Randomized Algorithms

Lecture 13: Monte Carlo Method and DNF

Raul Garcia-Patron

School of Informatics
University of Edinburgh

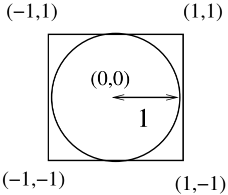
The Monte Carlo Method

- The Monte Carlo method refers to a collection of tools for estimating values through sampling and simulation. Monte Carlo techniques are used extensively in almost all areas of physical sciences and engineering.
- The key ideas:
  1. Make you quantity of interest the expectation value of a probability distribution.
  2. Sample from that specific probability distribution to estimate the expectation value.
- Monte Carlo techniques can be used to compute areas and integrals, as we will see shortly.

The Monte Carlo Method II

- A typical CS scenario for the Monte Carlo Method arises when the value we want to estimate is *the count of the number of combinatorial structures* satisfying a given criterion.
  1. We will usually rely on a close relationship between the problem of *counting the number of combinatorial structures* and *sampling one of the structures uniformly at random*.
- A Markov chain can sometimes be employed to do the sampling, which will be leveraged to estimate our value of interest.
- Ideally we want to design efficient (polynomial time) sampling algorithms.

Approximate π



Algorithm ESTIMATEPI(m)

1. $count \leftarrow 0$
2. **for** $i \leftarrow 1$ **to** m
3. draw (X, Y) uniformly at random from the square
   ie draw each of X, Y uniformly at random from the continuous distribution on $[-1, 1]$
4. **if** $X^2 + Y^2 \leq 1$ **then**
5. $count \leftarrow count + 1$
6. **return** $\frac{4 \cdot count}{m}$

Approximate π - Proof via Chernoff bound

Can let Z_i be the indicator variable for the "i-th" (X, Y) lying inside the circle. Then for $Z = \sum_{i=1}^{m} Z_i$,

$$\mathrm{E}[Z] = \sum_{i=1}^{m} \mathrm{E}[Z_i] = m\frac{\pi \cdot 1^2}{2^2} = \frac{\pi m}{4}.$$

Define new variable $Z' = \frac{4Z}{m}$, which satisfies $\mathrm{E}[Z'] = \frac{4}{m}\mathrm{E}[Z] = \pi$.

Approximate π - Proof via Chernoff bound

- Remember: $Z' = \frac{4Z}{m}$, which satisfies $\mathrm{E}[Z'] = \frac{4}{m}\mathrm{E}[Z] = \pi$.
- Better estimate the higher m is.
- By Chernoff (4.6) if we have m samples, then for arbitrary $\epsilon \in (0, 1)$,

$$\begin{aligned} \Pr[|Z' - \mathrm{E}[Z']| \geq \epsilon\pi] &= \Pr\left[\left|Z - \frac{\pi m}{4}\right| \geq \frac{\epsilon\pi m}{4}\right] \\ &= \Pr[|Z - \mathrm{E}[Z]| \geq \epsilon \mathrm{E}[Z]] \\ &\leq 2e^{-\epsilon^2 \pi m/12}. \end{aligned}$$

- We can achieve: $2e^{-\epsilon^2 \pi m/12} \leq \delta$, if $m \geq \frac{12 \ln(\frac{2}{\delta})}{\pi\epsilon^2}$.
  - Where ϵ is a relative error.
  - Where δ is the probability of failure of estimate.

Definition of (ϵ, δ)–approximation

Definition (Definition 11.1)

A randomized algorithm for estimating a (positive) quantity V (usually depending on certain input parameters) is said to give an (ϵ, δ) approximation if its output X satisfies

$$\Pr[|X - V| \geq \epsilon V] \leq \delta.$$

- The algorithm EstimatePi gives an

$$(\epsilon, 2e^{-\epsilon^2 \pi m/12})$$

approximation.

Monte Carlo Method

Definition (Generalization (Theorem 11.1))

Let $X_1, \ldots, X_m$ be independent and identically distributed indicator random variables (ie Bernoulli with a fixed parameter), and $\mu = \sum_{i=1}^{m} \mathrm{E}[X_i]$. Then if $m \geq \frac{3\ln(\frac{2}{\delta})}{\epsilon^2\mu}$, we have

$$\Pr\left(\left|\frac{1}{m}\sum_{i=1}^{m} X_i - \mu\right| \geq \epsilon\mu\right) \leq \delta.$$

So for this m, sampling gives a (ϵ, δ)-approximation of μ.

Definition (FPRAS (Definition 11.2))

A *fully polynomial randomized approximation scheme (FPRAS)*:

- Given input x, we want $(\epsilon, \delta)-$approximation of $V(x)$.
- Achieved in time polynomial in $1/\epsilon$, in $\ln(1/\delta)$ and size of x.

