Chris Piech
CS109

Win 2025

CS109 Final Exam

This is a closed calculator/computer/phone/smart-watch/smart-toothbrush exam. You are, however, allowed to use notes in the exam. You have 3 hours (180 minutes) to take the exam. The exam is 180 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) or Φ^{-1}. 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. Short Answer [23 points]

a. (5 points) Let $X \sim \text{Exp}(\lambda = 0.2)$. What is the probability that X is between 5 and 10?

We can solve this problem using either the CDF or the integral of the PDF for the Exponential.

CDF:

$$\begin{aligned} P(5 < X < 10) &= P(X < 10) - P(X < 5) \\ &= 1 - e^{-0.2\cdot 10} - \left(1 - e^{-0.2\cdot 5}\right) \\ &= 1 - e^{-2} - 1 + e^{-1} \\ &= e^{-1} - e^{-2} \end{aligned}$$

Integral of PDF:

$$\begin{aligned} P(5 < X < 10) &= \int_5^{10} \lambda e^{-\lambda x} dx \\ &= \left[-0.2 \cdot 5 \cdot e^{-0.2x}\right]_{x=5}^{10} \\ &= -e^{-0.2\cdot 10} + e^{-0.2\cdot 5} \\ &= e^{-1} - e^{-2} \end{aligned}$$

b. (6 points) When it rains in the morning, your neighbor's dog has a 70% chance of being mucky. On mornings without rain, their dog is mucky only 20% of the time. Each day has a 10% chance of rain in the morning. Today, their dog was mucky. What is the probability it was raining in the morning?

Let M be the event the dog is mucky and R being the event it's raining. Using Bayes' Theorem:

$$\begin{aligned} P(R|M) &= \frac{P(M|R)P(R)}{P(M)} \\ &= \frac{P(M|R)P(R)}{P(M|R)P(R) + P(M|R^C)P(R^C)} \\ &= \frac{0.7 \cdot 0.1}{(0.7 \cdot 0.1) + (0.2 \cdot 0.9)} \\ &= \frac{0.07}{0.25} \\ &= 0.25 \end{aligned}$$

c. (6 points) A standard deck of cards has 12 royals and 40 non-royal cards. If the top card is a royal you will have a 80% chance of winning. If the top card is a non-royal card, you have a 40% chance of winning. What is your probability of winning the game?

Let W be the event you win the game. Let R be the event that the top card is a royal card Using The Law of Total Probability:

$$\begin{aligned} P(W) &= P(W|R)P(R) + P(W|R^C)P(R^C) \\ &= 0.8 \cdot \frac{12}{52} + 0.4 \cdot \frac{52-12}{52} \end{aligned}$$

d. (6 points) For the CS109 Midterm, there are 300 students who are sent to one of three rooms based on their last name. Each student, independently, has a 30% chance of going to room 1, a 20% chance of going to room 2 and a 50% chance of going to room 3. What is the chance that exactly 90 students go to room 1, 60 students go to room 2, and 150 go to room 3?

Let X_1 be the number of students going to room 1, X_2 the number of students going to room 2, and X_3 the number of students going to room 3. The joint distribution between these RVs is a Multinomial, analogous to rolling a dice with three sides for each student to determine their room. We can use the Multinomial PMF:

$$\begin{aligned} P(X_1 = 90, X_2 = 60, X_3 = 150) &= \binom{300}{90, 60, 150} 0.3^{90} 0.2^{60} 0.5^{150} \\ &= \frac{300!}{90!60!150!} 0.3^{90} 0.2^{60} 0.5^{150} \end{aligned}$$

2. In-Vitro Fertilization [13 points]

A goal of an In-Vitro Fertilization (IVF) cycle is to produce as many healthy embryos as possible. At the start of a cycle, a patient has a certain number of growing antran follicles. Each antran follicle must successfully pass through three stages to become a healthy embryo:

- Maturity: An antran follicle has a **0.9** chance of reaching full maturity.
- Retrieval: A fully mature follicle has a **0.8** chance of successfully being retrieved.
- Fertilization: A retrieved follicle has a **0.7** probability of being successfully fertilized into a healthy embryo.

Assume each follicle's journey is independent.

a. (6 points) What is the probability that a single antran follicle will become a healthy embryo?

