Model Counting for Logical Theories

Thursday

Dmitry Chistikov Rayna Dimitrova

Department of Computer Science
University of Oxford, UK

Max Planck Institute for Software Systems (MPI-SWS)
Kaiserslautern and Saarbrücken, Germany

ESSLLI 2016

$$\begin{tikzcd}
\begin{array}{c}\mathrm{SAT}\\ \text{satisfiability}\end{array} \arrow[r] \arrow[d] & \begin{array}{c}\mathrm{SMT}\\ \text{satisfiability}\\ \text{modulo theories}\end{array} \arrow[d] \\
\begin{array}{c}\#\mathrm{SAT}\\ \text{model counting}\end{array} \arrow[r] & \begin{array}{c}\#\mathrm{SMT}\\ \text{model counting}\\ \text{modulo theories}\end{array}
\end{tikzcd}$$

Agenda

Tuesday	computational complexity, probability theory
Wednesday	randomized algorithms, Monte Carlo methods
Thursday	hashing-based approach to model counting
Friday	from discrete to continuous model counting

Outline

1. Markov chain Monte Carlo method (continued)
2. Approximate model counting for $\#\mathrm{SAT}$
3. Universal hashing

Markov chain Monte Carlo recap

Goal: Sample from a probability distribution P over a set Ω.

Problem: We cannot sample directly from P,
but we can evaluate queries $P(s)$ for any state s in the universe.

MCMC:

1. Construct a Markov chain whose stationary distribution is P.

   We implicitly define a graph and the transition probabilities on its edges to make the stationary distribution P.

2. Take a random walk of sufficient length on the Markov chain.

3. Output the reached state s.

Markov chain Monte Carlo

Markov chain Monte Carlo (MCMC) is a technique for sampling from a complicated distribution using local information.

The main challenge is to obtain good bounds on the number of steps a Markov chain takes to converge to the desired distribution.

MCMC may provide efficient (i.e., polynomial time) solution techniques.

Computing the volume of a convex body

Given a convex body $K \subseteq \mathbb{R}^n$, compute its volume $Vol(K)$.

The computational effort required increases as n increases.

[Dyer and Frieze'88] Computing the volume exactly is $\#\mathbf{P}$-hard.

[Dyer, Frieze and Kannan'91] Polynomial randomized approximation algorithm via Markov chain Monte Carlo.

Input to the algorithm

K is given as a membership oracle.

Two n-dimensional balls $B_0 \subseteq K \subseteq B_r$ of non-zero radius.

By simple transformations of K it can be ensured that B_0 is the unit ball and that B_r has radius $cn \log n$ for some constant c.

Note: The volume of the smallest ball containing K might be exponential in $Vol(K)$, hence naive Monte Carlo is hopeless.

From volume computation to uniform sampling

Construct a sequence of concentric balls
$B_0 \subseteq B_1 \subseteq \ldots \subseteq K \subseteq B_r$.

$$Vol(K) = \frac{Vol(K \cap B_r)}{Vol(K \cap B_{r-1})} \cdot \frac{Vol(K \cap B_{r-1})}{Vol(K \cap B_{r-2})} \cdot \ldots \cdot \frac{Vol(K \cap B_1)}{Vol(K \cap B_0)} \cdot Vol(K \cap B_0)$$

$Vol(K \cap B_0) = Vol(B_0)$ known.

Estimate each ratio $\frac{Vol(K \cap B_i)}{Vol(K \cap B_{i-1})}$.
Sample uniformly at random from $K \cap B_i$ using MCMC and count the proportion of samples falling into B_{i-1}.

To ensure that the number of samples needed is small, ensure that the ratio $\frac{Vol(K \cap B_i)}{Vol(K \cap B_{i-1})}$ is small by making the balls grow slowly.
This implies $r = cn \log n$ for some constant c.

Time complexity

The original algorithm has time complexity $O(n^{23})$.

Later it was improved to $O(n^4)$.

Key ingredient: sample uniformly at random from from the points in a convex body in polynomial time. For this, the Markov chain has to converge in polynomial time to the uniform distribution.

Markov chains

Recall the definition of finite (discrete) Markov chains.

