HOMEWORK 5

Nick Boddy
nboddy
GitHub repo: https://github.com/Nick-Boddy/CS760-HW5-Clustering-PCA

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.

$$\mathcal{J}_K(X^n) = \min_{\mu^K, f} J(\mu^K, f; X^n)$$

$$\mathcal{J}_K(X^n) = \min_{\mu^K, f} \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}(f(X_i) = k) \|X_i - \mu_k\|^2$$

Conceptually speaking, when given K, we wish to find the K centroids that best represent our data X^n. By best representation, we mean to minimize the squared distance of every data point to the centroid it is assigned. Logically, it makes sense to assign each data point to its nearest centroid then (this will be our f).

Thus, now knowing how our rule f should map $\mathcal{X}$ to $\mathcal{K}$, we must consider how to minimize our objective with the centroids μ^K. Since we're mapping each X to the nearest $\mu \in \mu^K$, and our objective is proportional to the sum of the squared distances of all X to their respective μ, the μ^K that minimizes our objective is the set of centroids such that each centroid is at the center of the X belonging to it. And so, minimizing the objective gives us clusters, of a sort.

With an increasing K, we can divide the data further into more clusters of smaller sizes. With this, our centroids will be closer its data points than they were with smaller K.

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.

When X^n is a finite set, and we then have a finite number of clusters (since $k < n$), there is a finite set of possible assignments of data points in X to clusters in μ^k.
By the nature of the K-means algorithm (alternating between assigning each x_i to its nearest centroid in μ^k and recalculating the mean of each centroid), we know that the algorithm must terminate in a finite number of steps because:

1. The objective function, if it changes, decreases due to finding a closer centroid for at least one data point.
2. The objective function is non-increasing, so if it does change, it decreases.
3. If no change to the objective function occurs after an iteration, the program terminates.
4. There is a finite number of possible centroid assignments, and therefore a finite number of changes to the objective function.

Due to these reasons, the K-means clustering algorithm, under the given parameters, will terminate in a 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.

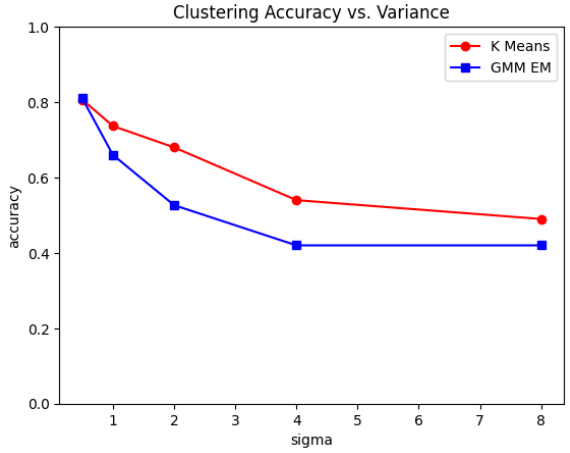
First implement K-means clustering and the expectation maximization algorithm for GMMs. Execute both methods on five synthetic datasets, generated as shown above with $\sigma \in \{0.5, 1, 2, 4, 8\}$. Finally, evaluate both methods on *(i)* the clustering objective (1) and *(ii)* the clustering accuracy.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

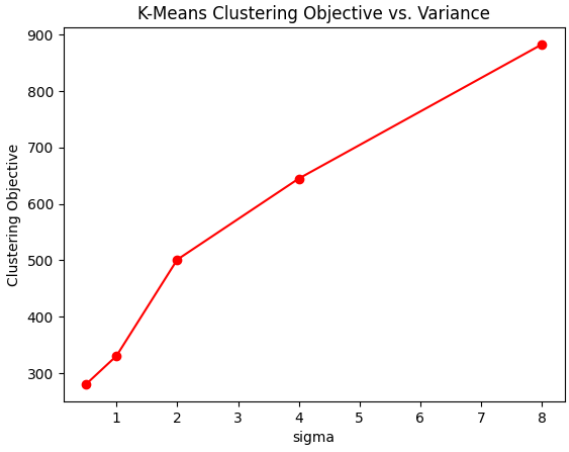
Guidelines:

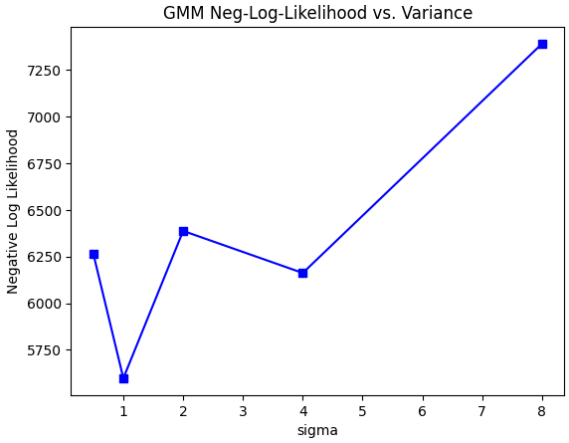
- Both algorithms are only guaranteed to find only a local optimum so we recommend trying multiple restarts and picking the one with the lowest objective value (This is (1) for K-means and the negative log likelihood for GMMs). You may also experiment with a smart initialization strategy (such as kmeans++).
- 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.

