

Randomized Approximation of MAX-CUT

Russell Impagliazzo
Notes : Fritz Schneider

January 24, 2001

1 Last Class

In the previous class we discussed averaging arguments, derandomization, the method of conditional expectation, and looked at the general technique of relaxation and randomized rounding with respect to MAX-SAT. To solve (approximate) MAX-SAT, we relaxed the requirement that the variables be boolean and instead let them take on real values in the interval $[0, 1]$. We then solved the relaxed problem using a black box linear programming (LP) algorithm and treated the resulting real solutions as probabilities for the variables to be rounded to $\{0, 1\}$.

2 The plan

In this class Russell presents a similar relaxation/rounding technique. He first briefly describes the Ellipsoid Method for solving linear programming problems and then describes a randomized algorithm for approximating the max cut of a graph (Goemans-Williamson). I will cite references at the end of these notes.

Russell's Disclaimer: There will be hand-waving.

Fritz's Disclaimer: Projecting a hyperplane from n -space onto this 2-space is beyond my $\text{\LaTeX}$ ability. Sorry for the lack of diagrams.

3 The Ellipsoid Method for LP

One can think of each constraint in an LP problem as a plane in n -dimensions which partitions space into two regions: one region which does not violate the constraint and the complementary region which does. So the multiple constraints of a LP problem describe a polytope inside of which one wants to find a point. Usually it is desirable that this point also maximize some potential function, but we can reduce this problem to the "find a point" version by asking "Find a solution with potential greater than y " and doing binary search on y .

The approach of the ellipsoid method is to start by drawing a large "sphere" around the area of interest. Then a point near the center of the sphere is picked and checked to see if it lies inside the region of interest. If not, a plane can be constructed through that point which rules out a portion of the "search sphere" (sort of like a binary search in n -dimensions). Note here that we are assuming the existence of an algorithm that given a point not in a region finds a hyperplane through that point that separates it from the region. The algorithm then proceeds to "narrow" the search space by finding an ellipsoid that contains the half-sphere and is smaller than the full sphere. This continues until a satisfactory point can be found or the constraints are found to be inconsistent, (because the remaining space is too small).

This method will be used in the algorithm described shortly, but with a generic convex region. (*Russell's hands start to flutter*).

4 Randomized Approximation of MAX-CUT

4.1 The problem

A *cut* of a graph is of course a partition of the vertices into sets S_0 and S_1 . The size of the cut is the number of edges of the graph having one endpoint in S_0 and the other in S_1 . The traditional MAX-CUT decision problem asks whether a cut of size $\geq k$ exists in a given graph. We address the "harder" problem of finding a *maximal* cut in the graph. (I say "harder" because *finding* a max cut is sufficient to answer the MAX-CUT decision problem). In particular, we are concerned with *randomized* algorithms that *approximate* the maximal cut.

4.2 The trivial case

A trivial algorithm would be to flip coins and assign vertices to S_0 and S_1 at random. The expected value of the cut according to this method is $\frac{1}{2}m$ where m is the number of edges in the graph. This result is obviously $\geq \frac{1}{2}Opt$. **Exercise:** Use the method of conditional expectation (see last class) to devise a deterministic (derandomized) $\frac{1}{2}Opt$ algorithm.

4.3 The general idea

We can think of the problem in terms of a boolean equation: for each vertex i assign the variable $x_i = 0$ if $x \in S_0$ or $x_i = 1$ if $x \in S_1$. Relaxing the problem to real values and using randomized rounding as per last class would seem to result in each variable being assigned probability $\frac{1}{2}$, so the algorithm is reduced to the trivial case above.

So instead of relaxing booleans to real values, we relax the requirements so that the booleans become *vectors* in R^n of unit norm. This results in each vertex being mapped to a point on an n -dimensional unit “sphere.” The idea is to maximize our potential function such that well connected vertices lie “far” apart on this surface. Our aim will then be to partition the sphere with a hyperplane so that the plane “cuts” many connected vertices into two different sets (S_0 and S_1).

But what is meant by “far” apart? Any two distinct vertices on the surface of a sphere describe a circle. We could define the distance between two points to be the (smallest) distance between the two points on such a great circle. Equivalently, we define distance as the *angle* of that arch (with respect to the origin at the center of the sphere).

Note that an equivalent statement can be made for dimension ≥ 3 but 3-space is a natural simplification that is easy to visualize. It turns out that this is a fortuitous relaxation of the problem in the sense that maximizing a potential function of this distance in n -space translates into the maximized solution in 1-space.

4.4 Our potential function

Two vectors $\vec{y}_i = \langle a_{i_1} \cdots a_{i_n} \rangle$, $\vec{y}_j = \langle a_{j_1} \cdots a_{j_n} \rangle$ will be far apart if the angle between them θ_{ij} is large. The scalar product:

$$\begin{aligned}
\langle \vec{y}_i, \vec{y}_j \rangle &= \sum_{k=1}^n a_{i_k} a_{j_k} \\
&= \|\vec{y}_i\| \|\vec{y}_j\| \cos \theta_{ij} \\
&= \cos \theta_{ij}
\end{aligned}$$

is our definition of distance (actually, 1- the scalar product). Remember, we're simplifying things and our vectors are of unit (Euclidian) norm.

