Andrew Liu
Nakul Verma
Unsupervised learning
Columbia University
October 27, 2022

UML HW2

Problem 1: Max diameter clustering

Consider an instance $D = ((x_1, \dots, x_n), r)$ of the maximum diameter clustering problem. We note that if x_i is assigned to cluster 1, then no point x_j such that $\rho(x_i, x_j) > r$ can be assigned to the same cluster. Hence, we construct a 2-SAT instance ϕ as follows: For every pair (x_i, x_j) in $(x_1, \dots, x_n)$ (there are $\binom{n}{2}$ such pairs), if $\rho(x_i, x_j) > r$, add a condition $u_i \mathrm{XOR}\, u_j = (u_i \vee u_j) \wedge (\neg u_i \vee \neg\, u_j)$ to the 2-SAT expression, where u_i and u_j are the cluster assignments (0 or 1).

Clearly, if a solution to D exists, then each XOR clause should return true (i.e. the 2-SAT problem ϕ is satisfiable) and vice versa. Hence, D reduces to ϕ in polynomial time. Therefore:

$$\mathrm{maximum\_diameter\_clustering} \leq_{\mathrm{p}} \mathrm{2SAT}$$

Since 2-SAT is known to be easy, maximum diameter clustering is easy as well.

Problem 2: Mass-spring embeddings

The problem says we consider the 1-dimensional embedding. We start with all points at 0 so the initial energy, moment of inertia, and polar moment of inertia are all zero. We then adjust the points to reach a configuration of $J = \sum_i f_i^2 = \|f\|^2 = 1$.

We want to minimize the change in energy: $\Delta\mathcal{E} = \sum_{e \in E} l_e^2 - 0 = \sum_{i,j \in E} (f_i - f_j)^2 = f^T L f$

Hence, our problem is:

$$\min_f f^T L f$$

Such that: $\|f\| = 1$, $\sum_i f_i = 0$ (center of mass at 0) and I_L is the same for any line L passing the origin. Since we are working in 1D, we assume all lines passing the center of mass are perpendicular to the dimension. Hence, all lines L automatically induce the same I_L (since the distance from a point to L is the distance of a point to the origin).

We see that this is the same as approximate solution to the min-cut problem. Minimizing the energy is thus equivalent to minimizing the curvature of f on the nodes.

We see that assigning the same value (0) to all entries of f gives energy 0 but $J = 0$ as well. So this the "lowest energy state" $f = 0$ does not work.

At the next lowest eigenvector, if $|V|$ is even, we must have half of the entries in f be $-1/\sqrt{|V|}$ and the other half be $+1/\sqrt{|V|}$. If the number of nodes $|V|$ is odd, we have one of the nodes assigned to 0 and the others to $\pm 1/\sqrt{|V-1|}$. This assignment satisfies $\|f\| = 1, \sum_i f_i = 0$ (center of mass maintained at 0 for simplicity and WLOG) and minimizes the energy/curvature. This achieves energy:

$$f^T L f = \sum_{i=1}^{|V|}\sum_{j=1}^{|V|} f_i L_{ij} f_j = \sum_{i=1}^{|V|}\sum_{j=1}^{|V|} f_i D_{ij} f_j - \sum_{i=1}^{|V|}\sum_{j=1}^{|V|} f_i A_{ij} f_j = \sum_{i=1}^{|V|} \deg(i)\, f_i^2 - \sum_{i=1}^{|V|}\sum_{j=1}^{|V|} f_i A_{ij} f_j$$

Look at the final two terms separately. The double sums have $|V|^2$ terms. **First consider if $|V|$ is even**. Then half of these $|V|^2$ terms $f_i f_j$ will be $-\frac{1}{|V|}$ and the other half $f_i f_j = \frac{1}{|V|}$. We note:

$$\sum_{i=1}^{|V|} \deg(i)\, f_i^2 = \frac{1}{|V|}\sum_{i=1}^{|V|} \deg(i) = \frac{2|E|}{|V|}$$

And for the second term, we note that there is a minimum of one $(i, j) \in E$ edge such that $f_i f_j = -1/|V|$ (since the graph is connected). Hence:

$$\sum_{i=1}^{|V|}\sum_{j=1}^{|V|} f_i A_{ij} f_j = 2 \sum_{(i,j)\in E} f_i f_j \leq 2\left(1 * \left(-\frac{1}{|V|}\right) + (|E| - 1) * \left(\frac{1}{|V|}\right)\right) = \frac{2|E| - 4}{|V|}$$

So:

$$f^T L f \geq \frac{2|E|}{|V|} - \frac{2|E| - 4}{|V|} = \frac{4}{|V|}$$

We note that the minimum energy decreases with $|V|$ because the more nodes we have, the less separation between them we need to achieve the polar moment of inertia $J = 1$, and hence the lower the energy.

