COMS 4774 Unsupervised Learning Fall 2024
Problem Set #2

Brian Chen - `bc2924@columbia.edu`

September 19, 2024

Problem 1

A uniqueness theorem for clustering by Zadeh, Ben-David

In class, we discussed impossibility theorems on clustering. We saw that under certain properties of clustering algorithms, it is impossible to create an algorithm that satisfies all of the properties simultaneously. In this paper, the authors relax the assumptions of these axioms on clustering, and are thus able to derive clustering algorhtms that *can* satisfy all of the axioms. They are analyzed in the context of the Single-Linkage clustering function and the paper then explores how the relaxation of the richness property to the k-richness property may allow for a more natural axiomatic framework for clustering algorithms.

Definition 1. Scale-invariance. For any distance function d, number of clusters k, and $\alpha > 0$, $F(d, k) = F(\alpha d, k)$

Definition 2. Order-consistency. For any distance functions d, d', number of clusters k, and given a set of edges between points, if the order of edges under distance d is the same as distance d', $F(d, k) = F(d', k)$

The order-consistency property essentially enforces that no operations are done on thee edge weights themselves.

Definition 3. k-Richness. For number of clusters k, range$(F(d, k))$ equals the set of all k-partitions of S.

The k-richness property enforces that all possible partitions are possible under a choice of distance function.

Definition 4. Consistency. For fixed k, distance function d, and a transformation d' of d, then $F(d, k) = F(d', k)$

This property, as discussed in class, essentially enforces that shrinking intra-cluster distance or increasing inter-cluster distances should not change the clustering.

Theorem 1. *Single-linkage clustering satisfies all four properties.*

Proof. Order-consistency: clearly the order does not change under scalar changes in the distance function, and single-linkage is only ever making comparisons of single edges. Intuitively, since the edge weights are not being transformed, order-consistency is preserved.

Scale-invariance: satisfied for the same reason. Since only single edge comparisons are being made, scaling the distance function does not make a difference.

k-richness: we can always derive any k-partitioning via setting the distances of the intra-point edges as 1 for the clusters in the target partitioning Γ, and 2 for all others. Thus the partition will be achieved with the needed number of clusters.

Consistency: define an "inner" edge as an edge where both vertices are contained inside the cluster, and "outer" edges have points in two different clusters. Note that the process of single-linkage iteratively converts the smallest outer edge into an inner edge, in the process also lowering the number of clusters. Consider all possible transformations of our distance measure d. If we shrink an inner edge of d, then clearly the new set of edges will not change since we are not looking at an outer edge to convert into an inner edge. If we change an outer edge to be longer, the new clustering action will also not be changed. Therefore, $SL(d, k) = SL(d', k)$ and consistency is preserved. □

Definition 5. MST-coherence. If d and d' are distance functions s.t. $MST(d) = MST(d')$, e.g. the minimum spanning trees created on these distances are the same, then for all k, $F(d, k) = F(d', k)$.

Theorem 2. *Single-linkage also satisfies this property (along with the other properties in the previous theorem).*

Proof. Any cut made in the MST cut procedure depends entirely on the MST of the input graph. If the MST doesn't change, then, the clustering output clearly won't change either because the MST cut decision is deterministic. Therefore, $F(d, k) = F(d', k)$. □

Theorem 3. *Consistency and k-richness are necessary to characterize single-linkage.*

Proof. Consistency: the order of the edges are never changing, so when we choose the top n edges to cut it doesn't matter if we change the weights so long as the ordinal quality is preserved.

k-richness: we can always construct a d such that all distances are large except among the cluster points that one wants to make into a cluster. This guarantees that under the MST clustering procedure we get whatever st of clusters we want (we're flexible in this choice since we are operating only on the graph, we don't need to make it a metric space). It is necessary because we can take the MST clustering function given by taking the first $n-k+1$ elements of S and returning the other $k-1$ points as clusters of a single point. This returns k total clusters, but clearly is not k-rich since we can never return a partitioning that has no singletons. Therefore there are some clusterings of k clusters that are not achievable, so k-richness is violated. □

Theorem 4. *Given a consistent partitioning function F and a distance function d with edges in ascending order of weight, then for all $k > 0$ if e_p and e_q are both inner edges or outer edges then*

$$F(\langle e_1, e_2, \ldots, e_q, e_p, \ldots e_{nC2}\rangle, k) = F(d, k)$$

Proof. We can always swap outer edges because e_p can be expanded until $w(e_p) > w(e_q)$ while maintaining the clustering order by the consistency property. Similarly, if they are both inner edges, then we can shrink e_q until we get $w(e_p) > w(e_q)$. □

Theorem 5. *Single linkage is the only consistent, k-rich, MST-coherent, and order-consistent partitioning function.*

Proof. Let d be a distance function with edges in ascending order of weight as shown in Theorem 4. Then note that all of the edges up to some t will be inner edges. Therefore, this set of inner edges are uniquely able to identify the resultant partitioning. Now further notice that if there are nay redundant inner edges, they will not affect the MST since the MST will only contain the edge of the smallest weight between two nodes, not any redundant ones. Now the proof builds the SL transformation by changing the distance calculation d to show that this ordering of the edges under d_i is the same as single-linkage.

By k-richness, there is a d_1 such that $F(d_1, k) = SL(d, k) = \Gamma$. Then since by definition the set of edges up to t are inner edges, we can make a new ordering of these t edges such that under an ordering of the edges, these inner edges are $\leqq t$. By consistency the resultant clustering $F(d_2, k) = \Gamma$ still by consistency. We can do a similar action with the outer edges: reorder them with a choice of d to make them the same order as the target d. Now the inner and outer edges are now in the same order as in the SL clustering.

For any redundant inner edges, we need to expand them (we are allowed to do this because of coherence) until they are also placed in the correct order under d.

