

CS109 Midterm Exam 2

This is a closed calculator/computer/phone/smart-watch/smart-toothbrush exam. You are, however, allowed to use 5 pages (front and back) of handwritten notes in the exam. You have 75 minutes to take the exam. The exam is 65 points. You may want to use the point allocation for each problem as a rough 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. You can only leave your answer in **terms of code** when the problem explicitly says so.

Stanford Honor Code: The Honor Code is an undertaking of the Stanford academic community, individually and collectively. Its purpose is to uphold a culture of academic honesty. Students will support this culture of academic honesty by neither giving nor accepting unpermitted academic aid on this examination.

This course is participating in the proctoring pilot overseen by the Academic Integrity Working Group (AIWG), therefore proctors will be present in the exam room. The purpose of this pilot is to determine the efficacy of proctoring and develop effective practices for proctoring in-person exams at Stanford.

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: _____

First and Last Name (print): _____

Stanford Email (@stanford.edu): _____

Exam Break Sign-out

I pledge that during my exam break:

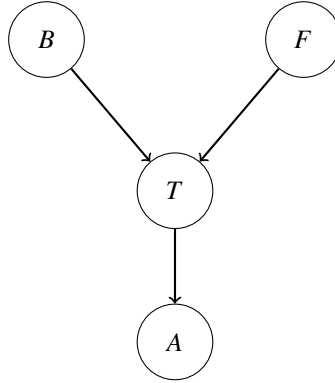
- I will not bring any paper, electronic devices (phone, smart watch, smart glasses, etc), or aid (permitted or unpermitted) *out of or into* the exam room, nor access any aid during the break.
- I will not communicate with anyone other than the course instructional staff about the content of the exam.

Signature Confirming Honor Code	Exit Time	Return Time	Proctor Initial	Length (mins)

If you are feeling unwell and are not able to complete the exam, please speak with the proctor.

1 Continuous Integration [13 Points]

Your team's continuous integration (CI) system is modeled by the Bayesian network below. All four random variables are binary (1 = true, 0 = false): B = the commit under test contains a **bug**, F = the test machine is running **flaky** infrastructure, T = the **test** suite fails, and A = the on-call engineer is **al**erted.



$$P(B = 1) = 0.1$$

$$P(F = 1) = 0.2$$

$$P(T = 1 \mid B = 1, F = 1) = 0.99 \quad P(T = 1 \mid B = 1, F = 0) = 0.90$$

$$P(T = 1 \mid B = 0, F = 1) = 0.50 \quad P(T = 1 \mid B = 0, F = 0) = 0.02$$

$$P(A = 1 \mid T = 1) = 0.95$$

$$P(A = 1 \mid T = 0) = 0.01$$

- (a) (3 points) Write the joint distribution $P(B, F, T, A)$ as a product of conditional probabilities implied by the network, and use it to compute $P(B = 0, F = 1, T = 0, A = 1)$.

Each node depends only on its parents, so the joint factors as

$$P(B, F, T, A) = P(B)P(F)P(T \mid B, F)P(A \mid T).$$

Reading the four needed numbers off the tables ($P(B = 0) = 0.9$, $P(F = 1) = 0.2$, $P(T = 0 \mid B = 0, F = 1) = 1 - 0.50 = 0.50$, $P(A = 1 \mid T = 0) = 0.01$):

$$P(B = 0, F = 1, T = 0, A = 1) = (0.9)(0.2)(0.50)(0.01) = 0.0009.$$

- (b) (4 points) The test suite just failed. What is $P(B = 1 \mid T = 1)$, the probability the commit really contains a bug?

Note that A is a descendant of T and is not observed, so it does not affect this computation. Marginalize over F using the law of total probability. Since B and F are independent (no edge between them, and neither is a descendant of the other):

$$\begin{aligned} P(B = 1, T = 1) &= P(B = 1) [P(F = 1)P(T = 1 | B = 1, F = 1) + P(F = 0)P(T = 1 | B = 1, F = 0)] \\ &= 0.1 [(0.2)(0.99) + (0.8)(0.90)] = 0.1(0.198 + 0.720) = 0.0918. \end{aligned}$$

$$P(B = 0, T = 1) = 0.9 [(0.2)(0.50) + (0.8)(0.02)] = 0.9(0.100 + 0.016) = 0.1044.$$

So $P(T = 1) = 0.0918 + 0.1044 = 0.1962$, and

$$P(B = 1 | T = 1) = \frac{0.0918}{0.1962} = \frac{51}{109} \approx 0.468.$$

- (c) (2 points) Suppose you already know that the test suite failed ($T = 1$). If you additionally learn that the commit contains a bug ($B = 1$), does the probability that the on-call engineer is alerted change? Answer yes or no and justify your answer in one sentence using the structure of the network.

