CS264: Beyond Worst-Case Analysis
Lectures #9 and 10: Spectral Algorithms for Planted Bisection and Planted Clique[*]

Tim Roughgarden[†]

February 7 and 9, 2017

1 Preamble

Lectures #6–8 studied deterministic conditions (specifically, perturbation stability), for several NP-hard clustering and graph partitioning problems, under which the optimal solution can be recovered exactly in polynomial time. These stability conditions were reasonably natural, in that there was a plausible narrative about why "real-world" instances might tend to satisfy them, at least approximately.

Today we continue our ongoing study of the polynomial-time exact recovery of "planted" or "ground truth" solutions. We'll study probabilistic models (i.e., input distributions) that share some spirit with the stability conditions of Lectures #6–8, in that the optimal solution tends to "stick out." The goals are to design and analyze polynomial-time algorithms that recover the optimal solution with high probability (over the input distribution), and to understand how far the optimal solution to an NP-hard problem has to stick out before exact recovery is possible (w.h.p.) in polynomial time.

2 The Planted Bisection Problem

Our first case study concerns the Minimum Bisection problem, which is yet another cut problem. This problem is the same as the minimum cut problem, except that the two sides of the graph are additionally constrained to have equal size. That is, the input is an undirected graph $G = (V, E)$ with an even number of vertices, and the goal is to identify the cut (S, T) with $|S| = |T|$ that has the fewest number of crossing edges. This problem is NP-hard, in contrast to the Minimum Cut problem without the constraint of balanced sides. But

[†] Department of Computer Science, Stanford University, 474 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

perhaps we can solve the problem in polynomial time on "typical" instances? But then what do we mean by "typical?" We next develop generative models (i.e., distributions over inputs) that posit answers to this question.

2.1 Erdös-Renyi Random Graphs

We first review *Erdös-Renyi random graphs* as a starting point, before proceeding to a more informative model. Recall that a random graph from $\mathcal{G}(n, p)$ is a simple undirected graph on n vertices, where each of the $\binom{n}{2}$ possible edges is present independently with probability p. The special case of $p = \frac{1}{2}$ is the uniform distribution over all n-vertex graphs. It can also be interesting to study the case of small p (like $\Theta(\frac{\log n}{n})$ or even $\Theta(\frac{1}{n})$), reflecting the fact that real-world graphs tend to be sparse.

The $\mathcal{G}(n, p)$ model has at least two drawbacks that interfere with it usefully informing the design of graph algorithms. First, as with most pure average-case analysis frameworks, the data model is too specific; second, the model often fails to meaningfully differentiate between different algorithms.

Returning to the Minimum Bisection problem, assume for simplicity that the edge probability p is a constant. Then for every bisection (S, T) of a set of n vertices, the expected number of crossing edges in a random graph $G \in \mathcal{G}(n, p)$ is $pn^2/4$. A straightforward application of the Chernoff bound (see Homework #5) shows that, with high probability, the number of edges crossing *every* bisection is this same quantity, up to a $1 \pm o(1)$ factor. Thus even an algorithm that computes a *maximum* bisection is an almost optimal algorithm for computing a minimum bisection!

2.2 Random Graphs with a Planted Solution

Recall our primary motivation for our stability conditions in Lectures #6–8: we are often only interested in inputs that have an obviously meaningful solution, which we identify with being "clearly optimal" in some sense. *Planted graph models* are a nice probabilistic analog of such stability conditions, where such a "clearly optimal" solution exists with high probability.

Next we'll look at a planted version of the minimum bisection problem, originally proposed (in different communities) by Holland et al. [8] and Bui et al. [5].[1]

The idea is for "nature" to generate an instance of the Minimum Bisection problem according to the following random process (for a fixed vertex set V, with $|V|$ even, and parameters $p, q \in [0, 1]$):

1. Choose a partition (S, T) of V with $|S| = |T|$ uniformly at random.
2. Independently for each pair (i, j) of vertices inside the same cluster (S or T), include the edge (i, j) with probability p.

[1] Strictly speaking, the model in [5] is slightly different from the one considered here: it considers a random graph with a given number of edges and a given minimum bisection size.

3. Independently for each pair (i, j) of vertices in different clusters, include the edge (i, j) with probability q.

Thus the expected edge density inside the clusters is p, and between the clusters is q.

The difficulty of recovering the planted bisection (S, T) clearly depends on the gap between p and q. If $p = q$, then the problem is impossible (every bisection is equally likely to be the planted one). If $p = 1$ and $q = 0$ then the problem is trivial (the input is one clique on S and a disjoint clique on T). So the key question in this model is: how big does the gap $p - q$ need to be before exact recovery is possible in polynomial time (with high probability)? We will see an impressive result: even with $p - q \to 0$ as $n \to \infty$ (i.e., an asymptotically vanishing gap), computationally efficient exact recovery is possible.

2.3 Algorithmic Approaches

How would one tackle recovery problems in generative models with planted solutions? Many different approaches have been tried. We next list three of the most well-studied genres of algorithms in this context, listed roughly in order of increasing complexity and power.

1. *Combinatorial approaches.* We leave the term "combinatorial" safely undefined, but basically it refers to algorithms that work directly with the graph, rather than resorting to any continuous methods. (Why on earth would continuous methods be useful? See below.) For example, an algorithm that looks only at vertex degrees, subgraphs, shortest paths, etc., would be considered combinatorial. These algorithms are generally the simplest and will show up primarily on the homeworks.

2. *Spectral algorithms.* By "spectral," we mean an algorithm that uses linear algebra. Generally, this involves computing and using the eigenvectors of a suitable matrix derived from the input graph. These lectures discuss spectral algorithms in detail.

3. *Semidefinite programming (SDP).* We saw a brief glimpse of semidefinite programming last lecture, for exact recovery in sufficiently perturbation-stable instances of the Maximum Cut problem. As we'll see, they're also very useful for recovering planted solutions in the context of generative models, and are strictly more powerful than spectral algorithms in the "semi-random" models that we'll discuss in Lectures #11 and #12.

For the Planted Bisection problem in the parameter regime with $p, q = \Omega(1)$ and also $p - q = \Omega(1)$, simple combinatorial algorithms already recover the planted solution with high probability (see Homework #5). This seems great, no? Unfortunately, these algorithms do not resemble those that perform well in practice, and so this result is not very satisfying. To make the problem more difficult, we focus on the regime where $p - q$ is going to 0 with n; this will force us to work harder and develop better algorithms. It will turn out that a spectral algorithm achieves exact recovery (w.h.p.) provided $p - q = \Omega\left(\sqrt{\frac{\log n}{n}}\right)$. This is much more satisfying, not only because it is a stronger positive result, but also because the result is for a potentially practically useful algorithm. (Spectral techniques are one of the most popular methods of doing graph partitioning in practice.) Thus switching the problem from the $p-q = \Omega(1)$ regime to the $p-q = o(1)$ regime is valuable not because we literally believe that the latter is so much more faithful to "typical" instances, but because it encourages better algorithm design.[2,3]

2.4 A Canonical Spectral Algorithm

To give you a sense of what we'll be talking about, we'll record here the basic algorithm that we'll look at. The final algorithm requires a couple of modifications, but this is the gist. The modified algorithm will recover planted partitions even when $p - q$ is as small as $c\sqrt{\frac{\log n}{n}}$, for a sufficiently large constant c. This algorithm and result are due to McSherry [14]; our presentation is inspired by Spielman [17].

Canonical Spectral Algorithm

1. Let $\mathbf{M}$ denote the adjacency matrix of the given graph $G = (V, E)$—the (symmetric) $V \times V$ matrix with $M_{ij} = 1$ if $(i, j) \in E$ and 0 otherwise.

2. Compute the second eigenvector $\mathbf{u}$ of $\mathbf{M}$.

3. Set
$$A = \{i \in V : u_i > 0\}$$
and
$$B = \{i \in V : u_i \leq 0\}.$$

What's an eigenvector, again? Section 3 reviews the relevant linear algebra. What do eigenvectors have to do with graph partitioning? Section 4 explains the intuition.

3 Linear Algebra Review

This section reviews basic facts about the eigenvalues and eigenvectors of symmetric matrices, as covered in any standard linear algebra course. Section 4 instantiates these general facts for adjacency matrices of graphs and explains the meaning of eigenvectors in the context of the Planted Bisection problem.

[2] Sometimes, an initially dubious-looking mathematical model winds up leading to the "right" solution, which in turn provides an ex post facto justification for the model.