Finally, under order-consistency we can make the weights identical to d (without changing the order of any of them), which shows that we have effectively reconstructed the SL clustering. □

Theorem 6. *Min-sum k clustering is consistent, k-rich, and scale invariant, but not order consistent or MST consistent.*

Proof. Scale invariant is clearly retained because the objective retains the same ordering of points under linear scaling. We can get k-richness in a similar manner to a previous theorem by setting the distances between points within a target cluster to 1 and all between cluster distances to 2, which we saw before yields the k clusters we need for any choice of partitioning.

Finally, to show that MST coherence and order consistency are not retained, first define a distance d on n points. Arbitrarily split the data points into pairs of size $n/2$ into parts called A, B. For all $x, y \in B, d(x, y) = \epsilon$, and for all points $x \in A, y \in B, d(x, y) = 2$. For all $x, y \in A, d(x, y) = \epsilon$ except one edge that we set to 3. WLOG let this edge be between x_0, y_0. If we run min-sum on this graph, the partitioning will be A, B for $k = 2$. However, if we increase the distance between $x_0, y_0 \to 3n$, then min-sum will put x_0, y_0 in separate clusters. Therefore, order consistency and MST coherence is violated. $\square$

Rates of convergence for the cluster tree by Chaudri and Dasgupta. The paper defines a *high density cluster* for some density f on $\mathbb{R}^d$. The set of all high density clusters gives a cluster tree. This is significant because it prodives a statistical interpretation of clustering, e.g. how different levels of clustering imply things about the distribution from which the data was sampled. For different levels of the clustering, for instance, we get different levels of clustering that tells us how tightly clumped are the clusters. In doing so the authors provide a natural understanding of what clusters are (high density connected regions), and gives a relationship between *all* possible clustering of the cluster tree. The analysis is given in the context of single linkage and its rate of convergence, which is proved to be bounded and finite.

Given a function $f : \mathcal{X} \to \mathbb{R}$ from $\mathbb{R} \to \Pi(\mathcal{X})$, the cluster tree is a function $\mathbb{C}_f : \mathbb{R} \to \Pi(\mathcal{X})$ given by

$$\mathbb{C}_f(\lambda) = \text{connected components of } \{x \in \mathcal{X} : f(x) \geq \lambda\}$$

Lemma 1. *For any $\lambda' \leq \lambda$, there exists a cluster C in the cluster tree such that $C \subseteq C'$*

Proof. Trivially, we can always choose C' as the cluster that contains all of the datapoints. Then any cluster C obeys the property. $\square$

Lemma 2. *For any $\lambda' \leq \lambda$, any C and C' in cluster trees parametrized by λ, λ' respectively, either $C \subseteq C'$ or $C \cap C' = \emptyset$*

Proof. Also immediately obvious via inspection. Either C and C' occupy overlapping domains of $\mathcal{X}$, in which case $C \subseteq C'$ without loss of generality, or they do not and $C \cap C' = \emptyset$. $\square$

Theorem 7. *Theorem 6 in the paper (not copying down the theorem statement for brevity)*

Proof. First part of the proof is *separation*. We get the following set of inequalities

$$\begin{aligned} f(B) \geq \frac{C_\delta d \log n}{n} \quad &\implies \quad f_n(B) > 0 \\ f(B) \geq \frac{k}{n} + \frac{C_\delta}{n}\sqrt{kd\log n} \quad &\implies \quad f_n(B) \geq \frac{k}{n} \\ f(B) \leq \frac{k}{n} - \frac{C_\delta}{n}\sqrt{kd\log n} \quad &\implies \quad f_n(B) < \frac{k}{n} \end{aligned}$$

by setting $C_\delta = 2C_o \log(2/\delta)$. The proof follows over an expectation that holds for all $g \in \mathcal{G}$:

$$-\min(\beta_n\sqrt{\mathbb{E}_n g}, \beta_n^2 + \beta_n\sqrt{\mathbb{E}g}) \ \leq \ \mathbb{E}g - \mathbb{E}_n g \ \leq \ \min(\beta_n^2 + \beta_n\sqrt{\mathbb{E}_n g}, \beta_n\sqrt{\mathbb{E}g}),$$

from which we can use this where $\mathcal{G}$ is the class of indicator functions over balls. We get

$$f(B) \geq \frac{C_o}{n}\left(d\log n + \log\frac{1}{\delta}\right) \implies f_n(B) > 0$$
$$f(B) \geq \frac{k}{n} + \frac{C_o}{n}\left(d\log n + \log\frac{1}{\delta} + \sqrt{k\left(d\log n + \log\frac{1}{\delta}\right)}\right) \implies f_n(B) \geq \frac{k}{n}$$
$$f(B) < \frac{k}{n} - \frac{C_o}{n}\left(d\log n + \log\frac{1}{\delta} + \sqrt{k\left(d\log n + \log\frac{1}{\delta}\right)}\right) \implies f_n(B) < \frac{k}{n}$$

by using the bounds $f(B) - f_n(B) \leq \beta_n\sqrt{f(B)}$, $f(B) - f_n(B) \leq \beta_n^2 + \beta_n\sqrt{k/n}$, and $f(B) - f_n(B) \geq -(\beta_n^2 + \beta_n\sqrt{k/n})$ respectively. For $k \geq d\log n$ the inequalities hold.

