Lisa Yan
CS 109

Section #5
Oct 30-Nov 1, 2019

Section 5 Solution

Adapted for Autumn 2019 by Alex Tsun

Congratulations on making it through the midterm! :)

1. **Binary Tree**: Consider the following function for constructing binary trees:

```
struct Node {
    Node *left;
    Node *right;
};

Node *randomTree(float p) {
    if (randomBool(p)) {  // returns true with probability p
        Node *newNode = new Node;
        newNode->left = randomTree(p);
        newNode->right = randomTree(p);
        return newNode;
    } else {
        return nullptr;
    }
}
```

The `if` branch is taken with probability p (and the `else` branch with probability $1 - p$). A tree with no nodes is represented by `nullptr`; so a tree node with no left child has `nullptr` for the `left` field (and the same for the right child).

Let X be the number of nodes in a tree returned by `randomTree`. You can assume $0 < p < 0.5$. What is $E[X]$, in terms of p?

Let X_1 and X_2 be number of nodes the left and right calls to `randomTree`.
$E[X_1] = E[X_2] = E[X]$.

$$\begin{aligned} E[X] &= p \cdot E[X \mid \texttt{if}] + (1-p)E[X \mid \texttt{else}] \\ &= p \cdot E[1 + X_1 + X_2] + (1-p) \cdot 0 \\ &= p \cdot (1 + E[X] + E[X]) \\ &= p + 2pE[X] \\ (1-2p)E[X] &= p \\ E[X] &= \frac{p}{1-2p} \end{aligned}$$

2. **Hat-Check Again??** Recall the hat-check problem from section 2: n people go to a party and drop off their hats to a hat-check person. When the party is over, a different hat-check person is on duty, and returns the n hats randomly back to each person. Let X be the random variable representing the number of people who get their own hat back. We showed last time that $E[X] = 1$ for any n. What is $Var(X)$? Hint: Be careful when taking the variance of a sum of random variables.

Similarly to last time, let $X_i \sim Bernoulli(p = 1/n)$ be the indicator variable for whether the i^{th} person gets their hat back, so that $X = \sum_{i=1}^{n} X_i$. Then,

$$Var(X) = Var\left(\sum_{i=1}^{n} X_i\right) = \sum_{i=1}^{n} Var(X_i) + 2\sum_{i<j} Cov(X_i, X_j)$$

The first term is simply $np(1-p) = n\left(\frac{1}{n}\right)\left(1 - \frac{1}{n}\right) = \frac{n-1}{n}$ since each individual variance is $p(1-p)$.

To compute $Cov(X_i, X_j)$ for $i \neq j$, we note that $Cov(X_i, X_j) = E[X_iX_j] - E[X_i]E[X_j]$. The random variable X_iX_j is also Bernoulli, with parameter $1/n \cdot 1/(n-1)$ since both have to get their hat back. So

$$Cov(X_i, X_j) = E[X_iX_j] - E[X_i]E[X_j] = \frac{1}{n(n-1)} - \left(\frac{1}{n}\right)^2 = \frac{1}{n^2(n-1)}$$

Noting that there are $\binom{n}{2} = \frac{n(n-1)}{2}$ identical terms in the summation over $i < j$ and putting this all together gives

$$Var(X) = \frac{n-1}{n} + 2\binom{n}{2}\frac{1}{n^2(n-1)} = \frac{n-1}{n} + \frac{1}{n} = 1.$$

So $E[X] = Var(X) = 1$. What a coincidence!

3. **Timing Attack**:

In this problem we are going to show you how to crack a password in linear time, by measuring how long the password check takes to execute (see code below). Assume that our server takes T ms to execute any line in the code where $T \sim N(\mu = 5, \sigma^2 = 0.5)$ seconds. The amount of time taken to execute a line is always independent of other values of T.

```
# An insecure string comparison
def stringEquals(guess, password):
   nGuess = len(guess)
   nPassword = len(password)
```

```
    if nGuess != nPassword:
       return False                   # 4 lines executed to get here
    for i in range(nGuess):
       if guess[i] != password[i]:
          return False                # 6 + 2i lines executed to get here
    return True                       # 5 + 2n lines executed to get here
```

On our site all passwords are length 5 through 10 (inclusive) and are composed of lower case letters only. A hacker is trying to crack the root password which is “gobayes" by carefully measuring how long we take to tell them that her guesses are incorrect.

a. What is the distribution of time that it takes our server to execute k lines of code? Recall that each line independently takes $T \sim N(\mu = 5, \sigma^2 = 0.5)$ ms.

Let Y be the amount of time to execute k lines. $Y = \sum_{i=1}^{k} X_i$ where X_i is the amount of time to execute line i. $X_i \sim N(\mu = 5, \sigma^2 = 0.5)$.
Since Y is the sum of independent normals:

$$\begin{aligned} Y &\sim N(\mu = \sum_{i=1}^{k} 5, \sigma^2 = \sum_{i=1}^{k} 0.5) \\ &\sim N(\mu = 5k, \sigma^2 = 0.5k) \end{aligned}$$

b. First the hacker needs to find out the length of the password. What is the probability that the time taken to test a guess of correct length (server executes 6 lines) is longer than the time taken to test a guess of an incorrect length (server executes 4 lines)? Assume that the first letter of the guess does not match the first letter of the password. Hint: $P(A > B)$ is the same as $P(A - B > 0)$.

From last problem:
Time to run 6 lines of code $A \sim N(\mu = 30, \sigma^2 = 3)$
Time to run 4 lines of code $B \sim N(\mu = 20, \sigma^2 = 2)$

$$\begin{aligned} -B &\sim N(\mu = -20, \sigma^2 = 2) \\ A - B &\sim N(\mu = 10, \sigma^2 = 5) \\ P(A > B) &= P(A - B > 0) \\ &= 1 - F_{A-B}(0) \\ &= 1 - \Phi\Big(\frac{0 - 10}{\sqrt{5}}\Big) \\ &\approx 1.0 \end{aligned}$$

c. Now that our hacker knows the length of the password, to get the actual string she is going to try and figure out each letter one at a time, starting with the first letter. The hacker tries the string “aaaaaaa" and it takes 27s. Based on this timing, how much more probable is it that first character did not match (server executes 6 lines) than the first character did match (server executes 8 lines)? Assume that all letters in the alphabet are equally likely to be the first letter.

Let M be the event that the first letter matched.

$$\begin{aligned}
\frac{P(M^C|T=27)}{P(M|T=27)} &= \frac{f(T=27|M^C)P(M^C)}{f(T=27|M)P(M)} \\
&= \frac{f(T=27|M^C)\frac{25}{26}}{f(T=27|M)\frac{1}{26}} \\
&= 25 \cdot \frac{f(T=27|M^C)}{f(T=27|M)} \\
&= 25 \cdot \frac{\frac{1}{\sqrt{6\pi}}e^{-\frac{(27-30)^2}{6}}}{\frac{1}{\sqrt{8\pi}}e^{-\frac{(27-40)^2}{8}}} \\
&= 25 \cdot \frac{\sqrt{8}}{\sqrt{6}} \cdot \frac{e^{-\frac{9}{6}}}{e^{-\frac{169}{8}}} \\
&\approx 9.6 \text{ million}
\end{aligned}$$

d. If it takes the hacker 6 guesses to find the length of the password, and 26 guesses per letter to crack the password string, how many attempts does she need to crack our password, “gobayes"? Yikes!

$7 \cdot 26 + 6 = 188$