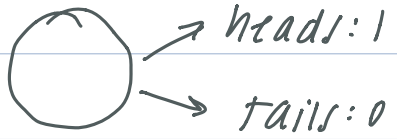
CS121 Section 9 November 12

Midterm Review

- lectures 11-21 (Restricted models to modeling randomized computation)
- Reductions are important
  - write definitions of NP-hard problems
  - do practice reductions

Basics of Probability



$X \in \{0,1\}^1 \rightarrow X_n \in \{0,1\}^n$

n flips of a coin

sample space for n=2

11 (B)	10 (C)
01	00

Ω = sample space

C = event we get 10

B = event both flips same

$P(C) = \frac{1}{4}$, $P(B) = \frac{1}{2}$ } assuming independence of flips

Independence

$P(A \cap B) = P(A) \cdot P(B)$

Random Variables

$X(A) \rightarrow$ some output

ex: $X(11) = \$10$

$X(01) = \$50$

$Z_i = 1$ when event that i^{th} flip is 1

$Z = \sum_{i=0}^{n} Z_i$ = total # of 1s

$$E[X] = \sum_{X \in \{0,1\}^n} \frac{1}{2^n} \cdot X(X)$$

Randomized Algorithms

Benefit: can compute harder algos w/ tradeoff

BPP

Let $F: \{0,1\}^* \to \{0,1\}$. $F \in BPP$ iff $\exists\, a, b \in \mathbb{N}$, $\exists\, G \in \{0,1\}^* \to \{0,1\}$ s.t.

① $G \in P$ → can write such a poly-time algo G

② $\forall X \in \{0,1\}^*$

$$\Pr_{r \sim \{0,1\}^{a|X|^b}} [G(X,r) = F(X)] \geq \frac{2}{3}$$

→ random string, polynomial in length of X

Walksat, randomized max cut are ex. of algos in BPP

Q1. Is maxcut in BPP?

Potential evidence: (given $P_{MC} \in P$)

- $\exists$ program P_{MC} s.t. for all but 1 particular graph G, (needs to work for every G)

  $P_{MC}(G, k, r) = MAXCUT(G, k)$ ✗

- On all inputs to maxcut, P_{MC} is only right 70% of the time for each input ✓

- On all inputs to maxcut, P_{MC} is only right 70% of the time for all inputs ✗ (→ has to be for each)

Q2. $P \subseteq BPP$? yes, throw away randomness bits r

$BPP \subseteq EXP$? yes, iterate through all 2^{r} inputs ($2^{a|x|^b} \cdot poly$) and majority vote

$BPP \subseteq P/poly$? yes, for inputs of fixed size n, we can show there is a specific r that is correct for all values of x

$BPP \subseteq NP$? Unknown

$NP \subseteq BPP$? Unknown

Example randomized algorithm: MAXCUT

MAXCUT(G, k)

for any graph G w/ n vertices and m edges, compute MAXCUT(G, m/2).

$P_{cut}(G)$ returns potential cut S of graph G

- $z \tilde{}_{unif} \{0,1\}^r$, where z is random seed
- put vertices $\{i \mid z_i = 1\}$ in S
- define some r.v. X_j: $\begin{cases} 1 & \text{if edge } j \text{ is cut by } S \\ 0 & \text{o/w} \end{cases}$

$X = \sum_{j=0}^{m} X_j$

- Claim: $\mathbb{E}[X] = m/2$

for any edge j between vertices a and b

(z_a, z_b in S?)

$X_j = 1$, edge is cut $\equiv z_a \neq z_m$ (one in S, one in S')

↖ 1/2 probability

$$E[X] = E\left[\sum_{j=0}^{m} X_j\right] = \sum_{j=0}^{m} E[X_j] = m\left(\frac{1}{2}\right)$$

Lemma (18.2): $Pr[X \geq m/2] \geq \frac{1}{2m}$

prob. randomly assigning vertices is at least half is $1/2m$.

