COMPSCI 514: Problem Set 4

Released: Tuesday 11/3.

Due: Wednesday 11/18 by 8:00pm in Gradescope.

Instructions:

- Each group should work together to produce a single solution set. One member should submit a solution pdf to Gradescope, marking the other members as part of their group.
- You may talk to members of other groups at a high level about the problems but not work through the solutions in detail together.
- You must show your work/derive any answers as part of the solutions to receive full credit. Include code (screenshots are fine) for any problem where it is used.

1. The Many Applications of SVD (11 points)

1. (2 points) Consider invertible $\mathbf{A} \in \mathbb{R}^{d \times d}$ with SVD $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. Prove that $\mathbf{A}^{-1} = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^T$.
2. (4 points) Consider any $\mathbf{A} \in \mathbb{R}^{n \times d}$ with SVD $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. One of the most classic data fitting methods, least squares regression is: given a vector $\vec{y} \in \mathbb{R}^n$, find:

$$\vec{\beta}_* \in \arg\min_{\vec{\beta} \in \mathbb{R}^d} \|\mathbf{A}\vec{\beta} - \vec{y}\|_2^2. \tag{1}$$

   The rows of $\mathbf{A}$ represent d-dimensional data points, the entries of $\vec{y}$ represent observations at these points, and $\mathbf{A}\vec{\beta}_*$ is the 'line of best fit', which attempts to fit these observations as closely as possible with a linear function of the rows. Prove that $\vec{\beta}_* = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^T\vec{y}$ satisfies equation (1) above. Avoid using any calculus in your proof. **Hint:** The solution will involve a projection matrix.
3. (1 point) Describe in a few sentences how part (2) relates to part (1). What is $\|\mathbf{A}\vec{\beta}_* - \vec{y}\|_2^2$ when $\mathbf{A}$ is square and invertible?
4. (2 points) Prove that $\|\mathbf{A}\|_F^2 = \sum_{i=1}^{\text{rank}(\mathbf{A})} \sigma_i(\mathbf{A})^2$, where $\sigma_i(\mathbf{A})$ is the i^{th} singular value of $\mathbf{A}$. **Hint:** One approach is to use the identity shown in the last problem set, that $\|\mathbf{A}\|_F^2 = \text{tr}(\mathbf{A}^T\mathbf{A}) = \text{tr}(\mathbf{A}\mathbf{A}^T)$.
5. (2 points) The *spectral norm* (or operator norm) of a matrix $\mathbf{A}$ is the maximum amount that applying that matrix can increase the length of a vector. Formally, $\|\mathbf{A}\|_2 = \max_{\vec{x}:\|\vec{x}\|_2=1} \|\mathbf{A}\vec{x}\|_2$. Beware that the notation is a bit confusing – for the matrix $\mathbf{A}$, $\|\mathbf{A}\|_2$ denotes the spectral norm, while for the vector $\mathbf{A}\vec{x}$, $\|\mathbf{A}\vec{x}\|_2$ denotes the standard Euclidean norm.

   Prove that $\|\mathbf{A}\|_2 = \sigma_1(\mathbf{A})$. **Hint:** There are many approaches here. One is to use the Courant-Fischer characterization of the eigenvalues of a matrix discussed in class.

2. Recovering Locations from Distances (13 points + 9 points bonus)

Suppose you are given all pairs distances between a set of n points $\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_n \in \mathbb{R}^d$, with $n > d$. Formally, you are given an $n \times n$ matrix $\mathbf{D}$ with $\mathbf{D}_{i,j} = \|\vec{p}_i - \vec{p}_j\|_2^2$. You would like to recover the location of the original points, up to possible translations, rotations, and reflections, which will not change the pairwise distances.[1] Let $\mathbf{P} \in \mathbb{R}^{n \times d}$ be the matrix with the n points as rows.

1. (3 points) Give an upper bound on rank($\mathbf{D}$). **Hint:** Expand out $\|\vec{p}_i - \vec{p}_j\|_2^2$ as a dot product.

2. (4 points) Show that:

$$(\mathbf{P}\mathbf{P}^T)_{i,j} = -\frac{1}{2}\left[\mathbf{D}_{i,j} - \frac{1}{n}\sum_{k=1}^{n}\mathbf{D}_{i,k} - \frac{1}{n}\sum_{k=1}^{n}\mathbf{D}_{j,k} + \frac{1}{n^2}\sum_{\ell=1}^{n}\sum_{k=1}^{n}\mathbf{D}_{k,\ell}\right].$$

   Hint: Since we can only recover the points up to translations anyways, you can assume without loss of generality that the points are have zero mean. I.e., $\sum_{i=1}^{n}\vec{p}_i = \vec{0}$.

3. (3 points) Describe an algorithm that, given $\mathbf{D}$, uses the formula above to recover $\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_n \in \mathbb{R}^d$ up to rotation and translation. **Hint:** Even if you haven't figured out part (2), you can use the given formula to solve this part.