In order to define our potential function let $m_{ij} = \cos \theta_{ij}$. So $-1 \leq m_{ij} \leq 1$ and $m_{ij} \rightarrow -1$ the farther apart $\vec{y}_i$ and $\vec{y}_j$. Then given a graph (V, E) let our potential function be:

$$\Phi = \sum_{(i,j) \in E} \frac{1 - m_{ij}}{2} \quad (\text{potential function})$$

In other words, two connected vertices $(i, j) \in E$ contribute more to the potential function the farther they are apart. Here comes the magic show (i.e., get ready for some serious handwaving).

4.5 The algorithm

Let the m_{ij} 's define a (symmetric) matrix M . This matrix is used to define and optimize a system of linear inequalities that will maximize our potential function. Given that $m_{ij} = \langle \vec{y}_i, \vec{y}_j \rangle$ for some set of vectors $\vec{y}_1, \dots, \vec{y}_n$:

1. There is a set of linear inequalities L such that $m \in L$ if and only if satisfies each inequality in L
2. There exists a poly time algorithm given $m \notin M$ finds an inequality in L that M does not satisfy
3. There exists a poly time algorithm given $m \in M$ that returns vectors $\vec{y}_1, \dots, \vec{y}_n$ such that $m_{ij} = \langle \vec{y}_i, \vec{y}_j \rangle$

The assertion is that by (1) and (2) above we can use the ellipsoid method to optimize our LP problem to maximize Φ and then by (3) find the corresponding (optimized) points on the unit sphere. We then partition the sphere

into two sets by drawing a random hyperplane through its center. Or, equivalently, partition by picking a random vector representing a point on the sphere and examining the scalar product of that vector with the other points on the sphere. So our algorithm works as follows:

1. Use the ellipsoid method to optimize our potential function over a set of linear inequalities defined by M
2. Convert the optimized solution to the corresponding vectors $\vec{y}_1, \dots, \vec{y}_n$
3. Pick a random vector $\vec{r}$ defining a point on the sphere
4. For each vertex $i \in V$ if $\langle \vec{y}_i, \vec{r} \rangle \geq 0$ let vertex $i \in S_0$ else let $i \in S_1$

So we partition the graph according to whether the inner product of a point with the random vector is positive or negative. Because the optimization step tries to place connected vertices on “opposite” sides of the sphere, this product is likely to have opposite sign for connected nodes.

4.6 Basic analysis

Let:

$$Opt_R = \sum_{(i,j) \in E} \frac{1 - \langle y_i, y_j \rangle}{2} = \frac{1 - \cos \theta_{ij}}{2}$$

The expected number of edges in our cut is:

$$Exp = \sum_{e \in E} Pr\{e \in cut\} = \sum_{(i,j) \in E} \frac{\theta_{ij}}{\pi}$$

This is intuitive since as θ approaches π the probability that the “arch” between the two points will be cut by a random hyperplane increases. So with $0 \leq \theta \leq \pi$:

$$\frac{Exp}{Opt_R} \geq \min_{\theta} \left\{ \frac{\frac{\theta}{\pi}}{\frac{1 - \cos \theta}{2}} \right\} = \frac{2}{\pi} \cdot \min_{\theta} \left\{ \frac{\theta}{1 - \cos \theta} \right\}$$

This equation can be shown to be $\approx .6$, a considerable (20%) improvement over the trivial algorithm originally given.

The (more complicated) *real* version of Geomans-Williamson takes considerably more analysis but can be shown to be $\approx .88$ of the optimal, a formidable approximation.

We get the same bound on the ratio between the expected value and the optimal cut by noting that the optimal cut has a corresponding geometric relaxation whose potential equals the cut: map everything in S_0 to one point on the sphere, and everything in S_1 to the opposite point. Each edge between S_0 and S_1 contributes $(1 - \cos \pi)/2 = 1$ to the potential.

5 Conclusions

The main idea was to see that in order to approximate important problems we can not only relax the requirements to real values (probabilities), but also to vectors as well. Once an optimized solution in this higher dimension is obtained, it is “randomly” mapped back to a scalar solution that gives a good approximation for the original problem.

So we’ve now seen randomized rounding in two different contexts. Last class it was used explicitly to round from probabilities to booleans. In this class it was used to map vectors in n -dimensions to a binary partition of graph vertices.

This geometric version of randomized rounding has been employed to obtain some of the best-known approximations for $NP - hard$ problems. This lecture was intended to convey the basic idea of how such algorithms work. However in the process we’ve glossed over *a lot* of details. The original Goemans-Williamson paper contains far more in-depth analysis and applications to other $NP - hard$ problems.

6 References

The original Goemans-Williamson paper is:

M. Goemans and D. Williamson, “Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming,” first appeared in *26th Annual ACM Symposium on Theory of Computation*,
<http://citeseer.nj.nec.com/goemans95improved.html>

Motwani and Raghavan (“Randomized Algorithms”) suggest the following related paper:

D. Karger, R. Motwani, and M. Sudan. “Approximate graph coloring by semidefinite programming,” *Proceedings of the 35th Annual IEEE FOCS*, pages 2-13, 1994.

and in particular the following survey:

R. Motwani, J. Naor, and P. Raghavan. “Randomization in approximation algorithms,” in *Approximation Algorithms*, 1995.