CPSC 440: Advanced Machine Learning

Monte Carlo Methods

Mark Schmidt

University of British Columbia

Winter 2021

Last Time: Markov Chains

- We can use Markov chains for density estimation,

$$p(x) = \underbrace{p(x_1)}_{\text{initial prob.}} \prod_{j=2}^{d} \underbrace{p(x_j \mid x_{j-1})}_{\text{transition prob.}},$$

  which model dependency between adjacent features.
    - Different than mixture models which focus on clusters in the data.

- Homogeneous chains use same transition probability for all j (parameter tieing).
    - Gives more data to estimate transitions, allows examples of different sizes.

- Inhomogeneous chains allow different transitions at different times.
    - More flexible, but need more data.

- MLE is discrete-state Markov chains has simple closed-form "counting" solution.

Training Markov Chains

- Some common setups for fitting the parameters Markov chains:
  1. We have one long sequence, and fit parameters of a homogeneous Markov chain.
     - Here, we just focus on the transition probabilities.
  2. We have many sequences of different lengths, and fit a homogeneous chain.
     - And we can use it to model sequences of any length.
  3. We have many sequences of same length, and fit an inhomgeneous Markov chain.
     - This allows "position-specific" effects.
  4. We use domain knowledge to guess the initial and transition probabilities.

Fun with Markov Chains

- Markov Chains "Explained Visually":
  http://setosa.io/ev/markov-chains
- Snakes and Ladders:
  http://datagenetics.com/blog/november12011/index.html
- Candyland:
  http://www.datagenetics.com/blog/december12011/index.html
- Yahtzee:
  http://www.datagenetics.com/blog/january42012/
- Chess pieces returning home and K-pop vs. ska:
  https://www.youtube.com/watch?v=63HHmjlh794

Inference in Markov Chains

- Given a Markov chain model, these are the most common inference tasks:
  1. Sampling: generate sequences that follow the probability.
  2. Marginalization: compute probability of being in state c at time j.
  3. Decoding: compute assignment to the x_j with highest joint probability.
     - Decoding and marginalization will be important when we return to supervised learning.
  4. Conditioning: do any of the above, assuming $x_j = c$ for some j and c.
     - For example, "filling in" missing parts of the image.
  5. Stationary distribution: probability of being in state c as j goes to ∞.
     - Usually for homogeneous Markov chains.

Outline

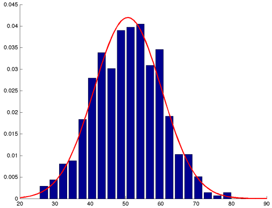
Fundamental Problem: Sampling from a Density

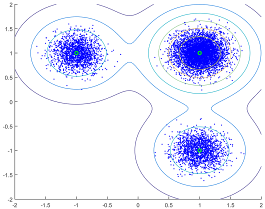
- A common inference task is sampling from a density.
  - Generating examples x^i that are distributed according to a given density $p(x)$.
  - Basically, the "opposite" of density estimation: going from a model to data.

$$p(x) = \begin{cases} 1 & \text{w.p. } 0.5 \\ 2 & \text{w.p. } 0.25 \\ 3 & \text{w.p. } 0.25 \end{cases} \quad \Rightarrow \quad X = \begin{bmatrix} 1 \\ 2 \\ 1 \\ 1 \\ 3 \\ 2 \\ 1 \\ 3 \end{bmatrix}.$$

Fundamental Problem: Sampling from a Density

- A common inference task is sampling from a density.
  - Generating examples x^i that are distributed according to a given density $p(x)$.
  - Basically, the "opposite" of density estimation: going from a model to data.





- We've been using pictures of samples to "tell us what the model has learned".
  - If the samples look like real data, then we have a good density model.

- Samples can also be used in Monte Carlo estimation (today):
  - Replace complicated $p(x)$ with samples to solve hard problems at test time.