Similarly, note that for any $x, x' \in A \cap X_n$, there is a sequence of points $x = x_1, \ldots, x'$ such that for every point in the sequence, the distance between consecutive points is $\leq \alpha r$, which means that x and x' are connected. Since for every point along this path $\pi(x_{i+1})$ is further than $\pi(x_i)$, the process must terminate by continuity. The distance between adjacent points is

$$\|x_i - x_{i+1}\|^2 = \|x_i - \pi(x_i) + \pi(x_i) - x_{i+1}\|^2$$

$$\leq 2r^2 + \frac{2\xi r^2}{\sqrt{d}}$$

$$\leq \alpha^2 r^2$$

Finally we can take $k = 4C^2(d/\epsilon^2)\log(n)$ (some constant number that satisfies the requirements of the inequalities and the above lemma, which means that

$$vr^d\lambda = \frac{k}{n}(1 + \epsilon/2)$$

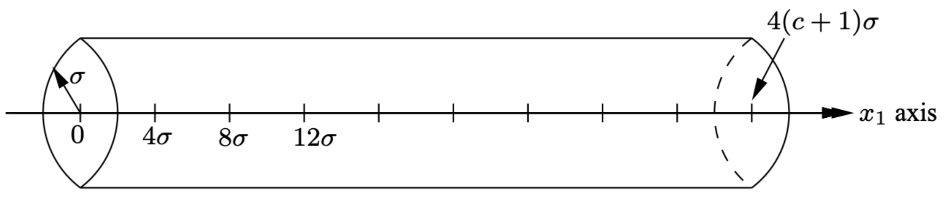
in other words clusters at level λ emerge when r reaches about $(k/(\lambda v_d n))^{1/d}$. □

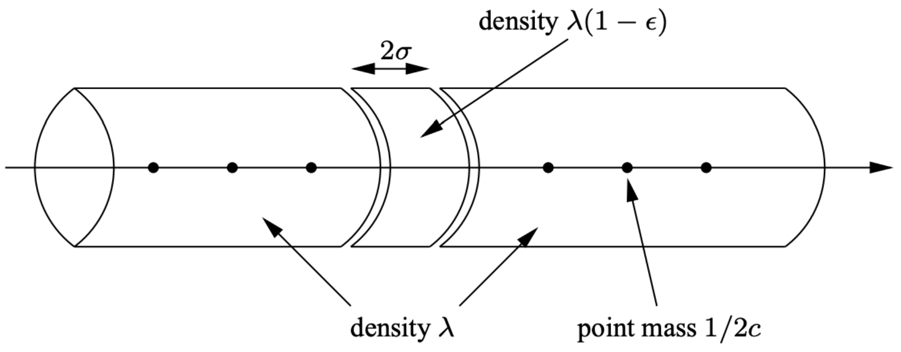
Theorem 8. *For any algorithm with $n \geq 100$ iid samples from some $\theta_i \in \Theta$, and outputs a tree in which the smallest cluster containing $A_i \cap X_n$ is disjoint from the smallest cluster containing $A_i' \cap X_n$, we need to choose*

$$n = \Omega\left(\frac{1}{v_d\sigma^d\lambda\epsilon^2 d^{1/2}}\log\frac{1}{v_d\sigma^d\lambda}\right)$$

for there to exist an input space X, a family of densities Θ, and subsets $A_i, A_i', S_i \subset X$ such that A_i and A_i' are (σ, ϵ)-separated by S_i for density θ_i.

Proof. The proof generally follows a geometric argument in constructing the probability mass needed for the condition in the theorem to hold. Construct a family of probability densities Θ such that θ_i is defined as:





- Density λ on the left and right region of the cylinder, with mass $\lambda v_{d-1}\sigma^d(4(c+1)-2)$
- Density $\lambda(1-\epsilon)$ on $4\sigma i+\sigma, 4\sigma i+3\sigma) \times \sigma B_{d-1}$
- Sharp point masses at $4n\sigma$ along the middle axis
- The remaining mass places in some fixed manner on $\mathcal{X}_1$

The proof argues that if we choose c to be some small constant, then even a small sized sample set X will be likely to contain all of the point masses of mass $1/2c$ each. For the algorithm to separate A_I from A'_I, it needs to connect only the masses between A_I without overlapping the two groups. Using Fano's inequality we get

$$n = \Omega(\log c/\beta)$$

. For larger values of c, we can use the same construction to show that the algorithm is capable of determining I without overlap. We can use Fano's inequality again to get the bound in the problem statement, $n = \Omega((\log c)/\beta)$ $\square$

Problem 2

Given an input to the max diameter problem (a set of points, and a max distance r), we construct an instance of 2-SAT as follows.

1. Assign a variable v_i to each point x_i
2. For each point x_i, if $d(x_i, x_j) > r$, create two new clause constraints $(v_i \vee v_j)$ and $(\neg v_i \vee \neg v_j)$ and append it with an $\wedge$ to our overall boolean expression. Note that these can only both evaluate to true if x_i and x_j are in different clusters.

If there is a satisfaction of the boolean expression constructed here, it means that there is a way to assign points to the two clusters such that the distance constraint is satisfied. If there is not, then there is no way to assign points to the two clusters. This solution can be intuitively interpreted as: each variable v_i defines "is x_i assigned to cluster 1 (wlog)"; the boolean variables thus enforce the pairwise distance constraint.

Problem 3

The energy can be written using the graph Laplacian, where $f \in \mathbb{R}^V$ is the embedding in $\mathbb{R}^1$, as such:

$$E = f^\intercal L f$$

as per Problem 5 on HW0. Furthermore, if f has center of mass at zero, that means that the sum of the components of f is zero

$$\sum_i f_i = 0$$

$$\Rightarrow \|f\|_1 = 0$$

Since the moment of inertia is just the squared components of f, the constraint of having a moment of inertia $= 1$ is equivalent to

$$\sum_i f_i^2 = 1$$

$$\Rightarrow \|f\|_2 = 1$$

In other words, f is a unit vector.

From class we know that minimizing a quadratic form $f^\intercal L f$ can be solved by setting f to the eigenvector with the smallest eigenvalue, subject to whatever other constraints we have. The constraint that $\|f\|_1 = 0$ requires that f is orthogonal to the ones vector, which has eigenvalue zero, so f cannot be the smallest eigenvector. Therefore $f^\intercal L f$ is minimized by setting f to the eigenvector of L with the *second* smallest eigenvalue, similar to what we did for spectral clustering in class.