If $|V|$ is odd, repeat the same analysis, except set one of the nodes to 0. WLOG, let that the 0 embedding node be the zeroth node.

$$\sum_{i=1}^{|V|} \deg(i)\, f_i^2 = \frac{1}{|V| - 1}\sum_{i=2}^{|V|} \deg(i) = \frac{2|E| - \deg(0)}{|V| - 1} \geq \frac{2|E| - 2}{|V| - 1}$$

The last inequality arises from the fact that having the zero node connect with one positive node and one negative node achieves less energy than having the positive and negative nodes connect directly, since $\left|\frac{1}{\sqrt{|V|-1}}-0\right|^2+\left|-\frac{1}{\sqrt{|V|-1}}-0\right|^2=\frac{2}{|V|-1}\leq\left|\frac{1}{\sqrt{|V|-1}}-\left(-\frac{1}{\sqrt{|V|-1}}\right)\right|^2=\frac{4}{|V|-1}$. This, along with minimizing the energy, require us to choose $degree(0)=2$, where one of its neighbors is positive and the other is negative. With this in mind, consider the other term:

$$\sum_{i=1}^{|V|}\sum_{j=1}^{|V|}f_iA_{ij}f_j=2\sum_{(i,j)\in E}f_if_j\leq 2\left((|E|-2)*\left(\frac{1}{|V|-1}\right)\right)=\frac{2|E|-4}{|V|-1}$$

$$f^TLf\geq\frac{2|E|-2}{|V|-1}-\frac{2|E|-4}{|V|-1}=\frac{2}{|V|-1}$$

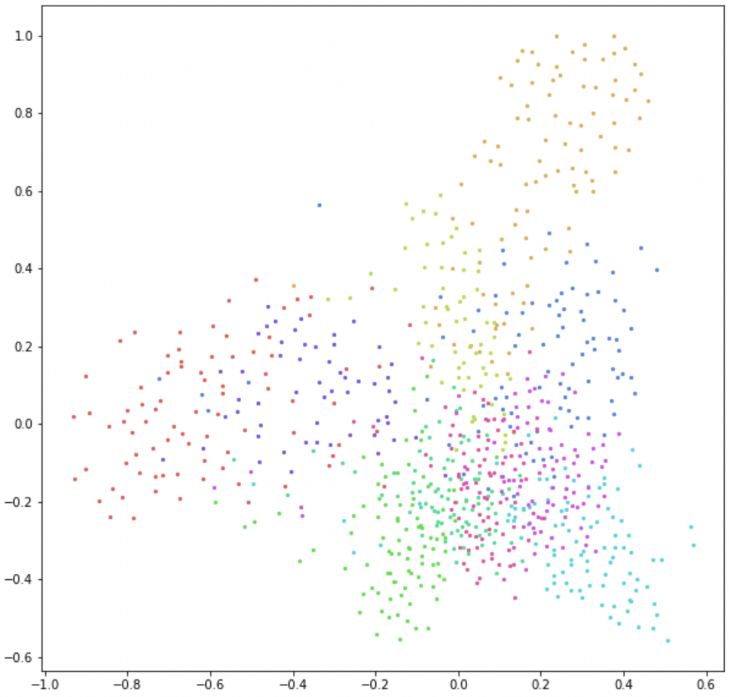
So, in conclusion:

$$\mathcal{E}\geq\begin{cases}\frac{4}{|V|}\text{ if }|V|\text{ even}\\ \frac{2}{|V|-1}\text{ if }|V|\text{ odd}\end{cases}$$

Sanity check:
If $|V|=2$, we must have $f=\left(-\frac{1}{\sqrt{2}},+\frac{1}{\sqrt{2}}\right)$. These two nodes must be connected, so $\mathcal{E}_{min}=2$.
If $|V|=3$, we must have $f=\left(-\frac{1}{\sqrt{2}},0,+\frac{1}{\sqrt{2}}\right)$. The minimum energy achieved with this configuration is $\frac{1}{2}+\frac{1}{2}=1$.
If $|V|=5$, we do get a minimum energy of $\frac{1}{2}$.

Problem 3: Discovering butterfly species

Visualizing the data:



3.1.a:

We can use the Rand Index, which measures the similarity between true clusters and cluster assignments. Given true clustering C and computed assignments K, the metric is given by:

$$RI = \frac{a+b}{C_2^{n_{samples}}}$$

Where a is the number of pairs of elements that are assigned same cluster in C and same cluster in K, b is the number of pairs of elements that are assigned different clusters in C and different clusters in K, and $C_2^{n_{samples}}$ is the total number of possible pairs in the dataset. In other words, if C says a pair of data are same/different, we want K to agree.

Advantages of this metric:

- Interpretable in the sense that it is proportional to the number of pairs whose labels are the same (or different) in both the true assignment and computed assignments
- Versatile and can be used to evaluate all types of clustering algorithms
- Random assignments get scores close to zero for any choice of number of clusters and number of samples.