Using the chain rule, we can multiply the probabilities of each independent stage to get the overall probabability:

$$P(\text{healthy}) = P(\text{healthy}|\text{retrieved})P(\text{retrieved}|\text{mature})P(\text{mature})$$

$$= 0.9 \cdot 0.8 \cdot 0.7$$

b. (7 points) A particular patient has 14 antran follicles at the start of a cycle. Let your answer to part (a) be p_a. What is the probability that the patient will have k healthy embryos (for $k = 0, 1, \ldots, 14$)?

Let X be the number of healthy embryos. We can use the Binomial distribution with $n = 14$ and success probability p_a.

$$P(X = k) = \binom{14}{k} p_a^k (1 - p_a)^{14-k}$$

3. Bag of Coins [20 points]

A bag has 10 coins that *appear* identical. However, some coins are of type A, some coins are of type B.

- A coin of type A has probability **0.3** of landing Heads.
- A coin of type B has probability **0.6** of landing Heads.

All coins are independent of one another. All 10 coins are flipped, and we count the number of heads.

a. (7 points) Imagine 7 of the coins were type A, and 3 of the coins were type B. What is the probability that we get exactly two heads from the type A coins and exactly two heads from the type B coins?

Solution: Let H_A be the number of heads from A-coins, and let H_B be the number of heads from B-coins. From the problem statement:

$$H_A \sim \text{Bin}(n = 7, p = 0.3)$$

$$H_B \sim \text{Bin}(n = 3, p = 0.6)$$

We are asking the probability of the "and" of the two events $H_A = 2$ and $H_B = 2$. Since these two events are independent, we can thus compute:

$$P(H_A = 2, H_B = 2) = P(H_A = 2)P(H_B = 2) = \binom{7}{2}(0.3)^2(0.7)^5 \cdot \binom{3}{2}(0.6)^2(0.4)^1.$$

b. (7 points) Under the same assumption (7 coins of type A, 3 of type B), what is the probability of observing 4 heads total?

Solution: As in Part (a), we have

$$H_A \sim \text{Bin}(n = 7, p = 0.3)$$
$$H_B \sim \text{Bin}(n = 3, p = 0.6)$$

We need to sum over all ways to get a total of 4 heads from 7 A-coins and 3 B-coins:

$$P(4 \text{ heads total}) = P(H_A + H_B = 4)$$

There are several different ways to write the latter probability out, all of which use some form of convolution:

- Method 1: Explicit listing: 4 heads total can happen with:
  - 4 A-coin heads and 0 B-coin heads: $P(H_A = 4, H_B = 0) = \binom{7}{4}(0.3)^4(0.7)^3 \cdot \binom{3}{0}(0.6)^0(0.4)^3$
  - 3 A-coin heads and 1 B-coin heads: $P(H_A = 3, H_B = 1) = \binom{7}{3}(0.3)^3(0.7)^4 \cdot \binom{3}{1}(0.6)^1(0.4)^2$
  - 2 A-coin heads and 2 B-coin heads: $P(H_A = 2, H_B = 2) = \binom{7}{2}(0.3)^2(0.7)^5 \cdot \binom{3}{2}(0.6)^2(0.4)^1$
  - 1 A-coin heads and 3 B-coin heads: $P(H_A = 1, H_B = 3) = \binom{7}{1}(0.3)^1(0.7)^6 \cdot \binom{3}{3}(0.6)^3(0.4)^0$

  We then sum these four cases. (Note there is no case corresponding to $H_A = 0, H_B = 4$ due to there only being 3 B-coins.)
- Method 2: Sum over the number of A-coin heads:

$$\begin{aligned} P(H_A + H_B = 4) &= \sum_{i=0}^{4} P(H_A = i) \cdot P(H_B = 4 - i) \\ &= \sum_{i=0}^{4} \binom{7}{i}(0.3)^i(0.7)^{7-i} \cdot \binom{3}{4-i}(0.6)^{4-i}(0.4)^{3-(4-i)} \end{aligned}$$

  Note that the same summation from $i = 1$ to 4 is also correct: When $i = 0$, the second combination is $\binom{3}{4} = 0$. This accounts for the fact that there are only 3 B-coins, so the $i = 0$ case (0 A-coin heads and 4 B-coin heads) is impossible.
- Method 3: Sum over the number of B-coin heads:

$$\begin{aligned} p(H_A + H_B = 4) &= \sum_{i=0}^{3} P(H_A = 4 - i) \cdot P(H_B = i) \\ &= \sum_{i=0}^{3} \binom{7}{4-i}(0.3)^{4-i}(0.7)^{7-(4-i)} \cdot \binom{3}{i}(0.6)^i(0.4)^{3-i} \end{aligned}$$

c. (6 points) Let X be a discrete random variable for the number of type A coins in the bag (from 0 to 10). Your prior belief is $P(X = x) = \frac{1}{11}$. Let H be the total number of heads.