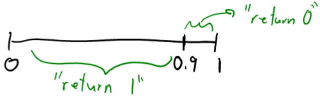
Simplest Case: Sampling from a Bernoulli

- Consider sampling from a Bernoulli, for example

$$p(x = 1) = 0.9, \quad p(x = 0) = 0.1.$$

- Sampling methods assume we can sample uniformly over $[0, 1]$.
  - Usually, a “pseudo-random” number generator is good enough (like Julia’s *rand*).

- How to use a uniform sample to sample from the Bernoulli above:
  1. Generate a uniform sample $u \sim \mathcal{U}(0, 1)$.
  2. If $u \leq 0.9$, set $x = 1$ (otherwise, set $x = 0$).



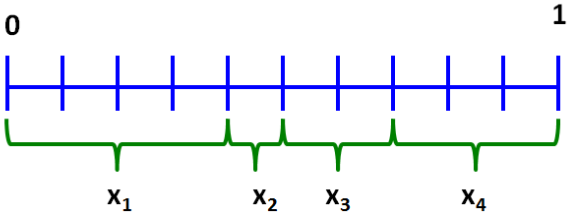
- If uniform samples are “good enough”, then we have $x = 1$ with probability 0.9.

Sampling from a Categorical Distribution

- Consider a more general categorical density like

$$p(x = 1) = 0.4, \quad p(x = 2) = 0.1, \quad p(x = 3) = 0.2, \quad p(x = 4) = 0.3,$$

we can divide up the $[0, 1]$ interval based on probability values:



- If $u \sim \mathcal{U}(0, 1)$, 40% of the time it lands in x_1 region, 10% of time in x_2, and so on.

Sampling from a Categorical Distribution

- Consider a more general categorical density like

$$p(x=1)=0.4, \quad p(x=2)=0.1, \quad p(x=3)=0.2, \quad p(x=4)=0.3.$$

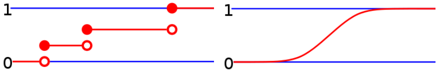
- To sample from this categorical density we can use (*sampleDiscrete* function):
  1. Generate $u \sim \mathcal{U}(0,1)$.
  2. If $u \leq 0.4$, output 1.
  3. If $u \leq 0.4 + 0.1$, output 2.
  4. If $u \leq 0.4 + 0.1 + 0.2$, output 3.
  5. Otherwise, output 4.

Sampling from a Categorical Distribution

- General case for sampling from categorical.
  1. Generate $u \sim \mathcal{U}(0, 1)$.
  2. If $u \leq p(x \leq 1)$, output 1.
  3. If $u \leq p(x \leq 2)$, output 2.
  4. If $u \leq p(x \leq 3)$, output 3.
  5. ...
- The value $p(x \leq c) = p(x = 1) + p(x = 2) + \cdots + p(x = c)$ is the CDF.
  - "Cumulative distribution function".
- Worst case cost with k possible states is $O(k)$ by incrementally computing CDFs.
- But to generate t samples only costs $O(k + t \log k)$ instead of $O(tk)$:
  - One-time $O(k)$ cost to store the CDF $p(x \leq c)$ for each c.
  - Per-sample $O(\log k)$ cost to do binary search for smallest c with $u \leq p(x \leq c)$.

Cumulative Distribution Function (CDF)

- We often use $F(c) = p(x \leq c)$ to denote the CDF.
  - $F(c)$ is between 0 and 1, giving proportion of times x is below c.
  - $F(c)$ monotically increases with c.
  - F can be used for discrete and continuous variables:



https://en.wikipedia.org/wiki/Cumulative_distribution_function

- The "binary search for smallest c" method finds smallest c such that $u \leq F(c)$.
  - This same approach works for continuous and general densities.