We can solve the minimization explicitly using lagrange multipliers. Let the lagrange multipliers be λ_1 for the $\|f\|_1$ constraint and λ_2 for the $\|f\|_2$ constraint. The lagrangian is

$$\mathcal{L}(f, \lambda_1, \lambda_2) = f^\intercal L f + \lambda_1 \sum_i f_i + \lambda_2 (\sum_i f_i^2 - 1)$$

The minimum is found by taking the gradient and setting it to zero.

$$\nabla_f \mathcal{L} = 2Lf + \lambda_1 \mathbf{1} + 2\lambda_2 f = 0$$

$$Lf = -\frac{\lambda_1}{2}\mathbf{1} - \lambda_2 f$$

Note that $\mathbf{1}^\intercal L = 0$ since the sum of all of the rows of the Laplacian is just all of the degrees of each node minus the number of total adjacencies, which is clearly 0. Furthermore, $\mathbf{1}^\intercal f = 0$

from the first constraint. Therefore multiplying $\mathbf{1}^\intercal$ on both sides yields

$$\mathbf{1}^\intercal Lf = -\frac{\lambda_1}{2}\mathbf{1}^\intercal\mathbf{1} - \lambda_2\mathbf{1}^\intercal f$$

$$\Rightarrow 0 = -\frac{\lambda_1}{2}\mathbf{1}^\intercal\mathbf{1}$$

so $\lambda_1 = 0$, which we expected from the discussion above. This is the smallest eigenvalue but doens't satisfy the second constraint. Therefore we have

$$Lf = \lambda_2 f$$

which is an eigenvalue problem where f is an eigenvector with eigenvalue λ_2, where λ_2 is the second smallest eigenvalue.

Problem 4

4.1a

A metric that we can use to evaluate the quality of our clustering is the *sillouhette score*, which measures how well clusters are separated from other clusters, and how similar objects with each cluster are to each other. The strengths of this cluster quality evaluation metric is that it does not require any label information; however, it becomes difficult to quantify some of these metrics in higher dimensions (curse of dimensionality, calculating distances between points and their center become less meaningful) and is also less effective if clusters are oddly shaped.

Specifically, silhouette score for a single point is calculated as[1]

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i)}$$

where

$$a(i) = \frac{1}{|C_I| - 1} \sum_{j \in C_i, j \neq i} d(i, j)$$

the mean distance between i and all of the other points in cluster C_I, and

$$b(i) = \min \frac{1}{|C_J| \sum_{j \in C_j} d(i, j)}$$

the average distance of i to the points in the nearest cluster. The overall silhouette score is then the average of all of the silhouette scores of all of the points, and satisfies $-1 \leq S \leq 1$.

4.2a

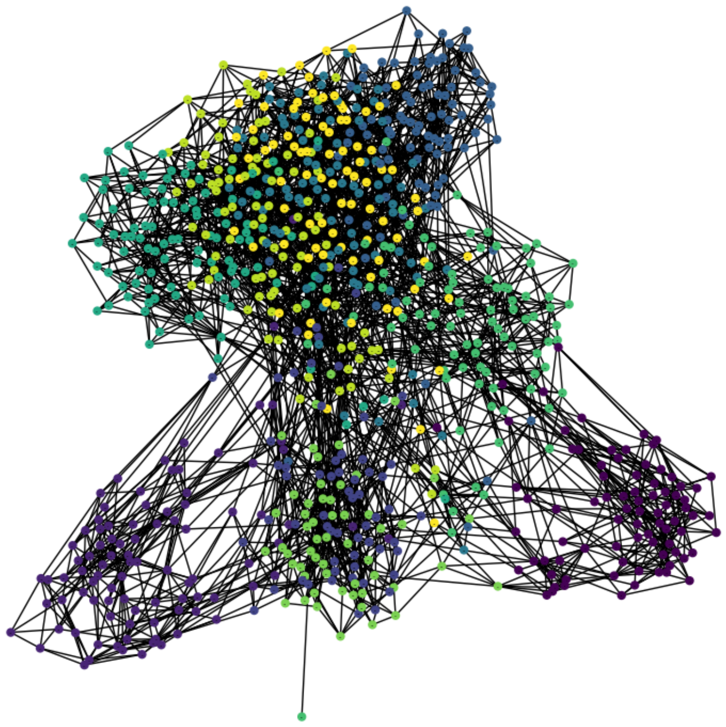
ϵ-neighbor graphs are highly dependent on the choice of ϵ. Too large of a value and the graph can get too strongly connected, whereas too small of a value can lead to disconnected components. ϵ-neighbor graphs are also highly density-dependent, so there may be extremely strongly connected regions in some areas and very sparsely connected regions in others.

A k-nearest neighbor graph can be suboptimal because it may not adequately capture density variations; there could be too few connections in very dense areas and too many in not very dense areas. k-nearest neighbor graphs also suffer from a lack of symmetry, as a point could be in the k nearest neighbors of another point, but not necessarily the same way around – any use of a k-nearest neighbor graph would need to symmetrize this in some way (or be a directed graph).

