# Approximate Inference: Monte Carlo, Laplace Approximation

## CPSC 440/550: Advanced Machine Learning

cs.ubc.ca/~dsuth/440/24w2

University of British Columbia, on unceded Musqueam land

2024-25 Winter Term 2 (Jan–Apr 2025)

# Overview of Bayesian Inference Tasks

- Bayesian inference requires computing expectations with respect to posterior,

$$\mathbb{E}[f(\theta)] = \int_{\theta} f(\theta)\, p(\theta \mid x) \mathrm{d}\theta$$

  - If $f(\theta) = \theta$, we get posterior mean of $\theta$
  - If $f(\theta) = p(\tilde{x} \mid \theta)$, we get posterior predictive
  - If $f(\theta) = \mathbb{1}(\theta \in S)$ we get probability of $S$ (e.g., marginals)

- But posterior often doesn't have a closed-form expression
  - Bayesian linear regression – $w \sim \mathcal{N}(m, V)$; $y \mid x, w \sim \mathcal{N}(w^{\mathsf{T}} x, \sigma^2)$ – does
  - Bayesian logistic regression – $p(y \mid x, w) = 1/(1 + \exp(-y\, w^{\mathsf{T}} x))$ – doesn't
  - More complex models almost never do

- Our two main tools for approximate inference:
  1. Monte Carlo methods
  2. Variational methods
- Classic ideas from statistical physics that revolutionized Bayesian stats

# Approximate Inference

Two main strategies for approximate inference:

1. Monte Carlo methods:
   - Approximate expectations based on samples,

$$\mathop{\mathbb{E}}_{X \sim p} f(X) \approx \frac{1}{n} \sum_{i=1}^{n} f(x^{(i)})$$

   - Turns inference into sampling
2. Variational methods:
   - Approximate $p$ with "closest" distribution $q$ from a tractable family,

$$\mathop{\mathbb{E}}_{X \sim p} f(X) \approx \mathop{\mathbb{E}}_{X \sim q} f(X)$$

   - $q$ could be Gaussian, product of Bernoulli, any other model with easy inference...
   - Turns inference into optimization

# Outline

# Monte Carlo: estimation by sampling

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

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

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

- 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 $f(X)$:

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

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

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

# Monte Carlo: theory

- Let $\mu = \mathbb{E}[f(X)]$ be the value we want to compute (and assume it exists)
- Estimate is $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} f\left(x^{(i)}\right)$ (can view as an instance of SGD, see bonus)
- With iid samples, Monte Carlo gives an unbiased estimate of $\mu$:

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

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

- Assume $\sigma^2 = \mathrm{Var}[f(X)]$ exists and is bounded ("not infinite")
- Then expected squared error is exactly

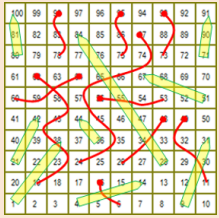
$$\mathbb{E}(\hat{\mu} - \mu)^2 = \mathrm{Var}(\hat{\mu}) = \mathrm{Var}\left(\frac{1}{n}\sum_{i=1}^{n} f\left(x^{(i)}\right)\right) = \frac{1}{n^2}\sum_{i=1}^{n} \sigma^2 = \frac{\sigma^2}{n}$$

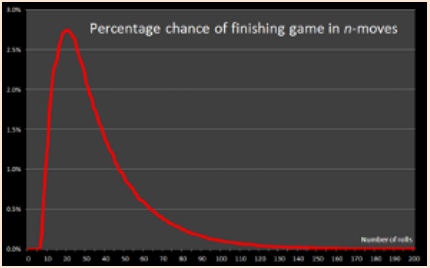
- $\hat{\mu}$ is *approximately* normal with mean $\mu$ and variance $\frac{\sigma^2}{n}$ (central limit theorem)

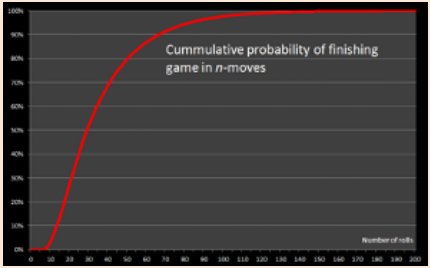
# Example application: Snakes and Ladders

bonus!

