the minimum value of $\|X - ZA^\top - \mathbf{1}b^\top\|_F^2$ over all possible rank d matrices.

We now check that Z has the desired properties. Since U_1 has orthonormal columns, $S = \frac{1}{n}Z^\top Z = I_d$. Furthermore, since $Y\mathbf{1}_n = 0$, and no zero singular values/vectors are included in Σ_1 and $V_1^\top$, we have that $\bar{Z} = \frac{1}{n}Z\mathbf{1}_n = 0$.

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.)

   We can compute $z_* = A^T(x_* - b)$. Reconstructing x_* is given by $Az_* + b$.

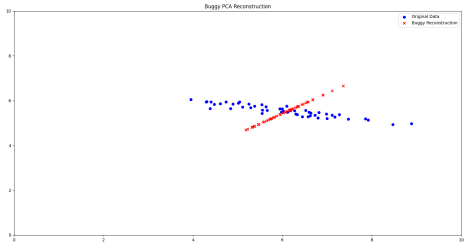
2.3 Experiment (34 points)

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.
- 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.
- **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?
  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?
- 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)**
- Questions **(4 Points)**

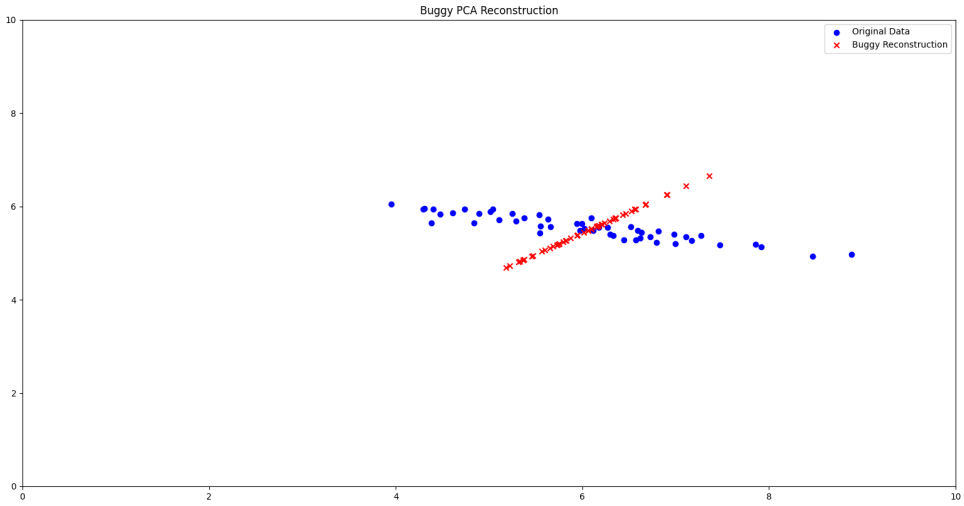
Answer format:
The graph bellow is in example of how a plot of one of the algorithms for the 2D dataset may look like:

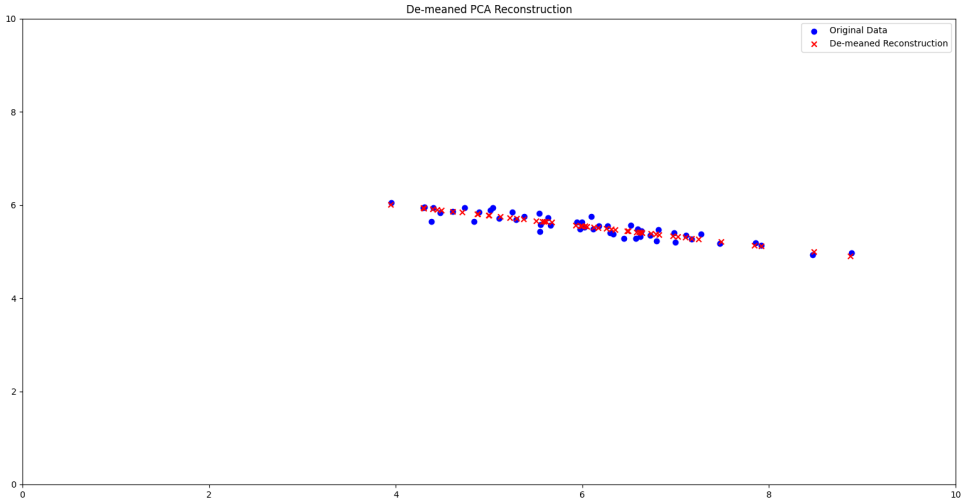


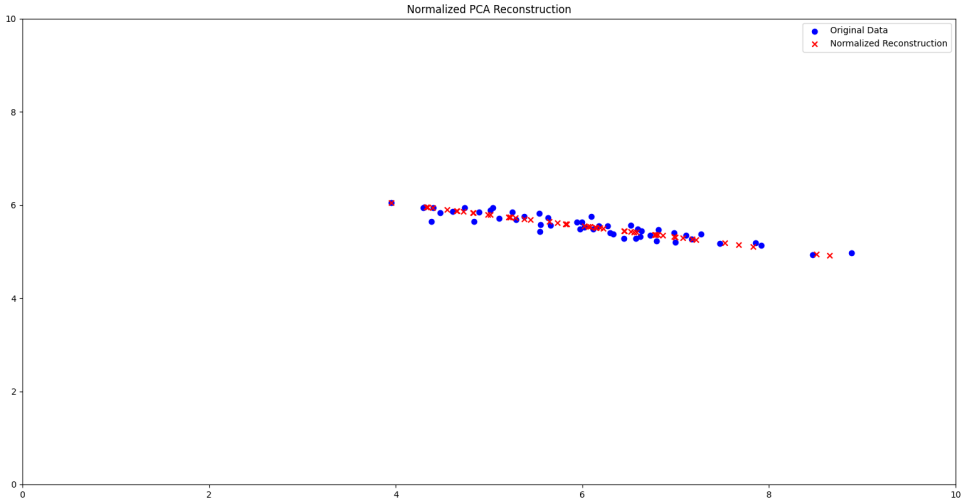
The blue circles are from the original dataset and the red crosses are the reconstructed points.