Disadvantages of this metric:

- Requires knowledge of the ground truth classes.
- Rand Index may be close to 1 even if the clusterings differ. This is because most pairs will be assigned “different” in C and K, which leads to high score.

Source cited:

https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation

3.2.a:

Downsides of ϵ-neighborhood graphs:

1. It is unclear what value for ϵ is appropriate. If the similarities generally take on small values, then an ϵ that is too large may lead to removal of edges between nodes that are in fact related.
2. Additionally, ϵ-neighborhood graphs are unaware of degree. For example, a node with high degree but whose edge weights are all less than ϵ will have all its edges removed even though this node is highly related to its neighborhood.

Downsides of k-nearest neighbor graphs: (keeping k largest edges)

1. We do not know how to pick k.
2. Some nodes might have much larger degrees than others so picking k nearest neighbors might lead to disproportionate pruning of their nodes.
3. We might end up removing edges that have large weight if the top k weights are even larger. These edges should in principle be preserved.

3.2.b:

A way to combine ϵ-neighborhood and k-nearest neighbors:

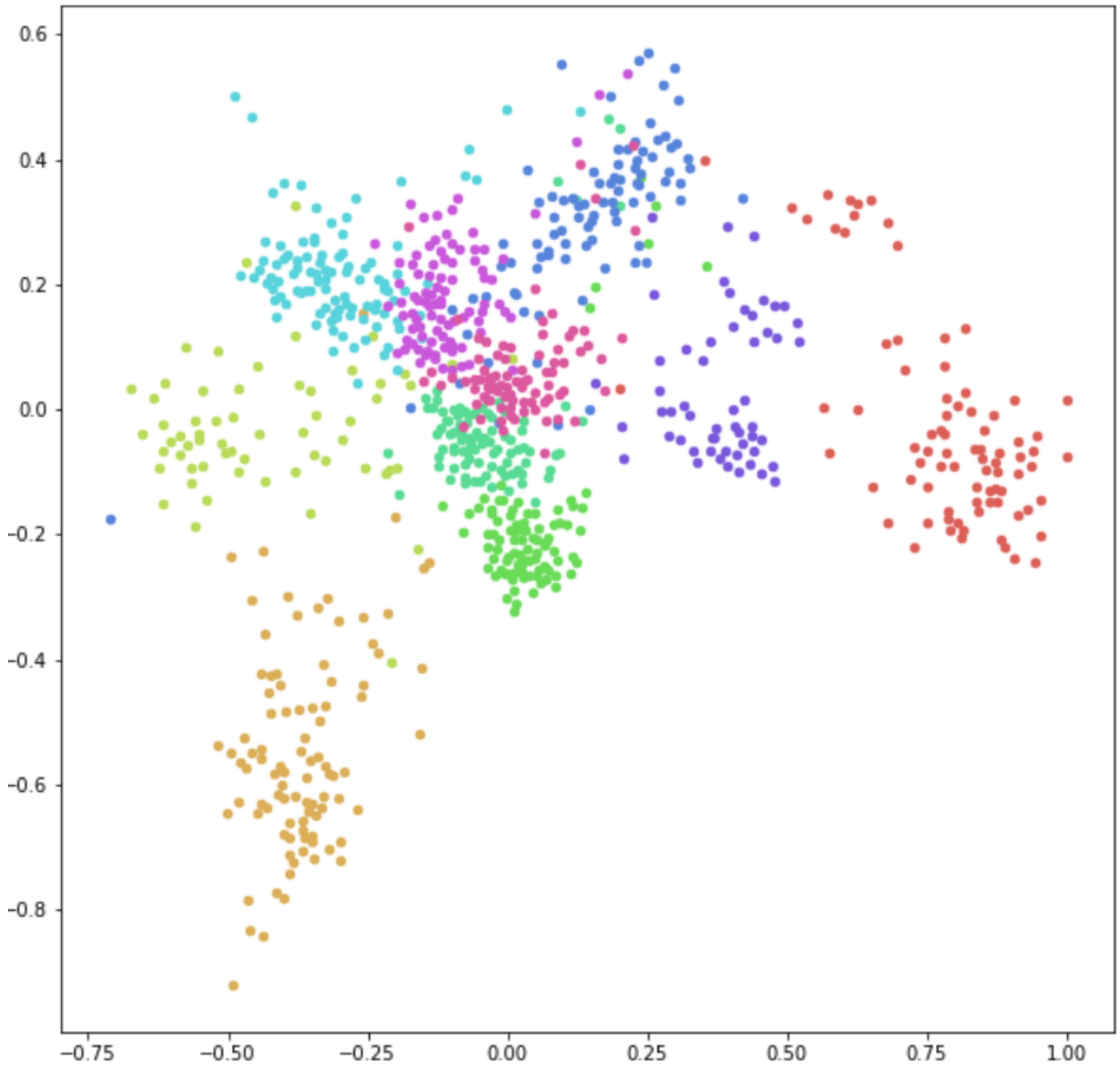
Let $n_{<\epsilon}^{i}$ be the number of edges around node i that have weight less than ϵ. If $\deg(i) - n_{<\epsilon}^{i} < k$ (i.e. removing all the $n_{<\epsilon}^{i}$ edges would give degree less than k), then iteratively remove the weights less than ϵ until we reach $\leq k$ edges. Note that this also means if $\deg(i) < k$ then we don't remove any edges. This ensures we don't remove too many edges for nodes with low edge weights but many edges.

