Machine Learning for Graphs and Sequential Data Exercise Sheet 11

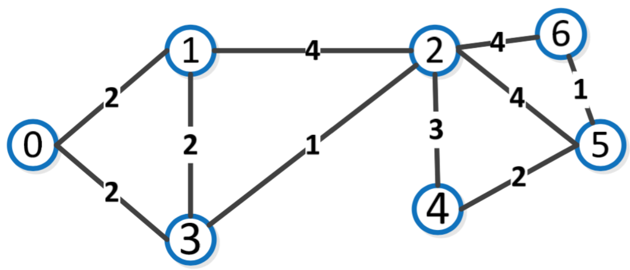
Graphs: Clustering

Problem 1: Given the graph below, find the following partitionings of the graph for $k = 2$:

a) The partitioning giving the global minimum cut

b) A partitioning approximately minimizing the ratio cut

c) A partitioning approximately minimizing the normalized cut




```
import itertools

import numpy as np
from scipy.linalg import eigh
from scipy.sparse import coo_matrix

def laplacian(A):
    D = A.sum(axis=0)
    return np.identity(A.shape[0]) * D - A

def format_partitioning(f):
    c1 = (f > 0.0).nonzero()[0]
    c2 = (f < 0.0).nonzero()[0]
    return f"{{{', '.join(map(str, c1))}}} {{{', '.join(map(str, c2))}}}"

edges = np.array([
    (0, 1, 2), (0, 3, 2), (1, 2, 4), (1, 3, 2), (2, 3, 1),
    (2, 4, 3), (2, 5, 4), (2, 6, 4), (4, 5, 2), (5, 6, 1)]).T
A = coo_matrix((edges[2], (edges[0], edges[1])), shape=(7, 7))
A = A.toarray()
A = A + A.T
L = laplacian(A)

# Global minimum cut
```

```python
fs = [((f := np.array(f_)) @ L @ f / 4, f)
      for f_ in itertools.product([-1, 1], repeat=A.shape[0])
      if len(set(f_)) > 1]
min_cost, min_cut = min(fs, key=lambda f: f[0])
print(f"Global minimum cut is {format_partitioning(min_cut)} at cost {min_cost}")

# Approximate ratio cut
lambda_, v = eigh(L, eigvals=(1, 1))
print(f"Approximate ratio cut is {format_partitioning(v.squeeze())}")

# Approximate normalized cut
D_sqrt_inv = np.diag(1 / np.sqrt(A.sum(axis=1)))
L_normalized = D_sqrt_inv @ L @ D_sqrt_inv
lambda_, v = eigh(L_normalized, eigvals=(1, 1))
print(f"Approximate normalized cut is {format_partitioning(v.squeeze())}")
```

The above solution outputs

```
Global minimum cut is {1, 2, 3, 4, 5, 6} {0} at cost 4.0
Approximate ratio cut is {0, 1, 3} {2, 4, 5, 6}
Approximate normalized cut is {2, 4, 5, 6} {0, 1, 3}
```

Problem 2: Consider minizing the ratio cut on a graph with two clusters C_1 and C_2 and N nodes in total. The indicator vector

$$f_{C_1,i} = \begin{cases} +\sqrt{\frac{|\overline{C_1}|}{|C_1|}} & \text{if } v_i \in C_1 \\ -\sqrt{\frac{|C_1|}{|\overline{C_1}|}} & \text{otherwise} \end{cases}$$

is defined as in the lecture. Prove the following three properties about $\boldsymbol{f}_{C_1}$.

a) $1^T \boldsymbol{f}_{C_1} = \sum_i f_{C_1,i} = 0$

b) $\boldsymbol{f}_{C_1}^T \boldsymbol{f}_{C_1} = \|\boldsymbol{f}_{C_1}\|_2^2 = |V|$

c) $\boldsymbol{f}_{C_1}^T L \boldsymbol{f}_{C_1} = |V| \left[\frac{\text{cut}(C_1,C_2)}{|C_1|} + \frac{\text{cut}(C_1,C_2)}{|\overline{C_1}|}\right]$

