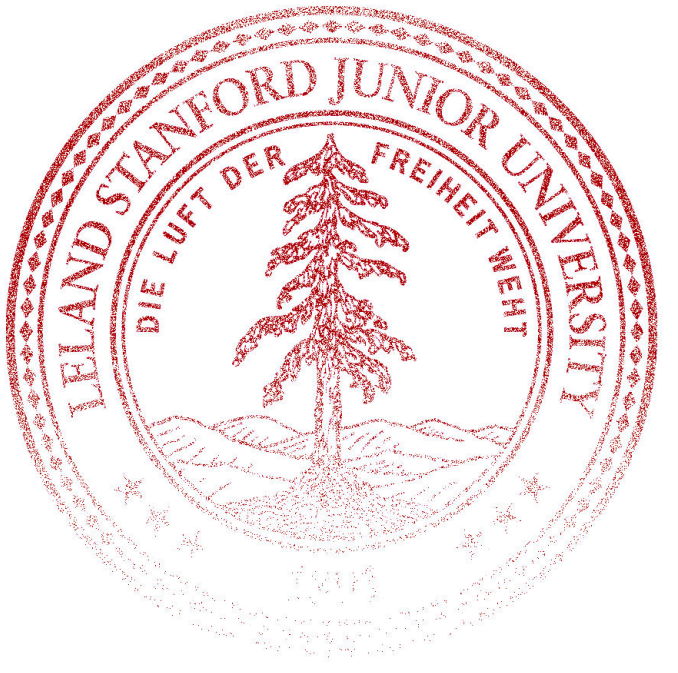
Chris Piech Fall 2022
CS109

CS109 Final Exam

This is a closed calculator/computer exam. You are, however, allowed to use notes in 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} (the inverse CDF). For example $\Phi\left(\frac{3}{4}\right)$ is an acceptable final answer. Recall that the exam is going to be "curved" according to the difficulty of the questions and as such hard questions will not translate to lower grades.



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 (22 points)

Answer each of the following questions. You must give a brief justification for your answer.

a. (5 points) What is the probability that a randomly chosen three-digit integer (from 0 to 999 inclusive) will be divisible by 5? Note that 0 is divisible by 5.

There are 1000 total numbers between 0 and 999, so 1000 is our sample space. The event space is 20×10 (where 20 is the number of digits divisible by 5 from 0 to 99 and there are 10 sets of 100 between 0 and 999. Thus our probability is $\frac{200}{1000} = 0.2$

3cm

b. (9 points) Suppose X is a random variable that is normally distributed with a mean of 100 and a standard deviation of 15. What is the probability that a random sample of size 10 from this distribution will have a mean between 95 and 105?

We know that $X \sim \mathcal{N}(\mu = 100, \sigma = 15)$. The sample mean is given by:

$$\bar{X} \sim \frac{\sum_{i=1}^{10} X_i}{n} = \mathcal{N}(\mu = 100, \sigma = \frac{15}{\sqrt{10}})$$

Normalizing we get that:

$$P(\frac{95 - 100}{4.74} < \bar{X} < \frac{105 - 100}{4.74})$$
$$P(-1.05 < Z < 1.05)$$
$$P(1.05) - P(-1.05)$$
$$P(1.05) - 1 + P(1.05)$$

c. (8 points) Each child in a daycare has a 0.2 probability of having disease A and has an independent 0.4 probability of having disease B. A child is sick if they have either disease A or disease B. If there are 10 children in a daycare what is the probability that 2 or more are sick?

