Jerry Cain
CS109

Section Handout 7
May 29, 2025

Section 7 Solution: Machine Learning

Chris Piech, Jerry Cain, and the CS109 teaching teams

Before you leave lab, click here and complete the sequence of questions to demonstrate you've attended this week's section. The CA leading your section can enter the password needed once you've submitted.

Problem 1 specifically exercises the machine learning you learned last Wednesday and Friday. The remaining problems exercise material from the latter half of the course and are representative of the types of problems you might see on your own final exam.

1 The Most Important Features

Let's explore saliency, a measure of how important a feature is for classification. We define the saliency of the ith input feature for a given example $(\mathbf{x}, y)$ to be the absolute value of the partial derivative of the log likelihood, with respect to that input feature $\left|\frac{\partial LL}{\partial x_i}\right|$. Below, we show both input images and the corresponding saliency of their features (in this case, pixels) for an image classification model:

Consider a trained logistic regression classifier with weights θ that predicts binary class labels, y. In this question, we allow the values of $\mathbf{x}$ to be real numbers, which doesn't change the algorithm at all (neither training nor testing).

a. What is the log-likelihood of a single training example $(\mathbf{x}, y)$ for this logistic regression classifier?

The log-likelihood here is the same as it was when we covered logistic regression in class:
$LL(\theta) = y \cdot \log \sigma\left(\theta^T \cdot \mathbf{x}\right) + \left(1 - y\right)\log\left[1 - \sigma\left(\theta^T \cdot \mathbf{x}\right)\right]$

b. Calculate the saliency of a single feature (x_i) for one training example ($\mathbf{x}$, y). Note that the derivative of the sigmoid function, $\sigma(x)$ is $\sigma'(x) = \sigma(x)(1 - \sigma(x))$.

We can calculate the saliency for a single feature as follows. Note that we are taking the derivative with respect to x_i, not θ_i, which is different from the usual derivative we find for MLE optimization.

$$LL(\theta) = y \log z + \Big(1 - y\Big) \log \Big(1 - z\Big) \qquad \text{where } z = \sigma\Big(\theta^T \cdot \mathbf{x}\Big)$$

$$\frac{\partial LL}{\partial x_i} = \frac{\partial LL}{\partial z} \cdot \frac{\partial z}{\partial x_i} \qquad \text{chain rule}$$

$$= \Big(\frac{y}{z} - \frac{1-y}{1-z}\Big) \cdot \Big(z(1-z)\theta_i\Big) \qquad \text{partial derivatives}$$

$$\text{saliency} = \Big| \Big(\frac{y}{z} - \frac{1-y}{1-z}\Big) z(1-z)\theta_i \Big|$$

c. Show that the ratio of saliency for features i and j is the ratio of the absolute value of their weights $\frac{|\theta_i|}{|\theta_j|}$.

We can take the ratio as follows using our expression above.

$$\text{saliency for feature } i, S_i = \Big| \Big(\frac{y}{z} - \frac{1-y}{1-z}\Big) z(1-z)\theta_i \Big| \text{, and same for } S_j$$

$$\frac{S_i}{S_j} = \frac{\Big| \Big(\frac{y}{z} - \frac{1-y}{1-z}\Big) z(1-z)\theta_i \Big|}{\Big| \Big(\frac{y}{z} - \frac{1-y}{1-z}\Big) z(1-z)\theta_j \Big|} = \frac{S_i}{S_j} = \frac{\Big|\theta_i\Big|}{\Big|\theta_j\Big|} \text{ by elimination}$$

2 Timing Attacks

In this problem, we will see how to crack a password in linear time by measuring how long the password check takes to execute. Consider the following function definition:

```
# An insecure string comparison
def does_password_match(guess, password):
    n_guess = len(guess)
    n_password = len(password)
    if n_guess != n_password:
        return False                    # 4 lines executed to get here
    for i in range(n_guess):
        if guess[i] != password[i]:
            return False                # 6 + 2i lines executed to get here
    return True                         # 5 + 2n lines executed to get here
```

Assume that our server takes T ms to execute any line in the code where $T \sim N(\mu = 5, \sigma^2 = 0.5)$ milliseconds. The amount of time taken to execute a line is always independent of other lines.

