

# Binary Density Estimation

## CPSC 440/550: Advanced Machine Learning

`cs.ubc.ca/~dsuth/440/24w2`

University of British Columbia, on unceded Musqueam land

2024-25 Winter Term 2 (Jan–Apr 2025)

## Motivation: COVID-19 prevalence

- What percentage of UBC students have COVID-19 right now?
- “Brute force” approach (census):
  - Line up every single student, test them all, count the portion that test positive
- Statistical approach (survey):
  - Grab an “independent and identically distributed” (iid) sample of students
  - Estimate the proportion that have it, based on the sample

## General problem: binary density estimation

- This is a special case of **density estimation** with binary data:
  - Input:  $n$  **iid samples** of binary values  $x^{(1)}, x^{(2)}, \dots, x^{(n)} \in \{0, 1\}$
  - Output: a **probability model** for a random variable  $X$ : here, just  $\Pr(X = 1)$
- As a picture:

$\mathbf{X} \in \mathbb{R}^{n \times 1}$  contains our **sample data**  
 $X$  is a **random variable** over  $\{0, 1\}$  from the distribution

$$\mathbf{X} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \xrightarrow{\text{density estimator}} \Pr(X = 1) = 0.4$$

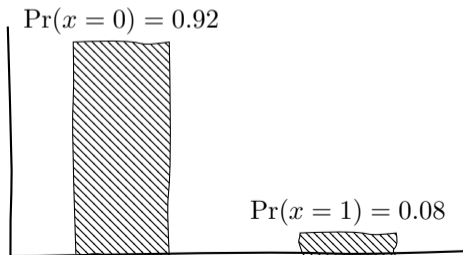
- We'll start by discussing major concepts for this very simple case
  - We'll slowly build to more complicated cases
  - Beyond binary data, more than one variable, conditional versions, deep versions, etc

## Other applications of binary density estimation

- Some other questions we might ask:
  - ① What's the probability this medical treatment works?
  - ② What's the probability that if you plant 10 seeds, at least one will germinate?
  - ③ How many lottery tickets should you expect to buy before you win?
- In the first example, we're computing  $\Pr(X = 1)$  like before
- For the other two, we're using the model to compute some other quantity
  - We call all three “inference” with this model

## Model definition: Bernoulli distribution

- We're going to start by using a **parameterized** probability model
  - i.e. a model with some **parameters we can learn**
- For binary variables, we usually use the **Bernoulli distribution**
- $x$  is Bernoulli with **parameter  $\theta$** , or  $x \sim \text{Bern}(\theta)$ , if  $\Pr(X = 1 \mid \theta) = \theta$ 
  - In the COVID example, if  $\theta = 0.08$ , we think 8% of the population has COVID
- **Require that  $0 \leq \theta \leq 1$**  for a valid probability distribution



## Digression: “inference” in statistics vs. ML

bonus!

- In machine learning, the usual terminology is:
  - “Learning” is the task of going from data  $\mathbf{X}$  to parameters  $\theta$
  - “Inference” is the task of using the parameters  $\theta$  to infer/predict something
- Statisticians sometimes use a “reverse” terminology:
  - Given data, you can “infer” parameters  $\theta$
  - Given parameters  $\theta$ , you can predict something
- This is partly influenced by the history of the two communities:
  - Statisticians often assume there’s a “true” parameter we can infer things about
  - ML hackers often focus on making predictions
- Some people use “inference” in both ways!
- We’ll use the ML terminology

## Inference task: computing probabilities

- An inference task: given  $\theta$ , compute  $\Pr(X = 0 \mid \theta)$
- We'll also sometimes write this as  $p(0 \mid \theta)$ ,  $p_\theta(0)$ , or just  $p(0)$ 
  - Be careful you know what we're abbreviating! "Explicit is better than implicit"
- Recall that probabilities add up to 1: since  $X \in \{0, 1\}$ ,

$$\Pr(X = 0 \mid \theta) + \Pr(X = 1 \mid \theta) = 1$$

- Since  $\Pr(X = 1 \mid \theta) = \theta$  by definition, this gives us

$$\Pr(X = 0 \mid \theta) + \theta = 1$$

- and so if  $X \sim \text{Bern}(\theta)$ , we know  $\Pr(X = 0 \mid \theta) = 1 - \theta$
- First inference task down!

