Approximation Algorithms **Spring 2019**

Lecture 7: Rounding LPs 2

Scribe: Ayush Kamat April 5th, 2019

7.1 Introduction

In this talk, we provide a randomized 1/2-approximation algorithm for the Maximum Satisfiability Problem, then improve the algorithm incrementally until we reach a 3/4-approximation algorithm which uses randomized linear programs.

Definition 7.1 (MAX-SAT)**.** *The Maximum Satisfiability Problem, henceforth abbreviated MAX-SAT, is defined as follows. Take as inputs n boolean variables $x_1, x_2, \ldots, x_n$ and m boolean expressions $C_1, C_2, \ldots, C_m$, with corresponding weights $w_1, w_2, \ldots, w_m$, each of the form*

$$C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \overline{x}_i,$$

where I and I' are index sets satisfying $P_j, N_j \subseteq \{1, 2, \ldots, n\}$ and $P_j \cap N_j = \varnothing$. Then, we return values of the x_i that maximize the sum of the weights of all C_i that evaluate to true.

In the above, it is important to note two conventions we uphold for the rest of this presentation. Firstly, that no two clauses are identical (if this were the casw, we could just add the weights), and secondly, that each clause contains at most one of x_i and $\overline{x}_i$ (as otherwise, the clause would always be satisfied and virtually useless).

Furthermore, in the context of the MAX-SAT problem, we find it useful to define a *literal* to be one of the boolean variables x_i, and further define x_i to be a *positive literal* and $\overline{x}_i$ to be a *negative literal.* We also denote the number of literals in C_i by l_i. Furthermore, we abide by the convention that the letter m will always be used to denote the number of clauses C_i and the letter n will always be used to denote the number of literals x_i.

Now, we first consider the most trivial random algorithm - set the value of each x_i `true` with probability $\frac{1}{2}$ and `false` otherwise.

Theorem 7.2. *Randomly choosing the value of each x_i with uniform probability is a 1/2-approximation algorithm.*

Proof. For a given clause C_i with length l_i, the probability that we pick the exact combination of values of the x_i that make C_i `false` is $\left(\frac{1}{2}\right)^{l_i}$ so the probability C_i is `true` is the complement of this value: $1 - \left(\frac{1}{2}\right)^{l_i}$. If we then define OPT as the optimal solution and Y_i the $0-1$ random variable corresponding to the truth of C_i, we have from linearity of expectation that

$$\mathbb{E}\left[\sum_i w_i Y_i\right] = \sum_i \mathbb{E}\left[w_i Y_i\right] = \sum_i w_i \left(1 - \frac{1}{2^{l_i}}\right) \geq \frac{1}{2}\sum_i w_i \geq \frac{1}{2}\text{OPT},$$

hence this algorithm on average produces a solution that is half as good as optimal. Thus, it is a 1/2-approximation algorithm. □

7.2 Optimizing with Arbitrary p

How can we optimize our random algorithm? One approach is to bias our random selection so that x_i is `true` with probability p, as opposed to 1/2.

To prove many of the results of this approach, we must first prove a few intermediate lemmas. Furthermore, we first assume that all C_i are not any unital negated terms - in other words, $C_i \neq \overline{x}_j$ for all i, j. We then show that we can drop this assumption and still retain the same results.

Lemma 7.3. *If each x_i is set to* `true` *with probability $p > \frac{1}{2}$, and no clauses are unital negated terms, then the probability that any clause C_i is* `true` *is at least* $\min\left(p, 1-p^2\right)$.

Proof. If C_i is a unit clause, then it will be `true` with probability $p \geq \min\left(p, 1-p^2\right)$. Otherwise then C_i consists of a 'positive' (not negated) unit clauses and b negated unit clauses, with $a+b = l_i \geq 2$. Since $p > \frac{1}{2}$, $p \geq 1-p$, hence we can thus write

$$\Pr(Y_i = 1) = \left(1 - p^a(1-p)^b\right) \geq 1 - p^{l_i} \geq 1 - p^2 \geq \min\left(p, 1-p^2\right),$$

which is as desired. □

This then has the natural extension below:

Lemma 7.4. *If we constrain each C_i to not be a unit negated term, then this procedure induces a* $\min(p, 1-p^2)$*-approximation algorithm.*

Proof. Since each C_i is `true` with probability at least $\min(p, 1-p^2)$, we have that

$$\mathbb{E}\left[\sum_i Y_i\right] = \sum_i \mathbb{E}\left[Y_i\right] \geq \min(p, 1-p^2)\sum_i w_i \geq \min(p, 1-p^2)\text{OPT},$$

hence this is a $\min(p, 1-p^2)$-approximation algorithm, as desired. □

Naturally it'd be beneficial to eliminate the constraint and allow unit negative terms, which we can do by first placing a stronger bound on OPT.

