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, 7].

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 ith 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 $x^T Ax = \sum_{1 \le i,j \le n} A_{ij}x_ix_j$. Here are some useful facts about the quadratic form for L:

Fact 1. *For any vector $x \in \mathbb{R}^n$, $x^T Lx = \sum_{(i,j)\in E} w_{ij}(x_i - x_j)^2$.*

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: λ_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 [6].

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

$$x^T L x = \sum_{(i,j) \in E} w_{ij}(x_i - x_j)^2 = \sum_{(i,j) \in E} w_{ij} 4(1 - I_{x_i = x_j}) = 8 \sum_{i \in S, j \in \bar{S}} w_{ij} = 8 \cdot \text{cut}(S) \tag{3}$$

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 [10]. 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 λ_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 Multi-way spectral clustering with Ratio cut

In general, we might want to simultaneously find k clusters of nodes instead of just finding a partition of the graph. To do this, we will try to minimize the ratio cut objective, following the derivation in [9]. Consider k disjoint sets of nodes $S_1, S_2, \ldots, S_k$ such that $\cup_{i=1}^k S_i = V$. The *ratio cut* is

$$\text{RatioCut}(S_1, \ldots, S_k) = \sum_{i=1}^{k} \frac{\text{cut}(S_i)}{|S_i|} \tag{5}$$

Trying to minimize the ratio cut is a sensible approach. We want each cluster S_i to be well separated but not too small; thus, we minimize the ratios of cut to size for each cluster.

Suppose we have an assignment matrix X such that

$$X_{ir} = \begin{cases} \frac{1}{\sqrt{|S_r|}} & \text{node } i \in S_r \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Let x_r be the rth column of X. Then

$$\begin{aligned} x_r^T L x_r &= \sum_{(i,j)\in E} w_{ij}(x_{ir} - x_{jr})^2 \\ &= 2 \cdot \sum_{i \in S_r, j \in \bar{S}_r} w_{ij} \frac{1}{|S_r|} \\ &= 2 \cdot \frac{\text{cut}(S_r)}{|S_r|}. \end{aligned}$$

Recall that the *trace* of a matrix A, denoted $\text{tr}(A)$ is the sum of the diagonal entries of A. We have that

$$\sum_{r=1}^{k} x_r^T L x_r = \text{tr}(X^T L X) \propto \text{RatioCut}(S_1, \ldots, S_k) \tag{7}$$

We claim that our assignment matrix X is orthogonal, i.e., that $X^T X = I$, the identity matrix. Indeed,

$$(X^T X)_{rr} = \sum_{i=1}^{n} x_{ir}^2 = \sum_{i=1}^{n} I_{i \in S_r} \frac{1}{|S_r|} = 1$$

and $x_{ir}x_{ir'}$ for $r' \neq r$ is always 0. Thus, we may write our ratio cut optimization problem as:

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \operatorname{tr}(X^T L X) \\ \text{subject to} \quad & X^T X = I \\ & X \text{ as in Equation 6} \end{aligned}$$

The constraint that X take the form of Equation 6 makes this optimization problem difficult. We again take our "relax and round" approach by removing this constraint. Our relaxation is simply to remove the assignment constraint (while keeping the orthogonality constraint):

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \operatorname{tr}(X^T L X) \\ \text{subject to} \quad & X^T X = I \end{aligned} \tag{8}$$

Theorem 7. *The minimizer of Equation 8 is the matrix V whose columns are the k eigenvectors of L corresponding to the k smallest eigenvalues.*

Proof. The result is a consequence of the Courant-Fischer min-max theorem. □

Thus, our relaxed solution is given by the first k eigenvectors. Let V denote the $n \times k$ matrix formed by these eigenvectors. We still have to round this solution back to a clustering assignment. The standard approach is to consider the ith row V to be an embedding of node i into $\mathbb{R}^k$ and then use some point cloud clustering technique (such as k-means or DBSCAN).