If $\deg(i) - n_{<\epsilon}^{i} > k$, i.e. there are many nodes with sufficiently large weights, then we should remove the ones with weight $< \epsilon$ and keep all the remaining edges. This ensures we don't remove edges of large weight.

All in all, this ensures that if a node began with more than k edges, it will never go decrease below k edges and if a node starts with k or fewer edges it won't be pruned at all.

Alternative preprocessing:

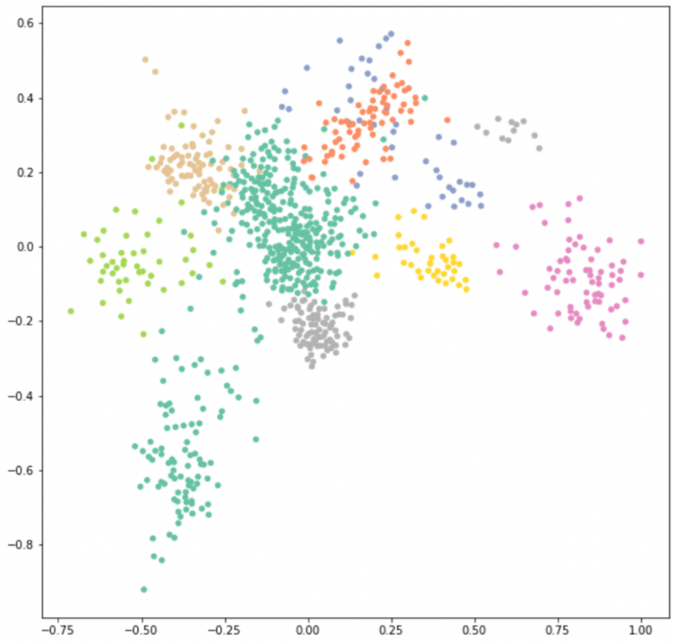
We want to use the ϵ-neighboring strategy to "weaken" the ties between different clusters. However, this also weakens dependencies within clusters. To remedy this, for each node i, we can consider its nearest neighbor j, then j's next nearest neighbor(s) k and connect i and k by adding a node (i, k), whose weight will be the product of $weight(i, j)$, $weight(j, k)$.



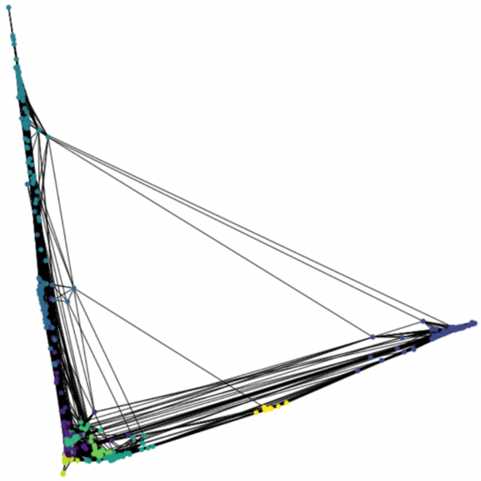
(We see that the weakening procedure was quite effective at "propelling" different clusters apart in the spring embedding. Some clusters however, proved too similar to effectively separate.)

3.3a

Spring layout:

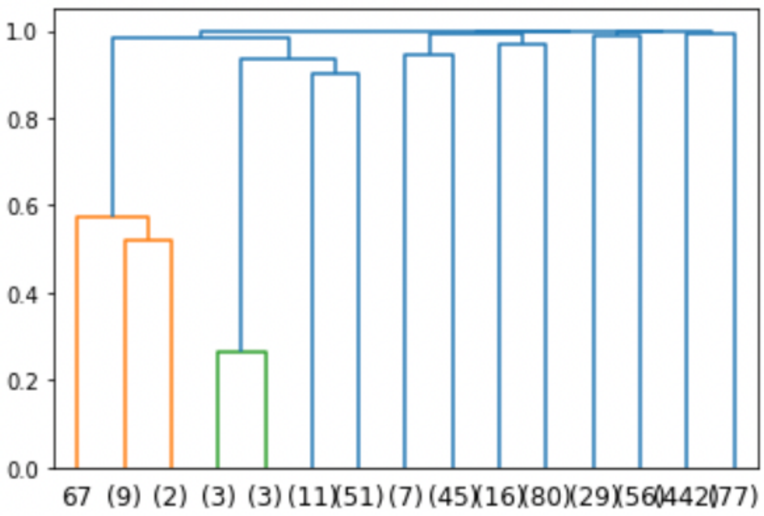


Spectral layout:

