Introduction to Randomized Algorithms

1. Introduction
2. Max-3-CNF satisfiability
3. The general Max-CNF problem
4. Monte-Carlo methods
5. Las Vegas algorithms

1. Introduction

Consider the following problem:
Instance: Real numbers $x_1, \ldots, x_n$.
Problem: Find x_i with $x_i \geq \overline{x}$, where $\overline{x}$ is the $\lfloor n/2 \rfloor$-smallest element.

A deterministic algorithm takes $\Theta(n)$ steps.

Let x_i and x_j with $i \neq j$ be chosen randomly. Assume $x_i \geq x_j$. One has:

$$\mathbf{P}[x_i \geq \overline{x}] \geq 1/2 \qquad \Longleftrightarrow \qquad \mathbf{P}[x_i \leq \overline{x}] \leq 1/2.$$

$$\Downarrow$$

$$\mathbf{P}[(x_i \leq \overline{x}) \text{ and } (x_j \leq \overline{x})] \leq 1/4.$$

Since $x_i \geq x_j$ we get:

$$\mathbf{P}[(x_i \leq \overline{x}) \text{ and } (x_j \leq \overline{x})] = \mathbf{P}[x_i \leq \overline{x}] \leq 1/4.$$

Similarly, performing this choice k times delivers $x_{i_1}, \ldots, x_{i_k}$ so that for $\tilde{x} = \max\{x_{i_1}, \ldots, x_{i_k}\}$ it holds

$$\mathbf{P}[\tilde{x} \leq \overline{x}] \leq 2^{-k}.$$

Hence, the probability for $\tilde{x} \geq \overline{x}$ is large (at least $1 - 2^{-k}$).

If $k = 10 \Rightarrow \mathbf{P}[\tilde{x} \geq \overline{x}] \geq 0.999$.
If $k = 20 \Rightarrow \mathbf{P}[\tilde{x} \geq \overline{x}] \geq 0.999999$.

The running time of this method does not depend on n !

2. The MAX-3-CNF problem:

Instance: A function $f(x_1, \ldots, x_n) = C_1 \wedge \cdots \wedge C_m$ in CNF, with each clause having exactly 3 literals.
Problem: Find a truth assignment to the variables $x_1, \ldots, x_n$ so that the number of satisfied clauses is maximum.

We set independently each variable to 0 with prob. 1/2 and to 1 with prob. 1/2. Assuming w.l.o.g. that no clause has a variable and its negation, the settings of 3 literals in a clause is independent.
$\Rightarrow$ a clause is not satisfied with prob. $1/2^3 = 1/8$.

Theorem 1 *The above algorithm has approximation rate* $8/7$*.*

Proof.
Let a variable Y_i be defined as follows:

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

Since $\mathbf{P}[Y_i = 1] = 1 - 1/8 = 7/8$, $E[Y_i] = 7/8 \cdot 1 + 1/8 \cdot 0 = 7/8$.
So, for $Y = Y_1 + Y_2 + \cdots + Y_m$ (= # of satisfied clauses) one has

$$\begin{aligned} E[Y] &= E\left[\sum_{i=1}^{m} Y_i\right] \\ &= \sum_{i=1}^{m} E[Y_i] \qquad \text{(linearity of expectation)} \\ &= \sum_{i=1}^{m} 7/8 \\ &= 7m/8. \end{aligned}$$

So, the approx. rate of the method is at most $m/(7m/8) = 8/7$. $\square$

3. The general MAX-CNF problem:

Instance: A function $f(x_1, \dots, x_n) = C_1 \wedge \cdots \wedge C_m$ in CNF.
Problem: Find a truth assignment to the variables $x_1, \dots, x_n$ so that the number of satisfied clauses is maximum.

Let M^* be the maximum number of satisfied clauses. We construct an algorithm with approximation rate 3/4.

Method 1:
Set $x_i = T$ for $i = 1, \dots, n$ independently with probability 1/2.
$\Rightarrow$ a clause with k literals is not satisfied with prob. $1/2^k$.

Let n_1 denote the number of clauses satisfied by this method. One has $n_1 \geq (3/4)M^*$, if each clause consists of at least 2 literals.

Method 2:
We formulate the MAX-CNF problem as an IP (Integer Programming) problem:

- For each clause C_i we introduce a binary variable z_i so that $z_i = 1 \Leftrightarrow C_i$ is satisfied.
- For each variable x_i we introduce a binary variable y_i so that $y_i = 1 \Leftrightarrow x_i = T$.

