CS369N: Beyond Worst-Case Analysis
Lecture #4: Probabilistic and Semirandom Models for Clustering and Graph Partitioning*

Tim Roughgarden†

April 25, 2010

1 Learning Mixtures of Gaussians

This lecture continues last week's theme of somewhat specific data models to inform the analysis of clustering and graph partitioning heuristics. We begin with the classical notion of a *mixture model*.

1.1 Learning Mixtures of Gaussians

We consider k distributions $D_1, D_2, \ldots, D_k$ on $\mathcal{R}^n$. Suppose that D_i has a *mixing weight* w_i, where the mixing weights are nonnegative and sum to 1. We consider the following 2-stage sampling procedure (see Figure 1): first, we pick a distribution D_i randomly according to the mixing weights; then we pick a random point $x \in \mathcal{R}^n$ according to D_i.

To illustrate the flavor of this line of research, we focus on the relatively easy case in which each D_i is a *spherical Gaussian*. In other words, each D_i is characterized by a mean $\mu_i \in \mathcal{R}^n$ and a common standard deviation $\sigma_i \in \mathcal{R}$ in every direction. (So the covariance matrix is σ_i times the identity.) Precisely, the distribution of such a spherical Gaussian D_i is given by the density function

$$\prod_{j=1}^{n} \left[\frac{1}{\sqrt{2\pi}\sigma_i} e^{-(x_j - \mu_{ij})^2/2\sigma_i^2} \right],$$

where x_j denotes the jth coordinate of the sample point and μ_{ij} denotes the jth coordinate of the mean μ_i of D_i.

*©2009–2010, Tim Roughgarden. Thanks to Ankur Moitra and Greg Valiant for helpful comments on an earlier draft.

†Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

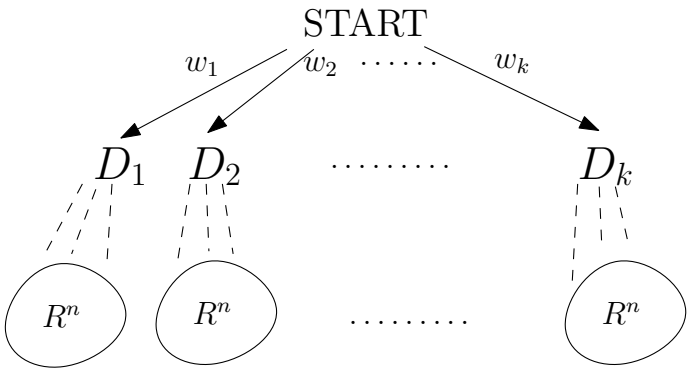

Figure 1: Sampling from a mixture model via a two-stage procedure.

Our goal is the following: given sufficiently many independent samples from a mixture of spherical Gaussians, "learn the model" (The D_i's are unknown a priori; we think of k as known.) Now, "learning the model" can mean several different things. In this lecture, we strive for the following: with high probability over the sample, we want to label the sample points so that two points x, y have the same label if and only if they come from the same distribution D_i.

This is a strong goal. In a sense, for almost all samples we are recovering a target clustering of sorts, where the ith cluster is the sample points that were drawn from D_i. If we can accomplish this goal, then it's easy to reverse engineer good estimates of all of the model parameters (the μ_i's and the σ_i's). For example, our estimate of μ_i would just be the center of mass of the samples that we labeled i.[1]

This strong goal is obviously impossible without further assumptions. For example, if two of the D_i's are identical, there's no way to label correctly the sample points that they generate with high probability. More generally, significant overlap between two distributions leads to inherent ambiguity and precludes a solution to the labeling problem above. Thus our goal is: assuming that the distributions D_i are sufficiently separated, correctly label a sufficiently large (but polynomial size) sample with high probability.[2]

While learning mixture models is an old idea, the version above was only identified 10 years ago by Dasgupta [12]. Our focus will be on a very special case of the subsequent work by Arora and Kannan [4]. Many other papers on the topic have followed (see Section 1.4 for some pointers).

