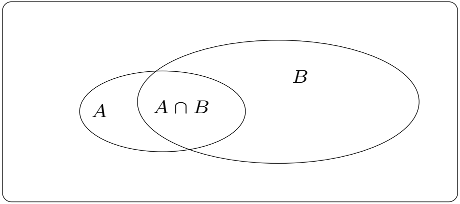
Lecture 2: Randomized Algorithms

- Independence & Conditional Probability
- Random Variables
- Expectation & Conditional Expectation
- Law of Total Probability
- Law of Total Expectation
- Derandomization Using Conditional Expectation

PTCF: Independence Events and Conditional Probabilities



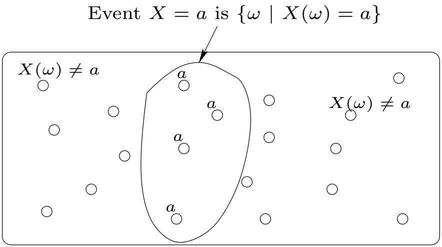
- The conditional probability of A given B is

$$\text{Prob}[A \mid B] := \frac{\text{Prob}[A \cap B]}{\text{Prob}[B]}$$

- A and B are independent if and only if $\text{Prob}[A \mid B] = \text{Prob}[A]$
- Equivalently, A and B are independent if and only if

$$\text{Prob}[A \cap B] = \text{Prob}[A] \cdot \text{Prob}[B]$$

PTCF: Discrete Random Variable



- A random variable is a function $X : \Omega \to \mathbb{R}$
- $p_X(a) = \text{Prob}[X = a]$ is called the probability mass function of X
- $P_X(a) = \text{Prob}[X \leq a]$ is called the (cumulative/probability) distribution function of X

PTCF: Expectation and its Linearity

- The expected value of X is defined as

$$\mathsf{E}[X] := \sum_a a\,\mathsf{Prob}[X = a].$$

- For any set $X_1, \dots, X_n$ of random variables, and any constants $c_1, \dots, c_n$

$$\mathsf{E}[c_1 X_1 + \cdots + c_n X_n] = c_1 \mathsf{E}[X_1] + \cdots + c_n \mathsf{E}[X_n]$$

  This fact is called linearity of expectation

PTCF: Indicator/Bernoulli Random Variable

$$X : \Omega \to \{0, 1\}$$

$$p = \mathsf{Prob}[X = 1]$$

X is called a Bernoulli random variable with parameter p

If $X = 1$ only for outcomes ω belonging to some event A, then X is called an indicator variable for A

$$\begin{aligned} \mathsf{E}[X] &= p \\ \mathsf{Var}[X] &= p(1-p) \end{aligned}$$

PTCF: Law of Total Probabilities

- Let $A_1, A_2, \dots$ be any partition of Ω, then

$$\mathsf{Prob}[A] = \sum_{i \geq 1} \mathsf{Prob}[A \mid A_i] \, \mathsf{Prob}[A_i]$$

(Strictly speaking, we also need "*and each A_i is measurable*," but that always holds for finite Ω.)

Example 1: Randomized Quicksort

$\textsc{Randomized-Quicksort}(A)$

```
n ← length(A)
if n = 1 then
    Return A
else
    Pick i ∈ {1, . . . , n} uniformly at random, A[i] is called the pivot
    L ← elements ≤ A[i]
    R ← elements > A[i]
    // the above takes one pass through A
    L ← Randomized-Quicksort(L)
    R ← Randomized-Quicksort(R)
    Return L · A[i] · R
end if
```

Analysis of Randomized Quicksort (0)

- The running time is proportional to the number of comparisons
- Let $b_1 \leq b_2 \leq \cdots \leq b_n$ be A sorted non-decreasingly
- For each $i < j$, let X_{ij} be the indicator random variable indicating if b_i was ever compared with b_j
- The expected number of comparisons is

$$\mathsf{E}\left[\sum_{i<j} X_{ij}\right] = \sum_{i<j} \mathsf{E}[X_{ij}] = \sum_{i<j} \text{Prob}[b_i \ \& \ b_j \text{ were compared}]$$