Denote by S_i^+ (resp. S_i^-) the set of all variables in C_i that are not negated (resp. are negated).

IP problem

$$\begin{array}{ll} \text{maximize} & \sum_{j=1}^{m} z_j \\ \text{subject to} & \sum_{S_j^+} y_i + \sum_{S_j^-}(1-y_i) \geq z_j, \quad j=1,\ldots,m \\ & y_i, z_j \in \{0,1\}, \qquad i=1,\ldots,n \qquad j=1,\ldots,m. \end{array}$$

We relax the conditions $y_i, z_j \in \{0,1\}$ with $y_i, z_j \in [0,1]$ and obtain an LP (Linear Programming) problem.

Let $\hat{y}_i$ and $\hat{z}_i$ be a solution to the LP. Obviously,

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

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

Lemma 1 *Let C_j be a clause with k literals. Then*

$$\mathbf{Pr}[C_j \textit{ is satisfied}] \geq \beta_k \hat{z}_j.$$

Proof: W.l.o.g. we can assume that no variable in C_j is negated, i.e. $C_j = x_1 \vee \cdots \vee x_k$. Note that $\mathbf{Pr}[x_i = 1] = \hat{y}_i$.

Since the LP restrictions are satisfied,

$$\hat{y}_1 + \cdots + \hat{y}_k \geq \hat{z}_j \quad \Leftrightarrow \quad \sum_{i=1}^{k}(1-\hat{y}_i) \leq k - \hat{z}_j.$$

This implies $\Pi_{i=1}^{k}(1-\hat{y}_i) \leq \Pi_{i=1}^{k}\Sigma_{i=1}^{k}\frac{1-\hat{y}_i}{k} \leq \Pi_{i=1}^{k}\frac{k-\hat{z}_j}{k}$, so

$$\begin{aligned} \mathbf{Pr}[C_j \text{ is satisfied}] &= 1 - \prod_{i=1}^{k}(1-\hat{y}_i) \geq 1 - \prod_{i=1}^{k}(1-\hat{z}_j/k) \\ &= 1 - (1-\hat{z}_j/k)^k \geq \beta_k \hat{z}_j. \qquad \Box \end{aligned}$$

Let C^k denote the set of all clauses consisting of k literals. Then

$$\begin{aligned} n_2 &= \sum_{k\geq 1} \mathbf{Ex}[|\{C_j \in C^k \mid C_j \text{ is satisfied}\}|] \\ &= \sum_{k\geq 1} \sum_{C_j\in C^k} \mathbf{Pr}[C_j \text{ is satisfied}] \geq \sum_{k\geq 1} \sum_{C_j\in C^k} \beta_k \hat{z}_j. \end{aligned}$$

Algorithm 1 MAX-CNF;

1. Apply Method 1 and compute n_1.
2. Apply Method 2 and compute n_2.
3. Choose the best solution out of those.

Theorem 2 *It holds*

$$\max\{n_1, n_2\} \geq \frac{3}{4} \sum_{j=1}^{m} \hat{z}_j \geq \frac{3}{4} M^*.$$

Proof:
Denote $\alpha_k = 1 - 1/2^k$. One has

$$\begin{aligned} n_1 &= \sum_{k\geq 1} \sum_{C_j\in C^k} \alpha_k \geq \sum_{k\geq 1} \sum_{C_j\in C^k} \alpha_k \hat{z}_j, \\ n_2 &\geq \sum_{k\geq 1} \sum_{C_j\in C^k} \beta_k \hat{z}_j. \end{aligned}$$

Note that for $k \geq 1$

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

This implies

$$\max\{n_1, n_2\} \geq \frac{n_1 + n_2}{2} \geq \sum_{k\geq 1} \sum_{C_j\in C^k} \frac{\alpha_k + \beta_k}{2} \hat{z}_j \geq \frac{3}{4} \sum_{j=1}^{m} \hat{z}_j.$$

4. The DNF Satisfiability Problem

Instance: Boolean function $f(x_1, \dots, x_n)$ in DNF (i.e. $f = C_1 \vee C_2 \vee \dots \vee C_m$).
Problem: Compute $\#(F)$ (the number of tuples $(x_1, \dots, x_n) \in \{0,1\}^n$ that satisfy f).

