

CSCI 699 - ProbGen

# Probabilistic and Generative Models

Willie Neiswanger

# **Lecture 5 - Gradient-based MCMC and Variational Inference**

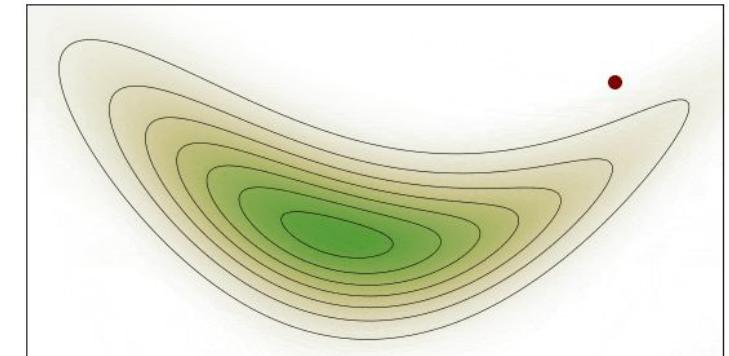
# Today

**Lecture:** Approximate inference algorithms in probabilistic models, including the classics – Markov chain Monte Carlo (MCMC), and variational inference (VI).

# Today

**Lecture:** Approximate inference algorithms in probabilistic models, including the classics – Markov chain Monte Carlo (MCMC), and variational inference (VI).

- Markov chain Monte Carlo (MCMC) methods, including:
  - MC integration, rejection/importance sampling.
  - Metropolis–Hastings algorithm, Gibbs sampling.
  - Gradient-based methods: Langevin Monte Carlo, Hamiltonian Monte Carlo.

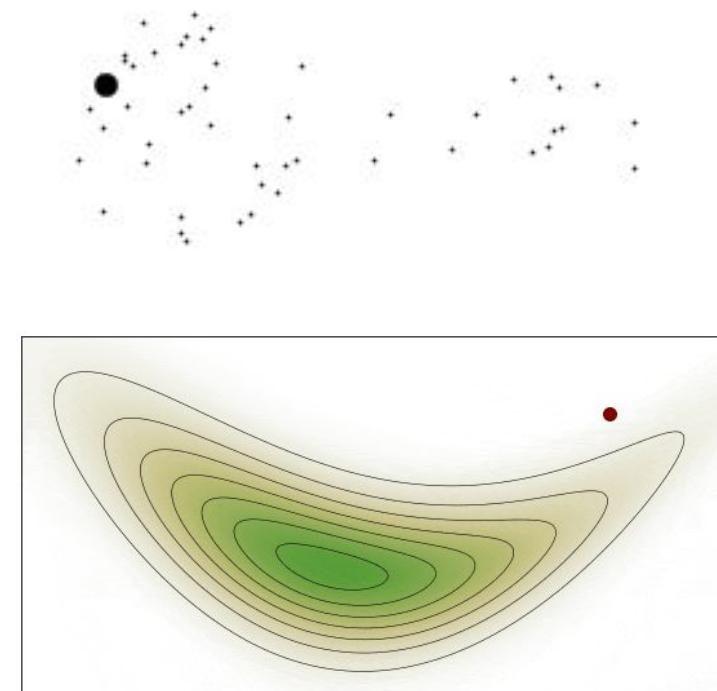


Source: "Creating animations with MCMC", Krepl 2018,  
Wikimedia commons,

# Today

**Lecture:** Approximate inference algorithms in probabilistic models, including the classics – Markov chain Monte Carlo (MCMC), and variational inference (VI).

- Markov chain Monte Carlo (MCMC) methods, including:
  - MC integration, rejection/importance sampling.
  - Metropolis–Hastings algorithm, Gibbs sampling.
  - Gradient-based methods: Langevin Monte Carlo, Hamiltonian Monte Carlo.
- Variational inference (VI) methods, including
  - Evidence lower bound (ELBO), Stochastic VI, BBVI.



Source: "Creating animations with MCMC", Krepl 2018,  
Wikimedia commons,

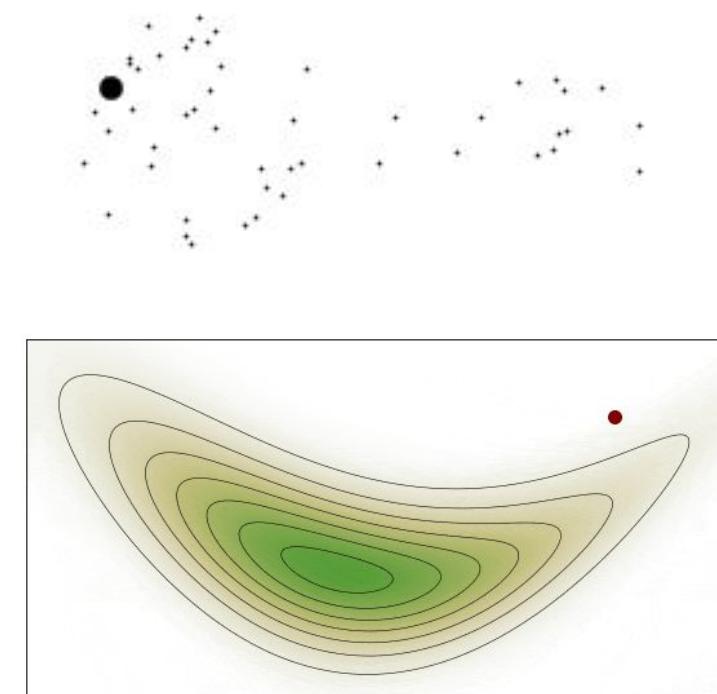
# Today

**Lecture:** Approximate inference algorithms in probabilistic models, including the classics – Markov chain Monte Carlo (MCMC), and variational inference (VI).

- Markov chain Monte Carlo (MCMC) methods, including:
  - MC integration, rejection/importance sampling.
  - Metropolis–Hastings algorithm, Gibbs sampling.
  - Gradient-based methods: Langevin Monte Carlo, Hamiltonian Monte Carlo.
- Variational inference (VI) methods, including
  - Evidence lower bound (ELBO), Stochastic VI, BBVI.

**After:**

- Project Pitches #2: Groups 6 - 9
- Paper Presentation (+ Roles) Sign Up



Source: "Creating animations with MCMC", Krepl 2018,  
Wikimedia commons,

# **Review: Sampling, Monte Carlo, Markov chains, and MCMC**

# Review – Last Class on Sampling

Last week we covered:

- Approximate inference via random sampling.
- Monte Carlo methods for sample-based expectations.
- Rejection Sampling.
- Importance Sampling.

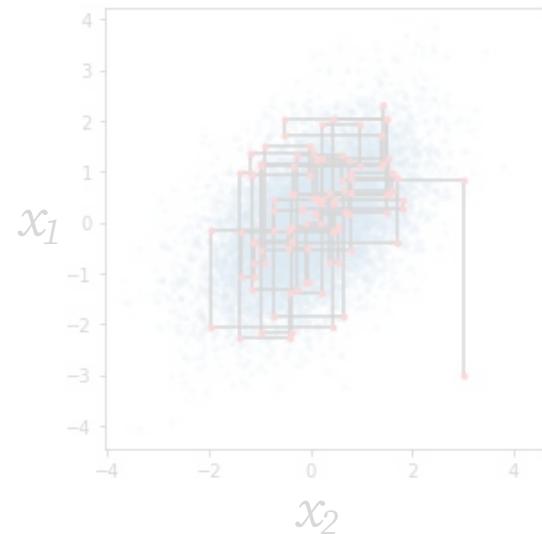
# Review – Key Types of Approximate Inference Algorithms

# Review – Key Types of Approximate Inference Algorithms

At a high level: two popular “types” of approximate inference algorithms:

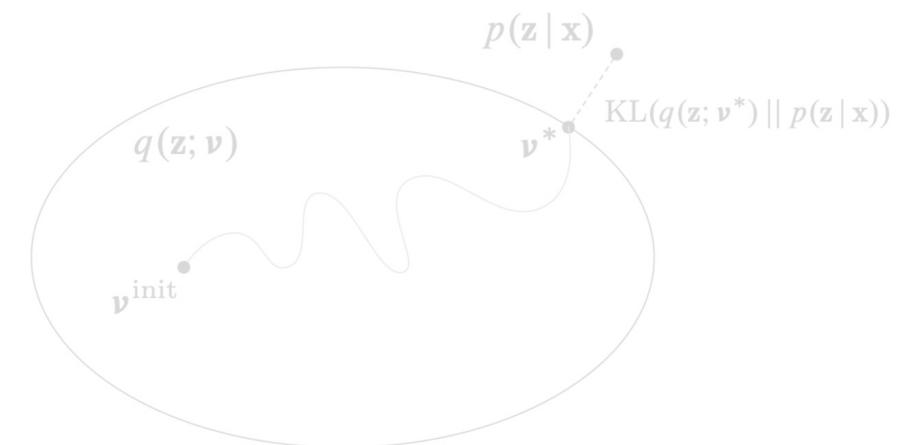
Markov chain Monte Carlo (MCMC)

E.g., Gibbs Sampling



Variational Inference (VI)

E.g., black-box variational inference

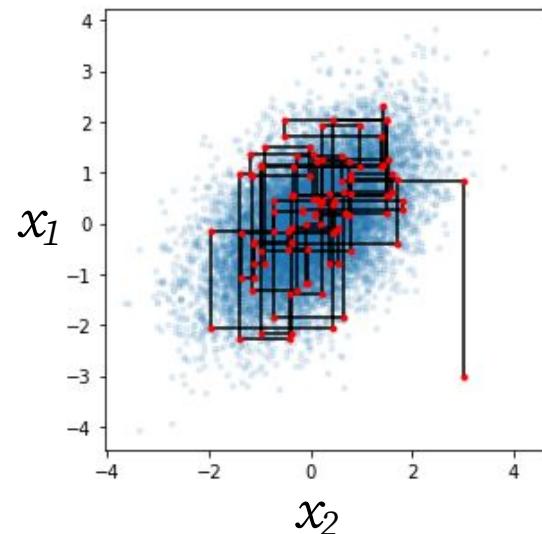


# Review – Key Types of Approximate Inference Algorithms

At a high level: two popular “types” of approximate inference algorithms:

## Markov chain Monte Carlo (MCMC)

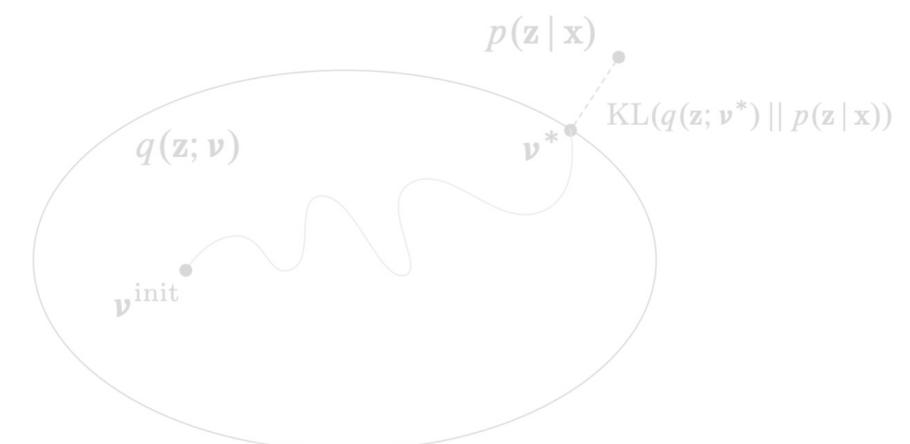
E.g., Gibbs Sampling



Source: Jessica Stringham, Gibbs Sampling in Python

## Variational Inference (VI)

E.g., black-box variational inference



Source: Dave Blei, "Variational Inference"

# Review – Key Types of Approximate Inference Algorithms

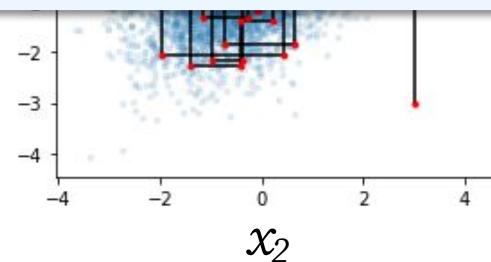
At a high level: two popular “types” of approximate inference algorithms:

## Markov chain Monte Carlo (MCMC)

E.g., Gibbs Sampling

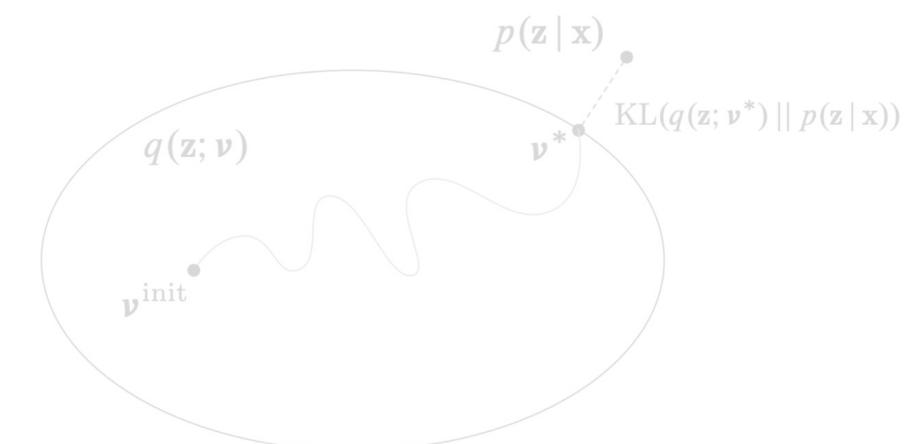


Perform inference by drawing samples from a distribution of interest.



## Variational Inference (VI)

E.g., black-box variational inference

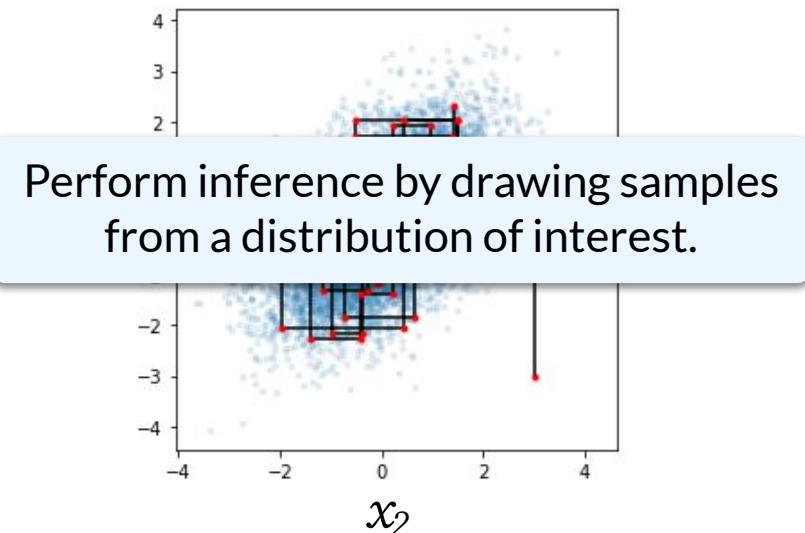


# Review – Key Types of Approximate Inference Algorithms

At a high level: two popular “types” of approximate inference algorithms:

## Markov chain Monte Carlo (MCMC)

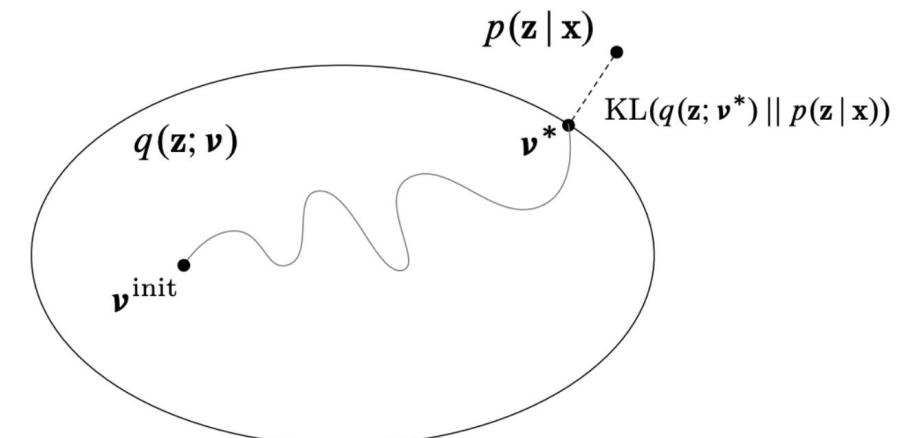
E.g., Gibbs Sampling



Source: Jessica Stringham, Gibbs Sampling in Python

## Variational Inference (VI)

E.g., black-box variational inference



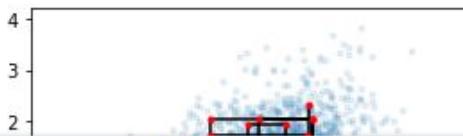
Source: Dave Blei, “Variational Inference”

# Review – Key Types of Approximate Inference Algorithms

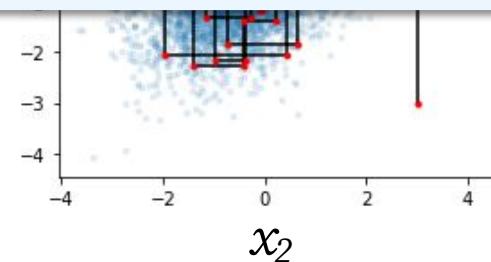
At a high level: two popular “types” of approximate inference algorithms:

## Markov chain Monte Carlo (MCMC)

E.g., Gibbs Sampling



Perform inference by drawing samples from a distribution of interest.



## Variational Inference (VI)

E.g., black-box variational inference

$$p(\mathbf{z} | \mathbf{x})$$
$$KL(q(\mathbf{z} | \mathbf{v}^*) || p(\mathbf{z} | \mathbf{x}))$$

Perform inference by optimizing over a space of tractable distributions.

$\mathbf{v}^{\text{init}}$

# Review – Monte Carlo (MC) Estimation

## Review – Monte Carlo (MC) Estimation

Samples from a distribution allow us to estimate expectations/integrals of the form:

$$\mathbb{E}_{x \sim p(x)} [g(x)] = \int_{\mathcal{X}} g(x)p(x)dx$$

$g(x)$  is some arbitrary  
“test function”.

## Review – Monte Carlo (MC) Estimation

Samples from a distribution allow us to estimate expectations/integrals of the form:

$$\mathbb{E}_{x \sim p(x)} [g(x)] = \int_{\mathcal{X}} g(x)p(x)dx$$

$g(x)$  is some arbitrary  
“test function”.

In general, if  $g(x)$  does not have specific form well-suited to  $p(x)$  (or matching structure of Bayes net associated with  $p(x)$ )  $\Rightarrow$  not possible to perform analytically.

## Review – Monte Carlo (MC) Estimation

Samples from a distribution allow us to estimate expectations/integrals of the form:

$$\mathbb{E}_{x \sim p(x)} [g(x)] = \int_{\mathcal{X}} g(x)p(x)dx$$

$g(x)$  is some arbitrary  
“test function”.

In general, if  $g(x)$  does not have specific form well-suited to  $p(x)$  (or matching structure of Bayes net associated with  $p(x)$ )  $\Rightarrow$  not possible to perform analytically.

Instead, we can approximate it using a large number of samples from  $p(x)$ .  
 $\Rightarrow$  Algorithms such as this are referred to as **Monte Carlo methods**.

## Review – Monte Carlo (MC) Estimation

Samples from a distribution allow us to estimate expectations/integrals of the form:

$$\mathbb{E}_{x \sim p(x)} [g(x)] = \int_{\mathcal{X}} g(x)p(x)dx$$

$g(x)$  is some arbitrary  
“test function”.

## Review – Monte Carlo (MC) Estimation

Samples from a distribution allow us to estimate expectations/integrals of the form:

$$\mathbb{E}_{x \sim p(x)} [g(x)] = \int_{\mathcal{X}} g(x)p(x)dx$$

$g(x)$  is some arbitrary  
“test function”.

