HOMEWORK 5

>>Martin Diges<<
>>9080689699<<

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.
https://github.com/missingnoglitch0/cs760/tree/main/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.

ANSWER

Suppose $\mathcal{J}_K(X_1^n) = e'$ for some $\mu_1^{K\prime}$ and some f'.
For $K+1$, we can choose

- $f''(X_i) = f'(X_i)$
- $\mu_1^{K+1\prime\prime} = \mu_1^{K\prime} + \mu_{K+1}$ where μ_{K+1} is such that $f''(X_i) = f'(X_i) \neq K+1$. For the traditional algorithm, this means μ_{K+1} satisfies $||X_i - \mu_{K+1}||^2 > ||X_i - \mu_i||^2$ for $i \leq K$

We see then that

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

$$= \sum_{i=1}^{n} \left(\left(\sum_{k=1}^{K} \mathbb{1}(f''(X_i) = k)\|X_i - \mu_k\|^2 \right) + \mathbb{1}(f''(X_i) = K+1)\|X_i - \mu_{K+1}\|^2 \right)$$

$$= \sum_{i=1}^{n} \left(\left(\sum_{k=1}^{K} \mathbb{1}(f''(X_i) = k)\|X_i - \mu_k\|^2 \right) + 0 \right)$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}(f''(X_i) = k)\|X_i - \mu_k\|^2$$

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

$$\mathcal{J}_{K+1}(X_1^n) = \min_{\mu_1^{K+1}, f} J(\mu_1^{K+1}, f; X_1^n) \leq J(\mu_1^{K+1\prime\prime}, f''; X_1^n) = J(\mu_1^{K\prime}, f'; X_1^n) = \mathcal{J}_K(X_1^n)$$

$$\mathcal{J}_{K+1}(X_1^n) \leq \mathcal{J}_K(X_1^n)$$

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.

   ANSWER

   From class, we have the following K-means algorithm:

   (a) Select k cluster centers

   (b) For each point, determine its cluster assignment by finding the closest center in Euclidean space

   (c) Update all cluster centers as the centroids of the points assigned to them

   (d) Repeat steps 2, 3 until there are no changes to the objective

   We define J_{beg} as the objective before an iteration of the algorithm (before step 2).
   We define J_{end} as the objective calculated at the end of step 3.

   If there was no change in point assignment, then $J_{end} = J_{beg} \leftrightarrow$
   If $J_{end} \neq J_{beg}$, then there must have been some change in point assignments during the step.

   - If $J_{end} > J_{beg}$, then at least one point X_i's assignment changed from $f(X_i) = a$ to $f(X_i) = b$ and resutled in $||X_i - \mu_b||^2 > ||X_i - \mu_a||^2$. However, we know $||X_i - \mu_b|| \leq ||X_i - \mu_a||$ for the assignment to have changed, which is a contradiction. Thus, it cannot be that $J_{end} > J_{beg}$, meaning $J_{end} \leq J_{beg}$.
   - If $J_{end} = J_{beg}$, the algorithm terminates.
   - If $J_{end} < J_{beg}$, the algorithm advances to the next iteration. However, the point assignments at the end of Step 3 a_{cur} must be different from other assignments examined in the past a_{past}, i.e. $a_{cur} \neq a_{past}$. This is because $a_{cur} = a_{past} \implies (J_{end} < J_{beg} \leq J_{past} \implies J_{end} < J_{end})$, which is a contradiction. Thus, the number of times the algorithm advances to the next iteration must be bounded by the number of distinct point assignments, which is finite.

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.
ANSWER
Implementations for each method can be found in the GitHub repository, in clustering.ipynb Running the entire notebook should yield results similar to below within a minute:

For the kmeans algorithm:

	Sigma	Objective Value	Accuracy
0	0.5	329.413801	0.846667
1	1.0	484.501959	0.720000
2	2.0	899.716767	0.610000
3	4.0	1888.004757	0.563333
4	8.0	3148.756013	0.550000

For the GMM algorithm:

	Sigma	Objective Value	Accuracy
0	0.5	452.017711	0.743333
1	1.0	579.635779	0.716667
2	2.0	1289.609795	0.550000
3	4.0	3459.513697	0.473333
4	8.0	6319.726476	0.400000

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.

