Approximations for MAX-SAT Problem

Chihao Zhang

BASICS, Shanghai Jiao Tong University

Oct. 23, 2012

The Weighted MAX-SAT Problem

Input:	n Boolean variables $x_1, \ldots, x_n$, a CNF $\varphi = \bigwedge_{j=1}^{m} C_j$ and a nonnegative weight w_j for each C_j.
Problem:	Find an assignment to x_i-s that maximizes the weight of satisfied clauses.

The Weighted MAX-SAT Problem

Input:	n Boolean variables $x_1, \ldots, x_n$, a CNF $\varphi = \bigwedge_{j=1}^{m} C_j$ and a nonnegative weight w_j for each C_j.
Problem:	Find an assignment to x_i-s that maximizes the weight of satisfied clauses.

- Obviously *NP*-hard.

Flipping a Coin

- A very straightforawrd randomized approximation algorithm is to set each x_i to `true` independently with probability $1/2$.

Flipping a Coin

- A very straightforawrd randomized approximation algorithm is to set each x_i to `true` independently with probability $1/2$.

Theorem

Setting each x_i to $\mathtt{true}$ with probability $1/2$ independently gives a randomized $\frac{1}{2}$-approximation algorithm for weighted MAX-SAT.

Proof

Proof.

Let W be a random variable that is equal to the total weight of the satisfied clauses. Define an indicator random variable Y_j for each clause C_j such that $Y_j = 1$ if and only if C_j is satisfied. Then

$$W = \sum_{j=1}^{m} w_j Y_j$$

We use OPT to denote value of optimum solution, then

$$E[W] = \sum_{j=1}^{m} w_j E[Y_j] = \sum_{j=1}^{m} w_j \cdot \Pr[\text{clause } C_j \text{ satisfied}]$$

Proof (cont'd)

Since each variable is set to `true` independently, we have

$$\Pr[\text{clause } C_j \text{ satisfied}] = \left(1 - \left(\frac{1}{2}\right)^{l_j}\right) \geq \frac{1}{2}$$

where l_j is the number of literals in clause C_j. Hence,

$$E[W] \geq \frac{1}{2}\sum_{j=1}^{m} w_j \geq \frac{1}{2}\mathrm{OPT}.$$

Proof (cont'd)

Since each variable is set to `true` independently, we have

$$\Pr[\text{clause } C_j \text{ satisfied}] = \left(1 - \left(\frac{1}{2}\right)^{l_j}\right) \geq \frac{1}{2}$$

where l_j is the number of literals in clause C_j. Hence,

$$E[W] \geq \frac{1}{2}\sum_{j=1}^{m} w_j \geq \frac{1}{2}\mathrm{OPT}.$$

From the analysis, we can see that the performance of the algorithm is better on instances consisting of long clauses.

Derandomization by Conditional Expectation

The previous randomized algorithm can be derandomized. Note that

$$\begin{aligned} E[W] &= E[W \mid x_1 \leftarrow \texttt{true}] \cdot \Pr[x_1 \leftarrow \texttt{true}] \\ &\quad + E[W \mid x_1 \leftarrow \texttt{false}] \cdot \Pr[x_1 \leftarrow \texttt{false}] \\ &= \frac{1}{2}(E[W \mid x_1 \leftarrow \texttt{true}] + E[W \mid x_1 \leftarrow \texttt{false}]) \end{aligned}$$

Derandomization by Conditional Expectation

The previous randomized algorithm can be derandomized. Note that

$$\begin{aligned} E[W] &= E[W \mid x_1 \leftarrow \texttt{true}] \cdot \Pr[x_1 \leftarrow \texttt{true}] \\ &\quad + E[W \mid x_1 \leftarrow \texttt{false}] \cdot \Pr[x_1 \leftarrow \texttt{false}] \\ &= \frac{1}{2}(E[W \mid x_1 \leftarrow \texttt{true}] + E[W \mid x_1 \leftarrow \texttt{false}]) \end{aligned}$$

We set b_1 `true` if $E[W \mid x_1 \leftarrow \texttt{true}] \geq E[W \mid x_1 \leftarrow \texttt{false}]$ and set b_1 `false` otherwise. Let the value of x_1 be b_1.