[3] Analogues of our bisection recovery results continue to hold for p, q as small as $\Theta((\log n)/n)$. For recent results where p, q are very small (meaning $O(1/n)$), and only approximate recovery is possible, see [13, 15, 16].

3.1 Eigenvectors and Eigenvalues

Happily, as we consider only undirected graphs, we will only need to think about *symmetric* matrices, meaning square $(n \times n)$ matrices $\mathbf{M}$ with $M_{ij} = M_{ji}$ for every $i, j \in \{1, 2, \ldots, n\}$. To develop our geometric intuition, we will often think about $\mathbf{M}$ in terms of the corresponding operator (i.e., function from $\mathbb{R}^n$ to $\mathbb{R}^n$) given by $\mathbf{v} \mapsto \mathbf{M}\mathbf{v}$. Thinking this way, we can talk about how $\mathbf{M}$ "moves" different vectors $\mathbf{v}$.

An *eigenvector* of $\mathbf{M}$ is a nonzero vector $\mathbf{v}$ such that $\mathbf{M}\mathbf{v} = \lambda\mathbf{v}$ for some $\lambda \in \mathbb{R}$. The scalar λ is the corresponding *eigenvalue*. Thus $\mathbf{M}$ simply scales the vector $\mathbf{v}$, possibly after flipping it to the opposite direction (in the case that $\lambda < 0$). We can think of an eigenvector as a "direction of stretch" for $\mathbf{M}$, and the eigenvalue as the magnitude (and orientation) of the stretch. Note that if $\mathbf{v}$ is an eigenvector, then so is $\alpha\mathbf{v}$ for every $\alpha \neq 0$ (with the same eigenvalue). For this reason, we are free to assume for convenience that eigenvectors have unit length.[4]

3.2 Spectral Decomposition

The following "spectral theorem" will be very important for us. See any linear algebra book for a proof (e.g., by induction on the number of dimensions).

Theorem 3.1 (Spectral Theorem for Symmetric Matrices) *If* $\mathbf{M}$ *is an* $n \times n$ *symmetric matrix, then there is an orthonormal basis* $\mathbf{u}_1, \ldots, \mathbf{u}_n$ *of eigenvectors with real corresponding eigenvalues* $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$.

A few reminders: a *basis* of $\mathbb{R}^n$ is a set of n linearly independent vectors (i.e., none is a linear combination of the others), which necessarily span the whole space. Once a basis is fixed, every vector $\mathbf{v}$ has a unique representation as a linear combination of the basis vectors. A basis is *orthonormal* if every basis vector has unit length and each pair $\mathbf{v}, \mathbf{w}$ of distinct basis vectors are orthogonal (i.e., have inner product $\langle \mathbf{v}, \mathbf{w} \rangle = \sum_{i=1}^n v_i w_i$ equal to 0).

Theorem 3.1 implies that every symmetric matrix, viewed as an operator, is geometrically really, really simple—as simple as a diagonal matrix. To see this, take your favorite vector $\mathbf{v} \in \mathbb{R}^n$ and write it as a linear combination of $\mathbf{M}$'s eigenvectors:

$$\mathbf{v} = \sum_{i=1}^{n} \alpha_i \mathbf{u}_i, \tag{1}$$

with coefficients given by the projections of $\mathbf{v}$ onto the $\mathbf{u}_i$'s:

$$\alpha_i = \langle \mathbf{u}_i, \mathbf{v} \rangle \tag{2}$$

for $i = 1, 2, \ldots, n$. (To see that (2) follows from (1), take inner products of both sides with $\mathbf{u}_i$ and use that the $\mathbf{u}_j$'s are orthonormal.) Now apply the matrix $\mathbf{M}$. By linearity, $\mathbf{M}$ acts

[4] Unless otherwise stated, all distances are Euclidean. So the length $\|\mathbf{v}\|$ of a vector $\mathbf{v} \in \mathbb{R}^d$ is its 2-norm, $\sqrt{\sum_{i=1}^d v_i^2}$.

independently along each eigenvector:

$$\mathbf{M}\mathbf{v} = \sum_{i=1}^{n} \alpha_i \mathbf{M}\mathbf{u}_i = \sum_{i=1}^{n} \lambda_i \alpha_i \mathbf{u}_i.$$

All of this gives the following *spectral decomposition* of $\mathbf{M}$:

$$\mathbf{M} = \underbrace{\begin{pmatrix} | & | & & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_n \\ | & | & & | \end{pmatrix}}_{\mathbf{Q}} \cdot \underbrace{\begin{pmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_n \end{pmatrix}}_{\mathbf{D}} \cdot \underbrace{\begin{pmatrix} - & \mathbf{u}_1 & - \\ - & \mathbf{u}_2 & - \\ & \vdots & \\ - & \mathbf{u}_m & - \end{pmatrix}}_{\mathbf{Q}^\top}. \tag{3}$$

Here $\mathbf{Q}^\top$ re-represents a vector $\mathbf{v}$ in terms of the basis of $\mathbf{u}_i$'s (since $\mathbf{Q}^\top\mathbf{v} = (\langle \mathbf{u}_1, \mathbf{v}\rangle, \ldots, \langle \mathbf{u}_n, \mathbf{v}\rangle)$, i.e., the vector of α_i's from (1)). The diagonal matrix $\mathbf{D}$ then scales and possibly flips independently along the directions of the $\mathbf{u}_i$'s. Finally, $\mathbf{Q}$ translates the result back into the standard basis.[5] In this sense, every symmetric matrix is really a "diagonal in disguise," acting like a diagonal matrix after a suitable "rotation" of $\mathbb{R}^n$.[6]

3.3 Variational Characterization of Eigenvalues

The eigenvalues and eigenvectors of a symmetric matrix can be computed efficiently, for example using the singular value decomposition (SVD). In Matlab, it just takes one line (via the command `eig`). But suppose we need to get a handle on the eigenvalues or eigenvectors of a matrix analytically, for the purposes of a proof? We next describe an alternative definition of eigenvalues that will be extremely useful over the next several lectures. The key point is that *eigenvalues and eignevectors arise as the optimal solutions of natural optimization problems.*

Theorem 3.2 (Variational Characterization) *Let* $\mathbf{M}$ *be an* $n \times n$ *symmetric matrix. Then the first eigenvalue and corresponding eigenvector of* $\mathbf{M}$ *satisfy*

$$\lambda_1 = \max_{\|\mathbf{u}\|=1} \mathbf{u}^\top \mathbf{M}\mathbf{u}$$

and

$$\mathbf{u}_1 \in \operatorname*{argmax}_{\|\mathbf{u}\|=1} \mathbf{u}^\top \mathbf{M}\mathbf{u},$$

respectively. For $i = 2, 3, \ldots, n$*,*

$$\lambda_i = \max_{\substack{\|\mathbf{u}\|=1 \\ \mathbf{u} \perp \mathbf{u}_1, \ldots, \mathbf{u}_{i-1}}} \mathbf{u}^\top \mathbf{M}\mathbf{u}$$

[5] Note that when $\mathbf{Q}$ is an orthogonal matrix (i.e., is a square matrix with orthonormal columns), then $\mathbf{Q}\mathbf{Q}^\top = \mathbf{Q}^\top\mathbf{Q} = \mathbf{I}$. Hence the inverse of an orthogonal matrix $\mathbf{Q}$ is just its transpose $\mathbf{Q}^\top$.

[6] The identity in (3) makes it clear why the spectral theorem is often described as "symmetric matrices are diagonalizable by orthogonal matrices."

and

$$\mathbf{u}_i = \operatorname*{argmax}_{\substack{\|\mathbf{u}\|=1 \\ \mathbf{u}\perp\mathbf{u}_1,\ldots,\mathbf{u}_{i-1}}} \mathbf{u}^\top \mathbf{M}\mathbf{u},$$

where $\mathbf{u}\perp\mathbf{v}$ *denotes that* $\mathbf{u},\mathbf{v}$ *are orthogonal.*[7]

The proof is not difficult, and we leave it to Homework #5. Intuitively, if we parse $\mathbf{u}^\top\mathbf{M}\mathbf{u}$—known as the *quadratic form* defined by $\mathbf{M}$—as $\mathbf{u}^\top(\mathbf{M}\mathbf{u})$, we see that the quadratic form measures the projection length of a vector $\mathbf{u}$ onto its image under the map $\mathbf{M}$. Since projections are maximized for colinear vectors (all else being equal), it makes sense that eigenvectors arise as maximizers of a quadratic form.