   ANSWER

   My linear algebra skills are not well developed enough to rigorously prove the statement above at this moment, but here is an explanation in case it is of any aid.

   From doing SVD, we get matrices U, Σ, V. The first right vector of X is the first vector of V and is associated with the greatest singular value in Σ. Thanks to V being unitary, a_1 satisfies $a_1^\top a_1 = 1$. Furthermore, since we know we can reconstruct $X = U\Sigma V$, then the greatest $\Sigma V'$ where V' only has one nonzero column will be achieved where that column is a_1, as this maximizes the product of a unitary vector and some scalar, in this case the greatest value in Σ.

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.

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.

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, $A_2 = A_1M^{-1}$, $b_2 = b_1 - A_1M^{-1}p$ and $Z_2 = Z_1M^\top + \mathbf{1}p^\top$. Show that both (A_1, b_1, Z_1) and (A_2, b_2, Z_2) achieve the same objective value J (3).

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.

   ANSWER

   We are trying to minimize the objective

   $$J(A, b, Z) = \frac{1}{n}\sum_{i=1}^{n} \|x_i - Az_i - b\|^2 = \|X - ZA^\top - \mathbf{1}b^\top\|_F^2. \tag{3}$$

   This can be thought of as the Frobenius norm of the difference between the data and our reconstruction/approximation of the data. As we want to minimize this, we know our objective is to obtain Z, A, b s.t. $ZA^\top + \mathbf{1}b^\top \approx X$.

   Per the provided hint, we can obtain an approximation of X by taking its SVD and zeroing out all but the first k singular values.

   One way to proceed which I tried was to set b = the mean of X, then projecting the demeaned X onto the first k singular vectors (A) to obtain Z.

   The below is speculation: I noticed that this problem appeared close in its formulation to least squares regression. We want to regress X on our singular vectors A such that we obtain representations Z and intercept b. Like regression, but instead of obtaining parameters we obtain vectors. My idea, which I did not have time to implement, was to run this regression.

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.)

   $z_i = (x_i - b)A^\top$

2.3 Experiment (34 points)

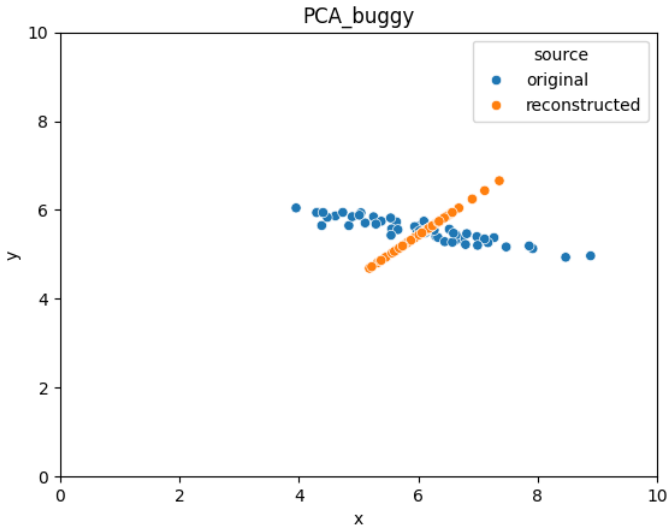
FOR ANSWERS, PLEASE GO TO THE BOTTOM OF THE DOCUMENT
Here we will compare the above three methods on two data sets.