Show that $P(X = x|H = 4) = k \cdot P(H = 4|X = x)$ where k is a constant. You do not need to solve for k.

Solution: Since we are provided a prior $P(X = x)$, we can use Bayes Theorem to rewrite the posterior:

$$P(X = x|H = 4) = \frac{P(H = 4|X = x)P(X = x)}{P(H = 4)}$$

We know that that H (the total number of heads across all coin types) depends on the number of type A coins. In part b, we found the probability of getting 4 heads when there are exactly 7 type A coins. However, in part c, we need to consider the probability of getting 4 heads, across all cases of possible number of type A coins. All cases are mutually exclusive so we can apply the law of total probability to find the probability of $P(H = 4)$, summing over all possible numbers of type A coins:

$$P(H = 4) = \sum_{i=0}^{10} P(H = 4|X = i)P(X = i)$$

Plugging in our expression for $P(H = 4)$ and $\frac{1}{11}$ for $P(X = x)$, we see:

$$P(X = x|H = 4) = \frac{P(H = 4|X = x) \cdot \frac{1}{11}}{\sum_{i=0}^{10} P(H = 4|X = i) \cdot \frac{1}{11}}$$

We can isolate all constant terms in the expression.

$$k = \frac{\frac{1}{11}}{\sum_{i=0}^{10} P(H = 4|X = i)\frac{1}{11}}$$

We consider these terms constant because they will always be the same value across all values of x. Thus, we see that

$$\begin{aligned} P(X = x|H = 4) &= \frac{P(H = 4|X = x) \cdot \frac{1}{11}}{\sum_{i=0}^{10} P(H = 4|X = i) \cdot \frac{1}{11}} \\ &= k \cdot P(H = 4|X = x) \end{aligned}$$

4. Microcontroller Precision Error [15 points]

A low-powered microcontroller can only represent numbers to 3 decimal places.

Let X_i be an i.i.d. sample from Uniform$(0, 1)$ with infinite precision. Let Y_i be the representation of X_i on the microcontroller. Y_i is equal to X_i truncated to 3 decimal places. Here are three examples:

X_i	Y_i
0.12340809...	0.123
0.28374110...	0.283
0.55555555...	0.555

a. (5 points) What is the distribution of $(X_i - Y_i)$? In other words, what is the distribution for the error when a single uniformly sampled value is represented in the microcontroller?

Solution: For each $X_i \in [0, 1]$, the rounding error is uniformly distributed between 0 and 0.001, so:

$$(X_i - Y_i) \sim \text{Unif}(0, 0.001)$$

b. (10 points) We sample 1000 values, where X_i is the precise value and Y_i is the truncated value of X_i:

$$X = \sum_{i=1}^{1000} X_i \qquad Y = \sum_{i=1}^{1000} Y_i$$

What is the probability that the difference between X and Y is more than 0.51?

Solution: Apply Central Limit Theorem (CLT) since we are observing the distribution of the sum of 1000 IID samples. First, calculate the mean error and variance per sample (from the uniform distribution found in part a):

$$E[X_i - Y_i] = \frac{1}{2}(0.001) = 0.0005$$

$$Var(X_i - Y_i) = \frac{1}{12}(0.001)^2$$

We cannot apply a linear transformation between individual distributions for X and Y since Y_i is discrete and we don't have have a distribution for it. Instead, let $X - Y = \sum_{i=1}^{1000}(X_i - Y_i)$ and apply CLT to approximate $(X - Y)$ as a normal distribution:

$$(X - Y) \sim \mathcal{N}(\mu = 1000 * 0.0005, \sigma^2 = 1000 * \frac{1}{12}(0.001)^2)$$