- General approach uses the inverse CDF (or "quantile") function:
  - If F is invertible, then F^{-1} is the usual inverse:
    - $F^{-1}(u) = c$ for the unique c where $F(c) = u$.
  - The generalization that covers non-invertible cases is $F^{-1}(u) = \inf\{c \mid F(c) \geq u\}$.
    - "Return the smallest c where $F(c)$ is at least u."

Inverse Transform Method (Exact 1D Sampling)

- Inverse transfrom method for exact sampling in 1D:
  1. Sample $u \sim \mathcal{U}(0,1)$.
  2. Return $F^{-1}(u)$.

- Why this works (invertible case);

$$\begin{aligned} p(F^{-1}(u) \leq c) &= p(u \leq F(c)) && \text{(apply monotonic } F \text{ to both sides)} \\ &= F(c) && \text{(since } p(u \leq y) = y \text{ for uniform } u\text{)} \end{aligned}$$

  - So this algorithm has the same CDF as the distribution we want to sample.

- Video on pseudo-random numbers and inverse-transform sampling:
  - `https://www.youtube.com/watch?v=C82JyCmtKWg`

Example: Sampling from a 1D Gaussian

- Consider a Gaussian distribution,

$$x \sim \mathcal{N}(\mu, \sigma^2).$$

- CDF has the form

$$F(c) = p(x \leq c) = \frac{1}{2}\left[1 + \text{erf}\left(\frac{c-\mu}{\sigma\sqrt{2}}\right)\right],$$

where the "error function" is $\text{erf}(c) = \frac{2}{\pi}\int_0^c \exp(-t^2)dt$.
  - In $[-1, 1]$ with no closed-form solution, but is typically available in stats packages.

- Inverse CDF has the form

$$F^{-1}(u) = \mu + \sigma\sqrt{2}\text{erf}^{-1}(2u-1).$$

- To sample from a Gaussian:
  1. Generate $u \sim \mathcal{U}(0, 1)$.
  2. Return $\mu + \sigma\sqrt{2}\text{erf}^{-1}(2u-1)$.

Sampling from a Product Distribution

- Consider a product distribution,

$$p(x_1, x_2, \ldots, x_d) = p(x_1)p(x_2)\cdots p(x_d).$$

- Because variables are independent, we can sample independently:
  - Sample x_1 from $p(x_1)$.
  - Sample x_2 from $p(x_2)$.
  - ...
  - Sample x_d from $p(x_d)$.

- Example: sampling from a multivariate Gaussian with diagonal covariance.
  - Sample each variable independently based on μ_j and σ_j^2.

Digression: Sampling from a Multivariate Gaussian

- In some cases we can sample from multivariate distributions by transformation.
- Recall the affine property of multivariate Gaussian:
  - If $x \sim \mathcal{N}(\mu, \Sigma)$, then $Ax + b \sim \mathcal{N}(A\mu + b, A\Sigma A^T)$.
- To sample from a general multivariate Gaussian $\mathcal{N}(\mu, \Sigma)$:
  1. Sample x from a $\mathcal{N}(0, I)$ (each x_j coming independently from $\mathcal{N}(0, 1)$).
  2. Transform to a sample from the right Gaussian using the affine property:

$$Ax + \mu \sim \mathcal{N}(\mu, AA^T),$$

where we choose A so that $AA^T = \Sigma$ (e.g., by Cholesky factorization).

Ancestral Sampling

- To sample dependent random variables we can use the chain rule of probability,

$$p(x_1, x_2, x_3, \dots, x_d) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_2, x_1) \cdots p(x_d \mid x_{d-1}, x_{d-2}, \dots, x_1).$$

- The chain rule suggests the following sampling strategy:
  - Sample x_1 from $p(x_1)$.
  - Given x_1, sample x_2 from $p(x_2 \mid x_1)$.
  - Given x_1 and x_2, sample x_3 from $p(x_3 \mid x_2, x_1)$.
  - $\dots$
  - Given x_1 through x_{d-1}, sample x_d from $p(x_d \mid x_{d-1}, x_{d-2}, \dots x_1)$.