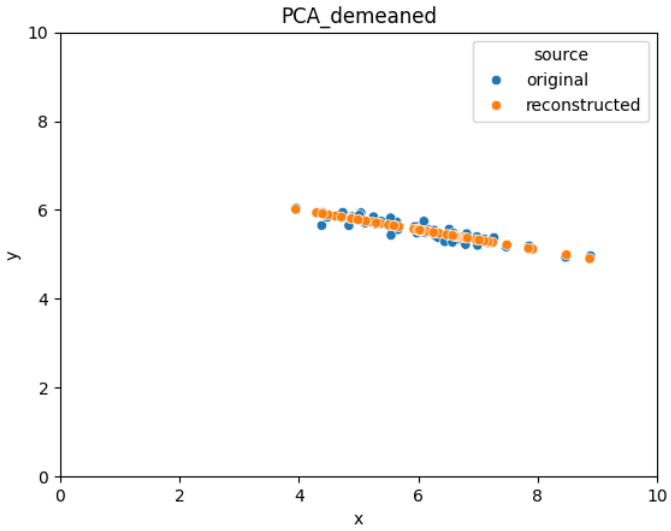
- We will implement three variants of PCA:
  1. "buggy PCA": PCA applied directly on the matrix X.
  2. "demeaned PCA": We subtract the mean along each dimension before applying PCA.
  3. "normalized PCA": Before applying PCA, we subtract the mean and scale each dimension so that the sample mean and 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.

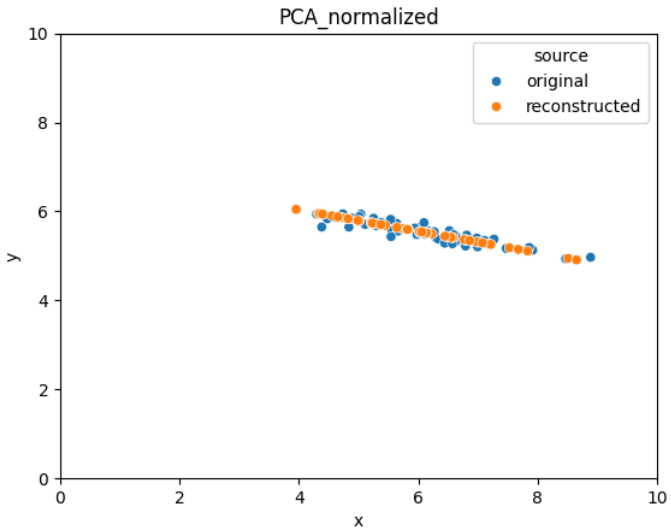
- Please implement code for the methods: Buggy PCA (just take the SVD of X) , Demeaned PCA, Normalized PCA, DRO. In all cases your function should take in an $n \times d$ data matrix and d as an argument. It should return the the d dimensional representations, the estimated parameters, and the reconstructions 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.

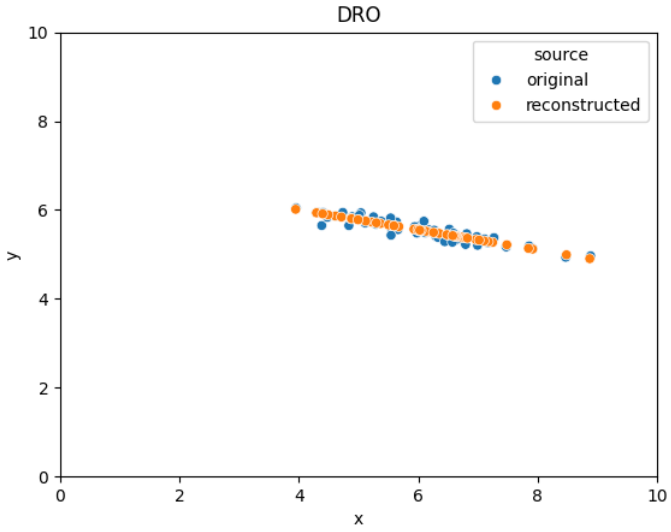
  1. Look at the results for Buggy PCA. The reconstruction error is bad and the reconstructed points don't seem to well represent the original points. Why is this?
     Hint: Which subspace is Buggy PCA trying to project the points onto?
     ANSWER
     With PCA, we try to find a direction vector which captures as much variance as possible. However, since we do not demean for Buggy PCA, there is no frame of reference for what the points are varying relative to. Thus, the points further away from the origin contribute more toward the principal component. This results in the direction vector being roughly in the direction of the cloud of points, as most of the variance relative to the origin is achieved in that direction.
  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?
     ANSWER
     It is not terribly surprising because the methods are somewhat analogous to each other. For one, PCA extracts the d largest eigenvalues and accompanying eigenvectors. DRO extracts the d largest singular values and accompanying singular vectors for its calculations. Both demeaned PCA and DRO keep a vector (mean of data for PCA, vector b for DRO) which is subtracted and added back during "training" and reconstruction respectively.

