CPSC 440: Machine Learning

Monte Carlo Approximation

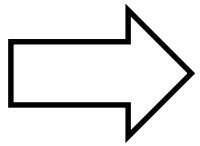
Winter 2022

Last Time: Categoraical Density Estimation

- We considered density estimation with a categorical variable:
  - Input: 'n' IID samples of categorical values x^1, x^2, x^3,…, x^n from population.
  - Output: model of probability that x=1, x=2, x=3,…,x=k.
- Categorical density estimation as a picture:

X =

Party?
LIB
CPC
NDP
LIB
GRN



p(x = LIB) = 0.34, p(x=NDP) = 0.34,
p(x = CPC) = 0.27, p(x=GRN) = 0.03,
p(x = PPC) = 0.02.

- Example: polling for which political party people will support.
- For categorical variables, we do not assume there is an ordering.

Parameterization of Categorical Probabilities

- We typically parameterize using the categorical distribution:
  - Sometimes called "multinoulli".
  - Has parameters $\theta_1, \theta_2, \ldots, \theta_k$ when we have 'k' categories.
  - Defines probabilities using:

$$p(x=1 \mid \theta_1, \theta_2, \ldots, \theta_k) = \theta_1 \quad p(x=2 \mid \theta_1, \theta_2, \ldots, \theta_k) = \theta_2 \quad \cdots \quad p(x=k \mid \theta_1, \theta_2, \ldots, \theta_k) = \theta_k$$

  - Because probabilities sum to 1, we require: $\sum_{c=1}^{K} \theta_c = 1$

- One way to write this for a generic 'x':

$$p(x \mid \Theta) = \theta_1^{\mathbb{I}(x=1)} \theta_2^{\mathbb{I}(x=2)} \cdots \theta_K^{\mathbb{I}(x=k)}$$

$$\Theta = \{\theta_1, \theta_2, \ldots, \theta_k\}$$

Inference Task: Union

- Inference task : given θ, compute probabilities of unions:
  - For example, p(x = LIB ∪ x = NDP | Θ).
    - What fraction of votes would these parties their supporters voted together?

- We assume the categories are mutually exclusive.
  - "You can only pick one."
  - This allows us to compute unions with addition:
    - p(x = 2 ∪ x = 3 ∪ x = 4 | Θ) = $\theta_2 + \theta_3 + \theta_4$.

- A variation on this task is computing p(x ≤ c) for some value 'c'.
  - Easy to do: p(x ≤ 4) = $\theta_1 + \theta_2 + \theta_3 + \theta_4$.
  - We often want to do this even though the categories are unordered.
  - We call p(x ≤ c) the cumulative distribution function (CDF).

Inference Task: Decoding

- Inference task: given θ, find 'x' that maximizes p(x | Θ) (decoding).

- Probably the most relevant inference for the elections example:
  - The decoding is "who wins the election".

- Computing the decoding using "argmax" notation:

$$x_{decode} \in \underset{c}{argmax}\{ p(x = c \mid \theta) \}$$
$$\equiv \underset{c}{argmax}\{ \theta_c \}$$

  - So the decoding is the category 'c' where θ_c is the largest.

Inference Task: Computing Dataset Probabilities

- Inference task : given Θ and IID data, compute $p(x^1, x^2, \ldots, x^n \mid \Theta)$.
  - The likelihood of training/validation/testing data.

- Assuming "independence of IID data given parameters", we have:

$$p(x^1, x^2, \ldots, x^n \mid \Theta) = \prod_{i=1}^{n} p(x^i \mid \Theta) \qquad \text{(this is the conditional independence assumption)}$$

$$= \prod_{i=1}^{n} \theta_1^{I(x^i=1)} \theta_2^{I(x^i=2)} \cdots \theta_k^{I(x^i=k)} \qquad \text{(definition of categorical distribution)}$$

$$= \theta_1^{\sum_{i=1}^{n} I(x^i=1)} \theta_2^{\sum_{i=1}^{n} I(x^i=2)} \cdots \theta_k^{\sum_{i=1}^{n} I(x^i=k)} \qquad (\theta_1 \theta_1 \theta_1 \theta_2 \theta_1 \theta_2 = \theta_1^4 \theta_2^2)$$

$$= \theta_1^{n_1} \theta_2^{n_2} \cdots \theta_k^{n_k} \qquad \left(n_c = \sum_{i=1}^{n} I(x^i = c)\right)$$

- Where n_1 is "number of 1s", n_2 is "number 2s", and so on.
  - Similar to the Bernoulli, the likelihood only depends on the counts.

