Rounding Techniques in Approximation Algorithms

Lecture 20: SDPs and Max Cut

Lecturer: Nathan Klein

1 Max Cut

In the Max Cut problem, we are given a graph $G = (V, E)$ and we want to find a set $S \subseteq V$ so that $|\delta(S)|$ is maximized.

It's not immediately clear how to find an integer linear program that solves max cut. We might try to assign a variable $x_v \in \{0, 1\}$ for each vertex, and then maximize the number of edges $e = (u, v)$ for which $x_u \neq x_v$. The natural way to implement this would be to create variables x_{uv} for each $e = \{u, v\}$ and maximize $\sum_{\{u,v\} \in E} x_{uv}$.

Now, we need to ensure that $x_{uv} \leq |x_u - x_v|$. It turns out this would work for *min cut* where we minimize $\sum_{\{u,v\} \in E} x_{uv}$ since we can write $x_{uv} \geq x_u - x_v$ and $x_{uv} \geq x_v - x_u$, and now we are only allowed to set x_{uv} to 0 if the edges do not cross the cut. But for max cut, this is an issue. I could try $x_{uv} \leq x_u - x_v$ and $x_{uv} \leq x_v - x_u$, but this is hopeless, as if $x_u \neq x_v$ we will constrain $x_{uv} \leq -1$. At a high level, this is because absolute value is a convex function, and we can minimize things over convex functions. But we cannot maximize.

There are, it turns out, some LPs we can write down that are valid relaxations. But they are all fairly useless, with an integrality gap of 2, whereas taking a random half of the vertices gives us a cut of size at least $\frac{1}{2}|E| \geq \frac{1}{2}OPT$.

In this course, usually this would lead to an open problem, as was in the case of vertex cover. However, in this instance, there turns out to be something quite beautiful we can do that leads to a much better approximation algorithm. This idea is due to Delorme and Poljak [DP93] and was then used by Goemans and Williamson [GW95] to obtain a better approximation for Max Cut.

1.1 Nonlinear Constraints

Let's keep using our indicator variables $x_v \in \{0, 1\}$ to let us know if v is in our cut $S \subseteq V$ or not. But now *let's allow ourselves to use non-linear constraints*. And for ease of notation, let's let $x_v \in \{-1, 1\}$ instead. Then, we can create variables y_{uv} for each $u, v \in V$ and write:

$$\begin{aligned} \max \quad & \sum_{\{u,v\} \in E} \frac{1}{2}(1 - y_{uv}) & \\ \text{s.t.} \quad & y_{uv} = x_u x_v & \forall u, v \in V \\ & x_v \in \{-1, 1\} & \forall v \in V \end{aligned}$$

We will create variables y_{vv} as well, taking $u, v \in V$ to not necessarily be distinct. We will also let y_{uv} and y_{vu} be the same variable.

This program works because if x_u and x_v take different signs (i.e. are on different sides of the cut) then $y_{uv} = -1$ and this edge contributes 1 to the objective function. Otherwise $y_{uv} = 1$ and

this edge contributes 0. Here's another way to write the same program:

$$\begin{aligned} \max \quad & \sum_{\{u,v\}\in E} \frac{1}{2}(1-y_{uv}) \\ \text{s.t.} \quad & y_{uv} = x_u x_v && \forall u,v \in V \\ & y_{vv} = 1 && \forall v \in V \end{aligned}$$

This is equivalent since $y_{vv} = x_v^2 = 1$ if and only if $x_v \in \{-1, 1\}$. But now, there is only one problematic set of constraints, this $y_{uv} = x_u x_v$, and we don't have to worry about the integer versus linear program aspect. Note that x_v is unconstrained.

Let's "bash on regardless" and try to model this with linear constraints. And let's not even worry about writing them down: we have the ellipsoid method, after all. Let's just delete the constraint $y_{uv} = x_u x_v$ and solve the LP and hope we can implement a separation oracle.

Notice that this means that x_u, x_v are not in the LP any more, and the y_{uv} values are unconstrained. So the LP will now lie to us. It will say, look, here's a great solution! $y_{uv} = -1000$ for all y_{uv} and, sure, I'll set $y_{vv} = 1$ for all $v \in V$. This will have a huge objective value and the LP will be very proud for having found such a good max cut.