Finally, solve for the final probability with the normal CDF:

$$P(X - Y \geq 0.51) = 1 - P(X - Y \leq 0.51) = 1 - \phi\left(\frac{0.51 - 0.5}{\sqrt{\frac{1000}{12}(0.001)^2}}\right) = 1 - \phi\left(\frac{0.01}{\sqrt{\frac{0.001}{12}}}\right)$$

Note: We accept both a one-tailed or two-tailed solution. However, since the event of interest is not symmetric around the mean, we cannot simply multiply the above probability by 2 for the two-tailed solution. Instead, we need to separately calculate $P(X - Y \leq -0.51)$, as well. The two tailed solution is:

$$P(|X - Y| \geq 0.51) = (1 - P(X - Y < 0.51)) + P(X - Y < -0.51)$$

5. Exponential Backoff [18 points]

Two computers simultaneously attempt to send messages over a shared wire channel at second zero, causing a first collision. The messages were not delivered and they will have to re-attempt.

Each computer is using an algorithm called Exponential Backoff. If they have had n collisions, they independently wait for a random integer number of seconds uniformly chosen from the set $\{1, \ldots 2^n\}$ before attempting to retransmit over the shared wire:

Number of collisions, n	Possible wait times (seconds)
1	$\{1, 2\}$
2	$\{1, 2, 3, 4\}$
3	$\{1, 2, 3, 4, 5, 6, 7, 8\}$
4	$\{1, 2, 3, 4, 5, 6, 7, 8, \ldots 16\}$
$\ldots$	$\ldots$

If the computers choose the same number of seconds to wait, they will have another collision. This process continues until there is no collision.

For example, after their initial collision, $n = 1$, so they will both randomly choose to wait 1 or 2 seconds. If they both choose 1, or if they both choose 2, there will be a second collision. Otherwise the messages will successfully send.

a. (8 points) What is the probability of having **exactly 3 collisions total**? (Equivalently, two more collisions after the initial one at second 0, followed by a successful send.)

Solution: This is similar to a Geometric random variable (trials until success), but the probability p changes when n changes! For instance, the probability of a collision when $n = 1$ is $P(\text{both choose 1}) + P(\text{both choose 2}) = \frac{1}{2^2} + \frac{1}{2^2} = \frac{1}{2}$. In general:

$$\begin{aligned} P(\text{collision} \mid n) &= 2^n \cdot \frac{1}{2^n}\frac{1}{2^n} \\ &= \frac{1}{2^n} \end{aligned}$$

For exactly three collisions total, we multiply the probabilities at $n = 1$ and $n = 2$, then the *complement* at $n = 3$ (all these events are independent):

$$\begin{aligned} P(\text{3 collisions, then success}) &= P(\text{collision \#2} \mid n = 1) \cdot P(\text{collision \#3} \mid n = 2) \cdot P(\text{no collision} \mid n = 3) \\ &= \frac{1}{2^1} \cdot \frac{1}{2^2} \cdot \left(1 - \frac{1}{2^3}\right) \\ &= \boxed{\frac{7}{64}} \end{aligned}$$

(Note that we don't include $P(\text{collision \#1} \mid n = 0)$ since the first collision is given at second 0.)

b. (10 points) The computers have exactly 3 collisions and then successfully send on the next attempt. What is the expected amount of time it took the two computers to successfully send both their messages? *Note: The only time you have to consider is the time that the computers are waiting during exponential backoff, not the time it takes a message to pass over the wire, nor the time it takes to recognize a collision.*

Solution: Let T be the total time it takes to send both messages, with T_i as the time of the attempt when $n = i$. We can use the linearity of expectation to divide $\mathbb{E}[T]$:

$$\begin{aligned}\mathbb{E}[T] &= \mathbb{E}[T_0 + T_1 + T_2 + T_3] \\ &= \mathbb{E}[T_0] + \mathbb{E}[T_1] + \mathbb{E}[T_2] + \mathbb{E}[T_3]\end{aligned}$$

$\mathbb{E}[T_0] = 0$ since collision #1 happens at second 0. Since the next attempt failed, either both programs waited 1 second or 2 seconds, with equal chance: $\mathbb{E}[T_1] = 1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{2} = 1.5$. The same applies for T_2: $\mathbb{E}[T_1] = \sum_{i=1}^{4} i \cdot \frac{1}{4} = 2.5$.