Let A and B be the events that a child has disease A and disease B, respectively. A child is healthy if they have neither disease A nor disease B. So,

$$\begin{aligned}
P(\text{sick}) &= 1 - P(\text{healthy}) \\
&= 1 - P(A^C, B^C) \\
&= 1 - P(A^C)P(B^C) && (A \perp B) \\
&= 1 - (1 - P(A))(1 - P(B)) \\
&= 1 - (0.8)(0.6) \\
&= 1 - (0.48) \\
&= 0.52
\end{aligned}$$

Let Y be the number of children that are sick. We can write this as $Y \sim \text{Bin}(10, P(\text{sick}))$. Thus, we have

$$\begin{aligned} P(Y \geq 2) &= 1 - P(Y < 2) \\ &= 1 - \sum_{k=0}^{1} (0.52)^k (1 - 0.52)^{10-k}. \end{aligned}$$

enumerate

2 Machine Learning (21 points)

a. (5 points) When implementing logistic regression, a student decides to add a second intercept value. To do so they add an extra feature with value 0 to each datapoint. How will this impact training?

There will be no impact on training since we have a value of 0, so when we compute $\theta^T x$, this new feature will have no contribution to our probability.

b. (8 points) A Naive Bayes classifier is trained on a dataset with 100 examples, 30 of which are labeled as positive and 70 of which are labeled as negative. Instead of using a Laplace prior, you use a Beta($a = 3$, $b = 4$) prior. What is your estimate for the probability $Y = 1$?

This implies that we have 5 imaginary trials with 2 successes and 3 failures. We can update our MAP probability to be:

$$P(Y = 1) = \frac{30 + 2}{100 + 5} = \frac{32}{105}$$

c. (8 points) Sometimes, we would like to incorporate additional terms that represent interactions between different features into a logistic regression model. Imagine a dataset with two features, x_1 and x_2, and a corresponding label y for each datapoint. We will add the second-order feature $x_1 x_2$, so that our model is $P(Y = y|X = x) = \sigma(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 \cdot x_1 x_2)$. Explain in 1 or 2 sentences how you would change your logistic regression code in order to train this model.

We would have to update our code to compute this feature ($x_1 x_2$) and add it to our features list. We also extend our theta and gradient list to accommodate the new theta value.

3 Night Sight (20 points)

In this problem we explore how to use probability theory to take photos in the dark. Digital cameras have a sensor that capture photons over the duration of a photo shot to produce pictures. However, these sensors are subject to "shot noise" which are random fluctuations in the amount of photons that hit the lens. In the scope of this problem, we **only consider a single pixel**. The arrival of shot noise photons on a surface is independent with constant rate.

a. (6 points) Shot noise photons land on a particular pixel at a rate of 10 photons per microsecond (μs). If the time duration of a photo shot is 1000 μs, what is the variance of the amount of photons captured by the pixel during a single photo?

Answer. By memoryless property, we can model the amount of photons captured by the pixel as

$$C \sim \text{Poi}(10,000).$$

Then, the variance of the photons captured is 10,000 (quite high).

b. (14 points) To mitigate shot noise, Stanford graduates realized that you can take a shutter shot (many camera shots in quick succession) and average the number of photons captured. The largest number of photos a camera can take in 1000μs is 15 photos, each with a duration of 66μs. Let X be the average quantity of shot noise photons across the 15 photos, captured by the single pixel. What is $\text{Var}(X)$?

Answer. By the first part,

$$C_i \sim \text{Poi}(10,000)$$

By CLT, the variance is

$$\text{Var}\left(\frac{C_1 + \cdots + C_{15}}{15}\right) = \frac{1}{15^2}\text{Var}(C_i) = \frac{10000}{15^2} \approx 666/15$$

since each C_i are independent. We see this reduces the noise of the photons in the pixel by a factor of k^2 where k is the number of shots.

4 Penalty Shootout (20 points)

Soccer games may end up with a penalty shootout. Use probability to estimate the probability that a particular team will win. In a penalty shoot out each team takes 5 shots. If after 5 shots both teams have the same number of goals, they repeat taking one more shot each, until one team has more goals.

Assume that: Players on team A have a 0.8 probability of scoring on each penalty shot. Players on team B have a 0.7 probability of scoring on each penalty shot. Assume that each shot is independent. What is the exact probability that team A wins?