Code for Categorical Likelihood

- We just showed that the categorical likelihood can be written:

$$p(x^1, x^2, \ldots, x^n \mid \theta) = \theta_1^{n_1} \theta_2^{n_2} \cdots \theta_k^{n_k}$$

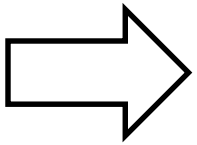
```
nc = zeros(k)
for i in 1:n
    nc[X[k]] += 1
end
logLik = 0
for c in 1:k
    logLik += nc[c] * log(theta[c])
end
```

- Will be very small for large 'n'.
  - Compute the log-likelihood in practice.
- Runtime: O(n + k).
  - If n >> k (many samples, few categories), this is O(n).
  - If k >> n (many categories, few samples), you could also get O(n).
    - By only tracking/summing over the classes with non-zero counts.

Inference Task: Sampling

- Inference task: given Θ,
  generate samples of ‘x’ distributed according to $p(x \mid \Theta)$.

$p(x = \text{LIB}) = 0.34$, $p(x=\text{NDP}) = 0.34$,
$p(x = \text{CPC}) = 0.27$, $p(x=\text{GRN}) = 0.03$,
$p(x = \text{PPC}) = 0.02$.



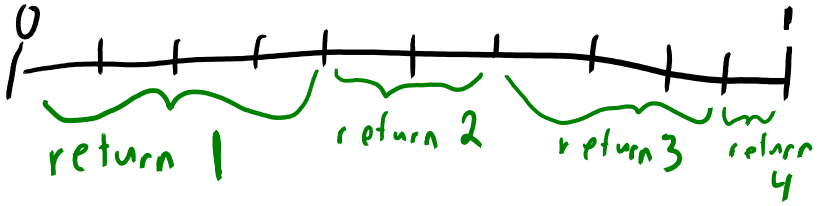
Party?
LIB
CPC
NDP
LIB
GRN

- Notice that we are not “sampling a value for each of the classes”.
  - Each sample will belong to one category.
    - 34% of the samples should be LIB, 27% will be CPC, and so on.

Inference Task: Sampling

- Recall we assume we can sample uniformly between 0 and 1.
  - And we want to turn this into a sample over the 'k' categories.

- Sampling from categorical distribution with $\Theta = \{0.4, 0.2, 0.3, 0.1\}$:
  - Generate a uniform sample 'u'.
  - If u < 0.4, return 1 (like sampling from a Bernoulli with $\theta = 0.4$).
  - If u > 0.4 but less than 0.6, return 2 (like sampling Bernoulli with $\theta = 0.2$).
  - If u > 0.6 but less than 0.9, return 3 (like sampling Bernoulli with $\theta = 0.3$).
  - If u > 0.9, return 0 (like sampling Bernoulli with θ = 0.1).



Inference Task: Sampling

- Formally, the sampler "returns the 'c' such that $p(x \leq c-1) < u < p(x \leq c)$".
  - Where the CDF for categorical is $p(x \leq c) = \theta_1 + \theta_2 + \cdots + \theta_c$.

- Sampling from a categorical distribution with 'k' categories:
  1. Generate 'u' uniformly on the interval between 0 and 1.
  2. If $u \leq p(x \leq 1)$, return 1.
  3. If $u \leq p(x \leq 2)$, return 2.
  4. If $u \leq p(x \leq 3)$, return 3.
  5. …

- Runtime:
  - If you compute $p(x \leq c)$ from scratch at each step, costs $O(k^2)$.
  - If you use that $p(x \leq c) = p(x \leq c-1) + \theta\_c$, costs $O(k)$.

Inference Task: Sampling

- In code:

```
u = rand()
CDF = 0
For c in 1:k
    CDF += theta[c]
    if u < CDF
        return c
    end
end
```

```julia
### For vector p giving discrete probabilities, generates a random sample
function sampleDiscrete(p)
    findfirst(cumsum(p[:]).> rand())
end
```

- Calling this function each time costs O(k).

- You can go faster if you have CDF values stored.
  - In this case, do a binary search for the 'c' such that $p(x \leq c-1) < u < p(x \leq c)$.
- Using this faster procedure, it costs O(k + t log k) to generate 't' samples.
  - O(k) to compute all the CDFs.
  - O(log k) to do a binary search to generate each sample.