To do this, we introduce the following notation: if the unit clause x_i (should it exist) has weight w_i, then we let v_i denote the weight of the negated unit clause $\overline{x}_i$ (should it also exist). Now, WLOG assume that $w_i \geq v_i$ for all i such that both x_i and $\overline{x}_i$ are present. If this is not the case, then we can simply negate both terms to switch w_i and v_i.

Lemma 7.5. *Given the above assumption, we can enforce the bound*

$$\text{OPT} \leq \sum_j w_j - \sum_i v_i,$$

where summation ranges over all clauses C_j in the first case and all i such that both x_i and $\overline{x}_i$ are clauses in the second case.

Proof. If both x_i and $\overline{x}_i$ are clauses in the problem, because they are mutually exclusive, any solution can only every satisfy one of them. Since the optimal solution can at best only take the clause with higher weight, the result follows. $\square$

From here, we can finally generalize the result:

Theorem 7.6. *Randomly assigning each x_i the value* `true` *with probability p is a $\min(p, 1-p^2)$-approximation algorithm.*

Proof. Consider the subset $U \subseteq \{1, 2, \ldots, m\}$ consisting of all j such that C_j is not of the form $\overline{x}_i$ for some literal x_i. Then, again assuming without loss of generality that the weight of any negated unit clause is not greater than the corresponding unit clause, we have

$$\sum_{i \in U} w_i = \sum_j w_j - \sum_i v_i,$$

where summation is taken over all clauses' weights in the first case and all negated clauses' weights in the second.

From here, we have that

$$\begin{aligned}
\mathbb{E}\left[\sum_i Y_i\right] &= \sum_i \mathbb{E}[Y_i] \\
&\geq \sum_{i \in U} \mathbb{E}[Y_i] \\
&\geq \min(p, 1-p^2) \sum_{i \in U} w_i \\
&= \min(p, 1-p^2) \left(\sum_j w_j - \sum_i v_i\right) \\
&\geq \min(p, 1-p^2)\textsc{opt},
\end{aligned}$$

as desired. $\square$

Now, in order to make this as effective as possible, we need to maximize the value of $\min(p, 1-p^2)$. This is done at $p = \frac{\sqrt{5}-1}{2} \sim .618$, as at this value of p, $p = 1 - p^2$. Therefore, we have produced a $\frac{\sqrt{5}-1}{2}$-approximation algorithm.

7.3 Recasting to Linear Programming

The idea of the previous rounding schemes was to bias all variables equally by introducing a global variable p. However, we can still do better by biasing each variable a different amount. How much we bias each variable can be determined empirically by formulating the MAX-SAT problem as first an integer program, and then relaxing that to a linear program.

Before we begin, we need two lemmas:

Lemma 7.7 (AM-GM Inequality)**.** *For a positive integer k and nonnegative reals $a_1, \dots, a_k$, it is always true that*

$$\frac{1}{k}\sum_{i=1}^{k} a_i \geq \left(\prod_{i=1}^{k} a_i\right)^{\frac{1}{k}}.$$

Proof. See here. □

Lemma 7.8. *If f is a concave function on $[0,1]$, meaning that $f''(x) \leq 0$ for all $x \in (0,1)$, and $f(1)-f(0) = b$, then $f(x) \geq f(0) + bx$ for all $x \in [0,1]$.*

Proof. Use the Mean Value Theorem and the fact that f' is nonincreasing. □

Before we give an integer programming formulation of MAX-SAT, we first introduce some notation: for each literal x_i and clause C_j define

$$y_i = \begin{cases} 1 \text{ if } x_i \text{ is } \texttt{true}, \\ 0 \text{ otherwise,} \end{cases} \qquad \text{and} \qquad z_j = \begin{cases} 1 \text{ if } C_j \text{ is satisfied,} \\ 0 \text{ otherwise.} \end{cases}$$

Further, recall that any clause C_j can be written as

$$C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \overline{x}_i.$$

With this notation, observe that for any j we always have

$$\sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq z_j.$$

This follows as the left side is always a nonnegative integer and is 0 if and only if each positive literal is set to `false` and every negated literal is set to `true`, which also implies that $z_j = 0$ as C_j is not satisfied.

Now, we give the following integer programming formulation of the MAX-SAT problem.

Maximize

$$J(z) = \sum_{j=1}^{m} w_j z_j$$

subject to the conditions

$$\begin{aligned} y_i &\in \{0,1\} & i &\in \{1, \cdots, n\} \\ z_j &\in \{0,1\} & j &\in \{1, \cdots, n\} \\ \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) &\geq z_j & j &\in \{1, \cdots, n\} \\ & & C_j &= \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \overline{x}_i. \end{aligned}$$

It is immediately clear that the value of $J(z)$ where (y, z) is the solution to the integer program is OPT. Now, since integer programs are hard, we relax this to the linear program below.