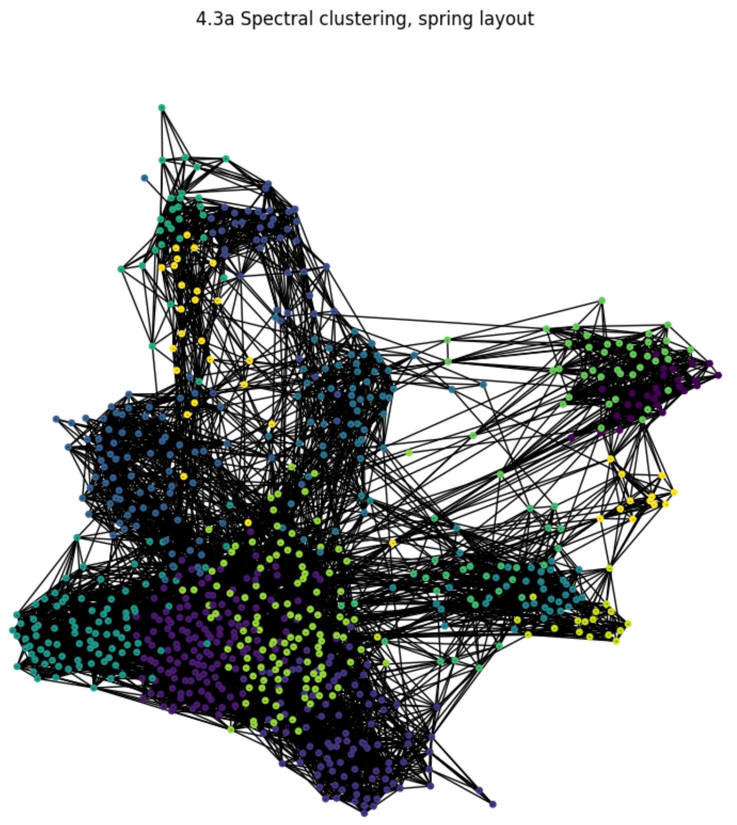
[1]Definition taken from `https://en.wikipedia.org/wiki/Silhouette_(clustering)`

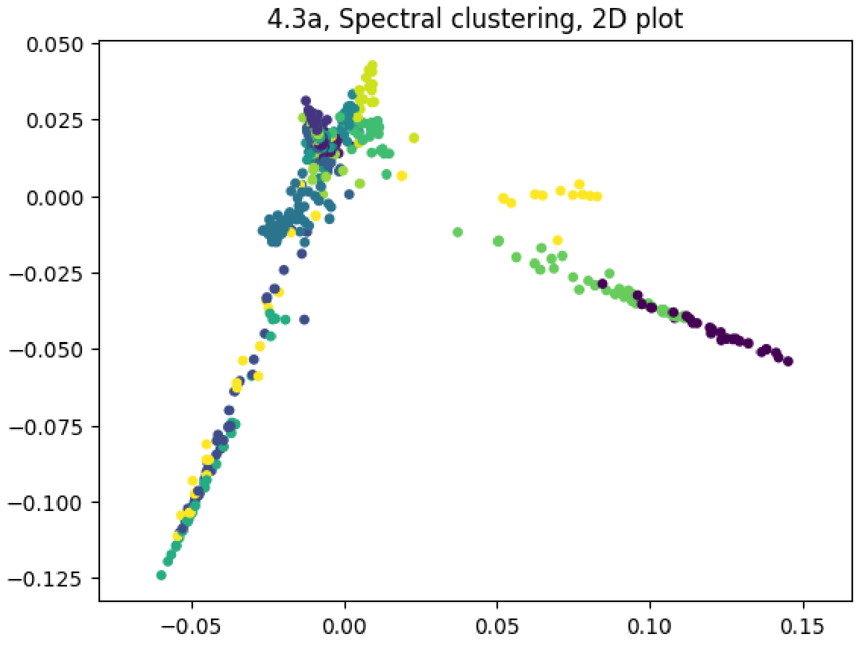
4.2b

We combine these two by creating a k-nearest neighbor graph, and then removing any connections that are more than ϵ away from each other. In this case, we set $k = 5$ and $\epsilon = 0.1$.