$\mathbb{E}[T_3]$ is trickier since the programs wait different times, so the PMF of T_3 *not* equally uniform. Let S_1, S_2 be random variables for the wait times chosen by computers 1 and 2 respectively at $n = 3$. The time taken to send both messages will be $T_3 = \max(S_1, S_2)$ given $S_1 \neq S_2$, e.g. $T_3 = 2$ corresponds to $S_1 = 2, S_2 = 1$ and $S_1 = 1, S_2 = 2$. We can compute $P(T_3 = t)$ by counting:

$$\begin{aligned}P(T_3 = 1) &= 0 \\ P(T_3 = 2) &= \frac{2}{56} \\ P(T_3 = 3) &= \frac{4}{56} \\ &\cdots \\ P(T_3 = t) &= \frac{2(t-1)}{56}\end{aligned}$$

Note that the size of the sample space is 56 (64 joint possibilities for $(S_1, S_2$ minus 8 cases where $S_1 = S_2$). From here, we use the definition of expectation:

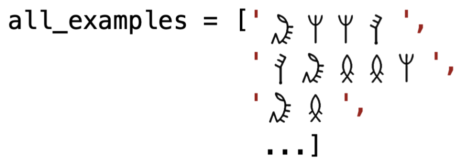
$$\begin{aligned}\mathbb{E}[T_3] &= \sum_{t=1}^{8} t \cdot P(T_3 = t) \\ &= \sum_{t=1}^{8} t \cdot \frac{2(t-1)}{56} \\ &= 6\end{aligned}$$

Plugging everything back in, we get $\mathbb{E}[T] = 0 + 1.5 + 2.5 + 6 = \boxed{10}$.

6. Indus Valley Script [19 points]

The Indus civilization - one of the world's earliest urban societies - emerged 5,300 years ago. The script they used is one of the last remaining undeciphered alphabets.

We are using entropy analysis to show that the script represents a true linguistic system. Let **`all_examples`** be the list of all recorded examples of the script, where each item in the list is one example string:



```
all_examples = ['    ',
                '     ',
                '  ',
                ...]
```

a. (5 points) An AI chatbot answered a question on how to compute the entropy of the first character in a string from the Indus Valley Script. Find any mistakes.

Chatbot response of the entropy of the first character:

Let $P(X = x)$ be the probability that the first character in a script string is x. Let n be the total number of strings in **`all_examples`**, and let n_x be the number of times that character x is the first character in the example.

$$\text{Entropy}(X) = \sum_x \log_2 \frac{1}{P(X = x)}$$

Next, we apply a log transformation rule to simplify the expression:

$$\text{Entropy}(X) = -\sum_x \log_2 P(X = x) \quad \text{(Log rule)}$$

We then substitute $P(X = x) = \frac{n_x}{n}$ to reflect the frequency across all examples:

$$\text{Entropy}(X) = -\sum_x \log_2 \frac{n_x}{n} \quad (P(X = x) = n_x/n)$$

Finally, we refactor out the log:

$$\text{Entropy}(X) = -\log_2 \sum_x \frac{n_x}{n} \quad \text{(Sum of logs)}$$

There are two errors. First, the entropy formula is missing the multiplication of the log by the probability of x. The correct formula is $H(X) = \sum_x (\log_2 \frac{1}{P(X=x)}) P(X = x)$. Second, the log is pulled out of the sum incorrectly on the last line.

b. (7 points) Write a function **`pr_y_given_x(all_examples, x, y)`** which returns the conditional probability that the second character in the string is **`y`**, given that the first character in the string is **`x`**.

We count occurrences where the first character is x, and the occurrences where the first two characters are xy. Then, the conditional probability is the ratio

$$P(Y = y \mid X = x) = \frac{P(X = x, Y = y)}{P(X = x)} = \frac{n_{xy}/n}{n_y/n} = \frac{n_{xy}}{n_y}.$$

```
def pr_y_given_x(all_examples, x, y):
    count_x = 0
    count_x_and_y = 0
    for example in all_examples:
        if example[0] == x:
            count_x += 1
            if example[1] == y:
                count_x_and_y += 1

    return count_x_and_y / count_x
```

c. (7 points) On average over all examples of the script, how much does the entropy of the second character decrease once you have observed the first character? Let $P(Y = y|X = x)$ be your answer to part b. Let $P(X = x)$ be the probability that the first character is x. Your answer can be a math expression or code.

