LECTURE 6 (September 11th)

Probability Theory Review

$\boxed{\text{Motivation}}$ Randomized Algorithms

For the next few weeks, we will look at randomized algorithms

These are algorithms which, apart from doing anything a deterministic algorithm does, also have access to a library function

Random (k) $\longrightarrow$ outputs a uniformly random number in $\{1, 2, \ldots\ldots, k\}$

We will assume axiomatically that such a library function can be implemented and not worry about generating perfect random numbers

Why would we want to do this?

$\boxed{1}$ Faster and Simpler Algorithms

Worst-case run time of an algorithm $= \max\limits_{|x|=n} T(x)$

Example Recall, the quicksort algorithm does the following

Given an array $A = \boxed{\quad\quad\quad | p | \quad\quad\quad}$

- Pick a pivot p
- Put all elements $<p$ to the left of p in the array
  $(>p)$ (right)

Worst-case complexity $= \theta(n^2)$ for any simple pivot rule

If we pick the pivot randomly, then worst-case expected run-time is $O(n \cdot \log n)$

Now, we are looking at $\max\limits_{|x|=n} \mathbb{E}[T(x)]$ ↳ over the randomness used in the algorithm

In fact, $O(n \log n)$ time with high probability for any input x
so, in practice, we have an $O(n \cdot \log n)$ time algorithm

Note : This is not average-case analysis

We are not assuming anything about the worst-case input
An adversary who knows the algorithm can pick the worst-case input
But the algorithm generates its own randomness on the fly
which the adversary does not know

[2] No known deterministic algorithms

Example Generating Prime Numbers

Oversimplifying, the RSA cryptosystem which forms the backbone of most practical encryption is based on the following fact :

- Take two large primes p and q
- Release $n = p \cdot q$ as the public-key
- Given n one can encrypt the message, but to decrypt, one needs its prime factors, which is believed to be a hard task

So, this leads to the question : how do we find large prime numbers ?
i.e.

Given n, find a prime in $\underbrace{[n, 2n]}_{\text{Bertrand's postulate guarantees its existence}}$ in time $\text{poly}(\underbrace{\log(n)}_{\text{input-size of } n})$

There is no deterministic algorithm known to generate / find primes but we can check if a given number is prime or not

Prime number Theorem # primes in an interval of size $n \approx \dfrac{n}{\log n}$

Algorithm Pick a random number in $[n, 2n]$
Accept if it is prime & repeat otherwise

Whp in $\log^4 n$ time, algorithm outputs a prime number

Probability Theory

A discrete probability space $(\Omega, \mathbb{P})$ has two components :

[1] Sample space Ω This is a non-empty countable set. You can imagine these as the set of all things that can possibly happen

2 Probability measure $\mathbb{P}$

This is a function $\mathbb{P} : \Omega \to \mathbb{R}$ satisfying

$$\mathbb{P}[\omega] \geq 0 \quad \forall\, \omega \in \Omega \quad \text{and} \quad \sum_{\omega \in \Omega} \mathbb{P}[\Omega] = 1$$

Example

- Fair Coin $\Omega = \{H, T\}$, $\mathbb{P}[H] = \mathbb{P}[T] = 1/2$
- Biased coin $\Omega = \{H, T\}$, $\mathbb{P}[H] = 1/3$ and $\mathbb{P}[T] = 2/3$
- Fair six-sided die
- Random card from a deck

Events These are subsets of Ω. In particular, singleton sets $\{\omega\}$ are called elementary events or atoms. You can think of these as all possible yes-no questions, you can ask about the things that happen

If $\mathcal{E} \subseteq \Omega$ is an event, its probability

$$\mathbb{P}[\mathcal{E}] = \sum_{\omega \in \Omega} \mathbb{P}[\omega] \qquad \text{e.g. } \mathbb{P}[\emptyset] = 0, \quad \mathbb{P}[\Omega] = 1$$

Note: we have extended the function $\mathbb{P} : \Omega \to [0,1]$ to a function $\mathbb{P} : 2^{\Omega} \to [0,1]$

Example

- Roll two fair die, one red and the other blue

$$\Omega = \{1,2,3,4,5,6\} \times \{1,2,3,4,5,6\}$$

$$\mathbb{P}[\omega] = \frac{1}{36} \quad \forall\, \omega \in \Omega$$

$$\mathbb{P}[\text{two } 5\text{'s}] = \mathbb{P}[(5,5)] = \frac{1}{36}$$

$$\mathbb{P}[\text{at most one } 5\text{'s}] = \mathbb{P}[\neg \text{ two } 5\text{s}] = \frac{35}{36}$$

Typically, we use boolean logic operation to denote combination of events

E.g. $A \cup B$ by $A \vee B$

$A \cap B$ by $A \wedge B$

$\bar{A}$ by $\neg A$

Events A and B are disjoint if $A \cap B = \emptyset$

Conditional Probability Probability of A conditioned on B is defined as

$$\mathbb{P}[A \mid B] = \frac{\mathbb{P}[A \wedge B]}{\mathbb{P}[B]}$$

Example, $\mathbb{P}[\text{red } 5 \mid \text{at least one } 5] = \dfrac{\mathbb{P}[\text{red } 5 \wedge \text{atleast one } 5]}{\mathbb{P}[\text{at least one } 5]}$

$$= \frac{\frac{1}{6}}{1 - \left(\frac{5}{6}\right)^2} = \frac{6}{11}$$

Remark, Conditional probability is unintuitive

Puzzle by Gary Foshee

I have two children. One of them is a boy. What is the probability that I have two boys?