Let A_r be the random variable for the total number of penalties scored by team A, and B_r be the random variable for the total number of penalties scored by team B in the first five shots.
These random variables can be modelled using binomials

$$A_r \sim \text{Bin}(5, 0.8) \qquad\qquad B_r \sim \text{Bin}(5, 0.7)$$

Lets calculate p_r the probability that A wins in the first five goals $p_r = P(A_r > B_r)$:

$$\begin{aligned} p_r &= \sum_{a=1}^{5} \sum_{b=0}^{a-1} P(A_r = a, B_r = b) \\ &= \sum_{a=1}^{5} \sum_{b=0}^{a-1} P(A_r = a)(B_r = b) \\ &= \sum_{a=1}^{5} \sum_{b=0}^{a-1} \binom{5}{a} 0.8^a \cdot 0.2^{5-a} \binom{5}{b} 0.7^b \cdot 0.3^{5-b} \end{aligned}$$

Let p_t be the probability that the two teams tie in the first five goals:

$$\begin{aligned} p_t &= \sum_{i=0}^{5} P(A_r = i, B_r = i) \\ &= \sum_{i=0}^{5} \binom{5}{i} 0.8^i \cdot 0.2^{5-i} \binom{5}{i} 0.7^i \cdot 0.3^{5-i} \\ &= \sum_{i=0}^{5} \binom{5}{i}^2 (0.8 \cdot 0.7)^i \cdot (0.2 \cdot 0.3)^{5-i} \end{aligned}$$

If they tie in the first five goals, let p_e be the probability that team A wins in the extra shots:

$$p_e = \sum_{i=0}^{\infty} (0.8 \cdot 0.3)(0.2 \cdot 0.3 + 0.8 \cdot 0.7)^i$$

Aside: We can also compute p_e recursively:

$$\begin{aligned} p_e &= P(\text{A wins and no tie in round 1}) + P(\text{A wins and tie in round 1}) \\ &= (0.8) * (0.3) + (0.8 * 0.7 + 0.2 * 0.3) * p_e \\ p_e - 0.62 * p_e &= 0.24 \\ p_e(1 - 0.62) &= 0.24 \\ p_e &= \frac{0.24}{0.38} = 0.63 \end{aligned}$$

The overall probability is $p_r + p_t \cdot p_e$

$$\approx (NUM) + (0.273) * (0.6316)$$

5 Better (20 points)

Write a function `better` which returns the approximate probability that video A has a higher probability of being liked than video B, based on historical observations. Each video has two values: `likes` and `not_likes`. Model the probability that a viewer likes a movie as a random variable and use a Laplace prior for the random variable. You may use sampling if it is helpful (use on the order of 1 million samples).

```
def better(a_likes, a_not_likes, b_likes, b_not_likes):
    num_samples = 1000000

    count_where_aprob_gt_bprob = 0
    for samp in (range(num_samples)):
        # params for beta are (num_successes + 2) and (num_fails +2), with LaPlace
            prior
        a_sample_prob = scipy.stats.beta.rvs(a_likes + 2, a_not_likes + 2)
        b_sample_prob = scipy.stats.beta.rvs(b_likes + 2, b_not_likes + 2)
        if a_sample_prob > b_sample_prob:
            count_where_aprob_gt_bprob += 1

    return count_where_aprob_gt_bprob / num_samples
```

6 B-Reel (35 points)

A social media application "B-Reel" promises to send users a notification exactly once each day "randomly" in a 10 hour period. You want to test if the time that notifications come in are truly uniform. You have recorded 100 IID historical values: $[x_1, x_2, \ldots, x_{100}]$ where $x_i \in [0, 10]$ is the time the notification came in for the ith day, measured in hours from the start of the time period.

a. (5 points) Calculate the likelihood of the dataset given each value is IID from a uniform(0, 10). In otherwords, the density of each value.

