Unsupervised Learning HW3

Due: Sat Nov 19, 2022 at 11:59pm

All homeworks (including this one) should be typesetted properly in pdf format. Late homeworks or handwritten solutions will not be accepted. You must include your name and UNI in your homework submission. To receive credit, a typesetted copy of the homework pdf must be uploaded to Gradescope by the due date. You must show your work to receive full credit. Discussing possible solutions for homework questions is encouraged on piazza and with your peers, but everyone must write their own individual solutions. You must cite all external references you used (including the names of individuals you discussed the solutions with) to complete the homework.

1 Readings

Let x be the last digit of your UNI, and define $y := (x \mod 6) + 1$. Read the y^{th} paper from the list below (yes, we'll check). Summarize the main results of your assigned paper, discuss their significance and provide a short proof sketch of their technical results..

- "Nearest neighbor preserving embeddings" by Indyk and Naor.
- "On the Impossibility of Dimension Reduction in ℓ_1" by Brinkman and Charikar.
- "Similarity Search in High Dimensions via Hashing" by Gionis, Indyk and Motwani.
- "Horseshoes in Multidimensional Scaling and Local Kernel Methods" by Diaconis, Goel and Holmes.
- "Some theory for ordinal embedding" by Arias-Castro.
- "Learning Sparsely Used Overcomplete Dictionaries via Alternating Minimization" by Agarwal, Anandkumar, Jain, and Netrapalli.

2 Further Questions on Embeddings

(i) Recall from lecture that:

Theorem: Let q be any integer ≥ 2, and define $D := 2q - 1$. Let (X, ρ) be an n-point metric space. Then there exists a D-embedding of X into ℓ_∞^d with $d = O(qn^{1/q} \ln n)$.

Using the above result, show that every n-point metric space can be D-embedded into ℓ_2^d with $D = O(\log^2 n)$ and $d = O(\log^2 n)$.

(ii) Show that any finite tree (with arbitrary branching factor) can be isometrically embedded into ℓ_1.(*hint:* use induction!)

3 From finiteness to (structured) infinity (and beyond)

Recall (from the previous homework) that a random projection map can approximately preserve interpoint distances between a set of finite number of points. In particular, it was proven that

Lemma: Given a set of n vectors $\{x_1, \ldots, x_n\} = X \subset \mathbb{R}^D$, an $\epsilon > 0$ and $d \in \mathbb{Z}_+$, let G be a $d \times D$ matrix with entries i.i.d. $N(0, 1/d)$. If $d \geq \Omega(\ln(n)/\epsilon^2)$, then with probability at least $3/4$ (over the choice of G)

$$(1-\epsilon)^2\|x_i - x_j\|^2 \leq \|Gx_i - Gx_j\|^2 \leq (1+\epsilon)^2\|x_i - x_j\|^2.$$

Observe that this result only guarantees approximate preservation of distances among *finite* pair of points. (For unbounded $|X|$, i.e. unbounded n, the bound on d relaxes to ∞). In some applications one is interested in preserving interpoint distances amongst *infinite* pairs of points. Unfortunately one cannot get such result (with significant compression) for an arbitrary infinite-size set of points. But if the infinite-size set has some sort of *structure* progress can be made.

Consider a fixed but unknown k-dimensional affine space $S \subset \mathbb{R}^D$. Show that: Pick any $\epsilon > 0$. If $d = O(\frac{k}{\epsilon^2} \log \frac{1}{\epsilon})$, then with probability at least $3/4$ the $d \times D$ Gaussian random matrix G (with i.i.d. entries $N(0, 1/d)$) will have the property that for all $u, v \in S$

$$(1-\epsilon)\|u - v\| \leq \|Gu - Gv\| \leq (1+\epsilon)\|u - v\|.$$

Hint: Consider a finite cover of an appropriate subset of S and apply previously proven Lemma to it. Using that, argue that it implies preservation of interpoint distances between all points in S. *A correct argument will earn you half of the points on this problem.*

It is instructive to note that such results can be extended even further to k-dimensional manifolds!

4 Hardness of Sparse PCA

The 1-dimensional PCA problem is the following: on an input symmetric matrix A, find a direction v that maximizes the explained variance. That is,

$$\begin{aligned} &\max_{v \in \mathbb{R}^d} \ v^T A v \\ &\text{subject to } \|v\|_2 = 1 \end{aligned}$$

PCA has no guarantees on the sparsity of v. Therefore, we study the *Sparse PCA* problem:

$$\begin{aligned} &\max_{v \in \mathbb{R}^d} \ v^T A v \\ &\text{subject to } \|v\|_2 = 1 \\ &\boxed{\|v\|_0 \leq m} \end{aligned}$$

The boxed constraint enforces the sparsity of v. However, this is not a convex optimization problem. As most optimization problems with L0 constraints are, there is no known efficient algorithm solving this. Here, we formally show this by proving that Sparse PCA is **NP**-hard.

