Max-cut approx. alg. via SDP

Max-Cut Problem:

Given weighted graph $G=(V,E)$ with weights $w(e)>0$ for $e\in E$

Goal: Find cut $(S,\bar{S})$ which maximizes

$$\sum_{\substack{(v,y)\in E: \\ v\in S, y\in \bar{S}}} w(v,y)$$

Simple $\frac{1}{2}$-approx.:

For each $v\in V$, assign v to S with prob. $1/2$
& to $\bar{S}$ " $1/2$.

Denote this cut by $Y: V\to\{0,1\}$

Hence,
$$Y(v) = \begin{cases} +1 & \text{with prob. } 1/2 \\ 0 & \text{with prob. } 1/2 \end{cases}$$

$$E[\text{cut weight}] = E\left[\sum_{(v,z)\in E} w(v,z)\, \mathbb{1}\big(Y(v) \neq Y(z)\big)\right]$$

indicator function

$$= \sum_{(v,z)\in E} w(v,z)\, \Pr\big(Y(v) \neq Y(z)\big)$$

$$= \sum_{(v,z)\in E} w(v,z) \times \frac{1}{2}$$

$$= \frac{W}{2} \quad \text{where } W = \sum_{(v,z)\in E} w(v,z) = \text{total weight of all edges.}$$

So the expected weight of the cut is $\geq \frac{1}{2}$ of optimal.
And we can derandomize by the method of conditional expectations (that we saw last lecture).

Let's do better.

Let's write max-cut as an ILP:

For each vertex v, create variable $y_v \in \{0,1\}$

For each edge (u,v), create variable $z_{uv} \in \{0,1\}$

(idea: $z_{uv} = 1$ iff $y_u \neq y_v$)

Constraints $\quad y_u + y_v \geq z_{uv} \quad (*)$

$\& \; 2 - (y_u + y_v) \geq z_{uv} \quad (**)$

if $y_u = y_v = 0$, then $(*) \Rightarrow z_{uv} = 0$

if $y_u = y_v = 1$, then $(**) \Rightarrow z_{uv} = 0$

if $y_u \neq y_v$ then $z_{uv} \in \{0,1\}$ but we will maximize it.

ILP:

$$\max \sum_{(u,v)\in E} w(u,v)\, z_{uv}$$

s.t.

for all $(u,v) \in E$, $\quad y_u + y_v \geq z_{uv}$

$2 - (y_u + y_v) \geq z_{uv}$

$z_{uv} \in \{0,1\}$

for all $v \in V$, $\quad y_v \in \{0,1\}$

Consider the LP relaxation by replacing

$y_v \in \{0,1\}$ by $0 \le y_v \le 1$

& $z_{uv} \in \{0,1\}$ by $0 \le z_{uv} \le 1$

But this LP is a poor estimate of the ILP:

Set $y_v = 1/2 \quad \forall v \in V$.

Then the LP has obj. function $= W$

So this is off by a factor of 2 in some graphs

& it is equivalent when we do randomized rounding to the simple rand. alg.

Different formulation:

Instead of $Y: V \to \{0,1\}$ do $Y: V \to \{-1,+1\}$

Then $Y(u) \neq Y(v) \Leftrightarrow Y(u)Y(v) = -1$

& $$\frac{1 - Y(u)Y(v)}{2} = \begin{cases} 1 & \text{if } Y(u) \neq Y(v) \\ 0 & \text{if } Y(u) = Y(v) \end{cases}$$

Now we can write Max-Cut as an IQP (Integer Quadratic Program)

$$\max \sum_{(i,j)\in E} w(i,j)\left(\frac{1 - v_i v_j}{2}\right)$$

$$\text{s.t. } \forall i \in V,\ v_i^2 = 1 \ \&\ v_i \in \mathbb{R}.$$

thus $v_i \in \{+1, -1\}$.

IQP is still NP-hard but we can relax it so that each v_i is a unit-vector in n-dimensions instead of 1-dimension.

& $v_i v_j$ becomes the dot-product $v_i \cdot v_j$ or inner product

For $a, b \in \mathbb{R}^n$, $a \cdot b = \sum_{i=1}^{n} a_i b_i = \|a\| \, \|b\| \cos\theta$

We'll have $\|a\| = \|b\| = 1$, so $a \cdot b = \cos\theta$ where θ = angle between them.

This is a SDP = semidefinite program which can be solved in poly-time.

SDP:

$$\max \sum_{(u,v)\in E} w(u,v)\left(\frac{1-\vec{y_u}\cdot\vec{y_v}}{2}\right)$$

$$\text{s.t. } \forall v\in V, \vec{y_v}\cdot\vec{y_v}=1$$

$\|y_v\|=1$ so it's unit length.

Take a solution $\vec{y_v}$ to this SDP.
Now let's "round" somehow to find a cut $(S,\bar{S})$.

Take a random hyperplane H through zero.
Everything on one side of H is assigned to S
& on other side of H is $\bar{S}$.

How to define H?
Choose random unit vector $\vec{r}$ &
$\vec{r}$ is normal to the hyperplane.