- put vertices $\{i \mid z_i = 1\}$ in S. compute $k :=$ edges crossing

  if $(k \geq m/2)$: return 1

  else: return 0

Amplification: if prob of success is $\geq 1/2m$,

$Pr[X < m/2] \leq (1 - \frac{1}{2m})$

probability of failure

probability of failing after running t times: $(1 - \frac{1}{2m})^t$

① one-sided

program P may output 1 when $F(X) = 0$

$P(X) = 0$ when $F(X) = 1$ → only wrong in one of cases

max cut only has this kind of error

② two-sided

- both errors can happen

Example randomized algorithm: WALKSAT

WALKSAT(φ):

do T times:

$X \sim_{unif} \{0,1\}^n$ ← initial assignments to variables

do S times:

if $\varphi(X) == 1$: return 1

else:

choose random clause $(l_1 \vee l_2 \vee l_3)$ ← that is not satisfied

choose random variable in the clause (flip x or l_2)

flip its value

$\exists$ assignment X^* s.t. $\varphi(X^*) = 1$.

$\Delta(X, X^*)$ = # of variable assignments that differ between X and X^*

Claim. In every local improvement step s, the [prob that the $\Delta(X, X^*)$ decreases by 1] is at least $1/3$.

Intuition:

unsatisfied clause $(X_1 \vee X_2 \vee X_3)$

X_1	0	X_1^*	1
X_2	1	X_2^*	1
X_3	0	X_3^*	0

CS 121: Introduction to Theoretical Computer Science Fall 2019

Section 9 — November 9, 2017

Prof. Boaz Barak *Pratap Singh, Prayaag Venkat*

1 Probability Theory

In this section we will review the basic notion of probability theory that we will use.

- The basic probabilistic experiment corresponds to tossing n coins or choosing x uniformly at random from $\{0,1\}^n$.

- Random variables assign a real number to every result of a coin toss. The expectation of a random variable is its average value (also called the *mean*). There are several "concentration" results stating that (under certain conditions) the probability that a random variable deviates significantly from its expectation is small.

- Throughout, we will study the running example of the random variable $Z = \sum_{i=1}^{n} Z_i$, where $Z_1, \ldots, Z_n$ are independent, unbiased bits. That is, each Z_i takes the values $0, 1$ with probability $\frac{1}{2}$ each. In short, Z is a count of the number of heads after n fair coin tosses.

1.1 Random Coins

- Setting: toss n random, unbiased and independent coins.
- Event: subset A of $\{0,1\}^n$. Probability of A is $\mathbb{P}_{x\sim\{0,1\}^n}[A] = \frac{|A|}{2^n}$.

1.2 Random Variables

- Expectation of a random variable X is denoted by $\mathbb{E}[X]$, is the average value that this number takes, taken over all draws from the probabilistic experiment.

$$\mathbb{E}[X] = \sum_{x\in\{0,1\}^n} 2^{-n} X(x)$$

- For the coin tossing example,

$$\mathbb{E}[Z] = \sum_{i=1}^{n} \mathbb{E}[Z_i] = \frac{n}{2}$$

1.3 Correlations and Independence

- Two events A and B are independent if the fact that A happened does not make B neither more not less likely to happen. Think about the experiment of tossing three random coins described in the lecture notes. The formal definition is that events A and B are independent if $\mathbb{P}[A \cap B] = \mathbb{P}[A] \cdot \mathbb{P}[B]$.
- Positively correlated: $\mathbb{P}[A \cap B] > \mathbb{P}[A] \cdot \mathbb{P}[B]$
- Negatively correlated: $\mathbb{P}[A \cap B] < \mathbb{P}[A] \cdot \mathbb{P}[B]$

1.4 Concentration

In this section, we want to capture the following intuition mathematically: if I toss 100 fair coins, how likely is it that I see a number of heads between 40 and 60? Now, if I toss 1000 coins, how likely is it that I see a number of heads between 400 and 600?

- Much of probability theory is concerned with so called concentration or tail bounds, which are upper bounds on the probability that a random variable X deviates too much from its expectation. The simplest one of these is:

  Theorem 1 (Markov's Inequality)**.** *If X is a non-negative random variable then $\mathbb{P}[X \geq k\mathbb{E}[X]] \leq 1/k$.*

  For the coin tossing example, we already calculated $\mathbb{E}[Z] = \frac{n}{2}$. Taking $k = 1.1$, for example, Markov's Inequality tells us that $\mathbb{P}[Z \geq 0.55n] \leq 0.91$, which is a pretty weak statement.