For example, Theorem 3.2 immediately gives an easy way to lower bound the value of the top eigenvalue of a matrix—just exhibit any unit vector $\mathbf{v}$, and $\mathbf{v}^\top\mathbf{M}\mathbf{v}$ bounds λ_1 from below. The same trick can be used for lower eigenvalues, provided the previous eigenvectors have been identified (so that the orthogonality constraint can be met).

4 A Spectral Approach to Planted Bisection

4.1 The Spectrum of the Expected Adjacency Matrix

We now return to the problem of recovering the planted bisection in the model from Section 2. Recalling the canonical spectral algorithm (Section 2.4), let $\mathbf{M}$ denote the adjacency matrix of the input graph $G = (V, E)$ (a random variable). The key question for understanding the algorithm is: what does the second eigenvector of $\mathbf{M}$ have to do with anything?

Fix the random choice (S, T) of a planted bisection. Let's look at the "expected adjacency matrix" $\widehat{\mathbf{M}}$, where the randomness is over the choice of edges (given the choice of (S, T)). After permuting the rows and columns so that those corresponding to S come before those in T, $\widehat{\mathbf{M}}$ looks like

$$\widehat{\mathbf{M}} = \left(\begin{array}{cccc|cccc} p & p & \cdots & p & q & q & \cdots & q \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ p & p & \cdots & p & q & q & \cdots & q \\ \hline q & q & \cdots & q & p & p & \cdots & p \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ q & q & \cdots & q & p & p & \cdots & p \end{array}\right). \tag{4}$$

Well almost: the adjacency matrix $\mathbf{M}$ has zeroes along its diagonal (with probability 1), while the matrix $\widehat{\mathbf{M}}$ has p's along its diagonal. That it, $\widehat{\mathbf{M}}$ is the expected value of $\mathbf{M}$, plus $p\mathbf{I}$, where $\mathbf{I}$ is the $n \times n$ identify matrix. We won't think twice about adding or subtracting a multiple $\alpha\mathbf{I}$ of the identity matrix for convenience—it leaves the eigenvectors unchanged (why?) and every eigenvalue shifts by α.

[7] This holds no matter how $\mathbf{u}_1, \ldots, \mathbf{u}_{i-1}$ are chosen from the previous argmax's.

The point is that the matrix $\widehat{\mathbf{M}}$ is simple enough that we can just explicitly compute its eigenvectors and eigenvalues. First, note that there are only two distinct types of columns; this implies that $\widehat{\mathbf{M}}$ has rank (at most) 2.[8] The rank of a symmetric matrix is exactly the number of nonzero eigenvalues (counting multiplicity). This is evident from the spectral decomposition (3)—the rank of the diagonal matrix $\mathbf{D}$ is obviously the number of nonzero entries, and the orthogonal matrices $\mathbf{Q}$ and $\mathbf{Q}^\top$, as nonsingular matrices, preserve the rank. Thus $n-2$ of $\widehat{\mathbf{M}}$'s eigenvalues are 0. It remains to determine the other two.

Staring at (4), one eigenvector is easy to guess. Let $\mathbf{v}$ be the all-ones vector. Then $\widehat{\mathbf{M}}\mathbf{v} = (\frac{n}{2}(p+q), \ldots, \frac{n}{2}(p+q)) = \frac{n}{2}(p+q)\mathbf{v}$, and so $\widehat{\mathbf{M}}$ has one eigenvalue equal to $\frac{n}{2}(p+q)$.

Remember that eigenvectors are orthogonal to each other. So our next guess should be orthogonal to the all-ones vector—equivalently, the entries of the vector should sum to 0. The simplest scenario would be if the eigenvector takes on only two possible values (one positive, one negative). So the sensible next guess is to take $\mathbf{w}$ to be the vector that is 1 on coordinates corresponding to S and -1 on coordinates corresponding to T. (Our algorithm doesn't know S and T, of course, but we're just doing analysis here.) Since $|S| = |T|$, we have $\langle \mathbf{v}, \mathbf{w} \rangle = 0$. And $\widehat{\mathbf{M}}\mathbf{w}$ is $\frac{n}{2}(p-q)$ in coordinates corresponding to S, and $\frac{n}{2}(q-p)$ in coordinates corresponding to T (as you should check). Thus $\mathbf{w}$ is indeed an eigenvector, with eigenvalue $\frac{n}{2}(p-q)$.

To summarize, as $p > q > 0$, the expected adjacency matrix $\widehat{\mathbf{M}}$ has first eigenvalue $\frac{n}{2}(p+q)$, second eigenvalue $\frac{n}{2}(p-q)$, and remaining eigenvalues 0. Moreover, the second eigenvector has opposite signs for vertices in S and for vertices in T. Thus the canonical spectral algorithm (Section 2.4) works perfectly when applied to $\widehat{\mathbf{M}}$—the planted bisection can be immediately read off from the sign pattern of its second eigenvector (!).

4.2 The High-Level Approach

The hope is that, since the adjacency matrix $\mathbf{M}$ of G has expected value $\widehat{\mathbf{M}}$ (minus $p\mathbf{I}$), the canonical spectral algorithm continues to work well (with high probability) when applied to $\mathbf{M}$. We will execute this plan using some basic facts from matrix perturbation theory, which track how the eigenvalues and eigenvectors of a matrix change when another (ideally "small") matrix is added to it.

In more detail, define $\mathbf{R} = \mathbf{M} - \widehat{\mathbf{M}} - p\mathbf{I}$, so that $\mathbf{M} = \widehat{\mathbf{M}} + \mathbf{R} + p\mathbf{I}$. We think of $\widehat{\mathbf{M}}$ as the "base matrix" and $\mathbf{R}$ as the "perturbing matrix." The random matrix $\mathbf{R}$ is easy to describe:

$$\mathbf{R} = \left(\begin{array}{c|c} -p/(1-p) & -q/(1-q) \\ \hline -q/(1-q) & -p/(1-p) \end{array} \right). \qquad (5)$$

[8] Recall there are many equivalent definitions of the rank of a matrix, including the maximum number of linearly independent columns, the maximum number of linearly independent rows, and the dimension of the range of the matrix (which is a subspace of the target space).

where "$-p/(1-p)$" means that an entry is either $-p$ (with probability $1-p$) or $1-p$ (with probability p), and similarly for "$-q/(1-q)$." The picture in (5) is inaccurate on the diagonal, which is $-p$ with probability 1. The entries above the diagonal are independent; their values then force the values of the entries below the diagonal (as the matrix is symmetric with probability 1).

The hope is now that:

1. The "noise" $\mathbf{R}$ is small relative to the "signal" $\widehat{\mathbf{M}}$ about the planted bisection.

2. When noise is small compared to the signal, adding the noise doesn't change the signal (i.e., the second eigenvector) by much.

Of course, to implement this we need to define what we mean by a matrix being "small" and by a vector "not changing much." We tackle these issues in Sections 5 and 6, respectively.

5 Eigenvalues of Random Symmetric Matrices

Write $\widehat{\mathbf{R}} = \mathbf{R} + p\mathbf{I}$, so that $\widehat{\mathbf{R}}$ has the form in (5) except with zeroes on the diagonal (with probability 1). Note that every entry of $\widehat{\mathbf{R}}$ has expectation 0 (as you should check) and is bounded between -1 and 1. Does this imply that $\widehat{\mathbf{R}}$ is "small" with high probability?

We'll measure the "size" of a symmetric matrix using the operator norm, which is just the largest magnitude of one of its eigenvalues. That is, if $\mathbf{M}$ has eigenvalues $\lambda_1 \geq \cdots \lambda_2 \geq \cdots \geq \lambda_n$, then $\|\mathbf{M}\| = \max_i |\lambda_i|$, which in turn is either λ_1 or $-\lambda_n$, whichever is larger. Equivalently, the operator norm is largest magnitude by which $\mathbf{M}$ stretches any vector.[9]

As you can imagine, there has been a lot of work on understanding the distribution of eigenvalues of random symmetric matrices.[10] Here is tightest quantitative bound known (as a function of n).

Theorem 5.1 ([18]) *Let* $\mathbf{P}$ *be a random matrix with the following properties:*

(i) With probability 1, $\mathbf{P}$ *is symmetric, with only zeroes on the diagonal.*

(ii) Each random variable P_{ij} *has expectation 0, and is bounded between -1 and 1.*

(iii) The random variables $\{P_{ij}\}_{1 \leq i < j \leq n}$ *are independent.*

Then with probability approaching 1 as $n \to \infty$,

$$\|\mathbf{P}\| \leq \sqrt{n} + O(n^{1/4}).$$

[9] In particular, $\|\mathbf{Mv}\| \leq \|\mathbf{M}\|\|\mathbf{v}\|$ for every $\mathbf{v}$, where the first and third norms denote the Euclidean norm, and the middle one the operator norm.