Next Topic: Monte Carlo Approximation

Motivation: Probabilistic Inference

- Given a probabilistic model, we often want to make inferences:
  - Marginals: what is the probability that $x_j = c$?
  - Conditionals: what is the probability that $x_j = c$ given $x_{j'} = c'$?

- This is simple for the models we have seen so far.
  - For Bernoulli/categorical, computing probabilities is straightforward.
  - For multivariate models, we assumed everything was independent.
    - Perhaps conditioned on a label or the final hidden layer of a neural network.

- For many models, inference has no closed form or is NP-hard.
  - For these problems, we often use Monte Carlo approximations.

Monte Carlo: Marginalization by Sampling

- A basic Monte Carlo method for estimating probabilities of events:
  - Generate a large number of samples x^i from the model:

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

  - Compute frequency that the event happened in the samples:

$$p(x_2 = 1) \approx 3/4$$
$$p(x_3 = 0) \approx 0/4$$

- Monte Carlo methods are the second most important type of ML algorithms.
  - Originally developed to build better atomic bombs ☹
    - Runs physics simulator to "sample", then see if it leads to a chain reaction.

Monte Carlo for Approximating Probabilities

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

$$\frac{\text{number of samples where } A \text{ happened}}{\text{number of samples}}$$

- You can think of this as the MLE for a binary variable:
  - The binary variable is '1' for samples where 'A' happened, '0' otherwise.
- Approximating probability of a pair of independent dice rolling a sum of 7:
  - Roll two dice, check if the sum is 7.
  - Roll two dice, check if the sum is 7.
  - Roll two dice, check if the sum is 7.
  - Roll two dice, check if the sum is 7.
  - Roll two dice, check if the sum is 7.
  - …
  - Monte Carlo estimate: fraction of samples where sum is 7.

Monte Carlo for Approximating Probabilities

- Recall our motivating problem:
  - Building a model of voters among categories (LIB, CPC, NDP, GRN, PPC).
- You might consider the following inference problem:
  - In 100 samples, what the probability that $n_{LIB} > \max\{n_{CPC}, n_{NDP}, n_{GRN}, n_{PPC}\}$?
    - "What is the probability that LIBs win the election again?"

- You can do some math to figure out the answer, or do Monte Carlo:
  - Generate 100 samples, check who won.
  - Generate 100 samples, check who won.
  - …
  - Approximate probability by fraction of times they won.

Monte Carlo for Inequalities

- Consider probability that a variable is above a threshold.
  - Probability that a beta variable is above 0.7.
  - Probability that a standard normal variable is above -1.2.
- Monte Carlo estimate for $p(x \leq \tau)$ for some threshold τ:
  - Fraction of samples that are above the threshold.

Monte Carlo Method for the Mean

- A Monte Carlo approximation of the mean:
  - Approximate the mean by the mean of the samples.

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

- A Monte Carlo approximation of expected value of x^2:
  - Approximate $\mathbb{E}[x^2]$ by the average value of x^2 on the samples.

- A Monte Carlo approximation of the expected value of function ‘g’.
  - Approximate $\mathbb{E}[g(x)]$ by the average value of g(x) on the samples.

Monte Carlo Method: Definition

- Monte Carlo approximates expectation of random functions:

$$\underbrace{\mathbb{E}[g(x)] = \sum_{x \in \mathcal{X}} g(x)p(x)}_{\text{discrete } x} \quad \text{or} \quad \underbrace{\mathbb{E}[g(x)] = \int_{x \in \mathcal{X}} g(x)p(x)dx}_{\text{continuous } x}$$

- Computing mean is a special case, if we use g(x) = x.
- Computing probability of an event 'A' is also a special case:
  - Set g(x) = I["A happened in sample x"], indicator function for event 'A'.

- Monte Carlo methods generate 'n' samples x^i from p(x) and then use:

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

Summary of Monte Carlo Theory

- Let $\mu = \mathbb{E}[g(x)]$, the value we want to compute.
  - And assume variance of the samples σ^2 is bounded ("not infinite").
- With IID samples, Monte Carlo gives an unbiased approximation of μ.
  - Expected value of Monte Carlo estimate, averaged over samplings, is μ.
- Monte Carlo estimate "converges" to μ as sample size 'n' goes to ∞.
  - Estimate get arbitrarily close to μ as your number of samples gets large.
- Expected squared error between estimate and μ is σ^2/n with 'n' samples.
  - This is the speed at which you converge to μ as you increase 'n'.
- Monte Carlo can be written as a special case of SGD.
  - See the post-lecture bonus slides for some details on all of the above.