- This is called ancestral sampling.
  - It's easy if (conditional) probabilities are simple, since sampling in 1D is usually easy.
  - But may not be simple, binary conditional j has 2^j values of $\{x_1, x_2, \dots, x_j\}$.

Ancestral Sampling Examples

- For Markov chains the chain rule simplifies to

$$p(x_1, x_2, x_3, \ldots, x_d) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_2) \cdots p(x_d \mid x_{d-1}),$$

- So ancestral sampling simplifies too:
  1. Sample x_1 from initial probabilities $p(x_1)$.
  2. Given x_1, sample x_2 from transition probabilities $p(x_2 \mid x_1)$.
  3. Given x_2, sample x_3 from transition probabilities $p(x_3 \mid x_2)$.
  4. ...
  5. Given x_{d-1}, sample x_d from transition probabilities $p(x_d \mid x_{d-1})$.

- For mixture models with cluster variables z we could write

$$p(x, z) = p(z)p(x \mid z),$$

so we can first sample cluster z and then sample x given cluster z.
  - If you want samples of x, sample (x, z) pairs and ignore the z values.

Markov Chain Toy Example: CS Grad Career

- "Computer science grad career" Markov chain:
  - Initial probabilities:

State	Probability	Description
Industry	0.60	They work for a company or own their own company.
Grad School	0.30	They are trying to get a Masters or PhD degree.
Video Games	0.10	They mostly play video games.

  - Transition probabilities (from row to column):

From\to	Video Games	Industry	Grad School	Video Games (with PhD)	Industry (with PhD)	Academia	Deceased
Video Games	0.08	0.90	0.01	0	0	0	0.01
Industry	0.03	0.95	0.01	0	0	0	0.01
Grad School	0.06	0.06	0.75	0.05	0.05	0.02	0.01
Video Games (with PhD)	0	0	0	0.30	0.60	0.09	0.01
Industry (with PhD)	0	0	0	0.02	0.95	0.02	0.01
Academia	0	0	0	0.01	0.01	0.97	0.01
Deceased	0	0	0	0	0	0	1

- So $p(x_t = \text{"Grad School"} \mid x_{t-1} = \text{"Industry"}) = 0.01$.

Example of Sampling x_1

- Initial probabilities are:
  - 0.1 (Video Games)
  - 0.6 (Industry)
  - 0.3 (Grad School)
  - 0 (Video Games with PhD)
  - 0 (Academia)
  - 0 (Deceased)
- So initial CDF is:
  - 0.1 (Video Games)
  - 0.7 (Industry)
  - 1 (Grad School)
  - 1 (Video Games with PhD)
  - 1 (Academia)
  - 1 (Deceased)
- To sample the initial state x_1:
  - First generate a uniform number u, for example $u = 0.724$.
  - Now find the first CDF value bigger than u, which in this case is "Grad School".

Example of Sampling x_2, Given $x_1 =$ "Grad School"

- So we sampled $x_1 =$ "Grad School".
  - To sample x_2, we'll use the "Grad School" row in transition probabilities:

From\to	Video Games	Industry	Grad School	Video Games (with PhD)	Industry (with PhD)	Academia	Deceased
Video Games	0.08	0.90	0.01	0	0	0	0.01
Industry	0.03	0.95	0.01	0	0	0	0.01
Grad School	0.06	0.06	0.75	0.05	0.05	0.02	0.01
Video Games (with PhD)	0	0	0	0.30	0.60	0.09	0.01
Industry (with PhD)	0	0	0	0.02	0.95	0.02	0.01
Academia	0	0	0	0.01	0.01	0.97	0.01
Deceased	0	0	0	0	0	0	1

Example of Sampling x_2, Given $x_1 =$ "Grad School"