[10] For example, *Wigner's semicircle law* states that as the dimension n of a random symmetric matrix goes to infinity (where each entry has mean 0 and variance 1), the distribution over its eigenvalues (after scaling by $\sqrt{n}$) has limiting density $f(x) = \frac{1}{2\pi}\sqrt{(4-x^2)_+}$, where y_+ denotes $\max\{0, y\}$ (i.e., a semi-circle) [19].

Theorem 5.1 takes a fair bit of work and is outside the scope of this class. It's not so difficult to prove slightly weaker bounds, however, which are good enough for our purposes. We'll sketch a proof of a bound that is weaker by a $\Theta(\sqrt{\log n})$ factor.

Theorem 5.2 *Under the assumptions of Theorem 5.1, there is a constant $c_2 > 0$ such that, for every $\delta > 0$, with probability at least $1 - \delta$,*

$$\|\mathbf{P}\| \le c_2 \sqrt{n \log n \log \tfrac{1}{\delta}}.$$

A slightly more complicated variation of the following argument gives a bound of $O(\sqrt{n})$, which is weaker than the bound in Theorem 5.1 by only a constant factor (see Theorem 12.1 and Homework #5).

The variational characterization of eigenvalues (Theorem 3.2) implies that to upper bound the operator norm of a matrix, it is enough to bound the maximum magnitude by which any vector gets stretched (possibly after flipping) by the matrix. Our proof sketch now has three steps: (1) proving that any given direction is unlikely to be stretched much by $\mathbf{P}$; (2) that the set of all directions is well approximated by a finite set, to which the union bound applies; (3) if no direction in the finite set is stretched much by $\mathbf{P}$, then no direction at all is stretched much. Homework #5 asks you to fill in the details of each of the steps, corresponding to the following three lemmas.

Lemma 5.3 *Let $\mathbf{u}$ be a unit vector and $t \ge 0$. Under the assumptions of Theorem 5.1,*

$$\mathbf{Pr}_{\mathbf{P}}\left[|\mathbf{u}^\top \mathbf{P} \mathbf{u}| \ge t\right] \le 2e^{-t^2}.$$

Note that assumption (ii), that every entry of $\mathbf{P}$ has mean zero, implies that, for every fixed $\mathbf{u}$, the expected value of $\mathbf{u}^\top \mathbf{P} \mathbf{u} = \sum_{i,j} P_{ij} u_i u_j$ is 0 (by linearity of expectation). Lemma 5.3 then follows from a concentration inequality known as Hoeffding's inequality, which is a cousin of the Chernoff bound, parameterized by additive rather than multiplicative error.

The next lemma saves us from trying to take a union bound over the infinite set of unit vectors. By an *ϵ-net* of a metric space (X, d), we mean a subset $S \subseteq X$ such that, for every $x \in X$ there exists $y \in S$ with $d(x, y) < \epsilon$. That is, every point of X is close to one of the representatives in the ϵ-net S.

Lemma 5.4 *The sphere in $\mathbb{R}^n$ admits an ϵ-net N with $|N| \approx (\frac{2}{\epsilon})^n$ (with respect to Euclidean distance).*

The ϵ-net can be constructed greedily, with termination following from a volume argument.

The final lemma extends an upper bound on $|\mathbf{u}^\top \mathbf{P} \mathbf{u}|$ for all ϵ-net points $\mathbf{u} \in N$ to one for all points of the unit sphere (with minor loss).

Lemma 5.5 *If $|\mathbf{u}^\top \mathbf{P} \mathbf{u}| \le t$ for all vectors $\mathbf{u}$ in an ϵ-net of the sphere in $\mathbb{R}^n$, then $|\mathbf{u}^\top \mathbf{P} \mathbf{u}| \le t + n\epsilon$ for all unit vectors $\mathbf{u} \in \mathbb{R}^n$.*

Lemmas 5.3–5.5 imply Theorem 5.2. To see this, take $\epsilon = \frac{1}{n}$ (say). Then the size of the ϵ-net N in Lemma 5.4 is $\approx (2n)^n$. Combining Lemma 5.3 with a union bound over the ϵ-net points in Lemma 5.4 shows that $|\mathbf{u}^\top \mathbf{P}\mathbf{u}| \leq t$ for all $\mathbf{u} \in N$, except with probability at most $(2n)^n \cdot 2e^{-t^2}$. This failure probability is at most δ provided $t = \Omega(\sqrt{n \log n \log \frac{1}{\delta}})$. In this case, by Lemma 5.5, $|\mathbf{u}^\top \mathbf{P}\mathbf{u}| \leq t + 1 = \Theta(\sqrt{n \log n})$ for all unit vectors $\mathbf{u}$ (for constant δ). The variational characterization of eigenvalues (Theorem 3.2) then implies that $\|\mathbf{P}\| = O(\sqrt{n \log n})$ with arbitrarily large constant probability.

6 A Little Matrix Perturbation Theory

In this section we make precise the idea that adding a perturbation matrix $\mathbf{P}$ to a base matrix $\widehat{\mathbf{M}}$ should not change the eigenvectors much, provided the "noise" is significantly smaller than "the signal." The formal statement is a famous result (from 1970) known as the Davis-Kahan theorem.[11]

Theorem 6.1 (Davis-Kahan Theorem [6]) *Let $\widehat{\mathbf{M}}$ be an $n \times n$ symmetric matrix with eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ and corresponding unit-length eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n$. Let $\mathbf{P}$ be an $n \times n$ symmetric matrix, and let $\mathbf{M} := \widehat{\mathbf{M}} + \mathbf{P}$ have unit-length eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$. Let θ_i be the smaller angle between the lines through $\mathbf{u}_i$ and $\mathbf{v}_i$. Then*

$$\sin \theta_i \leq \frac{2\|\mathbf{P}\|}{\min_{j \neq i} |\lambda_i - \lambda_j|}. \tag{6}$$

We interpret each side of the bound (6) in turn. On the left hand side, note that $\theta_i \leq \frac{\pi}{4}$ (as the smaller angle between two vectors), and we can multiply $\mathbf{u}_i$ and/or $\mathbf{v}_i$ by -1 so that the angle between them is θ_i. Elementary trigonometry then implies that

$$\|\mathbf{u}_i - \mathbf{v}_i\| \leq \sqrt{2} \sin \theta_i, \tag{7}$$

with equality holding when $\theta_i = \frac{\pi}{4}$ (as you should check). Thus the left-hand side of (6) is really just a bound on the Euclidean distance between the ith eigenvector before and after the perturbation matrix $\mathbf{P}$ is added. On the right-hand side of (6), the denominator is the size of the "buffer zone" between the ith eigenvalue and the other eigenvalues.[12] The numerator of the right-hand side is the "size" of the perturbation, as measured by the operator norm (i.e., largest magnitude of any eigenvalue). So what Theorem 6.1 really says is:

> if all of the eigenvalues of the perturbation matrix have magnitude well less than the buffer zone around the ith eigenvalue of the base matrix, then the perturbation has little effect on the ith eigenvector.

[11] Actually, the real theorem does not have the "2" in the numerator, but the weaker bound here is sufficient for our purposes.

[12] Simple examples show that when two eigenvalues are very close together, even very small perturbations can radically change the corresponding eigenvectors (Homework #5).

The proof of Theorem 6.1 is actually not all that hard, and we give it in Section 7. The reader is also free to skip to Section 8, where we put all of our tools together for the recovery of a planted bisection.

7 Proof of the Davis-Kahan Theorem

First, we need the following lemma from linear algebra, which is sort of a "triangle inequality for eigenvalues." It states that when one matrix $\mathbf{A}$ is added to another one $\mathbf{B}$, all of the eigenvalues of $\mathbf{A}$ move by at least the minimum and by at most the maximum eigenvalue of $\mathbf{B}$. We leave the proof to Homework #5.

Lemma 7.1 *Let $\mathbf{A}, \mathbf{B}$ be $n \times n$ symmetric matrices with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$ and $\mu_1 \geq \cdots \geq \mu_n$. For every $i = 1, 2, \ldots, n$, the ith eigenvalue ν_i of $\mathbf{A} + \mathbf{B}$ satisfies*

$$\lambda_i + \mu_n \leq \nu_i \leq \lambda_i + \mu_1.$$

We can now prove Theorem 6.1.