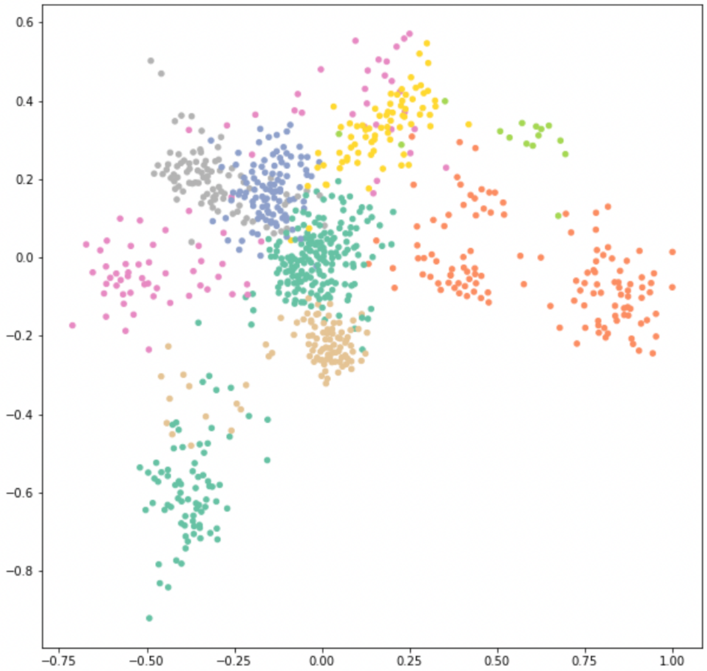


3.3b:
Sklearn reported a rand_score of 0.78 (see code). However, the cluster assignments do seem somewhat reasonable in the spring layout.

3.4.a:
Dendrogram:



Spring layout:



3.4.b:

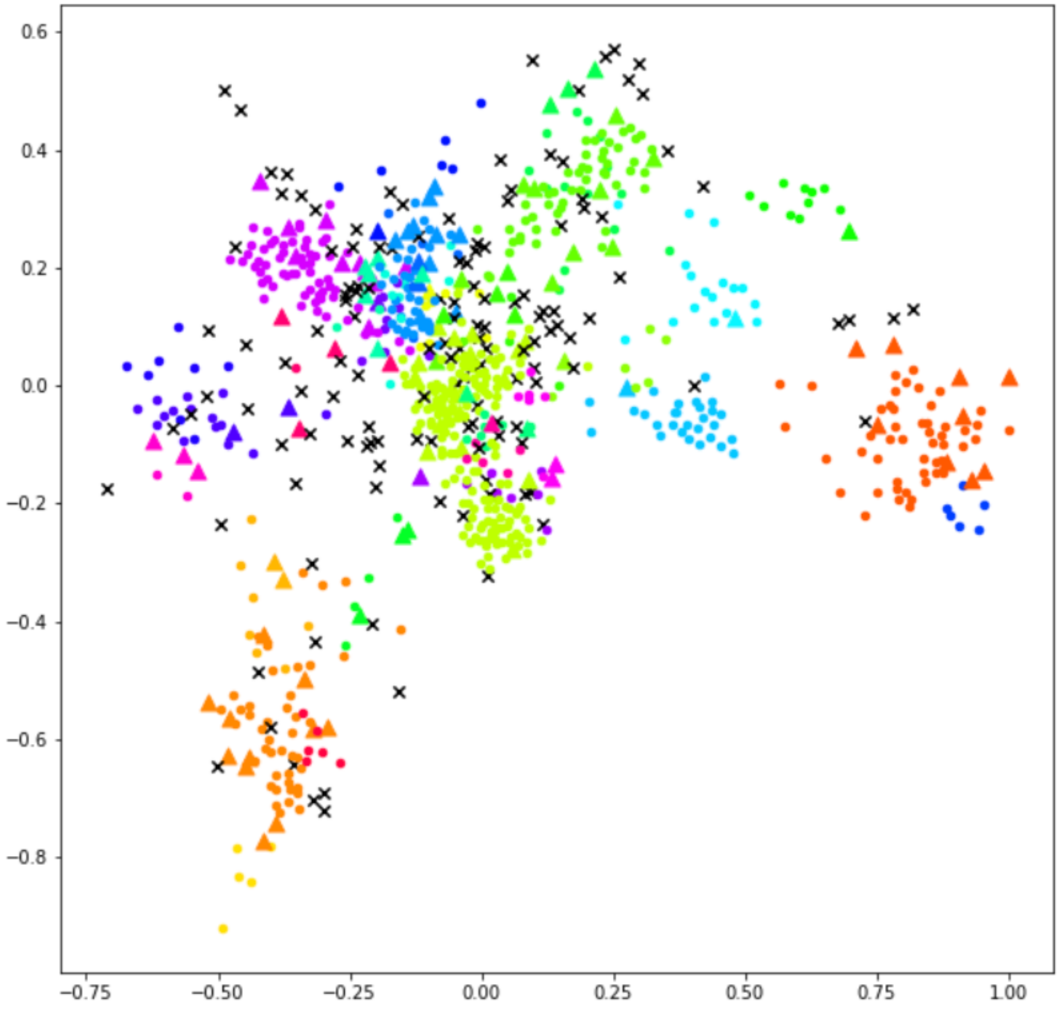
In the dendrogram, we can locate the largest vertical distance between nodes and bisect it with a line. The number of vertical lines that the bisecting line passes can be used as the number of clusters.