I have two children. One of them is a boy born on a Tuesday. What is the probability that I have two boys?

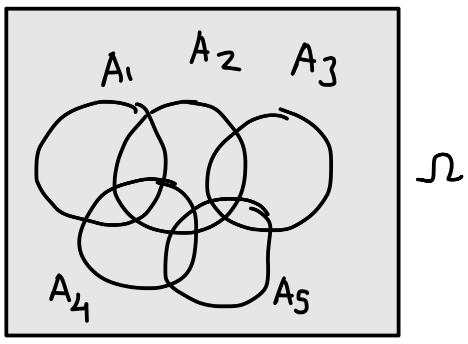
Independence Two events A and B are independent iff

$$\mathbb{P}[A \wedge B] = \mathbb{P}[A] \cdot \mathbb{P}[B] \text{ or equivalently}$$

$$\mathbb{P}[A \mid B] = \mathbb{P}[A] \text{ and vice-versa}$$

Remark Independence and disjointness are two very different conditions

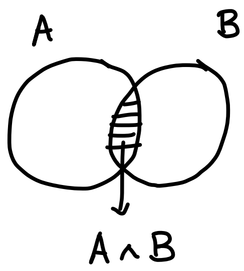
Basic Identities Union bound $A_1, \dots A_n$ are events in $(\Omega, \mathbb{P})$



Then, $\mathbb{P}\left[\bigvee_{i=1}^{n} A_i\right] \leq \sum_{i=1}^{n} \mathbb{P}[A_i]$

If A_i's were pairwise disjoint, then we get an equality above

Inclusion - Exclusion $\mathbb{P}[A \vee B] = \mathbb{P}[A] + \mathbb{P}[B] - \mathbb{P}[A \wedge B]$



For events $A_1, \dots, A_n$, we get

$$\mathbb{P}\left[\bigvee_{i=1}^{n} A_i\right] = 1 - \sum_{I \subseteq [n]} (-1)^{|I|} \, \mathbb{P}\left[\bigwedge_{i \in I} A_i\right]$$

Independent Union If A and B are independent events, then A^c and B^c, A and B^c, A^c and B are also independent [Prove it]
This also generalizes to n mutually or fully independent events

If $A_1, \dots, A_n$ are independent

$$\mathbb{P}\left[\bigvee_{i=1}^{n} A_i\right] = 1 - \mathbb{P}\left[\bigwedge_{i=1}^{n} A_i^c\right]$$
$$= 1 - \prod_{i=1}^{n}(1 - \mathbb{P}[A_i])$$

E.g. Toss n independent unbiased coin
A_i = "i^{th} coin is H"

$$\mathbb{P}[\text{atleast one } H] = \mathbb{P}\left[\bigvee_{i=1}^{n} A_i\right] = 1 - \left(\frac{1}{2}\right)^n$$

Bayes' Rule $$\mathbb{P}[A \wedge B] = \mathbb{P}[A] \cdot \mathbb{P}[B|A]$$
$$= \mathbb{P}[B] \cdot \mathbb{P}[A|B]$$

Equality Testing

Given two binary vectors $u, v \in \{0,1\}^n$
Decide if they are equal or not

Only operation that is allowed : DOTPRODUCT(a,b) $\longrightarrow$ Time $B(n)$

take dot product (mod 2) of any two binary vectors $a, b \in \{0,1\}^n$
i.e. output is

$$\underbrace{\langle a,b\rangle}_{=\sum_{i=1}^{n} a_i b_i} \bmod 2 = \sum_{i=1}^{n} a_i b_i \pmod 2 = \begin{cases} 1 & \text{if } \langle a,b\rangle \text{ is odd} \\ 0 & \text{o/w} \end{cases}$$

Algorithm
- Pick a random vector $r \in \{0,1\}^n$
- If $\langle u,r\rangle = \langle v,r\rangle \bmod 2$, then output EQUAL
- Else output NOT EQUAL

$\boxed{\text{Theorem}}$ $\mathbb{P}[\text{Algorithm errs}] \leq \frac{1}{2}$ and is running time is $\underbrace{O(n + B(n))}_{\text{obvious}}$

$\hookrightarrow$ Proof in the next lecture

Repetition/Amplification Trick Run the algorithm $t = \lceil \log \frac{1}{\delta} \rceil$ times independently
If any execution says NOT EQUAL $\Rightarrow$ output NOT EQUAL
o/w $\Rightarrow$ output EQUAL

Again, algorithm only errs if $u \neq v$,

$$\mathbb{P}[\text{Algorithm errs}] = \mathbb{P}[\text{all } i \text{ iteration return EQUAL}]$$

$$= \prod_{i=1}^{t} \frac{1}{2} = 2^{-t} = 2^{-\lceil \log 1/\delta \rceil} \leq \delta$$

Runtime is now $O(n + B(n) \cdot \log \frac{1}{\delta})$