4.3a

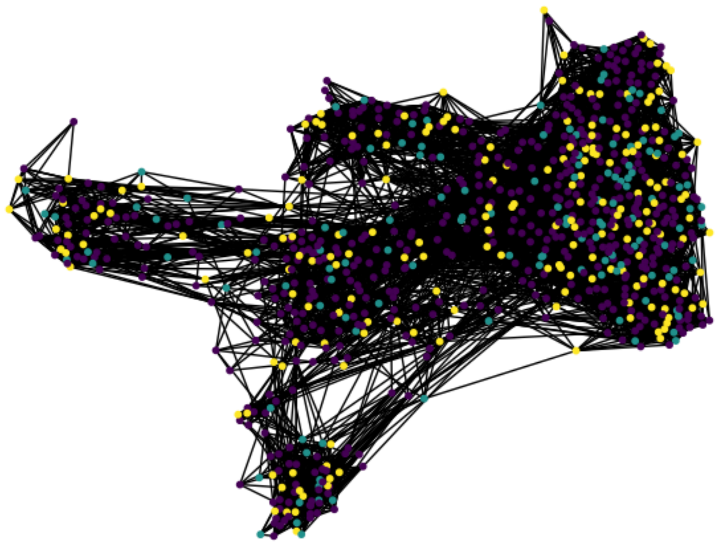

4.3a Spectral clustering, spring layout

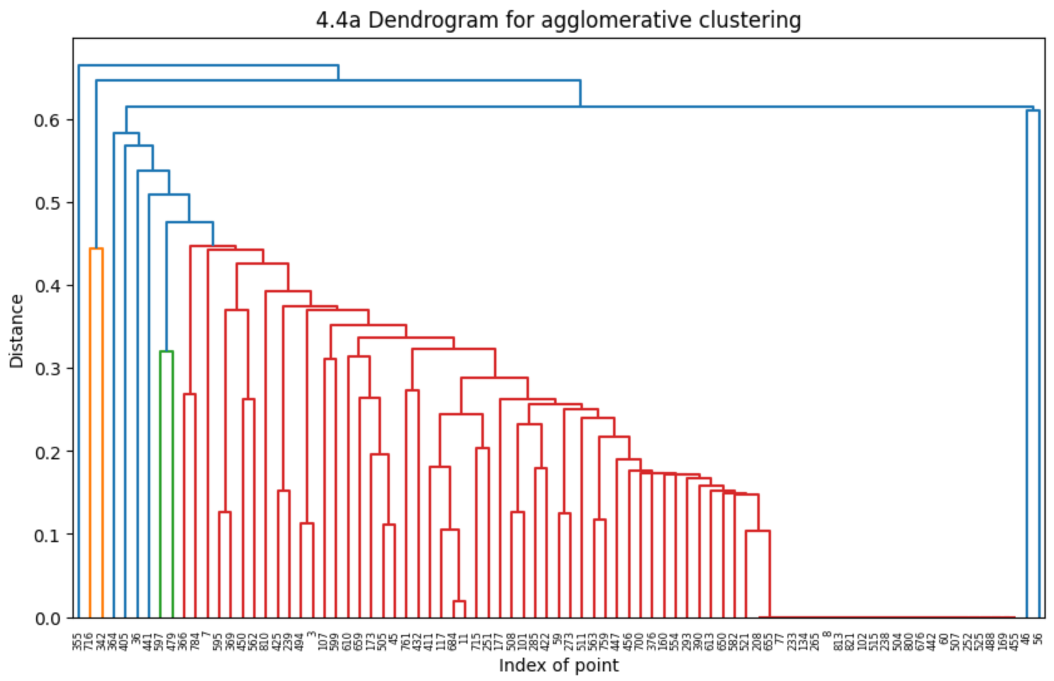

4.3a, Spectral clustering, 2D plot
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100
−0.125
−0.05
0.00
0.05
0.10
0.15

4.3b

Silhouette score for spectral clustering: 0.024433547291264686 based on the 2D spectral layout.

4.4a





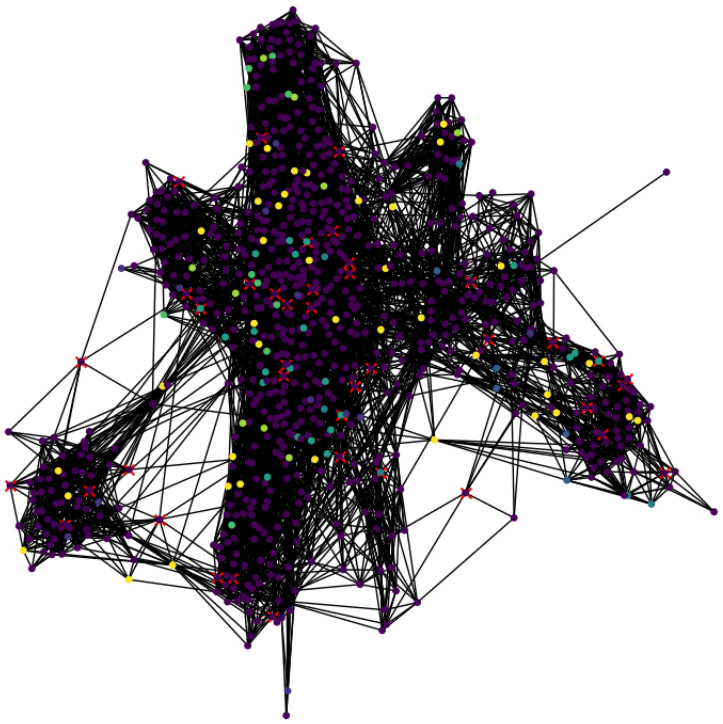
4.4b

In hierarchical clustering, if we have the dendrogram, we can choose the number of clusters by finding a horizontal "cut" of the dendrogram at a point where the distance between splits is the widest. Here, we see that this is widest at the third cluster, so we have chosen $k = 3$.

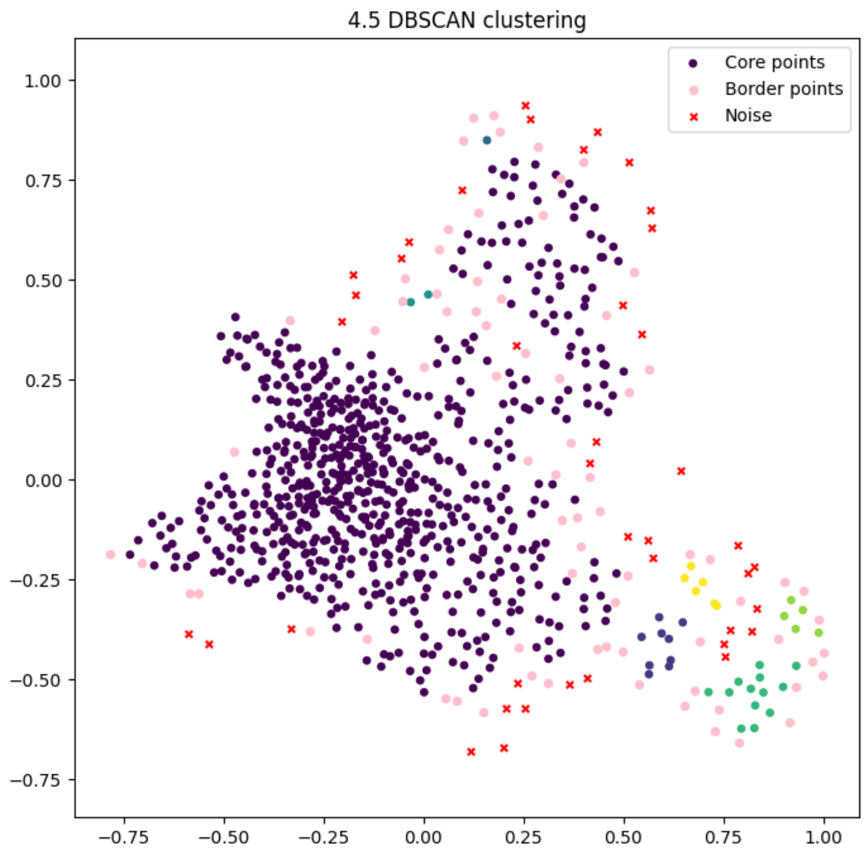
The silhouette score of the clustering using the embedded distances from the graph is -0.01666311482590362.