Proof of Theorem 6.1: The key idea is to focus on how much the new ith eigenvector moves under the old matrix $\widehat{\mathbf{M}}$ (i.e., $\widehat{\mathbf{M}}\mathbf{v}_i$), relative to how it would move were it actually its ith eigenvector. The proof has two steps:

(1) When λ_i is well separated from the other eigenvalues, if θ_i is big, then $\widehat{\mathbf{M}}\mathbf{v}_i$ must be far from $\lambda_i \mathbf{v}_i$.

(2) If $\|R\|$ is small, then $\widehat{\mathbf{M}}\mathbf{v}_i$ is not far from $\lambda_i \mathbf{v}_i$.

For the Step 1 details, first write the new ith eigenvector $\mathbf{v}_i$ in the basis of the old ones:

$$\mathbf{v}_i = \sum_{j=1}^{n} c_j \mathbf{u}_j.$$

Since the $\mathbf{u}_j$'s are orthonormal, $c_j = \langle \mathbf{v}_i, \mathbf{u}_j \rangle$. Since $\mathbf{u}_i, \mathbf{v}_i$ are both unit vectors, we have $c_i = \cos\theta_i$.

Next, let δ denote $\min_{j \neq i} |\lambda_j - \lambda_i|$. Since $\widehat{\mathbf{M}}$ operates separately on the $\mathbf{u}_j$'s, we have

$$\widehat{\mathbf{M}}\mathbf{v}_i = \sum_{j=1}^{n} \lambda_j c_j \mathbf{u}_j,$$

or

$$\widehat{\mathbf{M}}\mathbf{v}_i - \lambda_i \mathbf{v}_i = \sum_{j=1}^{n} (\lambda_j - \lambda_i) c_j \mathbf{u}_j.$$

Thus,

$$\begin{aligned}\|\widehat{\mathbf{M}}\mathbf{v}_i - \lambda_i \mathbf{v}_i\|^2 &= \sum_{j=1}^{n} (\lambda_j - \lambda_i)^2 c_j^2 \\ &\geq \delta^2 \sum_{j \neq i} c_j^2 \\ &= \delta^2 \underbrace{(1 - c_i^2)}_{=1-\cos^2 \theta_i} \qquad (8) \\ &= \delta^2 \sin^2 \theta_i,\end{aligned}$$

where in (8) we're using that $\sum_{j=1}^{n} c_j^2 = 1$ (since the $\mathbf{u}_j$'s are an orthonormal basis and $\mathbf{v}_i$ is a unit vector). Rewriting, we have

$$\sin \theta_i \leq \frac{1}{\delta} \|\widehat{\mathbf{M}}\mathbf{v}_i - \lambda_i \mathbf{v}_i\|. \qquad (9)$$

Thus θ_i is big only if $\widehat{\mathbf{M}}\mathbf{v}_i$ is far from $\lambda_i \mathbf{v}_i$.

For the details of the second step, we write

$$\begin{aligned}\|\widehat{\mathbf{M}}\mathbf{v}_i - \lambda_i \mathbf{v}_i\| &= \|(\mathbf{M} - \mathbf{P})\mathbf{v}_i - \lambda_i \mathbf{v}_i\| \\ &= \|(\mathbf{M} - \lambda_i \mathbf{I})\mathbf{v}_i - \mathbf{P}\mathbf{v}_i\| \\ &\leq \|(\mathbf{M} - \lambda_i \mathbf{I})\mathbf{v}_i\| + \|\mathbf{P}\mathbf{v}_i\| \\ &= \|(\mu_i - \lambda_i)\mathbf{v}_i\| + \|\mathbf{P}\mathbf{v}_i\|,\end{aligned}$$

where the inequality is from the triangle inequality for the Euclidean norm, and μ_i denotes the ith eigenvalue of $\mathbf{M}$ (corresponding to the eigenvector $\mathbf{v}_i$). Now, Lemma 7.1 implies that

$$\mu_i \leq \lambda_i + \|\mathbf{P}\|,$$

and hence

$$\|\widehat{\mathbf{M}}\mathbf{v}_i - \lambda_i \mathbf{v}_i\| \leq \|\|\mathbf{P}\|\mathbf{v}_i\| + \|\mathbf{P}\mathbf{v}_i\| \leq 2\|\mathbf{P}\|, \qquad (10)$$

since $\mathbf{v}_i$ is a unit vector.

Combining (9) and (10) yields

$$\sin \theta_i \leq \frac{2\|\mathbf{P}\|}{\delta},$$

as desired. ■

8 Planted Bisection: Partial Recovery

We can now put all of the pieces together. First, we have $\widehat{\mathbf{M}}$ (see (4)), the expected adjacency matrix plus $p\mathbf{I}$, for which $\lambda_1(\widehat{\mathbf{M}}) = \frac{n}{2}(p+q)$, $\lambda_2(\widehat{\mathbf{M}}) = \frac{n}{2}(p-q)$, and $\lambda_i(\widehat{\mathbf{M}}) = 0$ for all $i \geq 3$

(Section 4.1). We can assume that $q > \frac{p}{3}$ (otherwise the problem is easy—why?) and so $\lambda_2(\widehat{\mathbf{M}})$ is closer to 0 than to $\lambda_1(\widehat{\mathbf{M}})$. Thus, if

$$p - q \geq c_1 \frac{\sqrt{\log n}}{\sqrt{n}}$$

for a constant $c_1 > 0$, then

$$\min_{i \neq 2} \left| \lambda_2(\widehat{\mathbf{M}}) - \lambda_i(\widehat{\mathbf{M}}) \right| = \lambda_2(\widehat{\mathbf{M}}) \geq \frac{c_1 \sqrt{n \log n}}{2}. \tag{11}$$

Also, recall from Section 4.1 that the second eigenvector $\mathbf{u}_2$ of $\widehat{\mathbf{M}}$ corresponds precisely to the planted bisection—all entries of $\mathbf{u}_2$ are $\pm\frac{1}{\sqrt{n}}$, with the sign depending on which side of the bisection a vertex is on.

Second, we have the perturbation matrix $\mathbf{R}$ in (5), which is defined so that $\widehat{\mathbf{M}} + \mathbf{R}$ equals the (random) adjacency matrix $\mathbf{M}$ of the input graph. We can apply Theorem 5.2 to the random matrix $\widehat{\mathbf{R}} = \mathbf{R} + p\mathbf{I}$, to obtain

$$\|\mathbf{R}\| = \max_{i=1}^{n} |\lambda_i(\mathbf{R})| = -p + \max_{i=1}^{n} \left| \lambda_i(\widehat{\mathbf{R}}) \right| \leq c_3 \sqrt{n \log n} \tag{12}$$

with high probability, where $c_3 > 0$ is a sufficiently large constant.

Let $\mathbf{v}_2$ denote the second eigenvector of the actual adjacency matrix $\mathbf{M}$. Multiply $\mathbf{u}_2$ and/or $\mathbf{v}_2$ by -1 as necessary, so that the angle between them is at most $\frac{\pi}{4}$. Plugging in (11) and (12) into the numerator and denominator of the bound (6) of the Davis-Kahan theorem, and using (7), we have (with large probability)

$$\|\mathbf{u}_2 - \mathbf{v}_2\| \leq \frac{4\sqrt{2} c_3}{c_1}. \tag{13}$$

Now let (A, B) be the cut defined by $\mathbf{v}_2$, as in the third step of the canonical spectral algorithm of Section 2.4, and suppose that ℓ vertices are classified incorrectly with respect to the planted bisection. (Note that (A, B) need not be a bisection.) Equivalently, let ℓ denote the number of coordinates for which $\mathbf{u}_2, \mathbf{v}_2$ have opposite signs. Since all entries of $\mathbf{u}_2$ are $\pm\frac{1}{\sqrt{n}}$, we have

$$\|\mathbf{u}_2 - \mathbf{v}_2\| \geq \sqrt{\frac{\ell}{n}},$$

and hence (with large probability, using (13))

$$\ell \leq \left(\frac{32 c_3^2}{c_1^2} \right) n,$$

which is at most $\frac{n}{32}$, provided we take $c_1 \geq 32 c_3$. (Recall that c_3 is derived from Theorem 5.1 and hence is not under our control, but we can choose c_1 in our assumption about the gap between p and q.)