Monte Carlo sampling approximates this via:

$$\mathbb{E}_{x \sim p(x)} [g(x)] \approx \frac{1}{T} \sum_{t=1}^T g(x^{(t)}), \quad x^{(1)}, \dots, x^{(T)} \sim p(x)$$

where samples  $x^{(1)}, \dots, x^{(T)} \sim p(x)$  are drawn i.i.d. from the distribution  $p(x)$ .

## Review – Monte Carlo (MC) Estimation

Samples from a distribution allow us to estimate expectations/integrals of the form:

$$\mathbb{E}_{x \sim p(x)} [g(x)] = \int_{\mathcal{X}} g(x)p(x)dx$$

$g(x)$  is some arbitrary  
“test function”.

Monte Carlo sampling approximates this via:

A “sample average”.

$$\mathbb{E}_{x \sim p(x)} [g(x)] \approx \frac{1}{T} \sum_{t=1}^T g(x^{(t)}), \quad x^{(1)}, \dots, x^{(T)} \sim p(x)$$

where samples  $x^{(1)}, \dots, x^{(T)} \sim p(x)$  are drawn i.i.d. from the distribution  $p(x)$ .

# Review – Rejection Sampling

# Review – Rejection Sampling

Intuition:

Compute the area of a region  $R$  by sampling from a larger region with a known area and recording the fraction of samples that falls within  $R$ .

# Review – Rejection Sampling

More formally:

# Review – Rejection Sampling

This method works even if  
we only know the  
unnormalized PDF  $\tilde{p}(x)$ .

More formally:

Suppose we have a complex PDF, with unknown normalization,  $p(x) = \frac{1}{Z} \tilde{p}(x)$ .

# Review – Rejection Sampling

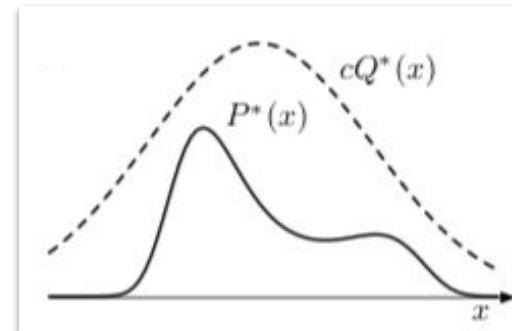
This method works even if we only know the unnormalized PDF  $\tilde{p}(x)$ .

More formally:

Suppose we have a complex PDF, with unknown normalization,  $p(x) = \frac{1}{Z} \tilde{p}(x)$ .

We can propose, and sample from, a (more tractable) *proposal density*, denoted  $q(x)$ , if we assume that we know a value of a constant  $c$  such that:

$$cq(x) > \tilde{p}(x), \quad \forall x$$



Source: Jesse Bettencourt, Probabilistic Machine Learning

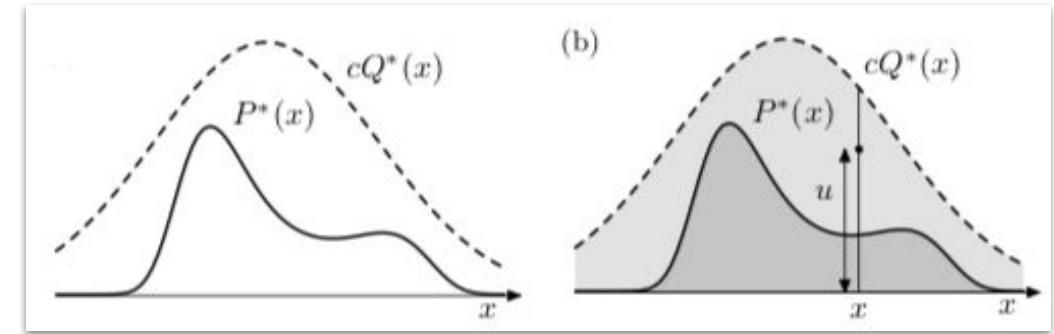
## Review – Rejection Sampling

Then you can do the following:

# Review – Rejection Sampling

Then you can do the following:

- Draw two random numbers:
  - First:  $x \sim q(x)$ .
  - Second:  $u \sim \text{Uniform}([0, cq(x)])$

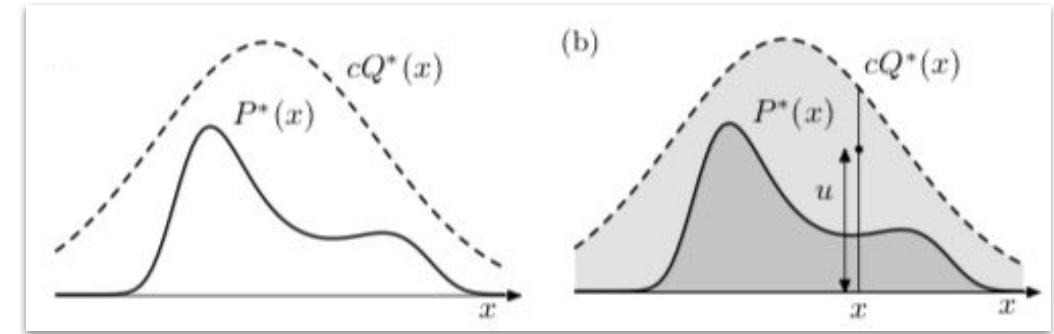


Source: Jesse Bettencourt, Probabilistic Machine Learning

# Review – Rejection Sampling

Then you can do the following:

- Draw two random numbers:
  - First:  $x \sim q(x)$ .
  - Second:  $u \sim \text{Uniform}([0, cq(x)])$
- If  $u \leq \tilde{p}(x)$ :
  - Keep  $x$ .
- Otherwise, reject  $x$ .

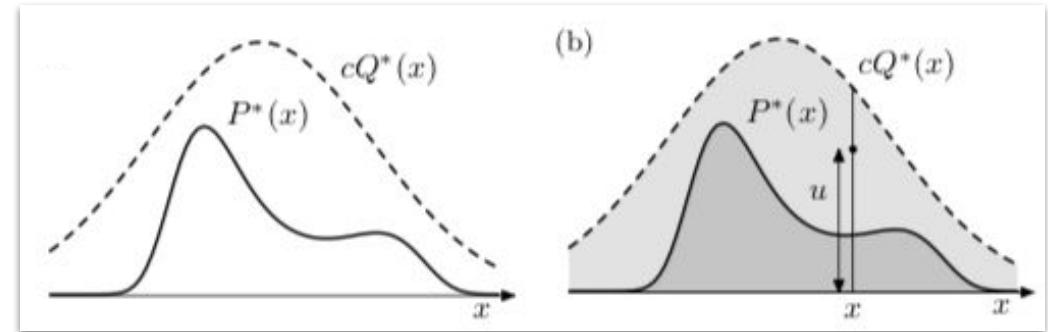


Source: Jesse Bettencourt, Probabilistic Machine Learning

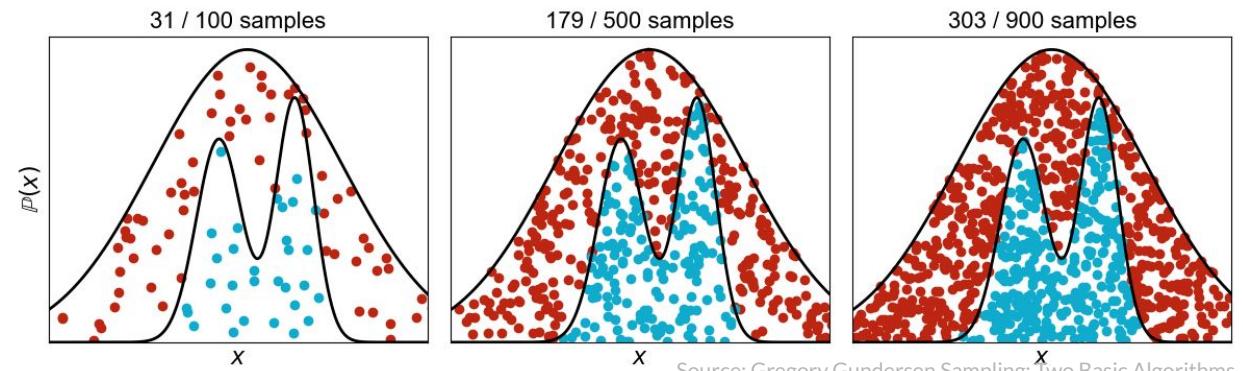
# Review – Rejection Sampling

Then you can do the following:

- Draw two random numbers:
  - First:  $x \sim q(x)$ .
  - Second:  $u \sim \text{Uniform}([0, cq(x)])$
- If  $u \leq \tilde{p}(x)$ :
  - Keep  $x$ .
- Otherwise, reject  $x$ .



Source: Jesse Bettencourt, Probabilistic Machine Learning



Source: Gregory Gundersen, Sampling: Two Basic Algorithms

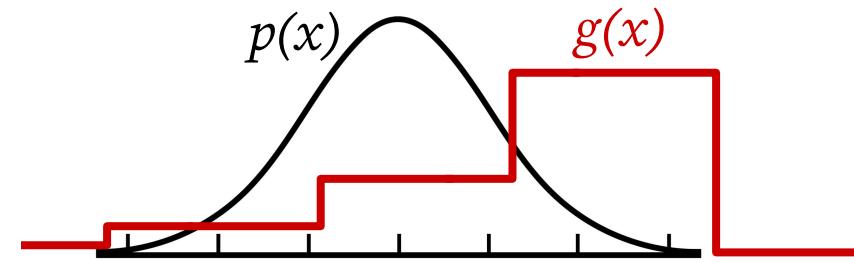
This samples uniformly over AUC  $\Rightarrow x$  is drawn proportional to PDF!

# Review – Importance Sampling

# Review – Importance Sampling

Intuition:

- 1. Sample from a proposal distribution  $q(x)$  (ideally with  $q(x)$  roughly proportional to  $g(x) \cdot p(x)$ ).
- 2. Reweight the samples (in a principled way), so that their sum still approximates the desired integral.



## Review – Importance Sampling

More formally, suppose we want to estimate the expectation  $\mathbb{E}_{x \sim p(x)} [g(x)]$ .

## Review – Importance Sampling

More formally, suppose we want to estimate the expectation  $\mathbb{E}_{x \sim p(x)} [g(x)]$ .

(However, suppose  $p(x)$  is complex and thus difficult to sample from and form an MC estimate).

# Review – Importance Sampling

More formally, suppose we want to estimate the expectation  $\mathbb{E}_{x \sim p(x)} [g(x)]$ .

We can rewrite this expectation as:

(Assuming discrete RVs, but same argument holds for continuous RVs.)

$$\begin{aligned}\mathbb{E}_{x \sim p(x)} [g(x)] &= \sum_x g(x)p(x) \\ &= \sum_x g(x) \frac{p(x)}{q(x)} q(x) \\ &= \mathbb{E}_{x \sim q(x)} [g(x)w(x)], \quad \text{where } w(x) = \frac{p(x)}{q(x)} \\ &\approx \frac{1}{T} \sum_{t=1}^T g(x^{(t)})w(x^{(t)}), \quad \text{where } x^{(1)}, \dots, x^{(T)} \stackrel{\text{iid}}{\sim} q(x)\end{aligned}$$

# Review – Importance Sampling

More formally, suppose we want to estimate the expectation  $\mathbb{E}_{x \sim p(x)} [g(x)]$ .

We can rewrite this expectation as:

$$\begin{aligned}\mathbb{E}_{x \sim p(x)} [g(x)] &= \sum_x g(x)p(x) && \text{Definition of expected value} \\ &= \sum_x g(x) \frac{p(x)}{q(x)} q(x) \\ &= \mathbb{E}_{x \sim q(x)} [g(x)w(x)], \quad \text{where } w(x) = \frac{p(x)}{q(x)} \\ &\approx \frac{1}{T} \sum_{t=1}^T g(x^{(t)})w(x^{(t)}), \quad \text{where } x^{(1)}, \dots, x^{(T)} \stackrel{\text{iid}}{\sim} q(x)\end{aligned}$$

## Review – Importance Sampling

More formally, suppose we want to estimate the expectation  $\mathbb{E}_{x \sim p(x)} [g(x)]$ .

We can rewrite this expectation as:

$$\begin{aligned}\mathbb{E}_{x \sim p(x)} [g(x)] &= \sum_x g(x)p(x) \\ &= \sum_x g(x) \frac{p(x)}{q(x)} q(x) \quad \text{Multiply by } 1 = q(x)/q(x) \\ &= \mathbb{E}_{x \sim q(x)} [g(x)w(x)], \quad \text{where } w(x) = \frac{p(x)}{q(x)} \\ &\approx \frac{1}{T} \sum_{t=1}^T g(x^{(t)})w(x^{(t)}), \quad \text{where } x^{(1)}, \dots, x^{(T)} \stackrel{\text{iid}}{\sim} q(x)\end{aligned}$$

## Review – Importance Sampling

More formally, suppose we want to estimate the expectation  $\mathbb{E}_{x \sim p(x)} [g(x)]$ .

We can rewrite this expectation as:

$$\begin{aligned}\mathbb{E}_{x \sim p(x)} [g(x)] &= \sum_x g(x)p(x) \\ &= \sum_x g(x) \frac{p(x)}{q(x)} q(x)\end{aligned}$$

$$= \mathbb{E}_{x \sim q(x)} [g(x)w(x)], \quad \text{where } w(x) = \frac{p(x)}{q(x)}$$

$$\approx \frac{1}{T} \sum_{t=1}^T g(x^{(t)})w(x^{(t)}), \quad \text{where } x^{(1)}, \dots, x^{(T)} \stackrel{\text{iid}}{\sim} q(x)$$

Rewrite as  
expectation  
w.r.t.  $q(x)$

## Review – Importance Sampling

More formally, suppose we want to estimate the expectation  $\mathbb{E}_{x \sim p(x)} [g(x)]$ .

We can rewrite this expectation as:

$$\begin{aligned}\mathbb{E}_{x \sim p(x)} [g(x)] &= \sum_x g(x)p(x) \\ &= \sum_x g(x) \frac{p(x)}{q(x)} q(x) \\ &= \mathbb{E}_{x \sim q(x)} [g(x)w(x)], \quad \text{where } w(x) = \frac{p(x)}{q(x)} \\ &\approx \frac{1}{T} \sum_{t=1}^T g(x^{(t)})w(x^{(t)}), \quad \text{where } x^{(1)}, \dots, x^{(T)} \stackrel{\text{iid}}{\sim} q(x)\end{aligned}$$

MC  
Estimate!

## Review – Importance Sampling

More formally, suppose we want to estimate the expectation  $\mathbb{E}_{x \sim p(x)} [g(x)]$ .

We can rewrite this expectation as:

$$\begin{aligned}\mathbb{E}_{x \sim p(x)} [g(x)] &= \sum_x g(x)p(x) \\ &= \sum_x g(x) \frac{p(x)}{q(x)} q(x) \\ &= \mathbb{E}_{x \sim q(x)} [g(x)w(x)], \quad \text{where } w(x) = \frac{p(x)}{q(x)}\end{aligned}$$

Sometimes referred  
to as an *IS estimate*.

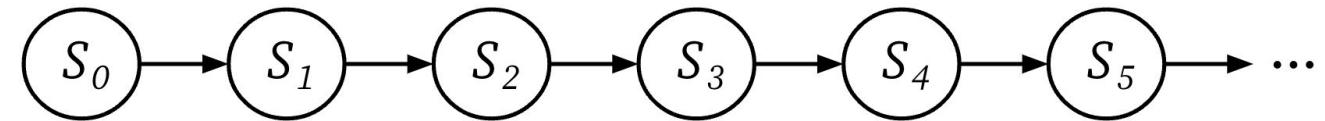
$$\Rightarrow \approx \frac{1}{T} \sum_{t=1}^T g(x^{(t)})w(x^{(t)}), \quad \text{where } x^{(1)}, \dots, x^{(T)} \stackrel{\text{iid}}{\sim} q(x)$$

MC  
Estimate!

# Review – Markov Chains

# Review – Markov Chains

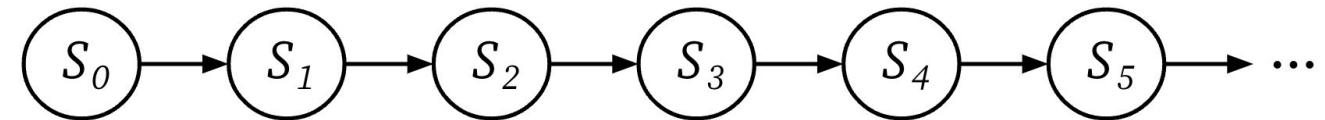
Review on Markov chains:



- A (*discrete-time*) *Markov chain* is a sequence of random variables:

# Review – Markov Chains

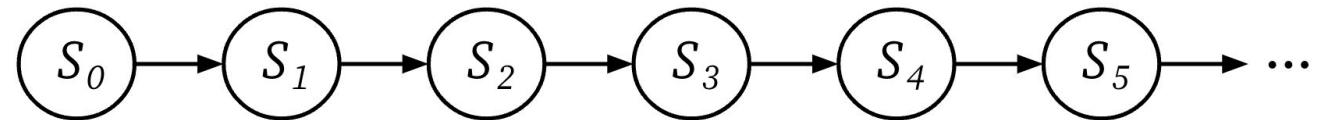
Review on Markov chains:



- A (*discrete-time*) *Markov chain* is a sequence of random variables:
- The initial state is distributed according to a probability  $p(S_0)$ .

# Review – Markov Chains

Review on Markov chains:

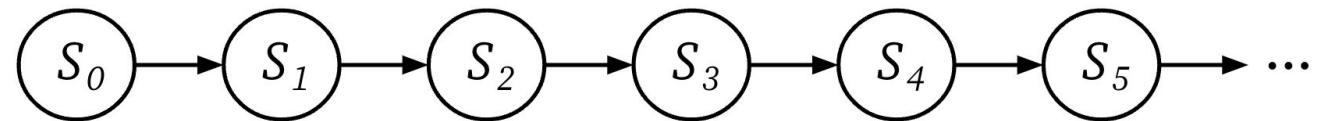


- A (*discrete-time*) *Markov chain* is a sequence of random variables:
- The initial state is distributed according to a probability  $p(S_0)$ .
- All subsequent states are conditionally distributed according to:  $p(S_i \mid S_{i-1})$ .

*Transition matrix or  
Transition distribution*

# Review – Markov Chains

Review on Markov chains:

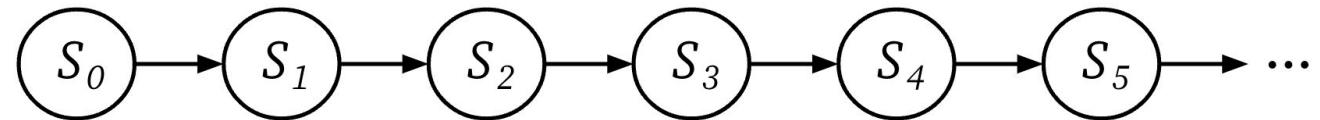


- A (*discrete-time*) *Markov chain* is a sequence of random variables:
- The initial state is distributed according to a probability  $p(S_0)$ .
- All subsequent states are conditionally distributed according to:  $p(S_i \mid S_{i-1})$ .
  - Where the conditional probability is same for each  $i \dots$

*Transition matrix or  
Transition distribution*

# Review – Markov Chains

Review on Markov chains:



- A (*discrete-time*) *Markov chain* is a sequence of random variables:
- The initial state is distributed according to a probability  $p(S_0)$ .
- All subsequent states are conditionally distributed according to:  $p(S_i \mid S_{i-1})$ .
  - Where the conditional probability is same for each  $i \dots$
- $\Rightarrow$  the transition probabilities at any time depend only on the given state and not on the history of how we got there (“Markov assumption”).

*Transition matrix or  
Transition distribution*

“Given the present, the future does not depend on the past”

# Review – Markov Chains

Stationary Distributions.

## Review – Markov Chains

Stationary Distributions.

Suppose we write the initial state distribution  $p(S_0)$  as a vector  $p_0$ .

And the transition distribution as a matrix  $T_{ij} = p(S_i = i \mid S_{i-1} = j)$ .

## Review – Markov Chains

### Stationary Distributions.

Suppose we write the initial state distribution  $p(S_0)$  as a vector  $p_0$ .