4.5a

Spring layout plot of the DBSCAN clustering output:



4.5b



The border points here can clearly be seen as the points for each cluster that are on the "edge" of each cluster (marked in pink). Based on the silhouette score we get a clustering quality of 0.025670070054885523, which again is on the lower side because we are not working with nicely *shaped* clusters.

Problem 5

Part (a)

Given that W is a weighted adjacency matrix where W_{ij} is the weight of the edge e_{ij}, and a matrix X, we first note that the goal of the max-cut problem is to partition the vertices into sets S and $V \backslash S$ such that we maximize the weight of the edges that cross the cut. In other words, we want to maximize

$$\sum_{i,j \text{ that cross the cut}} W_{ij}$$

Following the hint, let u be a unit vector in $\mathbb{R}^{|V|}$ that determines whether to assign a vertex to S or $V \backslash S$. Specifically, if $u_i > 0$, then v_i is assigned to set S, and otherwise if $u_i < 0$ then assign v_i to set $V \backslash S$. Now note that

$$\text{Tr}(-WX) = -\sum_{ij} W_{ij} X_{ij}$$

$$= -\sum_{ij} W_{ij} u_i u_j$$

because X is rank one so it can be expressed as $X = uu^\top$. The first row of X corresponds (without loss of generality) to a partitioning of every vertex to either S or $V \backslash S$. We maximize the sum of the weights of the cut by maximizing the expression above, so we have the SDP with the following constraints

$$\begin{aligned} \text{maximize}_X \quad & \text{Tr}(-WX) \\ \text{subject to} \quad & \text{diag}(X) = 1 \\ & X \text{ PSD} \\ & \text{rank}(X) = 1 \end{aligned}$$

note that the diagonal constraint is needed to prevent degenerate solutions.

Part (b)

The relaxed version of the SDP drops the rank constraint,

$$\begin{aligned} \text{maximize}_X \quad & \text{Tr}(-WX) \\ \text{subject to} \quad & \text{diag}(X) = 1 \\ & X \text{ PSD} \end{aligned}$$

Following the problem statement, we decompose $X = V^\top V$. We then take a random vector h from the unit sphere and partition the vertices of the graph with respect to the signs of

$V^\mathsf{T} h$. Let v_i be the i-th column of V. Since h is a random unit vector, it can be interpreted as a normal to a hyperplane that divides $\mathbb{R}^k$ into two halves, and the sign of $v_i^\mathsf{T} h$ tells us which size of the hyperplane v_i lives on.

Given v_i and v_j, the probability that they get separated by a hyperplane defined by the normal vector h is a function of the angle of separation between them. Specifically, WLOG if we fix v_i, then the probability that v_j is on the same side of the hyperplane is

$$\frac{\arccos v_i \cdot v_j}{\pi}$$

so we have

$$\Pr\left[\operatorname{sign}(v_i^\mathsf{T} h) \neq \operatorname{sign}(v_j^\mathsf{T} h)\right] = \frac{\arccos v_i \cdot v_j}{\pi} = \frac{\theta}{\pi}$$

where $\theta = \arccos v_i \cdot v_j$.

The actual expected cost of cutting an edge (i, j) is therefore

$$W_{ij} \cdot \frac{\theta_{ij}}{\pi}$$

and the expected cost overall of the cut is

$$\mathbb{E}[\text{cut}] = \sum_{i<j} W_{ij} \cdot \frac{\theta_{ij}}{\pi}$$

To maximize the cut, we would want v_i and v_j to be on opposite sides of the hyperplane because this maximizes the angle between them. This is optimally

$$\text{optimal cut} = \frac{1}{2} W_{ij}(1 - \cos\theta_{ij})$$

from which the hint immediately leads us to the answer, namely that

$$\begin{aligned}\mathbb{E}[\text{cut}] &= \sum_{i<j} W_{ij} \cdot \frac{\theta_{ij}}{\pi} \\ &\geq \frac{7}{8} \cdot \frac{1}{2}(1 - \cos\theta) \\ &\geq \frac{7}{8} \cdot \text{optimal}\end{aligned}$$

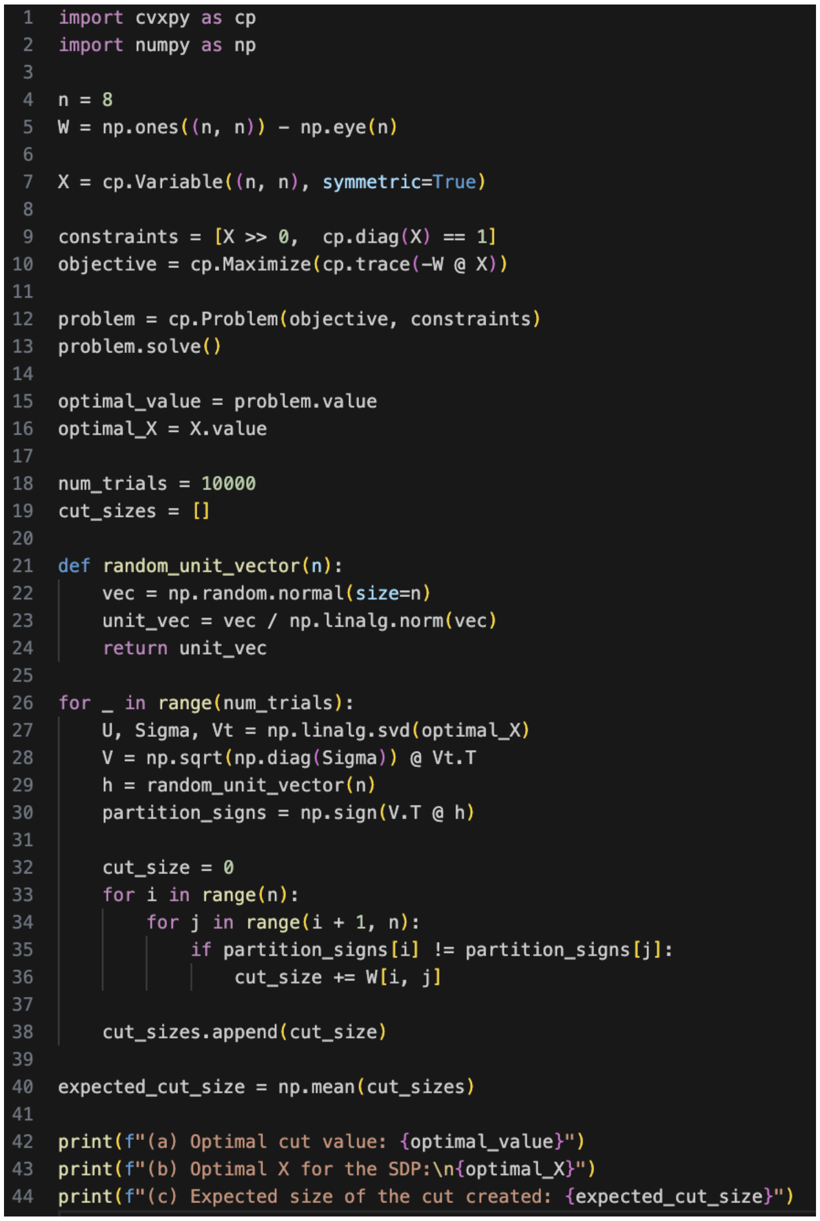
Part (c)