Let $M = |C_1|$.

a)

$$\begin{aligned}
1^T \boldsymbol{f}_{C_1} &= \sum_i f_{C_1,i} \\
&= M\sqrt{\frac{|\overline{C_1}|}{|C_1|}} - (N-M)\sqrt{\frac{|C_1|}{|\overline{C_1}|}} \\
&= M\sqrt{\frac{N-M}{M}} - (N-M)\sqrt{\frac{M}{N-M}} \\
&= \sqrt{M(N-M)} - \sqrt{(N-M)M} = 0
\end{aligned}$$

b)

$$\begin{aligned}
\|\boldsymbol{f}_{C_1}\|_2^2 = \boldsymbol{f}_{C_1}^T \boldsymbol{f}_{C_1} &= \sum_i f_{C_1,i}^2 \\
&= \sum_i \begin{cases} \frac{|\overline{C_1}|}{|C_1|} & \text{if } v_i \in C_1 \\ \frac{|C_1|}{|\overline{C_1}|} & \text{otherwise} \end{cases} \\
&= M \cdot \frac{N-M}{M} + (N-M) \cdot \frac{M}{N-M} = N = |V|
\end{aligned}$$

c)

$$\begin{aligned}
\boldsymbol{f}_{C_1}^T L \boldsymbol{f}_{C_1} &= \frac{1}{2} \sum_{(u,v)\in E} W_{uv} \left(f_{C_1,u} - f_{C_1,v}\right)^2 \qquad \text{(Observation 1)} \\
&= \frac{1}{2} \Bigg(\sum_{\substack{(u,v)\in E \\ u,v\in C_1}} W_{uv} \left(f_{C_1,u} - f_{C_1,v}\right)^2 + \sum_{\substack{(u,v)\in E \\ u,v\in C_2}} W_{uv} \left(f_{C_1,u} - f_{C_1,v}\right)^2 \\
&\qquad + 2 \sum_{\substack{(u,v)\in E \\ u\in C_1, v\in C_2}} W_{uv} \left(f_{C_1,u} - f_{C_1,v}\right)^2 \Bigg)
\end{aligned}$$

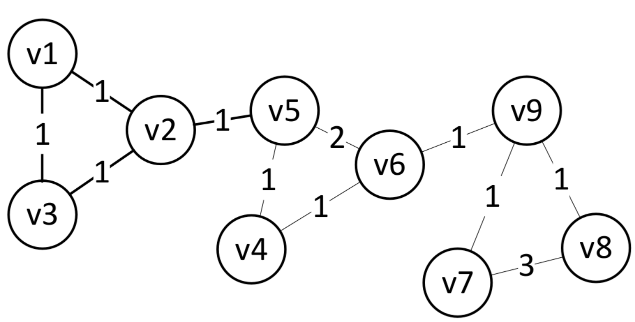
The first two terms are 0 because $f_{C_1,u} = f_{C_1,v}$ if u and v are in the same cluster.

$$
\begin{aligned}
&= \sum_{\substack{(u,v)\in E \\ u\in C_1, v\in C_2}} W_{uv} \left(f_{C_1,u} - f_{C_1,v}\right)^2 \\
&= \sum_{\substack{(u,v)\in E \\ u\in C_1, v\in C_2}} W_{uv} \left(f_{C_1,u}^2 - 2 f_{C_1,u} f_{C_1,v} + f_{C_1,v}^2\right) \\
&= \sum_{\substack{(u,v)\in E \\ u\in C_1, v\in C_2}} W_{uv} \left(\frac{N-M}{M} + 2\sqrt{\frac{(N-M)M}{M(N-M)}} + \frac{M}{N-M}\right) \\
&= \sum_{\substack{(u,v)\in E \\ u\in C_1, v\in C_2}} W_{uv} \left(\frac{N-M}{M} + 2 + \frac{M}{N-M}\right) \\
&= \left(\frac{N-M}{M} + 2 + \frac{M}{N-M}\right) \sum_{\substack{(u,v)\in E \\ u\in C_1, v\in C_2}} W_{uv}
\end{aligned}
$$