## Bernoulli distribution notation

- It's sometimes helpful to combine the Bernoulli distribution into one expression:

$$p(x \mid \theta) = \theta^x (1 - \theta)^{1-x} = \theta^{\mathbb{1}(x=1)} (1 - \theta)^{\mathbb{1}(x=0)}$$

- $\mathbb{1}$  is an “indicator function”:  $\mathbb{1}(E)$  is 1 if the condition  $E$  is true, and 0 if it's not

## Aside: $p$ for probability masses

bonus!

- If you're like me, you might be bothered by using a lowercase  $p$  in  $p(0 \mid \theta)$ 
  - It's a probability mass, not a density!
- This is really really common among ML people, but when I first taught this class I started trying to change them all to  $P$  – or even to change everything to  $\Pr$
- ...it got really really messy (why this is really really common among ML people)
- If you're like me, this might be reassuring:
  - $p$  actually **is** a probability density for the Bernoulli distribution
  - It's just the Radon-Nikodym derivative wrt  $\mu(A) = \mathbb{1}(0 \in A) + \mathbb{1}(1 \in A)$
- If you haven't seen measure-theoretic probability, don't worry – it's not actually relevant to this course
- But it justifies “mixing” masses and densities willy-nilly

# Outline

- 1 Bernoulli distributions
- 2 Bernoulli inference tasks

## Inference task: computing dataset probabilities

- **Inference task**: given  $\theta$  and an iid sample, compute  $p(x^{(1)}, x^{(2)}, \dots, x^{(n)} \mid \theta)$
- Also called the “likelihood”:  $\Pr(X^{(1)} = x^{(1)}, X^{(2)} = x^{(2)}, \dots, X^{(n)} = x^{(n)} \mid \theta)$ 
  - Many ways to estimate/learn  $\theta$  need this, e.g. maximum likelihood estimation
  - Also helpful in comparing models on validation/test data
- Assuming the  $X^{(i)}$  are independent given  $\theta$ , we have

$$p(x^{(1)}, x^{(2)}, \dots, x^{(n)} \mid \theta) = \prod_{i=1}^n p(x^{(i)} \mid \theta)$$

- We'll talk more explicitly about conditional independence a little later in the course

## Inference task: computing dataset probabilities

- Using the independence property, for example,  $p(1, 0, 1, 1 \mid \theta)$  is

$$\begin{aligned} p\left(x^{(1)}, \dots, x^{(4)} \mid \theta\right) &= \prod_{i=1}^4 p\left(x^{(i)} \mid \theta\right) \\ &= p\left(x^{(1)} \mid \theta\right) \quad p\left(x^{(2)} \mid \theta\right) \quad p\left(x^{(3)} \mid \theta\right) \quad p\left(x^{(4)} \mid \theta\right) \\ &= \theta \quad (1-\theta) \quad \theta \quad \theta \\ &= \theta^3(1-\theta) \end{aligned}$$

- More generally, we can write

$$\begin{aligned} p(\mathbf{X} \mid \theta) &= \theta^{\sum_{i=1}^n x_i} (1-\theta)^{\sum_{i=1}^n (1-x_i)} \\ &= \theta^{\sum_{i=1}^n \mathbb{1}(x_i=1)} (1-\theta)^{\sum_{i=1}^n \mathbb{1}(x_i=0)} \\ &= \theta^{n_1} (1-\theta)^{n_0} \end{aligned}$$

## Inference task: computing dataset probabilities

```
n_1 = 0
n_0 = 0
for i in range(n):
    if X[i] == 1:
        n_1 += 1
    else: # binary data
        n_0 += 1
p = theta ** n_1 * (1 - theta) ** n_0
```

Better version:

```
n_1 = X.sum()
n_0 = X.shape[0] - n_1
log_p = n_1 * np.log(theta) \
        + n_0 * np.log1p(-theta)
```

- Computational complexity (of either):  $\mathcal{O}(n)$ 
  - Look at each element once, doing a single addition each time, then a constant number of operations for final value
- Operating in “log space” is very practically helpful:
  - If  $n$  is huge and/or  $\theta$  is very close to 0 or 1, the probability is tiny
  - Calculation might underflow and return zero / be very inaccurate
  - Logarithms give you much bigger range of effective floating point computation
  - $\text{np.log1p}(t)$  is  $\log(1 + t)$ , but floats are much more accurate near 0 than 1!