Putting everything together leads to the following algorithm:

1. Compute the k eigenvectors corresponding to the k smallest eigenvalues of L (denoted by the $n \times k$ matrix V).
2. Let $z_i \in \mathbb{R}^k$ be the ith row in V.
3. Run your favorite point cloud clustering algorithm on $\{z_i\}_{i=1}^n$ (e.g., k-means).

4 Volume-based measures: conductance and normalized cut

In the previous two sections, we balanced the cut metric with the number of nodes in the clusters. For the Fiedler vector, the combinatorial optimization problem partitioned the graph into two sets with the same number of nodes and for the ratio cut objective divided the cut by the cardinality of the clusters. Now we consider a couple measures that take into account the *volume* of the clusters, which is the sum of (weighted) degrees of nodes in a cluster. Formally, we denote the volume of a set S by

$$\operatorname{vol}(S) = \sum_{i \in S} \sum_{j \in V} w_{ij}. \tag{9}$$

As we will see below, algorithms for solving our volume-based metrics will use the normalized Laplacian $\tilde{L}$ instead of the Laplacian L. Recall that the $\tilde{L} = D^{-1/2}LD^{-1/2} = I - D^{-1/2}WD^{-1/2}$.

It is important to note that so far, our "relax and round" approaches were purely a heuristic. We have no guarantees about the quality of our rounded solutions, although the methods work well in practice. We will get a guarantee with Cheeger's inequality for the conductance metric discussed below.

4.1 Conductance

The *conductance* of a set S is defined as follows:

$$\phi(S) = \frac{\text{cut}(S)}{\min(\text{vol}(S), \text{vol}(\bar{S}))}$$

The following theorem gives some bounds on the conductance of clusters obtained from the second eigenvector of $\tilde{L}$.

Theorem 8 (Cheeger's inequality). *Let $\phi_G = \min_{S \subset V} \phi(S)$. Then*

$$\tilde{\lambda}_2/2 \le \phi_G \le \sqrt{2\tilde{\lambda}_2} \tag{10}$$

Furthermore, let v be the eigenvector corresponding to the second smallest eigenvalue of $\tilde{L}$ and let $v' = D^{-1/2}v$. Let $Y_t = \{i \mid v'_i \le t\}$. Then there exists a t for which

$$\phi(Y_t) \le 2\sqrt{\phi_G} \tag{11}$$

We will not prove this here. See Dan Spielman's lecture notes for a proof [8].

Note that there are only a finite number of unique sets defined by Y_t. In practice, we can define S_i to be the first i indices of the sorted ordering of v'. Then, we pick the set S_i, $i = 1, \ldots, n$, with smallest conductance to be the cluster. With clever counting, we can find the best set in linear time in the number of edges and nodes in the graph. (This is often referred to as the *sweep* procedure.) Equation 11 says that this set is within a quadratic factor of optimal.

4.2 Normalized cut

With the ratio cut objective, we used set size as the normalizing parameter in the denominator. The normalized cut objective uses set volume instead. Again consider k disjoint sets of nodes $S_1, S_2, \ldots, S_k$ such that $\cup_{i=1}^{k} S_i = V$. The *normalized cut* is

$$\text{NormalizedCut}(S_1, \ldots, S_k) = \sum_{i=1}^{k} \frac{\text{cut}(S_i)}{\text{vol}(S_i)}$$

Using the assignment vector

$$X_{ir} = \begin{cases} \frac{1}{\sqrt{\mathrm{vol}(S_r)}} & \text{node } i \in S_r \\ 0 & \text{otherwise} \end{cases}$$

and a similar relaxation technique as for ratio cut, we end up with the following optimization problem:

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \mathrm{tr}(X^T \tilde{L} X) \\ \text{subject to} \quad & X^T X = I \end{aligned}$$

(You will derive something similar in Homework 4.)

Now, the optimal solution is the first k eigenvectors of $\tilde{L}$, leading to the following algorithm:

1. Compute the k eigenvectors corresponding to the k smallest eigenvalues of $\tilde{L}$ (denoted by the $n \times k$ matrix $\tilde{V}$).
2. Let $\tilde{z}_i \in \mathbb{R}^k$ be the ith row in $\tilde{V}$.
3. Run your favorite point cloud clustering algorithm on $\{\tilde{z}_i\}_{i=1}^n$ (e.g., k-means).