- Kid's game "Snakes and Ladders":
  - Start at 1, roll die, move the marker, follow snake/ladder
  - Absolutely no decision-making: can simulate the game
- How long does this game go for?
  - Run the game lots of times, see how many turns it took






https://www.datagenetics.com/blog/november12011/

# Conditional probabilities with Monte Carlo

- "How much looooonger will this game go?"
  - Just simulate starting from current game state

- "What's the probability the game will go >100 turns, if it's already gone 50?"
- One approach:

$$\Pr(A \mid B) = \frac{\Pr(A \cap B)}{\Pr(B)} \approx \frac{\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(A \text{ and } B \text{ happened on } x^{(i)})}{\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(B \text{ happened on } x^{(i)})}$$

- This is one instance of rejection sampling (more later)
- If $B$ is rare, most samples are wasted

# Monte Carlo for Bayesian inference

- We usually want to compute some kind of expectation under

$$p(\Theta \mid \mathbf{X}) = \frac{p(\mathbf{X} \mid \Theta)p(\Theta)}{p(\mathbf{X})}$$

- We can usually easily compute $p(\mathbf{X} \mid \theta)$ and $p(\theta)$
- But $p(\mathbf{X}) = \int p(\mathbf{X} \mid \theta)p(\theta)\mathrm{d}\theta$ is usually tough
- Even if we have $p(\theta \mid \mathbf{X})$, inverse CDF sampling is hard in high dimensions

- There are various ways to do approximate sampling from unnormalized densities
- Especially Markov chain Monte Carlo (MCMC)
  - We'll need to study Markov chains first!
- Today: two other ways

# Motivating problem: Bayesian Logistic Regression

- A classic way to fit a binary classifier is L2-regularized logistic loss,

$$\hat{w} \in \arg\max_{w} \sum_{i=1}^{n} \log(1 + \exp(-y^{(i)}\, w^{\mathsf{T}} x^{(i)})) + \frac{\lambda}{2}\|w\|^2$$

- This corresponds to using a sigmoid likelihood and Gaussian prior,

$$p(y \mid x, w) = \frac{1}{1 + \exp(-y\, w^{\mathsf{T}} x)}, \quad w \sim \mathcal{N}\left(0, \frac{1}{\lambda}\mathbf{I}\right)$$

- In Bayesian logistic regression, we'd work with the posterior
  - But the posterior isn't Gaussian: so this isn't a conjugate prior
  - We don't have a nice expression for the posterior predictive or marginal likelihood

# Motivation: Monte Carlo for Bayesian Logistic Regression

- Posterior predictive in Bayesian logistic regression has the form

$$p(\tilde{y} \mid \tilde{x}, \mathbf{X}, \mathbf{y}, \lambda) = \int_w p(\tilde{y} \mid \tilde{x}, w)\, p(w \mid \mathbf{X}, \mathbf{y}, \lambda)\, \mathrm{d}w$$
$$= \mathop{\mathbb{E}}_{w}\big[p(\tilde{y} \mid \tilde{x}, w) \mid \mathbf{X}, \mathbf{y}, \lambda\big]$$

- Given $w$, we can compute $p(\tilde{y} \mid \tilde{x}, w) = 1/\left(1 + \exp\left(-\tilde{y}\, w^{\mathsf{T}} \tilde{x}\right)\right)$ just fine
- If we could sample from the posterior for $w$, we could estimate with Monte Carlo!
  - But we don't know how to generate IID samples from this posterior

- Soon, we'll cover MCMC, which is a standard method in scenarios like this

- But we'll start simpler: rejection sampling and importance sampling
- These methods assume you can generate from a simple distribution $q$
  - for example, a Gaussian
- but you really want to solve an integral for a complicated distribution $p$
  - for example, the posterior for Bayesian logistic regression