[1] One could try to use maximum likelihood estimation directly on the mixture, rather than solving the labeling problem as an intermediate step. But unlike estimating one Gaussian (which is easy), it is not clear how to solve directly for the maximum likelihood mixture of Gaussians.

[2] Recent work [17] shows how to bypass the labeling step and directly estimate the μ_i's and the σ_i's, without any separation assumptions.

1.2 Counterintuitive Facts about High-Dimensional Gaussians

The discussion in this section will be heuristic for simplicity, but it can be made rigorous using elementary (if sometimes tedious) arguments. The Arora-Kannan algorithm is motivated by some counterintuitive and useful properties of high-dimensional distributions. Basically, the Law of Large Numbers has some pretty amazing geometric interpretations.

Counterintuitive Fact #1: *For a large number n of dimensions, a spherical Gaussian is essentially the uniform distribution on a sphere of appropriate radius.*

For example, consider the case of an n-vector x, where each component is an independent draw from the standard 1-dimensional Gaussian $\mathcal{N}(0,1)$. By definition $\mathbf{E}[x_i^2] = \mathbf{Var}[x_i] = 1$. By linearity of expectation, $\mathbf{E}\left[\|x\|^2\right] = n$. (In this lecture, the norm $\|\cdot\|$ always denotes the Euclidean norm.) Moreover, there is sharp concentration around this value for large n — even better than a Chernoff-type bound would suggest, since Gaussians already have quickly vanishing tails. Thus x is very likely to have norm very close to $\sqrt{n}$. For a general spherical Gaussian with mean $\mu \in \mathcal{R}^n$ and directional variance $\sigma^2 \in \mathcal{R}$, $\|x - \mu\|$ is very likely to be very close to $\sqrt{n} \cdot \sigma$.

Counterintuitive Fact #2: *For a large number n of dimensions, almost all vectors are orthogonal to each other.*

What we really mean is that the Pythagorean theorem approximately holds for two random vectors x, y from a spherical Gaussian with mean 0, with high probability. To prove this in the unit-variance case, by spherical symmetry and the first fact we can assume that $y = (1, 1, \ldots, 1)$ is the all ones vector and consider a random x. Recall that

$$\|x + y\|^2 = \underbrace{\|x\|^2}_{\approx n \text{ w.h.p.}} + \underbrace{\|y\|^2}_{=n} + 2\langle x, y\rangle.$$

Since $y = (1, 1, \ldots, 1)$,

$$\mathbf{E}[|\langle x, y\rangle|] = \mathbf{E}\left[\left|\sum_{i=1}^{n} x_i\right|\right];$$

the right-hand side is the standard deviation of a univariate Gaussian with mean 0 and variance n, also known as $\sqrt{n}$. Again, there is a quite sharp concentration around this value.

In general, if x, y are random vectors from a common spherical Gaussian with mean μ and directional variance σ^2, the above argument implies that

$$\|x - y\|^2 \approx \underbrace{\|x - \mu\|^2}_{\approx n\sigma^2} + \underbrace{\|\mu - y\|^2}_{\approx n\sigma^2} + O(\sqrt{n}\sigma^2)$$

with high probability.

1.3 The Arora-Kannan Algorithm

The Arora-Kannan algorithm and the separation condition it requires are directly motivated by the above two facts. First, a thought experiment. Define R_i as $\sqrt{n}\sigma_i$, where σ_i^2 is the directional variance of the ith Gaussian D_i. If $x, y \sim D_i$ are independent and the mean of D_i is μ_i, then our two counterintuitive facts imply that

$$\|x - y\|^2 \approx \|x - \mu_i\|^2 + \|\mu_i - y\|^2 \approx 2R_i^2, \tag{1}$$

up to an error term of $O(R_i^2/\sqrt{n})$, with high probability.

