CPSC 540: Machine Learning

Monte Carlo Methods

Mark Schmidt

University of British Columbia

Winter 2020

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.
- Given a Markov chain model, we overviewed common computational problems:
    - Sampling, marginalization, decoding, conditioning, and stationary distribution.

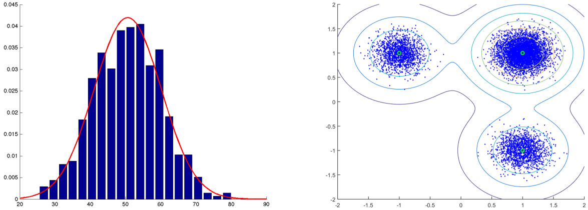
Fundamental Problem: Sampling from a Density

- 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.

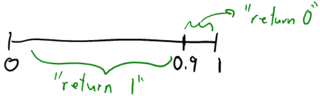
- Samples can also be used in Monte Carlo estimation (today):
  - Replace complicated $p(x)$ with samples to solve hard problems at test time.

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 $x = 0$).



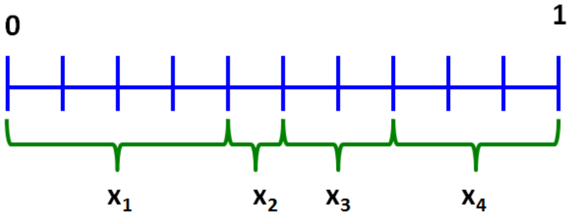
- If uniform samples are "good enough", then we have $x = 1$ with probability 0.9.

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

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

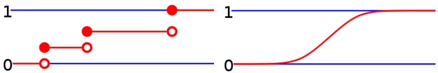
- To sample from this categorical density we can use (*sampleDiscrete* function):
  1. Generate $u \sim \mathcal{U}(0, 1)$.
  2. If $u \leq 0.4$, output 1.
  3. If $u \leq 0.4 + 0.1$, output 2.
  4. If $u \leq 0.4 + 0.1 + 0.2$, output 3.
  5. Otherwise, output 4.

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. $\dots$
- 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.

- Genral approach uses the inverse CDF (or "quantile") function:
  - $F^{-1}(u) = \inf\{c \mid F(c) \geq u\}$.
    - Given a number u between 0 and 1, returns smallest c with $p(x \leq c) = u$.
    - If F is invertible, then F^{-1} is the usual inverse.

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 "erf" is the CDF of $\mathcal{N}(0, 1)$.

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

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).

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.

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 $p(x_3 \mid x_2, x_1)$.
  - ...
  - 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"

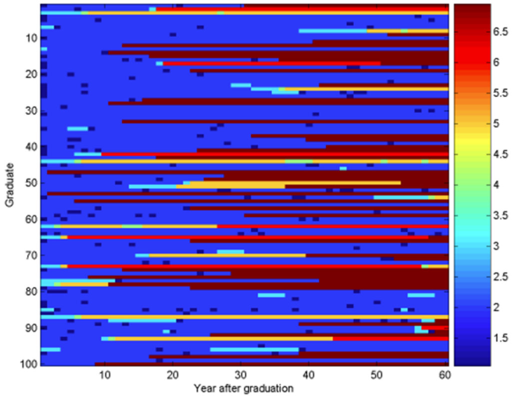
- Transition probabilities:
  - 0.06 (Video Games)
  - 0.06 (Industry)
  - 0.75 (Grad School)
  - 0.05 (Video Games with PhD)
  - 0.02 (Academia)
  - 0.01 (Deceased)

- 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: $p(x_j \mid x_{j'}) = p(x_j)$.

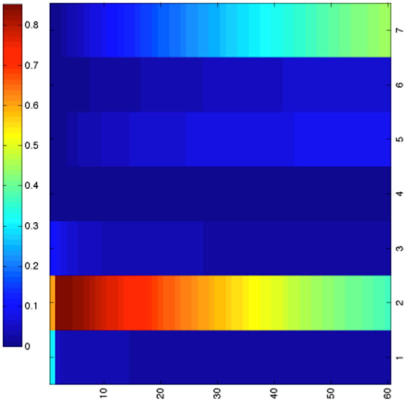
- This is also easy for mixtures of simple independent models.
  - Do inference for each mixture, add results using mixture probabilities:

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

- 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.

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.

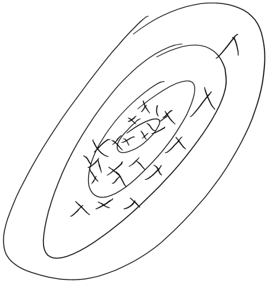
Monte Carlo Method for Inequalities

- Monte Carlo estimate of probability that variable is above threshold:
  - Compute fraction of examples where sample is above threshold.

Monte Carlo Method for Mean

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

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



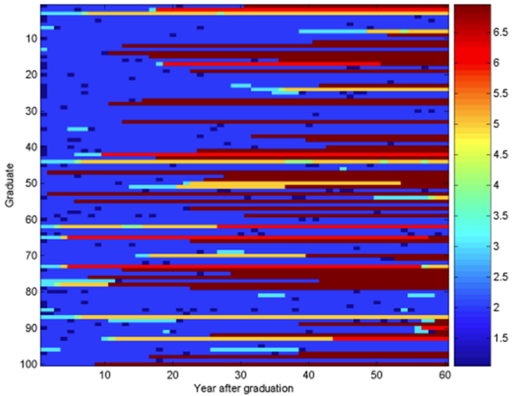
- Visual demo of Monte Carlo approximation of mean and vairance:
  - http://students.brown.edu/seeing-theory/basic-probability/index.html

Monte Carlo for Non-Uniform Coupon Collecting

- Recall the coupon collecting problem:
  - You have a set of n possible objects.
  - How many IID samples do you need before you see all n?
- With uniform probabilities, expected value is $O(n \log n)$.
- Witn non-uniform probabilities, you can approximate it with Monte Carlo:
  - Take IID samples until you have seen all objects.
  - Repeat many times and take the average time.
  - Don't even need to necessarily know the probabilities.