- b_i was compared with b_j if and only if either b_i or b_j was chosen as a pivot before any other in the set $\{b_i, b_{i+1}, \ldots, b_j\}$. They have equal chance of being pivot first. Hence, Prob$[b_i \ \& \ b_j$ were compared$] = \frac{2}{j-i+1}$
- Thus, the expected running time is $\Theta(n \lg n)$

Analysis of Randomized Quicksort (1)

- Uncomfortable? What is the sample space?
- Build a binary tree T, pivot is root, recursively build the left branch with L and right branch with R
- This process yields a random tree T built in n steps, t'th step picks tth pivot, pre-order traversal
- Collection $\mathcal{T}$ of all such trees is the sample space
- b_i & b_j compared iff one is an ancestor of the other in the tree T
- For simplicity, assume $b_1 < \cdots < b_n$.
- Define $I = \{b_i, b_{i+1}, \cdots, b_j\}$
- A_t = event that first member of I picked as a pivot at step t

Analysis of Randomized Quicksort (2)

- From law of total probability

$$\text{Prob}[b_i \text{ first pivot of } I] = \sum_t \text{Prob}[b_i \text{ first pivot of } I \mid A_t]\,\text{Prob}[A_t]$$

- At step t, all of I must belong to L or R of some subtree, say $I \subset L$
- At step t, each member of L chosen with equal probability
- Hence, each member of I chosen with equal probability
- Hence, conditioned on A_t, b_i chosen with probability

$$\frac{1}{|I|} = \frac{1}{j - i + 1}.$$

Example 2: Randomized Min-Cut

Min-Cut Problem

Given a multigraph G, find a cut with minimum size.

RANDOMIZED MIN-CUT(G)

for $i = 1$ **to** $n - 2$ **do**
    Pick an edge e_i in G uniformly at random
    Contract two end points of e_i (remove loops)
end for
// At this point, two vertices u, v left
Output all remaining edges between u and v

Analysis

- Let C be a minimum cut, $k = |C|$
- If no edge in C is chosen by the algorithm, then C will be returned in the end, and vice versa
- For $i = 1..n-2$, let A_i be the event that $e_i \notin C$ and B_i be the event that $\{e_1, \dots, e_i\} \cap C = \emptyset$

$$\begin{aligned}
& \text{Prob}[C \text{ is returned}] \\
= \; & \text{Prob}[B_{n-2}] \\
= \; & \text{Prob}[A_{n-2} \cap B_{n-3}] \\
= \; & \text{Prob}[A_{n-2} \mid B_{n-3}] \, \text{Prob}[B_{n-3}] \\
= \; & \dots \\
= \; & \text{Prob}[A_{n-2} \mid B_{n-3}] \, \text{Prob}[A_{n-3} \mid B_{n-4}] \cdots \text{Prob}[A_2 \mid B_1] \, \text{Prob}[B_1]
\end{aligned}$$

Analysis

- At step 1, G has min-degree $\geq k$, hence $\geq kn/2$ edges
- Thus,

$$\text{Prob}[B_1] = \text{Prob}[A_1] \geq 1 - \frac{k}{kn/2} = 1 - \frac{2}{n}$$

- At step 2, the min cut is still at least k, hence $\geq k(n-1)/2$ edges. Thus, similar to step 1

$$\text{Prob}[A_2 \mid B_1] \geq 1 - \frac{2}{n-1}$$

- In general,

$$\text{Prob}[A_j \mid B_{j-1}] \geq 1 - \frac{2}{n-j+1}$$

- Consequently,

$$\text{Prob}[C \text{ is returned}] \geq \prod_{i=1}^{n-2} \left(1 - \frac{2}{n-i+1}\right) = \frac{2}{n(n-1)}$$

