Categorical variables and Monte Carlo

CPSC 440/550: Advanced Machine Learning

`cs.ubc.ca/~dsuth/440/23w2`

University of British Columbia, on unceded Musqueam land

2023-24 Winter Term 2 (Jan–Apr 2024)

Admin

admin

- Quiz 1: ongoing. Let me know if there are issues
- Quiz 2: next week! Make sure to reserve
  - Covering lectures 5 through today, possibly some stuff from 8 (Monday) but lightly

- A1 solutions up (see Piazza/Canvas)
- Grading for A1 ongoing, probably out in about a week (sorry for delay)
- Grading for the short-answer parts of Q1 also probably by late next week

- Possible transit strike extension, Saturday – Monday
- I'll be here, and try to set up a livestream Monday (Panopto-but-live or Zoom)
- Shouldn't significantly affect Q2 plans, but keep an eye on your email

Outline

Motivating problem: political polling

- Want to know support for political parties among a voter group
  - Helps candidates/parties target campaigning, etc
- Where I live, the last election results:
  - ~~40.4%~~ 23.0% Liberal
  - ~~30.7%~~ 17.5% NDP
  - ~~21.6%~~ 12.31% Conservative
  - ~~3.9%~~ 2.2% Green
  - ~~3.2%~~ 1.8% PPC
  - 43% no vote
- We want to estimate these quantities based on a sample (a poll)

General problem: categorical density estimation

- Special case of density estimation with a categorical variable:
  - Input: n iid samples of categorical values $x^{(1)}, x^{(2)}, \dots, x^{(n)} \in \{1, 2, \dots, k\}$
  - Output: a probability model for $\Pr(X = 1)$, $\Pr(X = 2)$, ..., $\Pr(X = k)$
- We'll remember, but not usually write down, that 1 = `Lib`, 2 = `NDP`, ...
- As a picture: $\mathbf{X} \in \mathbb{R}^{n \times 1}$ contains our sample data

  X is a random variable over $\{1, 2, \dots, k\}$ from the distribution

$$\mathbf{X} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \\ 3 \end{bmatrix} \xrightarrow{\text{density estimator}} \begin{array}{l} \Pr(X = 1) = 0.4 \\ \Pr(X = 2) = 0.2 \\ \Pr(X = 3) = 0.4 \end{array}$$

- We'll start by revisiting previous concepts, but introduce some more

Other applications of categorical density estimation

- Some other questions we might ask:
  1. What portion of my customers use cash, credit, debit?
  2. What's the probability that a random patient will be able to receive this type of blood?
  3. How many random tweets should I expect to look at before I see this particular word?

- For categorical variables, we do not assume an ordering
  - Category 4 isn't "closer" to category 3 than it is to category 1

Ordinal variables

bonus!

- Ordered categorical variables are called ordinal
  - Results of rolling dice, if you're trying to beat a specific number
  - Survey results ("strongly disagree," "disagree," "neutral," ...)
  - Ratings (1 star, 2 stars, ...)
  - Tumour severity (Grade I, ..., Grade IV)

- We won't cover these for now, but lots of methods exist
- "Ordinal logistic regression": a loss function where "2 stars" is closer to "3 stars" than "4 stars"
  - But there might be a bigger "gap" between 2 and 3 stars than between 3 and 4
- Can use this "ordinal loss" in neural nets

Parametrizing categorical probabilities

- We typically use the categorical distribution (aka "multinoulli" (ugh))
- For k categories, have k parameters, $\theta_1, \ldots, \theta_k \geq 0$

$$\Pr(X = 1 \mid \theta_1, \ldots, \theta_k) = \theta_1 \ldots \Pr(X = k \mid \theta_1, \ldots, \theta_k) = \theta_k$$

- Categories are mutually exclusive: can only pick one
  - Require that $\sum_{c=1}^{k} \theta_c = 1$
- More succinctly: if $X \sim \mathrm{Cat}(\boldsymbol{\theta})$ with $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)$,

$$p(x \mid \boldsymbol{\theta}) = \theta_1^{\mathbb{1}(x=1)} \theta_2^{\mathbb{1}(x=2)} \cdots \theta_k^{\mathbb{1}(x=k)}$$

Inference task: union

- Inference task: given $\boldsymbol{\theta}$, compute probability of unions
- For example: $\Pr(X = \texttt{Lib} \cup X = \texttt{NDP} \mid \boldsymbol{\theta})$

- Can't be both, so: $\Pr(X = 2 \cup X = 4 \mid \boldsymbol{\theta}) = \theta_2 + \theta_4$

- Variation: $\Pr(X \leq c)$ for some c is $\theta_1 + \theta_2 + \cdots + \theta_c$
- Why do we care, since the categories are unordered?
- $F(c) = \Pr(X \leq c)$ is the cumulative distribution function (cdf)
  - Depends on (arbitrary) ordering, but very useful function as we'll see soon!