The sum of all edge weights of edges with one end in C_1 and the other in C_2 is exactly the size of the cut

$$
\begin{aligned}
&= \left(\frac{N-M}{M} + 2 + \frac{M}{N-M}\right) \operatorname{cut}(C_1, C_2) \\
&= \left(\frac{N-M}{M} + \frac{M}{M} + \frac{N-M}{N-M} + \frac{M}{N-M}\right) \operatorname{cut}(C_1, C_2) \\
&= \left(\frac{N}{M} + \frac{N}{N-M}\right) \operatorname{cut}(C_1, C_2) \\
&= |V| \left[\frac{\operatorname{cut}(C_1, C_2)}{|C_1|} + \frac{\operatorname{cut}(C_1, C_2)}{|C_2|}\right]
\end{aligned}
$$

Problem 3: Answer the following questions regarding the graph below. Formulate a conjecture first and then verify it computationally in a notebook.



a) How does the first eigenvector change when increasing the weight between node $v6$ and $v9$?

b) How does the spectral embedding change?

c) How does this change affect the final clustering?

The first eigenvector will not change because it is always the all-ones vectors scaled to have length 1. The embeddings of node 6 and 9 will move closer together because their connection becomes stronger, other nodes will adapt accordingly. Depending on the increase, node 9 might change into the same cluster as node 6.

We verify our hypotheses with the following program which produces Figure 1.

```
import matplotlib as mpl
mpl.use("Agg")
import matplotlib.pyplot as pp

import numpy as np
from scipy.linalg import eigh
from scipy.sparse import coo_matrix
from sklearn.cluster import KMeans
import seaborn as sns

sns.set()

def plot_cluster(ebds, epsilon=0.05):
    # Rotate embeddings such that node 4 is always embedded straight down
    four = ebds[3]
    alpha = np.arccos(-four[1] / np.linalg.norm(four))
    c, s = np.cos(alpha), np.sin(alpha)
    R = np.array([[c, -s], [s, c]])
    ebds = ebds @ R

    clusters = KMeans(n_clusters=3).fit_predict(ebds)

    fig, ax = pp.subplots(1, 1, figsize=(6, 6))
    xptp, yptp = ebds.ptp(axis=0) * 0.1
    ax.set_xlim((ebds[:, 0].min() - xptp, ebds[:, 0].max() + xptp))
    ax.set_ylim((ebds[:, 1].min() - yptp, ebds[:, 1].max() + yptp))

    # Disturb points to show nodes that get mapped to the same coordinates
    points = ebds + np.random.rand(*ebds.shape) * epsilon

    colors = ["firebrick", "seagreen", "dodgerblue"]
    bbox_props = dict(boxstyle="circle", alpha=0.5, ec="b", lw=2)
    for i, xyc in enumerate(zip(points, clusters)):
        xy, c = xyc
        bbox_props["fc"] = colors[c]
```

```
        pp.text(xy[0], xy[1], str(i + 1), bbox=bbox_props,
                horizontalalignment="center", verticalalignment="center")

    return fig, ax

def laplacian(A):
    D = A.sum(axis=0)
    return np.identity(A.shape[0]) * D - A

edges = np.array([
    (1, 2, 1), (1, 3, 1), (2, 3, 1), (2, 5, 1), (4, 5, 1),
    (4, 6, 1), (5, 6, 2), (6, 9, 1), (7, 8, 3), (7, 9, 1), (8, 9, 1)]).T
edges[:2] -= 1
A = coo_matrix((edges[2], (edges[0], edges[1])), shape=(9, 9)).toarray()
A = A + A.T

_, eig = eigh(laplacian(A), eigvals=(1, 2))
fig, _ = plot_cluster(eig)
fig.savefig("problem_3_pre.png")

A[5, 8] = 4
A[8, 5] = 4
_, eig = eigh(laplacian(A), eigvals=(1, 2))

fig, _ = plot_cluster(eig)
fig.savefig("problem_3_post.png")
```