How to Reduce the Failure Probability

- The basic algorithm has failure probability at most $1 - \frac{2}{n(n-1)}$
- How do we lower it?
- Run the algorithm multiple times, say $m \cdot n(n-1)/2$ times, return the smallest cut found
- The failure probability is at most

$$\left(1 - \frac{2}{n(n-1)}\right)^{m \cdot n(n-1)/2} < \frac{1}{e^m}.$$

PTCF: Mutually Independence and Independent Trials

- A set $A_1, \ldots, A_n$ of events are said to be independent or mutually independent if and only if, for any $k \leq n$ and $\{i_1, \ldots, i_k\} \subseteq [n]$ we have

$$\text{Prob}[A_{i_1} \cap \cdots \cap A_{i_k}] = \text{Prob}[A_{i_1}] \cdots \text{Prob}[A_{i_k}].$$

- If n independent experiments (or trials) are performed in a row, with the ith being "successful" with probability p_i, then

$$\text{Prob}[\text{all experiments are successful}] = p_1 \cdots p_n.$$

(Question: what is the sample space?)

Las Vegas and Monte Carlo Algorithms

Las Vegas Algorithm

A randomized algorithm which always gives the correct solution is called a Las Vegas algorithm.
Its running time is a random variable.

Monte Carlo Algorithm

A randomized algorithm which may give incorrect answers (with certain probability) is called a Monte Carlo algorithm.
Its running time may or may not be a random variable.

Example 3: Primality Testing

- Efficient Primality Testing is an important (practical) problem
- In 2002, Agrawal-Kayal-Saxena design a deterministic algorithm;
  - best current running time $\tilde{O}(\log^6 n)$, too slow
  - $\log n \approx 1024$ for 1024-bit crypto systems
- Actually, generating (large) random primes is also fundamental (used in RSA, e.g.)

  RANDOM-PRIME(n)

  1: $m \leftarrow$ RandInt(n) // random int $\leq n$
  2: **if** isPrime(m) **then**
  3:     Output m
  4: **else**
  5:     Goto 1
  6: **end if**

Expected Run-Time of Random-Prime

Theorem (Prime Number Theorem (Gauss, Legendre))

Let $\pi(n)$ *be the number of primes* $\leq n$*, then*

$$\lim_{n\to\infty} \frac{\pi(n)}{n/\ln n} = 1.$$

This means

$$\text{Prob}[m \text{ is prime}] = \frac{\pi(n)}{n} \approx \frac{1}{\ln n}.$$

Expected number of calls to isPrime(m) is thus $\ln n$.

Simple Prime Test based on Fermat Little Theorem

Theorem (Fermat, 1640)

If n is prime then $a^{n-1} = 1 \bmod n$, for all $a \in [n-1]$

SIMPLE-PRIME-TEST(n)

```
if 2^{n-1} ≠ 1 mod n then
    Return COMPOSITE // correct!
else
    Return PRIME // may fail, hopefully with small probability
end if
```

- Can show failure probability goes to 0 as $n \to \infty$
- Probability that a 1024-bit composite marked as prime is $\leq 10^{41}$

Composite Witness

Is-a-a-witness-for-n?(a, n)

// note: $a \in [n-1]$, and n is odd
Let $n - 1 = 2^t u$, u is odd
$x_0 \leftarrow a^u \bmod n$, // use repeated squaring
for $i = 1$ to t **do**
    $x_i \leftarrow x_{i-1}^2 \bmod n$
    if $x_i = 1$ and $x_{i-1} \neq \pm 1 \bmod n$ **then**
        return TRUE
    end if
end for
if $x_t \neq 1$ **then**
    return TRUE
end if
return FALSE

Theorem

If n is an odd composite then it has $\geq \frac{3}{4}(n-1)$ witnesses. If n is an odd prime then it has no witnesses.