# Outline

# Rejection Sampling for Conditionals

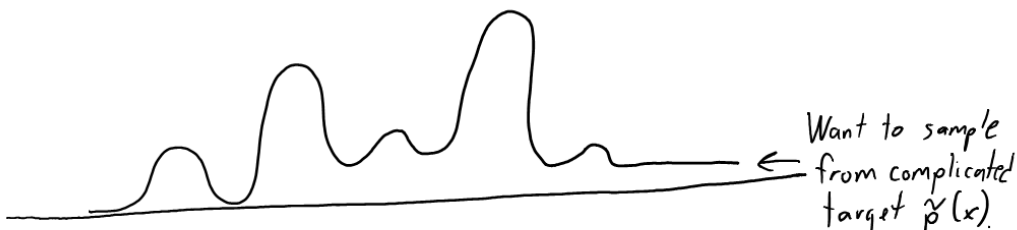
- We already mentioned rejection sampling for conditional sampling:
  - Example: sampling from a Gaussian conditional on knowing $x \in [-1, 1]$


  - Generate Gaussian samples, throw out ("reject") the ones that aren't in $[-1, 1]$
  - The remaining samples will follow the conditional distribution

- Can be used to generate IID samples from conditional distributions

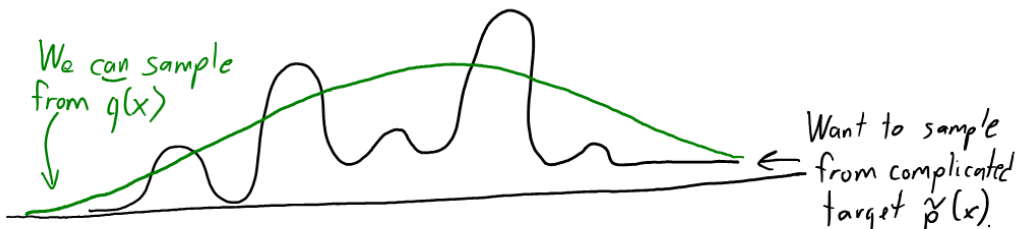
# General Rejection Sampling Algorithm

- General rejection sampling algorithm tries to "sample area under the graph":



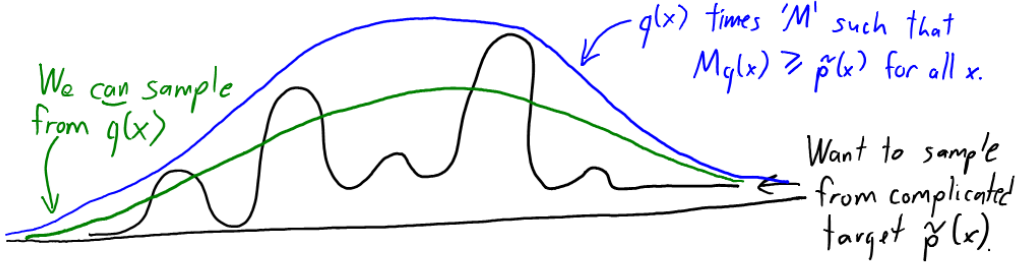
# General Rejection Sampling Algorithm

- General rejection sampling algorithm tries to "sample area under the graph":



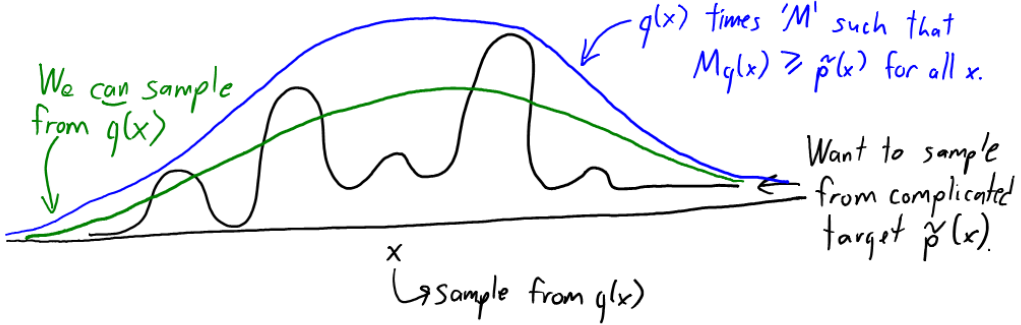
# General Rejection Sampling Algorithm

- General rejection sampling algorithm tries to "sample area under the graph":



