Graph clustering (or community detection or graph partitioning) is one of the most studied problems in network analysis. One reason for this is that there are a variety of ways to define a "cluster" or "community". The goal of this worksheet is to cover some common clustering techniques and explain some of the mathematics behind them. Most of this handout is focused on spectral graph theory to provide technical details not covered in class and to help you with parts of the final homework. This handout only covers a small fraction of graph clustering techniques. For a more comprehensive review, see some of the survey papers on the topic [3, 4, 6].

# 1 Matrix notation and preliminaries from spectral graph theory

Spectral graph theory studies properties of the eigenvalues and eigenvectors of matrices associated with a graph. In this handout, our graph $G = (V, E)$ will be weighted and undirected. Let $n = |V|$, $m = |E|$, and denote the weight of edge $(i, j) \in E$ by $w_{ij} > 0$ with the understanding that $w_{ij} = 0$ for $(i, j) \notin E$. There are a few important matrices that we will use in this handout:

- The weighted adjacency matrix $W$ of a graph is given by $W_{ij} = w_{ij}$ if $(i, j) \in E$ or $W_{ij} = 0$ otherwise.
- The diagonal degree matrix $D$ has the (weighted) degree of node $i$ as the $i$th diagonal entry: $D_{ii} = \sum_j w_{ij}$.
- The *Laplacian* of the graph is $L = D - W$.
- The *normalized Laplacian* of the graph is $\tilde{L} = D^{-1/2}LD^{-1/2} = I - D^{-1/2}WD^{-1/2}$, where $D^{-1/2}$ is a diagonal matrix with $(D^{-1/2})_{ii} = (D_{ii})^{-1/2}$.

We will deal with quadratic forms in this paper. For any real matrix $n \times n$ matrix $A$ and any vector $x \in \mathbb{R}^n$, the quadratic form of $A$ and $x$ is $x^T Ax = \sum_{1 \le i,j \le n} A_{ij} x_i x_j$. Here are some useful facts about the quadratic form for $L$:

**Fact 1.** *For any vector $x \in \mathbb{R}^n$, $x^T Lx = \sum_{(i,j)\in E} w_{ij}(x_i - x_j)^2$. Note that we aren't summing over the entire adjacency matrix and only count each edge once (rather than once for $(j, i)$ and again for $(j, i)$.*

**Fact 2.** *The Laplacian $L$ is positive semi-definite, i.e., $x^T Lx \ge 0$ for any $x \in \mathbb{R}^n$.*

*Proof.* This follows immediately from Fact 1 as the $w_{ij}$ are positive. □

**Fact 3.** *$L = \sum_{(i,j)\in E} w_{ij}(e_i - e_j)(e_i - e_j)^T$, where $e_k$ is the vector with a 1 in coordinate $k$ and a 0 everywhere else. Note that each term $w_{ij}(e_i - e_j)(e_i - e_j)^T$ is the Laplacian of a graph containing just a single edge $(i, j)$ with weight $w_{ij}$.*

**Fact 4.** *The vector $e$ of all ones is an eigenvector of $L$ with eigenvalue 0.*

*Proof.* By Fact 3, $Le = \sum_{(i,j)\in E} w_{ij}(e_i - e_j)(e_i - e_j)^T e = \sum_{(i,j)\in E} w_{ij}(e_i - e_j)0 = 0.$ □

By Fact 2, all of the eigenvalues of $L$ are nonnegative, so Fact 4 says that an eigenvector corresponding to the smallest eigenvalue of $L$ is the vector of all ones (with eigenvalue 0). Since $L$ is symmetric, it has a complete eigendecomposition. In general, we will denote the eigenvalues of $L$ by

$$0 = \lambda_1 \leq \lambda_2 \leq \ldots \lambda_{n-1} \leq \lambda_n.$$

It turns out that the zero eigenvalues determine the connected components of the graph:

**Fact 5.** *If $G$ has exactly $k$ connected components, then*

$$0 = \lambda_1 = \lambda_2 \ldots = \lambda_k < \lambda_{k+1}$$

*In other words, the first $k$ eigenvalues are 0, and the $(k+1)$st eigenvalue is positive.*

## 2 Fiedler vector

The *Fiedler vector* is the eigenvector corresponding to the second smallest eigenvalue of the graph Laplacian and dates back to Fiedler's work on spectral graph theory in the 1970s [2]. In other words, the Fiedler vector $v$ satisfies $Lv = \lambda_2 v$ (side note: $\lambda_2$ is called the *algebraic connectivity* of the graph $G$). The Fiedler vector may be used for partitioning a graph into two components. Here we present the derivation of Riolo and Newman [5].

Suppose we want to partition $G$ into two well-separated components $S$ and $\bar{S} = V \backslash S$. A natural measure of the "separation" between $S$ and $\bar{S}$ is the sum of the weight of edges that have one endpoint in $S$ and one end point in $\bar{S}$. This is commonly referred to as the *cut*:

$$\text{cut}(S) = \sum_{i\in S, j\in\bar{S}} w_{ij} \tag{1}$$

Note that the cut measure is symmetric in $S$ and $\bar{S}$, i.e., $\text{cut}(S) = \text{cut}(\bar{S})$. We can relate the cut to a quadratic form on $L$ with an assignment vector $x$ on the sets. Specifically, let $x$ be an assignment vector:

$$x_i = \begin{cases} 1 & \text{node } i \in S \\ -1 & \text{node } i \in \bar{S} \end{cases} \tag{2}$$

Then

$$x^T L x = \sum_{(i,j)\in E} w_{ij}(x_i - x_j)^2 = \sum_{(i,j)\in E} w_{ij} 4(1 - I_{x_i = x_j}) = 4 \sum_{i\in S, j\in\bar{S}} w_{ij} = 4 \cdot \text{cut}(S) \tag{3}$$