But it's easy to refute these values. Remember, we're trying to be the separation oracle ourselves. Now, we return the hyperplane:

$$y_{uu} + y_{vv} + 2y_{uv} \geq 0$$

Why is this valid? Well, let's rewrite it using the x values:

$$x_u^2 + x_v^2 + 2x_u x_v = (x_u + x_v)^2 \geq 0$$

Nice. And now, using that $y_{vv} = 1$ for all $v \in V$, our first inequality says $y_{uv} \geq -1$. So at least our LP now can get at most value $|E|$.

So, we continue. Now maybe the LP returns a solution which sets $y_{uv} = -1$ for all edges in a triangle u, v, w. This is obviously wrong, so let's see if we can refute this. The first thing we might try is the following:

$$(x_u x_v + x_u x_w + x_v x_w)^2 \geq 0$$

Which, expanded and replacing squared terms with 1 is:

$$3 + 2x_u x_v + 2x_v x_w + 2x_u x_w \geq 0$$

Which is a refutation, giving us $y_{uv} + y_{vw} + y_{uw} \geq -\frac{3}{2}$. Note that this says the objective value of a triangle is at most $\frac{3}{2} - \frac{1}{2}(y_{uv} + y_{vw} + y_{uw}) \leq \frac{3}{2} + \frac{1}{2} \cdot \frac{3}{2} = \frac{9}{4}$.

Now notice that all of our separating hyperplanes so far (which are linear in y_{uv}) have originated from inequalities like $(\sum_{v\in V} c_v x_v)^2 \geq 0$ for some $c \in \mathbb{R}^n$. Can we capture *all* such inequalities and automate this?

1.2 Semidefinite Programming and Second Moments

$(\sum_{v\in V} c_v x_v)^2 \geq 0$ is equivalent to $0 \leq \sum_{u,v\in V} c_u c_v x_u x_v = \sum_{u,v\in V} c_u c_v y_{uv}$ (where pairs u, v appear twice). But letting $Y = (y_{uv})_{u,v\in V}$, this is simply saying that $c^T Y c \geq 0$. Sound familiar?

This is exactly asking that this matrix Y be PSD. Let's think back for a moment to our constraint $y_{uv} + y_{vw} + y_{uw} \geq -\frac{3}{2}$ to let this sink in. Consider the matrix:

$$\begin{bmatrix} 1 & y_{uv} & y_{uw} \\ y_{uv} & 1 & y_{vw} \\ y_{uw} & y_{vw} & 1 \end{bmatrix}$$

Then, letting $c = (1, 1, 1)$, we have that $\mathbf{1}^T Y \mathbf{1} \geq 0$, or equivalently: $3 + 2y_{uv} + 2y_{vw} + 2y_{uw} \geq 0$. So asking this matrix of y values to be PSD is capturing all constraints of the above form, and we can already see that this is doing some non-trivial things. At least it's separating over some solutions that are clearly not optimal.

It's not too difficult to prove that we can efficiently separate over the infinite constraint set $c^T Y c \geq 0$ for all $c \in \mathbb{R}^n$. Notice that the resulting inequalities are linear in the y_{uv} variables. So, the ellipsoid method can solve this optimization problem up to any additive accuracy ϵ in time polynomial in $\log(1/\epsilon)$.[1] This leads us to the following semidefinite program:

$$\begin{aligned} \max \quad & \sum_{\{u,v\} \in E} \frac{1}{2}(1 - y_{uv}) \\ \text{s.t.} \quad & (y_{uv})_{u,v \in V} \succeq 0 \\ & y_{vv} = 1 \qquad\qquad \forall v \in V \end{aligned}$$

Notice that we just replaced the constraint $y_{uv} = x_u x_v$ with $(y_{uv})_{u,v \in V} \succeq 0$. This is of course weaker than the original constraint, but it's a relaxation. We're still following the relax-and-round framework, just instead of relaxing the integrality constraint (which did not even exist in our formulation) we relaxed this quadratic one.

Fact 1.1. *The integrality gap of this LP is at most* $\frac{2}{9/4} < 0.889$.[2]

Proof. We already saw for the triangle a bound of $\frac{9}{4}$, even though OPT is clearly 2. We can construct a lower bound by exhibiting a PSD matrix.

$$\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} & 1 \end{bmatrix}$$

A nice way to check that this is PSD is to see that it is diagonally dominant, i.e. $|a_i i| \geq \sum_{j i'} |a_{ij}|$ for all i. This implies that the matrix is PSD. One can also check Sylvester's criterion, which says that a matrix is PSD if and only if all of its principal minors are non-negative. □

Now comes the intuition behind this PSD criterion:

[1] Since the constraint family is infinite, we can no longer solve the problem exactly, but since we're approximating the solution anyway it's not a big deal.