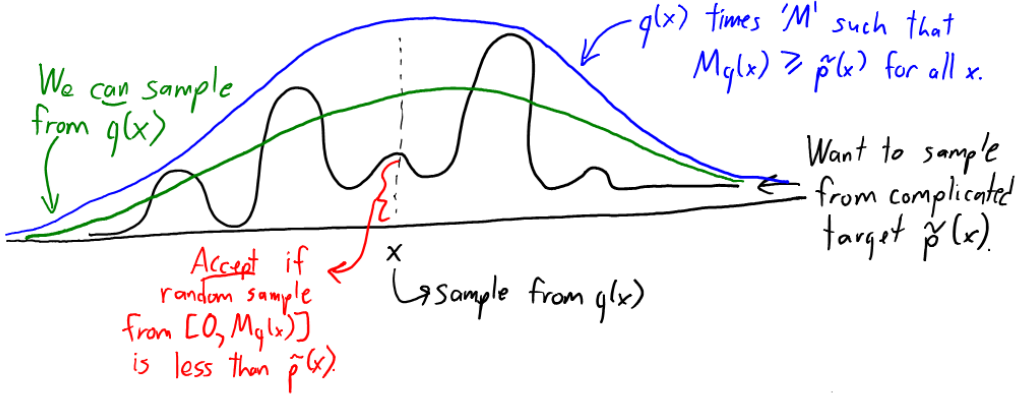
# General Rejection Sampling Algorithm

- General rejection sampling algorithm tries to "sample area under the graph":



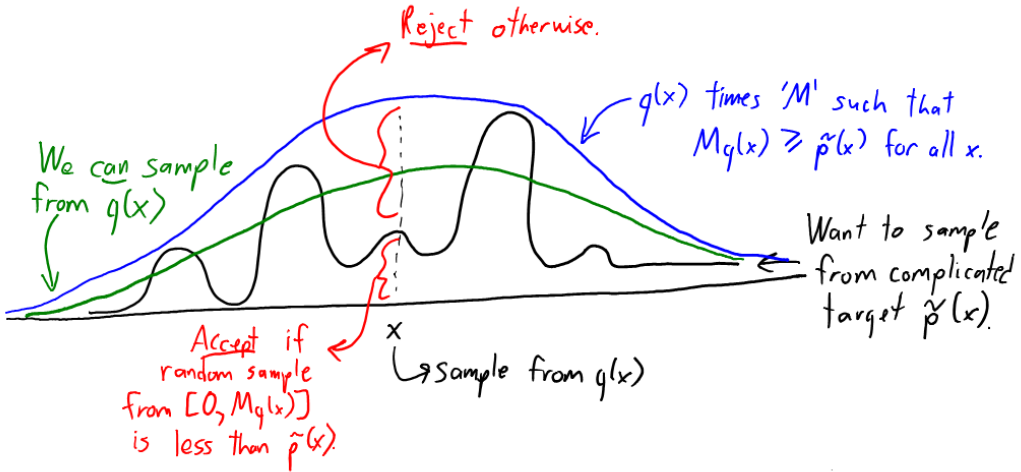
# General Rejection Sampling Algorithm

- General rejection sampling algorithm tries to "sample area under the graph":



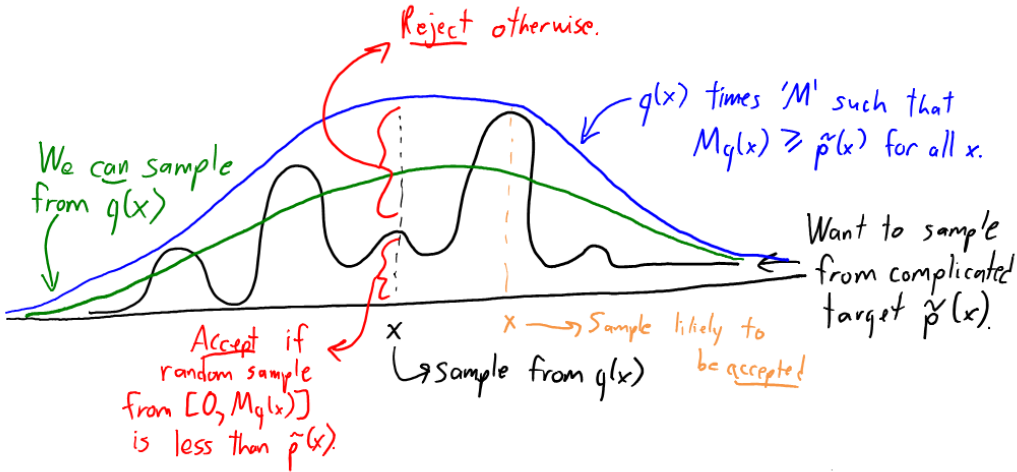
# General Rejection Sampling Algorithm

- General rejection sampling algorithm tries to "sample area under the graph":