Finite Markov chain $\mathfrak{M} = (\Omega, T)$

- finite set of states Ω,
- transition probability matrix T where

$$T_{s,s'} = \mathsf{P}(\text{next state will be } s' \mid \text{ the current state is } s)$$

Markov chains

Markov chain with continuous state-space $\mathfrak{M} = (\Omega, T)$

- continuous set of states Ω,
- transition kernel $T(s, A)$ which for $s \in \Omega$ and measurable set $A \subseteq \Omega$ defines the probability of reaching A from state s

$$T(s, A) = \int_{x \in A} p(s, x) dx,$$

where $p : \Omega \times \Omega \to \mathbb{R}$ is a non-negative function.
For a given s, $p(s, \cdot)$ is a probability density function

$$T(s, \Omega) = \int_{x \in \Omega} p(s, x) dx = 1$$

The concepts of irreducibility and aperiodicity can be redefined for continuous state spaces.

The random walk on cubes

1. Divide the space into n-dimensional (hyper)cubes of side δ.
   Choose δ such to provide a good approximation of K, while permitting the random walk on the Markov chain to converge to the stationary distribution in reasonable time.
2. Perform a random walk as follows. If C is the cube at time t, select uniformly at random an orthogonally adjacent cube C'.
   If C' is in K, then move to C', otherwise stay at C.

Properties:

- The uniform distribution is the unique stationary distribution.
- **Rapid mixing:** The Markov chain converges to the stationary distribution in number of steps polynomial in n.

A ball walk

Lovász and Simonovits proposed a walk with continuous space.

1. Pick $\delta \in \mathbb{R}$ by the same criteria as before.
2. Perform a random walk as follows.

   If at time t the walk is at $x \in \mathbb{R}^n$, the probability density function at time $t+1$ is uniform over $K \cap B(x, \delta)$ and 0 outside.

Properties:

- Rapid mixing argument similar to the walk on cubes.
- Saves a factor n in the number of oracle calls.
- Moves more complex, so no saving in time complexity.

Approximate model counting via MCMC

Theorem

If we can sample almost uniformly at random from Ω_{KNAPSACK} in polynomial time, then there is a polynomial-time randomized approximation algorithm for the knapsack counting problem.

Theorem

There exists a polynomial time randomized approximation algorithm $\mathcal{A}$ for computing $\mathsf{mc}(\varphi)$ for Real Arithmetic formulas of the form $\varphi = \bigwedge_i \left(\sum_{j=1}^{n} a_{i,j} x_{i,j} \leq b_i \right)$. That is,

$$\mathsf{P}[(1+\varepsilon)^{-1}\mathsf{mc}(\varphi) \leq \mathcal{A}(\varphi, \alpha, \varepsilon) \leq (1+\varepsilon)\mathsf{mc}(\varphi)] \geq 1 - \alpha,$$

and the running time of $\mathcal{A}$ is polynomial in n, $\frac{1}{1+\varepsilon}$ and $\log(\frac{1}{\alpha})$.

Outline

1. Markov chain Monte Carlo method (continued)
2. Approximate model counting for $\#\mathrm{SAT}$
3. Universal hashing

Counting with an $\mathbf{NP}$ oracle

Recall the problem $\#\mathrm{SAT}$.
Given a propositional formula $\varphi(x_1, \ldots, x_n)$
Compute $\mathsf{mc}(\varphi)$, i.e., the number of models of φ.

The decision problem is in $\mathbf{NP}$ and the counting problem is in $\#\mathbf{P}$.

We will devise a randomized approximation algorithm for $\#\mathrm{SAT}$

- randomized polynomial time algorithm,
- with bounded two-sided error,
- with access to an $\mathbf{NP}$ oracle (SAT oracle).

The number of queries to the $\mathbf{NP}$ oracle is at most polynomial.

Counting with an $\mathbf{NP}$ oracle

Recall the problem $\#\mathrm{SAT}$.
Given a propositional formula $\varphi(x_1, \ldots, x_n)$
Compute $\mathsf{mc}(\varphi)$, i.e., the number of models of φ.

We will devise a randomized approximation algorithm for $\#\mathrm{SAT}$.

