2021-04-27

1 From semi-supervised to unsupervised

In the last lecture, we discussed the use of graphs for *semi-supervised* learning tasks in which we are given labels on some nodes and want to find other nodes. In this lecture, we turn to the *unsupervised* task of interpreting the graph structure without the benefit of auxiliary data such as class labels. Specifically, we consider the closely related tasks of graph partitioning, graph clustering, graph embedding, and nonlinear manifold learning.

2 Graph bisection

We begin with a discussion of *spectral graph partitioning*. In the simplest setting, we are given an unweighted graph, and we want to *bisect* the graph: that is, we seek $\mathcal{V} = \mathcal{V}^+ \cup \mathcal{V}^-$ disjoint with $|\mathcal{V}^+| = |\mathcal{V}^-|$ so that there are as few edges as possible connecting $\mathcal{V}^+$ to $\mathcal{V}^-$. There are several reasons we might want to solve this problem:

1. For *data analysis*, we might want assign one of two class labels to every graph node based on a similarity measure encoded through the graph. However, we want to rule out the trivial solution where all nodes are assigned to the same class. In the semi-supervised problem from last class, we avoided this case with the help of labeled training examples. In the unsupervised setting, we use class size constraints toward the same end.

2. As we saw at the end of last lecture, in *nested dissection* ordering methods for solving linear systems, we want to recursively find small *vertex separators* that partition the graph into roughly equal size pieces. Of course, we have posed the problem in terms of finding good *edge separators* for a graph, but some of the same ideas apply to both problems. Also, we can construct vertex separators from edge separators by taking an endpoint for each cut edge.

3. In static *load balancing* problems in parallel computing, we sometimes want to distribute work (represented by nodes) to two processors in an

equitable way while ensuring that the interprocessor communication (represented by cut edges) is as small as possible.

In each of these cases, we may ultimately want more than two pieces, or we may want a more refined notion of the partition quality than the one we have considered. But this is a good starting point nonetheless.

We can encode the basic problem in matrix terms via the graph Laplacian. Let $x = \{\pm 1\}^n$ be an indicator for a partitioning; then

$$\begin{aligned} e^T x &= \sum_i x_i && = |\mathcal{V}^+| - |\mathcal{V}^-|, \\ x^T L x &= \sum_{(i,j)\in\mathcal{E}} (x_i - x_j)^2 && = 4 \times (\text{edges between } \mathcal{V}^+ \text{ and } \mathcal{V}^-), \end{aligned}$$

and the basic graph bisection problem is

$$\text{minimize} \frac{1}{4} x^T L x \text{ over } x \in \{\pm 1\}^n \text{ s.t. } e^T x = 0.$$

This is a classic NP-hard problem, but there is an easy relaxation where we replace the constraint $x \in \{\pm 1\}^n$ with $x \in \mathbb{R}^n$ such that $\|x\|_2^2 = n$. In this case, we have a *quadratically constrained quadratic program* for x; and as we have seen, such optimization problems can often be rewritten in terms of eigenvalue problems. Of course, in addition to the quadratic constraint $\|x\|_2^2 = n$, we have the linear constraint $e^T x = 0$. If we recognize that $Le = 0$, we can see this as the standard optimization formulation for (one quarter) the second-smallest eigenvalue[1] of the graph Laplacian and associated eigenvector x of the graph Laplacian, with a particular normalization for x. This vector is sometimes called the *Fiedler vector*.

The basic strategy for spectral graph partitioning then is:

1. Approximately compute the Fiedler vector x using the Lanczos iteration, which requires only matrix-vector products with L.
2. Partition based on the signs of the elements of x.
3. (Optionally) refine with a local algorithm such as Kernighan-Lin.

Spectral methods are not the only way; methods like multi-level Kernighan-Lin are also popular. But spectral partitioning is frequently used, it works well, and it fits well with the overall narrative of these lectures!

[1] We will assume throughout this lecture that our graphs have a single connected component, so that the zero eigenvalue of L has multiplicity 1.