Solution (math): The entropy of the second character without observing the first character is just calculated with the entropy formula

$$H(Y) = -\sum_y P(Y = y) \log_2 P(Y = y)$$

where we can compute the PMF of Y using LOTP or using the notation from part a:

$$P(Y = y) = \sum_x P(Y = y|X = x)P(X = x) = \frac{n_y}{n}.$$

After observing the first character is $X = x$, the entropy of the second character is

$$H(Y \mid X = x) = -\sum_y P(Y = y \mid X = x) \log_2 P(Y = y \mid X = x).$$

Thus, the *expected* entropy of the second character after observing the first character is the *weighted average* over all first character $X = x$:

$$\mathbb{E}_X[H(Y \mid X)] = -\sum_x P(X = x) \sum_y P(Y = y \mid X = x) \log_2 P(Y = y \mid X = x).$$

Therefore, the difference in entropy is:

$$\begin{aligned} H(Y) - \mathbb{E}_X[H(Y \mid X)] = &-\sum_y P(Y = y) \log_2 P(Y = y) \\ &+ \sum_x P(X = x) \sum_y P(Y = y \mid X = x) \log_2 P(Y = y \mid X = x). \end{aligned}$$

Solution (code):

```python
def entropy(pmf):
    H = 0
    for x in pmf:
        H -= pmf[x] * log2(pmf[x])
    return H

def solution():
    p_y = {}
    for x in p_x:
        for y in p_y_given_x[x]:
            p_y[y] += p_y_given_x[x][y] * p_x[x]

    H_y = entropy(p_y)
    E_H_y_given_x = 0
    for x in p_x:
        H_y_given_x = entropy(p_y_given_x[x])
        E_H_y_given_x += H_y_given_x * p_x[x]

    return H_y - E_H_y_given_x

# If you iterate over the set of examples, we get P(x) and P(x,y)
# for free, so we sum the surprise instead of entropy inside the loops.
def alternate_solution():
    p_y = {}
    for x in p_x:
        for y in p_y_given_x[x]:
            p_y[y] += p_y_given_x[x][y] * p_x[x]

    H_y = 0
    E_H_y_given_x = 0
    for example in all_examples:
        x = example[0]
        y = example[1]
        H_y -= log2(p_y[y])
        E_H_y_given_x -= log2(p_y_given_x[x][y])

    H_y /= len(all_examples)
    E_H_y_given_x /= len(all_examples)

    return H_y - E_H_y_given_x
```

7. Estimating Course Size [19 points]

In order to hire the correct number of TAs, Stanford needs to estimate final enrollment in CS109 two weeks before the start of the quarter. Two weeks before the start of the quarter 300 students are enrolled.

For the last 10 offerings of CS109 Stanford has recorded the ratio:

$$r_i = \frac{\text{(final enrollment for offering } i\text{)}}{\text{(enrollment two weeks before start for offering } i\text{)}}$$

which you can access as a list of values $[r_1, r_2, \ldots, r_{10}]$. Assume that each r_i is an i.i.d. sample from the true ratio random variable R. The number of students who end up taking the class this quarter, T, will be $T = 300 \cdot R$. From historical analysis we know that both R and T are normally distributed.

a. (5 points) Estimate $E[T]$, the expected class size. Provide your answer as a math expression.

$$\mathbb{E}[T] = 300 \cdot \mathbb{E}[R] = 300 \cdot \frac{\sum_{i=1}^{10} r_i}{10}$$

b. (6 points) Estimate $\text{Var}(T)$. Provide your answer as a math expression.

$$\text{Var}(T) = 300^2 \cdot \text{Var}(R)$$
$$= 300^2 \cdot \frac{1}{10-1} \sum_{i=1}^{10} \left(r_i - \frac{\sum_{i=1}^{10} r_i}{10} \right)^2$$

c. (8 points) Use *bootstrapping* to estimate the *standard deviation* of your estimate of $\text{Var}(T)$ from part (b) (i.e., the sampling variability due to having only 10 historical observations of R).

```
r_list = [r_1, r_2, ..., r_10]
var_list = []
for i in range(10000):
-> sub_sample = sample(r_list, replace = True, size = 10)
-> variance = 300**2 * sample_var(sub_sample) # Calculation from b
-> var_list.append(variance)
return std_dev(var_list)
```