Inference task: mode (decoding)

- Inference task: given $\boldsymbol{\theta}$, find the mode, $\arg\max_x p(x \mid \boldsymbol{\theta})$
  - "Who's going to win the election?"

- Also very easy: $\arg\max_c \theta_c$

Inference task: likelihood

- Inference task: given and data $\mathbf{X}$, find $p(\mathbf{X} \mid \boldsymbol{\theta})$
- Assuming data is iid from $\mathrm{Cat}(\boldsymbol{\theta})$,

$$
\begin{aligned}
p(\mathbf{X} \mid \boldsymbol{\theta}) = p(x^{(1)}, \ldots, x^{(n)} \mid \boldsymbol{\theta}) &= \prod_{i=1}^{n} p(x^{(i)} \mid \boldsymbol{\theta}) \\
&= \prod_{i=1}^{n} \theta_1^{\mathbb{1}(x^{(i)}=1)} \theta_2^{\mathbb{1}(x^{(i)}=2)} \cdots \theta_k^{\mathbb{1}(x^{(i)}=k)} \\
&= \theta_1^{\sum_{i=1}^{n} \mathbb{1}(x^{(i)}=1)} \theta_2^{\sum_{i=1}^{n} \mathbb{1}(x^{(i)}=2)} \cdots \theta_k^{\sum_{i=1}^{n} \mathbb{1}(x^{(i)}=k)} \\
&= \theta_1^{n_1} \theta_2^{n_2} \cdots \theta_k^{n_k}
\end{aligned}
$$

- ...defining at the end n_c as the number of cs in $\mathbf{X}$, like n_0/n_1 for binary data
- Like Bernoulli, the likelihood only depends on the counts

Code for categorical likelihood

```
counts = np.zeros(k)
for x in X:
    count[x] += 1
p = 1
for theta_c, n_c in zip(theta, counts):
    p *= theta_c ** n_c
```

Better version:

```
counts = np.bincount(X,
↪   minlength=k)
log_p = counts @ np.log(theta)
```

- Computation complexity (either way) is $\mathcal{O}(n + k)$
  - Usual case: $n \gg k$ (many samples, few categories), this is just $\mathcal{O}(n)$
  - If $k \gg n$, could also easily get $\mathcal{O}(n)$ by only tracking categories with nonzero counts

Inference task: sampling

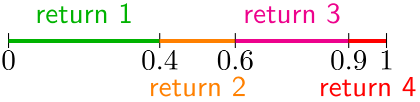
- Inference task: given $\boldsymbol{\theta}$, generate samples from $X \sim \text{Cat}(\boldsymbol{\theta})$

$$\begin{aligned}\Pr(X = 1) &= 0.4 \\ \Pr(X = 2) &= 0.2 \\ \Pr(X = 3) &= 0.4\end{aligned} \quad \xrightarrow{\text{sampling}} \quad \mathbf{X} = \begin{bmatrix} 1 \\ 3 \\ 3 \end{bmatrix}$$

- Notice: not sampling "one value per class"; each sample is in one category
  - Who will this voter (say they'll) vote for?

Categorical sampling algorithm

- Will use a uniform sample from $[0,1]$ to construct a sample from $\mathrm{Cat}(\boldsymbol{\theta})$
- Example: sample from $\boldsymbol{\theta} = \big(0.4, 0.2, 0.3, 0.1\big)$ based on a single $u \sim \mathrm{Unif}([0,1])$
  - Want $X = 1$ 40% of the time: if $u < 0.4$, return 1
  - Want $X = 2$ 20% of the time: if $0.4 \leq u < 0.6$, return 2
  - Want $X = 3$ 30% of the time: if $0.6 \leq u < 0.9$, return 3
  - Want $X = 4$ 10% of the time: if $0.9 \leq u$, return 4



- Use CDF, $\Pr(X \leq c) = \theta_1 + \cdots + \theta_c$:
  - Generate $u \sim \mathrm{Unif}([0,1])$
  - if $u \leq \Pr(X \leq 1)$, return 1
  - else if $u \leq \Pr(X \leq 2)$, return 2
  - ...
  - else return k

- Computing $\Pr(X \leq c)$ from $\boldsymbol{\theta}$ costs $\mathcal{O}(k)$
  - $\mathcal{O}(k^2)$ total time... but can precompute!

```
cdf = np.cumsum(theta)
u = rng.random_sample(n_to_samp)
samp = cdf.searchsorted(u, side='right')
```

- Takes $\mathcal{O}(k)$ upfront, $\mathcal{O}(\log k)$ per sample

Faster categorical sampling algorithms

bonus!