On the other hand, if $x \sim D_i$ and $y \sim D_j$ with $i \neq j$, then

$$\begin{aligned} \|x - y\|^2 &\approx \|x - \mu_i\|^2 + \|\mu_i - \mu_j\|^2 + \|\mu_j - y\|^2 \\ &\approx R_i^2 + R_j^2 + \|\mu_i - \mu_j\|^2, \end{aligned} \tag{2}$$

up to an error term of $O((R_i^2 + R_j^2 + \|\mu_i - \mu_j\|^2)/\sqrt{n})$, with high probability.

In words: all sample points from a common D_i will be roughly equidistant from each other, while sample points from different D_i's are at least some distance away. Inequalities (1) and (2) motivate the AK separation condition.

The Separation Condition. For every distinct i, j,

$$\|\mu_i - \mu_j\| \geq c \cdot \frac{\max\{R_i, R_j\}}{n^{1/4}}, \tag{3}$$

where c is a sufficiently large constant.[3] To interpret the right-hand side of (3), recall that most sample points from D_i are roughly $R_i = \sqrt{n}\sigma_i$ distance from the corresponding mean μ_i. The gap between the mean of cluster i and those of all other clusters needs to grow like $n^{1/4}\sigma_i$ for the following algorithm to work. Notice that this allows the spheres corresponding to different Gaussians to overlap. (This does not preclude reverse engineering labels for all sample points with high probability, because the amount of overlap between different distributions will be exponentially small.)

To motivate condition (3), consider the Gaussian D_i with the smallest radius R_i. Suppose that x, y are sample points from D_i and z is a sample point from D_j with $j \neq i$. Inequalities (1) and (2) imply that, with high probability, there is a noticeable gap between the squared distances $\|x - y\|^2$ and $\|x - z\|^2$ provided the mean separation $\|\mu_i - \mu_j\|$ is sufficiently large:

$$\|x - z\|^2 - \|x - y\|^2 \geq \|\mu_i - \mu_j\|^2 + (R_j^2 - R_i^2) + \Omega((R_i^2 + R_j^2 + \|\mu_i - \mu_j\|^2)/\sqrt{n}). \tag{4}$$

Now, R_j is at least R_i by our choice of i, but for all we know the two radii are equal. To ensure that (4) is positive and noticeably bounded away from zero we require that

$$\|\mu_i - \mu_j\|^2 = \Omega((R_i^2 + R_j^2 + \|\mu_i - \mu_j\|^2)/\sqrt{n}),$$

[3] Actually, this "constant" needs to be logarithmic in the eventual (polynomial) sample size, to achieve the desired concentration in (1) and (2) for all pairwise distances between the sample points. We ignore these logarithmic factors in these notes.

and the separation condition guarantees precisely this, for a suitably large choice of the constant c. The Arora-Kannan algorithm is now the natural one (Figure 2).

---

1. Take enough samples S so that one has at least a few from each distribution D_i. (Assume that the smallest mixing weight $\min_i w_i$ is known and at least inverse polynomial in n.)
2. Let x_0, y_0 be the sample points that minimize $\|y_0 - x_0\|$.
3. Peel off all the sample points in the ball around x_0 with radius $\|y_0 - x_0\| \cdot (1 + \frac{c'}{\sqrt{n}})$, where c' is a suitable constant derived from that in the separation condition (3).
4. Return to step 2 with the remaining sample points.

Figure 2: The Arora-Kannan algorithm.

Thus the Arora-Kannan algorithm uses the approximate identities (1) and (2) to identify all of the sample points that belong to the Gaussian with the smallest radius, and then recurses on the remaining points. Correctness follows because (1) implies that the pair x_0, y_0 will indeed belong to the Gaussian with the smallest radius (or at least almost the smallest), and also that all other points from this Gaussian have essentially this same distance from x_0; and because the separation condition is chosen large enough that no sample points from other Gaussians will be equally close to x_0.[4]

1.4 Extensions

The papers by Dasgupta [12] and Arora and Kannan [4] motivated a lot of followup work; see [9] for a survey. Some examples include the improvement of the separation condition from $\max\{R_i, R_j\}/n^{1/4}$ to $\max\{R_i, R_j\}/\sqrt{n}$, which turns out to be the best possible if one wants the "no inherent ambiguity" condition that enables the unique recovery of the sample points' labels [22]; polynomial-time recoverability results for non-spherical Gaussians, logconcave distributions, and more general distributions [4, 10, 11]; guarantees for spectral algorithms, which appear more likely to be robust (with good performance even when the data does not conform well to a mixture model) [1, 7, 18]; and other notions of "learning a mixture of Gaussians" [14, 17].