8. Bayesian RNA Sequencing [18 points]

Let N_A and N_B be the count of type-A RNA and type-B RNA in a person's body. Both are Poisson distributed:

$$N_A \sim \text{Poi}(12 \cdot X) \qquad N_B \sim \text{Poi}(12 \cdot (1 - X))$$

N_A and N_B have rates that come from an unknown continuous variable X. X takes on values between 0 and 1 and is an indicator of a person's overall health. N_A and N_B are **independent**, conditioned on knowing X.

a. (5 points) For part (a) only, assume $X = 1/4$. What is $P(N_A > 10)$?

$N_A \sim \text{Poi}(3)$, so $P(N_A > 10) = 1 - \sum_{k=0}^{10} \frac{3^k e^{-3}}{k!}$.

b. (10 points) Our prior belief is that $X \sim \text{Beta}(a = 2, b = 2)$. An experiment observes $N_A = 12$ and $N_B = 16$. Based on these observations, what is your updated distribution for X? In other words, what is $f(X = x|N_A = 12, N_B = 16)$? You can leave your answer in terms of a normalization constant.

Posterior is proportional to prior × likelihood. The likelihood is Poisson(Xk) for N_A and Poisson$((1 - X)k)$ for N_B. Combined with Beta$(2, 2)$ prior yields a known Beta update if we treat $N_A + N_B$ as total counts. The prior is updated based on the observations of likelihood N_A and N_B. Therefore, our posterior beta is:

$$\begin{aligned} f(X = x|N_A = 12, N_B = 16) &= \frac{P(N_A = 12, N_B = 16|X = x)f(X = x)}{P(N_A = 12, N_B = 16)} \\ &= \frac{\frac{(12x)^{N_A} \cdot e^{-12x}}{N_A!} \cdot \frac{(12(1-x))^{N_B} \cdot e^{-12(1-x)}}{N_B!} \cdot B \cdot x^{a-1}(1 - x)^{b-1}}{k} \\ &= K \cdot x^{a+N_A-1}(1 - x)^{b+N_b-1} \end{aligned}$$

This simplifies to the beta Beta$(2 + 12, 2 + 16) = \text{Beta}(14, 18)$.

c. (3 points) What is the variance of your updated belief distribution for X, found in part (b)?

Use the variance formula for the resulting Beta distribution, $\text{Var}(X) = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} = \frac{14 \cdot 18}{32^2 \cdot 33}$.

9. MLE of a Truncated Normal Distribution [18 points]

A normal distribution can allow negative values. A truncated normal is a version of the normal distribution that only supports positive values of x. If $X \sim \text{TruncatedNormal}(\mu)$ then it has the following PDF:

$$f(X = x) = \frac{\frac{1}{\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2}}}{\Phi(\mu)} \quad \text{for } x > 0.$$

Where Φ is the Cumulative Distribution Function for the Standard Normal.

You have n i.i.d. samples from a TruncatedNormal: [7.923, 12.388, 9.984, 12.019, 8.725, . . .]. Let x_i be the ith value. Explain how you would use MLE to estimate parameter μ and provide any necessary derivatives.

We first have to formulate the likelihood and log likelihood functions as follows:

$$L(\mu) = \prod_{i=1}^{n} \frac{\frac{1}{\sqrt{2\pi}} e^{-\frac{(x_i-\mu)^2}{2}}}{\Phi(\mu)}$$

$$LL(\mu) = \sum_{i=1}^{n} \log(\frac{1}{\sqrt{2\pi}}) - \frac{(x_i - \mu)^2}{2} - \log(\Phi(\mu))$$

We then take the derivative of the log likelihood w.r.t. μ. We use ϕ as the PDF of the standard normal. Notice that the derivative of the CDF is the PDF, so $\frac{\partial \Phi(\mu)}{\partial \mu} = \phi(\mu)$.

$$\begin{aligned} \frac{\partial LL(\mu)}{\partial \mu} &= \sum_{i=1}^{n} (x_i - \mu) - \frac{\frac{\partial \Phi(\mu)}{\partial \mu}}{\Phi(\mu)} \\ &= \sum_{i=1}^{n} (x_i - \mu) - \frac{\frac{1}{\sqrt{2\pi}} \cdot e^{\frac{-\mu^2}{2}}}{\Phi(\mu)} \end{aligned}$$

There is no closed form solution when setting the derivative equal to 0, so we would continue by using gradient ascent to get to an optimal $\hat{\mu}$ parameter.