Monte Carlo for Markov Chains

- Our samples from the CS grad student Markov chain:



- We can estimate probabilities by looking at frequencies in samples.
  - In how many out of the 100 chains did we have $x_{10} =$ "industry"?
- This works for continuous states too (for inequalities and expectations).

Monte Carlo Methods for Markov Chains

- Some Monte Carlo approximations of inference tasks in Markov chains:
  - Marginal $p(x_j = c)$ is the number of chains that were in state c at time j.
  - Average value at time j, $E[x_j]$, is approximated by average of x_j in the samples.
  - $p(x_j \leq 10)$ is approximate by frequency of x_j being less than 10.
  - $p(x_j \leq 10, x_{j+1} \geq 10)$ is approximated by number of chains where both happen.

Monte Carlo Methods

- Monte Carlo methods approximate expectations of random functions,

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

- Computing mean is the special case of $g(x) = x$.
- Computing probability of any event A is also a special case:
  - Set $g(x) = \mathcal{I}[\text{“}A \text{ happened in sample } x^i\text{”}]$.

- To approximate expectation, generate n samples x^i from $p(x)$ and use:

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

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,

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

- 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 $O(1/n)$,

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

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

Conditional Probabilities with Monte Carlo

- We often want to compute conditional probabilities in Markov chains.
  - We can ask "what lead to $x_{10} = 4$?" with queries like $p(x_1 \mid x_{10} = 4)$.
  - We can ask "where does $x_{10} = 4$ lead?" with queries like $p(x_d \mid x_{10} = 4)$.

- Monte Carlo approach to estimating $p(x_j \mid x_{j'})$:
  1. Generate a large number of samples from the Markov chain, $x^i \sim p(x_1, x_2, \dots, x_d)$.
  2. Use Monte Carlo estimates of $p(x_j = c, x_{j'} = c')$ and $p(x_{j'} = c')$ to give

$$p(x_j = c \mid x_{j'} = c') = \frac{p(x_j = c, x_{j'} = c')}{p(x_{j'} = c')} \approx \frac{\sum_{i=1}^n I[x_j^i = c, x_{j'}^i = c']}{\sum_{i=1}^n I[x_{j'}^i = c']},$$

frequency of first event in samples consistent with second event.

- This is a special case of rejection sampling (we'll see general case later).
  - Unfortunately, if $x_{j'} = c'$ is rare then most samples are "rejected" (ignored).

http://students.brown.edu/seeing-theory/compound-probability/index.html

Summary

- 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 methods approximate expectations using samples.
  - Can be used to approximate arbitrary probabilities in Markov chains.
- Next time: the original Google algorithm.

Monte Carlo as a Stochastic Gradient Method

- Consider case of using Monte Caro method to estimate mean $\mu = \mathbb{E}[x]$,

$$\mu \approx \frac{1}{n}\sum_{i=1}^{n} x^i.$$

- We can write this as minimizing the 1-strongly convex

$$f(w) = \frac{1}{2}\|w - \mu\|^2.$$

- The gradient is $\nabla f(w) = (w - \mu)$.
- Consider applying stochastic gradient descent on f using

$$\nabla f_i(w^k) = w^k - x^{k+1},$$

  which is unbiased since each x^i is unbiased μ approximation.
- Monte Carlo method is a stochastic gradient method with this approximation.

Monte Carlo as a Stochastic Gradient Method

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

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

- We know the rate of stochastic gradient for strongly-convex is $O(1/n)$.

Law of the Unconscious Statistician

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

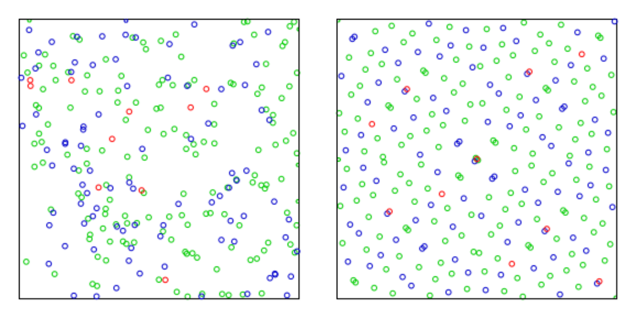
$$\mathbb{E}[g(x)] = \underbrace{\sum_{x \in \mathcal{X}} g(x)p(x)}_{\text{discrete } x} \quad \text{or} \quad \underbrace{\mathbb{E}[g(x)] = \int_{x \in \mathcal{X}} g(x)p(x)dx}_{\text{continuous } x} .$$

- 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.

Accelerated Monte Carlo: Quasi Monte Carlo

- Unlike stochastic gradient, there are some "accelerated" Monte Carlo methods.

- Quasi Monte Carlo methods achieve an accelerated rate of $O(1/n^2)$.
  - Key idea: fill the space strategically with a deterministic "low-discrepancy sequence".
  - Uniform random vs. deterministic low-discrepancy:



https://en.wikipedia.org/wiki/Quasi-Monte_Carlo_method