It is known that this problem is #P-complete. Obviously:

$$0 < \#F \leq 2^n.$$

First, we consider a general problem:
Let U be a finite set and $f : U \mapsto \{0,1\}$ be a function. We assume that the value of f can be computed fast. The question is to determine $|G|$, where

$$G = \{u \in U \mid f(u) = 1\}.$$

We apply the Monte-Carlo method and make N independent samples $u_1, \dots, u_N$ from U. Introduce random variables Y_i $(i = 1, \dots, N)$ defined as

$$Y_i = \begin{cases} 1, & \text{if } f(u_i) = 1 \\ 0, & \text{otherwise} \end{cases}$$

Furthermore, let

$$Z = |U| \cdot \sum_{i=1}^{N} \frac{Y_i}{N}.$$

Since $\mathbf{E}[Z] = |G|$, we hope that with a high probability Z is an ϵ-approximation for $|G|$. But this probability strictly depends on N.

Theorem 3 *Let* $\rho = |G|/|U|$*. Then the Monte-Carlo method provides an* ϵ*-approximation for* $|G|$ *with probability at least* $1-\delta$ *for a fixed* $\delta \in (0,1]$ *if*

$$N \geq \frac{4}{\epsilon^2 \rho} \ln \frac{2}{\delta}.$$

Proof:

Let $Y = \sum_{i=1}^{N} Y_i$. Then $\mathbf{E}[Y] = N\rho$.
We use the Chernoff inequalities:

$$\begin{aligned} \mathbf{Pr}[Y \leq (1-\epsilon)N\rho] &\leq e^{-N\rho\epsilon^2/2} \\ \mathbf{Pr}[(1+\epsilon)N\rho \leq Y] &\leq e^{-N\rho\epsilon^2/(2+\epsilon)}. \end{aligned}$$

Both upper bounds do not exceed $e^{-N\rho\epsilon^2/4}$. Hence,

$$\begin{aligned} & \mathbf{Pr}[(1-\epsilon)|G| \leq Z \leq (1+\epsilon)|G|] \\ = & \mathbf{Pr}[(1-\epsilon)N\rho \leq Y \leq (1+\epsilon)N\rho] \\ \geq & 1 - 2e^{-N\rho\epsilon^2/4}. \end{aligned}$$

So, $1 - 2e^{-N\rho\epsilon^2/4} \geq 1 - \delta$ iff $N \geq \frac{4}{\epsilon^2\rho} \ln \frac{2}{\delta}$. $\square$

The running time of this method is at least $N \geq 1/\rho$. The ratio $1/\rho$ is, however, not known in advance and can be exponentially large.

What is wrong in our approach is that the sample space U is very large. So if G is small, to evaluate $|G|$ with a high accuracy one has to do many samples, which makes N large.

Modification: decrease the size of sample space.

The set union problem:
Let V be a finite set and $H_1, \ldots, H_m \subseteq V$, so that for every i:

1. $|H_i|$ can be computed in polynomial time.
2. There is a way to select an element of H_i randomly and uniformly.
3. For every $v \in V$ it can be checked in polynomial time if $v \in H_i$.

Our goal: estimate the size of $H = H_1 \cup \cdots \cup H_m$.

Remark 1 *The above assumptions are satisfied for our original problem concerning DNF.*

We define a multiset $U = H_1 \uplus \cdots \uplus H_m$:

$$U = \{(v, i) \mid v \in H_i\}.$$

One has:

$$|U| = \sum_{j=1}^{m} |H_j| \geq |H|.$$

Furthermore, for $v \in V$ define a covering of v:

$$cov(v) = \{(v, i) \mid (v, i) \in U\}.$$

That is, $cov(v)$ is a set of subsets H_i that contain v.

We have the following observations:

1. The number of coverings sets is $|H|$ and they are simply computable.
2. $U = \bigcup_{v \in H} cov(v)$.
3. $|U| = \sum_{v \in H} |cov(v)|$.
4. $|cov(v)| \leq m$ for all $v \in H$.

We define

$$
\begin{aligned}
f((v,i)) &= \begin{cases} 1, & \text{if } i = \min\{j \mid v \in H_j\} \\ 0, & \text{otherwise} \end{cases} \\
G &= \{(v,i) \in U \mid f((v,i)) = 1\}.
\end{aligned}
$$

One has: $|G| = |H|$.