Given

- $\varphi(x_1, \ldots, x_n)$: propositional formula,
- $\alpha \in [0, 1]$: probability of error,
- $\varepsilon \in \mathbb{R}$: approximation factor,

the algorithm will compute a value $\mathcal{A}(\varphi, \alpha, \varepsilon)$ such that

$$\mathsf{P}[(1+\varepsilon)^{-1}\mathsf{mc}(\varphi) \leq \mathcal{A}(\varphi, \alpha, \varepsilon) \leq (1+\varepsilon)\mathsf{mc}(\varphi)] \geq 1 - \alpha.$$

An Estimate oracle $\mathcal{E}$

Suppose we have an **Estimate oracle** $\mathcal{E}$ that takes a formula φ and an integer $m \in \mathbb{N}$ and returns YES or NO so that

$$\mathsf{mc}(\varphi) \geq 2^{m+1} \implies \mathsf{P}[\mathcal{E}(\varphi, n) = \mathsf{YES}] \geq \frac{3}{4}$$

$$\mathsf{mc}(\varphi) \leq 2^{m} \implies \mathsf{P}[\mathcal{E}(\varphi, n) = \mathsf{NO}] \geq \frac{3}{4}$$

Note that if $2^m < \mathsf{mc}(\varphi) < 2^{m+1}$, the oracle provides no guarantees. This is the oracle's **blind spot**.

An Estimate oracle $\mathcal{E}$

Suppose we have an **Estimate oracle** $\mathcal{E}$ that takes
a formula φ and an integer $m \in \mathbb{N}$ and returns YES or NO so that

$$\mathsf{mc}(\varphi) \geq 2^{m+1} \Longrightarrow \mathsf{P}[\mathcal{E}(\varphi, n) = \mathsf{YES}] \geq \frac{3}{4}$$
$$\mathsf{mc}(\varphi) \leq 2^{m} \Longrightarrow \mathsf{P}[\mathcal{E}(\varphi, n) = \mathsf{NO}] \geq \frac{3}{4}$$

We make a sequence of calls to $\mathcal{E}$ to compute the value v

$$\mathcal{E}(\varphi, 0), \mathcal{E}(\varphi, 1), \ldots, \mathcal{E}(\varphi, m-1), \mathcal{E}(\varphi, m), \ldots, \mathcal{E}(\varphi, n+1)$$

until we get the first NO answer and then determine v:

- NO answer for $m = 0$, let $v = 0$ if φ is unsat., else $v = 1$,
- NO answer for $m > 0$, let $v = 2^m$.

2-Approximation

The above procedure with an oracle $\mathcal{E}$ such that

$$\mathsf{mc}(\varphi) \geq 2^{m+1} \implies \mathsf{P}[\mathcal{E}(\varphi, n) = \mathsf{YES}] \geq \frac{3}{4}$$

$$\mathsf{mc}(\varphi) \leq 2^{m} \implies \mathsf{P}[\mathcal{E}(\varphi, n) = \mathsf{NO}] \geq \frac{3}{4}$$

gives a 2-approximation of $\mathsf{mc}(\varphi)$ with high probability.

Cases

1. If first NO is for $m = 0$ then $v \in \{0, 1\}$ and $\mathsf{mc}(\varphi) < 2$ with high probability.
2. If first NO is for $m > 0$ then with high probability

   $$2^{m-1} < \mathsf{mc}(\varphi) < 2^{m+1}$$

   which implies

   $$\frac{1}{2}\mathsf{mc}(\varphi) < 2^{m} < 2\mathsf{mc}(\varphi).$$

2-Approximation

The above procedure with an oracle $\mathcal{E}$ such that

$$\mathsf{mc}(\varphi) \geq 2^{m+1} \implies \mathsf{P}[\mathcal{E}(\varphi, n) = \mathsf{YES}] \geq \frac{3}{4}$$
$$\mathsf{mc}(\varphi) \leq 2^{m} \implies \mathsf{P}[\mathcal{E}(\varphi, n) = \mathsf{NO}] \geq \frac{3}{4}$$

gives a 2-approximation of $\mathsf{mc}(\varphi)$ with high probability.