Derandomization by Conditional Expectation

The previous randomized algorithm can be derandomized. Note that

$$\begin{aligned} E[W] &= E[W \mid x_1 \leftarrow \texttt{true}] \cdot \Pr[x_1 \leftarrow \texttt{true}] \\ &\quad + E[W \mid x_1 \leftarrow \texttt{false}] \cdot \Pr[x_1 \leftarrow \texttt{false}] \\ &= \frac{1}{2}(E[W \mid x_1 \leftarrow \texttt{true}] + E[W \mid x_1 \leftarrow \texttt{false}]) \end{aligned}$$

We set b_1 `true` if $E[W \mid x_1 \leftarrow \texttt{true}] \geq E[W \mid x_1 \leftarrow \texttt{false}]$ and set b_1 `false` otherwise. Let the value of x_1 be b_1.

Continue this process until all b_i are found, i.e., all n variables have been set.

Derandomization by Conditional Expectation

This is a deterministic $\frac{1}{2}$-approximation algorithm because of the following two facts:

1. $E[W \mid x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i]$ can be computed in polynomial time for fixed $b_1, \ldots, b_i$.
2. $E[W \mid x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i, x_{i+1} \leftarrow b_{i+1}] \geq E[W \mid x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i]$ for all i.

Flipping biased coins

- Previously, we set each x_i `true` or `false` with probability $\frac{1}{2}$ independently. $\frac{1}{2}$ is nothing special here.

Flipping biased coins

- Previously, we set each x_i `true` or `false` with probability $\frac{1}{2}$ independently. $\frac{1}{2}$ is nothing special here.
- In the following, we set each x_i `true` with probability $p \geq \frac{1}{2}$.

Flipping biased coins

- Previously, we set each x_i `true` or `false` with probability $\frac{1}{2}$ independently. $\frac{1}{2}$ is nothing special here.
- In the following, we set each x_i `true` with probability $p \geq \frac{1}{2}$.

We first consider the case that no clause is of the form $C_j = \bar{x}_i$.

Flipping biased coins

- Previously, we set each x_i `true` or `false` with probability $\frac{1}{2}$ independently. $\frac{1}{2}$ is nothing special here.
- In the following, we set each x_i `true` with probability $p \geq \frac{1}{2}$.

We first consider the case that no clause is of the form $C_j = \bar{x}_i$.

Lemma

If each x_i is set to true with probability $p \geq 1/2$ independently, then the probability that any given clause is satisfied is at least $\min(p, 1 - p^2)$ for instances with no negated unit clauses.

Flipping biased coins (cont'd)

Armed with previous lemma, we then maximize $\min(p, 1-p^2)$, which is achieved when $p = 1 - p^2$, namely $p = \frac{1}{2}(\sqrt{5}-1) \approx 0.618$.

Flipping biased coins (cont'd)

Armed with previous lemma, we then maximize $\min(p, 1-p^2)$, which is achieved when $p = 1 - p^2$, namely $p = \frac{1}{2}(\sqrt{5}-1) \approx 0.618$.

We need more effort to deal with negated unite clauses, i.e., $C_j = \bar{x}_i$ for some j.

Flipping biased coins (cont'd)

Armed with previous lemma, we then maximize $\min(p, 1 - p^2)$, which is achieved when $p = 1 - p^2$, namely $p = \frac{1}{2}(\sqrt{5} - 1) \approx 0.618$.

We need more effort to deal with negated unite clauses, i.e., $C_j = \bar{x}_i$ for some j.

We distinguish between two cases:

Flipping biased coins (cont'd)

Armed with previous lemma, we then maximize $\min(p, 1-p^2)$, which is achieved when $p = 1 - p^2$, namely $p = \frac{1}{2}(\sqrt{5}-1) \approx 0.618$.

We need more effort to deal with negated unite clauses, i.e., $C_j = \bar{x}_i$ for some j.

We distinguish between two cases:

1. Assume $C_j = \bar{x}_i$ and there is no clause such that $C = x_i$. In this case, we can introduce a new variable y and replace the appearance of $\bar{x}_i$ in φ by y and the appearance of x_i by $\bar{y}$.