- A standard way to measure the deviation of a random variable from its expectation is using its standard deviation. For a random variable X, we define the variance of X as $Var[X] = \mathbb{E}[X - \mu]^2$ where $\mu = \mathbb{E}[X]$, i.e., the variance is the average square distance of X from its expectation. The standard deviation of X is defined as $\sigma[X] = \sqrt{Var[X]}$. (This is well defined since the variance, being an average of a square, is always a non-negative number.)

  Using Chebychev's inequality we can control the probability that a random variable is too many standard deviations away from its expectation.

  Theorem 2 (Chebyshev's Inequality)**.** *Suppose $\mu = \mathbb{E}[X]$ and $\sigma^2 = Var[X]$. Then for every $k > 0$, $\mathbb{P}[|X - \mu| \geq k\sigma] \leq 1/k^2$.*

  While Chebyshev's Inequality is more powerful, we need to calculate the variance to use it. For the coin tossing example, $Var[Z] = \sum_{i=1}^{n} Var[Z_i] = n/4$ (justify why this is true!). Chebyshev's Inequality then tells us for $k = 1.1$:

  $$\mathbb{P}[|Z - \frac{n}{2}| \geq .78\sqrt{n}] \leq .83,$$

  which is definitely better than what Markov gave, but still not what we should expect.

- The following extremely useful theorem shows that such exponential decay occurs every time we have a sum of independent and bounded variables. This theorem is known under many names in different communities, though it is mostly called the Chernoff bound in the computer science literature.

  Theorem 3 (Chernoff Bound)**.** *If $X_0, .., X_{n-1}$ are i.i.d random variables such that $X_i \in [0, 1]$ and $\mathbb{E}[X_i] = p$ for every i, then for every $\epsilon > 0$:*

  $$\mathbb{P}[|\sum_{i=0}^{n-1} X_i - pn| > \epsilon n] \leq 2\exp(-\epsilon^2 n/2)$$

  Now, applying this to the coin tossing example, with $p = \frac{1}{2}$ and $\epsilon = .1$, we get

  $$\mathbb{P}[|Z - \frac{n}{2}| > .1n] \leq 2\exp(-.005n).$$

  This is much stronger; it guarantees that the probability of deviations from the mean beyond $.1n$ are *exponentially small*. Compare this with Markov and Chebyshev, which only give *polynomial* decay. This is the strongest of the three bounds, but also requires the most assumptions.

2 Probability Problems

2.1 Random Variable Example

Give an example of random variables $X, Y : \{0,1\} \to R$ such that $\mathbb{E}[XY] \neq \mathbb{E}[X]\mathbb{E}[Y]$.

2.2 Example 19.8

Prove that $\mathbb{P}_{x\sim\{0,1\}^n}[\sum x_i = k] = \binom{n}{k}2^{-n}$.

≡ exactly k of n coins landed on heads (1)

$\binom{n}{k}$ = # ways to choose the k coins, $\left(\frac{1}{2}\right)^n$ probability of those k coins = 1 and the other $n-k$ coins are = 0.

3 Randomized Algorithms

If we augment our models of computation to be able to toss coins we can design algorithms that can take advantage of randomness. In some cases, adding randomness has enabled the development of algorithms that are faster than known deterministic algorithms.

When we analyze the performance of randomized algorithms, we typically analyze *expected* performance, either in terms of the quality of the answer or in terms of runtime.

As an example, consider the randomized max cut algorithm from the lecture notes:

Definition 4 (Max-Cut problem)**.** Given a graph $G = (V, E)$ find a cut $C \subset V$ that maximizes the quantity:

$$\Big|\{(a,b) \in E \mid (a \in C \text{ and } b \notin C) \text{ or } (a \notin C \text{ and } b \in C)\}\Big|$$

that is, maximizes the number of edges that cross the cut.

The lecture notes proposed a simple randomized algorithm for this problem:

Example 5 (Randomized Max-Cut)**.** Start with $C = \emptyset$. Then for each $v \in V$ set $C \leftarrow C \cup \{v\}$ with probability $\frac{1}{2}$. In simpler terms, for each vertex we flip a coin and if it is heads we add that vertex to the cut.