With a σ of 0.5, both K-means and GMM achieve about 80% accuracy and taper down K-means clustering seems to perform a bit better than GMM on these generated datasets, but don't differ too much. GMM definitely seems to be more inconsistent and more prone to fall into local minima. I'm not sure as to the best ways to initialize the latent variables, but that could help with its performance for sure.







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.

$$a_1 = \arg\max_{a_1} \frac{1}{n} \sum_{i=1}^{n} (a_1^T x_i)^2$$

$$= \arg\max_{a_1} \frac{1}{n} \sum_{i=1}^{n} a_1^T x_i x_i^T a_1$$

$$= \arg\max_{a_1} a_1^T \left(\frac{1}{n} \sum_{i=1}^{N} x_i x_i^T \right) a_1$$

$$= \arg\max_{a_1} \frac{1}{n} a_1^T X^T X a_1$$

Since $\frac{1}{n}$ is constant, we have

$$a_1 = \arg\max_{a_1} a_1^T X^T X a_1$$

By SVD, $X = U\Sigma V^T$ so

$$X^T X = (U\Sigma V^T)^T (U\Sigma V^T)$$
$$= (V\Sigma^T U^T)(U\Sigma V^T)$$
$$= V\Sigma^T \Sigma V^T \text{ since } U^T U = I$$

Substituting, we now have

$$a_1 = \arg\max_{a_1} a_1^T V\Sigma^T \Sigma V^T a_1$$

Because V is an orthogonal matrix and $V^\top V = I$, it doesn't affect our optimization constraint.

$$a_1 = \arg\max_{a_1} a_1^T \Sigma^T \Sigma a_1$$

$\Sigma^T\Sigma$ is the diagonal matrix of squared singular values: diag$(\sigma_1^2, \sigma_2^2, \ldots, \sigma_D^2)$ such that $\sigma_1^2 \geq \sigma_2^2 \geq \ldots \geq \sigma_D^2$.

Maximizing $a_1^T\Sigma^T\Sigma a_1$ is therefore the same as maximizing $\sigma_1^2 a_1^T a_1$,

Since $a_1^T a_1 = 1$, the optimal a_1 is the one in which a_1 is the unit vector that maximizes the term of σ_1^2, the first squared singular value. This a_1 must therefore be the first right singular vector.

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.

   Since $A_k A_k^\top$ is the projection matrix made by the first k principal components / right singular vectors, $A_k A_k^\top x_i$ is the projection of x_i onto the subspace spanned by $A_k = [a_1, \ldots, a_k]$.
   Therefore, $\tilde{x} = x_i - A_k A_k^\top x_i$ is the component of x_i that is orthogonal to that subspace. Similarly, $\tilde{X} = X - A_k A_k^\top X$, the residual matrix, is the part of X orthogonal to the subspace.

$$a_{k+1} = \arg\max_{a_{k+1}} \frac{1}{n}\sum_{i=1}^{n} (a_{k+1}^\top \tilde{x}_i)^2$$

$$= \arg\max_{a_{k+1}} \frac{1}{n}\sum_{i=1}^{n} (a_{k+1}^\top \tilde{x}_i)(\tilde{x}_i^\top a_{k+1})$$

$$= \arg\max_{a_{k+1}} \frac{1}{n} a_{k+1}^\top \left(\sum_{i=1}^{n} \tilde{x}_i \tilde{x}_i^\top\right) a_{k+1}$$

$$= \arg\max_{a_{k+1}} \frac{1}{n} a_{k+1}^\top \tilde{X}^\top \tilde{X} a_{k+1}$$

Since $\frac{1}{n}$ is a constant, we have

$$a_{k+1} = \arg\max_{a_{k+1}} a_{k+1}^\top \tilde{X}^\top \tilde{X} a_{k+1}$$

From a similar process of part 1 above, we use SVD of $\tilde{X}$ to arrive at

$$a_{k+1} = \arg\max_{a_{k+1}} \tilde{\sigma}_1^2 a_{k+1}^\top a_{k+1}$$

where $\tilde{\sigma}_1^2$ is the first (and largest) squared singular value of the residual matrix. The unit vector a_{k+1} that maximizes this is the corresponding to the first column of $\tilde{V}$, and thus the first right singular vector of $\tilde{X}$. And since $\tilde{X}$ is the residual matrix, the portion of X orthogonal to the subspace spanned by A_k, this a_{k+1} 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 $X = [x_1^\top; \ldots x_n^\top] \in \mathbb{R}^{n\times D}$. 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

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

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

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).