[2] More complicated examples can bring this to ≈ 0.878.

Second Moments

The way to think about this is that we want a distribution over the signs $x \in \{-1,1\}^n$, and instead of just looking at the *first moments* of this distribution, i.e. $\mathbb{E}[x]$ (as we have done for the entire course so far), we are also looking at the *second moments*, $\mathbb{E}\left[xx^T\right]$. And what do we know about any covariance matrix? It must be PSD, as for any distribution μ over $x \in \{-1,1\}^n$ we must have (as a reminder):

$$c^T\mathbb{E}\left[xx^T\right]c = c^T(\sum_S \mathbb{P}[S]\, xx^T)c = \sum_S \mathbb{P}[S]\,(y^Tx)^2 \geq 0$$

This opens up a new world of possibilities in terms of finding efficient relaxations. We will continue to explore this in the last part of the course.

There is an alternate intuition as well involving optimizing over vectors $x_v \in \mathbb{R}^n$ instead of $x_v \in \{-1,+1\}$ and trying to maximize the angle between adjacent vectors. You can check out the course notes linked from the website if you are interested in this perspective and vector programs.[3]

For now, let's see how to round a solution given the covariance matrix.

1.3 Rounding

Let's continue with our usual relax-and-round framework. Given a solution $Y = (y_{uv})_{u,v\in V}$, how do we actually round it?

Using our intuition from above, we treat Y as a covariance matrix and find a distribution μ which has covariance matrix Y. We already saw an example of how to do this in the previous unit: take Y, find its square root $Y^{1/2}$, and let $x = Y^{1/2}r$, where $r \in \{-1,+1\}^n$ samples each coordinate independent at random to be -1 or $+1$ each with probability $1/2$.

Here, we will do something slightly different. Instead of letting r be random ± 1 entries, we will each r_i be an independently sampled Gaussian with mean 0 and variance 1. Now, similar to the ± 1 case, we have:

$$\mathbb{E}\left[xx^T\right] = \mathbb{E}\left[Y^{1/2}rr^TY^{1/2}\right] = Y^{1/2}\mathbb{E}\left[rr^T\right]Y^{1/2} = Y$$

where we used that each Gaussian has variance 1 and each r_i is independent.

Unfortunately, $x = Y^{1/2}r$ will not be in $\{-1,+1\}^n$. If we could somehow sample from μ, get covariance matrix Y, and ensure x was in $-1, 1$, we would get a 1-approximation. This is exactly like rounding an LP: if we could get an integer solution with the same marginals as x, we would obtain a 1-approximation.

Fact 1.2. *Let μ be a distribution over vectors in $\{-1,+1\}^n$ and suppose $\mathbb{E}[x_ux_v] = y_{uv}$ for all $u, v \in V$. Then, we obtain a 1-approximation.*

Proof. The expected cost of our algorithm is $\sum_{\{u,v\}\in E}\frac{1}{2}(1 - \mathbb{P}_{S\sim\mu}[x_u \neq x_v]) = \sum_{\{u,v\}\in E}\frac{1}{2}(1 - \mathbb{E}[x_ux_v]) = c(y)$. □

[3] Personally, I prefer the covariance view because it immediately suggests the question: what about third or fourth moments? Which we will touch upon in the remainder of the course.

So, just like when rounding LPs, we must lose something when we get an integer distribution. What's the most natural way to round in this setting? Let's do the triangle example, and maybe you can guess what to do. There the covariance matrix and its square root are:

$$Y = \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} & 1 \end{bmatrix}, \quad Y^{1/2} \approx \begin{bmatrix} 0.816 & -0.408 & -0.408 \\ -0.408 & 0.816 & -0.408 \\ -0.408 & -0.408 & 0.816 \end{bmatrix}$$

Let's sample a random Gaussian, say we get $r = \begin{bmatrix} -0.14 \\ 0.78 \\ 0.37 \end{bmatrix}$ and look at $Y^{1/2}r$. This is $\begin{bmatrix} -0.76 \\ 0.89 \\ -0.12 \end{bmatrix}$. How might you round this to $\{-1, +1\}^3$?