On our site, all passwords only use lower case letters and are between 5 and 10 letters long, inclusive. A hacker is trying to crack the root password which is "gobayes" by carefully measuring how long the code takes to tell her that her guesses are incorrect.

a. What is the distribution for the time it takes to execute k lines of code?

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

$$T_k \sim N(\mu = 5k, \sigma^2 = 0.5k)$$

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

Time to run 6 lines of code: $T_6 \sim N(\mu = 30, \sigma^2 = 3)$
Time to run 4 lines of code: $T_4 \sim N(\mu = 20, \sigma^2 = 2)$ Then we apply a linear transform to T_4 so we can subtract it from T_6 by "adding" two Normal RVs.

$$\begin{aligned} -T_4 &\sim N(\mu = -20, \sigma^2 = 2) \\ T_6 - T_4 &\sim N(\mu = 10, \sigma^2 = 5) \\ P(T_6 > T_4) &= P(T_6 - T_4 > 0) \\ &= 1 - F_{T_6 - T_4}(0) \\ &= 1 - \Phi\left(\frac{0 - 10}{\sqrt{5}}\right) \approx 1.0 \end{aligned}$$

c. Now that our hacker knows the length of the password, to get the actual string, she will try to determine one letter at a time, starting with the first letter. To start, the hacker tries the string "aaaaaaa" and sees that it takes 27ms. 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. The problem's wording indicates that we are looking for a ratio between the probability of M^C to the probability of M, conditioned on observing a code execution time of 27ms. Let $T_?$ be the time it took to check the password, with the number of lines executed being unknown. To calculate this ratio, we can apply Bayes' Theorem:

$$\begin{aligned}\frac{P(M^C|T_? = 27)}{P(M|T_? = 27)} &= \frac{f(T_? = 27|M^C)P(M^C)}{f(T_? = 27|M)P(M)} \\ &= \frac{f(T_? = 27|M^C)\frac{25}{26}}{f(T_? = 27|M)\frac{1}{26}} \\ &= 25 \cdot \frac{f(T_? = 27|M^C)}{f(T_? = 27|M)}\end{aligned}$$

From here, we can reason about how $T_?$ is distributed if we know M or M^C. If M happened, then $T_? = T_8$; otherwise, $T_? = T_6$. So we can re-write the above as:

$$\begin{aligned}&= 25 \cdot \frac{f(T_6 = 27)}{f(T_8 = 27)} \\ &= 25 \cdot \frac{\frac{1}{\sqrt{6\pi}} e^{-\frac{(27-30)^2}{6}}}{\frac{1}{\sqrt{8\pi}} e^{-\frac{(27-40)^2}{8}}} \\ &= 25 \cdot \frac{\sqrt{8}}{\sqrt{6}} \cdot \frac{e^{-\frac{9}{6}}}{e^{-\frac{169}{8}}} \\ &\approx 9.6 \text{ million}\end{aligned}$$

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!

Since the password is length 7: $6 + 7 \cdot 26 = 188$.

3 When Milk Goes Bad

If you've ever shopped for milk while grocery shopping, you may have noticed the expiration date on pasteurized milk is always about a week after it's been packaged and shelved, whereas **organic** milk lasts much longer, at just over a month.

Organic milk lasts longer for reasons that have little to do with the fact that it's organic. Rather, organic milk generally lasts longer because those producing it use a different processone that heats the milk to some 280 F° for several secondsto preserve it.