3 Weights and normalization

So far, we have covered the case of spectral methods for bisection of unweighted graphs. But what if there are node and edge weights? That is, we suppose that we want partition to minimize a weighted edge cut subject to the constraint that a sum of node weights in both halves is zero. Let L be a weighted Laplacian and c a vector of node costs; then we would like to minimize $x^T L x$ subject to $c^T x = 0$ and $x \in \{\pm 1\}^n$. As before, we relax the integer constraint $x \in \{\pm 1\}^n$ to $x \in \mathbb{R}^n$ and $\|x\|^2 = n$. This gives us the problem

$$\text{minimize } x^T L x \text{ s.t. } x^T x = n \text{ and } x \in \mathbb{R}^n \text{ with } c^T x = 0.$$

The space of vectors orthogonal to c is an $n-1$-dimensional linear space; if we choose an orthonormal basis V for this space, we can write $x = Vy$ and obtain

$$\text{minimize } y^T (V^T L V) y \text{ s.t. } y^T y = n \text{ and } y \in \mathbb{R}^{n-1}.$$

The solution to this problem is $(V^T L V)y = \lambda y$ where λ is the smallest eigenvalue of $V^T L V$. We can also solve this problem as

$$(\Pi L \Pi) x = \lambda x.$$

where $\Pi = V^T V = I - cc^T/\|c\|^2$ is the projection onto the space orthogonal to c. In this case, λ is the second-smallest eigenvalue of $\Pi L \Pi$; as before, the smallest eigenvalue is zero. The matrix $\Pi L \Pi$ is not generally sparse, but we can do fast matrix-vector products with it. We can compute this second-smallest eigenvalue using the Lanczos algorithm, as before.

4 Normalized cuts

Another variant on the same idea uses the quadratic form

$$\rho_{A,D}(x) = \frac{x^T A x}{x^T D x}$$

which, given a 0-1 indicator x for a set, tells us what fraction of the link weight adjacent on the set is actually internal to the set. Closely related is the quadratic form

$$\rho_{L,D}(x) = \frac{x^T L x}{x^T D x} = 1 - \rho_{A,D}(x),$$

which tells us the ratio of a cut weight to the weight adjacent on the set. The *normalized cut* for a set is the symmetrized function

$$C(x) = \rho_{L,D}(x) + \rho_{L,D}(e - x).$$

Let $k = x^T d/(e^T d)$ be the fraction of all the weight in the graph that is incident on x; then

$$C(x) = \frac{x^T L x}{k w_{\text{all}}} + \frac{x^T L x}{(1-k) w_{\text{all}}} = \frac{1}{k(1-k)} \frac{x^T L x}{w_{\text{all}}} = \frac{1}{1-k} \left(\frac{x^T L x}{x^T D x} \right)$$

For a fixed target k, therefore, minimizing the cut measure $C(x)$ is equivalent to minimizing the generalized Rayleigh quotient $\rho_{L,D}(x)$. For a fixed k, we introduce

$$y = x + b(e - x), \quad b = \frac{k}{1-k}$$

i.e. an indicator that takes on the value $+1$ on the desired set and $-b$ on the complement; the normalized cut is then also proportional to $\rho_{L,D}(y)$, and the constraint is $d^T y = 0$. Hence, we have the optimization problem

$$\text{minimize } \rho_{L,D}(y) \text{ s.t. } d^T y = 0 \text{ and } y_i \in \{1, -b\};$$

and if we relax the final constraint, we find that we are minimizing $\rho_{L,D}(y)$ over the space $d^T y = e^T D y = 0$. This exactly characterizes the eigenpair of $Ly = \lambda Dy$ associated with the second-smallest eigenvalue; the eigenpair associated with $y = e$ and $\lambda = 0$ is explicitly removed from consideration by the constraint.