Cases

1. If first NO is for $m = 0$ then $v \in \{0, 1\}$ and $\mathsf{mc}(\varphi) < 2$ with high probability.
2. If first NO is for $m > 0$ then with high probability

$$2^{m-1} < \mathsf{mc}(\varphi) < 2^{m+1}$$

which implies

$$\frac{1}{2}\mathsf{mc}(\varphi) \leq v \leq 2\mathsf{mc}(\varphi).$$

From 2-approximation to $(1+\varepsilon)$-approximation

If we have an algorithm $\mathcal{A}'$ such that

$$\frac{1}{2}\mathsf{mc}(\varphi) \le \mathcal{A}'(\varphi) \le 2\mathsf{mc}(\varphi),$$

we can construct an algorithm $\mathcal{A}$ such that

$$(1+\varepsilon)^{-1}\mathsf{mc}(\varphi) \le \mathcal{A}(\varphi, \varepsilon) \le (1+\varepsilon)\mathsf{mc}(\varphi).$$

Let $\mathcal{A}(\varphi, \varepsilon) = \sqrt[q]{\mathcal{A}'(\varphi')}$,
where $\varphi' = \bigwedge_{i=1}^{q} \varphi(x_1^i, \ldots, x_n^i)$ and $q = \frac{1}{\log(1+\varepsilon)}$.

The formula φ' is a conjunction of q copies of φ, with pairwise disjoint sets of variables. Thus $\mathsf{mc}(\varphi') = \mathsf{mc}(\varphi)^q$.

The blind spot of $\mathcal{E}$

Thus, it suffices to have an Estimate oracle $\mathcal{E}$ such that for some constants $0 < c < C$ the oracle satisfies the conditions

$$\mathsf{mc}(\varphi) \geq C \cdot 2^m \Longrightarrow \mathsf{P}[\mathcal{E}(\varphi, n) = \mathsf{YES}] \geq \frac{3}{4}$$
$$\mathsf{mc}(\varphi) \leq c \cdot 2^m \Longrightarrow \mathsf{P}[\mathcal{E}(\varphi, n) = \mathsf{NO}] \geq \frac{3}{4}$$

The blind spot of $\mathcal{E}$ is $(c \cdot 2^m, C \cdot 2^m)$.

Probability amplification

Recall that we want an algorithm $\mathcal{A}(\varphi, \alpha, \varepsilon)$ such that

$$\mathsf{P}[(1+\varepsilon)^{-1}\mathsf{mc}(\varphi) \leq \mathcal{A}(\varphi, \alpha, \varepsilon) \leq (1+\varepsilon)\mathsf{mc}(\varphi)] \geq 1 - \alpha.$$

We can amplify the probability of the described method by doing multiple runs and a majority vote for each call to $\mathcal{E}$.

Approximate counting with an Estimate oracle

Assume we have an **Estimate oracle** $\mathcal{E}$ such that

$$\mathsf{mc}(\varphi) \geq C \cdot 2^m \Longrightarrow \mathsf{P}[\mathcal{E}(\varphi, n) = \mathsf{YES}] \geq \frac{3}{4}$$

$$\mathsf{mc}(\varphi) \leq c \cdot 2^m \Longrightarrow \mathsf{P}[\mathcal{E}(\varphi, n) = \mathsf{NO}] \geq \frac{3}{4}$$

Make a sequence of calls to $\mathcal{E}$

$$\mathcal{E}(\varphi, 0), \mathcal{E}(\varphi, 1), \ldots, \mathcal{E}(\varphi, m-1), \mathcal{E}(\varphi, m), \ldots, \mathcal{E}(\varphi, n+1)$$

until for some m we get the first NO answer.

- if $m = 0$, return 0 if φ is unsat., else return 1,
- if $m > 0$, return $c \cdot 2^m$.

Approximate counting with an Estimate oracle

Assume we have an **Estimate oracle** $\mathcal{E}$ such that

$$\mathsf{mc}(\varphi) \geq C \cdot 2^m \Longrightarrow \mathsf{P}[\mathcal{E}(\varphi, n) = \mathsf{YES}] \geq \frac{3}{4}$$
$$\mathsf{mc}(\varphi) \leq c \cdot 2^m \Longrightarrow \mathsf{P}[\mathcal{E}(\varphi, n) = \mathsf{NO}] \geq \frac{3}{4}$$

The difficult part: provide an Estimate oracle $\mathcal{E}$ that makes at most polynomial number of queries to the SAT oracle.

Approximate counting with an Estimate oracle

Assume we have an **Estimate oracle** $\mathcal{E}$ such that

$$\mathsf{mc}(\varphi) \geq C \cdot 2^m \Longrightarrow \mathsf{P}[\mathcal{E}(\varphi, n) = \mathsf{YES}] \geq \frac{3}{4}$$
$$\mathsf{mc}(\varphi) \leq c \cdot 2^m \Longrightarrow \mathsf{P}[\mathcal{E}(\varphi, n) = \mathsf{NO}] \geq \frac{3}{4}$$

The difficult part: provide an Estimate oracle $\mathcal{E}$ that makes at most polynomial number of queries to the SAT oracle.