The DNF counting problem

Disjunctive Normal Form (DNF):

- each *clause* is now a *conjunction* ($\wedge$, AND) literals
- we have disjunctions ($\vee$, OR) of clauses

For example:

$$(x_1 \wedge \bar{x_2} \wedge x_3) \vee (x_2 \wedge x_4) \vee (\bar{x_1} \wedge x_3 \wedge x_4).$$

We are interested in *counting the number of satisfying assignments*.

- It is easy to find satisfying assignments or prove not satisfiable.
- It is NP-hard to compute the *exact* number of satisfying assignments for a DNF:
  - we can easily construct a DNF for the negation of the SAT formula ϕ
  - The DNF has 2^n satisfying assignments $\Leftrightarrow$ ϕ was unsatisfiable
- Counting DNF assignments is $\sharp P$-complete.
- However, we can approximately count them.

The DNF counting problem - Naïve Approach

- let $c(F)$ denote number of satisfying assignments of a given DNF formula F over n variables.
- $c(F)$ will be 0 *only if* it is the case that *every clause* contains x_i and $\bar{x}_i$ for some i. Easy to notice and eliminate before we start.
- Naïve approach to counting DNF assignments is to sample m *uniform random assignments* to $x_1, \ldots, x_n$ (from the set $\{0, 1\}^n$) and check whether F is satisfied for each sample.
  - The random variable X_i will be 1 if the i-th trial satisfies F, 0 otherwise.
  - Then we estimate the fraction of these to satisfy F and we return estimate

    $$\hat{c}(F) = 2^n \frac{\sum_{i=1}^{m} X_i}{m}, \qquad (1)$$

    as the estimate of satisfying assignments $c(F)$.

The DNF counting problem - Naïve Approach

- In order for $\hat{c}(F)$ to be an (ϵ, δ)-approximation for $c(F)$, we require:

$$\left|2^n \frac{\sum_{i=1}^m X_i}{m} - c(F)\right| \leq \epsilon \cdot c(F) \Leftrightarrow \left|\sum_{i=1}^m X_i - \frac{mc(F)}{2^n}\right| \leq \epsilon \cdot \frac{mc(F)}{2^n} \tag{2}$$

- by Chernoff this holds $\Leftrightarrow$ we have $m \geq \frac{3 \cdot 2^n \ln(\frac{2}{\delta})}{\epsilon^2 c(F)}$.
- If $c(F)$ is much much smaller than 2^n, then we need a huge number of samples, as a random assignment is very unlikely to hit the good assignments.

FPRAS for DNF counting

Our formula is

$$F = C_1 \vee C_2 \vee \ldots \vee C_t,$$

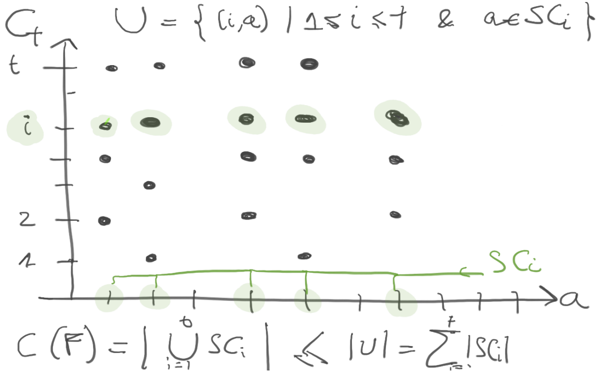
where every C_i is a *conjunction of literals*.

- If C_i contains the literals $x_j, \bar{x}_j$ for the *same* $j \in [n]$ (*opposing* literals), there is *no* assignment which can satisfy clause C_i.
- If C_i does not contain any opposing pair of literals, then C_i is satisfied by *any* assignment $a \in \{0,1\}^n$ which sets

$$a_j = \begin{cases} 1 & C_i \text{ contains the positive literal } x_j \\ 0 & C_i \text{ contains the negative literal } \bar{x}_j \\ 0/1 & \text{neither } x_j \text{ nor } \bar{x}_j \text{ appear in } C_i \end{cases}$$

- Assuming C_i has ℓ_i literals and no opposing pair, then there are *exactly* $2^{n-\ell_i}$ satisfying assignments for C_i.

