Monte Carlo Estimation

CMPUT 296: Basics of Machine Learning

Midterm Results

- The **midterm average** was **93 / 130** (**72%**)
  - Highest grade was 122 / 130 (94%)
- The TAs and I will be contacting some of you tomorrow to schedule **spot checks**

Recap: Bayesian Estimation

Bayes' Rule: $p(\theta \mid \mathscr{D}) = \dfrac{p(\mathscr{D} \mid \theta)p(\theta)}{p(\mathscr{D})} = \dfrac{p(\mathscr{D} \mid \theta)p(\theta)}{\int_\Theta p(\mathscr{D} \mid \theta)p(\theta)\, d\theta}$

- The MAP, MLE, and Bayes estimators for a model parameter are all **point estimates**
- MAP and MLE can be computed without computing $p(\mathscr{D})$
- **Conjugate priors** make it possible to perform Bayesian updates analytically
  - But many models don't have conjugate priors

Lecture Outline

1. Recap & Logistics
2. Hierarchical Bayesian Models
3. Estimation via Sampling
4. Sampling from Hard-to-Sample Distributions

Hierarchical Models

- Common approach in probabilistic modeling: **hierarchical models**
- Each individual acts according to a unique **individual distribution**
- The individual distributions' parameters are drawn from a single, **shared distribution**
- The shared distribution's parameter has a **prior**
- Two kinds of estimation:
    1. What is the parameter of the **shared distribution**?
    2. What is the parameter for a **given individual**?

Question: Is this kind of model useful even if we only care about question 2?

Hierarchical Models
Example: Modeling Game Play

- We have m games $G_1, \ldots, G_m$, and n players
- Each player i has a "level" k_i that represents their strategic reasoning ability
- Each player i has taken an action a_{ij} in each game G_j according to a distribution calculated by $f(a_{ij}, G_j, k_i)$ (where f is a complicated function)
- Levels are distributed according to a Poisson(λ) distribution with unknown parameter λ
- Our prior on λ is uniform between 0 and 10

Hierarchical Game Play Model

$$\mathscr{D} = \{(a_{ij}, G_j) \mid 1 \leq i \leq n, 1 \leq j \leq m\}$$

$$\lambda \sim \text{Uniform}[0,10] \quad k_i \sim \text{Poisson}(\lambda) \quad p(a_{ij} \mid G_j) = f(a_{ij}, G_j, k_i)$$

$$p(\mathscr{D} \mid \lambda, k_1, \ldots, k_n) = \prod_{i=1}^{n} \prod_{j=1}^{m} p(a_{ij} \mid G_j, k_i) = \prod_{i=1}^{n} \prod_{j=1}^{m} f(a_{ij}, G_j, k_i)$$

$$p(k_1, \ldots, k_n \mid \lambda) = \prod_{i=1}^{n} p(k_i \mid \lambda) = \prod_{i=1}^{n} \frac{\lambda^{k_i} e^{-\lambda}}{k_i!}$$

$$p(\lambda) = I_{0 \leq \lambda \leq 10}[\lambda] \frac{1}{10}$$

Question: How can we find the **posterior distribution over k_i**?

Posterior Distribution of k_i

Bayes' rule

$$p(\lambda, k_1, \ldots, k_i, \ldots, k_n \mid \mathscr{D}) = \frac{p(\mathscr{D} \mid \lambda, k_1, \ldots, k_n)p(\lambda, k_1, \ldots, k_n)}{p(\mathscr{D})}$$

OK

Chain rule

$$= \frac{p(\mathscr{D} \mid \lambda, k_1, \ldots, k_n)p(k_1, \ldots, k_n \mid \lambda)p(\lambda)}{p(\mathscr{D})}$$

OK

Marginalize

$$= \frac{p(\mathscr{D} \mid \lambda, k_1, \ldots, k_n)p(k_1, \ldots, k_n \mid \lambda)p(\lambda)}{\int_{[0,10]\times\mathbb{N}^n} p(\mathscr{D} \mid \lambda, k_1, \ldots, k_n)p(k_1, \ldots, k_n \mid \lambda)p(\lambda)\, d\lambda dk_1 \ldots dk_n}$$

Problem #1: Normalizing constant

Marginalize

$$p(k_i \mid \mathscr{D}) = \int_{[0,10]\times\mathbb{N}^{k-1}} \frac{p(\mathscr{D} \mid \lambda, k_1, \ldots, k_n)p(k_1, \ldots, k_n \mid \lambda)p(\lambda)}{\int_{[0,10]\times\mathbb{N}^n} p(\mathscr{D} \mid \lambda, k_1, \ldots, k_n)p(k_1, \ldots, k_n \mid \lambda)p(\lambda)\, d\lambda dk_1 \ldots dk_n} d\lambda dk_1 \ldots dk_{i-1} dk_{i+1} \ldots dk_n$$