- Point allocation:
  - Implementation of the three PCA methods: **(6 Points)**
    ANSWER
    Please see the GitHub repository, specifically the PCA.ipynb file. The PCA methods are implemented within a PCA class which supports "training" and reconstruction of points, including the necessary modifications for each variant (buggy, demeaned, normalized). Fields V and Z within the class allow parameters to easily be retrieved.
  - Implementation of DRO: **(6 points)**
    ANSWER
    Please see the PCA.ipynb file referenced above for the DRO class with analogous functionality to the PCA class.
  - Plots showing original points and reconstructed points for 2D dataset for each one of the 4 methods: **(10 points)**
    ANSWER


PCA_buggy
source
original
reconstructed
x
y


PCA_demeaned
source
original
reconstructed
x
y


PCA_normalized
source
original
reconstructed
x
y


DRO
source
original
reconstructed
10
8
6
4
2
0
y
0
2
4
6
8
10
x

	Method	Reconstruction Error
0	PCA_buggy	44.345154
1	PCA_demeaned	0.500304
2	PCA_normalized	2.473604
3	DRO	0.500304

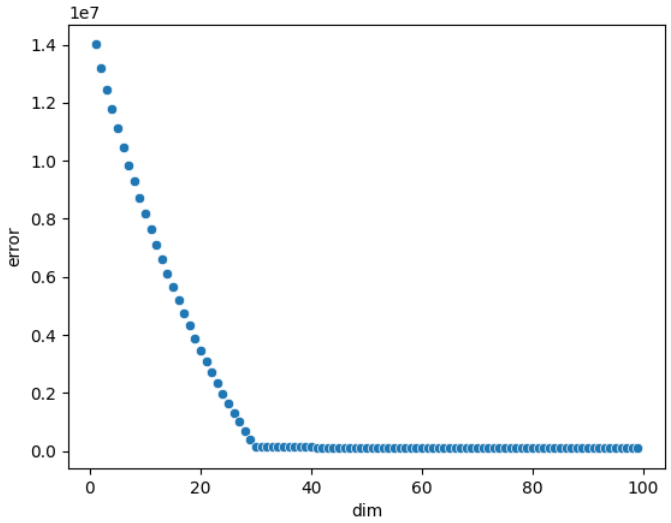
Figure 1: for data$_2D.csv$

	Method	Reconstruction Error
0	PCA_buggy	2.165382e+11
1	PCA_demeaned	7.057740e+08
2	PCA_normalized	6.981090e+08
3	DRO	1.365230e+05

Figure 2: for data$_1000D.csv$

– Implementing reconstructions and reporting results for each one of the 4 methods for the 2 datasets: **(5 points)**
ANSWER
Please see the reconstruct() functions within each of the classes for implementations of reconstructions.

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



Based on the graph shown here (and of course checking the table to ensure the right dimension), I chose dimension of 30, since the gains in reduced error of adding additional dimensions sharply dropped off.

– Questions **(4 Points)**
Questions answered where they are stated (i.e. before the "Point allocation" section