[4] A more reasonable interpretation of Step 3 is that the constant c' in the algorithm is chosen based on empirical performance, and this choice restricts the mixture models (according to the induced value of c in (3)) to which the theoretical analysis applies.

2 Probabilistic Planted Models

This section and the next study models of data for graph problems. Thus far, the only such example we've seen is the Bilu-Linial definition of γ-stable Max Cut instances. Here we take a more general approach.

2.1 Erdös-Renyi Random Graphs

We review *Erdös-Renyi random graphs* as a starting point. Recall that a random graph from $\mathcal{G}(n,p)$ is a simple undirected graph on n nodes, where each of the $\binom{n}{2}$ possible edges is present independently with probability p. There are at least two reasons why this model is usually not very useful for informing the design of graph algorithms. First, as with most average-case analysis frameworks, the data model is too specific; second, as we show below, it fails to meaningfully differentiate between different algorithms.

Example 2.1 (Minimum Bisection) The *graph bisection* 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 $(S, \bar{S})$ with $|S| = |\bar{S}|$ that has the fewest number of crossing edges.

Assume for simplicity that the edge probability p is a constant. Then for every bisection $(S, \bar{S})$ of a set of n nodes, 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 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!

Example 2.2 (Maximum Clique) In the *maximum clique* problem, the goal (given an undirected graph) is to identify the largest subset of vertices that are mutually adjacent. In a random graph in the $\mathcal{G}(n, \frac{1}{2})$ model, the size of the maximum clique is very likely to be $\approx 2\log_2 n$.[5] To see heuristically why this is true, note 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

[5] A canonical application of the "second moment method" [3] shows that this random variable is unbelievably concentrated: there as an integer $k \approx 2\log_2 n$ such that almost every graph has maximum clique either k or $k+1$.

arbitrary vertex that is adjacent to everything already chosen — already obtain the $\log_2 n$ bound with high probability. (Exercise: prove this.) Thus the Erdös-Renyi model fails to distinguish between different efficient heuristics for the Maximum Clique problem.[6]

2.2 Planted Random Graph Models

Recall our motivation for the Bilu-Linial stability definition and the Balcan-Blum-Gupta isolation definition: we may only be 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 family of probabilistic models in which such a "clearly optimal" solution is guaranteed to exist (with high probability). We follow the elegant formalism of McSherry [21].

The Model. There are n vertices, partitioned into t clusters $S_1, \ldots, S_t$. For each pair i, j of clusters (possibly with $i = j$) there is a parameter $p_{ij} \in (0, 1)$, with $p_{ij} = p_{ji}$ for every i, j. A random graph is sampled by independently including each edge with endpoints in S_i, S_j with probability p_{ij}.

The Goal. Given a random graph from the above model, we strive to reverse engineer the S_i's with high probability. Note that there needs to be nontrivial separation between certain p_{ij}'s; otherwise this goal is impossible due to inherent ambiguity in the sample.

Our first example is a planted version of the minimum bisection problem (recall Example 2.1), and it was originally proposed by Bui et al. [8]. There were several follow-up papers in the 1980s.

Example 2.3 (Planted Bisection) There are two clusters ($t = 2$) satisfying $|S_1| = |S_2| = n/2$. The parameters p_{11}, p_{22} are equal, say to p. The parameter p_{12}, denoted q, is less than p. The question is then: for how small a gap $p - q$ is polynomial-time recovery of the planted bisection possible?

We return to the graph bisection problem when we discuss semirandom models below.

Our second example is a planted version of the maximum clique problem (recall Example 2.2) and was first suggested by Karp [19].