So we've shown that, with high probability, the canonical spectral algorithm of Section 2.4 correctly classifies at least 97% of the vertices of the input graph. This type of result is known as *partial recovery*—the algorithm can make some errors, but it does much better than random guessing. The next section adds a postprocessing step to achieve *exact* recovery, with all vertices correctly labeled (w.h.p.).[13]

9 Planted Bisection: Exact Recovery

The basic idea for turning a mostly correct solution into an entirely correct one is to, for each vertex v in parallel, classify v according to the side of the mostly correctly cut that contains more of v's neighbors. In effect, v's neighbors "vote" for which side of the bisection v should be on. Intuitively, because $p - q = \Omega(\sqrt{\log n}/\sqrt{n})$ and there are not too many errors, every vertex will have more neighbors on the appropriate side than on the other side. In practice (and maybe also in theory), the most straightforward implementation of this should work fine. To avoid problematic dependencies in the analysis, however, it is convenient to first split the input graph into two, and use the mostly correct solution of each side to correctly classify the vertices on the other side (to enable the principle of deferred decisions).

The Final Planted Bisection Algorithm

1. Randomly partition the vertex set V into two equal-size groups, V_1 and V_2. Let H_1 and H_2 denote the subgraphs of G induced by V_1 and V_2.

2. Run the canonical spectral algorithm separately on H_1 and H_2, to obtain cuts (A_1, B_1) of H_1 and (A_2, B_2) of H_2.

3. Place each vertex $v \in V_1$ into either $\hat{A}_1$ or $\hat{B}_1$, according to whether v has more neighbors in A_2 or B_2 (breaking ties arbitrarily). Similarly, place each vertex $v \in V_2$ into either $\hat{A}_2$ or $\hat{B}_2$, according to whether v has more neighbors in A_1 or B_1.

4. Return either the cut $(\hat{A}_1 \cup \hat{A}_2, \hat{B}_1 \cup \hat{B}_2)$ or the cut $(\hat{A}_1 \cup \hat{B}_2, \hat{B}_1 \cup \hat{A}_2)$, whichever one is a bisection with fewer crossing edges. (If neither is a bisection, the algorithm fails.)

This is the final algorithm, and it correctly recovers the entire planted partition with high probability; see Homework #5 for more details.

[13] If we replace the coarse bound in Theorem 5.2 by that in Theorem 5.1 or 12.1, everything in this section remains true with an even smaller gap $p - q \geq \frac{c}{\sqrt{n}}$ for sufficiently large c. However, we still need the extra $\sqrt{\log n}$ factor in the gap to achieve exact recovery in Section 9.

10 The Planted Clique Problem

10.1 The Model

In the *maximum clique* problem, the goal is to identify the largest subset of vertices of an undirected graph that are mutually adjacent. The problem is fundamental and also comes up in applications, such as social network analysis (where cliques indicate a tightly knit community). The problem NP-hard, and is even NP-hard to approximate to within a factor of $n^{1-\epsilon}$ for any constant $\epsilon > 0$. (Note that an n-approximation is trivial — just take any vertex.) Thus worst-case analysis offers no guidance as to how to solve the problem, and it makes sense to think about the exact recovery of the maximum clique under assumptions, such as with a generative model with a planted solution.

Again, just to get calibrated, let's start by looking at Erdös-Renyi random graphs. In a random graph in the $\mathcal{G}(n, \frac{1}{2})$ model, the size of the maximum clique is very likely to be $\approx 2\log_2 n$.[14] To see heuristically why this should be true, note that for an integer k, the expected number of cliques on k vertices in a random graph of $\mathcal{G}(n, \frac{1}{2})$ is exactly

$$\binom{n}{k} 2^{-\binom{k}{2}} \approx n^k 2^{-k^2/2},$$

which is 1 precisely when $k = 2\log_2 n$. That is, $2\log_2 n$ is roughly the largest k for which we expect to see at least one k-clique.

On the other hand, there is no known polynomial-time algorithm that computes, with high probability, a clique significantly larger than $\approx \log_2 n$ in a random graph from $\mathcal{G}(n, \frac{1}{2})$. And trivial heuristics — like starting with an arbitrary vertex and repeatedly adding an arbitrary vertex that is adjacent to everything already chosen — already obtain the $\log_2 n$ bound with high probability [11]. Thus the Erdös-Renyi model fails to distinguish between different efficient heuristics for the Maximum Clique problem.

Jerrum [9] suggested the following planted version of the maximum clique problem. Fix a vertex set V with n vertices and a value for the parameter k.

1. Choose a random subset $Q \subseteq V$ of k vertices.

2. Add an edge between each pair of vertices in Q.

3. Independently for each pair (i, j) of vertices with at least one of i, j not in Q, include the edge (i, j) with probability $\frac{1}{2}$.

The difficulty of recovering the planted clique Q clearly depends on how big k is. When k is less than $2\log_2 n$, the planted clique will likely not even be a maximum clique, and will be impossible to recover with reasonable probability (even given unbounded computational power). If k is barely less than n, then the problem is easy to solve (e.g., using brute-force

[14] A canonical application of the "second moment method" [2] shows that this random variable is unbelievably concentrated: there is an integer $k \approx 2\log_2 n$ such that almost every n-vertex graph has maximum clique size either k or $k + 1$.

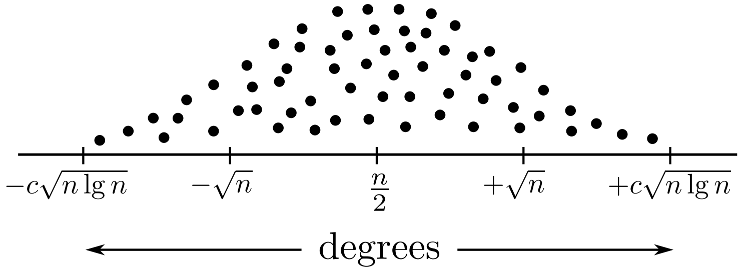

Figure 1: Degree distribution for $\mathcal{G}(n, 1/2)$, before planting the clique. The Hoeffding and union bounds imply that the spread is $O(\sqrt{n \lg n})$ with high probability. If $k = \Omega(\sqrt{n \lg n})$ then the planted clique will consist of the k vertices with the highest degrees.

search to identify the vertices not in Q). So the key question in this model is: how big does the planted clique size k need to be before exact recovery is possible in polynomial time (with high probability)? In contrast to the planted bisection problem, here this question is still open! We'll conclude this lecture by covering the state-of-the-art: polynomial-time exact recovery for $k = \Omega(\sqrt{n})$.

10.2 An Easy Positive Result

Recall the algorithmic approaches discussed in Section 2.3: combinatorial, spectral, and SDP-based (ordered from simplest and least powerful to most complex and powerful). For "overly big" values of the parameter k, the planted clique problem is "too easy," meaning that it can be solved by algorithms that are too naive to do well in practice. Most notably, Kucera [12] observed that the planted clique problem is easy when $k = \Omega(\sqrt{n \log n})$. To see this, think about generating a random instance of the planted clique problem in the following way: first sample a random graph from the usual Erdös-Renyi $\mathcal{G}(n, \frac{1}{2})$ model; then choose k vertices at random and "fill them in" to make them a clique. After the first step, the expected degree of each vertex is $(n-1)/2$, and with high probability, *all* vertex degrees are $\frac{n}{2} \pm c\sqrt{n \log n}$ for a suitable constant c (Figure 1). Filling in the k-clique boosts the degree of those k vertices by roughly $k/2$ each, without affecting the degrees of vertices outside the clique—so in Figure 1, the clique vertices all catapult $k/2$ positions to the right. Thus, if $k > 4c\sqrt{n \log n}$, the clique vertices are the k vertices of the graph with the largest degrees (with high probability); the clique is then obviously recoverable in linear time. For further details, see Homework #5.

11 Planted Clique: A Spectral Algorithm

To force ourselves to design better algorithms that are more likely to have robustly good performance in practice, let's move the goal posts and shoot for efficient exact recovery of the planted clique when $k = \Theta(\sqrt{n})$. As we'll see, even this modest reduction (from $\Theta(\sqrt{n \log n})$ to $\Theta(\sqrt{n})$) seems to require more sophisticated algorithms.

Amazingly, we'll use the same initial two steps of the canonical spectral algorithm (Section 2.4) to solve the planted clique problem as we did for the planted bisection problem. Thus, even though these are quite different problems, the second eigenvector of the adjacency matrix in some sense captures both of them! We will use a different third step and postprocessing step, however, which are tailored to the planted clique problem.