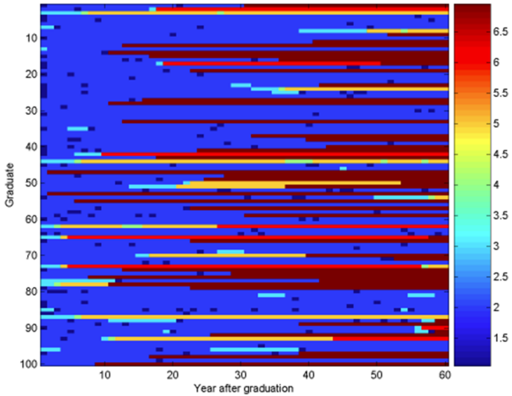
- Transition probabilities:
  - 0.06 (Video Games)
  - 0.06 (Industry)
  - 0.75 (Grad School)
  - 0.05 (Video Games with PhD)
  - 0.02 (Academia)
  - 0.01 (Deceased)

- So transition CDF is:
  - 0.06 (Video Games)
  - 0.12 (Industry)
  - 0.87 (Grad School)
  - 0.97 (Video Games with PhD)
  - 0.99 (Academia)
  - 1 (Deceased)

- To sample the second state x_2:
  - First generate a uniform number u, for example $u = 0.113$.
  - Now find the first CDF value bigger than u, which in this case is "Industry".

Markov Chain Toy Example: CS Grad Career

- Samples from "computer science grad career" Markov chain:



- State 7 ("deceased") is called an absorbing state (no probability of leaving).
- Samples often give you an idea of what model knows (and what should be fixed).

Outline

Marginalization and Conditioning

- Given density estimator, we often want to make probabilistic inferences:
  - Marginals: what is the probability that $x_j = c$?
    - What is the probability we're in industry 10 years after graduation?
  - Conditionals: what is the probability that $x_j = c$ given $x_{j'} = c'$?
    - What is the probability of industry after 10 years, if we immediately go to grad school?

- This is easy for simple independent models:
  - We directly model marginals $p(x_j)$, and conditional are marginals: $p(x_j \mid x_{j'}) = p(x_j)$.

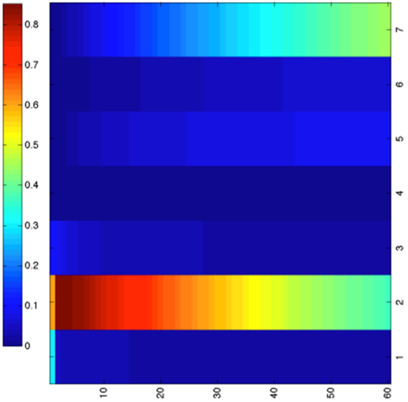
- This is also easy for mixtures of simple independent models.
  - Do inference for each mixture, add results using mixture probabilities:

$$p(x_j) = \sum_z p(z, x_j) = \sum_z p(z) \underbrace{p(x_j \mid z)}_{\text{inference within cluster}}.$$

- For Markov chains, it's more complicated...

Marginals in CS Grad Career

- All marginals $p(x_j = c)$ from "computer science grad career" Markov chain:



- Each row j is a state and each column c is a year (sum of values in column is 1).

Monte Carlo: Marginalization by Sampling

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

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

  2. 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 second most important class of ML algorithms.
  - Originally developed to build better atomic bombs :(
    - Run physics simulator to "sample", then see if it leads to a chain reaction.

Monte Carlo Method for Rolling Di

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

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

- Computing probability of a pair of 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.

Summary

- Markov chain inference tasks (MEMORIZE):
  - Sampling, marginalization, decoding, conditioning, stationary distributions.
- Inverse Transform generates samples from simple 1D distributions.
  - When we can easily invert the CDF.
- Ancestral sampling generates samples from multivariate distributions.
  - When conditionals have a nice form.
- Monte Carlo method for approximating probabilities of an event.
  - Generate samples, then count how many times event happened.
- Next time: the original Google algorithm.