Flipping biased coins (cont'd)

2. $C_j = \bar{x}_i$ and some clause $C_k = x_i$. W.L.O.G we assume $w(C_j) \le w(C_k)$. Note that for any assignment, C_j and C_k cannot be satisfied simultaneously. Let v_i be the weight of the unit clause $\bar{x}_i$ if it exists in the instance, and let v_i be zero otherwise, we have

$$\mathrm{OPT} \le \sum_{j=1}^{m} w_j - \sum_{i=1}^{n} v_i$$

Flipping biased coins (cont'd)

2. $C_j = \bar{x}_i$ and some clause $C_k = x_i$. W.L.O.G we assume $w(C_j) \le w(C_k)$. Note that for any assignment, C_j and C_k cannot be satisfied simultaneously. Let v_i be the weight of the unit clause $\bar{x}_i$ if it exists in the instance, and let v_i be zero otherwise, we have

$$\mathrm{OPT} \le \sum_{j=1}^{m} w_j - \sum_{i=1}^{n} v_i$$

We set each x_i `true` with probability $p = \frac{1}{2}(\sqrt{5} - 1)$, then

$$\begin{aligned} E[W] &= \sum_{j=1}^{m} w_j E[Y_j] \\ &\ge p \cdot \left(\sum_{j=1}^{m} w_j - \sum_{i=1}^{n} v_i \right) \\ &\ge p \cdot \mathrm{OPT} \end{aligned}$$

The Use of Linear Program

Integer Program Characterization:

$$
\begin{array}{rll}
\text{maximize} & \sum_{j=1}^{m} w_j z_j & \\
\text{subject to} & \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq z_j, & \forall C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \bar{x}_i, \\
& y_i \in \{0, 1\}, & i = 1, \ldots, n, \\
& z_j \in \{0, 1\}, & j = 1, \ldots, m.
\end{array}
$$

where y_i indicate the assignment of variable x_i and z_j indicates whether clause C_j is satisfied.

The Use of Linear Program

Linear Program Relaxation:

$$\begin{array}{rll} \text{maximize} & \sum_{j=1}^{m} w_j z_j & \\ \text{subject to} & \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq z_j, & \forall C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \bar{x}_i, \\ & 0 \leq y_i \leq 1, & i = 1, \ldots, n, \\ & 0 \leq z_j \leq 1, & j = 1, \ldots, m. \end{array}$$

where y_i indicate the assignment of variable x_i and z_j indicates whether clause C_j is satisfied.

Flipping Different Coins

- Let (y^*, z^*) be an optimal solution of the linear program.

Flipping Different Coins

- Let (y^*, z^*) be an optimal solution of the linear program.
- We set x_i to `true` with probability y_i^*.

Flipping Different Coins

- Let (y^*, z^*) be an optimal solution of the linear program.
- We set x_i to `true` with probability y_i^*.
- This can be viewed as flipping different coins for every variable.

Flipping Different Coins

- Let (y^*, z^*) be an optimal solution of the linear program.
- We set x_i to `true` with probability y_i^*.
- This can be viewed as flipping different coins for every variable.

Theorem

Randomized rounding gives a randomized $(1 - \frac{1}{e})$-approximation algorithm for MAX SAT.

Analysis

$$
\begin{aligned}
& \Pr[\text{clause } C_j \text{ not satisfied}] \\
= \quad & \prod_{i \in P_j} (1 - y_i^*) \prod_{i \in N_j} y_i^* \\
\leq \quad & \boxed{\left[\frac{1}{l_j} \left(\sum_{i \in P_j} (1 - y_i^*) + \sum_{i \in N_j} y_i^* \right) \right]^{l_j}} \\
= \quad & \left[1 - \frac{1}{l_j} \left(\sum_{i \in P_j} y_i^* + \sum_{i \in N_j} (1 - y_i^*) \right) \right]^{l_j} \leq \left(1 - \frac{z_j^*}{l_j} \right)^{l_j}
\end{aligned}
$$