And the transition distribution as a matrix  $T_{ij} = p(S_i = i \mid S_{i-1} = j)$ .

⇒ We can write the probability  $p_t$  of ending up in a given state as:

$$p_t = T^t p_0$$

$T^t$  is a matrix power  
(i.e., apply the matrix operator  $t$  times)

The limit  $\pi = \lim_{t \rightarrow \infty} p_t$  (when it exists) is called a **stationary distribution** of the MC.

# Review – Markov Chains

Stationary Distributions.

Sufficient conditions for a distribution  $\pi$  to be a stationary distribution:

# Review – Markov Chains

Stationary Distributions.

Sufficient conditions for a distribution  $\pi$  to be a stationary distribution:

- Detailed Balance:  $\pi_i T_{ij} = \pi_j T_{ji}$  for all  $i, j$

“The flow of probability between  
any two states is balanced”  
(after sampling from stationary dist.)

# Review – Markov Chains

## Stationary Distributions.

Sufficient conditions for a distribution  $\pi$  to be a stationary distribution:

- Detailed Balance:  $\pi_i T_{ij} = \pi_j T_{ji}$  for all  $i, j$

“The flow of probability between any two states is balanced”  
(after sampling from stationary dist.)

- Ergodicity:

*Irreducibility & Aperiodicity*

“It’s possible to transition from any state to any other state”

“Avoids transitioning between sets of states in a cycle”

# **Markov Chain Monte Carlo (MCMC)**

# Markov Chain Monte Carlo (MCMC) – Overview

High level idea of MCMC:

# Markov Chain Monte Carlo (MCMC) – Overview

High level idea of MCMC:

- Given a probability distribution of interest,  $p(x)$ .



E.g., posterior distribution  
or marginal distribution of a  
probability model

# Markov Chain Monte Carlo (MCMC) – Overview

High level idea of MCMC:

- Given a probability distribution of interest,  $p(x)$ .
- Construct a Markov chain.



E.g., posterior distribution  
or marginal distribution of a  
probability model

# Markov Chain Monte Carlo (MCMC) – Overview

High level idea of MCMC:

- Given a probability distribution of interest,  $p(x)$ .
- Construct a Markov chain.
- Whose states are assignments of all variables  $x$  (jointly) in the probability distribution.



E.g., posterior distribution  
or marginal distribution of a  
probability model

# Markov Chain Monte Carlo (MCMC) – Overview

High level idea of MCMC:

- Given a probability distribution of interest,  $p(x)$ .
- Construct a Markov chain.
- Whose states are assignments of all variables  $x$  (jointly) in the probability distribution.
- And whose stationary distribution equals the probability distribution  $p(x)$ .



E.g., posterior distribution  
or marginal distribution of a  
probability model

# Markov Chain Monte Carlo (MCMC) – Overview

High level idea of MCMC:

- Given a probability distribution of interest,  $p(x)$ .
- Construct a Markov chain.
- Whose states are assignments of all variables  $x$  (jointly) in the probability distribution.
- And whose stationary distribution equals the probability distribution  $p(x)$ .



E.g., posterior distribution or marginal distribution of a probability model

⇒ then we can sample from this Markov chain, and iteratively get closer to samples from  $p(x)$  (as we approach the stationary distribution).

# Markov Chain Monte Carlo (MCMC) – Overview

Thus, in an MCMC algorithm you typically perform the following two steps:  
(Given as input: a transition distribution  $T$ , and an initial assignment of variables,  $x_0$  ).

# Markov Chain Monte Carlo (MCMC) – Overview

Thus, in an MCMC algorithm you typically perform the following two steps:

(Given as input: a transition distribution  $T$ , and an initial assignment of variables,  $x_0$ ).

- 1. Sample from the Markov chain, starting with  $x_0$ , for  $B$  *burn-in* steps, which we discard.
- 2. Sample from the Markov chain for  $N$  *sampling steps*, and keep each of these samples.

# Markov Chain Monte Carlo (MCMC) – Overview

Thus, in an MCMC algorithm you typically perform the following two steps:

(Given as input: a transition distribution  $T$ , and an initial assignment of variables,  $x_0$ ).

- 1. Sample from the Markov chain, starting with  $x_0$ , for  $B$  *burn-in* steps, which we discard.
- 2. Sample from the Markov chain for  $N$  *sampling steps*, and keep each of these samples.

Assuming that  $B$  is sufficiently large, the latter set of  $N$  samples will approximately be drawn from  $p(x)$ .

# Markov Chain Monte Carlo (MCMC) – Algorithms

We'll review through a few different MCMC algorithms following these steps:

# Markov Chain Monte Carlo (MCMC) – Algorithms

We'll review through a few different MCMC algorithms following these steps:

- Metropolis-Hastings (MH) – a “core” MCMC algorithm
- Gibbs Sampling – can be viewed as a special case of MH.
- Langevin Monte Carlo – can be viewed as a *gradient-based* case of MH.
- Hamiltonian Monte Carlo – can be viewed as a *better gradient-based* case of MH.

# **MCMC - Metropolis-Hastings and Gibbs Sampling**

# Metropolis-Hastings (MH)

# Metropolis-Hastings (MH)

The **Metropolis Hastings Algorithm** constructs a transition distribution  $T(x' | x)$  using two components:

- A user-specified *transition kernel*  $Q(x' | x)$ .
- An *acceptance probability* for moves proposed by  $Q$ , defined to be:

$$A(x' | x) = \min \left( 1, \frac{p(x')Q(x | x')}{p(x)Q(x' | x)} \right)$$

Often called the  
*acceptance ratio*.

# Metropolis-Hastings (MH)

The **Metropolis Hastings Algorithm** constructs a transition distribution  $T(x' | x)$  using two components:

- A user-specified *transition kernel*  $Q(x' | x)$ .
- An *acceptance probability* for moves proposed by  $Q$ , defined to be:

$$A(x' | x) = \min \left( 1, \frac{p(x')Q(x | x')}{p(x)Q(x' | x)} \right)$$

Often called the  
*acceptance ratio*.

At each step of the MH, we sample  $x' \sim Q(x' | x)$  and then we either *accept* it with probability  $\alpha = A(x' | x)$  or reject it and remain at our previous state.

# Metropolis-Hastings (MH)

The **Metropolis Hastings Algorithm** constructs a transition distribution  $T(x' | x)$  using two components:

- A user-specified *transition kernel*  $Q(x' | x)$ .
- An *acceptance probability* for moves proposed by  $Q$ , defined to be:

$$A(x' | x) = \min \left( 1, \frac{\tilde{p}(x')Q(x | x')}{\tilde{p}(x)Q(x' | x)} \right)$$

Often called the  
*acceptance ratio*.

Important:  $p(x), p(x')$  can be unnormalized PDF!  $\Rightarrow$  easier to apply.

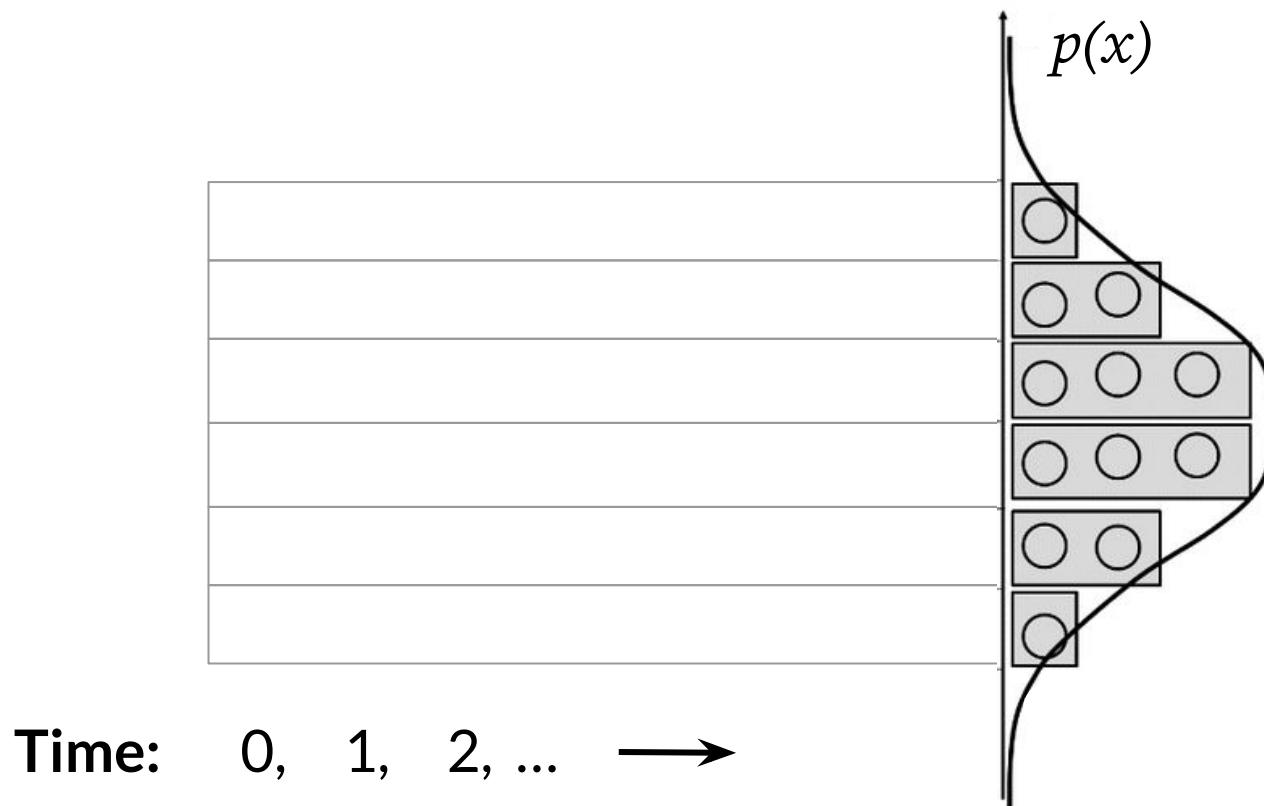
At each step of the MH, we sample  $x' \sim Q(x' | x)$  and then we either *accept* it with probability  $\alpha = A(x' | x)$  or reject it and remain at our previous state.

# Metropolis-Hastings (MH)

**Intuition:**

# Metropolis-Hastings (MH)

Intuition:

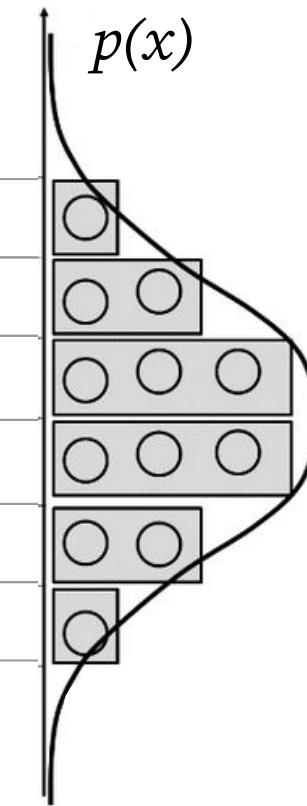
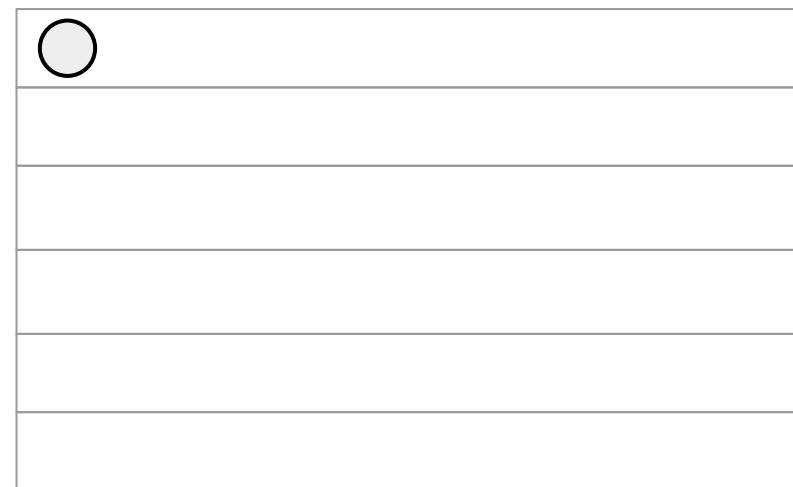


Source: Wikipedia, "Metropolis-Hastings Algorithm"

# Metropolis-Hastings (MH)

**Intuition:**

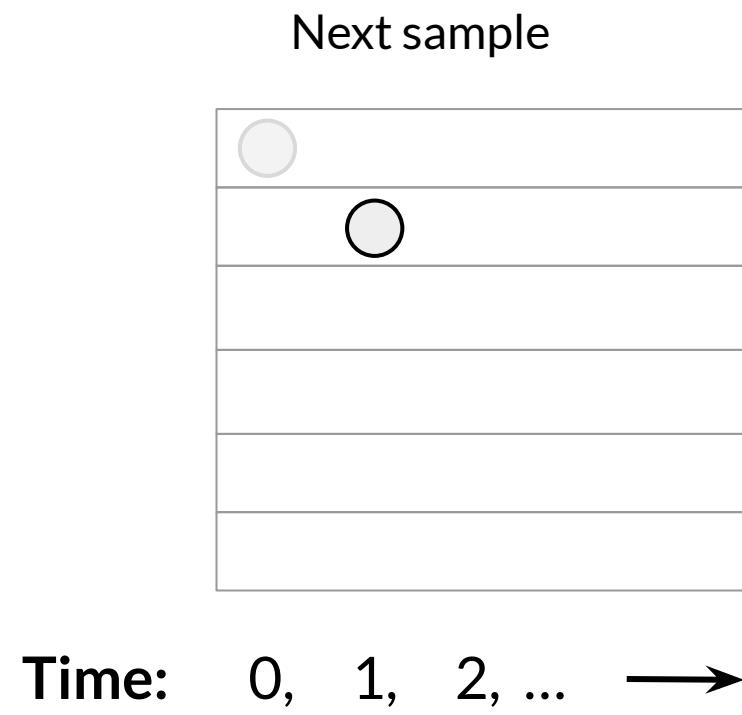
Initial sample



Source: Wikipedia, "Metropolis-Hastings Algorithm"

# Metropolis-Hastings (MH)

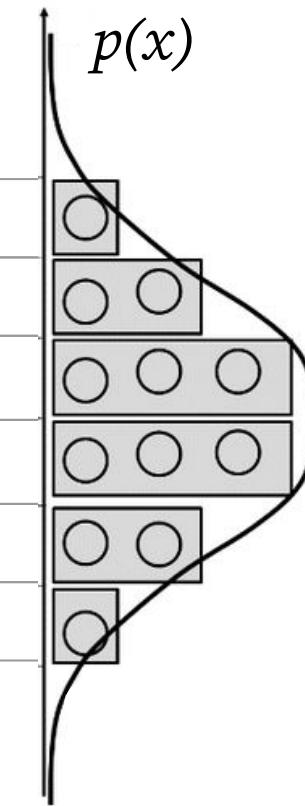
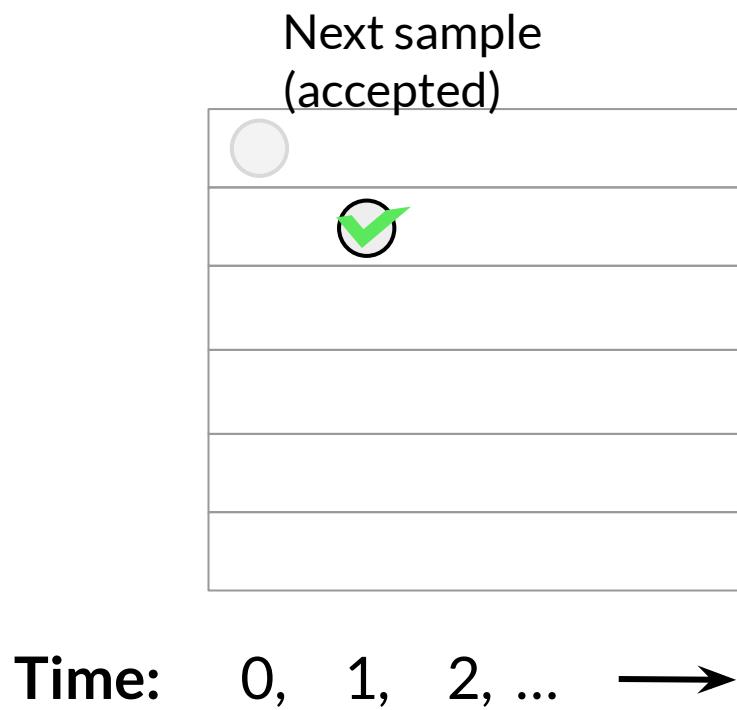
Intuition:



Source: Wikipedia, "Metropolis-Hastings Algorithm"

# Metropolis-Hastings (MH)

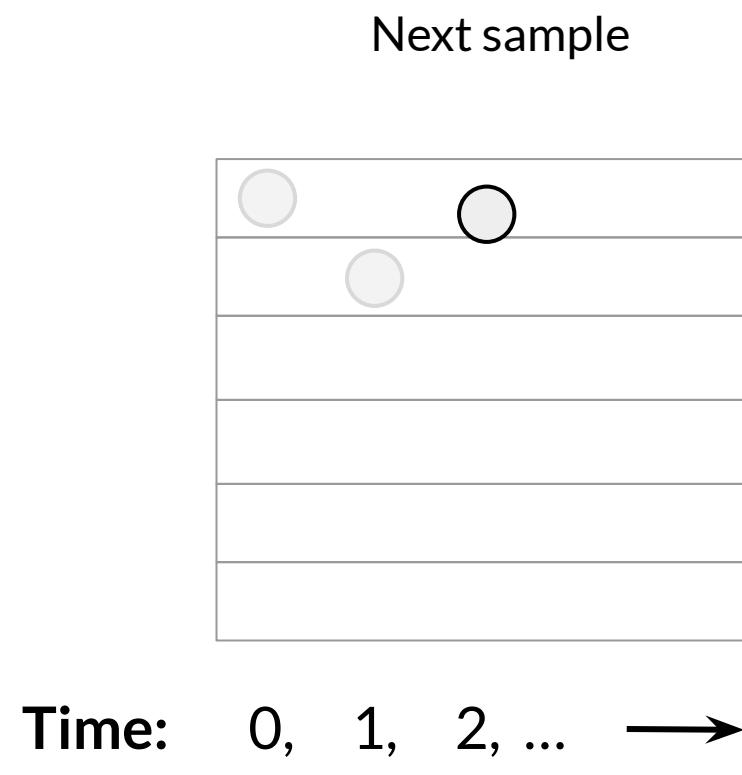
Intuition:



Source: Wikipedia, "Metropolis-Hastings Algorithm"

# Metropolis-Hastings (MH)

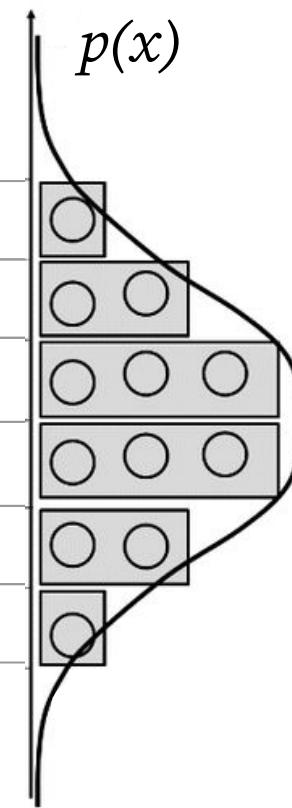
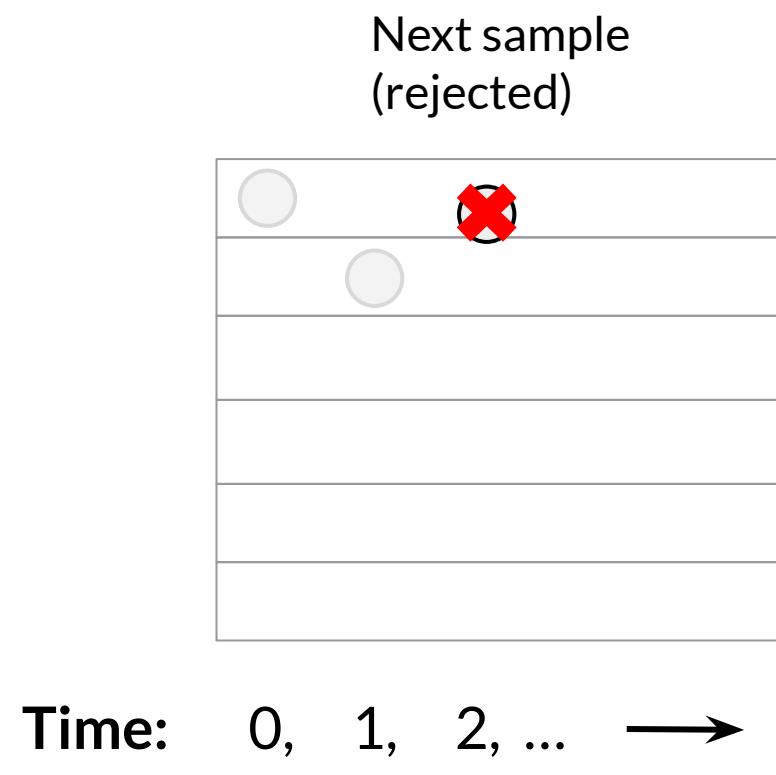
Intuition:



Source: Wikipedia, "Metropolis-Hastings Algorithm"

# Metropolis-Hastings (MH)

Intuition:

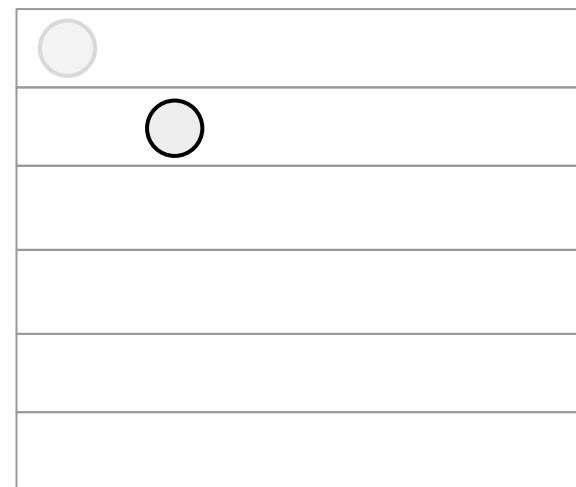


Source: Wikipedia, "Metropolis-Hastings Algorithm"

# Metropolis-Hastings (MH)

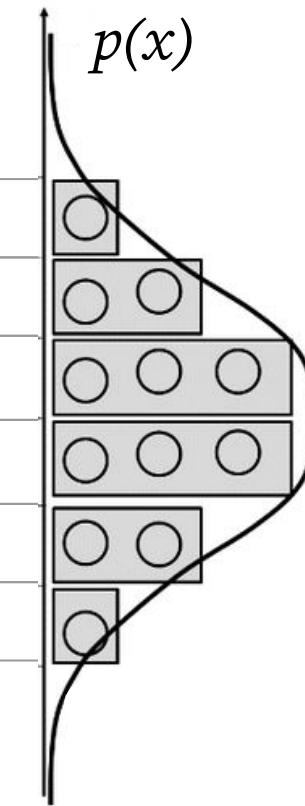
**Intuition:**

Go back to the  
previous sample



→

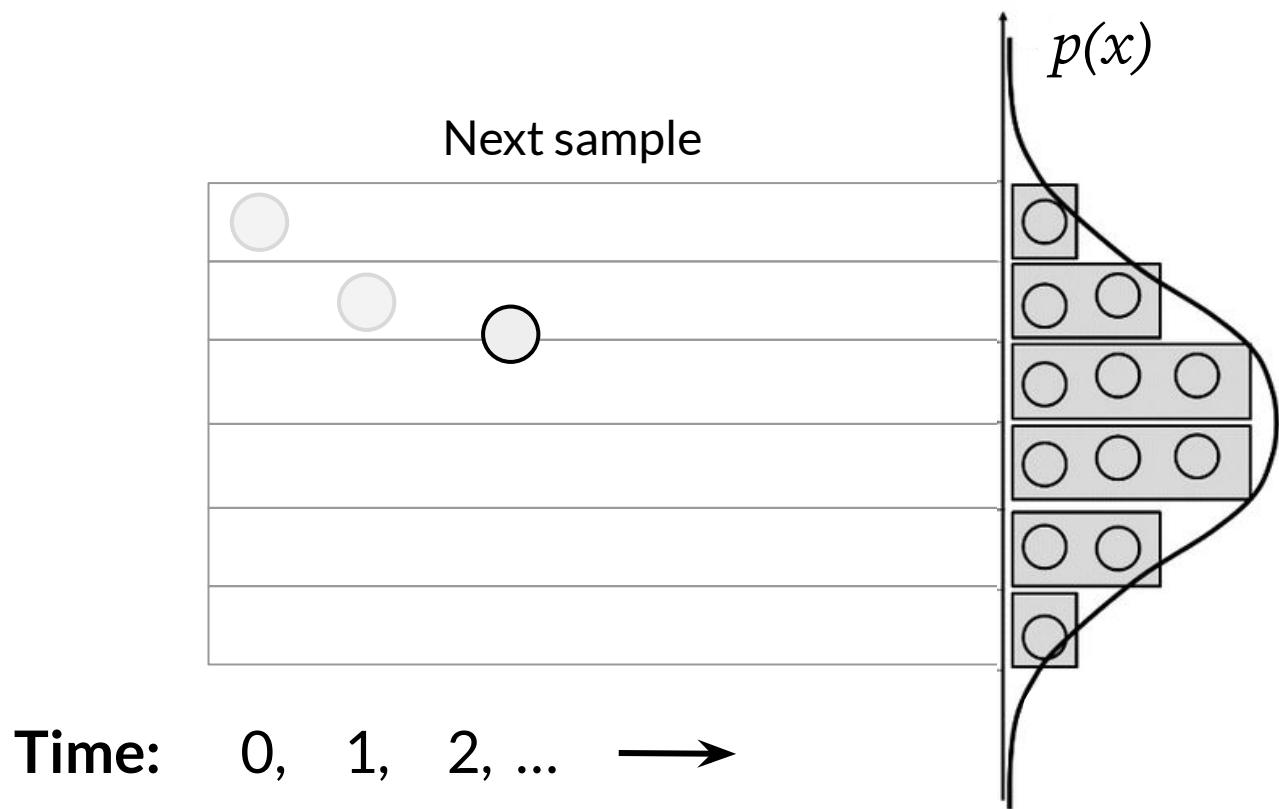
**Time:** 0, 1, 2, ... →



Source: Wikipedia, "Metropolis-Hastings Algorithm"

# Metropolis-Hastings (MH)

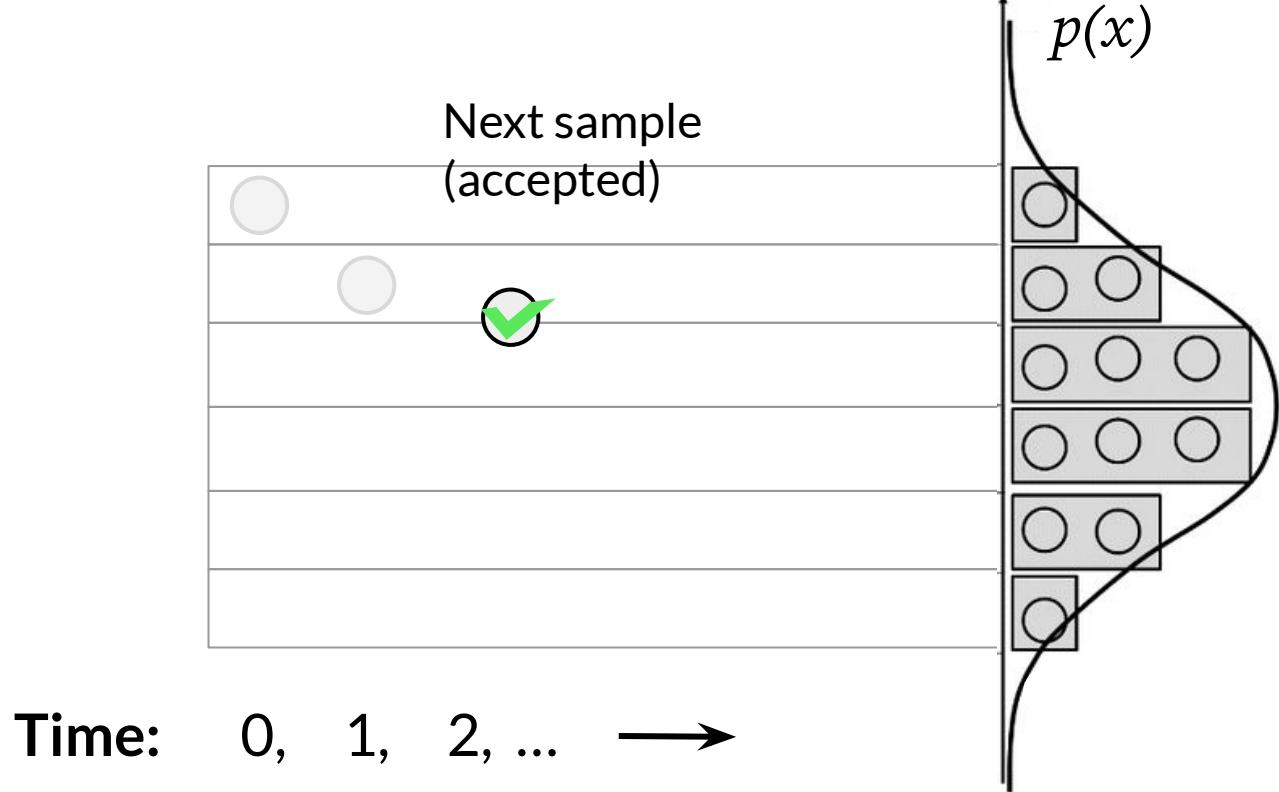
Intuition:



Source: Wikipedia, "Metropolis-Hastings Algorithm"

# Metropolis-Hastings (MH)

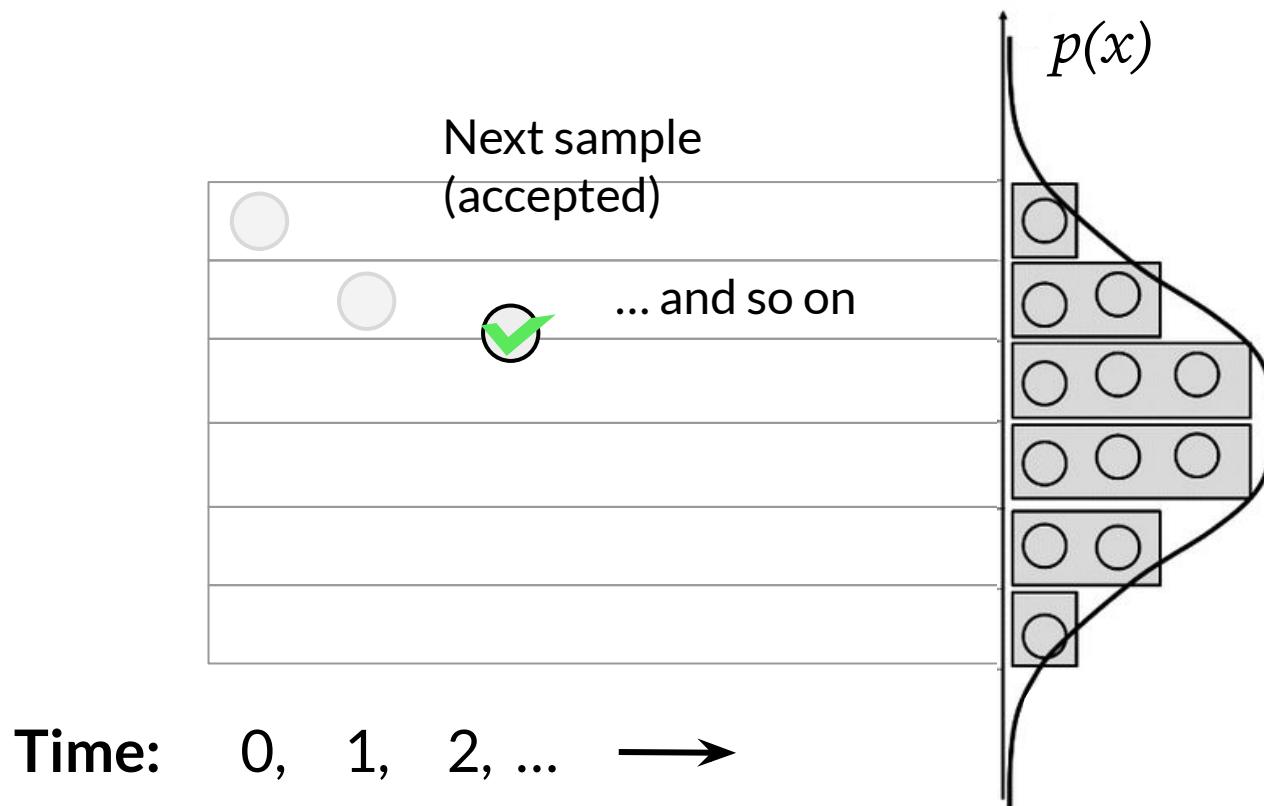
Intuition:



Source: Wikipedia, "Metropolis-Hastings Algorithm"

# Metropolis-Hastings (MH)

Intuition:



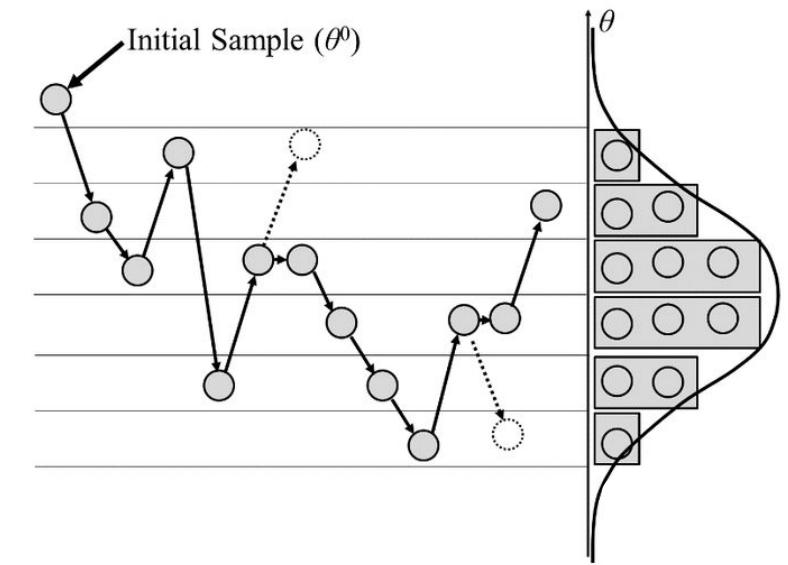
Source: Wikipedia, "Metropolis-Hastings Algorithm"

# Metropolis-Hastings (MH)

## Intuition:

The acceptance probability encourages us to move towards more-likely (higher-mass) regions of the distribution  $p(x)$ .

When  $Q$  suggests we move into a low-probability region, we follow that move only a fraction of the time.



Source: Wikipedia, "Metropolis-Hastings Algorithm"

# Metropolis-Hastings (MH)

## Intuition:

In practice, the distribution  $Q(x' | x)$  is often taken to be something simple, like a Gaussian centered at  $x$  (for continuous variables).

# Metropolis-Hastings (MH)

## Intuition:

In practice, the distribution  $Q(x' | x)$  is often taken to be something simple, like a Gaussian centered at  $x$  (for continuous variables).

Note that when  $Q(x' | x) = Q(x | x')$ , a symmetric distribution (e.g., Gaussians), then the acceptance ratio simplifies to:

$$A(x' | x) = \min \left( 1, \frac{p(x')}{p(x)} \right).$$

# Metropolis-Hastings (MH)

## Intuition:

In practice, the distribution  $Q(x' | x)$  is often taken to be something simple, like a Gaussian centered at  $x$  (for continuous variables).

Note that when  $Q(x' | x) = Q(x | x')$ , a symmetric distribution (e.g., Gaussians), then the acceptance ratio simplifies to:

$$A(x' | x) = \min \left( 1, \frac{p(x')}{p(x)} \right).$$

⇒ Metropolis Algorithm  
(symmetric proposals)

# **Metropolis-Hastings (MH)**

**Backstory:**

# Metropolis-Hastings (MH)

## Backstory:

This variant of the algorithm (*Metropolis Algorithm*) originally published in 1953!

THE JOURNAL OF CHEMICAL PHYSICS                    VOLUME 21, NUMBER 6                    JUNE, 1953

### Equation of State Calculations by Fast Computing Machines

NICHOLAS METROPOLIS, ARIANNA W. ROSENBLUTH, MARSHALL N. ROSENBLUTH, AND AUGUSTA H. TELLER,  
*Los Alamos Scientific Laboratory, Los Alamos, New Mexico*

AND

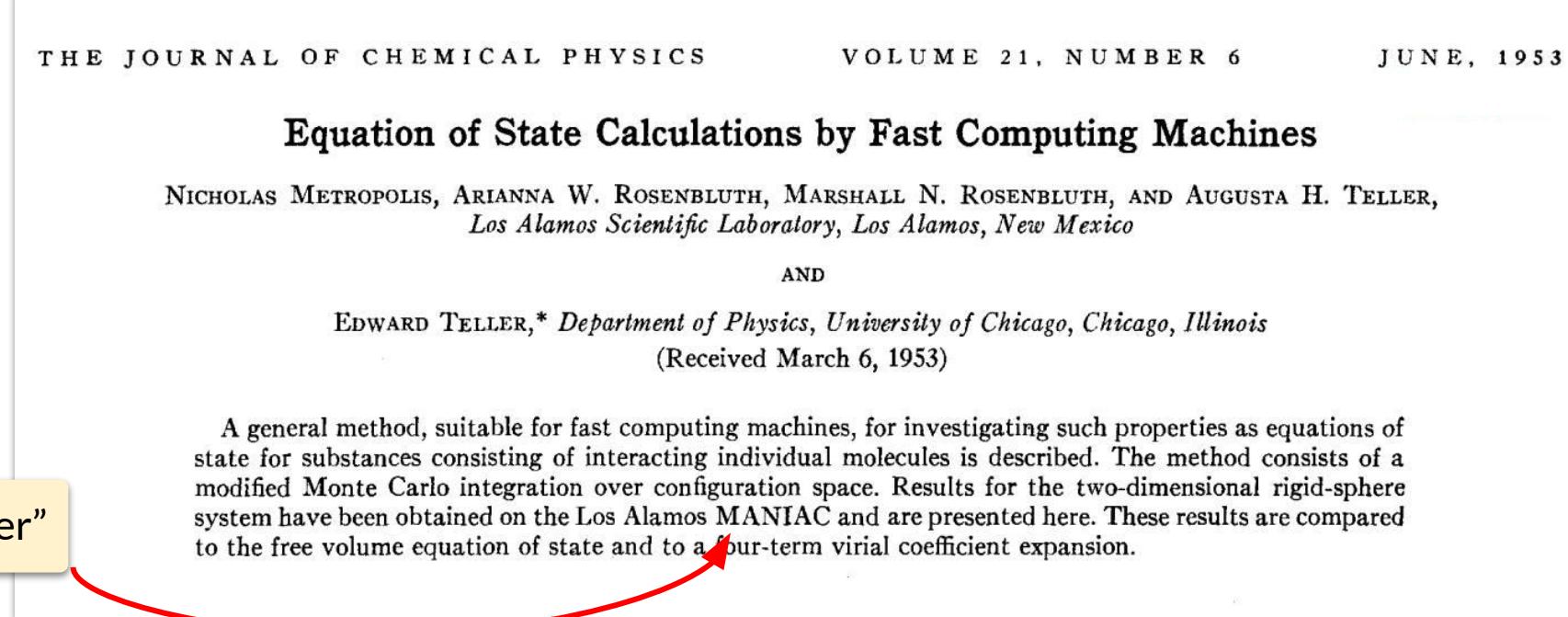
EDWARD TELLER,\* *Department of Physics, University of Chicago, Chicago, Illinois*  
(Received March 6, 1953)

A general method, suitable for fast computing machines, for investigating such properties as equations of state for substances consisting of interacting individual molecules is described. The method consists of a modified Monte Carlo integration over configuration space. Results for the two-dimensional rigid-sphere system have been obtained on the Los Alamos MANIAC and are presented here. These results are compared to the free volume equation of state and to a four-term virial coefficient expansion.

