CS 6550: Randomized Algorithms **Spring 2019**

Lecture 8: Method of Conditional Expectations & Randomized Rounding

February 5, 2019

Lecturer: Eric Vigoda *Scribes: Osman Dai & Alex Forney*

8.1 Maximum Satisfiability (MAX-SAT)

8.1.1 Problem Statement

Let $x_1, x_2, \cdots, x_n \in \{\text{TRUE}, \text{FALSE}\}$ be Boolean random variables and f be a Boolean formula in clausal normal form (CNF), i.e. it is decomposed into clauses as $f = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ where each clause is an OR disjunction of independent literals. (Or simply, it is written as an 'AND of ORs'.)

Example 8.1 *The following Boolean formula is written in CNF:*

$$f = C_1 \wedge C_2 = (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4 \vee \overline{x_5})$$

We want to find an assignment of the Boolean variables that satisfies as many of the clauses $C_1, \cdots, C_m$.

8.1.2 Random Assignment

Consider a random assignment of the variables, such that $\Pr[X_i = \text{TRUE}] = \Pr[X_i = FALSE] = 1/2$ for each $i \in \{1, 2, \cdots, n\}$, independently of all others. Define $Y_1, Y_2, \cdots, Y_m \in \{0, 1\}$ such that

$$Y_i = \begin{cases} 1, & \text{if } C_i = \text{TRUE} \\ 0, & \text{otherwise} \end{cases}$$

Then $Y_i = 0$ if and only if every Boolean variable in C_i has the 'bad' assignment. E.g. in example 8.1 $Y_1 = 0 \iff (X_1, X_2, X_3) = (\text{FALSE}, \text{TRUE}, \text{FALSE})$. Thus for any clause C_i consisting of k literals, the probability of not being satisfied is given as $\Pr[Y_i = 0] = \Pr[C_i = \text{FALSE}] = (1/2)^k$. Then $\Pr[Y_i = 1] = 1 - 2^{-k}$.

Lemma 8.2 *Let f be some Boolean formula in CNF, m the number of clauses and k the minimum number of literals inn any clause. Then there exists some assignment that satisfies at least $m(1-2^{-k})$ of the clauses.*

Proof: Let $Y = \sum Y_i$ denote the number of satisfied clauses. Let k_i denote the number of literals in clause C_i. Then $\mathbb{E}[Y] = \sum \mathbb{E}[Y_i] = \sum \Pr[Y_i = 1] = \sum (1 - 2^{-k}) \geq m(1 - 2^{-k})$. ■

By the fact that every clause has at least 1 literal, i.e. $k \geq 1$, we have $m(1 - 2^{-k}) \geq m/2$ which gives us the following corollary.

Corollary 8.3 *Let f be some Boolean formula in CNF, m the number of clauses. There exists some assignment that satisfies at least $m/2$ of the clauses.*

Algorithm 1: Derandomized 0.5-MAX-SAT

```
input  : Clauses $C_1, C_2, \cdots, C_m$.
output: Boolean variable assignments $x_1, x_2, \cdots, x_n$
1 for $i \leftarrow 1$ to $n$ do
2     Compute $\mathbb{E}[Y|(X_1, X_2, \cdots, X_{i-1}, X_i) = (x_1, x_2, \cdots, x_{i-1}, \text{TRUE})]$ and
       $\mathbb{E}[Y|(X_1, X_2, \cdots, X_{i-1}, X_i) = (x_1, x_2, \cdots, x_{i-1}, \text{FALSE})]$;
3     Pick the largest one and assign $x_i$ accordingly.
```

8.1.3 Derandomization

We can derandomize this algorithm to achieve the same guarantee on the number of satisfied clauses using a deterministic assignment.