5 Modularity maximization

As another example, we consider the problem of finding communities with high "modularity," defined to be the number of internal edges minus the number that we would expect in a "configuration model" that reflects the node degrees but otherwise assigns edges randomly. In matrix terms, if x is a 0-1 indicator for a set, the modularity is $x^T B x$ where

$$B = A - \frac{dd^T}{2m}$$

is the *modularity* matrix defined at the end of Tuesday's notes. Note that $Be = 0$, so that if $y = 2x - e$ is a ± 1 indicator for the same set, then

$$y^T B y = 4x^T B x - 4x^T B e + e^T B e = 4x^T B x.$$

Therefore, we consider maximizing the quadratic form $y^T B y$ over all ± 1 indicators. Applying the usual trick, we relax the constraint that $y \in \{\pm 1\}^n$ to the quadratic constraint $y^T y = n$ for real-valued y to obtain the eigenvalue problem

$$By = \lambda y.$$

Unlike in the previous examples, we have no linear normalization constraints here, and we are simply looking for the largest value and the corresponding eigenvector. The "high modularity" sets indicated by the positive (or negative) elements of y are often used as starting points for community-detection algorithms in social networks.

6 Mixing in random walks

So far, we have focused on spectral methods motivated by optimization, but this is not the only way to arrive at these approach. Eigenvalues and eigenvectors are also good for analyzing dynamics on networks, e.g. when thinking about the convergence of random walks.

A (discrete time) Markov chain is a time-indexed set of random variables $X(t)$ in which the state at time $t+1$ depends only on the state at time t. For a finite state space $[n] = \{1, 2, \ldots, n\}$, we can completely characterize the Markov chain in terms of a *transition matrix* P with entries[2].

$$p_{ij} = \text{Prob}\{X(t+1) = i | X(t) = j\}.$$

Let $\pi(t)$ denote the probability mass function for $X(t)$, represented as a vector, i.e. $\pi_i(t)$ denotes the probability that $X(t) = i$. Then we have the iteration equation

$$\pi(t+1) = P\pi(t) = P^{t+1}\pi(0).$$

[2]We are taking the numerical linear algebra convention that probabilities distributions represent column vectors, and $P_{:,j}$ denotes the probability mass function for $X(t+1)$ given $X(t) = j$. Statisticians often denote probability mass functions by rows, and reverse the roles of i and j

Our usual way of thinking about Markov chains advances a distribution over states from one time to the next. We can also think about the iteration equation in terms of a random variable on the state space, represented by a row vector f^T. The expected value of the random variable at time t is

$$f^T \pi(t) = f^T P^t \pi(0) = \left(f^T P^t\right) \pi(0).$$

That is, the same iteration that advances the distribution forward in time serves to push random variables backward in time.

We often think of discrete time Markov chains over a finite state space in terms or random walks on an associated graph. If A is a weighted adjacency matrix for some directed graph, with a_{ij} denoting the edge weight of a node from j to i, then we define a Markov chain with transition matrix

$$P = AD^{-1}$$

where D is the diagonal matrix of node (out-)degrees. That is, we consider a random walker who, at each step, chooses an outgoing edge to move on with probability proportional to the weight of the edge. A number of properties of the Markov chain are described in terms of the graph:

- We say i *accesses* j if there is a path from i to j through the graph.
- We say i and j *communicate* if i access j and vice-versa.
- We call the Markov chain *irreducible* if all nodes communicate.
- The *period* of node i is the GCD of the length of all closed paths beginning and ending at i. If there are no such paths, we say that the period of i is infinite. We note that if i and j communicate with each other, then they must have the same period.
- The Markov chain is *aperiodic* if all nodes have period one.

Every Markov chain over a finite state space has a *stationary distribution* π^* such that $P\pi^* = \pi^*$ (this is a consequence of the *Perron-Frobenius* theorem). The stationary distribution is unique if there is some state i that can access any other state. If the Markov chain is irreducible, then the stationary probability is not only unique, but is nonzero for all states. An aperiodic Markov chain will converge to *some* stationary distribution from any initial distribution. If a Markov chain is both irreducible and aperiodic, it is *ergodic*, and

from any initial distribution it will eventually converge to a unique stationary distribution supported on all nodes.