# Metropolis-Hastings (MH)

## Backstory:

This variant of the algorithm (*Metropolis Algorithm*) originally published in 1953!



# Metropolis-Hastings (MH)

## Backstory:

This variant of the algorithm (*Metropolis Algorithm*) originally published in 1953!

THE JOURNAL OF CHEMICAL PHYSICS      VOLUME 21, NUMBER 6      JUNE, 1953

### Equation of State Calculations by Fast Computing Machines

NICHOLAS METROPOLIS, ARIANNA W. ROSENBLUTH, MARSHALL N. ROSENBLUTH, AND AUGUSTA H. TELLER,  
*Los Alamos Scientific Laboratory, Los Alamos, New Mexico*

AND

EDWARD TELLER,\* *Department of Physics, University of Chicago, Chicago, Illinois*  
(Received March 6, 1953)

“Fast computer”

A general method, suitable for fast computing machines, for investigating such properties as equations of state for substances consisting of interacting individual molecules is described. The method consists of a modified Monte Carlo integration over configuration space. Results for the two-dimensional rigid-sphere system have been obtained on the Los Alamos MANIAC and are presented here. These results are compared to the free volume equation of state and to a four-term virial coefficient expansion.

Actually, the first computer to beat a human at a variant of chess!

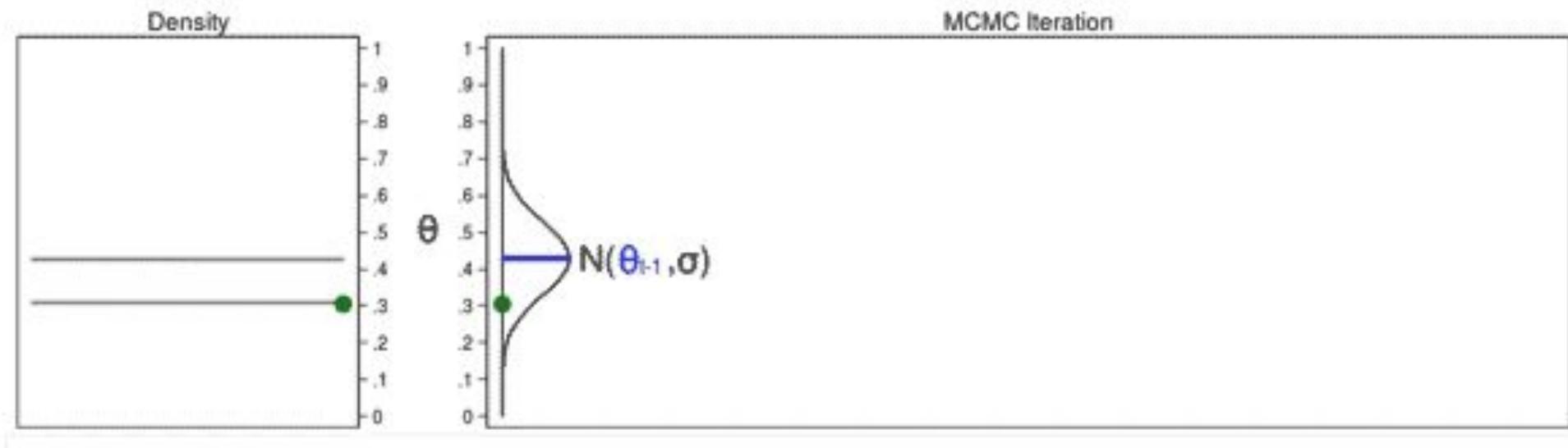
# Metropolis-Hastings (MH)

Animation of Metropolis Hastings Algorithm:

# Metropolis-Hastings (MH)

Animation of Metropolis Hastings Algorithm:

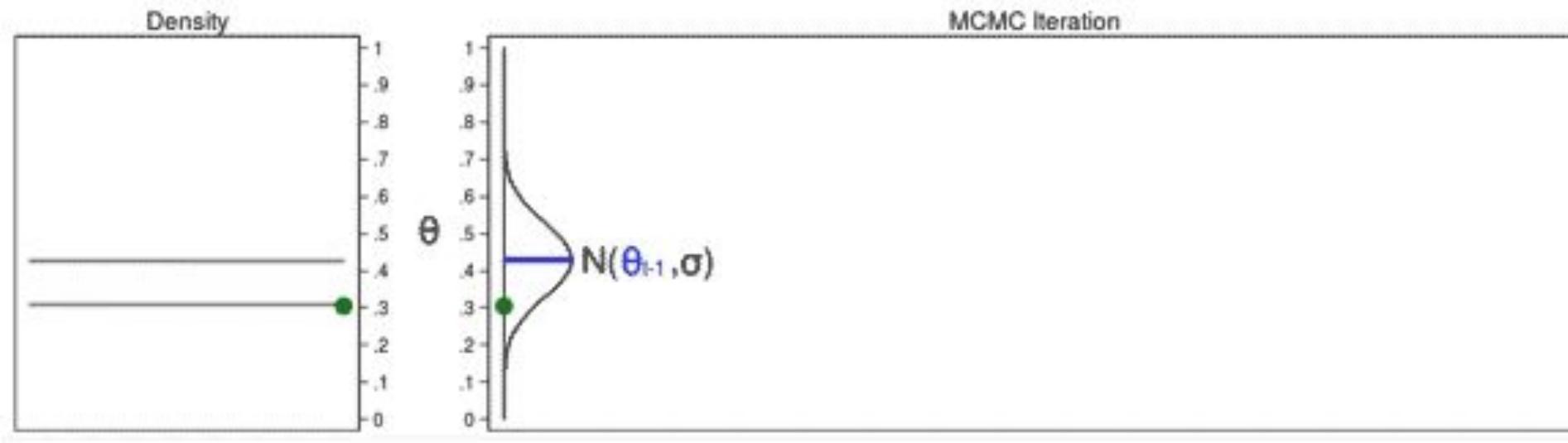
Uses a Gaussian proposal distribution to sample from a more complex distribution!  
(In this case, a Beta(5, 7) distribution).



# Metropolis-Hastings (MH)

Animation of Metropolis Hastings Algorithm:

Uses a Gaussian proposal distribution to sample from a more complex distribution!  
(In this case, a Beta(5, 7) distribution).

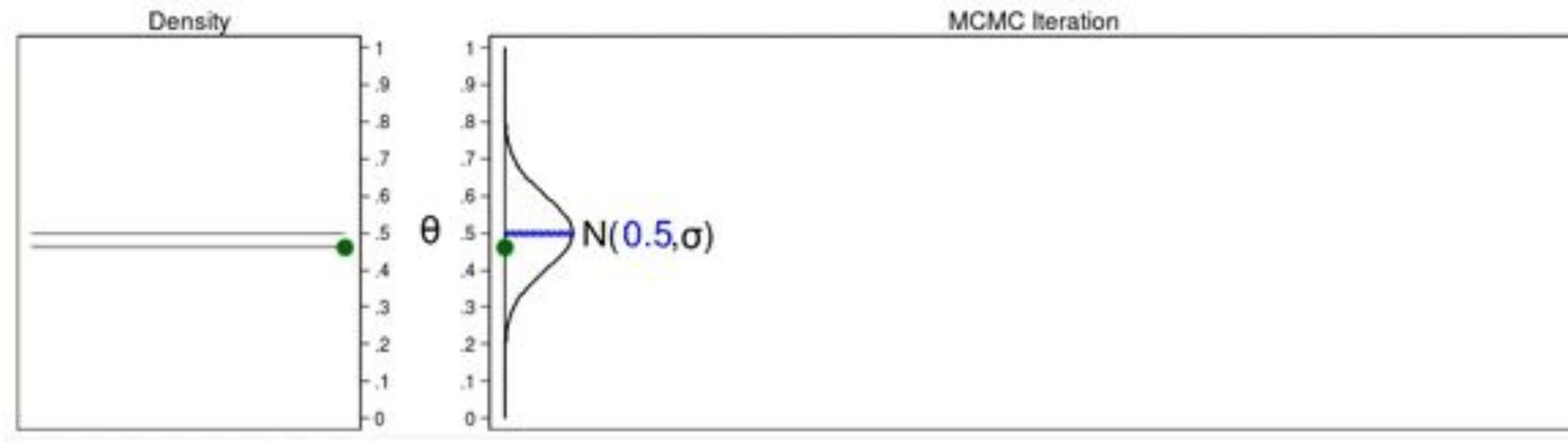


Notice that subsequent samples are (slightly) correlated.

# Metropolis-Hastings (MH)

Animation of Metropolis Hastings Algorithm:  
regular sampling from a simple distribution.

Compare that with exact sampling.  
E.g., from a Gaussian distribution.



Here, all samples are  
independent.

## Quick Note on “Posterior Sampling”

MCMC is often synonymous with “posterior sampling” (drawing samples from a posterior distribution).

However: methods so far have not really been posterior-distribution specific!

# Quick Note on “Posterior Sampling”

MCMC is often synonymous with “posterior sampling” (drawing samples from a posterior distribution).

However: methods so far have not really been posterior-distribution specific!

- ⇒ They just require: access to (unnormalized) PDF from which we want to sample.
- ⇒ Which is well suited to posterior sampling (but also to other inference as well).

# Gibbs Sampling

# Gibbs Sampling

A widely-used special case of MH is known as **Gibbs Sampling**.

Think of it as an “*axially aligned, carefully designed, transition function that always yields an acceptance ratio of 1*”.

# Gibbs Sampling

A widely-used special case of MH is known as **Gibbs Sampling**.

Think of it as an “*axially aligned, carefully designed, transition function that always yields an acceptance ratio of 1*”.

Why is it useful if the acceptance ratio is high?

- Less wasted samples!
- In high dimensional space, a bad (or even just simple) transition distribution can lead to very low acceptance ratio...  
    ⇒ bad numerical properties!

# Gibbs Sampling – Algorithm

# Gibbs Sampling – Algorithm

Suppose we have an ordered set of variables in our model:  $x_1, \dots, x_n$

And an initial configuration:  $x^0 = (x_1^0, \dots, x_n^0)$

# Gibbs Sampling – Algorithm

Suppose we have an ordered set of variables in our model:  $x_1, \dots, x_n$

And an initial configuration:  $x^0 = (x_1^0, \dots, x_n^0)$

Then carry out the following procedure (to generate samples from  $p(x)$ ):

Repeat until convergence for  $t = 1, 2, 3, \dots$

- Set  $x \leftarrow x^{t-1}$
- For each variable  $x_i, i = 1, \dots, n$ 
  - Sample  $x' \sim p(x_i | x_{-i})$
  - Update  $x \leftarrow (x_1, \dots, x'_i, \dots, x_n)$
- Set  $x^t \leftarrow x$

# Gibbs Sampling – Algorithm

Suppose we have an ordered set of variables in our model:  $x_1, \dots, x_n$

And an initial configuration:  $x^0 = (x_1^0, \dots, x_n^0)$

Then carry out the following procedure (to generate samples from  $p(x)$ ):

Repeat until convergence for  $t = 1, 2, 3, \dots$

- Set  $x \leftarrow x^{t-1}$
- For each variable  $x_i, i = 1, \dots, n$ 
  - Sample  $x' \sim p(x_i | x_{-i})$
  - Update  $x \leftarrow (x_1, \dots, x'_i, \dots, x_n)$
- Set  $x^t \leftarrow x$

For each sample...

# Gibbs Sampling – Algorithm

Suppose we have an ordered set of variables in our model:  $x_1, \dots, x_n$

And an initial configuration:  $x^0 = (x_1^0, \dots, x_n^0)$

Then carry out the following procedure (to generate samples from  $p(x)$ ):

Repeat until convergence for  $t = 1, 2, 3, \dots$

- Set  $x \leftarrow x^{t-1}$
- For each variable  $x_i, i = 1, \dots, n$ 
  - Sample  $x' \sim p(x_i | x_{-i})$
  - Update  $x \leftarrow (x_1, \dots, x'_i, \dots, x_n)$
- Set  $x^t \leftarrow x$

Initialize  $x$  to previous sample

# Gibbs Sampling – Algorithm

Suppose we have an ordered set of variables in our model:  $x_1, \dots, x_n$

And an initial configuration:  $x^0 = (x_1^0, \dots, x_n^0)$

Then carry out the following procedure (to generate samples from  $p(x)$ ):

Repeat until convergence for  $t = 1, 2, 3, \dots$

- Set  $x \leftarrow x^{t-1}$
- For each variable  $x_i$ ,  $i = 1, \dots, n$ 
  - Sample  $x' \sim p(x_i | x_{-i})$
  - Update  $x \leftarrow (x_1, \dots, x'_i, \dots, x_n)$
- Set  $x^t \leftarrow x$

For each variable (*dimension*)

# Gibbs Sampling – Algorithm

Suppose we have an ordered set of variables in our model:  $x_1, \dots, x_n$

And an initial configuration:  $x^0 = (x_1^0, \dots, x_n^0)$

Then carry out the following procedure (to generate samples from  $p(x)$ ):

Repeat until convergence for  $t = 1, 2, 3, \dots$

- Set  $x \leftarrow x^{t-1}$
- For each variable  $x_i, i = 1, \dots, n$ 
  - Sample  $x' \sim p(x_i | x_{-i})$
  - Update  $x \leftarrow (x_1, \dots, x'_i, \dots, x_n)$
- Set  $x^t \leftarrow x$

Sample from conditional of  $p(x)$   
( $x_{-i}$  denotes “all other variables”)

# Gibbs Sampling – Algorithm

Suppose we have an ordered set of variables in our model:  $x_1, \dots, x_n$

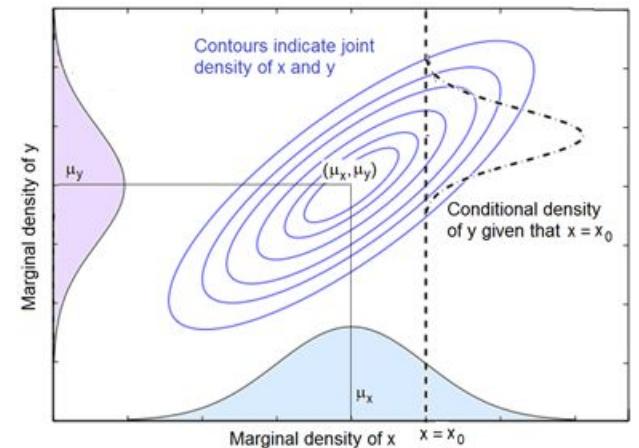
And an initial configuration:  $x^0 = (x_1^0, \dots, x_n^0)$

Then carry out the following procedure (to generate samples from  $p(x)$ ):

Repeat until convergence for  $t = 1, 2, 3, \dots$

- Set  $x \leftarrow x^{t-1}$
- For each variable  $x_i, i = 1, \dots, n$ 
  - Sample  $x' \sim p(x_i | x_{-i})$
  - Update  $x \leftarrow (x_1, \dots, x'_i, \dots, x_n)$
- Set  $x^t \leftarrow x$

Sample from conditional of  $p(x)$   
( $x_{-i}$  denotes “all other variables”)



Source: Pietro Vischia -  
Statistics for HEP

# Gibbs Sampling – Algorithm

Suppose we have an ordered set of variables in our model:  $x_1, \dots, x_n$

And an initial configuration:  $x^0 = (x_1^0, \dots, x_n^0)$

Then carry out the following procedure (to generate samples from  $p(x)$ ):

Repeat until convergence for  $t = 1, 2, 3, \dots$

- Set  $x \leftarrow x^{t-1}$
- For each variable  $x_i, i = 1, \dots, n$ 
  - Sample  $x' \sim p(x_i | x_{-i})$
  - Update  $x \leftarrow (x_1, \dots, x'_i, \dots, x_n)$
- Set  $x^t \leftarrow x$

Replace variable (*dimension*) with sample

# Gibbs Sampling – Algorithm

Suppose we have an ordered set of variables in our model:  $x_1, \dots, x_n$

And an initial configuration:  $x^0 = (x_1^0, \dots, x_n^0)$

Then carry out the following procedure (to generate samples from  $p(x)$ ):

Repeat until convergence for  $t = 1, 2, 3, \dots$

- Set  $x \leftarrow x^{t-1}$
- For each variable  $x_i, i = 1, \dots, n$ 
  - Sample  $x' \sim p(x_i | x_{-i})$
  - Update  $x \leftarrow (x_1, \dots, x'_i, \dots, x_n)$
- Set  $x^t \leftarrow x$

Set next sample to  $x$

# Gibbs Sampling – Algorithm

Suppose we have an ordered set of variables in our model:  $x_1, \dots, x_n$

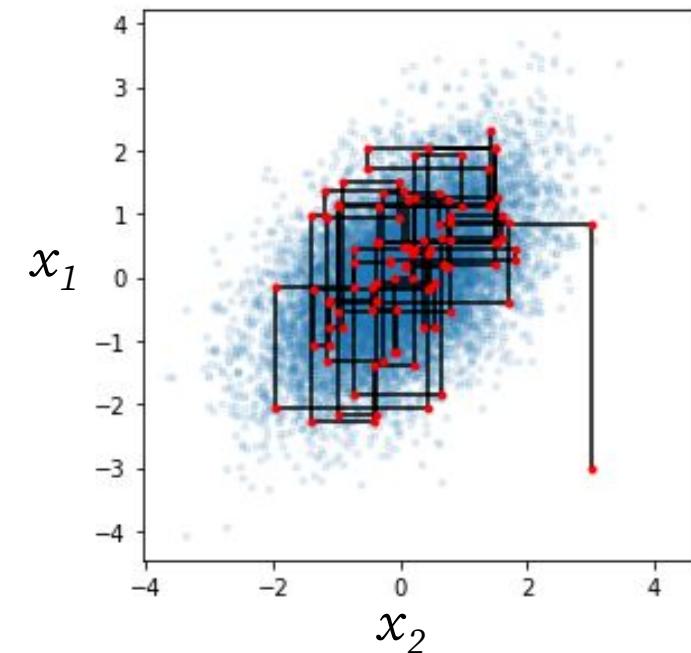
And an initial configuration:  $x^0 = (x_1^0, \dots, x_n^0)$

Then carry out the following procedure (to generate samples from  $p(x)$ ):

Looks like this:

Repeat until convergence for  $t = 1, 2, 3, \dots$

- Set  $x \leftarrow x^{t-1}$
- For each variable  $x_i, i = 1, \dots, n$ 
  - Sample  $x' \sim p(x_i | x_{-i})$
  - Update  $x \leftarrow (x_1, \dots, x'_i, \dots, x_n)$
- Set  $x^t \leftarrow x$



# Gibbs Sampling

But if this is a special case of MH, where is the acceptance ratio?

# Gibbs Sampling

But if this is a special case of MH, where is the acceptance ratio?

⇒ you can view this as a special case of MH with proposal:

$$Q(x' \mid x) = Q(x'_i, x_{-i} \mid x_i, x_{-i}) = p(x'_i \mid x_{-i})$$

# Gibbs Sampling

But if this is a special case of MH, where is the acceptance ratio?

⇒ you can view this as a special case of MH with proposal:

$$Q(x' \mid x) = Q(x'_i, x_{-i} \mid x_i, x_{-i}) = p(x'_i \mid x_{-i})$$

If you plug this into the acceptance ratio formula, you can see that the numerator and denominator cancel out!

$$A(x' \mid x) = \min \left( 1, \frac{p(x')Q(x \mid x')}{p(x)Q(x' \mid x)} \right)$$

# Gibbs Sampling

But if this is a special case of MH, where is the acceptance ratio?