4. (3 points) Run your algorithm on the U.S. cities dataset provided in `UScities.txt` and plot the output. The distances in the file are Euclidean distances $\|\vec{p}_i - \vec{p}_j\|_2$ so you need to square them to obtain $\mathbf{D}$. Does the output make sense? Plot the estimated city locations and identify a few cities in your plot. Submit your code with the problem set.

5. **Bonus:** (1 point) Plot the spectrum of the distance matrix $\mathbf{D}$ from part (4). Is the rank of $\mathbf{D}$ what was predicted in part (1)? What might be an explanation for any deviations?

6. **Bonus:** (8 points – quite challenging!) The problem of location recovery is closely related to both *triangulation in surveying/mapping* and *matrix completion*. Let's assume that for the U.S. cities dataset we actually only know the distance from every city to three other reference cities. That is, we know just three columns $\mathbf{D}$.

   (a) (3 points) Describe an algorithm that recovers the full distance matrix $\mathbf{D}$ using just these three columns. **Hint:** Given three columns of $\mathbf{D}$, think about how to find four vectors that span all columns of $\mathbf{D}$, using the ideas of parts (1)-(3). Then think about how to recover all the columns of $\mathbf{D}$ from this span.

   (b) (2 points) Describe the geometric intuition, perhaps using a picture, behind why we can recover all distances, and in turn city locations, given just the distances with three reference cities. This intuition doesn't have to exactly align with your algorithm above.

   (c) (3 points) Implement your algorithm and use it to recover the distance matrix $\mathbf{D}$ for the U.S. cities dataset. There will be some error due to approximation errors. Let $\tilde{\mathbf{D}}$ represent your recovered distance matrix. What is $\frac{\|\mathbf{D}-\tilde{\mathbf{D}}\|_F}{\|\mathbf{D}\|_F}$? Did you algorithm work well? Use your recovered matrix $\tilde{\mathbf{D}}$ to recover approximate positions of the U.S. cities. How do your results look in comparison to those of part (4).

[1] Formally, you want to recover the points up to a translation plus multiplication by an orthogonal matrix, which performs a unitary transformation https://en.wikipedia.org/wiki/Unitary_transformation

3. Stochastic Block Model Generalized (9 points + 4 bonus)

In class we applied spectral methods to partition a graph into two large subsets of vertices with relatively few connections between them. We discussed how spectral clustering can be used to partition a graph into $k > 2$ pieces by combining a rank-k spectral embedding with e.g., k-means clustering. In this problem we will consider this method applied to the stochastic block model with a larger number of communities.

Let $G_{n,3}(p, q)$ be the distribution over random graphs where n is divided into three subsets X, Y, Z each with $n/3$ nodes in them (assume for simplicity that n is divisible by 3). Node i, j are connected with probability p if they are in the same subset (X, Y, or Z) and with probability $q < p$ if they are in different subsets. Connections are all made independently.

1. (2 points) Consider drawing a random graph $G \sim G_{n,3}(p, q)$. Let $\mathbf{A}$ be its adjacency matrix and $\mathbf{L}$ be its Laplacian, with nodes sorted by community id. What is $\mathbb{E}[\mathbf{A}]$? What is $\mathbb{E}[\mathbf{L}]$?

2. (4 points) What are the top three eigenvectors and eigenvalues of $\mathbb{E}[\mathbf{A}]$? What are the bottom three eigenvectors and eigenvalues of $\mathbb{E}[\mathbf{L}]$? **Note:** the eigendecompositions of $\mathbb{E}[\mathbf{A}]$ and $\mathbb{E}[\mathbf{L}]$ are not unique. Just describe one valid set of eigenvectors.

3. (3 points) Consider computing $\vec{v}_{n-1}$ and $\vec{v}_{n-2}$, the second and third smallest eigenvectors of $\mathbf{L}$. Then represent node i with the embedding $\vec{x}_i = [\vec{v}_{n-1}(i), \vec{v}_{n-2}(i)]$. Partition the nodes by applying k-means clustering to this embedded data set. Assume that you can find the optimal clustering efficiently. If $\mathbf{A}, \mathbf{L}$ were exactly equal to their expectations, describe how this method would perform in recovering the communities X, Y, and Z. **Note:** You don't need to actually implement the method to answer this question. Just describe how it should work in theory.

4. **Bonus:** (4 points) Generate a 900 node graph from $G_{n,3}(p, q)$ with $p = .1$ and $q = .02$ and partition it with the above spectral clustering algorithm applied to $\mathbf{L}$. Plot the adjacency matrix $\mathbf{A}$, the spectral embedding (i.e., $x_i = [\vec{v}_{n-1}(i), \vec{v}_{n-2}(i)]$ for all i), and the output of the k-means algorithm. How well does the algorithm perform?