Analysis

$$
\begin{aligned}
& \Pr[\text{clause } C_j \text{ not satisfied}] \\
= \quad & \prod_{i \in P_j} (1 - y_i^*) \prod_{i \in N_j} y_i^* \\
\leq \quad & \left[\frac{1}{l_j} \left(\sum_{i \in P_j} (1 - y_i^*) + \sum_{i \in N_j} y_i^* \right) \right]^{l_j} \\
= \quad & \left[1 - \frac{1}{l_j} \left(\sum_{i \in P_j} y_i^* + \sum_{i \in N_j} (1 - y_i^*) \right) \right]^{l_j} \leq \left(1 - \frac{z_j^*}{l_j} \right)^{l_j}
\end{aligned}
$$

Arithmetic-Geometric Mean Inequality

Analysis (cont'd)

$$
\begin{aligned}
& \Pr[\text{clause } C_j \text{ satisfied}] \\
\geq \quad & 1 - \left(1 - \frac{z_j^*}{l_j}\right)^{l_j} \\
\geq \quad & \boxed{\left[1 - \left(1 - \tfrac{1}{l_j}\right)^{l_j}\right] z_j^*}
\end{aligned}
$$

Analysis (cont'd)

$$
\begin{aligned}
& \Pr[\text{clause } C_j \text{ satisfied}] \\
\geq \; & 1 - \left(1 - \frac{z_j^*}{l_j}\right)^{l_j} \\
\geq \; & \left[1 - \left(1 - \tfrac{1}{l_j}\right)^{l_j}\right] z_j^*
\end{aligned}
$$

Jensen's Inequality

Analysis (cont'd)

$$
\begin{aligned}
& \Pr[\text{clause } C_j \text{ satisfied}] \\
\geq \;& 1 - \left(1 - \frac{z_j^*}{l_j}\right)^{l_j} \\
\geq \;& \left[1 - \left(1 - \tfrac{1}{l_j}\right)^{l_j}\right] z_j^*
\end{aligned}
$$

Jensen's Inequality

Therefore, we have

$$
\begin{aligned}
E[W] &= \sum_{j=1}^{m} w_j \Pr[\text{clause } C_j \text{ satisfied}] \\
&\geq \sum_{j=1}^{m} w_j z_j^* \left[1 - \left(1 - \frac{1}{l_j}\right)^{l_j}\right] \\
&\geq \left(1 - \frac{1}{e}\right) \cdot \mathrm{OPT}
\end{aligned}
$$

Choosing the better of two

- The randomized rounding algorithm performs better when l_j-s are small.($\left(1 - \frac{1}{k}\right)^k$ is nondecreasing)

.

Choosing the better of two

- The randomized rounding algorithm performs better when l_j-s are small.($\left(1 - \frac{1}{k}\right)^k$ is nondecreasing)
- The unbiased randomized algorithm performs better when l_j-s are large.

.

Choosing the better of two

- The randomized rounding algorithm performs better when l_j-s are small.($\left(1-\frac{1}{k}\right)^k$ is nondecreasing)
- The unbiased randomized algorithm performs better when l_j-s are large.
- We will combine them together.

Choosing the better of two

- The randomized rounding algorithm performs better when l_j-s are small.($\left(1-\frac{1}{k}\right)^k$ is nondecreasing)
- The unbiased randomized algorithm performs better when l_j-s are large.
- We will combine them together.

Theorem

Choosing the better of the two solutions given by the two algorithms yields a randomized $\frac{3}{4}$-approximation algorithm for MAX SAT.

Analysis

Let W_1 and W_2 be the r.v. of value of solution of randomize rounding algorithm and unbiased randomized algorithm respectively. Then

$$
\begin{aligned}
E[\max(W_1, W_2)] &\geq E[\frac{1}{2}W_1 + \frac{1}{2}W_2] \\
&\geq \frac{1}{2}\sum_{j=1}^{m} w_j z_j^* \left[1 - \left(1 - \frac{1}{l_j}\right)^{l_j}\right] + \frac{1}{2}\sum_{j=1}^{m} w_j \left(1 - 2^{-l_j}\right) \\
&\geq \sum_{j=1}^{m} w_j z_j^* \left[\frac{1}{2}\left(1 - \left(1 - \frac{1}{l_j}\right)^{l_j}\right) + \frac{1}{2}\left(1 - 2^{-l_j}\right)\right] \\
&\geq \frac{3}{4} \cdot \mathrm{OPT}
\end{aligned}
$$