Use hashing.

Outline

1. Markov chain Monte Carlo method (continued)
2. Approximate model counting for $\#\mathrm{SAT}$
3. Universal hashing

Simple dictionary problem

[http://cs.au.dk/~bromille/Notes/un.pdf]

Develop a data structure that implements a **set** and supports $\mathrm{Insert}(e)$, $\mathrm{Delete}(e)$, and $\mathrm{Lookup}(e)$ operations.

1. Elements come from universe U.
2. Operations should be performed online.
3. The set will never grow beyond size N, where $N < |U|$.

Simple dictionary problem

[http://cs.au.dk/~bromille/Notes/un.pdf]

Develop a data structure that implements a **set** and supports $\mathrm{Insert}(e)$, $\mathrm{Delete}(e)$, and $\mathrm{Lookup}(e)$ operations.

1. Elements come from universe U.
2. Operations should be performed online.
3. The set will never grow beyond size N, where $N < |U|$.

Solution:
Use an array $A[1..N]$ and **hash function** $h\colon U \to \{1, \ldots, N\}$.
Chained hashing: $A[j]$ keeps a linked list of e for which $h(e) = j$.

Chained hashing

Use an array $A[1..N]$ and **hash function** $h\colon U \to \{1, \dots, N\}$.
Chained hashing: $A[j]$ keeps a linked list of e for which $h(e) = j$.

How to analyze the efficiency of this solution?

Chained hashing

Use an array $A[1..N]$ and **hash function** $h\colon U \to \{1, \ldots, N\}$.
Chained hashing: $A[j]$ keeps a linked list of e for which $h(e) = j$.

How to analyze the efficiency of this solution?

- Input $=$ sequence of $\mathrm{Insert}(e_i)$, $\mathrm{Delete}(e_i)$, and $\mathrm{Lookup}(e_i)$
- Cannot assume anything about input structure

Chained hashing

Use an array $A[1..N]$ and **hash function** $h\colon U \to \{1,\ldots,N\}$.
Chained hashing: $A[j]$ keeps a linked list of e for which $h(e) = j$.

How to analyze the efficiency of this solution?

- Input $=$ sequence of $\mathrm{Insert}(e_i)$, $\mathrm{Delete}(e_i)$, and $\mathrm{Lookup}(e_i)$
- Cannot assume anything about input structure

Note that we don't have to fix a specific $h\colon U \to \{1,\ldots,N\}$.
What if h is a random function $h\colon U \to \{1,\ldots,N\}$?

Analysis of chained hashing

Let's do another kind of analysis for randomized algorithms:
Compute the expectation of the random variable

$$T(\mathrm{Op}(e_1), \ldots, \mathrm{Op}(e_m)) = \sum_{i=1} T(\mathrm{Op}(e_i)).$$

Consider $\mathsf{E}\, T(\mathrm{Op}(e_i))$.

Analysis of chained hashing

Let's do another kind of analysis for randomized algorithms: Compute the expectation of the random variable

$$T(\mathrm{Op}(e_1), \ldots, \mathrm{Op}(e_m)) = \sum_{i=1} T(\mathrm{Op}(e_i)).$$

Consider $\mathsf{E}\, T(\mathrm{Op}(e_i))$.

Let S_i be the set after $\mathrm{Op}(e_i)$ is performed.

Analysis of chained hashing

Let's do another kind of analysis for randomized algorithms:
Compute the expectation of the random variable

$$T(\mathrm{Op}(e_1), \ldots, \mathrm{Op}(e_m)) = \sum_{i=1} T(\mathrm{Op}(e_i)).$$

Consider $\mathsf{E}\, T(\mathrm{Op}(e_i))$.

Let S_i be the set after $\mathrm{Op}(e_i)$ is performed.

Claim

$\mathsf{E}\, T(\mathrm{Op}(e_i)) \leq 3$.

Analysis of chained hashing

Let's do another kind of analysis for randomized algorithms:
Compute the expectation of the random variable

$$T(\mathrm{Op}(e_1), \ldots, \mathrm{Op}(e_m)) = \sum_{i=1} T(\mathrm{Op}(e_i)).$$

Consider $\mathsf{E}\, T(\mathrm{Op}(e_i))$.