Let D be the dataset event.

$$\begin{aligned} P(D|\text{uniform}) &= \prod_{i=1}^{100} P(X_i = x_i|\text{uniform}) \\ &= 0.1^{100} \end{aligned}$$

b. (10 points) You have an alternative hypothesis: there is a 0.4 probability a notification comes in the first half of the day, and a 0.6 probability that a notification comes in the second half of the day (and that the time is uniform within the halves). The historical data has 45 notifications in the first half of the day and 55 in the second half. What is the probability density of these 100 samples, given this alternative hypothesis?

$$\begin{aligned} P(D|\text{alternative}) &= \prod_{i=1}^{100} P(X_i = x_i|\text{alternative}) \\ &= (0.4 \cdot 1/5)^{45} * (0.6 \cdot 1/5)^{55} \end{aligned}$$

Many people will use 0.4 as the probability that an event will occur in the first 5 hours however, that isn't a probability density. The PDF is a two step process, the event comes in the first half and then we use the density of the uniform in the range 0 to 5, which is $\frac{1}{5-0}$.

Forgetting this adjustment or using a binomial are worth a good amount of partial credit, though importantly they aren't correct, because we want a value that would work for p_a. There is a very outside possibility that a student will use a binomial count here, which is consistent with a binomial count in part a. If they were consistent that would lead to the correct answer in part (c) and as such is worth almost full marks.

c. (8 points) Let p_a and p_b be your answers to part a and b respectively. What is the probability of your alternative hypothesis? Assume that the samples must come from either the uniform or the alternative hypothesis. Your prior belief that B-Reel is using a uniform(0,10) is 0.6.

$$P(\text{alternative}|D) = p_b * 0.4/(p_b * 0.4 + p_a * 0.6)$$

Allow error carried forward from previous parts (assume that p_a and p_b are correct).

d. (12 points) Perhaps we don't have enough data. Use bootstrapping to estimate the variance of the probability calculated, if you were to repeat this experiment 10,000 times. Let `data` be the list of historical values. You can use the function `var(list)` to estimate the sample variance from a list of values.

```
import numpy as np

def solution(data):
    n = len(data)
    probs = []
    for i in range(10000):
        resampled = np.random.choice(data, size=n, replace=True)

        # calculate likelihood of samples using method in part b.
        count_morning = sum([1 if num < 5 for num in resampled])
        count_night = n - count_morning
        prob_a = 0.1**100
        prob_b = ((0.4 * 0.2) ** count_morning) * ((0.6 * 0.2) **
            count_night)
        p_alternative = (prob_b * 0.4) / (prob_b * 0.4 + prob_a * 0.6)
        probs.append(p_alternative)
    return var(probs)
```

7 Code survival (20 points)

The Gopertz distribution can be used to model how long a piece of code will remain in production. It is defined by parameter a and has probability density function:

$$f(X = x) = 2a \cdot e^{3a - 2a \cdot e^{2x}}$$

We wish to model how long a particular code will last at a given company. To this end we collect N independent measurements of how long code lasts in production: $x_1, x_2, \ldots, x_N$. Explain, in words, how you would choose parameter a using the maximum likelihood estimation framework, and provide any necessary derivatives.

We start by defining our likelihood function

$$L(a) = \prod_{i=1}^{N} 2a \cdot e^{(3a - 2a \cdot e^{2x_i})}$$

We'll now compute the log likelihood as follows:

$$LL(a) = \sum_{i=1}^{N} \log(2a \cdot e^{(3a - 2a \cdot e^{2x_i})})$$

Now we can compute the derivative with respect to a.

$$\begin{aligned}
\frac{\partial LL}{\partial a} &= \frac{\partial}{\partial a} \sum_{i=1}^{N} \log(2a \cdot e^{(3a - 2a \cdot e^{2x_i})}) \\
&= \sum_{i=1}^{N} \frac{\partial}{\partial a} \log(2a \cdot e^{(3a - 2a \cdot e^{2x_i})}) \\
&= \sum_{i=1}^{N} \frac{\partial}{\partial a} (\log(2) + \log(a) + (3a - 2a \cdot e^{2x_i})) \\
&= \sum_{i=1}^{N} \frac{1}{a} + \frac{\partial}{\partial a} (3a - 2a \cdot e^{2x_i}) \\
&= \sum_{i=1}^{N} \frac{1}{a} + 3 - 2e^{2x_i}
\end{aligned}$$

We accept any answer that mentions gradient descent/ascent or setting the derivative equal to 0.