Example 2.4 (Planted Clique) There are again two clusters, with $|S_1| = k$ and $|S_2| = n - k$. We set $p_{11} = 1$ so that S_1 is a k-clique. The parameters p_{12} and p_{22} are set to a common value p. The question is then: for a given p ($\frac{1}{2}$, say), how big does k need to be before the planted clique can be recovered in polynomial time?

[6] The computational complexity of finding a maximum clique in a random graph is highly unclear. Note that the problem can be solved in quasi-polynomial (i.e., $n^{O(\log n)}$) time by brute-force search, with high probability. It is conjectured to be a hard problem, however, and there is even a (theoretical) cryptosystem based on it [16].

The planted clique problem is clearly easy when k is really huge, like $k = n - O(1)$. Kucera [20] observed that it is easy (for $p = \frac{1}{2}$) even when $k = \Omega(\sqrt{n \log n})$. To see this, think about generating a random sample of the planted maximum clique problem in the following way: first take a sample from the usual Erdös-Renyi $\mathcal{G}(n, p)$ 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 node is $(n-1)/2$, and Chernoff bounds imply sharp concentration: with high probability, *all* node degrees are $\frac{n}{2} \pm c\sqrt{n \log n}$ with high probability (for a suitable constant c). Filling in the k-clique boosts the degree of those k nodes by roughly $k/2$ each, without affecting the degrees of nodes outside the clique. Thus if $k > 4c\sqrt{n \log n}$, the clique nodes are the k nodes of the graph with the largest degrees (with high probability); the clique is then obviously recoverable in linear time.

Alon et al. [2] used sophisticated techniques to achieve a slight improvement: polynomial-time recovery of a planted clique with $p = \frac{1}{2}$ and $k = \Omega(\sqrt{n})$. Closing the gap between the upper bound of $k \approx \sqrt{n}$ and the lower bound of $k \approx \log n$ for polynomial-time solvability of the planted clique problem is a major open question.[7]

3 Semirandom Models

Semirandom graph modes are a very nice idea of Blum and Spencer [5]; the results we discuss are from Feige and Kilian [13]. The goal of these models is to have as robust a data model as possible, subject to preserving the existence of a "planted" or "clearly optimal" solution. Like upcoming lectures on pseudorandom data and on smoothed analysis, the idea of the model is to blend together a probabilistic and an adversarial approach.

3.1 Definition and Discussion

We explain the model in particular for the minimum bisection and maximum clique problems. Nature and an adversary conspire to produce a hard input as follows. First, nature chooses a random instance from the corresponding planted model (Examples 2.3 and 2.4). Second, the adversary can make the sparse regions sparser (by removing edges) and the dense regions denser (by adding edges) in an arbitrary way. So in the minimum bisection problem, the adversary can delete any edges across the planted bisection and add any edges that do not cross this bisection. In the maximum clique problem, the adversary can remove any edges that are not inside the planted clique.

An easy but key observation is that the planted solution survives arbitrary actions of the adversary. In fact, since the adversary can intuitively only make the planted solution "more optimal", one might wonder if the semirandom model is really any harder than the planted one. There is no known formal equivalence or separation between any planted problem and its semirandom counterpart. But there are certainly algorithms that work in a planted model

[7] Progress would have implications for other problems — for example, the hardness' of finding certain approximate Nash equilibria in two-player games is currently based on the presumed difficulty of solving the planted clique problem with $k = o(\sqrt{n})$ [15].

but not in the corresponding semirandom one. For example, the "top k degrees" algorithm above that recovers the planted maximum clique with $k = \Omega(\sqrt{n \log n})$ fails miserably in the semirandom model, even when k is linear in n: the adversary can sparsify the edges between the clique nodes and the other nodes, lowering the degrees of the clique nodes to right around $n/2$.