Problem #2: Marginalization

Estimating k_i

1. **MLE**
   - We can compute $p(\mathscr{D} \mid \lambda, k_1, \ldots, k_n)$
   - So "just" find its gradient, and then find $\arg \max_{\lambda, k_1, \ldots, k_n} p(\mathscr{D} \mid \lambda, k_1, \ldots, k_n)$
2. **MAP**
   - We can compute $p(\lambda, k_1, \ldots, k_n \mid \mathscr{D})p(\mathscr{D})$
   - We don't know what $p(\mathscr{D})$ is, but we know it's **constant** (with respect to $\lambda, k_1, \ldots, k_n$)
   - So $\arg \max_{\lambda, k_1, \ldots, k_n} p(\lambda, k_1, \ldots, k_n \mid \mathscr{D}) = \arg \max_{\lambda, k_1, \ldots, k_n} p(\lambda, k_1, \ldots, k_n \mid \mathscr{D})p(\mathscr{D})$
3. **Bayesian estimate** (i.e., $\mathbb{E}[k_i \mid \mathscr{D}]$)
   - We can compute $q(\lambda, k_1, \ldots, k_n) \propto p(\lambda, k_1, \ldots, k_n \mid \mathscr{D})$

Grid Search

Bayesian estimate (i.e., $\mathbb{E}[k_i \mid \mathscr{D}]$)

- We can compute $q(\lambda, k_1, \ldots, k_n) \propto p(\lambda, k_1, \ldots, k_n \mid \mathscr{D})$

- **Question:** If there were only 9 possible values of $\lambda, k_1, \ldots, k_n$, what could we do to compute $p(\lambda, k_1, \ldots, k_n \mid \mathscr{D})$?
- **Grid search** is exhaustively computing $q(\lambda, k_1, \ldots, k_n)$ over a finite "grid" of candidate values, and then dividing by the sum
- *Implicit assumption:* Only the grid values have positive probability
- The k_i's are already discrete, and it turns out to be reasonable to assume $k_i \in \{0,1,2,3\}$
- **Question:** How many possible values for the vector $(k_1, \ldots, k_n)$ under that assumption?
    - **"Curse of dimensionality"**: A typical experiment in this domain has $\geq$**40** players

Recap: Expectations from a Sample

Law of Large Numbers:
As the number R of independent samples $x^{(1)}, \ldots, x^{(R)}$ from a random variable X with distribution $p(x)$ approaches infinity, the **sample average** approaches the **expected value** of X.

$$\mathbb{E}[X] = \int_{\mathcal{X}} xp(x)\,dx \approx \frac{1}{R}\sum_{r=1}^{R} x^{(r)}$$

Since $Y = h(X)$ is also a random variable, this generalizes to arbitrary **functions** of X:

$$\mathbb{E}[h(X)] = \int_{\mathcal{X}} h(x)p(x)\,dx \approx \frac{1}{n}\sum_{r=1}^{R} h(x^{(r)})$$

If we can generate **independent samples** from a random variable's distribution, we can estimate the **expected value of arbitrary functions** of the random variable!

Probabilities from a Sample

- **Question:** Why might we want to estimate the **probability** of an event?
- Probability of an event A is just the expectation of its **indicator function**:

$$I_A[x] = \begin{cases} 1 & \text{if } x \in A, \\ 0 & \text{otherwise.} \end{cases}$$

e.g., $I_{X>4}[x] = \begin{cases} 1 & \text{if } x > 4, \\ 0 & \text{otherwise.} \end{cases}$

- So estimate that expectation as with any other function:

$$\Pr(A) = \mathbb{E}\left(I_A[X]\right) = \int_{\mathcal{X}} I_A[x]p(x)\,dx \approx \frac{1}{R}\sum_{r=1}^{R} I_A[x^{(r)}].$$

Marginals from Multivariate Samples

- In our example model, $\theta = (\lambda, k_1, \ldots, k_n)$
- **Question:** How can we estimate the value of just $\mathbb{E}[k_i \mid \mathscr{D}]$?
- Given a sample $\theta^{(r)} = (\lambda^{(r)}, k_1^{(r)}, \ldots, k_n^{(r)})$, let h_{k_i} be the **projection function**

$$h_{k_i}(\theta^{(r)}) = k_i^{(r)}$$

- Then the expectation of k_i is the expectation of the projection h_{k_i} of θ
- Estimate that expectation as with any other function:

$$\mathbb{E}[k_i \mid \mathscr{D}] = \mathbb{E}[h_{k_i}(\theta) \mid \mathscr{D}] = \int_\Theta h_{k_i}(\theta) p(\theta \mid \mathscr{D})\, d\theta \approx \frac{1}{R} \sum_{r=1}^{R} h_{k_i}(\theta^{(r)})$$

Generating Samples from a Single Variable

How can we generate samples from a distribution?