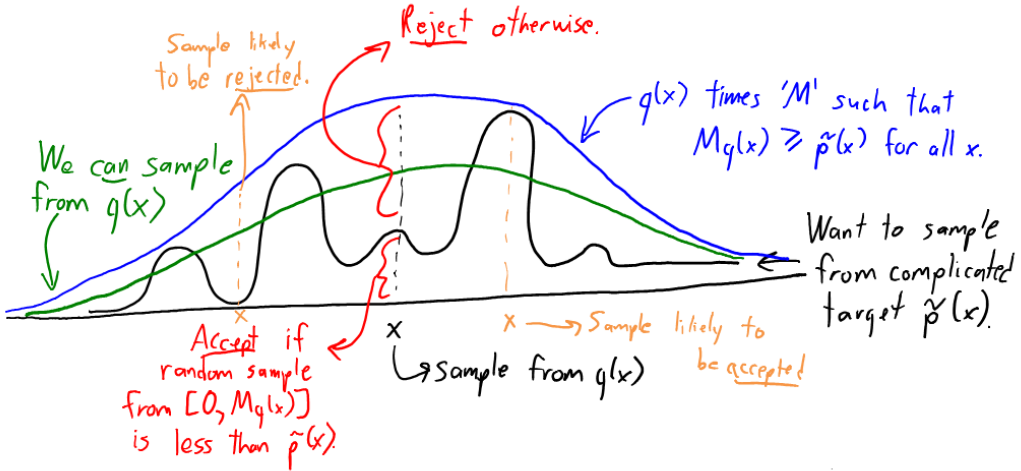
# General Rejection Sampling Algorithm

- General rejection sampling algorithm tries to "sample area under the graph":



# General Rejection Sampling Algorithm

- General rejection sampling algorithm tries to "sample area under the graph":



# General Rejection Sampling Algorithm

- Ingredients of the general rejection sampling algorithm:
  1. Ability to evaluate an unnormalized $\tilde{p}(x)$, so that $p(x) = \tilde{p}(x)/Z$
  2. A distribution $q$ that we can sample from
  3. An upper bound $M$ on $\tilde{p}(x)/q(x)$

- Rejection sampling algorithm:
  1. Sample $x$ from $q(x)$
  2. Keep the sample with probability $\tilde{p}(x)/(Mq(x))$:
     - Sample $u$ from $\mathrm{Unif}([0,1])$, keep the sample if $u \leq \tilde{p}(x) \,/\, (Mq(x))$
- The accepted samples will be from $p(x)$, as long as $M$ is a valid upper bound

- Then can use the accepted samples in Monte Carlo:

$$\mathop{\mathbb{E}}_{x\sim p} f(x) \approx \frac{1}{\sum_{i=1}^{m} \mathbb{1}\left(\text{accepted } x^{(i)}\right)} \sum_{i=1}^{m} \mathbb{1}\left(\text{accepted } x^{(i)}\right) f\left(x^{(i)}\right)$$

# General Rejection Sampling Algorithm

- For Bayesian logistic regression, we could propose samples from the prior:

$$\tilde{p}(w \mid \mathbf{X}, \mathbf{y}) = p(\mathbf{y} \mid \mathbf{X}, w)\, p(w) \qquad q(w) = p(w)$$

$$\frac{\tilde{p}(w \mid \mathbf{y}, \mathbf{X})}{q(w)} = \frac{p(\mathbf{y} \mid \mathbf{X}, w) p(w)}{p(w)} = p(\mathbf{y} \mid \mathbf{X}, w) \leq 1$$