Recent research by Lee et al. proves that a particular rounding approach in Step 3 leads to a clustering of the graph with guarantees like the Cheeger inequality [5].

5 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

$$Q(S_1, \ldots, S_k) = \frac{1}{4m} \sum_{1 \leq i,j \leq n} \left[W_{ij} - \frac{d_i d_j}{2m} \right] I_{c_i = c_j}. \tag{12}$$

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].

6 Probabilistic models overlapping clusters

Thus far, we have considered reasonable objective functions for measuring the quality of clusters. We now explore a different approach where we develop a reasonable model for how clusters form and then learn the model parameters from the network data.

6.1 Affiliation graph model

The *affiliation graph model* (AGM) is specified by the following parameters:

- a set of nodes V
- a set of communities C: $c \subset V$ for each $c \in C$
- a set of memberships M: $M_u \subset C$ for each node u
- a set of probabilities p: p_c is a link probability for each community c

Note that there is no restriction on the structure of C. Communities may be nested, partially overlapping, or disjoint. In the AGM, an edge is formed between nodes u and v with probability

$$P(u, v) = 1 - \prod_{c \in M_u \cap M_v} (1 - p_c). \tag{13}$$

We will take a maximum likelihood approach to finding the model parameters. The likelihood of a graph $G = (V, E)$ under AGM with parameters $\theta = (C, M, p)$ is

$$\ell(G; \theta) = \prod_{(u,v) \in E} P(u, v) \prod_{(u,v) \notin E} (1 - P(u, v)). \tag{14}$$

Given a graph G, we seek to find

$$\arg \max_{\theta = C, M, p} \ell(G; \theta). \tag{15}$$

6.2 BigCLAM

Solving Equation 15 is quite difficult for large graphs. One approach to make the problem tractable is to assume more about the model parameters in order to make learning easier. This is the idea behind BigCLAM [11]. Let F be a $|C| \times n$ nonnegative community affiliation matrix such that

$$F_{cu} = \text{affinity of node } u \text{ for community } v \tag{16}$$

Let f_u be the uth column of F. This vector represents the community affiliations of node u. The BigCLAM model then generates edges with probability

$$P(u, v) = 1 - e^{-f_u^T f_v}. \tag{17}$$

The log-likelihood of a graph under the BigCLAM model is then

$$LL(F) = \sum_{(u,v) \in E} (1 - e^{-f_u^T f_v}) - \sum_{(u,v) \notin E} f_u^T f_v. \tag{18}$$

In order to find the optimal F, we can employ a block coordinate gradient descent. In other words, we fix the rows of F for all but one node u and then optimize f_u. The gradient is

$$\Delta LL(f_u) = \sum_{v \in N(u)} f_v \frac{e^{-f_u^T f_v}}{1 - e^{-f_u^T f_v}} - \sum_{v \notin N(u)} f_v, \tag{19}$$

where $N(u)$ is the neighbor set of u. Naively, the two summations would require $O(|V|)$ work per gradient step. However, we can use the identity

$$\sum_{v \notin E} f_v = \sum_{v} f_v - \sum_{w \in N(u)} f_w. \tag{20}$$

Thus, we only need to keep track of the two terms in the right side of the equality, reducing the complexity to $O(|N(u)|)$. For real-world graphs that tend to be sparse, this is scalable.

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] J. R. Lee, S. O. Gharan, and L. Trevisan. Multiway spectral partitioning and higher-order cheeger inequalities. *Journal of the ACM (JACM)*, 61(6):37, 2014.

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

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

[8] D. Spielman. Conductance, the normalized laplacian, and cheeger's inequality. http://www.cs.yale.edu/homes/spielman/561/lect06-15.pdf, September 2015.

[9] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[10] 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.

[11] J. Yang and J. Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 587–596. ACM, 2013.