Summary

- Categorical distribution:
  - Probability over unordered categories, where p(x = c | Θ) = θ_c.
- Inference in categorical models:
  - CDF, decoding, likelihood, sampling.
- Monte Carlo methods:
  - Approximate expectations of random functions.
  - Generate a set of IID samples.
  - Take average value of function value applied to each sample.

- Next time: why we are doing everything wrong to make decisions.

Law of the Unconscious Statistician

- We use these identities to define the expectation of a function g applied to a random variable x,

$$\underbrace{\mathbb{E}[g(x)] = \sum_{x \in \mathcal{X}} g(x)p(x)}_{\text{discrete } x} \quad \text{or} \quad \underbrace{\mathbb{E}[g(x)] = \int_{x \in \mathcal{X}} g(x)p(x)dx}_{\text{continuous } x}.$$

- The transformation from expectation to sum/integral is known as the "law of the unconsciuos statistician".
  - It's usually taken as being true, but it's proof is a bit of a pain.

Unbiasedness of Monte Carlo Methods

- Let $\mu = \mathbb{E}[g(x)]$ be the value we want to approximate (not necessarily mean).

- The Monte Carlo estimate is an unbiased approximation of μ,

$$\begin{aligned}
\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} g(x^i)\right] &= \frac{1}{n}\mathbb{E}\left[\sum_{i=1}^{n} g(x^i)\right] && \text{(linearity of } \mathbb{E}\text{)} \\
&= \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}[g(x^i)] && \text{(linearity of } \mathbb{E}\text{)} \\
&= \frac{1}{n}\sum_{i=1}^{n} \mu && (x^i \text{ is IID with mean } \mu) \\
&= \mu.
\end{aligned}$$

- The law of large numbers says that:
  - Unbiased approximators "converge" (probabilistically) to expectation as $n \to \infty$.
  - So the more samples you get, the closer to the true value you expect to get.

Rate of Convergence of Monte Carlo Methods

- Let f be the squared error in a 1D Monte Carlo approximation,

$$f(x^1, x^2, \ldots, x^n) = \left(\frac{1}{n} \sum_{i=1}^{n} g(x^i) - \mu \right)^2 .$$

- If variance is bounded, error with n samples is $O(1/n)$,

$$\begin{aligned}
\mathbb{E}\left[\left(\frac{1}{n} \sum_{i=1}^{n} g(x^i) - \mu \right)^2 \right] &= \mathsf{Var}\left[\frac{1}{n} \sum_{i=1}^{n} g(x^i) \right] && \text{(unbiased and def'n of variance)} \\
&= \frac{1}{n^2} \mathsf{Var}\left[\sum_{i=1}^{n} g(x^i) \right] && (\mathsf{Var}(\alpha x) = \alpha^2 \mathsf{Var}(x)) \\
&= \frac{1}{n^2} \sum_{i=1}^{n} \mathsf{Var}[g(x^i)] && \text{(IID)} \\
&= \frac{1}{n^2} \sum_{i=1}^{n} \sigma^2 = \frac{\sigma^2}{n}. && (x^i \text{ is IID with var } \sigma^2)
\end{aligned}$$

- Similar $O(1/n)$ argument holds for $d > 1$ (notice that faster for small σ^2).

Monte Carlo as a Stochastic Gradient Method

- Monte Carlo approximation as a stochastic gradient method with $\alpha_i = 1/(i+1)$,

$$
\begin{aligned}
w^n &= w^{n-1} - \alpha_{n-1}(w^{n-1} - x^i) \\
&= (1 - \alpha_{n-1})w^{n-1} + \alpha_{n-1}x^i \\
&= \frac{n-1}{n}w^{n-1} + \frac{1}{n}x^i \\
&= \frac{n-1}{n}\left(\frac{n-2}{n-1}w^{n-2} + \frac{1}{n-1}x^{i-1}\right) + \frac{1}{n}x^i \\
&= \frac{n-2}{n}w^{n-2} + \frac{1}{n}\left(x^{i-1} + x^i\right) \\
&= \frac{n-3}{n}w^{n-3} + \frac{1}{n}\left(x^{i-2} + x^{i-1} + x^i\right) \\
&= \frac{1}{n}\sum_{i=1}^{n} x^i.
\end{aligned}
$$