Note that the runtime of this algorithm is $O(|V|)$ because for each vertex we only need to flip a coin to decide whether it is in the cut. Because the algorithm is randomized, to effectively discuss its performance we need to use the language of probability. As noted in the lecture notes, if $m = |E|$, the cut will have *expected* size $m/2$. However, note that in theory the algorithm could fail and produce a cut with smaller size.

Next, we work through the analysis of WalkSAT from the lecture notes. Recall that the 3SAT problem determines whether for a boolean expression of a specified format, there is an assignment of its variables that makes it evaluate to true (satisfies it).

The format for 3SAT is an AND of clauses of ORs of three variables (or their negations). For example,

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$$

is in 3-CNF, as needed for 3SAT.

One approach to solving 3SAT in a randomized way is just to try random assignments for all the variables. Also, even if our random assignment doesn't work, we can try to improve it. To avoid getting stuck after several tries on a particular random assignment, we will "give up" and pick a new starting random assignment. After enough rounds of this, we can give up on the whole thing and decide that the boolean expression is unsatisfiable.

Definition 6 (WalkSAT)**.** We write this process out in more detail. Let φ be the boolean expression provided to the algorithm and let n be the number of variables in φ.

for T steps **do**
  Choose $x \leftarrow \{0,1\}^n$ uniformly at random
  for S steps **do**
    if $\varphi(x) = 1$ **then**
      return x
    end if
    Choose random clause (x_i, x_j, x_k) not satisfied
    Choose random literal from $x_\ell \leftarrow \{x_i, x_j, x_k\}$
    Modify x to satisfy x_ℓ (flip the bit)
  end for
end for
return `Unsatisfiable`

In the lecture notes we saw that setting $T = 100 \cdot 3^{n/2}$ and $S = n/2$ got us reasonably good performance.

Clearly, if φ is not satisfiable WalkSAT will never find a satisfying solution and we will correctly return `Unsatisfiable`. What is the probability that we will fail for φ that *is* satisfiable? That is, what is the probability that WalkSAT will return `Unsatisfiable` even though φ is satisfiable?

For this proof we want to determine the probability of finding a satisfying assignment during the inner for loop (where we improve x). What is the probability that for random x we will walk our way to a satisfying assignment? Then, with this, we figure out the probability that during the T iterations of the outer loop we will fail each time to find a satisfying assignment.

Example 7 (Proof of Performance)**.** More specifically, let $\Delta(x, x')$ be the number of bit positions at which x and x' differ. For example,

$$\Delta(111, 001) = 2 \qquad \text{and} \qquad \Delta(101, 010) = 3.$$

Then let x^* be a satisfying assignment for φ. At each of the inner loop iterations we show that we decrease Δ by one with probability at least $\frac{1}{3}$.

Therefore if $\Delta(x^*, x) \leq n/2$ with probability at least $(1/3)^{n/2}$ we will randomly walk from x to x^*. This probability bound comes from the fact that one way to go from $x \to x^*$ is to make the "right" decisions at each random improvement step.

The probability of succeeding during any one inner loop is very low. This is why we run the outer loop many (T) times. This is an example of *amplification*, where we increase our overall probability of success by repeating some randomized process many times.

Next, we walk through the several lemmas needed for the proof.

Exercise 1 (Probability of Improvement)**.** Show that at each improvement step, the probability of decreasing $\Delta(x, x^*)$ is at least $1/3$.

This shows that starting from $\Delta(x, x^*) \leq n/2$ and taking $n/2$ steps we will walk to a satisfying solution with probability at least $(1/3)^{n/2} = \sqrt{3}^{-n}$. However, this requires that $\Delta(x, x^*) \leq n/2$. What is the probability of that?

Exercise 2. What is the probability that $\Delta(x, x^*) \leq n/2$?

Exercise 3 (Putting it Together)**.** What is the probability that a single iteration of the inner loops succeeds in finding a satisfying assignment? What is our overall probability of *failure* after all $S \cdot T$ iterations of both loops?