Assume that pasteurized milk sours after an amount of time that's governed by an Exponential distribution with $\lambda = \frac{1}{150}$ hours^{-1}, so that on average it lasts around 150 hours, or just under one week. Assume organic milk sours after an amount of time that's also guided by an Exponential, but with $\lambda = \frac{1}{750}$ hours^{-1}, so that on average organic milk sours after a little more than a month. Assume an isolated gallon of milk is organic with probability $p = \frac{1}{45}$, though by examining a single gallon of milk that's yet to be marked with an expiration date, it's impossible to tell whether or not it's organic.

Let T be how long a mystery gallon of milk lasts, and let B indicate whether that gallon of milk is organic.

a. Find the CDF of T, and from that CDF, derive the PDF.

The CDF is easier to generate first, since it's well defined for any Exponential. In our case, however, the CDF is a mixture of two Exponentials that can be managed by the Law of Total Probability.

$$\begin{aligned} P(T < t) &= P(T < t|B = 1)P(B = 1) + P(T < t|B = 0)P(B = 0) \\ &= \boxed{\frac{1}{45} \cdot (1 - e^{-t/750}) + \frac{44}{45} \cdot (1 - e^{-t/150})} \\ &= 1 - \frac{1}{45} \cdot e^{-t/750} - \frac{44}{45} \cdot e^{-t/150} \end{aligned}$$

The PDF is simply the derivative of the CDF with respect to t.

$$f(t) = \frac{1}{750} \cdot \frac{1}{45} e^{-t/750} + \frac{1}{150} \cdot \frac{44}{45} e^{-t/150}$$

b. Based on your answer from part a., state whether T is memoryless or not, and defend your answer.

Certainly not! Only pure Exponentials with a single rate parameter are memoryless, but not with a hybrid.

c. Find the conditional distribution of B given $T = t$that is, present an expression for $P(B = 1|T = t)$. To be clear, $B = 1$ means the gallon of milk is organic, and $B = 0$ means it's not.

$$P(B = 1|T = t) = \frac{f(T = t|B = 1)P(B = 1)}{f(t)}$$
$$= \boxed{\frac{\frac{1}{750} \cdot \frac{1}{45} e^{-t/750}}{\frac{1}{750} \cdot \frac{1}{45} e^{-t/750} + \frac{1}{150} \cdot \frac{44}{45} e^{-t/150}}} = \frac{e^{-t/750}}{e^{-t/750} + 220 \cdot e^{-t/150}}$$

d. What does your conditional probability approach as t approaches 0? Explain why your answer here makes intuitive sense.

The above approaches $P(B = 1|T = t) = \frac{1}{221}$ (or whatever value was consistent with your answer to part c). Before getting any timing information, we assume milk is organic with probability $p = \frac{1}{45}$, but after learning that $t = 0$i.e. that the milk went sour immediatelywe should be even less inclined to believe the milk is organic, since organic milk tends to last much longer than pasteurized.

e. What value of $T = t$ would lead you to believe the mystery gallon is just as likely to be organic as it is pasteurized? (You needn't actually solve for t, but you do need to be clear what equation should be solved for you to arrive at the correct t value.)

It's whatever value of t actually solves this:

$$P(B = 1|T = t) = \frac{e^{-t/750}}{e^{-t/750} + 220 \cdot e^{-t/150}} = \frac{1}{2}$$

While this can be solved exactly using ln to isolate the t's, we didn't require it as long as you knew that $P(B = 1|T = t) = \frac{1}{2}$ and were what equation needed to be solved.

4 The Aurora Borealis, Priors, and Posteriors

The Aurora Borealis, more colloquially known as the Northern Lights, is a magnetospheric phenomenon where expansive bands of green and sometimes red light illuminate the sky. The lights are generally visible in countries closer to Earth's magnetic north pole, though on occasion, as was the case this past May, they extend down into the Continental United States.

Because you were able to see the Northern Lights from Stanford campus, you've decided to take leave this coming winter quarter and live in Northern Finland for two months, far away from any major cities. Your expectation is that, because you'll be above the Arctic Circle, you'll be able to see the Northern Lights most nights.