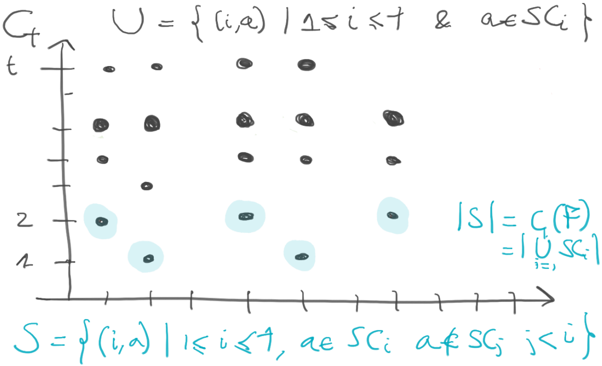
Definitions and intuition I



For every clause C_i, we define SC_i to be the set of $2^{n-\ell_i}$ assignments $a \in \{0,1\}^n$ which satisfy C_i: $U =_{def} \{(i,a) \mid 1 \leq i \leq t \text{ and } a \in SC_i\}$.

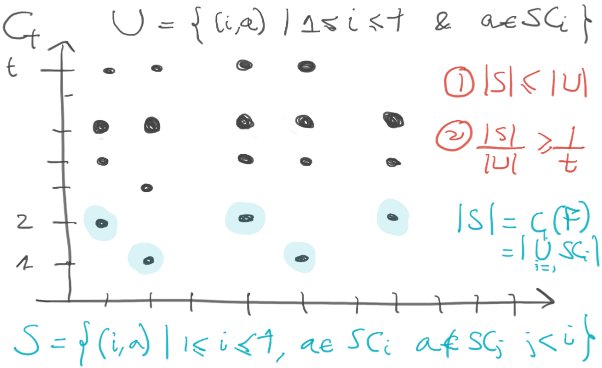
- The SC_i sets are *not* disjoint, as a satisfying assignment for one clause may *also* satisfy a different clause/clauses.

Definitions and intuition II



- To estimate $c(F)$ we need to define a subset S of U of size $c(F)$. For each assignment a there must be a single pair (i, a).
- We do so by choosing the lowest j that is satisfied by assignment a.

Relations between sets



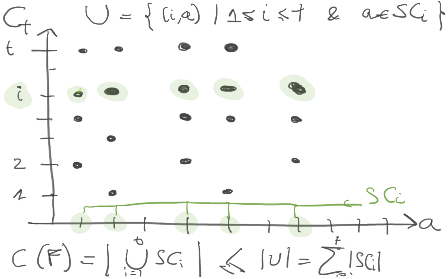
- We know how to compute $|U| = \sum_{i=1}^{t} s^{n-|C_i|}$
- S is approx. of same size as U: $\frac{|S|}{|U|} \geq \frac{1}{t}$. Key to make the sampling algorithm efficient.

Algorithm for sampling DNF assignments

Algorithm APPROXDNF$(n; m; C_1 \vee \ldots \vee C_t)$

count $\leftarrow 0$
cardU $\leftarrow 0$
for $i \leftarrow 1$ **to** t
    cardU $\leftarrow$ *cardU* $+ 2^{n-|C_i|}$
for $k \leftarrow 1$ **to** m
    Choose i with probability $\frac{2^{n-|C_i|}}{cardU}$.
    Sample $a \in SC_i$ by setting the literals of C_i to the required values, then randomly generating the other $n - |C_i|$ bits.
    if (a does not satisfy $C_{i'}$ for any $i' < i$) **then**
        count $\leftarrow$ *count* $+ 1$
return $\frac{count}{m} \cdot (cardU)$.

Sampling from U



$\Pr((i,a) \text{ is chosen}) = \Pr(i \text{ is chosen}) \cdot \Pr(a \text{ is chosen} | i \text{is chosen})$:

$$= \frac{|SC_i|}{|U|} \frac{1}{|SC_i|} = \frac{1}{|U|}$$

Remember:

- Choose i with probability $\frac{2^{n-|C_i|}}{cardU}$.
- Sample $a \in SC_i$ by setting the literals of C_i to the required values, then randomly generating the other $n - |C_i|$ bits.

FPRAS for DNF counting

Theorem (Theorem 11.2)

Our DNF counting algorithm gives a fully-polynomial randomized approximation scheme for the DNF counting problem if we set $m = \lceil \frac{3t}{\epsilon^2} \ln(\frac{2}{\delta}) \rceil$.

Proof.

- Using Theorem 11.1, if $m \geq \frac{3 \ln(\frac{2}{\delta})}{\epsilon^2 \mu}$, we have

$$\Pr\left(|\frac{1}{m} \sum_{i=1}^{m} X_i - \mu| \geq \epsilon\mu \right) \leq \delta$$

  we get a (ϵ, δ)-approximation of μ.
- X_i indicator that sample i belongs to subgroup S: $\mathrm{E}[X_i] = \frac{c(F)}{|U|} \geq \frac{1}{t}$.

□