Problem 4: Consider a PPM with intra-community edge probability p and inter-community edge probability q and a community assignment vector $\boldsymbol{z} \in \{1, -1\}^N$. Assume that the communities are balanced, i.e. $\sum_i z_i = 0$ and both communities have $\frac{N}{2}$ members. Show that the likelihood of an observed adjacency matrix $\boldsymbol{A} \in \{0, 1\}^{N \times N}$ is

$$\mathrm{p}(\boldsymbol{A} \mid \boldsymbol{z}) \propto \left(\frac{(1-p)q}{(1-q)p}\right)^{\mathrm{cut}(\boldsymbol{A};\boldsymbol{z})}$$

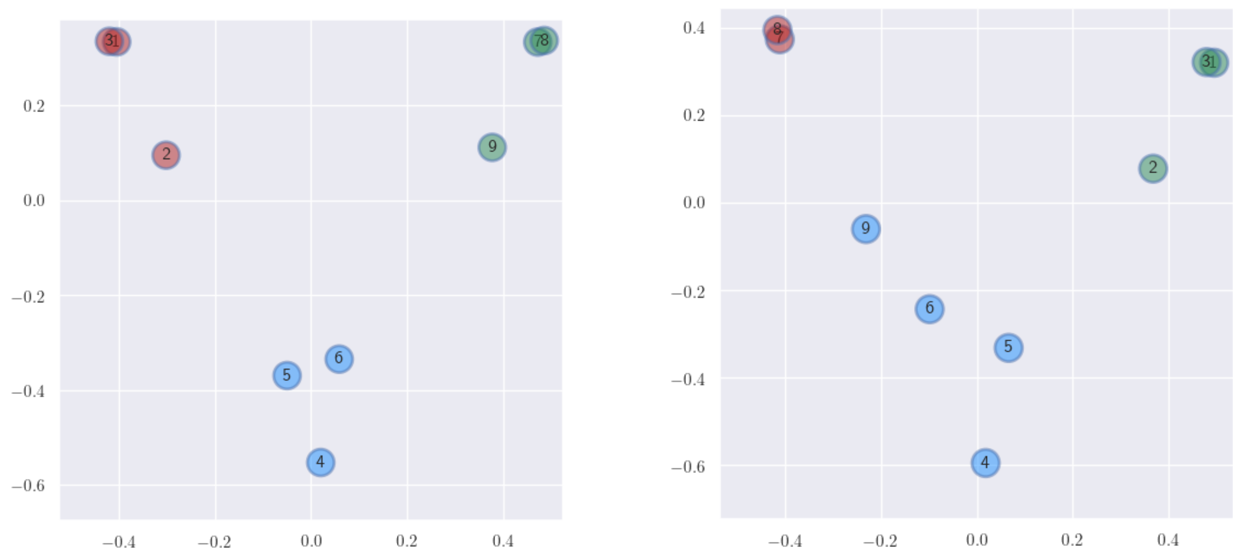
where

$$\mathrm{cut}(\boldsymbol{A};\boldsymbol{z}) = \sum_{i<j} A_{ij} \cdot \mathbb{I}(z_i \neq z_j)$$

is the size of the cut in graph structure $\boldsymbol{A}$. Note that $\boldsymbol{A}$ is observed and thus fixed and we are interested in $\mathrm{p}(\boldsymbol{A} \mid \boldsymbol{z})$ as a function of $\boldsymbol{z}$.