A Markov chain is *reversible* if it has a stationary distribution for which it satisfies *detailed balance*, i.e.

$$PD_{\pi^*} = D_{\pi^*}P^T$$

where D_{π^*} is the diagonal matrix formed from the stationary distribution vector. If the Markov chain satisfies detailed balance, it can be described as a random walk on an undirected graph with weight matrix $A = D_{\pi^*}^{-1/2} P D_{\pi^*}^{1/2}$.

The structure of the irreducible components of a Markov chain are reflected in in the nonzero structure of the stationary vectors. There is a basis of row eigenvectors that are indicators for *maximal accessing sets*, and a basis of stationary distributions that are supported on *minimal accessible sets*. For example, suppose a Markov chain is reducible with two irreducible components. If neither set can access the other, the transition matrix is block diagonal:

$$P = \begin{bmatrix} P_{11} & 0 \\ 0 & P_{22} \end{bmatrix}.$$

In this case, we can write row eigenvectors with eigenvalue 1 indicating the two blocks (both maximal accessible sets, since they cannot be accessed by any larger group), and column eigenvectors for stationary distributions on the two blocks, i.e.

$$\begin{bmatrix} e \\ 0 \end{bmatrix}^T \begin{bmatrix} P_{11} & 0 \\ 0 & P_{22} \end{bmatrix} = \begin{bmatrix} e \\ 0 \end{bmatrix}^T \qquad \begin{bmatrix} 0 \\ e \end{bmatrix}^T \begin{bmatrix} P_{11} & 0 \\ 0 & P_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ e \end{bmatrix}^T$$

$$\begin{bmatrix} P_{11} & 0 \\ 0 & P_{22} \end{bmatrix} \begin{bmatrix} \pi_1^* \\ 0 \end{bmatrix} = \begin{bmatrix} \pi_1^* \\ 0 \end{bmatrix} \qquad \begin{bmatrix} P_{11} & 0 \\ 0 & P_{22} \end{bmatrix} \begin{bmatrix} 0 \\ \pi_2^* \end{bmatrix}^T = \begin{bmatrix} 0 \\ \pi_2^* \end{bmatrix},$$

where e denotes the vector of all ones of a given size. Note that the eigenvectors are not uniquely determined: any row vector of the form $\begin{bmatrix} \alpha e^T & \beta e^T \end{bmatrix}$ is also a row vector of P in this case, and similiarly with the column eigenvectors.

If the second irreducible set can access the first irreducible set, the transition matrix is block upper triangular with a nonzero off-diaginal block:

$$P = \begin{bmatrix} P_{11} & 0 \\ 0 & P_{22} \end{bmatrix}.$$

In this case, the eigenvalue at 1 has multiplicity 1. The row eigenvectors with eigenvalue 1 indicates both blocks (the maximal accessing set, since some nodes can be accessed by everything), but the stationary distribution is only nonzero on the first block (the minimal accessible set), i.e.

$$\begin{bmatrix} e \\ e \end{bmatrix}^T \begin{bmatrix} P_{11} & P_{12} \\ 0 & P_{22} \end{bmatrix} = \begin{bmatrix} e \\ e \end{bmatrix}^T \qquad \begin{bmatrix} P_{11} & P_{12} \\ 0 & P_{22} \end{bmatrix} \begin{bmatrix} \pi_1^* \\ 0 \end{bmatrix} = \begin{bmatrix} \pi_1^* \\ 0 \end{bmatrix}.$$

Thus far, we have described properties of the Markov chain related to the stationary state or states. For an aperiodic chain with a unique stationary state, the rate of convergence to stationarity can be expressed in terms of the second largest eigenvalue, i.e.

$$\|\pi^* - \pi(t)\| \leq C|\lambda_2|^t$$

for some constant C. Sometimes, though, $|\lambda_2|$ is close to one, and this can be very telling. In particular, we see a much richer structure if we consider *metastable* or *slowly mixing* states associated with eigenvalues near one.