Using this heuristic, we achieve a rand score of 0.83.

3.5.a:

Using DBSCAN and marking the noise points with "x":

3.5.b:
We achieved a rand score of 0.836. Plotting the border points:



(Border points marked as triangles, core points as dots, and noisy samples as crosses)

We see that the plot is somewhat consistent with the name “border point”, as border points do tend to appear on the edges of cluster regions.

Problem 4: Semidefinite programming

4i.

We can let x denote the clustering assignments, with $x_i \in \{-1,1\}$. Then, the maximum cut problem corresponds to:

$$\max_x x^T L x$$

$$\text{s.t.}\, x_i^2 = 1 \;\; \forall i$$

We will show that this reduces to the form of the SDP given in the problem. We note that $x^T L x = x^T D x - x^T W x$, but since D is diagonal with positive entries, $x^T D x$ is a constant.

Hence, the problem amounts to:

$$\max_x -x^T W x$$
$$\text{s.t.}\, x_i^2 = 1 \;\; \forall i$$

We then note that:

$$-x^T W x = -\sum_{l,m}^{n} x_l W_{lm} x_m = -\sum_{l,m}^{n} W_{lm} X_{ml} = \mathrm{Tr}(-WX)$$

Where we let $X = xx^T$. Note that since $x_i^2 = 1 \;\; \forall i$, this is equivalent to X being PSD, rank 1, and $diag(X) = 1$. So the problem becomes:

$$\max_X \mathrm{Tr}(-WX)$$
$$\text{s.t.}\, X \succcurlyeq 0, \mathrm{diag}(X) = 1, \mathrm{rank}(X) = 1$$

Considering the first row of X, we see that $X_{0,j} = (xx^T)_{0,j} = 1$ if node 0 and node j are assigned to same cluster and -1 if node 0 and node j are in different clusters. Hence, we only need to look at the first row of X to partition the graph.

4ii.

Let v_i and v_j be two columns of V. Since we use the signs of $V^T h$ to determine the partition, consider $y_i = \mathrm{sign}(v_i^T h)$ and $y_j = \mathrm{sign}(v_j^T h)$. $y_i = y_j$ if and only if v_i and v_j are on the same side of the hyperplane induced by the normal vector h. It is trivial to see that this happens with probability $1 - \frac{\theta_{ij}}{\pi}$, and with probability $\frac{\theta_{ij}}{\pi}$ they are on opposite sides (where θ_{ij} is the angle between v_i and v_j).

Note also that:

$$cut(V_1, V_2) = \sum_{i \in V_1, j \in V_2} w_{ij} = \frac{1}{2} \sum_{(i,j) \in E} w_{ij}(1 - y_i y_j)$$

So:

$$\mathbb{E}[cut(V_1, V_2)] = \frac{1}{2} \sum_{(i,j) \in E} w_{ij}\left(1 - \mathbb{E}[y_i y_j]\right)$$
$$\mathbb{E}[y_i y_j] = (-1)\frac{\theta_{ij}}{\pi} + (+1)\left(1 - \frac{\theta_{ij}}{\pi}\right) = 1 - \frac{2\theta_{ij}}{\pi}$$