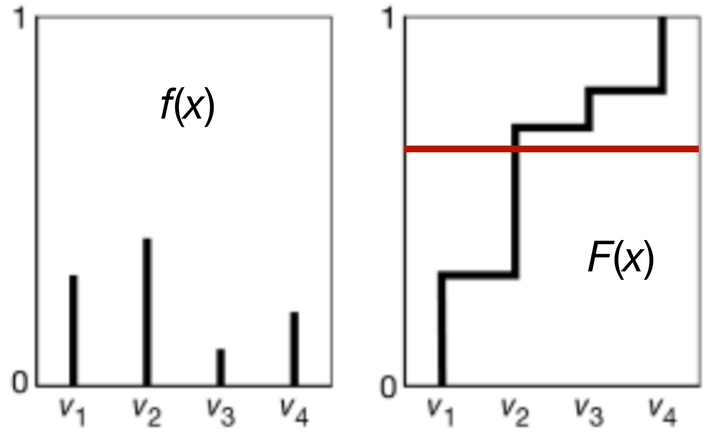
1. **Totally order** the domain of the variable
   (can be arbitrary for categorical variables)

2. **Cumulative distribution**: $F(x) = \Pr(X \leq x)$

$$F(x) = \int_{-\infty}^{x} f(z)dz \qquad F(x) = \sum_{x' \leq x} f(x')$$

3. Select a **uniform** random number $y \in [0,1]$

4. Return $x^{(r)} = F^{-1}(y)$



Hard-To-Sample Distributions

Often, we want to sample from distributions that are **hard** to sample from, especially large **joint distributions**

Question: Why might a distribution be hard to sample from?

1. Use samples from easier distributions:
    - Rejection Sampling
    - Importance Sampling
2. Go piece by piece through the joint distribution
    - Forward Sampling in a hierarchical model
    - Particle Filtering

Proposal Distributions

- Can we use an **easy-to-sample** distribution $g(x)$ to help us sample from $f(x)$?
  - Very common: We know an **unnormalized** $q(x)$, but not the properly normalized distribution $f(x) \propto q(x)$:

$$f(x) = \frac{q(x)}{\int_{\mathcal{X}} q(x)\, dz}$$

  - Typically, $q(\theta) = p(\mathcal{D} \mid \theta)p(\theta)$ and $f(\theta) = p(\theta \mid \mathcal{D}) = q(\theta) \big/ p(\mathcal{D})$
- $f(x)$ is the **target distribution**
  - $q(x)$ is the **unnormalized target distribution**
- $g(x)$ is the **proposal distribution**

Rejection Sampling

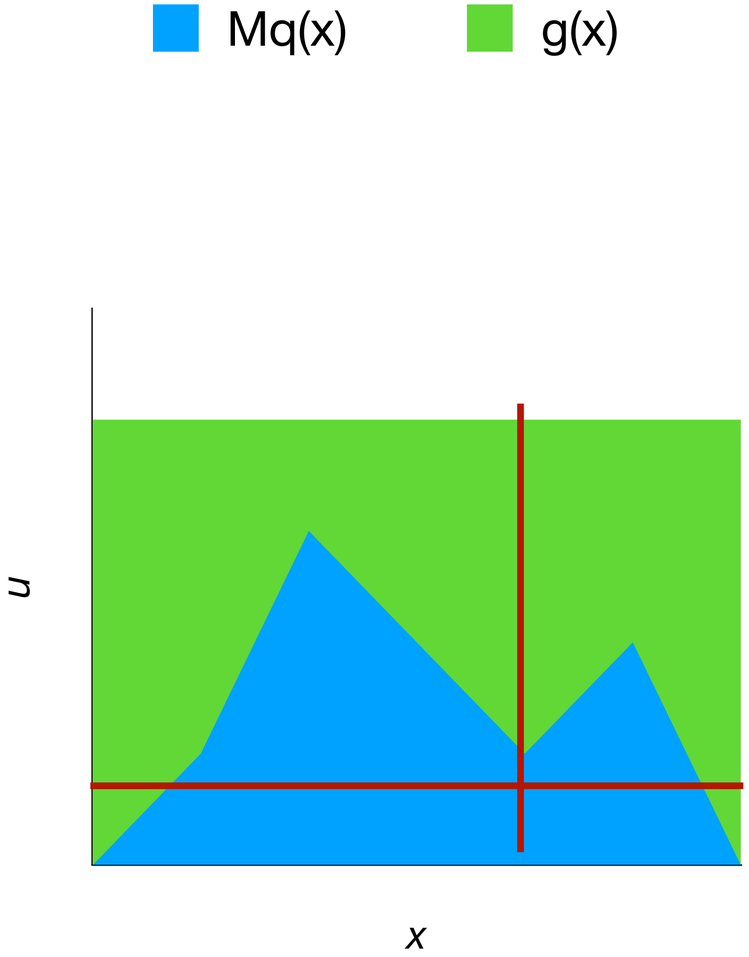
- Rejection sampling is one way to use a proposal distribution to sample from a target distribution
- *Assumption:* We know a constant M such that