⇒ you can view this as a special case of MH with proposal:

$$Q(x' \mid x) = Q(x'_i, x_{-i} \mid x_i, x_{-i}) = p(x'_i \mid x_{-i})$$

If you plug this into the acceptance ratio formula, you can see that the numerator and denominator cancel out!

$$A(x' \mid x) = \min \left( 1, \frac{p(x')Q(x \mid x')}{p(x)Q(x' \mid x)} \right)$$

$$\Rightarrow A(x' \mid x) = \min \left( 1, \frac{p(x'_i \mid x'_{-i})p(x'_{-i})}{p(x_i \mid x_{-i})p(x_{-i})} \frac{p(x_i \mid x'_{-i})}{p(x'_i \mid x_{-i})} \right)$$

# Gibbs Sampling

But if this is a special case of MH, where is the acceptance ratio?

⇒ you can view this as a special case of MH with proposal:

$$Q(x' \mid x) = Q(x'_i, x_{-i} \mid x_i, x_{-i}) = p(x'_i \mid x_{-i})$$

If you plug this into the acceptance ratio formula, you can see that the numerator and denominator cancel out!

$$A(x' \mid x) = \min \left( 1, \frac{p(x')Q(x \mid x')}{p(x)Q(x' \mid x)} \right)$$

$$\Rightarrow A(x' \mid x) = \min \left( 1, \frac{\cancel{p(x'_i \mid x'_{-i})} p(x'_{-i}) \cancel{p(x_i \mid x'_{-i})}}{\cancel{p(x_i \mid x_{-i})} p(x_{-i}) \cancel{p(x'_i \mid x_{-i})}} \right)$$

# Gibbs Sampling

But if this is a special case of MH, where is the acceptance ratio?

⇒ you can view this as a special case of MH with proposal:

$$Q(x' \mid x) = Q(x'_i, x_{-i} \mid x_i, x_{-i}) = p(x'_i \mid x_{-i})$$

If you plug this into the acceptance ratio formula, you can see that the numerator and denominator cancel out!

$$A(x' \mid x) = \min \left( 1, \frac{p(x')Q(x \mid x')}{p(x)Q(x' \mid x)} \right)$$

$$\Rightarrow A(x' \mid x) = \min \left( 1, \frac{\cancel{p(x'_i \mid x'_{-i})} p(x'_{-i}) \cancel{p(x_i \mid x'_{-i})}}{\cancel{p(x_i \mid x_{-i})} p(x_{-i}) \cancel{p(x'_i \mid x_{-i})}} \right)$$

# Gibbs Sampling

But if this is a special case of MH, where is the acceptance ratio?

⇒ you can view this as a special case of MH with proposal:

$$Q(x' \mid x) = Q(x'_i, x_{-i} \mid x_i, x_{-i}) = p(x'_i \mid x_{-i})$$

If you plug this into the acceptance ratio formula, you can see that the numerator and denominator cancel out!

$$A(x' \mid x) = \min \left( 1, \frac{p(x')Q(x \mid x')}{p(x)Q(x' \mid x)} \right)$$

$$\Rightarrow A(x' \mid x) = \min \left( 1, \frac{\cancel{p(x'_i \mid x'_{-i})} \cancel{p(x'_{-i})} \cancel{p(x_i \mid x'_{-i})}}{\cancel{p(x_i \mid x_{-i})} \cancel{p(x_{-i})} \cancel{p(x'_i \mid x_{-i})}} \right)$$

# Gibbs Sampling

One important question: how to sample from the conditional  $x' \sim p(x_i \mid x_{-i})$ ?

# Gibbs Sampling

One important question: how to sample from the conditional  $x' \sim p(x_i \mid x_{-i})$ ?

- ⇒ Gibbs sampling does not provide a general-purpose way to sample this!
- ⇒ “*It's up to you to figure out*” :D

# Gibbs Sampling

One important question: how to sample from the conditional  $x' \sim p(x_i | x_{-i})$ ?

⇒ Gibbs sampling does not provide a general-purpose way to sample this!

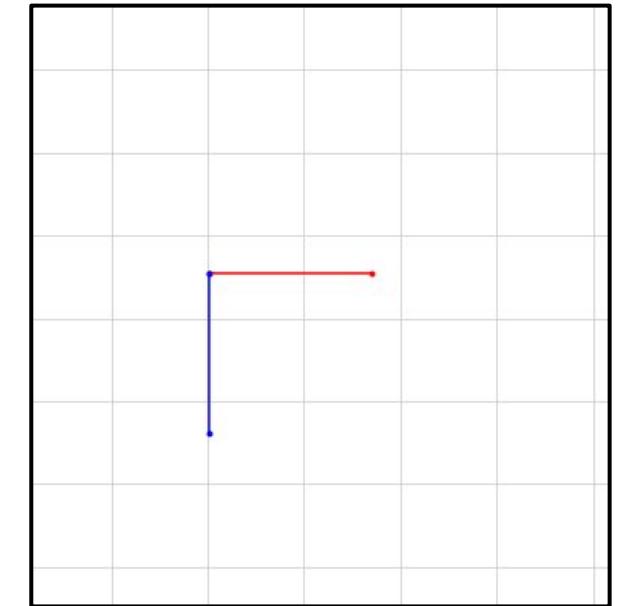
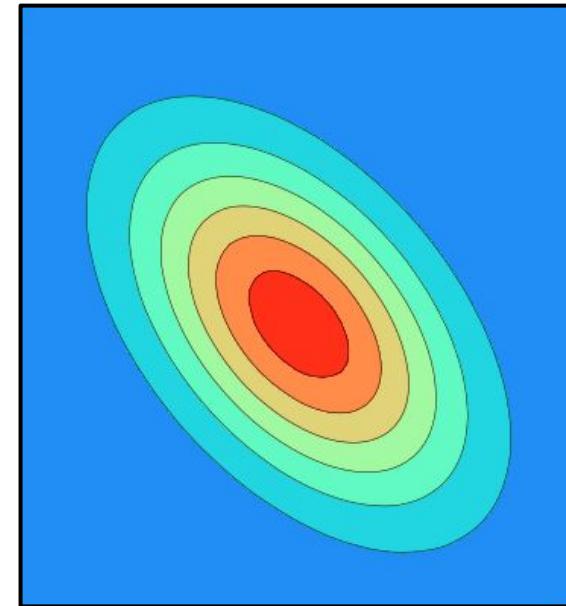
⇒ “It’s up to you to figure out” :D

In some cases:

- You can sample from this conditional in closed form (*in certain models, e.g., LDA*).
- You run a subroutine to sample from this conditional.
  - As an example: Metropolis-Hastings algorithm!
  - Sometimes this “fixes” low acceptance rate issues, since MH is now over only *one variable* (much lower dimensionality).

# Gibbs Sampling

**Animation:**  
Gibbs sampling in practice!



Source: Sandipan Dey, Bayesian Machine Learning

# **Gradient-based MCMC**

# Gradient Based MCMC

## Gradient Based MCMC

One final strategy to avoid low acceptance rate in Metropolis-Hastings (particularly in high dimensions) – using gradients ✨.

# Gradient Based MCMC

One final strategy to avoid low acceptance rate in Metropolis-Hastings (particularly in high dimensions) – using gradients ✨.

*Background* – the issue with simple proposals in MH  $\Rightarrow$  low acceptance rates.

# Gradient Based MCMC

One final strategy to avoid low acceptance rate in Metropolis-Hastings (particularly in high dimensions) – using gradients ✨.

*Background* – the issue with simple proposals in MH  $\Rightarrow$  low acceptance rates.

- You could achieve a higher acceptance rate by just taking small steps (e.g., use a symmetric distribution with small variance centered at previous sample).

# Gradient Based MCMC

One final strategy to avoid low acceptance rate in Metropolis-Hastings (particularly in high dimensions) — using gradients ✨.

*Background* – the issue with simple proposals in MH  $\Rightarrow$  low acceptance rates.

- You could achieve a higher acceptance rate by just taking small steps (e.g., use a symmetric distribution with small variance centered at previous sample).
- However, small steps  $\Rightarrow$  bad mixing and very slow progress of the algorithm (i.e., high correlation  $\Rightarrow$  need many more samples for same statistical properties).

## Gradient Based MCMC

A few popular MCMC methods leverage the *gradient of the PDF* in order to propose a sample in a high mass region from potentially far away in space.

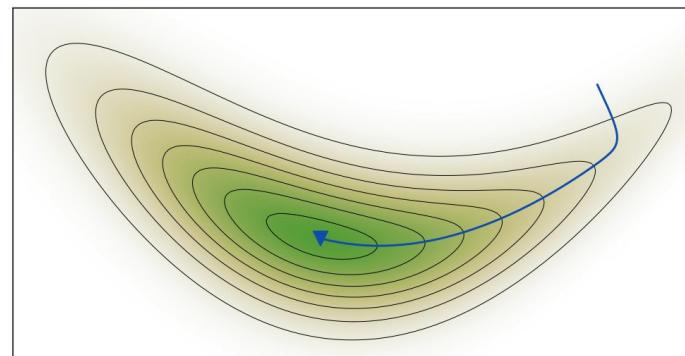
(These methods original came from physics, in particular the field of quantum chromodynamics).

# Gradient Based MCMC

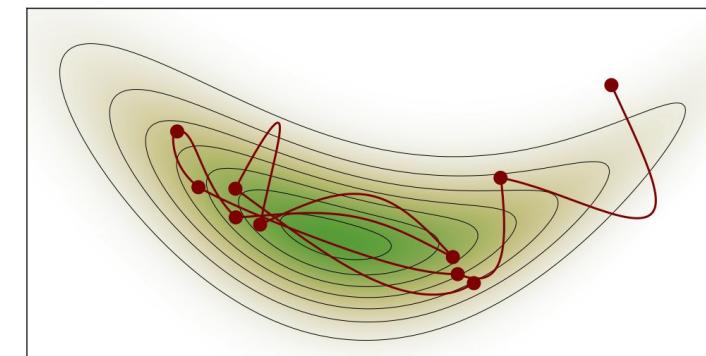
A few popular MCMC methods leverage the *gradient of the PDF* in order to propose a sample in a high mass region from potentially far away in space.

(These methods originally came from physics, in particular the field of quantum chromodynamics).

They are called: **Langevin Monte Carlo (LMC)** and **Hamiltonian Monte Carlo (HMC)**.



Gradient Descent Path



Hamiltonian Monte Carlo Samples

## Gradient Based MCMC

We could do a full lesson on these methods alone – so will give you the overview.  
(And I can provide more resources for those who are interested to learn more!)

# Gradient Based MCMC

Suppose we have a given PDF  $p(x)$ ,  $x \in \mathbb{R}^n$ .

Consider an objective function:  $f(x) = -\log p(x)$ .

# Gradient Based MCMC

Suppose we have a given PDF  $p(x)$ ,  $x \in \mathbb{R}^n$ .

Consider an objective function:  $f(x) = -\log p(x)$ .

The gradient of this objective function is:

$$\nabla_x f(x) = \left[ \frac{d}{dx_1} f(x), \dots, \frac{d}{dx_n} f(x) \right]^\top$$

## Gradient Based MCMC

If we wanted to perform *optimization* (e.g., to find a minimum of the objective), we could perform gradient descent:

$$x_t = x_{t-1} - \eta_t \nabla_x f(x_{t-1})$$

where  $\eta_t$  is the learning rate/step size at iteration  $t$ .

# Gradient Based MCMC

If we wanted to perform *optimization* (e.g., to find a minimum of the objective), we could perform gradient descent:

Next  $x$

$$x_t = x_{t-1} - \eta_t \nabla_x f(x_{t-1})$$

where  $\eta_t$  is the learning rate/step size at iteration  $t$ .

# Gradient Based MCMC

If we wanted to perform *optimization* (e.g., to find a minimum of the objective), we could perform gradient descent:

Prev.  $x$

$$x_t = x_{t-1} - \eta_t \nabla_x f(x_{t-1})$$

where  $\eta_t$  is the learning rate/step size at iteration  $t$ .

## Gradient Based MCMC

If we wanted to perform *optimization* (e.g., to find a minimum of the objective), we could perform gradient descent:

Step-size times gradient of objective (at prev.  $x$ )

$$x_t = x_{t-1} - \eta_t \nabla_x f(x_{t-1})$$

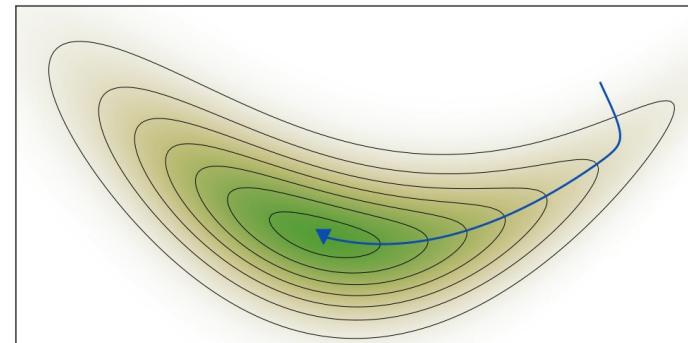
where  $\eta_t$  is the learning rate/step size at iteration  $t$ .

# Gradient Based MCMC

If we wanted to perform *optimization* (e.g., to find a minimum of the objective), we could perform gradient descent:

$$x_t = x_{t-1} - \eta_t \nabla_x f(x_{t-1})$$

where  $\eta_t$  is the learning rate/step size at iteration  $t$ .



Gradient Descent Path

Source: Wikipedia, "Hamiltonian Monte Carlo"

## **Gradient Based MCMC – Langevin Monte Carlo (LMC)**

Langevin Monte Carlo (LMC) is very similar!

## Gradient Based MCMC – Langevin Monte Carlo (LMC)

Langevin Monte Carlo (LMC) is very similar!

Each step in LMC can be written:

$$x_t = x_{t-1} - \frac{\epsilon_t}{2} \nabla_x f(x_{t-1}) + \sqrt{\epsilon_t} \mathcal{N}(0, I_n)$$

where  $\epsilon_t$  is the *time step/step size* at iteration  $t$ .

# Gradient Based MCMC – Langevin Monte Carlo (LMC)

Langevin Monte Carlo (LMC) is very similar!

Each step in LMC can be written:

Next  $x$

$$x_t = x_{t-1} - \frac{\epsilon_t}{2} \nabla_x f(x_{t-1}) + \sqrt{\epsilon_t} \mathcal{N}(0, I_n)$$

where  $\epsilon_t$  is the *time step/step size* at iteration  $t$ .

# Gradient Based MCMC – Langevin Monte Carlo (LMC)

Langevin Monte Carlo (LMC) is very similar!

Each step in LMC can be written:

$$x_t = x_{t-1} - \frac{\epsilon_t}{2} \nabla_x f(x_{t-1}) + \sqrt{\epsilon_t} \mathcal{N}(0, I_n)$$

where  $\epsilon_t$  is the *time step/step size* at iteration  $t$ .

# Gradient Based MCMC – Langevin Monte Carlo (LMC)

Langevin Monte Carlo (LMC) is very similar!

Each step in LMC can be written:

Time step (over 2) times gradient of objective (at prev.  $x$ )

$$x_t = x_{t-1} - \frac{\epsilon_t}{2} \nabla_x f(x_{t-1}) + \sqrt{\epsilon_t} \mathcal{N}(0, I_n)$$

where  $\epsilon_t$  is the *time step/step size* at iteration  $t$ .

# Gradient Based MCMC – Langevin Monte Carlo (LMC)

Langevin Monte Carlo (LMC) is very similar!

Each step in LMC can be written:

Time step (square root) times standard normal (multivariate).

$$x_t = x_{t-1} - \frac{\epsilon_t}{2} \nabla_x f(x_{t-1}) + \sqrt{\epsilon_t} \mathcal{N}(0, I_n)$$

where  $\epsilon_t$  is the *time step/step size* at iteration  $t$ .

# Gradient Based MCMC – Langevin Monte Carlo (LMC)

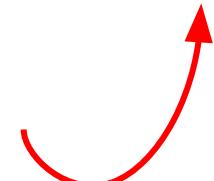
Langevin Monte Carlo (LMC) is very similar!

Each step in LMC can be written:

$$x_t = x_{t-1} - \frac{\epsilon_t}{2} \nabla_x f(x_{t-1}) + \sqrt{\epsilon_t} \mathcal{N}(0, I_n)$$

$$x_t = x_{t-1} - \eta_t \nabla_x f(x_{t-1})$$

Compare with gradient descent



# Gradient Based MCMC – Langevin Monte Carlo (LMC)

Langevin Monte Carlo (LMC) is very similar!

Each step in LMC can be written:

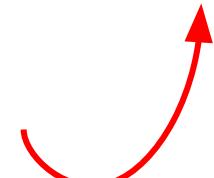
$$x_t = x_{t-1} - \frac{\epsilon_t}{2} \nabla_x f(x_{t-1}) + \sqrt{\epsilon_t} \mathcal{N}(0, I_n)$$

Main difference is an extra  
“stochastic term”

$$x_t = x_{t-1} - \eta_t \nabla_x f(x_{t-1})$$

Can view this as “gradient  
descent plus noise”

Compare with gradient descent



## Gradient Based MCMC – Langevin Monte Carlo (LMC)

Langevin Monte Carlo (LMC) is very similar!

Each step in LMC can be written:

$$x_t = x_{t-1} - \frac{\epsilon_t}{2} \nabla_x f(x_{t-1}) + \sqrt{\epsilon_t} \mathcal{N}(0, I_n)$$

Can also, equivalently, write this LMC update as a sample (from a Markov chain):

# Gradient Based MCMC – Langevin Monte Carlo (LMC)

Langevin Monte Carlo (LMC) is very similar!

Each step in LMC can be written:

$$x_t = x_{t-1} - \frac{\epsilon_t}{2} \nabla_x f(x_{t-1}) + \sqrt{\epsilon_t} \mathcal{N}(0, I_n)$$

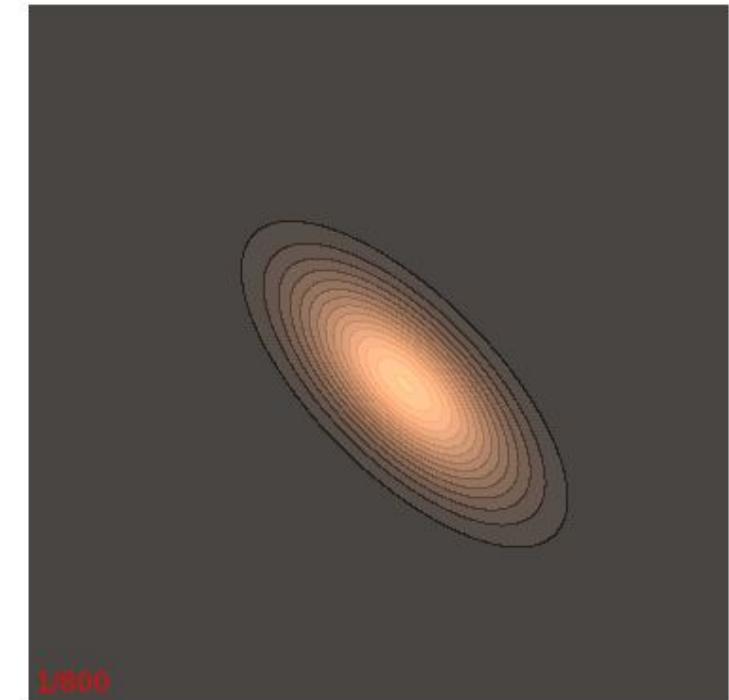
Can also, equivalently, write this LMC update as a sample (from a Markov chain):

$$x_t \sim p(x_t | x_{t-1}) = \mathcal{N} \left( x_t | \underbrace{x_{t-1} - \frac{\epsilon_t}{2} \nabla_x f(x_{t-1})}_{\text{Mean}}, \underbrace{\epsilon_t I_n}_{\text{Cov. Matrix}} \right)$$

# Gradient Based MCMC – Langevin Monte Carlo (LMC)

Intuitively:

You are moving toward higher-mass regions of the PDF (in expectation).