## Inference task: finding the mode (“decoding”)

- **Inference task:** given  $\theta$ , find the  $x$  that maximizes  $p(x \mid \theta)$ 
  - “What’s most likely to happen?”
- For Bernoulli models:
- If  $\theta < 0.5$ , the mode is  $x = 0$ 
  - If  $\theta = 0.03$ , it’s more likely that a random person **does not** have COVID-19
- If  $\theta > 0.5$ , the mode is  $x = 1$ 
  - If  $\theta = 0.6$ , it’s more likely that a random person **does** have COVID-19 (uh-oh)
- If  $\theta = 0.5$ , both  $x = 0$  and  $x = 1$  are valid modes
- This process isn’t very exciting for Bernoulli models
  - For more complex models, it can be pretty hard (and important)
  - We’ll see later that **classification can be viewed as finding a (conditional) mode**

## Inference task: finding the most likely dataset

- **Inference task:** given  $\theta$ , find the  $\mathbf{X}$  that maximizes  $p(\mathbf{X} \mid \theta)$ 
  - “What set of training example are we most likely to observe?”
- Recall for Bernoullis,  $p(\mathbf{X} \mid \theta) = \theta^{n_1}(1 - \theta)^{n_0}$
- If  $\theta < 0.5$ , the most likely dataset is  $\mathbf{X} = (0, 0, 0, 0, \dots)$ 
  - $p(\mathbf{X} \mid \theta)$  is maximized if  $n_0$  is as big as possible, and  $n_1$  small
  - If  $\theta = 0.3$ , the “most likely” sample has zero positives!
- **The modal dataset almost never represents “typical” behaviour**
  - If  $\theta = 0.3$ , we expect about 30% of samples to be 1, not 0%!
  - The modal  $\mathbf{X}$  has the highest probability, but **that probability might be really low**
    - There are many datasets with some 1s in them
    - *Each one* is lower-probability than the (single) all-zero dataset
    - As a whole they’re overwhelmingly more likely

## Inference task: sampling

- Inference task: given  $\theta$ , generate  $X$  according to  $p(X | \theta)$ 
  - Called sampling from the distribution
- Sampling is the “opposite” of density estimation:

$$\Pr(X = 1) = 0.4 \quad \xrightarrow{\text{sampling}} \quad \mathbf{X} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

- Given the model, your job is to generate IID examples
- Often write code to generate one sample, and call it many times

# Why sample?

bonus!

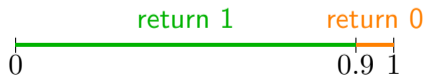
- Sampling isn't especially interesting for Bernoulli distributions
  - Knowing  $\theta$  tells you everything about the distribution
- But sampling will let us do neat things in more-complicated density models:
  - `thispersondoesnotexist.com`, DALL-E, ChatGPT, ...



- Sampling often helps us check whether the model is reasonable
  - If samples look nothing like the data, the model isn't very good

## Inference task: sampling

- Basic ingredient of typical sampling methods:
- We **assume we can sample uniformly on  $[0, 1]$**
- In practice, we use a “pseudo-random” number generator
  - `rng = np.random.default_rng(); t = rng.random()`
  - We won't talk about how this works; see CPSC 436R / [Nick's book](#)
- Consider sampling from  $\text{Bern}(0.9)$ 
  - 90% of the time, we should produce a 1
  - 10% of the time, we should produce a 0
- How can we do that with a sample from  $U \sim \text{Unif}([0, 1])$ ?
  - If  $U \leq 0.9$ , return 1; otherwise, return 0.



## Inference task: sampling

- Sampling from  $\text{Bern}(\theta)$ :
  - Generate  $U \sim \text{Unif}([0, 1])$ . If  $U \leq \theta$ , return 1; otherwise, return 0

```
u = rng.random()
```

```
if u <= theta:
```

```
    x = 1
```

```
else:
```

```
    x = 0
```

or

or

- Assuming the uniform RNG costs  $\mathcal{O}(1)$ , generates a single sample in  $\mathcal{O}(1)$  time
- To generate  $t$  samples, nothing smarter to do than just call it  $t$  times;  $\mathcal{O}(t)$  cost

# Summary

- **Binary density estimation**: models  $\Pr(X = 1)$  given iid samples  $x^{(1)}, \dots, x^{(n)}$
  - **Bernoulli distribution** over binary variables
    - **Parameterized** by  $\theta \in [0, 1]$  with  $\Pr(X = 1 \mid \theta) = \theta$
  - **Inference**: computing things from models, like **finding modes** and **sampling**
- 
- Next time: the exciting world of priors