For example, *Simon-Ando* theory deals with *almost*-reducible Markov chains, e.g.

$$P = P^{\text{ref}} + E, \quad P^{\text{ref}} = \begin{bmatrix} P_{11} & 0 \\ 0 & P_{22} \end{bmatrix}.$$

In this case, we have an *almost* invariant subspace spanned by the invariant distributions for the irreducible components in the reference problem

$$P \begin{bmatrix} \pi_1^{\text{ref}} & 0 \\ 0 & \pi_2^{\text{ref}} \end{bmatrix} \approx \begin{bmatrix} \pi_1^{\text{ref}} & 0 \\ 0 & \pi_2^{\text{ref}} \end{bmatrix}$$

and

$$\begin{bmatrix} e & 0 \\ 0 & e \end{bmatrix}^T P \approx \begin{bmatrix} e & 0 \\ 0 & e \end{bmatrix}^T.$$

Hence, convergence of the Markov chain has two phases: a rapid mixing phase determined by the eigenvalues of P_{11}^{ref} and P_{22}^{ref}, and a slow equilibration phase in which we have

$$\pi(t) \approx \begin{bmatrix} \alpha_1(t)\, \pi_1^{\text{ref}} \\ \alpha_2(t)\, \pi_2^{\text{ref}} \end{bmatrix}.$$

Put differently, after a few steps, we mostly forget anything about the initial distribution other than whether we likely started in the first or the second set.

In the case of a Markov chain with disjoint connected components, when there is an eigenvalue at one with geometric multiplicity greater than one, the eigenvectors at one are not uniquely determined. However, the invariant subspaces (left and right) spanned by the eigenvectors *are* uniquely determined. In the weakly coupled cases, the eigenvectors associated with eigenvalues close to one generally are uniquely determined, but they are sensitive to small changes in the chain, and they may individually be hard to compute using standard iterations. However, for many applications, we don't care about the eigenvectors, but about the invariant subspace that they span. It is this subspace that is useful in clustering, for example, and it is equally useful whether we represent it in terms of the eigenvector basis or in terms of another basis.

This perspective on mixing of Markov chains gives us another way of thinking about graph clustering and partitioning via normalized cuts. The eigenvalues problem that arises from normalized cuts is

$$Ax = \lambda Dx,$$

and pre-multiplication by D^{-1} gives

$$D^{-1}Ax = P^T x = \lambda x.$$

That is, computing the dominant eigenvectors for the normalized cuts problem can either be interpreted as approximately solving an optimization problem or as finding indicators for fast-mixing subchains in a Markov chain with overall slow mixing.

7 Geometric embedding

While we started the lecture with graph bisection problems in which all we looked at was a single eigenvector, we have now seen several different excuses to try to extract information about a graph from a low dimensional invariant subspaces of associated matrices:

- Our discussion of Markov chains suggests that we may want to look at a dominant invariant subspace of P or P^T where P is the transition matrix for a Markov . We can think of this subspace as interesting because of what it tells us about natural clusters from the dynamics perspective (fast-mixing subchains within a slowly-mixing chain).

- We can also think of this subspace as interesting because it consists of "smooth" functions that form a basis for good approximate solutions to an optimization problem.

- Another way to think about things is through the lens of kernel approximation. Recall from last lecture that we defined a kernel function associated with the pseudoinverse of the Laplacian, and associated "Laplace features" with that kernel.

- These Laplace features can also be seen in terms of a low-dimensional embedding of the graph nodes in a Euclidean space in order to optimally approximate the resistance distance. More generally, if we have a squared distance matrix A between objects (i.e. $a_{ij} = \|x_i - x_j\|^2$), then the centered distance matrix

  $$B = -\frac{1}{2} HAH, \quad H = I - \frac{1}{n} ee^T$$

  is a positive semi-definite Gram matrix, and the r eigenvectors of B associated with the largest eigenvalues give us a mapping of objects to points in $\mathbb{R}^d$ that approximates the distance matrix as well as possible in a least squares sense.

Finding an "optimal" geometric embedding frequently reduces to an eigenvalue problem, but we can also compute such coordinate systems via other factorizations, such as pivoted QR or Cholesky factorization. This corresponds to an embedding that may not be optimal, but is focused on the relation between general data objects and a handful of reference examples.

Once we have a low-dimensional embedding that we think captures the relevant geometry, we can apply geometric methods such as k-means to try to find clusters.