Lemma 2 *For the set union problem it holds:*

$$\rho = \frac{|G|}{|U|} \geq \frac{1}{m}.$$

Proof:

$$
\begin{aligned}
|U| &= \sum_{v \in H} |cov(v)| \\
&\leq \sum_{v \in H} m \\
&\leq m|H| = m|G|. \qquad \square
\end{aligned}
$$

Theorem 4 *The Monte-Carlo method provides an ϵ-approximation for $|G|$ with probability at least $1-\delta$ for a fixed $\delta \in (0,1]$ if*

$$N \geq \frac{4m}{\epsilon^2} \ln \frac{2}{\delta}.$$

Proof:

We make N independent samples (v,i) from U in two steps.

<u>**Step 1:**</u> choose an i randomly with probability

$$\mathbf{Pr}[i] = \frac{|H_i|}{|U|}.$$

<u>**Step 2:**</u> choose a $v \in H_i$ randomly and uniformly with probability

$$\mathbf{Pr}[v] = \frac{1}{|H_i|}.$$

This way the pairs (v,i) become uniformly distributed:

$$\mathbf{Pr}[(v,i)] = \frac{1}{|H_i|} \cdot \frac{|H_i|}{|U|} = \frac{1}{|U|}.$$

Let Y_i $\big(i = 1, \ldots, N\big)$ be random variables defined by

$$Y_i = \begin{cases} 1, & \text{if } f((v,i)) = 1 \\ 0, & \text{otherwise} \end{cases}$$

Furthermore, let

$$\begin{aligned} Y &= \sum_{i=1}^{N} Y_i \\ Z &= \frac{|U|}{N} Y. \end{aligned}$$

One has:

$$\begin{aligned}\mathbf{E}[Y] &= N\rho \\ \mathbf{E}[Z] &= |G|.\end{aligned}$$

We apply the Chernoff inequalities and obtain

$$\begin{aligned}& \mathbf{Pr}[(1-\epsilon)|G| \leq Z \leq (1+\epsilon)|G|] \\ =\ & \mathbf{Pr}[(1-\epsilon)N\rho \leq Y \leq (1+\epsilon)N\rho] \\ \geq\ & 1 - 2e^{-N\rho\epsilon^2/4} \\ \geq\ & 1 - 2e^{-N\epsilon^2/4m}.\end{aligned}$$

Therefore,

$$1 - 2e^{-N\epsilon^2/4m} \geq 1 - \delta,$$

which implies

$$N \geq \frac{4m}{\epsilon^2} \ln \frac{2}{\delta}.$$

The total running time of this method is polynomial w.r.t. m, $1/\epsilon$, and $\ln(1/\delta)$. $\square$

5. The Las-Vegas Algorithms

The algorithm from the previous section is an example of Monte-Carlo type algorithm. Such algorithms do not necessarily provide an exact solution, they just do it with a relatively high probability. However, their running time is usually much shorter compared to deterministic algorithms.

On the other hand, there are algorithms that surely provide a correct solution, however, one can estimate only their average running time.

Set-Coloring problem:
Instance: A set S with $|S| = n$ and subsets $F = \{S_i\}$, $S_i \subseteq S$, $|S_i| = r$, $i = 1, \ldots, k$, where $k \leq 2^{r-2}$.
Problem: Color every element $x \in S$ red or blue, so that each subset $S_i \in F$ contains elements of both colors.

Algorithm 2 2-COLORING(S, F);

1. Color every element of S randomly and independently in red or blue with probability 1/2.
2. Repeat step 1 until a valid coloring will be obtained.

How high is the probability that the coloring obtained after step 1 is invalid?

$$\mathbf{Pr}[\text{all elements of } S_i \text{ are red}] = 2^{-r}.$$

This implies

$$\mathbf{Pr}[\exists \text{ a ``red'' subset } S_i \in F] \leq k\,2^{-r} \leq 1/4.$$

The same inequality holds for an existence of a "blue" set. Hence,

$$\mathbf{Pr}[\text{the coloring is invalid}] \leq 1/2$$

and

$$\mathbf{Pr}[\text{the coloring is valid}] > 1/2.$$

Our algorithm is a Las-Vegas algorithm, since it constructs a new coloring until it becomes valid.

The last inequality implies that the expected number of repetitions of step 1 is only 2.