(a) Optimal cut value: 8.000000008862301

(b) Optimal X for the SDP:

$$\begin{bmatrix} 1 & -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} \\ -\frac{1}{7} & 1 & -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} \\ -\frac{1}{7} & -\frac{1}{7} & 1 & -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} \\ -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} & 1 & -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} \\ -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} & 1 & -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} \\ -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} & 1 & -\frac{1}{7} & -\frac{1}{7} \\ -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} & 1 & -\frac{1}{7} \\ -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} & -\frac{1}{7} & 1 \end{bmatrix}$$

(c) Expected size of the cut created: 14.2341



```python
import cvxpy as cp
import numpy as np

n = 8
W = np.ones((n, n)) - np.eye(n)

X = cp.Variable((n, n), symmetric=True)

constraints = [X >> 0,  cp.diag(X) == 1]
objective = cp.Maximize(cp.trace(-W @ X))

problem = cp.Problem(objective, constraints)
problem.solve()

optimal_value = problem.value
optimal_X = X.value

num_trials = 10000
cut_sizes = []

def random_unit_vector(n):
    vec = np.random.normal(size=n)
    unit_vec = vec / np.linalg.norm(vec)
    return unit_vec

for _ in range(num_trials):
    U, Sigma, Vt = np.linalg.svd(optimal_X)
    V = np.sqrt(np.diag(Sigma)) @ Vt.T
    h = random_unit_vector(n)
    partition_signs = np.sign(V.T @ h)

    cut_size = 0
    for i in range(n):
        for j in range(i + 1, n):
            if partition_signs[i] != partition_signs[j]:
                cut_size += W[i, j]

    cut_sizes.append(cut_size)

expected_cut_size = np.mean(cut_sizes)

print(f"(a) Optimal cut value: {optimal_value}")
print(f"(b) Optimal X for the SDP:\n{optimal_X}")
print(f"(c) Expected size of the cut created: {expected_cut_size}")
```

Problem 6

Part (a)

Recall from PSET1 that

$$\Pr\big[\|Gv\|^2 > (1+\epsilon)^2\big] < e^{-cd\epsilon^2}$$

for unit vector v. Define $v \equiv x_i - x_j$. Since we are assuming x_i and x_j are distinct, v is a nonzero vector, so we can rewrite it as $v = \frac{v}{\|v\|}\|v\|$. Plugging this in to the statement to be proved yields

$$\Pr\big[\|Gv\|^2 > (1+\epsilon)^2\|v\|\big] = \Pr\left[\left\|G\frac{v}{\|v\|}\|v\|\right\|^2 > (1+\epsilon)^2\|v\|\right]$$

$$= \Pr\left[\|v\|\left\|G\frac{v}{\|v\|}\right\|^2 > (1+\epsilon)^2\|v\|\right]$$

Let $u = \frac{v}{\|v\|}$, which by definition is a unit vector. We then see that

$$= \Pr\big[\|Gu\|^2 > (1+\epsilon)^2\big]$$

$$< e^{-cd\epsilon^2}$$

since the statement from PSET1 held only for unit vectors.

Part (b)

There are $\binom{n}{2}$ ways to select a pair x_i, x_j from X to test the assertion. By the union bound we have

$$\Pr\big[\exists x_i, x_j \in X \text{ s.t. } \|Gx_i - Gx_j\|^2 > (1+\epsilon)^2\|x_i - x_j\|\big] \le \sum_{i,j} \Pr\big[\|Gx_i - Gx_j\|^2 > (1+\epsilon)^2\|x_i - x_j\|\big]$$

$$\le \binom{n}{2} e^{-cd\epsilon^2}$$

$$\le n^2 e^{-cd\epsilon^2}$$

Part (c)

For there to be a 3/4 chance that the inequality is satisfied, we use the result from Part (b). Namely, we want to find d such that

$$n^2 e^{-cd\epsilon^2} \leq \frac{1}{4}$$

$$\Rightarrow \ln n^2 - cd\epsilon^2 \leq \ln 1/4$$

$$2\ln n - cd\epsilon^2 \leq -\ln 4$$

$$d \geq \frac{2\ln n + \ln 4}{c\epsilon^2}$$

so the smallest possible value of d is

$$\boxed{d = \frac{2\ln n + \ln 4}{c\epsilon^2}}$$