Let S_i be the set after $\mathrm{Op}(e_i)$ is performed.

Claim
$\mathsf{E}\, T(\mathrm{Op}(e_i)) \leq 3.$

We only used the fact that $\mathsf{P}[\, h(y) = h(e_i) \,] \leq 1/N$ for all $y \in U$ such that $y \neq e_i$.

Universal hashing

Definition

Let $\mathcal{H}$ be a family of functions mapping U to $\{1, \ldots, N\}$.
It is a **universal family** if for all pairs $x \neq y$ from U and for $h \in \mathcal{H}$ chosen uniformly at random

$$\mathsf{P}[\,h(x) = h(y)\,] \leq 1/N.$$

Universal hashing

Definition

Let $\mathcal{H}$ be a family of functions mapping U to $\{1, \ldots, N\}$.
It is a **universal family** if for all pairs $x \neq y$ from U
and for $h \in \mathcal{H}$ chosen uniformly at random

$$\mathsf{P}[\, h(x) = h(y)\,] \leq 1/N.$$

So for the simple dictionary problem we can as a first step of the algorithm pick a function at random from a universal family $\mathcal{H}$.

Example

$h_{a,b}(x) = ((ax + b) \bmod p) \bmod N$, $a, b \in \{0, 1, \ldots, p-1\}$,
forms a universal family if p is a prime greater than N.

Domain and range of hash functions

From now on instead of U and $\{1, \ldots, N\}$
we use $\{0,1\}^n$ and $\{0,1\}^m$.

Pairwise independent families

Definition

Let $\mathcal{H}$ be a family of functions mapping $\{0,1\}^n$ to $\{0,1\}^m$.
It is a **family of pairwise independent hash functions** if
for all pairs $\boldsymbol{x} \neq \boldsymbol{y}$ from $\{0,1\}^n$,
for all elements $\boldsymbol{w}_1, \boldsymbol{w}_2$ from $\{0,1\}^m$,
and for $h \in \mathcal{H}$ chosen uniformly at random

$$\mathsf{P}[\, h(\boldsymbol{x}) = \boldsymbol{w}_1, h(\boldsymbol{y}) = \boldsymbol{w}_2 \,] = (1/2^m)^2.$$

Hashing and model counting

Let $\varphi(x_1, \ldots, x_n)$ be our propositional formula.
Let $S = [\![\varphi]\!]$.

Hashing and model counting

Let $\varphi(x_1, \ldots, x_n)$ be our propositional formula.
Let $S = [\![\varphi]\!]$.

Suppose $h\colon \{0,1\}^n \to \{0,1\}^m$ is a nice hash function.
Then we can expect that h classfies elements S into 2^m bins more or less uniformly.

Hashing and model counting

Let $\varphi(x_1, \ldots, x_n)$ be our propositional formula.
Let $S = [\![\varphi]\!]$.

Suppose $h\colon \{0,1\}^n \to \{0,1\}^m$ is a nice hash function.
Then we can expect that h classfies elements S into 2^m bins more or less uniformly.

In particular, the expected cardinality of the 0^m-bin

$$\{\boldsymbol{x} \in S\colon h(\boldsymbol{x}) = 0^m\}$$

is $|S|/2^m$.

Hashing and model counting

Let $\varphi(x_1, \ldots, x_n)$ be our propositional formula.
Let $S = [\![\varphi]\!]$.

Suppose $h\colon \{0,1\}^n \to \{0,1\}^m$ is a nice hash function.
Then we can expect that h classfies elements S into 2^m bins more or less uniformly.

In particular, the expected cardinality of the 0^m-bin

$$\{\boldsymbol{x} \in S\colon h(\boldsymbol{x}) = 0^m\}$$

is $|S|/2^m$.

But for every h the size of a set is a nonnegative integer.
Is it 0 or non-0?

The Estimate oracle via hashing: Intuition

Intuition:
Consider $\gamma = |S|/2^m$, the expected cardinality of the 0^m-bin

$$\{\boldsymbol{x} \in S \colon h(\boldsymbol{x}) = 0^m\}.$$

If $\gamma \ll 1$, then the bin is likely to be empty.
If $\gamma \gg 1$, then the bin is likely to be non-empty.