No, it does not change. In the network A depends only on its parent T , and every path from B to A runs through T , so once T is known, B carries no additional information about A :

$$P(A = 1 | T = 1, B = 1) = P(A = 1 | T = 1) = 0.95.$$

- (d) (2 points) You could also have estimated the probability in part (b) using *rejection sampling*. State one advantage and one disadvantage of rejection sampling in general.

Advantage (any one): it is trivial to implement and requires only the ability to sample from the network, not to do any algebra; it scales to networks with many variables where exact marginalization would require summing over exponentially many assignments; it works unchanged for any query and any set of observed variables.

Disadvantage (any one): it produces only an approximation, with error that shrinks slowly (like $1/\sqrt{n}$) in the number of samples; and it is very wasteful when the observed evidence is unlikely, since nearly every sample gets thrown away. (This is the motivation for MCMC.)

- (e) (2 points) For *this particular* network and query, would you prefer to compute the answer by hand as in part (b), or to run rejection sampling? Justify your answer in one or two sentences.

By hand. The network is tiny (four binary variables, so only 16 joint assignments) and the evidence $T = 1$ is not rare ($P(T = 1) \approx 0.196$, so about 80% of samples would still be discarded). The exact calculation takes a few lines and gives the answer with no sampling error, whereas rejection sampling would only approximate $51/109$. Rejection sampling earns its keep on large networks where exact inference is intractable.

2 A Suspicious Coin [13 Points]

You come across a special coin, which has a heads side and a tails side. You can tell just by looking at it that it is weighted unusually, but you have no idea in which direction. Let p be the (unknown) probability that this coin comes up heads.

You may not use code to answer any part of this problem.

- (a) (3 points) Define a random variable, *including its parameters*, to represent your prior belief about p , before you have made any flips.

Let X be the random variable representing your belief about p . Take

$$X \sim \text{Beta}(a = 1, b = 1),$$

which is exactly the Uniform(0, 1) distribution. (Any symmetric Beta(a, a) is defensible if justified, but since you have no information at all about the direction of the weighting, Beta(1, 1) is the natural choice. A prior such as Beta(0.5, 0.5), which puts extra mass near 0 and 1, would also be a reasonable answer given the hint that the coin is weighted *unusually*.)

- (b) (2 points) Write a sentence justifying your choice of random variable.

The Beta distribution has support exactly on $[0, 1]$, which is the set of possible values of a probability, and it is the conjugate prior for a Bernoulli/Binomial likelihood, so the posterior after observing flips is again a Beta and can be written down without doing any integration. Choosing $a = b = 1$ makes the prior uniform, which encodes “every value of p is equally plausible” — appropriate when you know the coin is weighted but not which way.

- (c) (3 points) You flip the coin 8 times and observe 5 heads and 3 tails. What is the distribution for your *posterior* belief about p ?

For a Beta prior with a Binomial likelihood, the posterior just adds the observed counts to the parameters: with $a = 1$, $b = 1$ and 5 heads, 3 tails,

$$X \mid \text{data} \sim \text{Beta}(a = 1 + 5, b = 1 + 3) = \text{Beta}(6, 4).$$

(Its mean is $6/10 = 0.6$, pulled slightly toward $1/2$ from the observed $5/8 = 0.625$ by the prior.)

- (d) (3 points) For parts (d) and (e), suppose instead that you had observed an arbitrary x heads and y tails, with $x + y \geq 1$. Compared to your prior, did the *variance* of your belief distribution increase, decrease, or can you not say? Justify your answer.