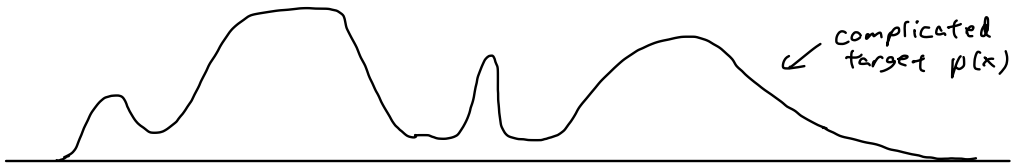
  - Recall $\mathbf{y}$ is discrete here, so $p(\mathbf{y} \mid \mathbf{X}, w) \leq 1$: we can use $M = 1$
  - $w$ sampled from prior would tend to be kept if they explain the data well

- Drawbacks of rejection sampling:
  - You need to know a bound $M$ on $\tilde{p}(x)/q(x)$ (may be hard/impossible to find)
    - If $x$ is unbounded and $p$ has heavier tails than $q$, no $M$ exists
  - You may reject a large number of samples
    - Most samples are rejected for high-dimensional complex distributions, or if $q$ is bad

# Outline

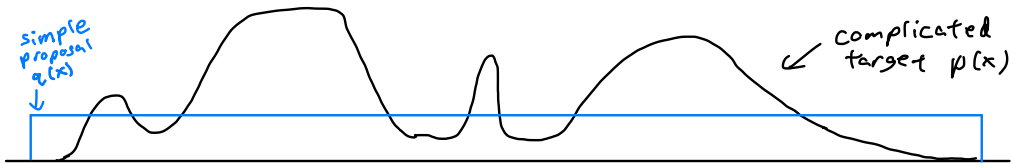
# Alternate approach: importance sampling

- Instead of rejection, importance sampling re-weights $q$ samples to look like $p$



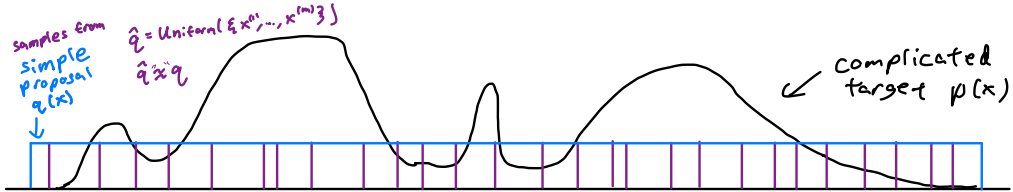
# Alternate approach: importance sampling

- Instead of rejection, importance sampling re-weights $q$ samples to look like $p$



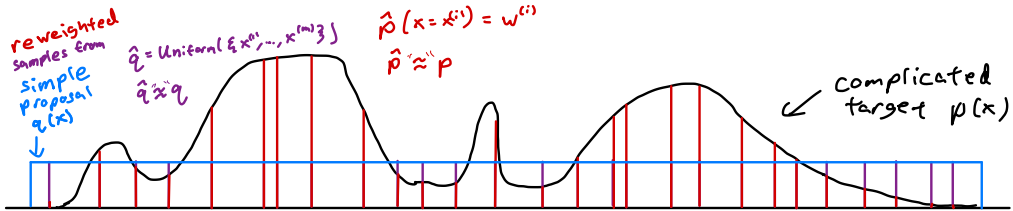
# Alternate approach: importance sampling

- Instead of rejection, importance sampling re-weights $q$ samples to look like $p$



# Alternate approach: importance sampling

- Instead of rejection, importance sampling re-weights $q$ samples to look like $p$



# Alternate approach: importance sampling

- Instead of rejection, importance sampling re-weights $q$ samples to look like $p$
- Derivation:

$$\begin{aligned}\mathop{\mathbb{E}}_{x\sim p}[f(x)] &= \int p(x)f(x)\,\mathrm{d}x \\ &= \int q(x)\frac{p(x)}{q(x)}f(x)\,\mathrm{d}x \\ &= \mathop{\mathbb{E}}_{x\sim q}\left[\frac{p(x)}{q(x)}f(x)\right] \approx \frac{1}{n}\sum_{i=1}^{n}\frac{p(x^{(i)})}{q(x^{(i)})}f(x^{(i)}),\end{aligned}$$

  using a Monte Carlo approximation with IID samples from $q$
  - Replace integral with a sum for discrete distributions
- We can sample from $q$, but reweight by $p(x)/q(x)$ to compute expectation
- Only assumption is that for all $x$ with nonzero $p$, $q$ is also nonzero