The Estimate oracle via hashing: Intuition

Intuition:
Consider $\gamma = |S|/2^m$, the expected cardinality of the 0^m-bin

$$\{\boldsymbol{x} \in S \colon h(\boldsymbol{x}) = 0^m\}.$$

If $\gamma \ll 1$, then the bin is likely to be empty.
If $\gamma \gg 1$, then the bin is likely to be non-empty.

So if the bin is empty, then it's unlikely that $\gamma \gg 1$.
And if the bin is non-empty, then it's unlikely that $\gamma \ll 1$.

The Estimate oracle via hashing: Intuition

Intuition:
Consider $\gamma = |S|/2^m$, the expected cardinality of the 0^m-bin

$$\{\boldsymbol{x} \in S \colon h(\boldsymbol{x}) = 0^m\}.$$

If $\gamma \ll 1$, then the bin is likely to be empty.
If $\gamma \gg 1$, then the bin is likely to be non-empty.

So if the bin is empty, then it's unlikely that $\gamma \gg 1$.
And if the bin is non-empty, then it's unlikely that $\gamma \ll 1$.

Emptiness of bin: It's a satisfiability question!

Leftover hash lemma (simplified)

Lemma

Let $\mathcal{H}$ be a family of pairwise independent hash functions $h\colon \{0,1\}^n \to \{0,1\}^m$.
Let $S \subseteq \{0,1\}^n$ satisfy $|S| \geq 4/\rho^2 \cdot 2^m$ for some $\rho > 0$.
For $h \in \mathcal{H}$, let Z be the cardinality of the set $\{w \in S\colon h(w) = 0^m\}$. Then

$$\mathsf{P}\left[\left|Z - \frac{|S|}{2^m}\right| \geq \rho \cdot \frac{|S|}{2^m}\right] \leq \frac{1}{4}.$$

The Estimate oracle via hashing

Given $\varphi(x_1, \ldots, x_n)$ and m:

1. Pick $h\colon \{0,1\}^n \to \{0,1\}^m$ from $\mathcal{H}$ uniformly at random
2. Check satisfiability of the formula

$$\varphi(\boldsymbol{x}) \wedge (h(\boldsymbol{x}) = 0^m)$$

3. Return " $\mathsf{mc}(\varphi) \geq C \cdot 2^m$ " if φ is satisfiable.
   Return " $\mathsf{mc}(\varphi) \leq c \cdot 2^m$ " if φ is unsatisfiable.

Claim

Suppose $\mathcal{H}$ is a family of pairwise independent hash functions. Then for appropriate constants $0 < c < C$, this oracle returns the correct answer with probability $\geq 3/4$.

Pairwise independency: Random affine operators

Claim

Functions $h_{A,\boldsymbol{b}}\colon \{0,1\}^n \to \{0,1\}^m$ defined by

$$h_{A,\boldsymbol{b}}(\boldsymbol{x}) = A \cdot \boldsymbol{x} + \boldsymbol{b} \bmod 2$$

where $A \in \{0,1\}^{m \times n}$ and $\boldsymbol{b} \in \{0,1\}^m$,
form a pairwise independent family of hash functions.

Conclusion: counting via hashing

Theorem

There is a polynomial-time randomized algorithm that, when given access to an $\mathbf{NP}$ oracle, approximates $\mathsf{mc}(\varphi)$ for a propositional formula φ up to factor $(1+\varepsilon)$ with confidence $\geq 1-\alpha$.

[Jerrum, Valiant and Vazirani'86; Valiant and Vazirani'86]

Conclusion: counting via hashing

Theorem

There is a polynomial-time randomized algorithm that, when given access to an $\mathbf{NP}$ oracle, approximates $\mathsf{mc}(\varphi)$ for a propositional formula φ up to factor $(1+\varepsilon)$ with confidence $\geq 1-\alpha$.

[Jerrum, Valiant and Vazirani'86; Valiant and Vazirani'86]

Approximate counting can be done in $\mathbf{BPP}^{\mathbf{NP}}$
(i.e., efficiently—but assuming a SAT solver).

Summary of today's lecture

- Volume estimation via MCMC
- Approximate counting for $\#\mathrm{SAT}$ using hashing

Agenda

Tuesday	computational complexity, probability theory
Wednesday	randomized algorithms, Monte Carlo methods
Thursday	hashing-based approach to model counting
Friday	from discrete to continuous model counting