Source: Roy Friedman, "A Simplified Overview of Langevin Dynamics"

Source: Roy Friedman, "A Simplified Overview of Langevin Dynamics"

# Gradient Based MCMC – Langevin Monte Carlo (LMC)

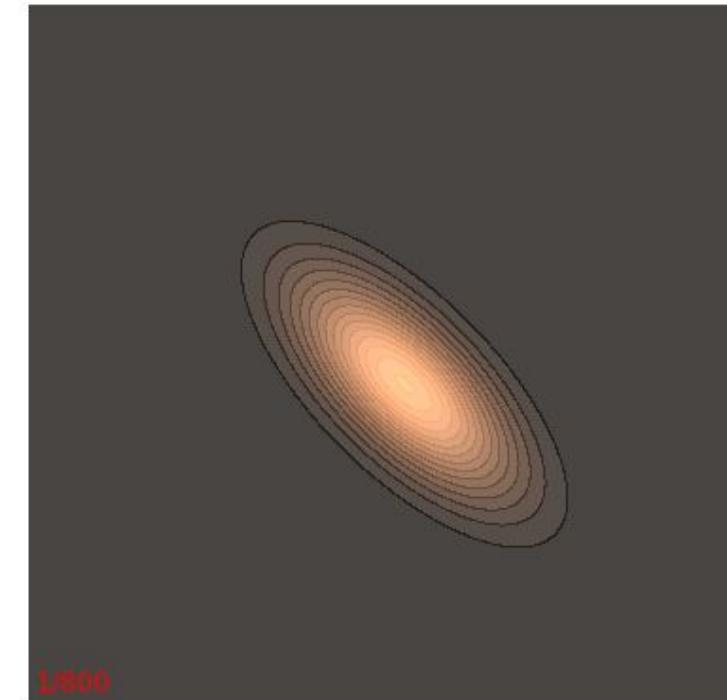
Intuitively:

You are moving toward higher-mass regions of the PDF (in expectation).

But also: adding noise such you have some probability of visiting lower-mass regions.

⇒ balanced so that the probability of visiting any region is proportional to the density in that region.

(Unlike gradient descent, where you only move towards higher-mass regions.)



Source: Roy Friedman, "A Simplified Overview of Langevin Dynamics"

Source: Roy Friedman, "A Simplified Overview of Langevin Dynamics"

# Gradient Based MCMC – Langevin Monte Carlo (LMC)

So does it work?

# Gradient Based MCMC – Langevin Monte Carlo (LMC)

So does it work?

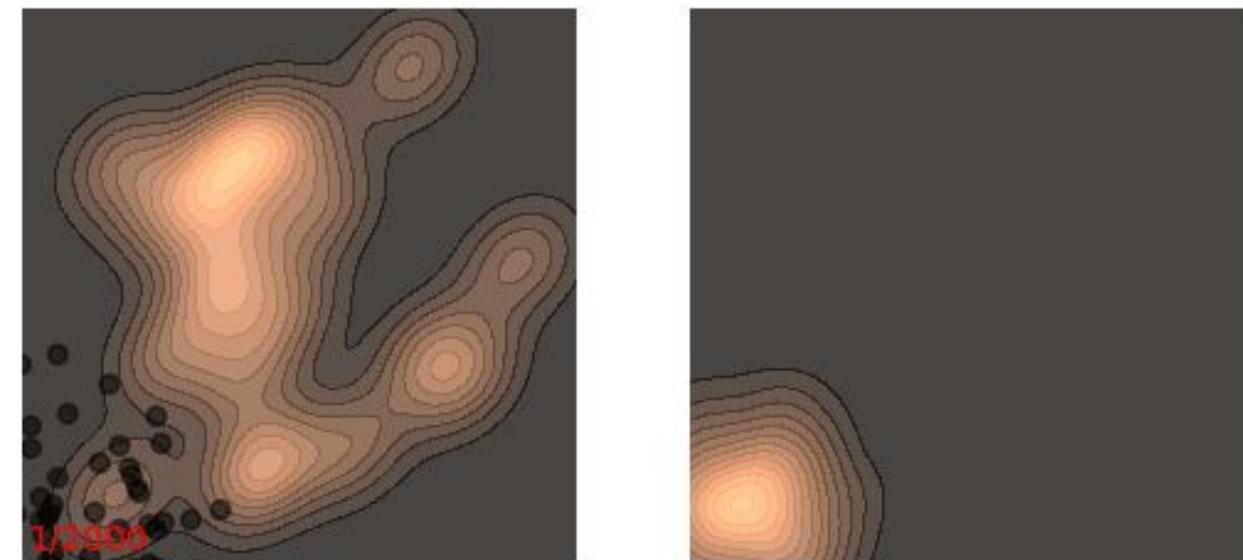
You can prove that the stationary distribution of this Markov chain is equal to  $p(x)$ , under the condition that: as  $T \rightarrow \infty$ , then  $\epsilon_t \rightarrow 0$ .

# Gradient Based MCMC – Langevin Monte Carlo (LMC)

So does it work?

You can prove that the stationary distribution of this Markov chain is equal to  $p(x)$ , under the condition that: as  $T \rightarrow \infty$ , then  $\epsilon_t \rightarrow 0$ .

For an empirical illustration:  
(using a small  $\epsilon_t$ ).



Source: Roy Friedman, "A Simplified Overview of Langevin Dynamics"

# Gradient Based MCMC – Langevin Monte Carlo (LMC)

Issues with this:

# Gradient Based MCMC – Langevin Monte Carlo (LMC)

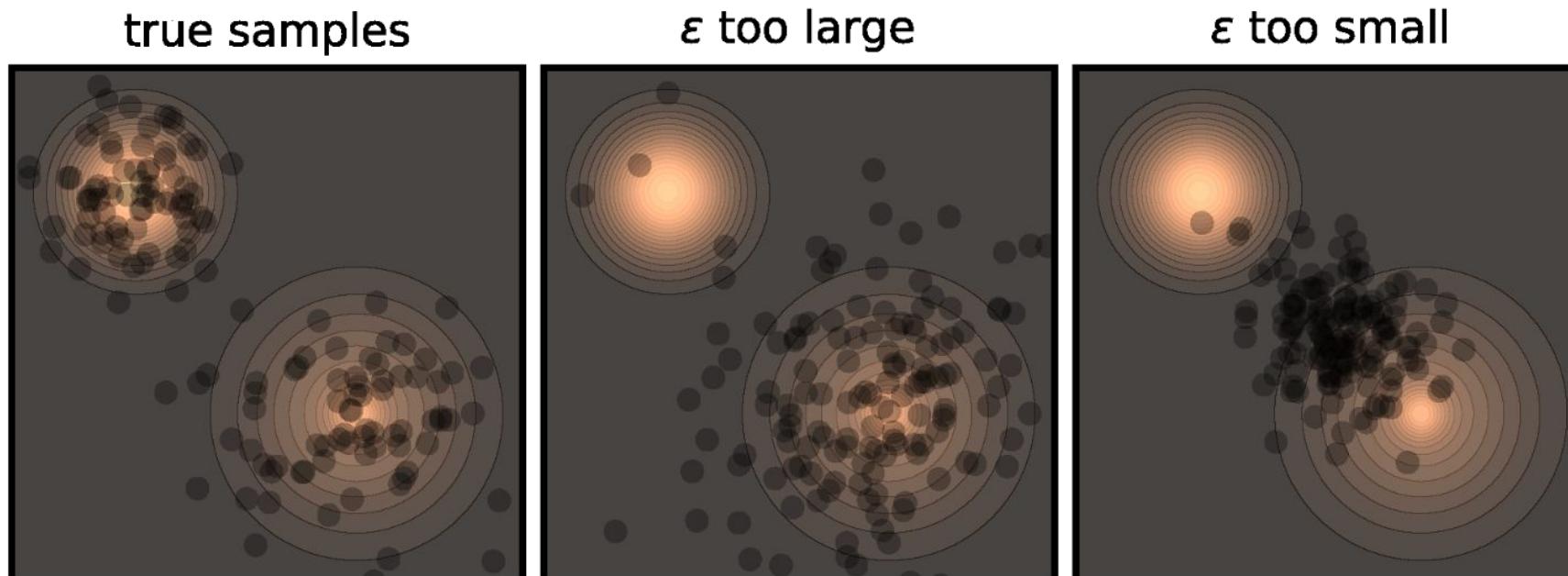
Issues with this:

- If step size is too large  $\Rightarrow$  samples will be biased (incorrect stationary distribution).
- If step size is too small  $\Rightarrow$  takes much longer (bad *mixing* properties).

# Gradient Based MCMC – Langevin Monte Carlo (LMC)

Issues with this:

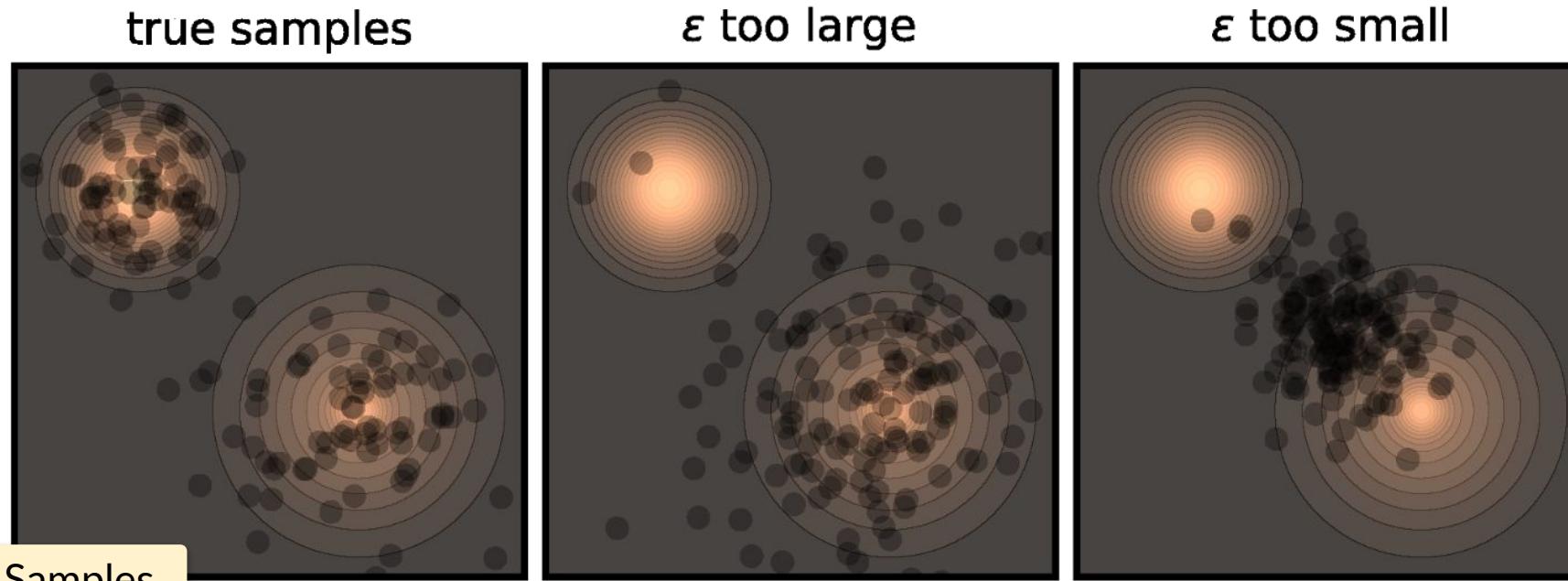
- If step size is too large  $\Rightarrow$  samples will be biased (incorrect stationary distribution).
- If step size is too small  $\Rightarrow$  takes much longer (bad *mixing* properties).



# Gradient Based MCMC – Langevin Monte Carlo (LMC)

Issues with this:

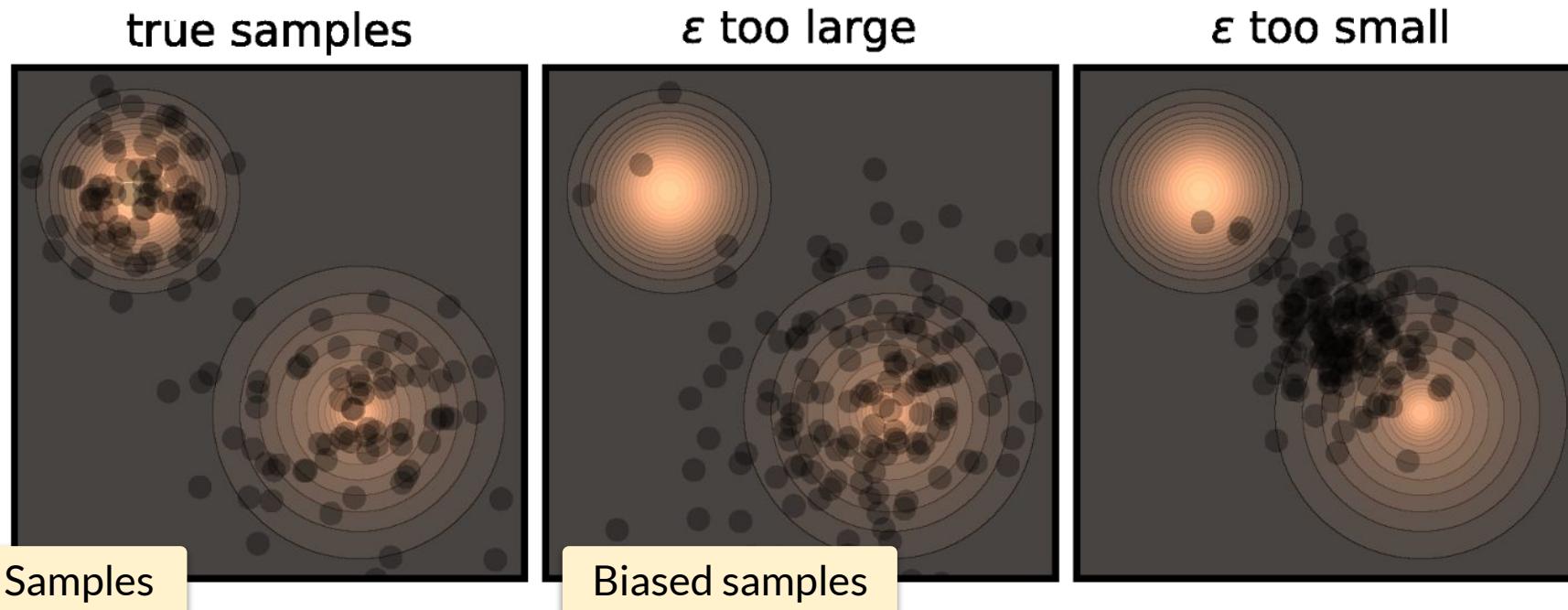
- If step size is too large  $\Rightarrow$  samples will be biased (incorrect stationary distribution).
- If step size is too small  $\Rightarrow$  takes much longer (bad *mixing* properties).



# Gradient Based MCMC – Langevin Monte Carlo (LMC)

Issues with this:

- If step size is too large  $\Rightarrow$  samples will be biased (incorrect stationary distribution).
- If step size is too small  $\Rightarrow$  takes much longer (bad *mixing* properties).

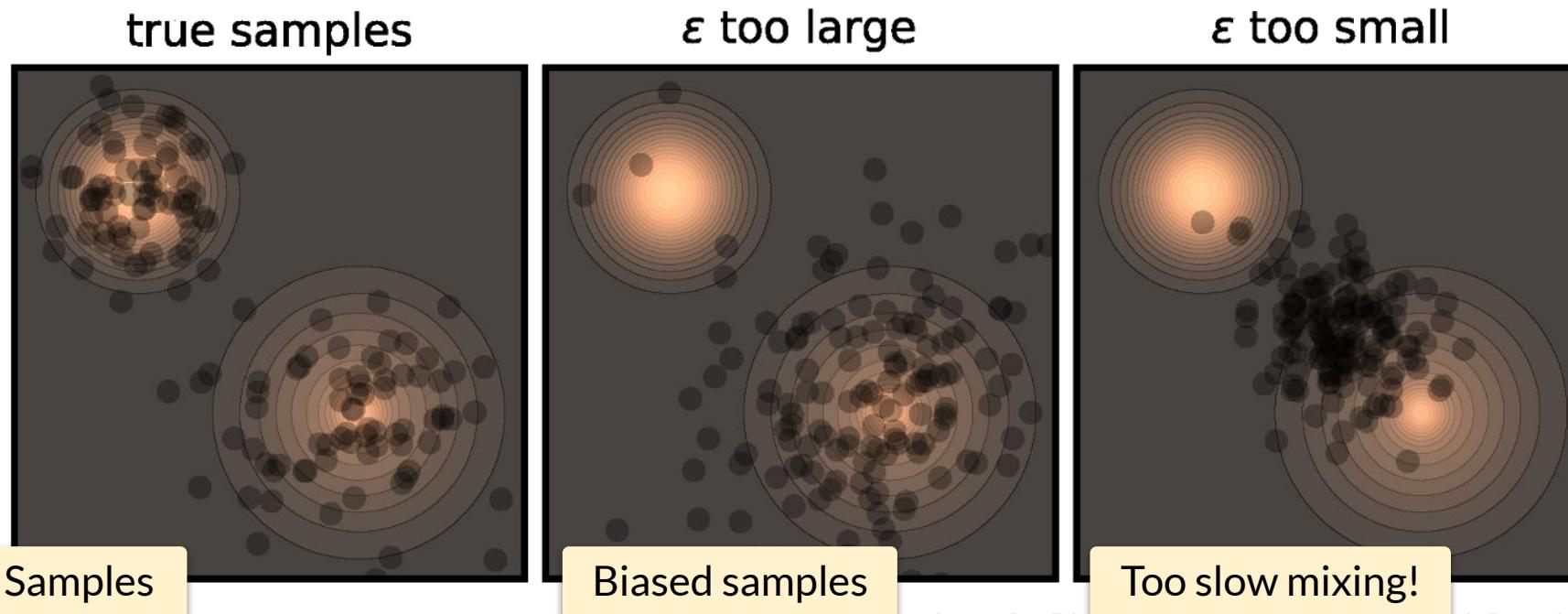


Source: Roy Friedman, "A Simplified Overview of Langevin Dynamics"

# Gradient Based MCMC – Langevin Monte Carlo (LMC)

Issues with this:

- If step size is too large  $\Rightarrow$  samples will be biased (incorrect stationary distribution).
- If step size is too small  $\Rightarrow$  takes much longer (bad *mixing* properties).



Source: Roy Friedman, "A Simplified Overview of Langevin Dynamics"

# Gradient Based MCMC – Langevin Monte Carlo (LMC)

A few solutions in practice:

# Gradient Based MCMC – Langevin Monte Carlo (LMC)

A few solutions in practice:

1. You can combine LMC with Metropolis-Hastings.

# Gradient Based MCMC – Langevin Monte Carlo (LMC)

A few solutions in practice:

1. You can combine LMC with Metropolis-Hastings.

In particular:

- Run LMC with a large step size (biased samples).
- Then accept/reject as you would typically do in Metropolis-Hastings.
- ⇒ Good mixing, and reasonable acceptance rate, even in high dimensions.

# Gradient Based MCMC – Langevin Monte Carlo (LMC)

A few solutions in practice:

1. You can combine LMC with Metropolis-Hastings.

In particular:

- Run LMC with a large step size (biased samples).
- Then accept/reject as you would typically do in Metropolis-Hastings.
- ⇒ Good mixing, and reasonable acceptance rate, even in high dimensions.

Note: this is often called MALA - Metropolis adjusted Langevin algorithm.

# Gradient Based MCMC – Langevin Monte Carlo (LMC)

A few solutions in practice:

2. You can anneal the step-size.

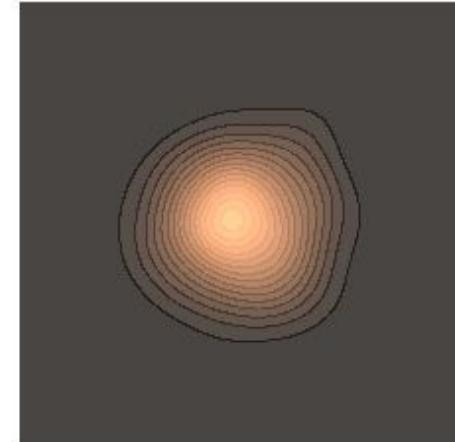
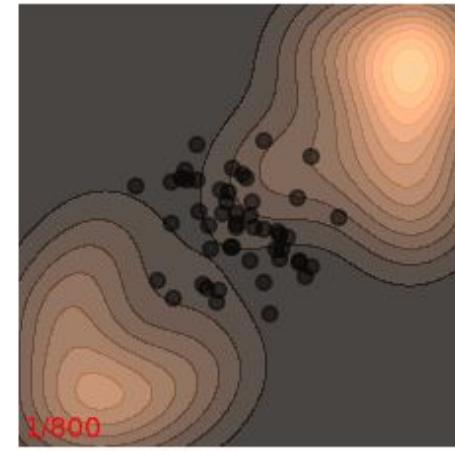
# Gradient Based MCMC – Langevin Monte Carlo (LMC)

A few solutions in practice:

2. You can anneal the step-size.

In particular:

- Start LMC with a large step size (biased samples).
- Then anneal the step size to a small value as you proceed with sampling.
- ⇒ You mix well at the start, and capture details of high-mass regions at the end.