Lemma 8.4 *Let k be the mininum number of literals any clause has. Then Algorithm 1 returns an assignment that satisfies at least $(1-2^{-k})m$ clauses.*

Proof: $\mathbb{E}[Y] = \frac{1}{2}\mathbb{E}[Y|X_1 = \text{TRUE}] + \frac{1}{2}\mathbb{E}[Y|X_1 = \text{FALSE}]$. Thus both $\mathbb{E}[Y|X_1 = \text{TRUE}]$ and $\mathbb{E}[Y|X_1 = \text{FALSE}]$ cannot be smaller than $\mathbb{E}[Y]$ and the largest must satisfy $\mathbb{E}Y|X_1 = x_1 \geq \mathbb{E}[Y]$.

We can continue to apply this conditioning recursively, always getting some expectation that is at least as large as the expectation in the previous step. This finally gives us $\mathbb{E}[Y|(X_1, X_2, \cdots, X_n) = (x_1, x_2, \cdots, x_n)] \geq (1-2^{-k})m$, where the value of Y is no longer random as it is conditioned on the assignment of all literals.

Finally, by the proof of Lemma 8.2, we have $\mathbb{E}[Y] \geq (1-2^{-k})m$. Then this final value cannot be less than $(1-2^{-k})m$. ■

8.1.4 Randomized Rounding

8.1.4.1 Integer (Linear) Programming

Recall that a linear program is an optimization problem where our goal is to maximize some linear objective function subject to linear constraints. An integer linear program additionally requires that some or all of the decision variables take integer values.

Given $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$, and $b \in \mathbb{R}^m$, the integer linear programming (ILP) problem can be written in canonical form as

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b \\ & x \geq 0 \\ & x \in \mathbb{Z}^n \end{aligned}$$

8.1.4.2 Reduction

We will now show that ILP is NP-hard via a reduction from SAT.

Before we proceed with the reduction, however, we shall introduce some notation. For a clause C_j in an instance of SAT, let C_j^+ denote the variables appearing in positive form in C_j and C_j^- denote the variables appearing in negative form in C_j. Consider the following example:

Example 8.5 *Suppose we have a clause*

$$C_1 = x_5 \vee \overline{x_3} \vee x_7.$$

Then, $C_1^+ = \{x_5, x_7\}$ and $C_1^- = \{x_3\}$.

Suppose we are given a SAT instance given by a formula f with n variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$. We reduce SAT to ILP in the following way:

- For each SAT variable x_i, create an ILP variable y_i. Additionally, add the constraint that $y_i \in [0, 1]$. Along with the integrality constraint, this will force $y_i \in \{0, 1\}$. We will choose additional constraints so that, When we solve the ILP, $y_i = 1$ corresponds to the assignment $x_i = T$ and $y_i = 0$ corresponds to the assignment $x_i = F$.

- For each SAT clause C_j, create an ILP variable z_j. As above, add the constraint $z_j \in [0, 1]$ so that we have a similar correspondence between $z_j = 1$ and the clause C_j being satisfied and $z_j = 0$ and the clause C_j being unsatisfied.

- For each SAT clause C_j, add the constraint

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

To see the necessity of the third set of constraints, we return to Example 8.5. In order to satisfy clause C_1, we must have at least one of $x_5 = T$, $x_3 = F$, or $x_7 = T$. Using the variables y_5, y_3, and y_7 defined in the reduction, this is equivalent to having at least one of $y_5 = 1$, $y_3 = 0$, or $y_7 = 1$. Additionally, C_j being satisfied corresponds to $z_1 = 1$. The third set of constraints gives us the inequality

$$y_5 + (1 - y_3) + y_7 \geq z_1,$$

so C_1 is satisfied if and only if $y_5 = 1$, $y_3 = 0$, or $y_7 = 1$. Otherwise, it is not.