It turns out the simplest idea works here. After sampling from our distribution μ over $\mathbb{R}^n$ with covariance matrix Y, for each coordinate x_i, let $x_i = 1$ if $x_i \geq 0$ and -1 otherwise. Amazingly, this gives an ≈ 0.878 approximation, the best known approximation algorithm for Max Cut. This matches the integrality gap of this SDP and is optimal under the Unique Games Conjecture. Let's prove that the approximation ratio is at least 0.878: it's not too difficult, in fact, using a result from the 19th century.

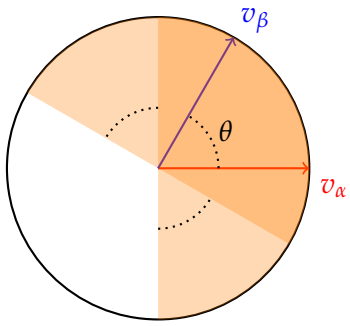

Figure 1: A visual proof that the regions in which the signs differ have total angle 2θ. In the solid yellow region, both signs will be positive, and in the white region, both will be negative. The remainder is the desired region with total angle 2θ.

Lemma 1.3 (Sheppard's Formula [She98]). *Let $\alpha, \beta \sim \mathcal{N}(0,1)$ be two correlated Gaussians such that $\mathbb{E}[\alpha\beta] = \rho$. Then, $\mathbb{P}[sign(\alpha) \neq sign(\beta)] = \frac{\arccos(\rho)}{\pi}$.*

Proof. (α, β) is a multivariate Gaussian with covariance matrix $\begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}$ and mean 0. This uniquely defines the distribution. A standard way to sample a multivariate Gaussian with covariance matrix C is to sample r where each $r_i \sim \mathcal{N}(0,1)$ independently and then output $C^{1/2}r$. We have already noticed this produces the desired covariance matrix.

Now, let v_α be the first row of $C^{1/2}$ and v_β the second. Then, $\alpha = \langle v_\alpha, r\rangle$ and $\beta = \langle v_\beta, r\rangle$. α will be positive if the angle between v_α and r is in the range $[-\frac{\pi}{2}, \frac{\pi}{2}]$ and similarly for β. The angle of r is uniformly random. So, the signs will be different according to Fig. 1, with probability equal to to twice the angle between v_α and v_β divided by 2π. Now:

$$\rho = \langle v_\alpha, v_\beta\rangle = \|v_\alpha\|\|v_\beta\|\cos(\theta) = \cos(\theta),$$

where in the first equality we used that the covariance matrix is as above. This completes the proof. □

We can now use this to finish the proof. We will use the following computational lemma, which we sketch by picture (see Fig. 2):

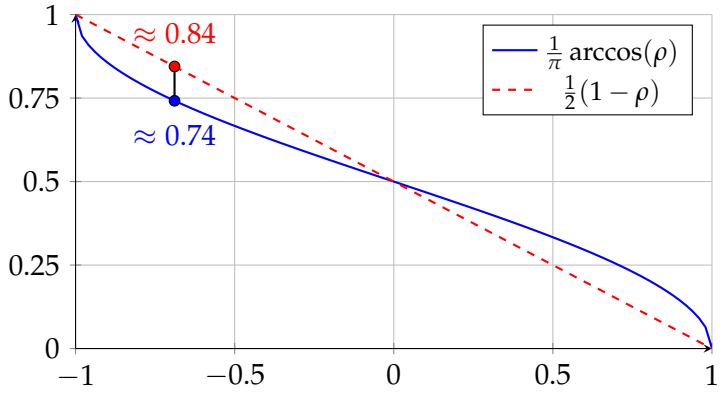

Figure 2: This can be verified mathematically, but one can check that the largest deviation occurs at approximately -0.689 and has a ratio of approximately 0.878.

Lemma 1.4. *For $\rho \in [-1, 1]$, we have*

$$\frac{\arccos(\rho)}{\pi} \geq 0.878 \cdot \frac{1}{2}(1-\rho)$$

But now we're done, as the expected number of edges cut is (by linearity of expectation):

$$\mathbb{E}\left[|\delta(S)|\right] = \sum_{\{u,v\}\in E} \frac{\arccos(y_{uv})}{\pi} \geq 0.878 \cdot \sum_{\{u,v\}\in E} \frac{1}{2}(1-y_{uv}) = 0.878 \cdot c(y)$$

As mentioned, this is also the integrality gap and this ratio can be improved unless the Unique Games Conjecture is false [Kho+07].