Let X be the random variable modeling the probability you see the Northern Lights on any particular night while living it up in Northern Finland. Based on your extensive Wikipedia research ahead of the trip, you learn that there are varying opinions on what this probability is, although most sources believe it to be fairly high. You decide to go with the prior belief distribution on X that's $\text{Beta}(47, 15)$.

You also confirm that whether or not you see the lights on any given night is independent of all other nights. Oh, and even though it's freezing, we'll assume the skies are always clear at night, so if there's light in the sky, you see it.

a. [2 points] What is p, which we'll take to be the mode of your prior belief distribution for X?

We're told that $X_{\text{prior}} \sim \text{Beta}(a = 47, b = 15)$, so the mode p is simply defined to be:

$$p = \frac{a-1}{a+b-2} = \boxed{\frac{47-1}{47+15-2}} = \frac{46}{60} = \frac{23}{30}$$

b. After 60 nights of clear skies in Northern Finland, you count 50 nights when you see the Northern Lights and 10 when you don't. What is q, which we'll assume to be the mode of your posterior belief distribution for X?

Because Beta is a conjugate prior of the Binomial, and these 60 nightly observations are guided by a Binomial, we get to add those 50 successes to a and those 10 failures to b, resulting in an updated posterior to be:

$$\boxed{X_{\text{posterior}} \sim \text{Beta}(a = 47 + 50 = 97, b = 15 + 10 = 25)}$$

The new mode is determined by the same type of computation, but with different a and b values. This time:

$$q = \frac{a-1}{a+b-2} = \boxed{\frac{97-1}{97+25-2}} = \frac{96}{120} = \frac{4}{5}$$

c. Using your **posterior distribution**, how many times more likely is $X = q$ than $X = p$? You don't need to arrive at a final number, but you should simplify your expression as much as possible.

$$\text{odds} = \frac{f_{\text{posterior}}(q)}{f_{\text{posterior}}(p)} = \boxed{\frac{\frac{1}{B(97,25)}q^{96}(1-q)^{24}}{\frac{1}{B(97,25)}p^{96}(1-p)^{24}}} = \frac{q^{96}(1-q)^{24}}{p^{96}(1-p)^{24}}$$
$$= \frac{(\frac{4}{5})^{96}(\frac{1}{5})^{24}}{(\frac{23}{30})^{96}(\frac{7}{30})^{24}} = (\frac{24}{23})^{96}(\frac{6}{7})^{24} \approx 1.47124$$

d. You learn that you would be extremely lucky to get two full months of clear skies, because on average, the skies are clear only 50% of the time. To better model your prior belief distribution now, you go with a mixture of two Betas, so that $X \sim \frac{1}{2}\text{Beta}(47, 15) + \frac{1}{2}\text{Beta}(1, 61)$. Explain why parameters of 1 and 61 are reasonable choices for that second Beta, and also explain whether the Binomial is still a conjugate of this hybrid form.

To choose 1 and 61 is to imagine 60 days without lights, which would almost certainly be the case should it be overcast or cloudy every night. And yes, the Binomial is still a conjugate here, since the prior is a linear combination of two Betas.

e. We certainly believe that the Beta distribution is a conjugate prior for the Bernoulli and Binomial distributions, but is it also a conjugate prior to the Geometric and Negative Binomial? Explain your answer.

Certainly, because the PMFs each have the same parametric form as the Binomial. For both, that form is $c\theta^m(1-\theta)^n$, and that meshes algebraically with the PDF of the Beta in precisely the same way the Binomial does.

5 The Beta Distribution, MLE, and Gradient Ascent

Recall that the Beta(a, b) random variable comes with the following probability density function:

$$f(x) = \frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1}$$

where $B(a, b)$ is in place so that the integral of $f(x)$ from 0 to 1 equals 1. To be clear, however, the value of $B(a, b)$ depends on both a and b, so it's technically a function on each.