# Self-Normalized Importance Sampling

- What if we only have $\tilde{p}$, with $p(x) = \tilde{p}(x)/Z$?

$$\mathop{\mathbb{E}}_{x\sim p}[f(x)] = \int p(x) f(x)\,\mathrm{d}x = \frac{1}{Z}\int q(x)\frac{\tilde{p}(x)}{q(x)} f(x)\,\mathrm{d}x$$
$$= \frac{\mathbb{E}_{x\sim q}\left[\frac{\tilde{p}(x)}{q(x)} f(x)\right]}{\int \tilde{p}(x)\,\mathrm{d}x} = \frac{\mathbb{E}_{x\sim q}\left[\frac{\tilde{p}(x)}{q(x)} f(x)\right]}{\int q(x)\frac{\tilde{p}(x)}{q(x)}\,\mathrm{d}x} = \frac{\mathbb{E}_{x\sim q}\left[\frac{\tilde{p}(x)}{q(x)} f(x)\right]}{\mathbb{E}_{x\sim q}\left[\frac{\tilde{p}(x)}{q(x)}\right]}$$

- Can use Monte Carlo estimator based on $m$ samples from $q$:

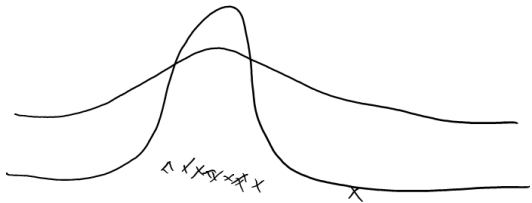
$$\mathop{\mathbb{E}}_{x\sim p}[f(x)] \approx \frac{\frac{1}{n}\sum_{i=1}^{m}\frac{\tilde{p}(x^{(i)})}{q(x^{(i)})} f(x^{(i)})}{\frac{1}{m}\sum_{i=1}^{m}\frac{\tilde{p}(x^{(i)})}{q(x^{(i)})}}$$

  - Weighted mean, normalized by $\tilde{p}(x^{(i)})/q(x^{(i)})$
  - Biased estimator: $\mathbb{E}\,\frac{1}{\hat{Z}} > \frac{1}{Z}$ for non-constant distributions (Jensen's inequality)

# Importance Sampling

- Importance sampling is only efficient if $q$ is close to $p$
- Otherwise, weights will be huge for a small number of samples
  - Even though unbiased, variance can be huge

- Can be problematic if $q$ has lighter "tails" than $p$:
  - You rarely sample the tails, so those samples get huge weights

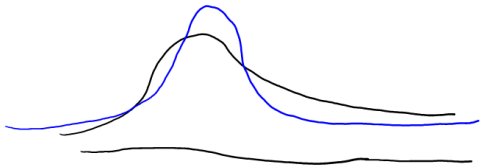


- As with rejection sampling, does not tend to work well in high dimensions
  - There's room, though, to cleverly design $q$
    - e.g. "alternate between sampling two Gaussians with different variances"

# Outline

# Variational Inference Illustration

- Approximate non-Gaussian $p$ by a Gaussian $q$:



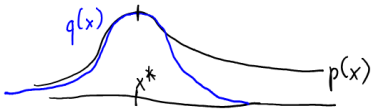
- Variational methods try to find simple distribution $q$ that is closest to target $p$
- Unlike Monte Carlo, does not converge to true solution
  - A Gaussian may not be able to perfectly model posterior
- Variational methods quickly give an approximate solution
  - Sometimes all we need
  - Sometimes, approximation is better than any reasonable amount of Monte Carlo!