$$\forall x : Mq(x) \leq g(x)$$

  - Much **easier** to find M than to find the constant that makes the integral come out to **exactly** 1

- **Repeat** until "enough" samples accepted:

  1. **Sample** $x \sim g(x)$ from the **proposal distribution**
  2. **Sample** $u \sim \text{Uniform}[0,1]$
  3. **If** $u \leq \left[Mq(x) \,/\, g(x) \right]$, **accept** x (add it to samples)
     Else reject



Importance Sampling

- Rejection sampling works, but it can be **wasteful**
  - Lots of samples get rejected when proposal and target distributions are very **different**
- What if we took a **weighted average** instead?
  1. Sample $x^{(1)}, x^{(2)}, \ldots, x^{(R)}$ from $g(x)$
  2. **Weight** each sample $x^{(r)}$ by $w^{(r)} = \frac{Mq(x^{(r)})}{g(x^{(r)})}$
  3. **Estimate** is $\frac{1}{\sum_{r=1}^{R} w^{(r)}} \sum_{x^{(r)} \sim g} w^{(r)} x^{(r)}$

$$\mathbb{E}[X] = \sum_x f(x)x$$

$$= \sum_x \frac{g(x)}{g(x)} f(x)x$$

$$= \sum_x g(x) \boxed{\frac{f(x)}{g(x)} x}$$

$$\approx \frac{1}{R} \sum_{x^{(r)} \sim g} \frac{f(x^{(r)})}{g(x^{(r)})} x^{(r)}$$

Forward Sampling in a Factored Joint Distribution

- Sometimes we know how to sample **parts** of a large joint distribution in terms of other parts
  - We might be able to **directly** sample from each **conditional distribution** but not from the **joint distribution**
  - E.g., sample $\lambda \sim \text{Uniform}[0,10]$ and then $k_1^{(r)}, \ldots, k_n^{(r)} \overset{i.i.d.}{\sim} \text{Poisson}(\lambda^{(r)})$ to get a sample from the joint prior $p(\lambda, k_1, \ldots, k_n)$

Forward sampling:

1. **Select** an ordering of variables consistent with the factoring
2. **Repeat** until enough samples generated:
   For each variable X_i in the ordering:
   Sample $x_i^{(r)} \sim P(X_i \mid x_1^{(r)}, \ldots, x_{i-1}^{(r)})$

Particle Filtering

- **Forward sampling** generates a value for each variable, then moves on to the next sample
- **Particle filtering** swaps the order:
  - Generate R values for variable X_1, then R values for variable X_2, etc.
  - Especially useful when there is no fixed number of variables (e.g., in sequential models)
- Each sample is called a **particle**. Update its **weight** each time a value is sampled.
- Periodically **resample** from the particles with replacement, resetting weights to 1
  - High-probability particles likely to be **duplicated**
  - Low-probability particles likely to be **discarded**
- Resampling means the particles cover the distribution better

Summary

- Often we **cannot directly estimate** probabilities or expectations from our model
  - E.g., **hierarchical models** with **nonconjugate** distributions
- **Monte Carlo estimates**: Use a **random sample** from the distribution to estimate expectations by sample averages
- Two families of techniques for hard to sample distributions:
  1. Use an easier-to-sample **proposal** distribution instead
  2. Sample parts of the model **sequentially**