

**Today's Topic:** Randomized Rounding Algorithm for MAX-SAT

**Problem Statement:** Given a CNF-formula  $\phi = c_1 \wedge c_2 \wedge \dots \wedge c_m$ , find an assignment that satisfies the maximum number of clauses. Exact solution of this problem is known to be NP-hard. We will improve on the randomized approximation algorithm of Lecture 2 by using the Randomized Rounding technique introduced in the previous lecture.

## 1 A randomized algorithm for MAX-SAT

MAX-SAT is a generalization of the MAX-3CNF problem that we discussed in Lecture 2. In MAX-3CNF, each formula  $\phi$  is a conjunction of clauses  $c_i$ , such that each  $c_i$  is a disjunction of three literals,  $x_{i_1}, x_{i_2}, x_{i_3}$ , where either  $x_{i_j}$  or  $\neg x_{i_j}$  appears in  $c_i$ . In the case of MAX-SAT, each  $c_i$  is a disjunction of finitely many literals (or their negations), but the number of literals in each clause may vary.

Recall from Lecture 2 that we presented and analyzed the following randomized algorithm that with high probability was able to satisfy  $7/8$  of the clauses of a MAX-3CNF formula.

**Algorithm 1.** Given input  $\phi(x_1, \dots, x_n) = c_1 \wedge \dots \wedge c_m$ , uniformly at random for each  $1 \leq i \leq n$ , set  $x_i = 1$  with half probability, and  $x_i = 0$  with half probability.

It turns out that this algorithm can be derandomized, and the resulting deterministic algorithm still satisfies  $7/8$  of the clauses of  $\phi$ . This is best possible in the sense that the existence of a  $7/8 + \epsilon$  approximation algorithm implies  $\mathbf{P} = \mathbf{NP}$ .

Now suppose we provide a CNF-formula as input to Algorithm 1, instead of a 3CNF formula. Look at clause  $c_j$ . Say  $c_j$  has  $l_j$  literals. Then

$$\Pr[c_j \text{ is satisfied}] = 1 - 2^{-l_j}.$$

Now define the following random variables.

$$\begin{aligned} X &\rightarrow \# \text{ of clauses satisfied} \\ X_i &\rightarrow = 1 \text{ if } c_i \text{ is satisfied} \\ &= 0 \text{ else.} \end{aligned}$$

Then  $X = \sum X_i$ . So consider the expected value of  $X$ :

$$\begin{aligned} E[X] &= \sum E[X_i] \\ &= \sum_{j=1}^m (1 - 2^{-l_j}) \\ &\geq m/2 \end{aligned}$$

where  $l_j$  equals the number of literals in  $c_j$ .

Since the expected number of clauses is actually an integer, and the the maximum value  $X$  takes is at most  $m$ , we can show that the above algorithm produces an assignment that satisfies at least  $m/2$  clauses with reasonable probability.

In general, Algorithm 1 gives us a  $1/2$ -approximation to the MAX-SAT problem. If each clause has exactly  $k$  variables, then we get a  $(1 - \frac{1}{2^k})$ -approximation.

## 2 Randomized Rounding

We will use the same technique as the previous lecture: formulate the problem as an Integer Linear Program, then relax the ILP to a Linear Program, and use Randomized Rounding to approximate the optimal solution of the ILP once we have a solution to the LP.

Suppose each clause  $c_i$  is of form  $\langle x_{i_1} \vee \overline{x_{i_2}} \vee \dots \vee x_{i_n} \rangle$ , and  $\phi$  has free variables  $x_1, \dots, x_m$ . For a clause  $c_i$  let  $P_i$  be the set of all positive variables in that clause and let  $N_i$  be the set of all negative variables in that clause.

Our Integer Linear Program then is

$$\begin{aligned} & \text{Maximize } \sum_{i=1}^m y_i \\ & \text{Subject to } (\forall j) \sum_{x \in P_j} x + \sum_{x \in N_j} (1 - x) \geq y_j \\ & \quad x_i, y_i \in \{0, 1\} \end{aligned}$$

Now we relax the constraints. Instead of requiring that  $x_i, y_i \in \{0, 1\}$ , we require only that  $0 \leq x_i, y_i \leq 1$ . Let  $OPT_{ILP}$  be the optimal value for the ILP, and  $OPT_{LP}$  be the optimal value for LP. Then  $OPT_{LP} \geq OPT_{ILP}$ .