$$\begin{aligned}
J(A_2, b_2, Z_2) = \frac{1}{n}\|X - Z_2 A_2^\top - 1b_2^\top\|_F^2 &= \frac{1}{n}\|X - (Z_1 M^\top + \mathbf{1}p^\top)(A_1 (M^{-1})^\top) - \mathbf{1}(b_1 - A_1 M^{-1} p)^\top\|_F^2 \\
&= \frac{1}{n}\|X - (Z_1 M^\top + \mathbf{1}p^\top)((M^{-1})^\top A_1^\top) - \mathbf{1}(b_1^\top - p^\top (M^{-1})^\top A_1^\top)\|_F^2 \\
&= \frac{1}{n}\|X - (Z_1 M^\top (M^{-1})^\top A_1^\top + \mathbf{1}p^\top (M^{-1})^\top A_1^\top) - (\mathbf{1}b_1^\top - \mathbf{1}p^\top (M^{-1})^\top A_1^\top)\|_F^2 \\
&= \frac{1}{n}\|X - Z_1 A_1^\top - \mathbf{1}p^\top (M^{-1})^\top A_1^\top - \mathbf{1}b_1^\top + \mathbf{1}p^\top (M^{-1})^\top A_1^\top\|_F^2 \\
&= \frac{1}{n}\|X - Z_1 A_1^\top - \mathbf{1}b_1^\top\|_F^2 \\
&= J(A_1, b_1, Z_1)
\end{aligned}$$

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

$$\begin{aligned}
A, Z, b &= \arg\min_{A,Z,b} \left(\frac{1}{n}\|X - ZA^\top - \mathbf{1}b^\top\|_F^2 \right) \\
&= \arg\min_{A,Z,b} \left(\frac{1}{n}\sum_{i=1}^{n} \|x_i - Az_i - b\|^2 \right)
\end{aligned}$$

Let's first consider what value b might take on when minimizing this J function.

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

$$\frac{\partial}{\partial b}\left(\frac{1}{n}\sum_{i=1}^{n} (x_i - Az_i - b)^\top (x_i - Az_i - b) \right) = 0$$

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

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

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

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

$$\frac{1}{n}\sum_{i=1}^{n} b - \frac{1}{n}\sum_{i=1}^{n} x_i + \frac{1}{n}\sum_{i=1}^{n} Az_i = 0$$

Under the constraints that z_i's have zero mean and identity covariance, this becomes

$$n \cdot b = \sum_{i=1}^{n} x_i$$

$$b = \frac{1}{n}\sum_{i=1}^{n} x_i$$

In other words, b ought to be the mean of x_i's.

(a) First, set $b = \frac{1}{n}\sum_{i=1}^{n} x_i$

(b) With b as a $D \times 1$ matrix, subtract $\mathbf{1}_n b^\top$ from X to get $\tilde{X}$.
This $\tilde{X}$ will be mean-centered around zero (and is therefore aligned with what we want Z to be).

(c) To minimize our function $\frac{1}{n}\|\tilde{X} - ZA^\top\|_F^2$, we want $ZA^\top = \tilde{X}$

(d) Deconstruct $\tilde{X}$ using SVD.
$\tilde{X} = U\Sigma V^\top$

(e) We now want to correspond Z with $U\Sigma$ and A with V.

(f) We do this by taking the d approximation of $\tilde{X}$.
Let U_d be $n \times d$, Σ_d be $d \times d$, and V_d be $n \times d$. $U_d\Sigma_d V_d^\top$ is the rank d approximation of $\tilde{X}$.

(g) Approximate $Z = U_d\Sigma_d$ and $A = V_d$.
To satisfy our constraints that Z has zero mean and identity covariance, we should rescale Z. After rescaling, we should then re-estimate A to minimize our J function. We may afterward want to iterate this process of estimating and scaling Z, and estimating A.

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.)

   After our process of approximating our parameters Z, A, and b, when given $x_\star$, which is $D \times 1$, we can do the following to calculate $z_\star$:

   (a) Subtract b from $x_\star$.
   $\tilde{x}_\star = x_\star - b$

   (b) Project $\tilde{x}_\star$ onto the d dimensional space by multiplying by A. This will be $z_\star$.
   $z_\star = A^\top \tilde{x}_\star$

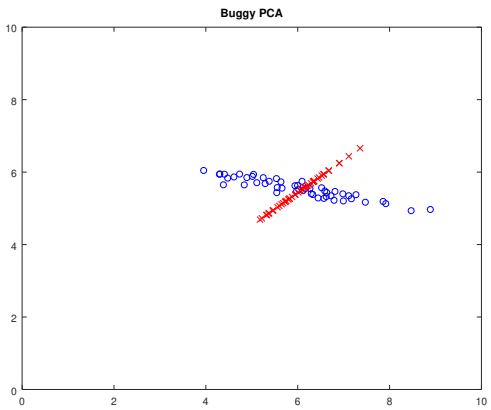
2.3 Experiment (34 points)

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