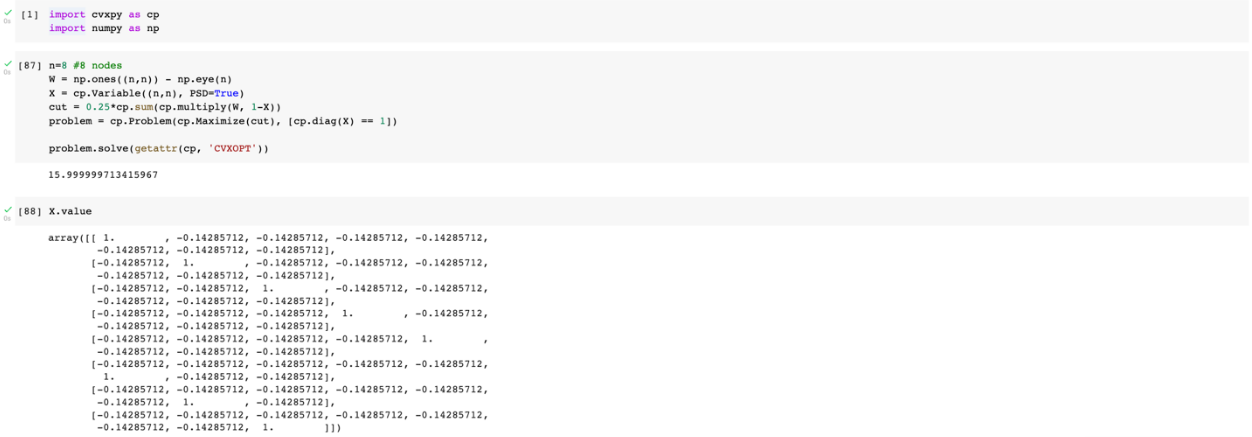
So:

$$\mathbb{E}[cut(V_1, V_2)] = \sum_{(i,j)\in E} \frac{\theta}{\pi} w_{ij} \geq \frac{7}{8}\left(\frac{1}{2}\sum_{(i,j)\in E} w_{ij}\left(1 - \cos\theta_{ij}\right)\right) \geq \frac{7}{8}\left(\frac{1}{2}\sum_{(i,j)\in E} w_{ij}\left(1 - \cos\theta_{ij}^*\right)\right)$$
$$= \frac{7}{8} OPT$$

Where the last inequality is due to relaxing the rank constraint and θ_{ij}^* is the angle we would induce if we did $X = V^T V$ in the rank-constrained case.

4iii.
8 vertices, edge weights all 1.

a. Optimal cut value: 16. Consider a partition of the graphs into two parts $|A| = k$ and $|\bar{A}| = n - k$. For every node in A, there are $n - k$ edges going across the cut to nodes in $\bar{A}$. Hence, there are $k(n - k)$ edges crossing the cut. This is maximized when $n - 2k = 0$ (just taking the derivative and setting to zero) i.e. $k = \frac{n}{2}$. So we get a max cut of $\frac{n}{2}\left(n - \frac{n}{2}\right) = \frac{n^2}{4}$. Here we have $n = 8$, so $max\_cut = 16$.



```
[1] import cvxpy as cp
    import numpy as np
```

```
[87] n=8 #8 nodes
     W = np.ones((n,n)) - np.eye(n)
     X = cp.Variable((n,n), PSD=True)
     cut = 0.25*cp.sum(cp.multiply(W, 1-X))
     problem = cp.Problem(cp.Maximize(cut), [cp.diag(X) == 1])

     problem.solve(getattr(cp, 'CVXOPT'))

     15.999999713415967
```

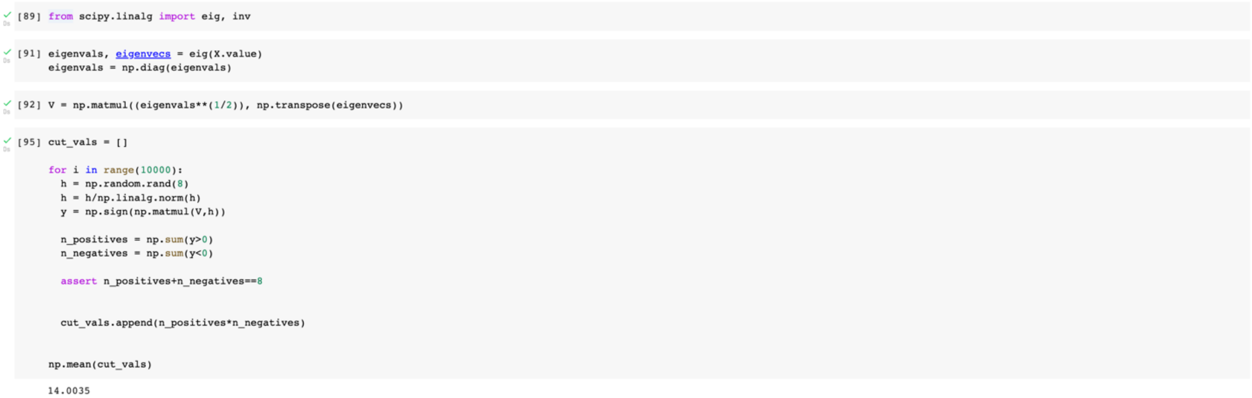
```
[88] X.value

     array([[ 1.        , -0.14285712, -0.14285712, -0.14285712, -0.14285712,
             -0.14285712, -0.14285712, -0.14285712],
            [-0.14285712,  1.        , -0.14285712, -0.14285712, -0.14285712,
             -0.14285712, -0.14285712, -0.14285712],
            [-0.14285712, -0.14285712,  1.        , -0.14285712, -0.14285712,
             -0.14285712, -0.14285712, -0.14285712],
            [-0.14285712, -0.14285712, -0.14285712,  1.        , -0.14285712,
             -0.14285712, -0.14285712, -0.14285712],
            [-0.14285712, -0.14285712, -0.14285712, -0.14285712,  1.        ,
             -0.14285712, -0.14285712, -0.14285712],
            [-0.14285712, -0.14285712, -0.14285712, -0.14285712, -0.14285712,
              1.        , -0.14285712, -0.14285712],
            [-0.14285712, -0.14285712, -0.14285712, -0.14285712, -0.14285712,
             -0.14285712,  1.        , -0.14285712],
            [-0.14285712, -0.14285712, -0.14285712, -0.14285712, -0.14285712,
             -0.14285712, -0.14285712,  1.        ]])
```