Maximize

$$J(z) = \sum_{j=1}^{m} w_j z_j$$

subject to the conditions

$$\begin{aligned} 0 \le y_i &\le 1 && i \in \{1, \cdots, n\} \\ 0 \le z_i &\le 1 && j \in \{1, \cdots, n\} \\ \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) &\ge z_j && j \in \{1, \cdots, n\} \\ & && C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \overline{x}_i. \end{aligned}$$

Note the key difference that y_i and z_i are allowed to range over the interval $[0, 1]$ now, as opposed to being fixed inside the finite set $\{0, 1\}$. Observe now that since this program is a relaxation of the previous program, the solution (y^*, z^*) has $J(z^*) \ge$ OPT. From here, we can state our main result.

Theorem 7.9. *Solving the above linear program and then applying randomized rounding to each x_i based on the value y_i^* results in a $\left(1 - \frac{1}{e}\right) \sim .632$-approximation algorithm.*

Proof. Let (y^*, z^*) be the solution to this linear program and let r_j be the probability that C_j is satisfied in this scheme. Then from the Lemma 7.7, we have that

$$1 - r_j = \prod_{i \in P_j} (1 - y_i^*) \prod_{i \in N_j} y_i^* \le \left[\frac{1}{l_j} \left(\sum_{i \in P_j} (1 - y_i^*) + \sum_{i \in N_j} y_i^* \right) \right]^{l_j}.$$

Furthermore, observe the identity

$$\left[\frac{1}{l_j} \left(\sum_{i \in P_j} (1 - y_i^*) + \sum_{i \in N_j} y_i^* \right) \right]^{l_j} = \left[1 - \frac{1}{l_j} \left(\sum_{i \in P_j} y_i^* + \sum_{i \in N_j} (1 - y_i^*) \right) \right]^{l_j}.$$

Since one of the defining inequalities of the linear program was that for all j,

$$\sum_{i \in P_j} y_i^* + \sum_{i \in N_j} (1 - y_i^*) \ge z_j^*,$$

hence we can see that

$$1 - r_j \le \left(1 - \frac{z_j}{l_j} \right)^{l_j}.$$

We can then view r_j as a concave function of z_j, hence we can invoke Lemma 7.8 to get the further bound

$$r_j \ge 1 - \left(1 - \frac{z_j}{l_j} \right)^{l_j} \ge \left[1 - \left(1 - \frac{1}{l_j} \right)^{l}_{j} \right] z_j^*.$$

Finally observe that the function $f(k) = 1 - \left(1 - \frac{1}{k}\right)^k)$ is monotonically decreasing on the positive integers, hence we have the lower bound

$$r_j \geq \left[1 - \left(1 - \frac{1}{l_j}\right)_j^l\right] z_j^* \geq z_j^* \left[\liminf_{k \in \mathbb{N}} 1 - (1-k)^k\right] = \left(1 - \frac{1}{e}\right) z_j^*.$$

With the above bounds, we then have

$$\mathbb{E}[J(z^*)] = \sum_{j=1}^m w_j r_j \geq \left(1 - \frac{1}{e}\right) \sum_{j=1}^m z_j^* \geq \left(1 - \frac{1}{e}\right) \text{OPT},$$

as desired. □

7.4 Combining the Two

To motivate this section, we note that the unbiased random algorithm provided in Section 7.2 and the biased randomized rounding algorithm provided in Section 7.3 are, informally, 'bad' at different places - outputs on which the biased algorithm gets close to optimal are ones where the unbiased does suboptimal, and vice versa. Hence, if we combine the two, we get the best of both worlds.

Theorem 7.10. *Computing both the biased and unbiased random algorithms, and then taking the larger of the two, is a $\frac{3}{4}$-approximation algorithm.*

Proof. Let W_1 and W_2 be the outputs of the biased and unbiased algorithms, respectively, so that we return $\max(W_1, W_2)$. Observe that

$$\begin{aligned}
\mathbb{E}[\max(W_1, W_2)] &\geq \mathbb{E}\left[\frac{1}{2}W_1 + \frac{1}{2}W_2\right] \\
&= \frac{1}{2}\left(\mathbb{E}[W_1] + \mathbb{E}[W_2]\right) \\
&\geq \frac{1}{2}\left(\sum_{j=1}^m w_j z_j^* \left[1 - \left(1 - \frac{1}{l_j}\right)^{l_j}\right] + \sum_{j=1}^m w_j \left[1 - 2^{-l_j}\right]\right) \\
&\geq \frac{1}{2}\sum_{j=1}^m w_j z_j^* \left[2 - 2^{-l_j} - \left(1 - \frac{1}{l_j}\right)^{l_j}\right],
\end{aligned}$$

and so it remains to show that the function

$$f(l_j) = \left[2 - 2^{-l_j} - \left(1 - \frac{1}{l_j}\right)^{l_j}\right] \geq \frac{3}{2}$$

for all positive integers l_j. Observe that $f(1) = f(2) = \frac{3}{2}$ and that for $l_j \geq 3$,

$$f(l_j) \geq \left(1 - \frac{1}{e}\right) + (1 - 2^{-3}) \approx 1.506 \geq \frac{3}{2},$$

hence this algorithm is a $\frac{3}{4}$-approximation algorithm, as desired. □