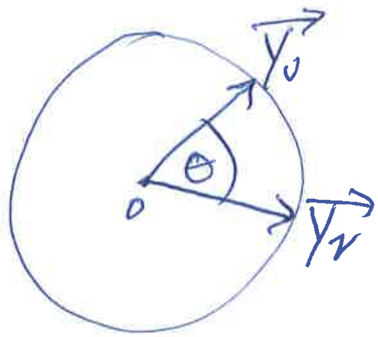
Then $f(i)=\mathrm{sgn}(\vec{r}\cdot\vec{y_i})$

~~if $\vec{y_i}\in H^+$ & $\vec{r}\in$~~

if $\vec{y_i}$ & $\vec{r}$ lie on same side of H,
then $\vec{r}\cdot\vec{y_i}>0$,
& if on diff. sides then $\vec{r}\cdot\vec{y_i}<0$.

$$\Pr\big(\,(u,v) \text{ crosses the cut } (S,\bar{S})\,\big)$$
$$= \Pr\big(H \text{ splits } \vec{y_u} \,\&\, \vec{y_v}\big)$$

Look at the 2-dimensional plane containing $\vec{y_u}$ & $\vec{y_v}$.
The angle between them is θ.



H is a random ~~vector~~ line through O in this 2-dimensional plane.

Thus, the probability that H cuts $\vec{y_u}$ & $\vec{y_v}$ is $\dfrac{\theta}{\pi}$

$$\Pr\left((u,v) \text{ is cut}\right) = \frac{\theta}{\pi} = \frac{\cos^{-1}(\vec{y_u} \cdot \vec{y_v})}{\pi}$$

Thus, $$E\left[\begin{matrix}\text{weight of} \\ \text{cut } (S, \bar{S})\end{matrix}\right] = \sum_{(u,v)\in E} w(u,v) \frac{\cos^{-1}(\vec{y_u} \cdot \vec{y_v})}{\pi}$$

Recall, objective value $= \sum_{(u,v)\in E} w(u,v)\left(\frac{1 - \vec{y_u}\cdot\vec{y_v}}{2}\right)$

$\geq$ optimal cut weight

since this is a relaxation of the IQP.

Claim: For $\alpha \approx .87856\ldots$,

for all $\sigma_{uv} \in [-1,1]$,

$$\frac{\cos^{-1}(\sigma_{uv})}{\pi} \geq \alpha\left(\frac{1-\sigma_{uv}}{2}\right)$$

Proof is simple calculus?

See Gupta-O'Donnell's notes from CMU

A symmetric $n \times n$ matrix X is PSD (Positive Semidefinite) if all eigenvalues are nonnegative.

Equivalently, $\exists$ $m \times n$ matrix V where $X = V^T V$ for some $m \leq n$

Thus, can view V^T as n vectors in $\mathbb{R}^m$ & X is the pairwise dot product.

SDP = semidefinite program: n^2 variables x_{ij} for $1 \leq i,j \leq n$

$$\max \quad \cancel{\mathbf{c}^T x} \quad \sum_{i,j} c_{ij} x_{ij}$$

$$\text{s.t.} \quad \cancel{Ax \leq b} \quad \sum_{i,j} a_{ijk} x_{ij} = b_k \quad \forall k$$

$$x_{ij} = x_{ji} \quad \forall i,j$$

$$X = (x_{ij}) \succeq 0.$$

X is PSD

SDP's can be solved in poly-time

6

Alternative form:

$$\max \sum_{i,j} c_{ij}(v_i \cdot v_j)$$

$$\text{s.t.} \sum_{i,j} a_{ijk}(v_i \cdot v_j) = b_k \quad \forall k$$

$$v_i \in \mathbb{R}^n \quad \forall i$$

These are equivalent since X is PSD iff $X = V^T V$ for some V.

Max-cut as an IQP:

$$\max \tfrac{1}{2}\sum_{(i,j)\in E} w_{ij}(1 - y_i y_j) = Z_{IQP}$$

$$\text{s.t.} \quad y_i \in \{-1, +1\} \quad \forall i$$

SDP relaxation:

$$\max \tfrac{1}{2}\sum_{(i,j)\in E} w_{ij}(1 - v_i \cdot v_j) = Z_{SDP}$$

$$\text{s.t.} \quad v_i \cdot v_i = 1 \quad \forall i$$

$$v_i \in \mathbb{R}^n$$

Note, $OPT = Z_{IQP}$ & $Z_{IQP} \leq Z_{SDP}$

since any feasible IQP gives a feasible SDP.

We saw a $\frac{3}{4}$-approx. algorithm for Max-SAT.

For Max-3SAT, can use SDP to get a $\frac{7}{8}$-approx. algorithm

(see [Karloff, Zwick '97]

This is best possible under the unique games conjecture (see [Khot, Kindler, Mossel, O'Donnell '07]

& [Mossel, O'Donnell, Oleszkiewicz '10]

* Can derandomize this Max-cut approx. alg.

(See [Mahajan, Ramesh '99]