It **decreased**, for every possible x and y with $x + y \geq 1$.
 The posterior is $\text{Beta}(a, b)$ with $a = 1 + x$, $b = 1 + y$, and

$$\text{Var}(\text{Beta}(a, b)) = \frac{ab}{(a+b)^2(a+b+1)}.$$

The prior $\text{Beta}(1, 1)$ has variance $\frac{1}{4 \cdot 3} = 1/12$. For the posterior, write $s = a + b = 2 + x + y \geq 3$.
 Since $ab \leq s^2/4$ for any a, b ,

$$\text{Var} = \frac{ab}{s^2(s+1)} \leq \frac{s^2/4}{s^2(s+1)} = \frac{1}{4(s+1)} \leq \frac{1}{16} < \frac{1}{12}.$$

Intuitively: data can only sharpen your belief. Every flip you observe concentrates the posterior, so the variance strictly decreases no matter what you see.

- (e) (2 points) Did the *entropy* of your belief distribution increase, decrease, or can you not say? Justify your answer.

It **decreased**.

Entropy measures your expected surprise, i.e. how uncertain you are. The prior $\text{Beta}(1, 1)$ is uniform on $[0, 1]$, and the uniform distribution is the maximum-entropy distribution on a bounded support — no other distribution on $[0, 1]$ is more uncertain. Any observation produces a non-uniform posterior, so the entropy must go down. (Formally, the differential entropy of $\text{Uniform}(0, 1)$ is 0, and every other density on $[0, 1]$ has differential entropy strictly less than 0.)

This is the information-theoretic statement of the same fact as part (d): observing data delivers information, and information reduces uncertainty.

3 Ride Sharing [12 Points]

You run a ride-sharing app. Each driver receives ride requests at a rate of 5 requests per hour, and each driver earns \$10 per ride fulfilled. Assume that requests arrive independently, that the number of requests is unbounded, and (for the sake of this problem) that a driver can fulfill an unlimited number of requests.

You may not use code to answer any part of this problem.

- (a) (3 points) What are the expected earnings over the course of one hour for an individual driver?

Requests arriving independently at a constant rate, with no upper bound on the count, is exactly the Poisson model. Let N be the number of requests one driver receives in an hour:

$$N \sim \text{Poi}(\lambda = 5), \quad E[N] = 5.$$

That driver's earnings are $W = 10N$, so by linearity of expectation

$$E[W] = 10E[N] = 10 \cdot 5 = \$50.$$

- (b) (3 points) What is the variance of an individual driver's earnings in one hour?

For a Poisson random variable the variance equals the rate, so $\text{Var}(N) = \lambda = 5$. Scaling a random variable by a constant scales its variance by the *square* of that constant:

$$\text{Var}(W) = \text{Var}(10N) = 10^2 \text{Var}(N) = 100 \cdot 5 = 500 \text{ (dollars}^2\text{)}.$$

(So the standard deviation of one driver's hourly earnings is $\sqrt{500} \approx \$22.36$.)

- (c) (6 points) Your fleet includes 100 drivers, who receive requests independently of one another. What is the probability that your fleet earns *at least* \$5,200 in one hour? You may use an approximation.

Let $T = \sum_{i=1}^{100} W_i$ be the fleet's total hourly earnings, where the W_i are i.i.d. copies of W from parts (a) and (b). By the Central Limit Theorem, a sum of many i.i.d. random variables is approximately Normal, with the means and variances adding:

$$E[T] = 100 \cdot 50 = 5000, \quad \text{Var}(T) = 100 \cdot 500 = 50000, \quad \sigma_T = \sqrt{50000} = 100\sqrt{5} \approx 223.6.$$

So $T \approx \mathcal{N}(\mu = 5000, \sigma^2 = 50000)$ and

$$P(T \geq 5200) \approx P\left(Z \geq \frac{5200 - 5000}{100\sqrt{5}}\right) = 1 - \Phi\left(\frac{2}{\sqrt{5}}\right) \approx 1 - \Phi(0.894) \approx 0.186.$$

Continuity correction (also accepted). T is always a multiple of 10, so $T \geq 5200$ is the same event as $T > 5190$, giving $1 - \Phi\left(\frac{190}{100\sqrt{5}}\right) = 1 - \Phi(0.872) \approx 0.192$.

Aside: here the exact answer is available too, since the total number of requests across the fleet is $\text{Poi}(500)$ and T is 10 times that; the CLT approximation is what we would reach for when the per-driver distribution is not a named one.

4 A Delicate Poll [14 Points]

You want to estimate the probability p that a randomly selected person believes the answer to a single delicate true/false question is “true.” Because the topic is sensitive, people will not answer it honestly if asked directly.

