[CS1961: Lecture 9] Linear Programming Rounding, Positive Semi-Definite Programming Rounding

Instructor: Chihao Zhang;
Scribed by Xianruo Yu, Youwei Zhong, Yuchen He

1 MaxSAT

Let $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ be a CNF formula where each clause C_j is the disjunction of ℓ_j literals. How to find an assignment that satisfies maximum number of clauses?

A simple idea is to derandomize the existence proof provided by the probabilistic method. The number of satisfied clauses in this algorithm is larger than $\mathbf{E}[s(\sigma)]$ [1] and

$$\mathbf{E}[s(\sigma)] = \sum_{j=1}^{m} \mathbf{Pr}[C_j \text{ is satisfied}] = \sum_{j=1}^{m} \left(1 - 2^{-\ell_j}\right) \geqslant \frac{m}{2} \geqslant \frac{\mathrm{OPT}}{2}.$$

[1] Let σ be an assignment that each variable x_i is sampled u.a.r. from $\{0,1\}$ where 0 stands for false and 1 stands for true. Then $s(\sigma)$ is the number of clauses that are satisfied under σ.

This is an $\frac{1}{2}$-approximation algorithm. In the last lecture, we showed that this approximation ratio can be improved to 0.618 by assigning the variables which appear negatively in a singleton clause as false with larger probability. However, these two algorithms are not optimal since intuitively, we prefer those variables whose appearances are mostly positive to be true. We then introduce a more clever algorithm which can further improve the approximation rate to 0.75.

1.1 Lineal Programming Rounding

Note that we can reduce a MaxSAT problem to *integer programming*. Write C_j as $\left(\bigvee_{i\in P_j} y_i\right) \vee \left(\bigvee_{k\in N_j} \overline{y}_k\right)$ where P_j and N_j are the sets of subscripts of the variables y_i and y_k that appears positively and negatively in C_j respectively. Let $z_j = \mathbf{1}[C_j \text{ is satisfied}]$. Then the MaxSAT problem is equivalent to solving the integer program:

$$\begin{aligned}
\max \quad & \sum_{j=1}^{m} z_j \\
\text{s.t.} \quad & \forall j \in [m],\ \sum_{i\in P_j} y_i + \sum_{k\in N_j} (1 - y_k) \geqslant z_j \\
& \forall i \in [n],\ y_i \in \{0,1\} \\
& \forall j \in [m],\ z_j \in \{0,1\}
\end{aligned}$$

However, solving this integer program is **NP**-hard (since it is equivalent to MaxSAT). The difficulty comes from the non-linear constraints $y_i \in \{0,1\}$ and $z_j \in \{0,1\}$. We can relax these constraints and obtain the following

linear program.

$$\begin{aligned}
\max \quad & \sum_{j=1}^{m} z_j \\
\text{s.t.} \quad & \forall j \in [m],\ \sum_{i \in P_j} y_i + \sum_{k \in N_j} (1 - y_k) \geqslant z_j \\
& \forall i \in [n],\ y_i \in [0,1] \\
& \forall j \in [m],\ z_j \in [0,1]
\end{aligned}$$

We can obtain an optimal solution $\{y_i^*\}_{i \in [n]}$ and $\{z_j^*\}_{j \in [n]}$ of this linear program in polynomial time. Intuitively, y_i^* indicates how likely the variable x_i is set to true in the optimal solution. Therefore, we toss a coin with bias y_i^* to determine the value of x_i. With such a random assignment,

$$\begin{aligned}
\mathbf{Pr}\left[C_j \text{ is not satisfied}\right] &= \prod_{i \in P_j} (1 - y_i^*) \prod_{k \in N_j} y_k^* \\
&\leqslant \left(\frac{1}{\ell_j} \left(\sum_{i \in P_j} (1 - y_i^*) + \sum_{k \in N_j} y_k^* \right) \right)^{\ell_j} \\
&\leqslant \left(\frac{1}{\ell_j} \left(\ell_j - \sum_{i \in P_j} y_i^* - \sum_{k \in N_j} (1 - y_k^*) \right) \right)^{\ell_j} \\
&\leqslant \left(1 - \frac{z_j^*}{\ell_j} \right)^{\ell_j}
\end{aligned}$$

where the first inequality follows from the *inequality of arithmetic and geometric means* and the last inequality follows from the constraints of $\{y_i^*\}$ and $\{z_j^*\}$.

AM-GM inequality: For any n non-negative numbers $x_1, \ldots, x_n$, it holds that $\frac{1}{n} \sum_{i \in [n]} x_i \geqslant \left(\prod_{i \in [n]} x_i \right)^{\frac{1}{n}}$.