10. Logistic Regression Conceptual [17 points]

a. (4 points) In training a logistic regression model, why do we typically optimize the *log-likelihood* directly, rather than the likelihood?

The likelihood function is typically a large product, once we assume our datapoints are IID. Training a logistic regression model requires finding the derivative of this as part of MLE, but taking a derivative over a big product is hard.

Because the log function is monotonic, if we take the log of the likelihood function, the argmax will not be changed. The log of the product is the sum of the logs, so the log of the likelihood will become a large sum instead, which will be easier to differentiate as part of gradient ascent.

b. (4 points) Why is logistic regression a more principled model than linear regression for binary classification?

Logistic regression outputs values in the range $[0, 1]$, since the sigmoid function ”squashes” all inputs into this range. We can interpret these outputs as probabilities, specifically $P(Y = 1)$, which is meaningful in binary classification because it expresses a level of confidence in our belief about a binary event.

Linear regression, on the other hand, can produce values in a continuous range outside $[0, 1]$, so it is less well-suited for classification tasks. While it is possible to train a linear regression model where $y_i \in \{0, 1\}$ for all datapoints, this model wouldn't be able to express $P(Y = 1)$ and would also output numbers way different from 0 or 1.

Another way of explaining this is that logistic regression tries to learn a decision boundary that divides a space in half, putting only datapoints from one class on one side of the line. Linear regression, on the other hand, tries to learn a line of best fit for datapoints. While both are linear models, one makes more sense for classification into discrete classes.

c. (4 points) Consider a logistic regression that predicts Y from features $X_1, \ldots, X_m$. If Y and a particular feature X_i are independent, what would you expect for the learned coefficient θ_i, and why?

If X_i is independent of Y, we expect $\theta_i \approx 0$.

In the pre-sigmoid computation of the dot product for logistic regression, $\theta_0 X_0 + \theta_1 X_1 + \cdots + \theta_m X_m$, we would like to set the θ value such that X_i's value will not have any impact on this computation. This is because, intuitively, the prediction of Y should not depend on X_i if they are independent. Setting $\theta_i = 0$ accomplishes this.

Practically, when the model is trained on a finite dataset, the value may not be exactly zero, but it should be close.

d. (5 points) Below is pseudocode for training for a logistic regression model:

```
1   thetas[j] = 0 for all j
2   repeat many times:
        gradients[j] = 0 for all j
3       for each training example (x, y):
            for each parameter j:
4               gradients[j] += x[j]*(y - sigmoid(dot_prod(theta, x))
5       thetas[j] += step_size * gradients[j] for all j
```

For each line of pseudocode 1-5, explain briefly, as if teaching, what the line of code does, and why it is necessary.

Line 1: First, we initialize each of our thetas (weights or parameters of the model) to zero. We start the thetas at zero because we need to start *somewhere* to do gradient ascent!

Line 2: Then, we start the training loop – we need to repeat the theta update process many times, since we will only improve the thetas slightly (taking a small step towards the optimal value) each time.

Line 3: We iterate over each training example to calculate the gradient over all data points.

Line 4: We then calculate the the gradient of the log-likelihood based on a single datapoint and add it to a building sum across all datapoints.

Line 5: Lastly, we update the thetas by moving a little bit (according to our step size) in the direction of the gradient!

That's all folks! Thank you for the lovely quarter. You were a wonderful class. IVF numbers are mostly real, but I removed a few steps. Exponential Backoff is the true algorithm used to handle congestion for everything from radio transmissions to the internet. In 2025 the chief minister of southern India's Tamil Nadu state announced a $1m prize for anyone who can crack the Indus Valley Script. Estimating Course Sizes is (very close to) the real algorithm the CS department uses. Bayesian RNA quantification is a twist on a widely used algorithm called cufflinks for estimating RNA isoform proportions. Truncated Normals are super useful! They have a variance parameter, a min value and a max value which I removed for exam simplicity.