So instead you hand each participant 7 true/false questions and ask them only to report *how many* of the 7 they believe are true. One of the 7 is the delicate question. The other 6 are unimportant, non-sensitive questions for which each person is equally likely to answer true or false, independently of every other answer.

Let X_i be the number of questions person i believes are true. You survey $n = 100$ people and observe $X_1, \dots, X_{100}$, stored in a list `samples`.

- (a) (3 points) Derive $E[X_i]$ in terms of p , and use it to give an estimator $\hat{p}$ for p in terms of the sample mean $\bar{X}$.

Write $X_i = D_i + \sum_{j=1}^6 U_{ij}$, where D_i is the indicator that person i believes the delicate question is true and U_{ij} is the indicator for the j th filler question. By linearity of expectation (which does *not* require independence):

$$E[X_i] = E[D_i] + \sum_{j=1}^6 E[U_{ij}] = p + 6 \cdot \frac{1}{2} = p + 3.$$

Setting the sample mean equal to this expectation and solving for p :

$$\hat{p} = \bar{X} - 3.$$

- (b) (6 points) The code below uses bootstrapping to estimate the probability that $\hat{p}$ is within 0.1 of the true value of p . Line (i) is written for you; fill in the three remaining blanks. `np.random.choice(a, size, replace)` draws `size` values from the list `a`; `np.mean(a)` returns the average of the list `a`.

```
import numpy as np
NUM_BOOTSTRAPS = 10000

def estimate_p(samples):
    return np.mean(samples) - 3

def bootstrap_p_within(samples):
    our_estimate = estimate_p(samples)
    n = len(samples)
    count = 0
    for i in range(NUM_BOOTSTRAPS):
        resample = np.random.choice(samples, size=n, replace=True) # (i)

        p_hat = _____ # (ii)

        if _____: # (iii)
            count += 1
```

```
return ----- # (iv)
```

```
(ii) p_hat = estimate_p(resample)
```

```
(iii) if abs(p_hat - our_estimate) < 0.1:
```

```
(iv) return count / NUM_BOOTSTRAPS
```

The idea: we cannot draw fresh samples of 100 people from the true population, so the bootstrap treats the sample we have as if it *were* the population, and draws new samples of the same size n from it *with replacement*. Each resample yields one new estimate $\hat{p}$, and the spread of those estimates around `our_estimate` approximates the spread of $\hat{p}$ around the true p . Line (iv) converts the tally into a proportion.

- (c) (3 points) Explain what would go wrong if line (i) were changed to use `replace=False`, and what would go wrong if it instead used `size=n//2`.

replace=False: drawing n items from a list of n without replacement returns a permutation of the original sample. Every resample would have the identical mean, so $\hat{p}$ would equal `our_estimate` every single time, `count` would be `NUM_BOOTSTRAPS`, and the function would always return 1.0. It would report zero uncertainty.

size=n//2: the resamples would be half as large as the real sample, so their means would be roughly $\sqrt{2}$ times as variable as the real estimator. The bootstrap distribution would be too wide and the function would *understate* the probability of landing within 0.1. The resample size must match the original sample size n , because that is the sampling situation we are trying to simulate.

- (d) (2 points) The number returned approximates $P(|\hat{p} - p| < 0.1)$. What assumption does the bootstrap rely on to make this valid?

The bootstrap assumes that the observed sample is a good stand-in for the true population — that is, that the empirical distribution of the 100 observed values closely resembles the true distribution of X , so that resampling from the sample mimics drawing fresh samples from the population.

5 Load Balancing [13 Points]

A load balancer routes each incoming request, independently, to one of three servers: server A with probability $1/2$, server B with probability $1/3$, and server C with probability $1/6$.

You may not use code to answer any part of this problem.

- (a) (3 points) Twelve requests arrive. What is the probability that exactly 6 go to A , exactly 4 go to B , and exactly 2 go to C ?

Let $(N_A, N_B, N_C) \sim \text{Multinomial}(n = 12; 1/2, 1/3, 1/6)$. Using the Multinomial PMF,

$$P(N_A = 6, N_B = 4, N_C = 2) = \binom{12}{6, 4, 2} \left(\frac{1}{2}\right)^6 \left(\frac{1}{3}\right)^4 \left(\frac{1}{6}\right)^2 = \frac{12!}{6!4!2!} \cdot \frac{1}{2^6 \cdot 3^4 \cdot 6^2}.$$

Numerically, $\frac{12!}{6!4!2!} = 13860$ and $2^6 \cdot 3^4 \cdot 6^2 = 186624$, so the probability is $\frac{13860}{186624} = \frac{385}{5184} \approx 0.074$.