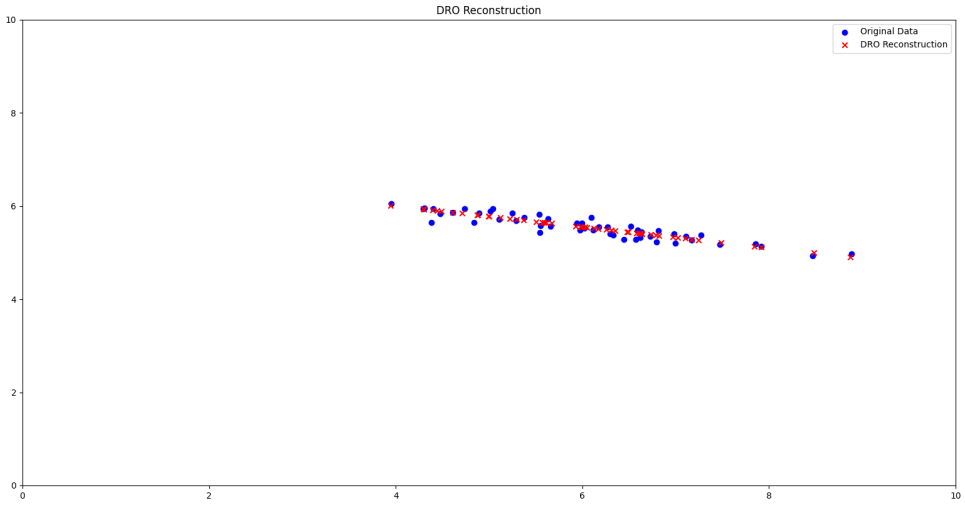
And this is how the reconstruction error may look like for Buggy PCA for the 2D dataset: 0.886903

Solution Visualization of 2D dataset: (see Github for better resolution)









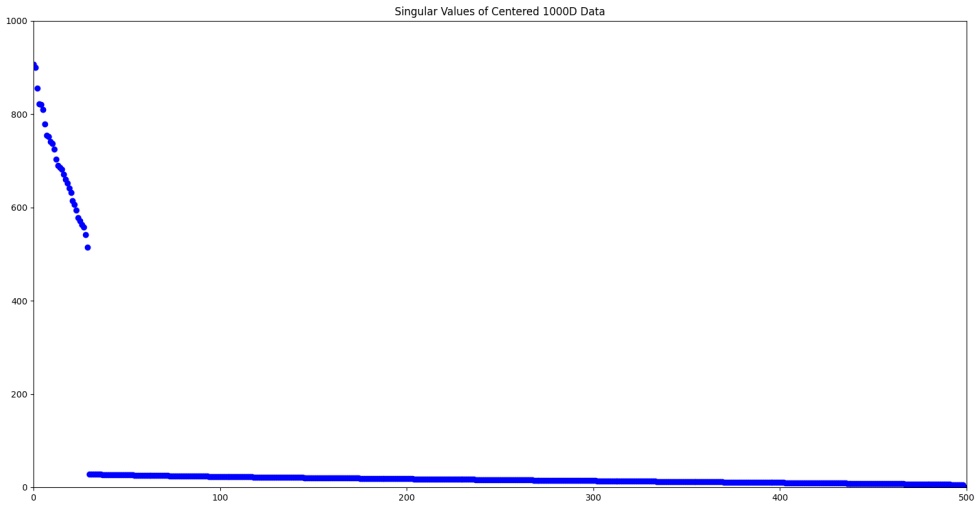
Reports for 2D dataset:

```
data2D.csv report:
Reconstruction Error for Buggy PCA:
44.34515418673973
Reconstruction Error for De-meaned PCA:
0.5003042814256458
Reconstruction Error for Normalized PCA:
2.473604172738534
Reconstruction Error for DRO:
0.500304281425646
```

Reports for 1000D dataset:

```
data1000D.csv report:
d = 30
Reconstruction Error for Buggy PCA:
401365.69931017887
Reconstruction Error for De-meaned PCA:
136522.97948930148
Reconstruction Error for Normalized PCA:
136814.2904988116
Reconstruction Error for DRO:
136522.97948930148
```

To select d for the 1000D dataset, I first centered the data and computed the right singular values of the data matrix in decreasing order. The result was this:



The singular values are sharply divided into two groups. One group contains singular values > 500, while the other contains singular values < 50. This is a good cutoff point. There are precisely $d = 30$ singular values greater than 500.

Questions

- The PCA algorithm is overwhelmingly likely to choose the direction of the mean $\overline{x}$ as the first choice of singular vector. However, this will not provide an actually useful description of the shape of the data. In the case of compressing 2D to 1D, this means that the representation of the data is entirely aligned along the line from the origin to $\overline{x}$, no matter the actual orientation of the data.
- DRO and demeaned PCA achieve identical reconstructions, so it is no surprise that they have the same error. Furthermore, attempting to normalize the standard deviations of each dimension unsurprisingly increases error, because any data that was previously clearly contained in a lower dimensional space is now stretched out.