The Planted Clique Algorithm

1. Let $\mathbf{M}$ denote the adjacency matrix of the given graph $G = (V, E)$—the $V \times V$ matrix with $M_{ij} = 1$ if and only if $(i, j) \in E$.
2. Compute the second eigenvector $\mathbf{v}_2$ of $\mathbf{M}$ (indexed by V).
3. Let $A \subseteq V$ denote the vertices that correspond to the k coordinates of $\mathbf{v}_2$ that have the largest magnitudes.
4. Return
$$B = \{i \in V \ : \ i \text{ has at least } \tfrac{3}{4}k \text{ neighbors in } A\}.$$

The first three steps are analogous to the canonical spectral algorithm in Section 2.4. The fourth step is a simpler version of the "voting-based postprocessing" used in Section 9, and again serves the purpose of turning a mostly correct solution into an exactly correct one.

This algorithm has the following guarantee.

Theorem 11.1 ([1]) *When $k \geq c_4\sqrt{n}$ for a sufficiently large constant $c_4 > 0$, with high probability, the spectral algorithm above recovers the planted clique.*

Spectral algorithms are not as popular in practice for clique-finding as they are for graph partitioning, but still, Theorem 11.1 is definitely a step in the right direction.

Improving the upper bound in Theorem 11.1 is a major open question—no better guarantees are known, so you are learning the state-of-the-art![15] The bound in Theorem 11.1 can be improved from $k \geq c_4\sqrt{n}$ for some constant c_4 to $k \geq \epsilon\sqrt{n}$ for every constant ϵ (at the expense of running time exponential in $\frac{1}{\epsilon}$); see Homework #5.

[15] The computational complexity of recovering the planted clique for $k = o(\sqrt{n})$ is highly unclear. As long as $k \geq c \log n$ for a sufficiently large constant c, the problem can be solved in quasi-polynomial (i.e., $n^{O(\log n)}$) time by brute-force search, with high probability. (This is clear when $k = \Theta(\log n)$, but is also true for larger k—why?) Many have conjectured that it is a hard problem for $k = o(\sqrt{n})$. Showing this via the theory of NP-completeness doesn't seem possible, since planted clique is fundamentally an average-case problem, while NP-completeness is for worst-case complexity. (There is a theory of average-case complexity (e.g. [4]), but it has not proven relevant for the planted clique problem.) The strongest evidence of intractability thus far is unconditional lower bounds for concrete but powerful models of computation, such as the sum-of-squares hierarchy [3].

Indeed, intractability of the planted clique problem with $k = o(\sqrt{n})$ is increasingly being used as a novel hardness assumption. There is even a (theoretical) cryptosystem based on it [10]. For example, the problem is used in [7] to identify a barrier to finding certain approximate Nash equilibria in two-player games.

12 Planted Clique: Analysis

In this final section, we give a proof sketch of Theorem 11.1. Happily, we can just reuse the exact same analysis template that worked so well for the planted bisection problem (Sections 2–9).

The first step is to examine the "expected adjacency matrix" $\widehat{\mathbf{M}}$ and compute its eigenvectors and eigenvalues. For the planted clique problem, with edge density $\frac{1}{2}$ outside the clique, we have

$$\widehat{\mathbf{M}} = \left(\begin{array}{cccc|cccc} 1 & 1 & \cdots & 1 & \frac{1}{2} & \frac{1}{2} & \cdots & \frac{1}{2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & \cdots & 1 & \frac{1}{2} & \frac{1}{2} & \cdots & \frac{1}{2} \\ \hline \frac{1}{2} & \frac{1}{2} & \cdots & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \cdots & \frac{1}{2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{2} & \frac{1}{2} & \cdots & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \cdots & \frac{1}{2} \end{array} \right), \tag{14}$$

where we have permuted the rows and columns so that the vertices of the planted clique Q come first. (So don't be misled by the picture: if $k = \Theta(\sqrt{n})$, then the left and top "halves" of the matrix contain far less than half of the columns and rows.) This matrix $\widehat{\mathbf{M}}$ differs from the expected adjacency matrix in that it has 1s and $\frac{1}{2}$s (instead of 0s) on the diagonal; but this difference has little effect on the eigenvectors or eigenvalues of the matrix, so we'll ignore this discrepancy. Note that $\widehat{\mathbf{M}}$ has only two different types of columns, so its rank is (at most) 2, and $n-2$ of its eigenvalues are zero. What about the others?

Despite its simplicity, it's fairly annoying to exactly compute the two non-zero eigenvalues of $\widehat{\mathbf{M}}$ and the corresponding eigenvectors. Instead, we'll identify two simple vectors that act as "approximate eigenvectors" in some sense; this will be sufficient to get the proof approach from Sections 2–9 to work.

First, let's try the same first vector as for planted bisection, the all-ones vector $\mathbf{y} = (1, 1, \ldots, 1)$. Then

$$\widehat{\mathbf{M}}\mathbf{y} = (\underbrace{k + \tfrac{n-k}{2}, \ldots, k + \tfrac{n-k}{2}}_{k \text{ times}}, \underbrace{\tfrac{n}{2}, \ldots, \tfrac{n}{2}}_{n-k \text{ times}}).$$

Thus $\mathbf{y}$ is not actually an eigenvector. But since $k = \Theta(\sqrt{n})$ is much less than n, the first k coordinates are not too far from $\frac{n}{2}$. So let's deem $\mathbf{y}$ an "approximate eigenvector with approximate eigenvalue $\frac{n}{2}$."

Next we identify a second vector, orthogonal to $\mathbf{y}$, that can act as another approximate eigenvector. Being orthogonal to $\mathbf{y}$ means that the entries of the vector $\mathbf{z}$ should sum to 0. Thus we'll need both positive and negative entries. The simplest solution would be to have all of the positive coordinates have the same value, and similarly for all of the negative coordinates. How should we choose which vertices are positive or negative? According to whether or not they're in the planted clique Q, of course.

Formally, define

$$\mathbf{z} = (\underbrace{n-k, \ldots, n-k}_{k \text{ times}}, \underbrace{-k, \ldots, -k}_{n-k \text{ times}}).$$

Then $\mathbf{z}$ is orthogonal to $\mathbf{y}$, and

$$\widehat{\mathbf{M}}\mathbf{z} = (\underbrace{\tfrac{1}{2}(n-k)k, \ldots, \tfrac{1}{2}(n-k)k}_{k \text{ times}}, \underbrace{0, \ldots, 0}_{n-k \text{ times}}).$$

Stretching the truth a bit further, let's deem $\mathbf{z}$ an "approximate eigenvector with eigenvalue $\frac{k}{2}$." Notice that the planted clique can be immediately read off of $\mathbf{z}$, as the vertices corresponding to the k coordinates of $\mathbf{z}$ with the largest magnitudes.

For the rest of the analysis, to keep things simple, we'll cheat and act as if $\mathbf{y}$ and $\mathbf{z}$ really are eigenvectors of $\widehat{\mathbf{M}}$, with eigenvalues $\frac{n}{2}$ and $\frac{k}{2}$. There are two ways to extend the analysis to make it legitimate. The first way is to compute the actual non-zero eigenvalues and corresponding eigenvectors of $\widehat{\mathbf{M}}$ and run the following analysis on them, instead.[16] The second way is to rework the matrix perturbation theory from Section 6 so that it continues to work for "approximate eigenvectors;" this is more or less what is done in the paper [1].

So, assume that $\mathbf{u}_1 = \frac{\mathbf{y}}{\|\mathbf{y}\|}$ and $\mathbf{u}_2 = \frac{\mathbf{z}}{\|\mathbf{z}\|}$ are the first and second eigenvectors of $\widehat{\mathbf{M}}$, with eigenvalues $\frac{n}{2}$ and $\frac{k}{2}$. (Recall that $n > k > 0$ and that the other $n-2$ eigenvalues are 0.) As in our planted bisection analysis, since it is the second eigenvector $\mathbf{u}_2$ that contains the "signal" about the planted clique, we care about the "buffer zone" around the corresponding eigenvalue. For $k = o(n)$, we have

$$\min_{j \neq 2} |\lambda_2 - \lambda_j| = \tfrac{k}{2}. \tag{15}$$

(Larger values of k are irrelevant since in that case we can just use the "top k degrees" algorithm from Section 10.)