Then we have

$$\begin{aligned}
\mathbf{E}_{\sigma \sim \{y_i^*\}}\left[s(\sigma)\right] &= \sum_{j=1}^{m} \left(1 - \mathbf{Pr}\left[C_j \text{ is not satisfied}\right]\right) \\
&\geqslant \sum_{j=1}^{m} \left(1 - \left(1 - \frac{z_j^*}{\ell_j} \right)^{\ell_j} \right) \\
&\geqslant \sum_{j=1}^{m} \left(1 - \left(1 - \frac{1}{\ell_j} \right)^{\ell_j} \right) z_j^* \qquad (1)
\end{aligned}$$

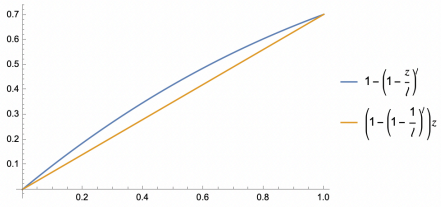

Figure 1: $\ell = 3$

where the last inequality follows from the fact that $h(z) = 1 - \left(1 - \frac{z}{\ell}\right)^{\ell}$ is concave in $[0,1]$ for any $\ell \in \mathbb{N}_+$ and therefore $h(z) \geqslant h(1) \cdot z$. Let $\mathrm{OPT}_{\mathrm{LP}}$ be the result corresponding to the optimal solution of the linear program. Note that Equation (1) achieves the minimum value when $\ell_j \to \infty$. We have

$$\mathbf{E}_{\sigma \sim \{y_i^*\}}\left[s(\sigma)\right] \geqslant \left(1 - \frac{1}{e}\right) \sum_{j=1}^{m} z_j^* = \left(1 - \frac{1}{e}\right) \mathrm{OPT}_{\mathrm{LP}}.$$

Since $\mathrm{OPT}_{\mathrm{LP}}$ is absolutely no less than the actual result OPT. We have

$$\mathbf{E}_{\sigma\sim\{y_i^*\}}[s(\sigma)] \geqslant \left(1-\frac{1}{e}\right)\mathrm{OPT} \approx 0.632\cdot\mathrm{OPT}.$$

Note that our previous algorithm with fair and this linear programming rounding algorithm are two extremes. The fair coin algorithm achieves the minimum value $\frac{\mathrm{OPT}}{2}$ when each $\ell_j = 1$ and this algorithm achieves $0.632\cdot\mathrm{OPT}_{\mathrm{LP}}$ when $\ell_j \to \infty$. We can take advantage of their strengths by running the two algorithms and choosing the better assignment. Denote by $s(\sigma_1)$ and $s(\sigma_3)$ the number of satisfied clauses in the derandomization algorithm and the rounding algorithm. Then

$$\begin{aligned}
\mathbf{E}[\max\{s(\sigma_1), s(\sigma_3)\}] &\geqslant \frac{1}{2}\mathbf{E}[s(\sigma_1)+s(\sigma_3)] \\
&\geqslant \frac{1}{2}\sum_{j=1}^{m}\left(1-2^{-l_j}\right)+\sum_{j=1}^{m}\left(1-\left(1-\frac{1}{l_j}\right)^{l_j}\right)z_j^* \\
&\geqslant \frac{1}{2}\sum_{j=1}^{m}\left(1-2^{-l_j}+1-\left(1-\frac{1}{l_j}\right)^{l_j}\right)z_j^* \\
&\geqslant 0.75\cdot\mathrm{OPT}.
\end{aligned}$$

2 MaxCut

Given an undirected graph $G = (V = [n], E)$, the MaxCut problem asks for a set $S \subseteq V$ that maximizes $\left|E(S, \overline{S})\right|$.

A simple idea is to randomly pick a set $S \subseteq V$ by letting v to be included in S with probability $\frac{1}{2}$ independently for all $v \in V$. It is clear that the expectation of this randomized algorithm is a $\frac{1}{2}$-approximation of MaxCut. We can again derandomize the algorithm by conditional probabilities to get a deterministic algorithm.

Like the case of MaxSAT, this approximation ratio can be further improved using *rounding* technique. However, it is known that the power of LP rounding is limited in this problem. We will round a positive semi-definite program instead.

2.1 Positive Semi-Definite Programming

A symmetric matrix $M \in \mathbb{R}^{n\times n}$ is called *positive semi-definite* if $\mathbf{f}^T M\mathbf{f} \geqslant 0$ for all $\mathbf{f} \in \mathbb{R}^n$, and is denoted as $M \succeq 0$. We can similarly define *positive definite*, denoted as $M \succ 0$. We know that the n eigenvalues of a symmetric matrix $\lambda_1, \lambda_2, \ldots, \lambda_n$ are all real numbers. A matrix M is positive semi-definite iff all the eigenvalues of M are non-negative real numbers.

Theorem 1 (Spectral Decomposition Theorem) *There exists* $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n \in$

$\mathbb{R}^n$ *satisfying*

$$\mathbf{v}_i^T \mathbf{v}_j = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{o.w.} \end{cases}$$

such that

$$M = [\mathbf{v}_1 \ \cdots \ \mathbf{v}_n] \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^T \\ \vdots \\ \mathbf{v}_n^T \end{bmatrix}.$$

Then we can write a positive semi-definite matrix M as $M = \sum_{i=1}^n \lambda_i \mathbf{v}_i \mathbf{v}_i^T$. Furthermore, if M is positive definite, M is invertible and $M^{-1} = \sum_{i=1}^n \lambda_i^{-1} \mathbf{v}_i \mathbf{v}_i^T$.

