Chris Piech
CS109

Win 2022
Feb 8th

CS109 Midterm Exam

This is a closed calculator/computer exam. You are, however, allowed to use notes in the exam. You have 2 hours (120 minutes) to take the exam. The exam is 120 points, meant to roughly correspond to one point per minute of the exam. You may want to use the point allocation for each problem as an indicator for pacing yourself on the exam.

In the event of an incorrect answer, any explanation you provide of how you obtained your answer can potentially allow us to give you partial credit for a problem. For example, describe the distributions and parameter values you used, where appropriate. It is fine for your answers to include summations, products, factorials, exponentials, and combinations.

You can leave your answer in terms of Φ (the CDF of the standard normal). For example $\Phi(3/4)$ is an acceptable final answer..

I acknowledge and accept the letter and spirit of the honor code. I pledge to write more neatly than I have in my entire life:

Signature: ______________________________

Family Name (print): ______________________________

Given Name (print): ______________________________

Email (preferably your gradescope email): ______________________________

1 Stanford Life [15 points]

a. (5 points) You and your friend agree to meet up at “Coupa Cafe”. There are five Coupa Cafes at Stanford and Palo Alto (really). You are both equally likely to go to any Coupa Cafe, and your choice is independent of one another. What is the probability you go to the same Coupa?

b. (10 points) Planes fly over Stanford at an average rate of 1 per minute (independently). You play a game with your friend: If exactly 0 **or** exactly 2 planes fly over Stanford in the next 5 minutes, you win. How likely are you to win?

2 Midterm Theory [10 points]

Let's consider the relationship between knowing the concepts used in a midterm question, and getting the question correct, taking into account guessing and making silly mistakes.
Let $p_a = 3/4$ be the probability that a learner knows the concepts to the midterm question.
Let $p_g = 1/4$ be the probability that a learner gets the answer correct if they **don't** know the concepts.
Let $p_s = 1/10$ be the probability that a learner gets the question incorrect given they **do** know the concepts.

What is the probability that a learner gets the question correct?

3 Algorithmic Analysis [10 points]

You are writing an algorithm that has two steps.
Step 1: create all possible subsets of 3 objects, chosen from 26 distinct objects (without replacement).
Step 2: run a test on each unique pair of sets.

As an example, imagine the 26 objects are letters in the alphabet.
Examples of subsets of 3 objects: ABC, DEF, CAT.
Examples of unique pairs of sets: (ABC, DEF), (ABC, CAT) and (CAT, MAT).
Note that the order in pairs does not matter so (CAT, MAT) is not distinct from (MAT, CAT).

How many unique pairs of sets are there?

4 ASL Fingerspelling [15 points]

In American Sign Language, some words are "fingerspelled" one letter at a time. A natural ASL signer can read these words very fast. This is possible because the reader doesn't need to be sure about every letter in order to understand what word is being spelled[1]. Let us work through an example where we are reading a 3 letter word. Here are the 3 hand shapes we saw and what letters we think they represent:

First Letter

Letter	Letter Probability
C	0.95
All other letters	Equally likely

Second Letter

Letter	Letter Probability
A	0.4
S	0.4
All other letters	Equally likely

Third Letter

Letter	Letter Probability
T	0.3
N	0.6
All other letters	Equally likely

Let L_1 be the first letter. Based on the table above, $P(L_1 = \text{`C'}) = 0.95$. All the letters which are not `'C'` are equally likely, so $P(L_1 = \text{`A'}) = P(L_1 = \text{`B'}) = P(L_1 = \text{`M'})$ and so on.

a. (5 points) What is $P(L_1 = \text{`M'})$? Recall that there are 26 letters in the alphabet.

b. (10 points) Assume that our belief in each letter is **independent** of the other letters. How many times more likely is it that the letters are `[C, A, T]`, compared to `[M, A, N]` under this assumption? You do not need to incorporate a prior belief over combinations of letters.

[1]A natural ASL signer can spell at an amazing rate of around 6 letters a second!

5 Binary Spoof [25 points]

Binary Spoof is a game played among the CS109 TAs. Here are the rules:

1. There are exactly six players.
2. Before the game is played, Chris discretely places a stone in each player's hand, independently, with probability 0.6. Players do not know what was chosen for other players.
3. One at a time, players guess aloud the total sum of stones across all six hands[2].
4. The stones are revealed. If a player guessed correctly, they win.

a. (10 points) Let us consider the first player's perspective. They know that they do **not** have a stone in their hand. What is the probability that the sum of stones in the other five player's hands is exactly 3?

[2]Guesses can not be repeated. This rule doesn't impact the question. It does make the game more fun.

b. (15 points) The first player guesses: “three!”. We assume that the probability that the first player says 3 is equal to the probability, from the first player’s perspective, that the sum is truly 3. Player two wants to infer whether player one has a stone in their hand. What is the probability that the first player has a stone in their hand given that they predicted “three”?

6 PSetApp Answer Checker [25 points]

When you hit the "Test Agent" button on a coding problem in the PSet App, the server needs to decide if your code is correct. This can be tricky when there is randomness involved. For example, recall the Counting Cards problem on PSet 1. In Counting Cards, you wrote code which could keep track of how many high cards were left in a deck. When "Test Agent" was pressed the app would run 1,000 games and if your algorithm earned enough money in the 1,000 games the app would mark your code as correct.

Because of the nature of the game, we know that the money won by the correct algorithm is distributed as a normal with mean $\mu =$ \$16,500 and variance $\sigma^2 =$\$800^2.

a. (10 points) A student wrote the correct algorithm. What is the probability that the student wins \$15,000 or above when they hit Test Agent?

b. (15 points) A particular student **either** has the correct answer **or** they have an off-by-one-error (OBOE). Based on historical frequencies, your prior probability that they have an OBOE is 0.2. If a student has an OBOE, their score will be normally distributed with mean \$16,000 and variance \$$750^2$. You run their code once and they get score of \$15,900. What is your updated belief that they have an OBOE?

7 Beyond the Binomial [20 points]

Note: this problem is a challenge problem. It is worth fewer points per minute than the others.

Write a pseudo-code function `probability_sum(p_list, k)` that calculates the exact probability of **exactly** `k` successes in `n` **independent** events if each event `i` has a **different** probability of success, `p_list[i]`. You will be passed in the success probabilities as a list called `p_list` which is of length `n`.

Here is an example use of your function. The UK is competing in 5 winter Olympic events. Their probabilities of winning a medal in each event are `[0.4, 0.6, 0.9, 0.2, 0.1]` respectively. Their chance of winning exactly three medals is: `probability_sum(p_list = [0.4, 0.6, 0.9, 0.2, 0.1], k = 3)`.

Feel free to use a function `unique_permutations` that returns all distinct permutations of a list of objects

```
>>> unique_permutations([0, 0, 1])
[[0, 0, 1], [0, 1, 0], [1, 0, 0]]
```

To make your code easier to write, you may assume that `len(p_list)` is always 5 and that `k` is always 3. If your code produces an approximate answer, you could get up to half the points on this problem.

Hint: recall the many coin flips story we used to derive the Binomial PMF. How does this change if we don't assume each probability is the same?

```
def probability_sum(p_list, k):
    n = len(p_list)
    assert(n == 5 and k == 3) # you can assume this
```