The benefit of the semirandom model over the planted one is that it should encourage more "robust" solutions — algorithms that are not overfitted to a particular data model. In the planted model, an algorithm can take advantage of two properties of the input: (i) there is a clearly optimal solution (a plausible property of many "real instances"); and (ii) the input shares lots of non-trivial and useful properties with completely random instances, such as highly predictable node degrees (a questionable property of "real instances"). The first advantage was the motivating point behind the planted model, while the second was only a side effect of our modeling choices. The semirandom model offers an algorithm only the first advantage, and is thus more faithful to our original goals.

3.2 Case Study: Graph Bisection

Recall the planted bisection problem (Example 2.3). Assume that p, q are constants with

$$p - q \geq c\sqrt{\frac{\log n}{n}}, \tag{5}$$

where c is a sufficiently large constant. Calculations show that this separation is necessary for the intended planted bisection (S_1, S_2) to be the minimum bisection with high probability [8].

Feige and Kilian [13] prove the following cool theorem.

Theorem 3.1 ([13]) *Under assumption* (5)*, there is a polynomial-time algorithm that computes the minimum bisection in the semirandom model, with high probability.*

Their approach is to use semidefinite programming, which was also mentioned in passing last lecture. Given a graph $G = (V, E)$, we define a relaxation (i.e., lower bound) of the minimum bisection problem as follows. We adopt the linear objective function

$$\sum_{(i,j) \in E \,:\, i<j} \frac{1 - x_{ij}}{2}, \tag{6}$$

and maximize it subject to the linear constraints

$$\sum_{i,j \in V} x_{ij} = 0 \tag{7}$$

and

$$x_{ii} = 1 \text{ for all } i \in V, \tag{8}$$

and also the possibly bizarre-sounding constraint that the $V \times V$ matrix X of the x_{ij}'s is symmetric and positive semidefinite.[8] This mathematical program can be solved in polynomial time (up to an arbitrarily small additive error term), using either the ellipsoid method or modern interior-point techniques. A good first-cut rule of thumb is that minimization mathematical programs are polynomial-time solvable if and only if both the objective function and the feasible region are convex. Observe that the set of symmetric and positive semidefinite matrices is indeed convex, so we should not be surprised that this mathematical program is tractable.

But what good is it? Let $b(G)$ denote the number of edges crossing the minimum bisection of G and let $h(G)$ denote the optimal solution to the semidefinite program above. We claim that $h(G) \leq b(G)$. To see this, let (S_1, S_2) be an optimal bisection of G and let $s \in \{\pm 1\}^V$ denote the corresponding characteristic vector. Form the matrix $X = ss^T$, which is clearly symmetric and positive semidefinite (and rank one, even). The constraint (8) on the diagonal of X clearly holds. Since $|S_1| = |S_2|$, the constraint (7) also holds. Finally, observe that the objective function value of this X is precisely $b(G)$. The optimal value $h(G)$ can only be less than this.

Next, we claim that if $\hat{G}$ is obtained from G be adding a single edge, then

$$h(G) \leq h(\hat{G}) \leq h(G) + 1. \tag{9}$$

We leave the verification of (9) as an easy exercise; the key point is that constraint (8) and the positive semidefinite constraint force all x_{ij}'s to have magnitude at most 1.

Since computing a minimum bisection is NP-hard and computing $h(G)$ can be done in polynomial time, we expect that $h(G) < b(G)$ for many graphs G. The main technical lemma in Feige and Kilian [13], which is inspired by Boppana [6], is that *the relaxation is exact in the planted model.*

Lemma 3.2 *For a random graph in the planted (non-semirandom) bisection model, under assumption* (5)*, $h(G) = b(G)$ with high probability.*

The proof of Lemma 3.2 is quite technical and we won't discuss it. It involves "guessing and checking" a solution to the dual semidefinite program that has objective function value $b(G)$.

Unlike previous algorithms for planted models, the semidefinite programming approach extends automatically to the semirandom model (and is therefore a "robust algorithm" in some sense).

Proof of Theorem 3.1: Begin with a random sample G_0 from the planted model. By Lemma 3.2, $h(G_0) = b(G_0)$ with high probability. The adversary adds or deletes edges one at a time, yielding a sequence $G_0, \ldots, G_t$. We claim that, by induction, $b(G_i) = h(G_i)$ for every i.