Assume you've observed a sample of iid random variables $(X_1, X_2, \ldots, X_n)$, where $X_i \sim \text{Beta}(a, b)$ for all n samples. You'd like to estimate the a and b parameters using MLE.

a. [4 points] What is the log-likelihood function $LL(\theta = (a, b))$ of the sample $(X_1, X_2, \ldots, X_n)$? Simplify using properties of logarithms where possible.

$$\begin{aligned} L(\theta) &= \prod_{i=1}^{n} \frac{1}{B(a,b)} X_i^{a-1}(1-X_i)^{b-1} \\ LL(\theta) = \log L(\theta) &= \sum_{i=1}^{n} \log \frac{1}{B(a,b)} X_i^{a-1}(1-X_i)^{b-1} \\ &= -n \log B(a,b) + (a-1)\sum_{i=1}^{n} \log X_i + (b-1)\sum_{i=1}^{n} \log (1-X_i) \end{aligned}$$

b. What are $\frac{\partial LL(\theta)}{\partial a}$ and $\frac{\partial LL(\theta)}{\partial b}$, the derivatives of the log-likelihood with respect to a and b? Note that $B(a, b)$ is a function of both a and b, so its partial derivatives with respect to each contribute to your answer. However, you can simply refer to those derivatives as $B'_a(a, b)$ and $B'_b(a, b)$.

$$\begin{aligned} \frac{\partial LL(\theta)}{\partial a} &= -n \frac{B'_a(a,b)}{B(a,b)} + \sum_{i=1}^{n} \log X_i \\ \frac{\partial LL(\theta)}{\partial b} &= -n \frac{B'_b(a,b)}{B(a,b)} + \sum_{i=1}^{n} \log (1-X_i) \end{aligned}$$

c. While we can't exactly solve for a_{MLE} and b_{MLE} all that easily, we can use **gradient ascent** to find the parameter values that optimize $LL(\theta)$. Using the expressions you derived in part b, implement the function `estimate_parameters` below. Your implementation should perform gradient ascent using the supplied arguments and ultimately return a_{MLE} and b_{MLE}, the maximum likelihood estimates of a and b, as a pair or in a list of length 2.

- `sample` is a list of values corresponding to a sample $(X_1, X_2, \ldots, X_n)$, where all $X_i \sim \text{Beta}(a, b)$ are iid.
- `max_steps` is the number of times to perform gradient ascent (default is 10000)
- `eta` is the learning rate to use when updating α (default is 0.0001)
- `init` is the initial value used for each of a and b prior to any gradient ascent (default is 1).

You can also rely on the following predefined functions:

- `log(x)`, which returns the natural logarithm of a value or list of values `x`.
- `sum(arr)`, which returns the sum of all elements in `arr`.
- `B(a, b)`, which returns precisely thatthat value of $B(a, b)$
- `dBa(a, b)`, which returns $B'_a(a, b)$.
- `dBb(a, b)`, which returns $B'_b(a, b)$.

Pseudocode is fine as long as your code accurately conveys your approach. You must, however, convey the details of gradient ascent and how the derivatives you computed in part b. contribute to your implementation.

```
# estimate_parameters uses gradient ascent to arrive at MLE estimates of a
    and b.
def estimate_parameters(sample, max_steps = 10000, eta = 0.0001, init = 1):
    a, b = init, init
    n = len(sample)
    for i in range(max_steps):
        grad_a, grad_b = -n * dBa(a, b) / B(a, b), -n * dBb(a, b) / B(a, b)
        for x in sample: grad_a, grad_b = grad_a + np.log(x), grad_b + np.
            log(1 - x)
        a, b = a + eta * grad_a, b + eta * grad_b
    return a, b
```

Your solution didn't need to be this compact, but it did need to convey a detailed understanding of the algorithm when applied to something other than linear and logistic regression.