- (b) (2 points) Still with 12 requests, let N_C be the number of requests routed to server C . State the distribution of N_C (including its parameters), and give $E[N_C]$ and $\text{Var}(N_C)$.

Collapse the outcome into “went to C ” versus “did not,” and each request is an independent trial with success probability $1/6$. Any single coordinate of a Multinomial is Binomial:

$$N_C \sim \text{Bin}(n = 12, p = \frac{1}{6}), \quad E[N_C] = 12 \cdot \frac{1}{6} = 2, \quad \text{Var}(N_C) = 12 \cdot \frac{1}{6} \cdot \frac{5}{6} = \frac{5}{3}.$$

- (c) (3 points) Let $S \in \{A, B, C\}$ be a random variable representing the server that handles a single request. Compute the entropy $H(S)$, using $\log_2$. Then state which routing probabilities would *maximize* $H(S)$, and what that maximum value is.

Entropy is expected surprise, $H(S) = \sum_s P(s) \log_2 \frac{1}{P(s)}$:

$$H(S) = \frac{1}{2} \log_2 2 + \frac{1}{3} \log_2 3 + \frac{1}{6} \log_2 6 = \frac{1}{2} + \frac{1}{3} \log_2 3 + \frac{1}{6} \log_2 6 \approx 0.500 + 0.528 + 0.431 = 1.459.$$

Entropy over a finite set is maximized by the uniform distribution, here $P(A) = P(B) = P(C) = 1/3$, giving

$$H_{\max} = \log_2 3 \approx 1.585.$$

Interpretation: with $\log_2$ the units are bits, and 1.459 is (about) the average number of bits you would need to record which server handled each request. The skewed routing is slightly more predictable than uniform routing, so it takes slightly fewer bits to describe.

- (d) (2 points) Server A processes a request in 2 ms, server B in 3 ms, and server C in 10 ms. What is the expected processing time of a single request?

Let T be the processing time. By the law of total expectation, conditioning on which server was chosen:

$$E[T] = \sum_s P(S = s) E[T | S = s] = \frac{1}{2}(2) + \frac{1}{3}(3) + \frac{1}{6}(10) = 1 + 1 + \frac{5}{3} = \frac{11}{3} \approx 3.67 \text{ ms}.$$

- (e) (3 points) Server C is overloaded. With probability $1/4$ it drops the request after spending its full 10 ms on it, and the dropped request is then sent back through the load balancer and routed again from scratch, independently of everything that came before. (Servers A and B never drop requests.) Let R be the total time from arrival until the request is finally processed. Compute $E[R]$.

Condition on the first routing decision. Note that if the request is dropped by C , the situation after the retry is *identical* to the situation at the start, so the additional expected time is again $E[R]$:

$$E[R] = \frac{1}{2}(2) + \frac{1}{3}(3) + \frac{1}{6}\left(10 + \frac{1}{4}E[R]\right).$$

Simplifying,

$$E[R] = 1 + 1 + \frac{10}{6} + \frac{1}{24}E[R] = \frac{11}{3} + \frac{1}{24}E[R].$$

Solving for $E[R]$:

$$\frac{23}{24}E[R] = \frac{11}{3} \implies E[R] = \frac{11}{3} \cdot \frac{24}{23} = \frac{88}{23} \approx 3.83 \text{ ms.}$$

(Sanity check: this is a little larger than the $11/3 \approx 3.67$ ms from part (d), as it must be, since retries can only add time.)

You're done! A few parting thoughts. The CI network in Problem 1 is a real pattern: "the tests failed" is much weaker evidence of a bug than it feels like in the moment, because flaky infrastructure explains failures too. Problem 2 is the whole Bayesian story in five parts — a prior, some data, a sharper posterior — and parts (d) and (e) say the same thing twice, once in the language of variance and once in the language of information. The delicate-polling design in Problem 4 is a genuine technique from survey statistics called randomized response, and the bootstrap is what lets you put error bars on it without knowing the population.