b. Optimal X: The code returns:

$$X_{ij}^* = \begin{cases} -0.14285712 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$$

c. Expected size of cut: The code returns 14.0035



```
[89] from scipy.linalg import eig, inv
```

```
[91] eigenvals, eigenvecs = eig(X.value)
     eigenvals = np.diag(eigenvals)
```

```
[92] V = np.matmul((eigenvals**(1/2)), np.transpose(eigenvecs))
```

```
[95] cut_vals = []

     for i in range(10000):
       h = np.random.rand(8)
       h = h/np.linalg.norm(h)
       y = np.sign(np.matmul(V,h))

       n_positives = np.sum(y>0)
       n_negatives = np.sum(y<0)

       assert n_positives+n_negatives==8


       cut_vals.append(n_positives*n_negatives)


     np.mean(cut_vals)

     14.0035
```

Problem 5: Of Vectors and Random Matrices

5.i:
From HW1, we showed that, for a unit vector w:

$$P_G[\|Gw\|^2 > (1+\epsilon)^2] < e^{-cd\epsilon^2}$$

Now consider a non-unit vector $v = x_i - x_j$. We know that $\frac{v}{\|v\|}$ is a unit vector so:

$$P_G\left[\left\|G\frac{v}{\|v\|}\right\|^2 > (1+\epsilon)^2\right] < e^{-cd\epsilon^2}$$

Hence:

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

$$P_G\left[\left\|Gx_i - Gx_j\right\|^2 > (1+\epsilon)^2\left\|x_i - x_j\right\|^2\right] < e^{-cd\epsilon^2}$$

5.ii:

$$P_G\left[\exists \text{ distinct pair } x_i \text{ and } x_j \text{ in } X \text{ s.t.} \left\|Gx_i - Gx_j\right\|^2 > (1+\epsilon)^2\left\|x_i - x_j\right\|^2\right]$$

Note that "∃ distinct pair x_i and x_j in X (s.t. $\left\|Gx_i - Gx_j\right\|^2 > (1+\epsilon)^2\left\|x_i - x_j\right\|^2$ is satisfied)" is equivalent to "(x_1, x_2) is a distinct pair OR (x_1, x_3) is a distinct pair OR … OR (x_1, x_n) is a distinct pair OR (x_2, x_3) is distinct OR … OR (x_2, x_n) is distinct OR … (x_{n-1}, x_n) is distinct". Note that we need to avoid double counting, so the total possible distinct pairs amounts to:

$$(n-1) + (n-2) + (n-3) + \cdots + 1 = \frac{n^2}{2}$$

Hence, invoking the union bound gives us:

$$P_G\left[\exists \text{ distinct pair } x_i \text{ and } x_j \text{ in } X \text{ s.t.} \left\|Gx_i - Gx_j\right\|^2 > (1+\epsilon)^2\left\|x_i - x_j\right\|^2\right] < \frac{n^2}{2}e^{-cd\epsilon^2}$$
$$< n^2e^{-cd\epsilon^2}$$

5.iii:

We want:

$$P_G\left[(1-\epsilon)^2\left\|x_i - x_j\right\|^2 \le \left\|Gx_i - Gx_j\right\|^2 \le (1+\epsilon)^2\left\|x_i - x_j\right\|^2\right]$$
$$= 1 - P_G\left[\left\|Gx_i - Gx_j\right\|^2 > (1+\epsilon)^2\left\|x_i - x_j\right\|^2\right]$$
$$- P_G\left[\left\|Gx_i - Gx_j\right\|^2 < (1-\epsilon)^2\left\|x_i - x_j\right\|^2\right] > 1 - 2n^2e^{-cd\epsilon^2} \ge \frac{3}{4}$$

In other words:

$$n^2e^{-cd\epsilon^2} \le \frac{1}{8}$$
$$-cd\epsilon^2 \le \ln\frac{1}{8n^2}$$
$$d \ge \frac{\ln 8n^2}{c\epsilon^2}$$

So $d_{min} = \frac{\ln 8n^2}{c\epsilon^2}$