First, define the following symbols

- Number of edges $E = \sum_{i<j} A_{ij}$
- Number of inter-community edges $E_\times = \sum_{i<j} \mathbb{I}(z_i \neq z_j) A_{ij} = \mathrm{cut}(A; z)$



(a) Original weights

(b) After increasing weight between nodes 6 and 9

Figure 1: Spectral embeddings in problem 3

- Number of intra-community edges $E_\circ = \sum_{i<j} \mathbb{I}(z_i = z_j)A_{ij} = E - E_\times = E - \text{cut}(A; z)$

- Number of missing inter-community edges $\overline{E}_\times = \sum_{i<j} \mathbb{I}(z_i \neq z_j)(1 - A_{ij}) = \overline{c}_\times - E_\times = \overline{c}_\times - \text{cut}(A; z)$ where $\overline{c}_\times = \frac{N}{2}\frac{N}{2}$ is the possible number of inter-community edges

- Number of missing intra-community edges $\overline{E}_\circ = \sum_{i<j} \mathbb{I}(z_i = z_j)(1 - A_{ij}) = \overline{c}_\circ - E_\circ = \overline{c}_\circ - E + \text{cut}(A; z)$ where $\overline{c}_\circ = 2\binom{\frac{N}{2}}{2}$ is the possible number of intra-community edges

We will first simplfy the exponents in log-space.

$$
\begin{aligned}
\log \mathrm{p}(\boldsymbol{A} \mid \boldsymbol{z}) &= \log \prod_{i<j} \left(p^{A_{ij}}(1-p)^{1-A_{ij}}\right)^{\mathbb{I}(z_i = z_j)} \left(q^{A_{ij}}(1-q)^{1-A_{ij}}\right)^{\mathbb{I}(z_i \neq z_j)} \\
&= \sum_{i<j} \mathbb{I}(z_i = z_j) \log \left(p^{A_{ij}}(1-p)^{1-A_{ij}}\right) + \mathbb{I}(z_i \neq z_j) \log \left(q^{A_{ij}}(1-q)^{1-A_{ij}}\right) \\
&= \sum_{i<j} \mathbb{I}(z_i = z_j) \left(A_{ij} \log p + (1-A_{ij}) \log(1-p)\right) + \mathbb{I}(z_i \neq z_j) \left(A_{ij} \log q + (1-A_{ij}) \log(1-q)\right) \\
&= \left(\sum_{i<j} \mathbb{I}(z_i = z_j) A_{ij}\right) \log p + \left(\sum_{i<j} \mathbb{I}(z_i = z_j)(1-A_{ij})\right) \log(1-p) \\
&\quad + \left(\sum_{i<j} \mathbb{I}(z_i \neq z_j) A_{ij}\right) \log q + \left(\sum_{i<j} \mathbb{I}(z_i \neq z_j)(1-A_{ij})\right) \log(1-q) \\
&= E_\circ \log p + \overline{E}_\circ \log(1-p) + E_\times \log q + \overline{E}_\times \log(1-q) \\
&= -E_\times \log p + E_\times \log(1-p) + E_\times \log q - E_\times \log(1-q) + c
\end{aligned}
$$

When we return from log-space, we get

$$
\mathrm{p}(\boldsymbol{A} \mid \boldsymbol{z}) \propto \left(\frac{(1-p)q}{(1-q)p}\right)^{E_\times} = \left(\frac{(1-p)q}{(1-q)p}\right)^{\mathrm{cut}(\boldsymbol{A};\boldsymbol{z})}.
$$

Regarding the base of the exponent, we have the relationships $p < q \Rightarrow (1-p) > (1-q) \Rightarrow \frac{(1-p)q}{(1-q)p} > 1$ and $p > q \Rightarrow (1-p) < (1-q) \Rightarrow \frac{(1-p)q}{(1-q)p} < 1$. This means that maximum likelihood inference of $\boldsymbol{z}$ is equivalent to min-cut if edges are more likely within a community than between them and equivalent to max-cut otherwise.