Miller-Rabin-Test:

- return COMPOSITE if any of the r independent choices of a is a composite witness for n

Failure probability $\leq (1/4)^r$.

PTCF: Law of Total Expectation

- The conditional expectation of X given A is defined by

$$\mathsf{E}[X \mid A] := \sum_a a \,\mathsf{Prob}[X = a \mid A].$$

- Let $A_1, A_2, \dots$ be any partition of Ω, then

$$\mathsf{E}[X] = \sum_{i \geq 1} \mathsf{E}[X \mid A_i] \,\mathsf{Prob}[A_i]$$

- In particular, let Y be any discrete random variable, then

$$\mathsf{E}[X] = \sum_y \mathsf{E}[X \mid Y = y] \,\mathsf{Prob}[Y = y]$$

- We often write the above formula as

$$\mathsf{E}[X] = \mathsf{E}[\mathsf{E}[X \mid Y]].$$

Example 4: Max-E3SAT

- An E3-CNF formula is a CNF formula φ in which each clause has *exactly* 3 literals. E.g.,

$$\varphi = \underbrace{(x_1 \vee \bar{x}_2 \vee x_4)}_{\text{Clause } 1} \wedge \underbrace{(x_1 \vee x_3 \vee \bar{x}_4)}_{\text{Clause } 2} \wedge \underbrace{(\bar{x}_2 \vee \bar{x}_3 \vee x_4)}_{\text{Clause } 3}$$

- Max-E3SAT Problem: given an E3-CNF formula φ, find a truth assignment satisfying as many clauses as possible

A Randomized Approximation Algorithm for Max-E3SAT

- Assign each variable to TRUE/FALSE with probability $1/2$

Analyzing the Randomized Approximation Algorithm

- Let X_C be the random variable indicating if clause C is satisfied
- Then, $\text{Prob}[X_C = 1] = 7/8$
- Let S_φ be the number of satisfied clauses. Then,

$$\mathsf{E}[S_\varphi] = \mathsf{E}\left[\sum_C X_C\right] = \sum_C \mathsf{E}[X_C] = 7m/8 \geq \frac{\text{OPT}}{8/7}$$

  (m is the number of clauses)
- So this is a randomized approximation algorithm with ratio $8/7$

Derandomization with Conditional Expectation Method

- Derandomization is to turn a randomized algorithm into a deterministic algorithm
- By conditional expectation

$$\mathsf{E}[S_\varphi] = \frac{1}{2}\mathsf{E}[S_\varphi \mid x_1 = \text{ TRUE}] + \frac{1}{2}\mathsf{E}[S_\varphi \mid x_1 = \text{ FALSE}]$$

- Both $\mathsf{E}[S_\varphi \mid x_1 = \text{ TRUE}]$ and $\mathsf{E}[S_\varphi \mid x_1 = \text{ FALSE}]$ can be computed in polynomial time
- Suppose $\mathsf{E}[S_\varphi \mid x_1 = \text{ TRUE}] \geq \mathsf{E}[S_\varphi \mid x_1 = \text{ FALSE}]$, then

$$\mathsf{E}[S_\varphi \mid x_1 = \text{ TRUE}] \geq \mathsf{E}[S_\varphi] \geq 7m/8$$

- Set $x_1 = \text{TRUE}$, let φ' be φ with c clauses containing x_1 removed, and all instances of $x_1, \bar{x}_1$ removed.
- Recursively find value for x_2

Some Key Ideas We've Learned

- To compute $\mathsf{E}[X]$, where X "counts" some combinatorial objects, try to "break" X into $X = X_1 + \cdots + X_n$ of indicator variables
- Then,
$$\mathsf{E}[X] = \sum_{i=1}^{n} \mathsf{E}[X_i] = \sum_{i=1}^{n} \mathsf{Prob}[X_i = 1]$$
- Also remember the law of total probability and conditional expectation