8 Approximate Counting Algorithm (22 points)

What if you wanted a counter that could count up to the number of atoms in the universe, but you wanted to store the counter in 8 bits? You could use the algorithm below. Show that the expected return value of `stochastic_counter(4)`, where `count` is called four times, is in fact equal to four.

```
def stochastic_counter(true_count):
    n = -1
    for i in range(true_count):
        n += count(n)
    return 2 ** n  # 2^n, aka 2 to the power of n

def count(n):
    # To return 1 you need n heads. Always returns 1 if n is <= 0
    for i in range(n):
        if not coin_flip():
            return 0
    return 1

def coin_flip():
    # returns true 50% of the time
    return random.random() < 0.5
```

Let X be a random variable for the value of n at the end of `stochastic_counter(4)`. Note that X is not a binomial because the probabilities of each outcome change.

Let R be the return value of the function. $R = 2^X$ which is a function of X. Use the law of unconscious statistician

$$E[R] = \sum_x 2^x \cdot P(X = x)$$

We can compute each of the probabilities $P(X = x)$ separately. Note that the first two calls to count will always return 1. Let H_i be the event that the ith call returns 1. Let T_i be the event that the ith call returns 0. X can't be less than 1 because the first two calls to count always return 1.
$P(X = 1) = P(T_3, T_4)$
$P(X = 2) = P(H_3, T_4) + P(T_3, H_4)$
$P(X = 3) = P(H_3, H_4)$

At the point of the third call to count, $n = 1$. If H_3 then $n = 2$ for the fourth call and the loop runs twice.

$$\begin{aligned} P(H_3, T_4) &= P(H_3) \cdot P(T_4|H_3) \\ &= \frac{1}{2} \cdot (\frac{1}{2} + \frac{1}{4}) \end{aligned}$$

$$\begin{aligned} P(H_3, H_4) &= P(H_3) \cdot P(H_4|H_3) \\ &= \frac{1}{2} \cdot \frac{1}{2} \end{aligned}$$

If T_3 then $n = 1$ for the fourth call.

$$\begin{aligned} P(T_3, H_4) &= P(T_3) \cdot P(H_4|T_3) \\ &= \frac{1}{2} \cdot \frac{1}{2} \end{aligned}$$

$$\begin{aligned} P(T_3, T_4) &= P(T_3) \cdot P(T_4|T_3) \\ &= \frac{1}{2} \cdot \frac{1}{2} \end{aligned}$$

Plug everything in:

$$\begin{aligned} E[R] &= \sum_{x=1}^{3} 2^x \cdot P(X = x) \\ &= 2 \cdot \frac{1}{4} + 4 \cdot \frac{5}{8} + 8 \cdot \frac{1}{8} \\ &= 4 \end{aligned}$$

9 Ranking (1 point)

Bonus point: rank the questions (1 - 8) from this exam in order of how confident you feel in the correctness of your answer. For example, writing $3 \leq 1 \leq 5$ would mean you are most confident in your answer for question 5 and least confident in your answer for question 3.

Please rank whole questions (1 - 8) and not subparts.

$$\_\_\_\_ \leq \_\_\_\_ \leq \_\_\_\_ \leq \_\_\_\_ \leq \_\_\_\_ \leq \_\_\_\_ \leq \_\_\_\_ \leq \_\_\_\_$$

That's all folks. Thank you all for the wonderful quarter and we hope you have a fantastic winter break. Night Sight is a real algorithm invented by Stanford CS folks and is now in production in Google pixel (and presumably other places too). Approximate counting is a real randomized algorithm which can count in log log space. Better is a more sophisticated way to rank videos than sorting by average likes.