(i) A *clique* is a set of vertices such that there is an edge between every pair of vertices. Let G be a graph on n vertices. Prove that the adjacency matrix of G has $n-1$ as an eigenvalue if and only if the vertices of G form a clique.

Hint: One way is to prove that removing an edge from a graph strictly decreases the top eigenvalue of its adjacency matrix. Express the top eigenvalue using the Rayleigh quotient. Another way is to show that the eigenvalue of any adjacency matrix can be at most $n-1$, using a series of inequalities. Then, prove that the inequality is tight if and only if the graph is a clique.

(ii) Prove that the decision version of Sparse PCA is **NP**-complete.

Input: A symmetric matrix A, target sparsity m, target explained variance M

Output: Is there a m-sparse unit vector v such that $v^T A v \geq M$?

You may find it useful that the Clique Problem, defined below, is **NP**-hard:

Input: A graph G and an integer k in binary

Output: Does G contain a clique of k vertices?

5 SDP for Multidimensional Scaling

In Homework 1, we explored classic *multidimensional scaling* (MDS), an algorithm that forms an embedding given the pairwise distances between points in a Euclidean space. What if the pairwise distances are incomplete, or if they don't come from a Euclidean space? In this case, we want to embed the points in higher-dimensional space such that the distances between points are maximized, that is, it is easier to distinguish between the points, while keeping already known distances.

For data points $x_1, ..., x_N$ say that only for pairs $(i, j) \in E$ we know $d(x_i, x_j)$. We want to find some n-dimensional Euclidean representation $y_1, ..., y_N$ that solves the optimization problem:

$$\text{maximize } \frac{1}{2N} \sum_{i,j \in [N]} \|y_i - y_j\|_2^2$$

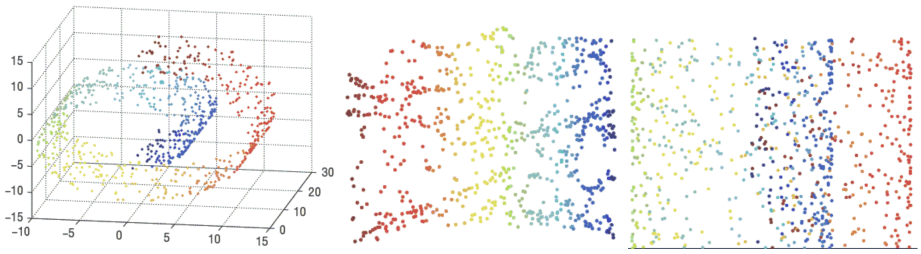
$$\text{subject to } d(x_i, x_j) = \|y_i - y_j\|_2 \text{ for all } (i, j) \in E$$

(i) One might argue that the optimization objective above is flawed because we can maximize distances by taking some y_i to infinity. Indeed, this can happen if there is a point x_i such that there is no x_j such that $(i, j) \in E$. Prove the inverse: if E defines a connected graph then the objective function does not diverge to infinity.

(ii) Reduce the above optimization problem into solving a semidefinite program. *Hint:* From Homework 1, if the Gram matrix (the matrix B of pairwise dot products) is psd, we can recover an embedding that preserves the dot products.

A remarkable application of this method is in manifold learning. Instead of considering all pairwise distances, we consider only the distances between points that are close by, e.g. by constructing a k-nearest neighbors graph. We are keeping local isometry (the constraint) yet pulling unrelated points farther away, effectively "unfolding" the manifold. After we unfold into high dimensions and estimate the pairwise distances, we use classical MDS to make a lower-dimensional representation.

This is an example of the algorithm applied on the "swiss roll":



Original Data Unfolding Linear Embedding

(iii) Create a dataset with 100 points and dimension 6 that demonstrates the effectiveness of the SDP "unfolding". In your pdf submission, include

(a) the 3D plot of the first three dimensions

(b) the 3D plot of the last three dimensions

(c) the 2D embedding from classical MDS

(d) the 2D embedding from unfolding the dataset using SDP and k-nearest neighbors

Colored plots like the one above would be most effective! Submit your code via gradescope.