We have reduced SAT to ILP, which shows that ILP is NP-hard. To apply our ILP to MAX-SAT, we will attempt to maximize the value of $z_1 + \cdots + z_m$, which corresponds to maximizing the number of satisfied clauses in f. Thus, we have the integer linear program

$$\begin{array}{lrl}
\max & \displaystyle\sum_{j=1}^{m} z_j & \\
\text{s.t.} & \displaystyle\sum_{i \in C_j^+} y_i + \sum_{i \in C_j^-} (1 - y_i) \geq z_j & \forall j \in \{1, \ldots, m\} \\
& y_i \in [0, 1] & \forall i \in \{1, \ldots, n\} \\
& z_j \in [0, 1] & \forall j \in \{1, \ldots, m\} \\
& y_i \in \mathbb{Z} & \forall i \in \{1, \ldots, n\} \\
& z_j \in \mathbb{Z} & \forall j \in \{1, \ldots, m\}
\end{array}$$

Solving this integer linear program is equivalent to solving MAX-SAT, so MAX-SAT reduces to ILP.

8.1.4.3 Algorithm

We can drop the integrality constraints in the above integer program in order to obtain a linear program. This linear program is the *LP relaxation* of our problem.

Unlike the integer program, we can solve the LP relaxation in polynomial time, so we would like to use the LP solution to find a solution to the ILP (which, in turn, gives us a valid assignment for MAX-SAT). Let $\hat{y}_i^*, \hat{z}_j^*$ denote the optimal solutions to the LP, which we can find, and let y_i^*, z_i^* denote the optimal solutions to the ILP, which we don't know. Notice that the y_i^*, z_i^* are feasible solutions to the LP relaxation, which implies that

$$\sum_{j=1}^{m} z_j^* \leq \sum_{j=1}^{m} \hat{z}_j^*. \tag{8.1}$$

Algorithm 2: Randomized Rounding

```
input  : MAX-SAT instance $f$ with $n$ variables $x_1, \dots, x_n$ and $m$ clauses $C_1, \dots, C_m$
output: T/F assignment for each $x_i$
1 Formulate the ILP corresponding to $f$ as above;
2 Solve the LP relaxation to obtain an optimal solution $\hat{y}_i^*, \hat{z}_i^*$;
3 for $i = 1, \dots, n$ do
4     Set $x_i = T$ with probability $\hat{y}_i^*$ and $x_i = F$ with probability $1 - \hat{y}_i^*$;
5 Output $x_1, \dots, x_n$;
```

That is, the ILP optimum is at most the LP optimum. This is why we say that the LP is a relaxation of the ILP. With this, we have the following algorithm:

Notice that this algorithm indeed gives a feasible solution to the ILP because our choice of each x_i is equivalent to setting each ILP variable

$$y_i = \begin{cases} 1, & \text{with probability } \hat{y}_i^* \\ 0, & \text{with probability } 1 - \hat{y}_i^*, \end{cases}$$

which is valid because $0 \le \hat{y}_i^* \le 1$.

Theorem 8.6 *Algorithm 2 is a $(1 - 1/e)$-approximation for MAX-SAT.*

Before we prove Theorem 8.6, we introduce additional notation. Let W denote the number of satisfied clauses by the assignment output by the algorithm. For each $j \in \{1, \dots, m\}$, let

$$W_j = \begin{cases} 1, & \text{if clause } C_j \text{ is satisfied} \\ 0, & \text{if clause } C_j \text{ is not satisfied.} \end{cases}$$

Notice that $W = \sum_{j=1}^m W_j$. We also prove the following preliminary results.

Claim 8.7 *Let $k \ge 1$ be an integer. For each $\alpha \in [0, 1]$,*

$$1 - \left(1 - \frac{\alpha}{k}\right)^k \ge 1 - \left(1 - \frac{1}{k}\right)^k \alpha.$$

Proof of Claim 8.7: Define $f : \mathbb{R} \to \mathbb{R}$ by

$$f(\alpha) = 1 - \left(1 - \frac{\alpha}{k}\right)^k.$$

Notice that

$$f''(\alpha) = \frac{-(k-1)}{k} \left(1 - \frac{\alpha}{k}\right)^{k-2} \le 0$$

if $\alpha \in [0, 1]$, so f is concave on $[0, 1]$. Moreover,

$$f(0) = 1 - \left(1 - \frac{0}{k}\right)^k = 0 = 1 - \left(1 - \frac{1}{k}\right)^k \alpha$$

and

$$f(1) = 1 - \left(1 - \frac{1}{k}\right)^k = 1 - \left(1 - \frac{1}{k}\right)^k \alpha.$$