We claim that a positive semi-definite matrix M can be represented as $U^T U$ for some $U = [\mathbf{u}_1 \ \cdots \ \mathbf{u}_n] \in \mathbb{R}^{n \times n}$. . Actually, if we define $M^{\frac{1}{2}} = \sum_{i=1}^n \lambda_i^{\frac{1}{2}} \mathbf{v}_i \mathbf{v}_i^T$, then we can choose $U = M^{\frac{1}{2}}$.

Consider the positive semi-definite program

$$\begin{aligned} \max \quad & C \bullet X \\ \text{s.t.} \quad & A_k \bullet X \leqslant b_k \ \forall k \in [m] \text{ and } X \succeq 0 \end{aligned}$$

Here $\bullet$ means the *Frobenius inner product.* That is, $C \bullet X = \sum_{i,j} C_{i,j} X_{i,j}$.

If we write X as $U^T U$, i.e., $X_{i,j} = \mathbf{u}_i^T \mathbf{u}_j$ for $i, j \in [n]$, we can rewrite the positive semi-definite program as the following vector program.

$$\begin{aligned} \max \quad & \sum_{i,j \in [n]} C_{i,j} \mathbf{u}_i^T \mathbf{u}_j \\ \text{s.t.} \quad & \forall k \in [m], \sum_{i,j \in [n]} A_k(i,j) \mathbf{u}_i^T \mathbf{u}_j \leqslant b_k \\ & \forall i \in [n], \mathbf{u}_i \in \mathbb{R}^n \end{aligned}$$

Both positive semi-definite programming and vector programming can be solved in polynomial time via *ellipsoid method* or *interior point method.*

2.2 Positive Semi-Definite Programming Rounding Algorithm

Back to the `MaxCut` problem, we can write it as

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{(i,j) \in E} 1 - x_i x_j \\ \text{s.t.} \quad & \forall i \in [n], x_i \in \{-1, 1\}. \end{aligned}$$

where $x_i = 1$ means vertex i is in S and $x_i = -1$ otherwise. We can relax the constraint $\{-1, 1\}$ to $[-1, 1]$ and get

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{(i,j) \in E} 1 - x_i x_j \\ \text{s.t.} \quad & \forall i \in [n], x_i \in [-1, 1]. \end{aligned}$$

However, this programming problem may not be solved in polynomial time since the target function is not convex. We can further regard each x_i as a vector and relax the constraint to get a vector program:

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{(i,j)\in E} 1 - \mathbf{x}_i \mathbf{x}_j \\ \text{s.t.} \quad & \forall i \in [n], \mathbf{x}_i \in \mathbb{R}^n, \|\mathbf{x}_i\| = 1. \end{aligned}$$

Note that this vector program can be solved in polynomial time. Let $\{\mathbf{x}_i^*\}_{i\in[n]}$ be the corresponding solution. Consider a hyperplane with normal vector $\mathbf{r}$ where $\mathbf{r}$ is sampled u.a.r. from a unit sphere[2]. The vertices $\{\mathbf{x}_1^*, \ldots, \mathbf{x}_n^*\}$ can be divided into two parts S and $\overline{S}$ by the hyperplane where $S = \{i \mid \langle \mathbf{x}_i^*, \mathbf{r}\rangle \geqslant 0\}$ and $\overline{S} = \left\{ j \;\middle|\; \langle \mathbf{x}_j^*, \mathbf{r}\rangle < 0 \right\}$. Then we have

$$\begin{aligned} \mathbf{E}\left[\left|E(S,\overline{S})\right|\right] &= \sum_{(i,j)\in E} \mathbf{Pr}\left[\mathbf{x}_i^*, \mathbf{x}_j^* \text{ are separated by the hyperplane}\right] \\ &= \sum_{(i,j)\in E} \frac{\arccos\langle \mathbf{x}_i^*, \mathbf{x}_j^*\rangle}{\pi} \\ &\geqslant \min_{z\in(-1,1)} \frac{2\arccos z}{\pi(1-z)} \sum_{(i,j)\in E} \frac{1 - \langle \mathbf{x}_i^*, \mathbf{x}_j^*\rangle}{2} \\ &\approx 0.878 \sum_{(i,j)\in E} \frac{1 - \langle \mathbf{x}_i^*, \mathbf{x}_j^*\rangle}{2} \\ &\geqslant 0.878 \cdot \mathrm{OPT}. \end{aligned}$$

This approximation ratio $\min_{z\in(-1,1)} \frac{2\arccos z}{\pi(1-z)}$ is optimal if the unique game conjecture is true.

[2] This can be realized by sampling each $r_i = \mathcal{N}(0,1)$ for all $i \in [n]$ and outputting the normalized vector. The probability of choosing a certain $(r_1, r_2, \ldots, r_n)$ is $\prod_{i=1}^n \frac{1}{\sqrt{2\pi}} e^{-\frac{r_i^2}{2}} = (2\pi)^{-\frac{n}{2}} e^{-\frac{\|r\|}{2}}$, which is determined by $\|\mathbf{r}\|$.