# Laplace Approximation

- The classic, simplest variational method is the Laplace approximation
  1. Find the mode $x^*$,
  $$x^* \in \arg\max_x \log p(x)$$
  2. Compute the second-order Taylor expansion of $\log p(x)$ at $x^*$
  $$\log p(x) \approx \log p(x^*) + \underbrace{\nabla \log p(x^*)}_{0}^{\mathsf{T}}(x - x^*) + \frac{1}{2}(x - x^*)^{\mathsf{T}} \nabla^2 \log p(x^*)\,(x - x^*)$$
  3. Use the distribution $q$ agreeing with this log-likelihood, up to normalization:
  $$\log q(x) = \frac{1}{2}(x - x^*)[\nabla^2 \log p(x^*)](x - x^*) + \text{const}$$
  meaning the distribution $q$ is exactly $\mathcal{N}(x^*, [\nabla^2 \log p(x^*)]^{-1})$
     - Same approximation as used by Newton's method in optimization

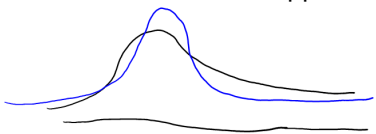
# Laplace Approximation

- Laplace approximation replaces a complicated $p$ with a Gaussian $q$
  - Centered at the mode, and agrees with 1st/2nd derivatives of log-likelihood there:




  - In the $n \to \infty$ limit, "nicely behaved" posteriors are asymptotically normal
    - Bernstein-von Mises theorem
- Now to compute $\mathbb{E}\, f(X)$, you only need to compute Gaussian integrals
  - Can do analytically, with linear algebra, for many $f$
    - If not, sampling from a Gaussian and doing Monte Carlo is easy
  - Fast: just maximize + find one Hessian
  - Bad approximation if posterior is heavy-tailed, multi-modal, skewed, etc

- It might not even give you the "best" Gaussian approximation:



# Summary

- Bayesian inference in non-conjugate models usually requires approximate inference
  - If we can sample from the posterior, can use Monte Carlo
  - Can find "best" approximation with variational methods

- Monte Carlo: estimate $\mathbb{E}_{X \sim p} f(X) \approx \frac{1}{n} \sum_{i=1}^{n} f(x^{(i)})$
  - Converges to true expectation asymptotically, unbiased estimate, variance $\mathrm{Var}(f(X))/n$
  - Need to be able to sample from $p$
  - Rejection sampling turns $q$ samples into $\tilde{p}$ samples, if you know $\max_x \tilde{p}(x)/q(x)$
    - Exact samples, but may be inefficient and hard to know $M$
  - Importance sampling re-weights $q$ samples to estimate expectations under $p$
    - Unbiased but can be high variance
    - Self-normalized IS for unnormalized $\tilde{p}$; same problems, plus bias
- Variational: choose $q \approx p$ and estimate $\mathbb{E}_{X \sim p} f(X) \approx \mathbb{E}_{X \sim q} f(X)$
  - Simplest approach: Laplace approximation, local normal approx at the mode

# Law of the Unconscious Statistician

bonus!

- These inequalities sometimes called "Law of the Unconscious Statistician":

$$\mathbb{E}[f(X)] = \sum_{x \in \mathcal{X}} f(x)p(x) \qquad \mathbb{E}[f(X)] = \int_{x \in \mathcal{X}} f(x)p(x)\mathrm{d}x$$

- Two explanations I've heard for "unconscious":
  - You can compute expectations without thinking
  - Or: people don't realize this is actually a theorem to prove, not a definition

$$\begin{aligned} Y &= f(X) \\ \mathbb{E}[Y] &= \sum_y y \Pr(Y = y) = \sum_y y \sum_{x : f(x) = y} p(x) = \sum_x f(x)p(x) \end{aligned}$$

# Monte Carlo as a stochastic gradient method

bonus!

Can view as SGD on $f(\hat{\mu}) = \frac{1}{n}\|\hat{\mu} - \mu\|^2$ with learning rate $\frac{1}{i+1}$:

$$\begin{aligned}
\hat{\mu}_n &= \hat{\mu}_{n-1} - \frac{1}{n}\left(\hat{\mu}_{n-1} - x^{(i)}\right) \\
&= \left(1 - \frac{1}{n}\right)\hat{\mu}_{n-1} + \frac{1}{n}x^{(i)} \\
&= \frac{n-1}{n}\left(\frac{1}{n-1}\sum_{i=1}^{n-1} x^{(i)}\right) + \frac{1}{n}x^{(i)} \\
&= \frac{1}{n}\sum_{i=1}^{n-1} x^{(i)} + \frac{1}{n}x^{(i)} \\
&= \frac{1}{n}\sum_{i=1}^{n} x^{(i)}
\end{aligned}$$