HOMEWORK 5

Sriram Ashokkumar
908 216 3750

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 $\mathcal{J}_K(X_1^n)$ be the minimum value of the K-means objective function for a fixed K, given by:

$$\mathcal{J}_K(X_1^n) = \min_{\mu_1^K, f} J(\mu_1^K, f; X_1^n). \tag{2}$$

We need to show that for any K and $K+1$ such that $K < K+1 < n$, the following holds:

$$\mathcal{J}_{K+1}(X_1^n) \leq \mathcal{J}_K(X_1^n). \tag{3}$$

Assume we have the optimal set of centers μ_1^K and the optimal assignment rule f for K clusters that achieves the minimum $\mathcal{J}_K(X_1^n)$. Now consider adding one more center to form $K+1$ clusters. Place the new center μ_{K+1} at the position of any existing center or at a new position, and define a new assignment rule f' which behaves exactly like f for the first K centers and does not initially assign any point to the new center μ_{K+1}. Therefore, initially we have:

$$J(\mu_1^{K+1}, f'; X_1^n) = J(\mu_1^K, f; X_1^n). \tag{4}$$

Since $\mathcal{J}_{K+1}(X_1^n)$ is the minimum for $K+1$ centers, by definition, it cannot be greater than any other value of the objective function with $K+1$ centers. Thus, we have:

$$\mathcal{J}_{K+1}(X_1^n) \leq J(\mu_1^{K+1}, f'; X_1^n). \tag{5}$$

Combining the last two equations, we get:

$$\mathcal{J}_{K+1}(X_1^n) \leq J(\mu_1^K, f; X_1^n) = \mathcal{J}_K(X_1^n). \tag{6}$$

This proves that $\mathcal{J}_K(X_1^n)$ is a non-increasing function of 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. We consider the standard K-means clustering algorithm. The algorithm proceeds iteratively as follows:

(a) Initialize K cluster centers μ_1^K in the data space.

(b) **Assignment step:** Assign each data point X_i to the nearest cluster center μ_k.

(c) **Update step:** Recompute the cluster centers as the mean of all points assigned to them.

(d) Repeat steps 2 and 3 until the assignment of data points to centers does not change.

To prove that the algorithm terminates in a finite number of steps, consider the following:

(a) The K-means algorithm partitions the data space into a finite number of regions, each corresponding to a Voronoi cell defined by the closest cluster center.

(b) In each iteration, data points are assigned to the nearest cluster center. This assignment is deterministic given the current cluster centers.

(c) When the cluster centers are updated, they are computed as the centroid of the data points assigned to them. This results in the center moving inside the convex hull of the assigned data points, which is a finite space.

(d) If there are no changes to the objective, it implies that the assignment of data points to the cluster centers remains the same. Given that the data set is finite, there are only a finite number of possible assignments.

(e) Therefore, the algorithm must converge in a finite number of steps because there are only a finite number of ways to partition a finite data set into K clusters.

(f) Furthermore, each iteration strictly decreases the objective function J (unless it has reached convergence), which is the sum of squared distances from each data point to its assigned cluster center.

(g) Since the objective function J is bounded below by zero and strictly decreases with each iteration (except at termination), and there are only a finite number of different partitionings of the data set, the algorithm cannot cycle indefinitely and must terminate.

Thus, the K-means algorithm 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.

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.

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

$$J(A, b, Z) = \frac{1}{n}\sum_{i=1}^n \|x_i - Az_i - b\|^2 = \|X - ZA^\top - \mathbf{1}b^\top\|_F^2. \tag{7}$$

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 (7).

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.

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.)

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.

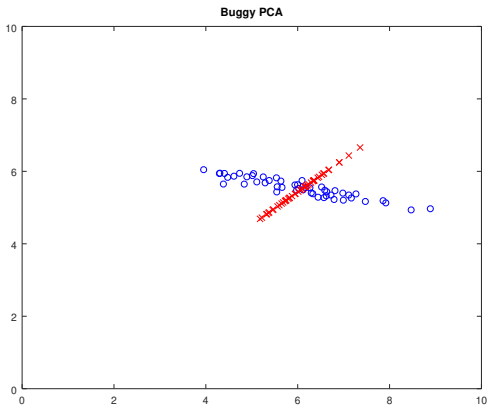
- **Questions:** After you have completed the experiments, please answer the following questions.

  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