At first glance, we might just want to find an assignment vector $x$ that minimizes the cut value. If we assign all nodes $i$ to $S$ then we get a cut value of 0, which is clearly the minimum. However, this is not an interesting partition of the graph. We would like to enforce some sort of balance in the partition. One approach is to minimize the cut under the constraint

that $S$ has exactly half the nodes (assuming the graph has an even number of nodes). In this case, we have that

$$\sum_i x_i = \sum_{i \in S} 1 + \sum_{i \in \bar{S}} (-1) = |S| - |\bar{S}| = 0.$$

In matrix notation, we can write this as $x^T e = 0$, where $e$ is the vector of all ones. This leads to the following optimization problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & x^T L x \\ \text{subject to} \quad & x^T e = 0 \\ & x_i \in \{-1, 1\} \end{aligned}$$

Unfortunately, the constraint that the $x_i$ take the value $-1$ or $+1$ makes the optimization NP-hard [7]. Thus, we use a common trick in combinatorial optimization: (i) relax the constraints to a tractable problem and (ii) round the solution from the relaxed problem to a solution in the original problem. In this case, we will relax the constraint that $x_i \in \{-1, 1\}$ to the constraint $x \in \mathbb{R}$ with $x^T x = n$. Note that the latter constraint is always satisfied in our original optimization problem—we use it here to get a bound on the size of $x$ in the relaxed problem. Our new relaxed optimization problem is:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & x^T L x \\ \text{subject to} \quad & x^T e = 0 \\ & x^T x = n \end{aligned} \tag{4}$$

It turns out that the Fiedler vector solves this optimization problem:

**Theorem 6.** *Let $G$ be connected. The minimizer of the optimization problem in Equation 4 is the Fiedler vector.*

*Proof.* Since $L$ is symmetric, there is an orthonormal basis for $\mathbb{R}^n$ consisting of eigenvectors of $L$. Thus, we can write any vector $x \in \mathbb{R}^n$ as

$$x = \sum_{i=1}^{n} w_i v_i,$$

where the $w_i$ are weights and $Lv_i = \lambda_i v_i$. Furthermore, since $G$ is connected, there is a single basis vector that spans the eigenspace corresponding to eigenvalue 0. By Fact 4, this vector is $v_1 = e/\|e\|_2 = \frac{1}{\sqrt{n}} e$, where $e$ is the vector of all ones. Since $x^T e = 0$, we must have that $w_1 = 0$ for any feasible solution, i.e., $x = \sum_{i=2}^{n} w_i v_i$. It is easy to show that

$$x^T x = \sum_{i=2}^{n} w_i^2$$

and

$$x^T L x = \sum_{i=2}^{n} w_i^2 \lambda_i$$

Thus, the optimization problem becomes

$$\begin{aligned} \underset{w_2,\ldots,w_n}{\text{minimize}} \quad & \sum_{i=2}^{n} w_i^2 \lambda_i \\ \text{subject to} \quad & \sum_{i=2}^{n} w_i^2 = n \end{aligned}$$

Clearly, we should put all of the "mass" on $\lambda_2$, the smallest of the eigenvalues that are non-zero. Thus, the minimizer has the weights $w_2 = \sqrt{n}$, $w_3 = w_4 = \ldots w_n = 0$. □

The above theorem shows how to solve the "relaxed" problem, but we still have to round the solution vector $\frac{1}{\sqrt{n}} v_2$ to a partition of the graph. There are a couple ways we might do this. We could just assign the nodes corresponding to the positive entries of the eigenvector to $S$ and the nodes corresponding to the negative entries to $\bar{S}$. Alternatively, we could run $k$-means clustering (with $k = 2$) on the $n$ real-valued points given by the eigenvector.

# 3 Modularity

Again let $S_1, \ldots, S_k$ be $k$ disjoint clusters that cover $V$. Let $c_i$ be the cluster to which node $i$ belongs. The *modularity* of the clusters is

$$Q(S_1, \ldots, S_k) = \frac{1}{2m} \sum_{1 \leq i,j \leq n} \left[ W_{ij} - \frac{d_i d_j}{2m} \right] I_{c_i = c_j}. \tag{5}$$

Here, each term $\frac{d_i d_j}{2m}$ is approximately the expected number of edges between $i$ and $j$ in a random multi-graph generated with the same degree sequence as the graph $G$. Thus, $W_{ij} - \frac{d_i d_j}{2m}$ measures how "surprising" the link is between nodes $i$ and $j$.

We want to find a clustering (community assignment) that maximizes modularity. In class and in the homework, we saw a spectral method for maximizing modularity in the special case when $k = 2$. However, the spectral ideas do not generalize to $k > 2$ in the same ways as the ratio cut and normalized cut objectives. A greedy approach to modularity maximization iteratively changes individual node affiliation to maximize modularity. This can be computed efficiently for small $k$ and is the basis for popular procedures such as the Louvain method [1].

# References

[1] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[2] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2):298–305, 1973.

[3] S. Fortunato. Community detection in graphs. *Physics reports*, 486(3):75–174, 2010.

[4] S. Fortunato and D. Hric. Community detection in networks: A user guide. *Physics Reports*, 659:1–44, 2016.

[5] M. A. Riolo and M. Newman. First-principles multiway spectral partitioning of graphs. *Journal of Complex Networks*, 2(2):121–140, 2014.

[6] S. E. Schaeffer. Graph clustering. *Computer science review*, 1(1):27–64, 2007.

[7] D. Wagner and F. Wagner. Between min cut and graph bisection. In *Proceedings of the 18th International Symposium on Mathematical Foundations of Computer Science*, pages 744–750, 1993.