With an eye toward the Davis-Kahan theorem (Theorem 6.1), we express the (random) adjacency matrix $\mathbf{M}$ of the input graph as the sum of the "base matrix" $\widehat{\mathbf{M}}$ and the "perturbation matrix" $\mathbf{R}$, where

$$\mathbf{R} = \left(\begin{array}{cccc|cccc}
0 & 0 & \cdots & 0 & \pm\frac{1}{2} & \pm\frac{1}{2} & \cdots & \pm\frac{1}{2} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & 0 & \pm\frac{1}{2} & \pm\frac{1}{2} & \cdots & \pm\frac{1}{2} \\
\hline
\pm\frac{1}{2} & \pm\frac{1}{2} & \cdots & \pm\frac{1}{2} & \pm\frac{1}{2} & \pm\frac{1}{2} & \cdots & \pm\frac{1}{2} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\pm\frac{1}{2} & \pm\frac{1}{2} & \cdots & \pm\frac{1}{2} & \pm\frac{1}{2} & \pm\frac{1}{2} & \cdots & \pm\frac{1}{2}
\end{array}\right).$$

After ignoring minor discrepancies on the diagonal, we have $\mathbf{M} = \widehat{\mathbf{M}} + \mathbf{R}$. The matrix $\mathbf{R}$ satisfies the assumptions of Theorem 5.2, but the coarse upper bound of that theorem only allows us to prove polynomial-time exact recovery for $k = \Omega(\sqrt{n \log n})$, a regime where the trivial "top k degrees" algorithm already works. To get the more interesting recovery result with $k = \Theta(\sqrt{n})$, we need a better bound on the eigenvalues of random symmetric matrices. Theorem 5.1 would certainly be good enough; so is the following bound, whose proof is outlined on Homework #5.

[16] This is messy but can be done, with a closed-form formula for the eigenvectors as a function of n and k. See also Homework #5.

Theorem 12.1 *Under the assumptions of Theorem 5.1, there is a constant $c_5 > 0$ such that, with probability approaching 1 as $n \to \infty$,*

$$\|\mathbf{P}\| \le c_5\sqrt{n}. \tag{16}$$

Let $\mathbf{v}_2$ denote the second eigenvector of $\mathbf{M}$. Plugging in (15) and (16) into the numerator and denominator of the bound (6) of the Davis-Kahan theorem, and using (7), we have (w.h.p.)

$$\|\mathbf{u}_2 - \mathbf{v}_2\| \le \sqrt{2} \cdot \frac{2c_5\sqrt{n}}{k/2} \le 4\sqrt{2} \cdot \frac{c_5}{c_4}, \tag{17}$$

since by assumption $k \ge c_4\sqrt{n}$.

It remains to connect the number of errors made in the third step of the spectral algorithm with the eigenvector $\mathbf{v}_2$ to the Euclidean distance between $\mathbf{u}_2$ and $\mathbf{v}_2$. Back in $\mathbf{z}$, the k coordinates corresponding to the clique vertices have value $n-k$ and the rest have value $-k$. The (approximate) eigenvector $\mathbf{u}_2$ is the same, except with all entries scaled down by $\|z\| = \sqrt{(n-k)nk}$.

Now suppose that there are ℓ vertices in A—recall A is the set of vertices with the k largest magnitudes in $\mathbf{v}_2$—that do not belong to the planted clique Q. How did this happen? All of the magnitudes of the ℓ vertices in $A \setminus Q$ must exceed all of the magnitudes of the ℓ vertices in $Q \setminus A$ (the excluded clique vertices). But in $\mathbf{u}_2$, there is a gap of $\frac{n-2k}{\sqrt{(n-k)nk}}$ between the magnitudes of clique and non-clique vertices. In particular, for at least one of the sets $A \setminus Q$ or $Q \setminus A$, every vertex in the set changed in magnitude by at least $\frac{n-2k}{2\sqrt{(n-k)nk}}$ (between $\mathbf{u}_2$ and $\mathbf{v}_2$). For if not, there would be vertices $v \in A \setminus Q$ and $w \in Q \setminus A$ such that v's magnitude in $\mathbf{v}_2$ is less than w's (contradicting that $v \in A$ while $w \notin A$).

The upshot is that if there are ℓ clique vertices missing from the set A computed by the algorithm, then

$$\|\mathbf{u}_2 - \mathbf{v}_2\| \ge \sqrt{\ell} \cdot \frac{n-2k}{2\sqrt{(n-k)nk}}$$

which, assuming that n is sufficiently large and using that $k = o(n)$, implies that

$$\|\mathbf{u}_2 - \mathbf{v}_2\| \ge \frac{\sqrt{\ell}}{3\sqrt{k}}. \tag{18}$$

Combining (17) and (18), we obtain that

$$\ell \le 9k \cdot \|\mathbf{u}_2 - \mathbf{v}_2\|^2 \le k \cdot 288 \cdot \frac{c_5^2}{c_4^2}$$

holds with high probability. This bound is at most $\frac{k}{6}$ provided we choose $c_4 \ge 17c_5$.[17]

[17] As mentioned earlier, an additional idea can reduce this value of c_4 to an arbitrarily small constant ϵ, at the expense of running time exponential in $\frac{1}{\epsilon}$ (Homework #10).

Thus, with high probability, the set A computed by our spectral algorithm is "mostly correct," in that it includes at least a $\frac{5}{6}$ fraction of the vertices in the planted clique. In this case, every clique vertex will have at least $\frac{5}{6}k$ neighbors in A. We expect non-clique vertices to have only $\approx \frac{k}{2}$ neighbors in A (see Homework #5 for the formal argument). Thus in the final step of the algorithm, the set $B = \{i \in V \, : \, i \text{ has at least } \frac{3}{4}k \text{ neighbors in } A\}$ will contain precisely the vertices of the planted clique Q (with high probability).

References

[1] N. Alon, M. Krivelevich, and B. Sudakov. Finding a large hiddin clique in a random graph. *Random Structures & Algorithms*, 13(3-4):457–466, 1998.

[2] N. Alon and J. H. Spencer. *The Probabilistic Method.* Wiley, 2008. Third edition.

[3] B. Barak, S. B. Hopkins, J. A. Kelner, P. Kothari, A. Moitra, and A. Potechin. A nearly tight sum-of-squares lower bound for the planted clique problem. In *Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 428–437, 2016.

[4] A. Bogdanov and L. Trevisan. Average-case complexity. *Foundations and Trends in Theoretical Computer Science*, 2(1):1–106, 2006.

[5] T. N. Bui, S. Chaudhuri, F. T. Leighton, and M. Sipser. Graph bisection algorithms with good average case behavior. *Combinatorica*, 7(2):171–191, 1987. Preliminary version in *FOCS '84*.

[6] C. Davis and W. M. Kahan. The rotation of eigenvectors by a pertubation. III. *Journal of Numerical Analysis*, 7:1–46, 1970.

[7] E. Hazan and R. Krauthgamer. How hard is it to approximate the best Nash equilibrium? In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 720–727, 2009.

[8] P. W. Holland, K. Lasket, and S. Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, 1983.

[9] M. Jerrum. Large cliques elude the metropolis process. *Random Structures & Algorithms*, 3(4):347–360, 1992.

[10] A. Juels and M. Peinado. Hiding cliques for cryptographic security. *Designs, Codes, and Cryptography*, 20(3):269–280, 2000.

[11] R. M. Karp. The probabilistic analysis of some combinatorial search algorithms. In J. F. Traub, editor, *Algorithms andComplexity: New Directions and Recent Results*, pages 1–19. Academic Press, 1976.

[12] L. Kucera. Expected complexity of graph partitioning problems. *Discrete Applied Mathematics*, 57(2-3):193–212, 1995.

[13] L. Massoulie. Community detection thresholds and the weak Ramanujan property. arXiv:1311.3085, 2013.

[14] F. McSherry. Spectral partitioning of random graphs. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 529–537, 2001.

[15] E. Mossel, J. Neeman, and A. Sly. Stochastic block models and reconstruction. arXiv:1202.1499, 2012.

[16] E. Mossel, J. Neeman, and A. Sly. A proof of the block model threshold conjecture. arXiv:1311.4115, 2013.

[17] D. A. Spielman. Lecture notes on spectral graph theory. Yale University, 2015.

[18] V. Vu. Spectral norm of random matrices. *Combinatorica*, 27(6):721–736, 2007.

[19] E. P. Wigner. On the distribution of the roots of certain symmetric matrices. *Annals of Mathematics*, 67(2):325–327, 1958.