Suppose  $\langle x_1^*, x_2^*, \dots, x_n^*, y_1^*, \dots, y_m^* \rangle$  is an optimal solution for the linear program. Our ideal strategy would be to satisfy each clause  $c_i$  with probability  $y_i^*$ . Then  $E[\text{clauses satisfied}] \geq OPT_{LP}$ . Consider the following modification of Algorithm 1 using randomized rounding.

**Algorithm 2.** For each  $x_i$ , set  $x_i = 1$  with probability  $x_i^*$ , and set  $x_i = 0$  with probability  $1 - x_i^*$ .

Define the following random variable:

$$\begin{aligned} Z_i & \rightarrow= 1 \text{ if } c_i \text{ satisfied} \\ & = 0 \text{ else} \end{aligned}$$

Then  $\Pr[Z_i = 1] = \Pr[c_i \text{ satisfied}]$ .

For ease of exposition, assume  $c_i$  has only positive literals. Suppose  $c_i$  has  $l$  literals, so  $c_i = \langle x_{i_1} \vee \dots \vee x_{i_l} \rangle$ . Then

$$\Pr[c_i \text{ not satisfied}] = \prod_{k=1}^l (1 - x_{i_k}^*).$$

From the LP, there is a constraint corresponding to this clause  $c_i$ , as follows:

$$\sum_{k=1}^l x_{i_k} \geq y_i.$$

Therefore we have

$$\begin{aligned} \sum x_i^* & \geq y_i^* \\ \sum (1 - x_i^*) & \leq l - y_i^*. \end{aligned}$$

Using that, we obtain the following:

$$\begin{aligned} \Pr[c_i \text{ is not satisfied}] &= \prod_{k=1}^l (1 - x_{i_k}^*) \\ &\leq \left(\frac{l - y_i^*}{l}\right)^l \\ &= \left(1 - \frac{y_i^*}{l}\right)^l. \end{aligned}$$

We can use this probability that each  $c_i$  is not satisfied to calculate the probability of its being satisfied.

$$\begin{aligned} \Pr[c_i \text{ is satisfied}] &\geq 1 - \left(1 - \frac{y_i^*}{l}\right)^l \\ &\geq \left(1 - \frac{1}{e}\right) y_i^*. \end{aligned}$$

From this we can calculate the expected value.

$$\begin{aligned} \mathbb{E}[\# \text{ of clauses satisfied}] &\geq \sum_{i=1}^m \left(1 - \frac{1}{e}\right) y_i^* \\ &\geq \left(1 - \frac{1}{e}\right) \sum y_i^* \\ &= \left(1 - \frac{1}{e}\right) OPT_{LP} \\ &\geq \left(1 - \frac{1}{e}\right) OPT_{ILP} \\ &\simeq 0.6 \cdot OPT_{ILP}. \end{aligned}$$

A 0.6-approximation is better than a 1/2-approximation.

### 3 Combining Algorithm 1 and Algorithm 2

Consider the satisfaction probability we just derived:

$$\Pr[c_i \text{ is satisfied}] \geq 1 - \left(1 - \frac{y_i^*}{l}\right)^l.$$

If the number of literals  $l$  grows large, the probability that  $c_i$  is satisfied decreases. This is because our relaxation of the ILP is less and less useful as  $l$  gets bigger. So that is a weakness of Algorithm 2. On the other hand, Algorithm 1 performs better when the number of literals in a clause increases.

This suggests following algorithm.

**Algorithm 3.** Toss a coin. If heads, run Algorithm 1. If tails, run Algorithm 2.

Let's figure out what fraction of the clauses are satisfied by Algorithm 3.

$$\Pr[\text{ Clause } C_j \text{ is satisfied}] = \frac{1}{2}(1 - 2^{l_j}) + \frac{1}{2}(1 - (1 - y_j^*)^{l_j})$$

When  $l_j = 1$ , this quantity is  $\frac{1}{2}(\frac{1}{2} + y_j^*) \geq \frac{3}{4}y_j^*$ . When  $l_j > 1$  this quantity is at least

$$\begin{aligned} \frac{1}{2}(1 - 2^{l_j}) + \frac{1}{2}(1 - 1/e)y_j^* &\geq \left[ \frac{1}{2}(1 - 2^{l_j}) + \frac{1}{2}(1 - 1/e) \right] y_j^* \\ &\geq \frac{3}{4}y_j^* \end{aligned}$$

Thus the expected number of clauses satisfied by the algorithm is  $\sum \frac{3}{4}y_j^* \geq \frac{3}{4}OPT_{LP} \geq \frac{3}{4}OPT_{ILP}$ .