Since $1 - \left(1 - \frac{1}{k}\right)^k \alpha$ is linear on $[0, 1]$, and equals f on the endpoints, it follows that

$$f(\alpha) = 1 - \left(1 - \frac{\alpha}{k}\right)^k \geq 1 - \left(1 - \frac{1}{k}\right)^k \alpha$$

if $\alpha \in [0, 1]$. ■

Lemma 8.8 *Let C_j be a clause, and let k denote the number of literals in C_j. Then,*

$$\mathbb{E}[W_j] \geq \beta_k \hat{z}_j^*,$$

where

$$\beta_k = 1 - \left(1 - \frac{1}{k}\right)^k.$$

Proof of Lemma 8.8: Consider the LP relaxation and its solution $\hat{y}_i^*, \hat{z}_j^*$. By feasibility, we have $0 \leq \hat{z}_j^* \leq 1$ and

$$\sum_{i \in C_j^+} \hat{y}_i^* + \sum_{i \in C_j^-} (1 - \hat{y}_i^*) \geq \hat{z}_j^*.$$

By the AMGM inequality,

$$\left[\prod_{i \in C_j^+} (1 - \hat{y}_i^*) \prod_{i \in C_j^-} \hat{y}_i^*\right]^{1/k} \leq \frac{1}{k}\left[\sum_{i \in C_j^+} (1 - \hat{y}_i^*) + \sum_{i \in C_j^-} \hat{y}_i^*\right] = 1 - \frac{1}{k}\left[\sum_{i \in C_j^+} \hat{y}_i^* + \sum_{i \in C_j^-} (1 - \hat{y}_i^*)\right] \leq 1 - \frac{\hat{z}_j^*}{k}.$$

Hence,

$$\prod_{i \in C_j^+} (1 - \hat{y}_i^*) \prod_{i \in C_j^-} \hat{y}_i^* \leq \left(1 - \frac{\hat{z}_j^*}{k}\right)^k.$$

Thus,

$$\begin{aligned}
\mathbb{E}[W_j] &= \Pr(W_j = 1) \\
&= 1 - \prod_{i \in C_j^+} (1 - \hat{y}_i^*) \prod_{i \in C_j^-} \hat{y}_i^* \\
&\geq 1 - \left(1 - \frac{\hat{z}_j^*}{k}\right)^k && \text{(By the AM-GM inequality)} \\
&\geq 1 - \left(1 - \frac{1}{k}\right)^k \hat{z}_j^* && \text{(By Claim 8.7)} \\
&= \beta_k \hat{z}_j^*.
\end{aligned}$$

■

Proof of Theorem 8.6: For each integer $k \geq 1$, recall that $1 - \frac{1}{k} \leq e^{-1/k}$ and so

$$\left(1 - \frac{1}{k}\right)^k \leq \frac{1}{e}.$$

Hence,

$$\beta_k = 1 - \left(1 - \frac{1}{k}\right)^k \geq 1 - \frac{1}{e}.$$

For each $j \in \{1, \ldots, m\}$, let k_j denote the number of literals in clause C_j. Applying the above observation to each k_j, the expected number of satisfied clauses is given by

$$\mathbb{E}[W] = \mathbb{E}\left[\sum_{j=1}^{m} W_j\right] = \sum_{j=1}^{m} \mathbb{E}[W_j] \geq \sum_{j=1}^{m} \beta_{k_j} \hat{z}_j^* \geq \left(1 - \frac{1}{e}\right) \sum_{j=1}^{m} \hat{z}_j^* \geq \left(1 - \frac{1}{e}\right) \sum_{j=1}^{m} z_j^*,$$

where we have used Equation 8.1 to obtain the final inequality. However, $\sum_{j=1}^{m} z_j^*$ is the maximum number of satisfied clauses in the MAX-SAT instance. This implies that Algorithm 2 is a $(1 - 1/e)$-approximation for MAX-SAT. ■