First, when the adversary adds an edge not in the planted bisection, $b(G_i) = b(G_{i-1})$. By (9), $h(G_i) \geq h(G_{i-1}) = b(G_{i-1}) = b(G_i)$. But since $h(G) \leq b(G)$ for every graph, we must have $b(G_i) = h(G_i)$.

[8] Recall there are many equivalent definitions of such matrices: the (full) set of real eigenvalues are all nonnegative; the quadratic form $y^T X y$ is nonnegative for all y; or X has a "square root" U with $X = UU^T$.

When the adversary removes an edge of the planted bisection, $b(G_i) = b(G_{i-1}) - 1$. By (9), $h(G_i) \geq h(G_{i-1}) - 1 = b(G_{i-1}) - 1 = b(G_i)$. But since $h(G) \leq b(G)$ for every graph, we must have $b(G_i) = h(G_i)$. ■

The proof above shows how to compute the *value* $b(G)$ of the minimum bisection in polynomial time with high probability (by computing $h(G)$ instead). We leave as a non-trivial exercise the task of using this subroutine to reconstruct the planted bisection itself in polynomial time.

References

[1] D. Achlioptas and F. McSherry. On spectral learning of mixtures of distributions. In *Proceedings of the 18th Annual Conference on Learning Theory (COLT)*, pages 458–469, 2005.

[2] 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. Preliminary version in *SODA '98*.

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

[4] S. Arora and R. Kannan. Learning mixtures of separated nonspherical Gaussians. *Annals of Applied Probability*, 15(1A):69–92, 2005. Preliminary version in *STOC '01*.

[5] A. Blum and J. H. Spencer. Coloring random and semi-random k-colorable graphs. *Journal of Algorithms*, 19(2):204–234, 1995.

[6] R. B. Boppana. Eigenvalues and graph bisection: An average-case analysis. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 280–285, 1987.

[7] S. C. Brubaker and S. Vempala. Isotropic PCA and affine-invariant clustering. In *Proceedings of the 49th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 551–560, 2008.

[8] 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*.

[9] K. Chaudhuri. *Learning Mixtures of Distributions*. PhD thesis, U.C. Berkeley, 2007.

[10] K. Chaudhuri and S. Rao. Beyond Gaussians: Spectral methods for learning mixtures of heavy-tailed distributions. In *Proceedings of the 21st Annual Conference on Learning Theory (COLT)*, pages 21–32, 2008.

[11] A. Dasgupta, J. E. Hopcroft, J. M. Kleinberg, and M. Sandler. On learning mixtures of heavy-tailed distributions. In *Proceedings of the 46th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 491–500, 2005.

[12] S. Dasgupta. Learning mixtures of Gaussians. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 634–644, 1999.

[13] U. Feige and J. Kilian. Heuristics for semirandom graph problems. *Journal of Computer and System Sciences*, 63(4):639–671, 2001. Preliminary version in *FOCS '98*.

[14] J. Feldman, R. O'Donell, and R. A. Servedio. Learning mixtures of product distributions over discrete domains. *SIAM Journal on Computing*, 37(5):1536–1564, 2008. Preliminary version in *FOCS '05*.

[15] 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.

[16] A. Juels and M. Peinado. Hiding cliques for cryptographic security. *Designs, Codes, and Cryptography*, 20(3):269–280, 2000. Preliminary version in *SODA '98*.

[17] A. T. Kalai, A. Moitra, and G. Valiant. Efficiently learning mixtures of two Gaussians. In *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing (STOC)*, pages ??–??, 2010.

[18] R. Kannan, H. Salmasian, and S. Vempala. The spectral method for general mixture models. In *Proceedings of the 18th Annual Conference on Learning Theory (COLT)*, pages 444–457, 2005.

[19] 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.

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

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

[22] S. Vempala and G. Wang. A spectral algorithm for learning mixture models. *Journal of Computer and System Sciences*, 68(4):841–860, 2004. Preliminary version in *FOCS '02*.