- Previous method is sometimes called "roulette wheel sampling"
  - $\mathcal{O}(k)$ preprocessing (computing the CDF), $\mathcal{O}(\log k)$ time per sample

- "Vose's alias method": $\mathcal{O}(k)$ preprocessing but only $\mathcal{O}(1)$ time per sample

- Really nice (long) article developing many variations:
  Darts, Dice, and Coins: Sampling from a Discrete Distribution by Keith Schwarz

Outline

Motivation: probabilistic inference

- Given a general model, we often want to make inferences
  - Marginals: what's the probability that $X_i = c$?
  - Conditionals: what's the probability that $X_i = c$, given that $X_{i'} = c'$?

- This has been simple for the models we've seen so far
  - For Bernoulli/categorical, computing probabilities is straightforward
  - For product of Bernoullis (or categoricals), assumed everything is independent

- For many models, inference has no closed form or might be NP-hard
- In these cases, we'll often use Monte Carlo approximations

Monte Carlo: marginalization by sampling

- A basic Monte Carlo method for estimating probabilities of events:
- Step 1: Generate a lot of samples $x^{(i)}$ from our model

$$X = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

- Step 2: Count how often the event occurred in the samples

$$\Pr(X_2 = 1) \approx \frac{3}{4} \qquad \Pr(X_3 = 0) \approx 0$$

- This very simple idea is one of the most important algorithms in ML/statistics
- Modern versions developed to build better nuclear weapons :/
  - "Sample" from a physics simulator, see how often it leads to a chain reaction

Monte Carlo to approximate probabilities

- Monte Carlo estimate of the probability of an event A:

$$\frac{\text{number of samples where } A \text{ happened}}{\text{number of samples}} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(A \text{ happened in } x^{(i)})$$

- You can think of this as the MLE of a binary variable $\mathbb{1}(A \text{ happened})$
- Approximating probability of a pair of independent dice adding to 7:
  - Roll two dice, check if they add to 7
  - Roll two dice, check if they add to 7
  - Roll two dice, check if they add to 7
  - Roll two dice, check if they add to 7
  - Roll two dice, check if they add to 7
  - Roll two dice, check if they add to 7
  - ...
  - Monte Carlo estimate: fraction where they add to 7

Monte Carlo to approximate probabilities

- Recall the motivating problem of modeling (Lib, CPC, NDP, GRN, PPC)
- From 100 samples, what's the probability that $n_{\texttt{Lib}} > \max(n_{\texttt{CPC}}, n_{\texttt{NDP}}, \ldots)$?

- Can answer this in closed form with math . . . or think less and do Monte Carlo
  - Generate 100 samples, check who won
  - Generate 100 samples, check who won
  - . . .
  - Approximate probability by fraction of times they won

- Another example: probability that $\mathrm{Beta}(\alpha, \beta)$ is above 0.7

Monte Carlo to estimate the mean

- A Monte Carlo estimate for the mean: the mean of the samples

$$\mathbb{E}[X] \approx \frac{1}{n}\sum_{i=1}^{n} x^{(i)}$$

- A Monte Carlo approximation of the expected value of X^2:

$$\mathbb{E}[X^2] \approx \frac{1}{n}\sum_{i=1}^{n} \left(x^{(i)}\right)^2$$

- A Monte Carlo approximation of the expected value of $g(X)$:

$$\mathbb{E}[g(X)] \approx \frac{1}{n}\sum_{i=1}^{n} g\left(x^{(i)}\right) \qquad \mathbb{E}[g(X)] = \sum_{x\in\mathcal{X}} p(x)g(x) \text{ or } \int_{x\in\mathcal{X}} p(x)g(x)\mathrm{d}x$$

  - Most general form: $g(x) = x$, $g(x) = x^2$, $g(x) = \mathbb{1}(A \text{ happens on } x)$

$$\mathbb{E}[\mathbb{1}(A \text{ happens on } x)] = \int_{x\in\mathcal{X}} p(x)\,\mathbb{1}(A \text{ happens on } x)\mathrm{d}x = \int_{x:A \text{ happens}} p(x)\mathrm{d}x = \Pr(A)$$

Monte Carlo: theory

- Let $\mu = \mathbb{E}[g(X)]$ be the value we want to compute
- Assume $\sigma^2 = \mathrm{Var}[g(X)]$ exists and is bounded ("not infinite")

- With iid samples, Monte Carlo gives an unbiased estimate of μ
  - Expected value of the Monte Carlo estimate, over samples we might draw, is exactly μ

- Monte Carlo estimate "converges to μ" as $n \to \infty$
  - Estimate gets arbitrarily close to μ as n increases: (strong) law of large numbers

- Expected squared error is exactly $\mathbb{E}(\hat{\mu} - \mu)^2 = \frac{\sigma^2}{n}$
- $\hat{\mu}$ is *approximately* normal with mean μ and variance $\frac{\sigma^2}{n}$ (central limit theorem)
- Can be viewed as a special case of SGD