8.1.5 A Third Algorithm

We have presented two algorithms for approximating MAX-SAT. On classes of size k, Algorithm 1 gives a $(1 - 2^{-k})$-approximation, while Algorithm 2 yields a $\beta_k = 1 - \left(1 - \frac{1}{k}\right)^k$-approximation. Thus, we have the following approximation factors for MAX-kSAT:

k	Simple	LP	max	avg
1	1/2	1	1	3/4
2	3/4	3/4	3/4	3/4
3	7/8	19/27	7/8	$> 3/4$
≥ 4			$1 - 2^{-k}$	

To obtain the best results, we may as well choose the best of the two algorithms. This idea was proposed by Goemans and Williamson [1].

```
Algorithm 3: Best of Two
  input  : MAX-SAT instance $f$ with $n$ variables $x_1, \ldots, x_n$ and $m$ clauses $C_1, \ldots, C_m$
  output: T/F assignment for each $x_i$
1 Run Algorithm 1 to get assignments $x_1^1, \ldots, x_n^1$;
2 Run Algorithm 2 to get assignments $x_1^2 \ldots, x_n^2$;
3 Output the assignment that maximizes the number of satisfied clauses;
```

Theorem 8.9 *Algorithm 3 is a 3/4-approximation algorithm for MAX-SAT.*

Proof: Let m_1 denote the expected number of satisfied clauses by Algorithm 1, m_2 denote the expected number of satisfied clauses by Algorithm 2, and m^* denote the optimal number of satisfied clauses. Also, let $\hat{y}_i^*, \hat{z}_j^*$ denote the optimal solution to the LP relaxation, and, for each integer $k \geq 1$, let S_k be the set of clauses with exactly k literals.

Since $0 \leq \hat{z}_j^* \leq 1$,

$$m_1 = \sum_{k=1}^{\infty} \sum_{C_j \in S_k} (1 - 2^{-k}) \geq \sum_{k=1}^{\infty} \sum_{C_j \in S_k} (1 - 2^{-k}) \hat{z}_j^*$$

and

$$m_2 \geq \sum_{k=1}^{\infty} \sum_{C_j \in S_k} \left(1 - \left(1 - \frac{1}{k}\right)^k\right) \hat{z}_j^*.$$

Notice that

$$(1 - 2^{-1}) + \left(1 - \left(1 - \frac{1}{1}\right)^1\right) = (1 - 2^{-2}) + \left(1 - \left(1 - \frac{1}{2}\right)^2\right) = \frac{3}{2}$$

and, for each integer $k \geq 3$,

$$\left(1-2^{-k}\right)+\left(1-\left(1-\frac{1}{k}\right)^{k}\right) \geq \frac{7}{8}+1-\frac{1}{e} \geq \frac{3}{2}.$$

Hence, for each integer $k \geq 1$, we have

$$\left(1-2^{-k}\right)+\left(1-\left(1-\frac{1}{k}\right)^{k}\right) \geq \frac{3}{2}.$$

Thus,

$$\max\{m_1, m_2\} \geq \frac{m_1+m_2}{2} \geq \sum_{k=1}^{\infty} \sum_{C_j \in S_k} \frac{\left(1-2^{-k}\right)+\left(1-\left(1-\frac{1}{k}\right)^{k}\right)}{2} \hat{z}_j^* \geq \sum_{k=1}^{\infty} \sum_{C_j \in S_k} \frac{3}{4} \hat{z}_j^* \geq \frac{3}{4} \sum_{j=1}^{m} \hat{z}_j^* \geq \frac{3}{4} m^*,$$

which shows that Algorithm 3 is a 3/4-approximation algorithm for MAX-SAT. ■

References

[1] Goemans, Michel X. and Williamson, David P. New 3/4-Approximation Algorithms for the Maximum Satisfiability Problem. *SIAM Journal on Discrete Mathematics*, 7(4):656-666, Nov 1994.