# Gradient Based MCMC – Langevin Monte Carlo (LMC)

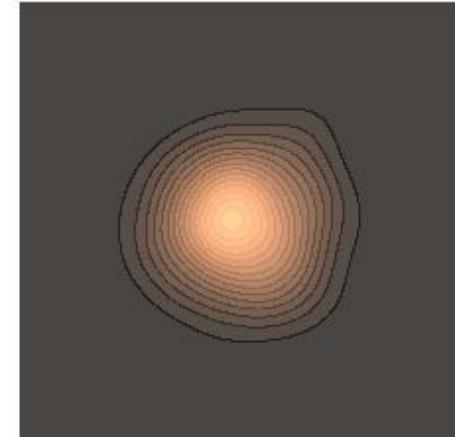
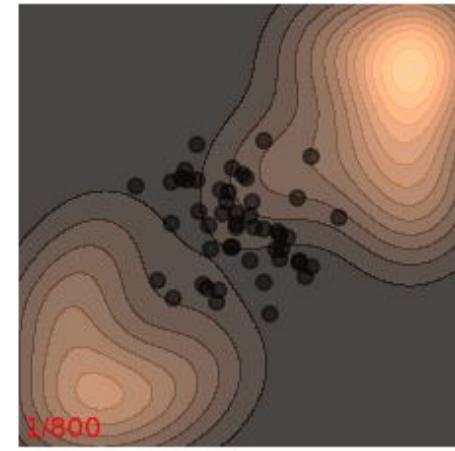
A few solutions in practice:

2. You can anneal the step-size.

In particular:

- Start LMC with a large step size (biased samples).
- Then anneal the step size to a small value as you proceed with sampling.
- ⇒ You mix well at the start, and capture details of high-mass regions at the end.

We will see that this strategy is used in variants of diffusion models, such as score-based generative models.



# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

A few solutions in practice:

3. Hamiltonian Monte Carlo (HMC)!

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

A few solutions in practice:

## 3. Hamiltonian Monte Carlo (HMC)!

In particular:

- Take a sequence of gradient steps (in a very specific way).
- Such that you can travel far away from your previous sample...
- ... but stay at a same / similar PDF value.

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

A few solutions in practice:

## 3. Hamiltonian Monte Carlo (HMC)!

In particular:

- Take a sequence of gradient steps (in a very specific way).
- Such that you can travel far away from your previous sample...
- ... but stay at a same / similar PDF value.
- (And then still use Metropolis-Hastings to accept/or reject final proposal).

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

A few solutions in practice:

## 3. Hamiltonian Monte Carlo (HMC)!

In particular:

- Take a sequence of gradient steps (in a very specific way).
- Such that you can travel far away from your previous sample...
- ... but stay at a same / similar PDF value.
- (And then still use Metropolis-Hastings to accept/or reject final proposal).
- ⇒ The potential for: even better mixing, and good acceptance rate, correct samples.

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

HMC Intuition (high level):

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

**HMC Intuition** (high level):

The goal of MCMC is to yield samples for Monte Carlo expectations of the form:

$$\begin{aligned}\mathbb{E}_{p(x)} [g(x)] &= \int p(x)g(x)dx \\ &= \frac{1}{T} \sum_{t=1}^T g(x^{(t)}), \quad x^{(t)} \sim p(x)\end{aligned}$$

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

HMC Intuition (high level):

The goal of MCMC is to yield samples for Monte Carlo expectations of the form:

$$\begin{aligned} \mathbb{E}_{p(x)} [g(x)] &= \int p(x)g(x)dx \\ &= \frac{1}{T} \sum_{t=1}^T g(x^{(t)}), \quad x^{(t)} \sim p(x) \end{aligned}$$

Expected value

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

HMC Intuition (high level):

The goal of MCMC is to yield samples for Monte Carlo expectations of the form:

$$\begin{aligned} \mathbb{E}_{p(x)} [g(x)] &= \int p(x)g(x)dx && \text{Expected value} \\ &= \frac{1}{T} \sum_{t=1}^T g(x^{(t)}), && x^{(t)} \sim p(x) && \text{Monte Carlo estimate,} \\ &&&&& \text{given samples} \end{aligned}$$

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

We will use a technique called *auxiliary variables*.

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

We will use a technique called *auxiliary variables*.

Note that, for a given probability model, you could always introduce *more variables*.

⇒ i.e., you can form a probability distribution over more variables whose marginal is equal to the original probability model).

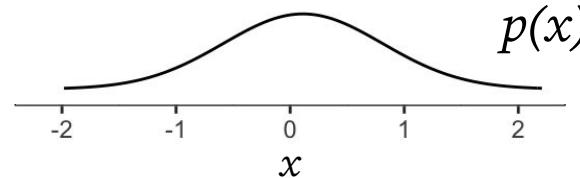
# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

We will use a technique called *auxiliary variables*.

Note that, for a given probability model, you could always introduce *more variables*.

⇒ i.e., you can form a probability distribution over more variables whose marginal is equal to the original probability model).

Suppose we have a probability model  $p(x)$



# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

We will use a technique called *auxiliary variables*.

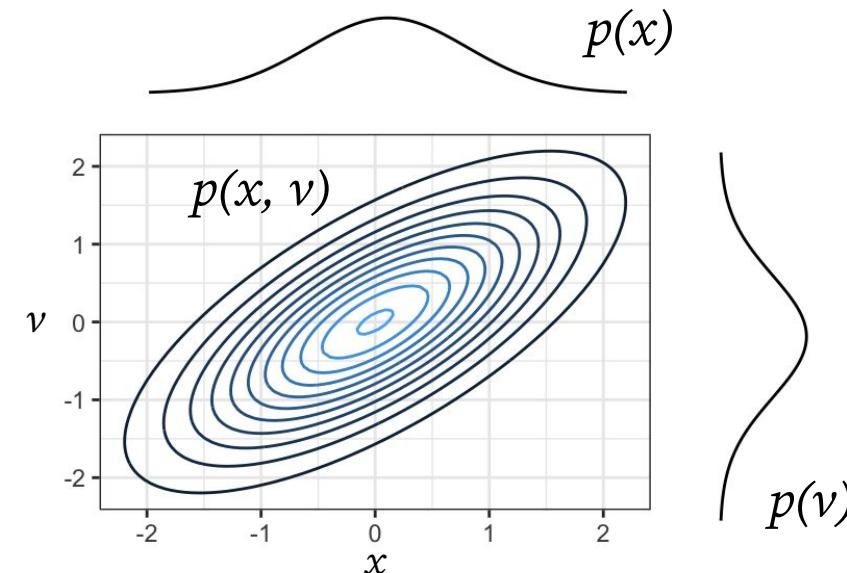
Note that, for a given probability model, you could always introduce *more variables*.

⇒ i.e., you can form a probability distribution over more variables whose marginal is equal to the original probability model).

Suppose we have a probability model  $p(x)$



Extend this to the model  $p(x, v)$



# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

We will use a technique called *auxiliary variables*.

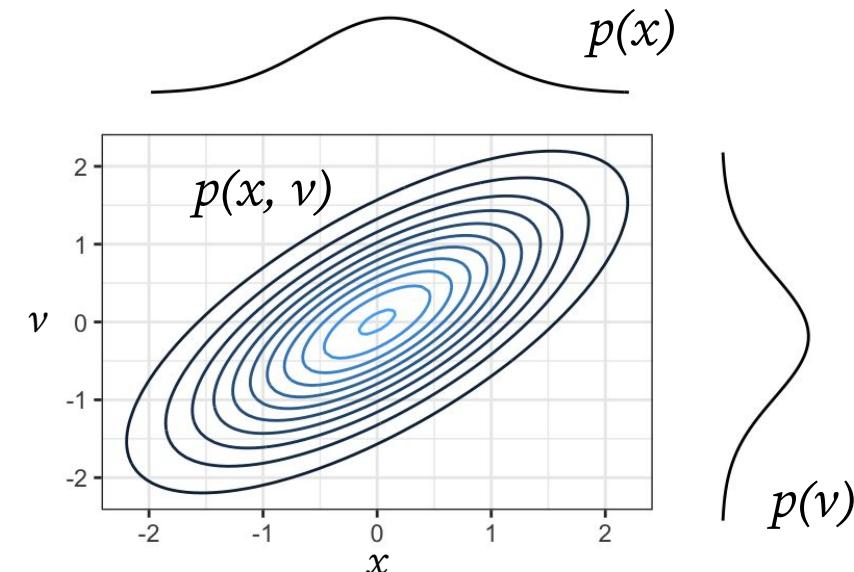
Note that, for a given probability model, you could always introduce *more variables*.

⇒ i.e., you can form a probability distribution over more variables whose marginal is equal to the original probability model).

Suppose we have a probability model  $p(x)$



Extend this to the model  $p(x, v)$



Variables  $v$  are called ***auxiliary variables***

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

When using auxiliary variables in the context of Monte Carlo sampling...

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

When using auxiliary variables in the context of Monte Carlo sampling...

This might look like:

$$\begin{aligned}\mathbb{E}_{p(x)} [g(x)] &= \int p(x)g(x)dx = \int p(x, v)g(x)dx dv \\ &= \frac{1}{T} \sum_{t=1}^T g(x^{(t)}), \quad x^{(t)}, v^{(t)} \sim p(x, v)\end{aligned}$$

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

When using auxiliary variables in the context of Monte Carlo sampling...

This might look like:

Expected value

$$\begin{aligned}\mathbb{E}_{p(x)} [g(x)] &= \int p(x)g(x)dx = \int p(x,v)g(x)dx\,dv \\ &= \frac{1}{T} \sum_{t=1}^T g(x^{(t)}), \quad x^{(t)}, v^{(t)} \sim p(x, v)\end{aligned}$$

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

When using auxiliary variables in the context of Monte Carlo sampling...

This might look like:

$$\begin{aligned} \mathbb{E}_{p(x)} [g(x)] &= \int p(x)g(x)dx = \int p(x, v)g(x)dx dv \\ &= \frac{1}{T} \sum_{t=1}^T g(x^{(t)}), \quad x^{(t)}, v^{(t)} \sim p(x, v) \end{aligned}$$

Expected value

Expected value given  
auxiliary variables

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

When using auxiliary variables in the context of Monte Carlo sampling...

This might look like:

$$\begin{aligned} \mathbb{E}_{p(x)} [g(x)] &= \int p(x)g(x)dx = \int p(x,v)g(x)dx\,dv \\ &= \frac{1}{T} \sum_{t=1}^T g(x^{(t)}), \quad x^{(t)}, v^{(t)} \sim p(x, v) \end{aligned}$$

Expected value

Expected value given  
auxiliary variables

Monte Carlo estimate  
(only uses  $x$  samples!)

## Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

In HMC, we define auxiliary variables such that (roughly speaking)...

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

In HMC, we define auxiliary variables such that (roughly speaking)...

You can write out the joint PDF in terms of a *Hamiltonian* (from physics/mechanics):

$$\begin{aligned} p(x, v) \propto p(x)p(v) &= e^{\log p(x)} e^{-\frac{1}{2}v^\top v} \\ &= e^{-E(x)} e^{-K(v)} \\ &= e^{-E(x)-K(v)} \\ &= e^{-H(x,v)} \end{aligned}$$

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

In HMC, we define auxiliary variables such that (roughly speaking)...

You can write out the joint PDF in terms of a *Hamiltonian* (from physics/mechanics):

$$\begin{aligned} p(x, v) \propto p(x)p(v) &= e^{\log p(x)} e^{-\frac{1}{2}v^\top v} \\ &= e^{-E(x)} e^{-K(v)} \\ &= e^{-E(x)-K(v)} \\ &= e^{-H(x,v)} \end{aligned}$$

Auxiliary variable  $v$  is typically Gaussian, is independent from  $x$ , and is viewed as *momentum*.

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

In HMC, we define auxiliary variables such that (roughly speaking)...

You can write out the joint PDF in terms of a *Hamiltonian* (from physics/mechanics):

$$\begin{aligned} p(x, v) \propto p(x)p(v) &= e^{\log p(x)} e^{-\frac{1}{2}v^\top v} \\ &= e^{-E(x)} e^{-K(v)} \\ &= e^{-E(x)-K(v)} \\ &= e^{-H(x,v)} \end{aligned}$$

A potential energy term and a kinetic energy term

Auxiliary variable  $v$  is typically Gaussian, is independent from  $x$ , and is viewed as *momentum*.

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

In HMC, we define auxiliary variables such that (roughly speaking)...

You can write out the joint PDF in terms of a *Hamiltonian* (from physics/mechanics):

$$\begin{aligned} p(x, v) \propto p(x)p(v) &= e^{\log p(x)} e^{-\frac{1}{2}v^\top v} \\ &= e^{-E(x)} e^{-K(v)} \\ &= e^{-E(x)-K(v)} \\ &= e^{-H(x,v)} \end{aligned}$$

A potential energy term and a kinetic energy term

Auxiliary variable  $v$  is typically Gaussian, is independent from  $x$ , and is viewed as *momentum*.

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

In HMC, we define auxiliary variables such that (roughly speaking)...

You can write out the joint PDF in terms of a *Hamiltonian* (from physics/mechanics):

$$\begin{aligned} p(x, v) \propto p(x)p(v) &= e^{\log p(x)} e^{-\frac{1}{2}v^\top v} \\ &= e^{-E(x)} e^{-K(v)} \\ &= e^{-E(x)-K(v)} \\ &= e^{-H(x,v)} \end{aligned}$$

A potential energy term and a kinetic energy term

Auxiliary variable  $v$  is typically Gaussian, is independent from  $x$ , and is viewed as *momentum*.

We call  $H$  the *Hamiltonian*

## Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

In HMC, we define auxiliary variables such that (roughly speaking)...

⇒ Then you can simulate *Hamiltonian Dynamics*.

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

In HMC, we define auxiliary variables such that (roughly speaking)...

- ⇒ Then you can simulate *Hamiltonian Dynamics*.
- ⇒ ( i.e., solve a differential equation to preserve the Hamiltonian energy...equivalent to keeping the same PDF value under the joint model  $p(x, v)$  . )

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

In HMC, we define auxiliary variables such that (roughly speaking)...

- ⇒ Then you can simulate *Hamiltonian Dynamics*.
- ⇒ ( i.e., solve a differential equation to preserve the Hamiltonian energy...equivalent to keeping the same PDF value under the joint model  $p(x, \nu)$  . )
- ⇒ In practice: iteratively update  $x$  and  $\nu$  according to some update rules... e.g.:

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

In HMC, we define auxiliary variables such that (roughly speaking)...

- ⇒ Then you can simulate *Hamiltonian Dynamics*.
- ⇒ ( i.e., solve a differential equation to preserve the Hamiltonian energy...equivalent to keeping the same PDF value under the joint model  $p(x, v)$  . )
- ⇒ In practice: iteratively update  $x$  and  $v$  according to some update rules... e.g.:

$$v \leftarrow v - \frac{\epsilon}{2} \nabla_x \log p(x)$$

$$x \leftarrow x + \epsilon v$$

$$v \leftarrow v - \frac{\epsilon}{2} \nabla_x \log p(x)$$

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

In HMC, we define auxiliary variables such that (roughly speaking)...

- ⇒ Then you can simulate *Hamiltonian Dynamics*.
- ⇒ ( i.e., solve a differential equation to preserve the Hamiltonian energy...equivalent to keeping the same PDF value under the joint model  $p(x, v)$  . )
- ⇒ In practice: iteratively update  $x$  and  $v$  according to some update rules... e.g.:

Called the *leapfrog method*.

$$v \leftarrow v - \frac{\epsilon}{2} \nabla_x \log p(x)$$

$$x \leftarrow x + \epsilon v$$

$$v \leftarrow v - \frac{\epsilon}{2} \nabla_x \log p(x)$$

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

In HMC, we define auxiliary variables such that (roughly speaking)...

- ⇒ Then you can simulate *Hamiltonian Dynamics*.
- ⇒ ( i.e., solve a differential equation to preserve the Hamiltonian energy...equivalent to keeping the same PDF value under the joint model  $p(x, v)$  . )
- ⇒ In practice: iteratively update  $x$  and  $v$  according to some update rules... e.g.:

Called the *leapfrog method*.

$$v \leftarrow v - \frac{\epsilon}{2} \nabla_x \log p(x)$$

$$x \leftarrow x + \epsilon v$$

$$v \leftarrow v - \frac{\epsilon}{2} \nabla_x \log p(x)$$

Not a perfect solution to Hamiltonian dynamics (a numerical approximation).

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

So to take a step (generate next sample in MCMC):

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

So to take a step (generate next sample in MCMC):

- First you perturb the auxiliary variable  $\nu$ .
  - Sample from a Gaussian distribution centered on current  $\nu$ .

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

So to take a step (generate next sample in MCMC):

- First you perturb the auxiliary variable  $\nu$ .
  - Sample from a Gaussian distribution centered on current  $\nu$ .
- Then you simulate Hamiltonian dynamics from this new point.
  - Following updates on previous slide → yields an updated  $(x, \nu)$  with similar PDF value.

# Gradient Based MCMC – Hamiltonian Monte Carlo (HMC)

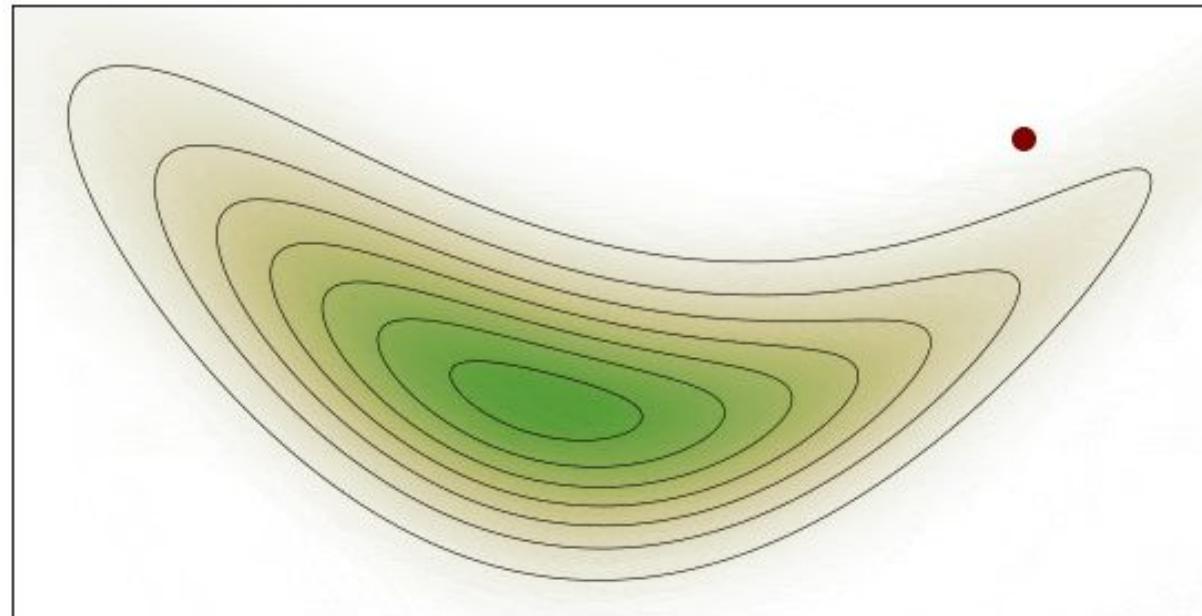
So to take a step (generate next sample in MCMC):

- First you perturb the auxiliary variable  $\nu$ .
  - Sample from a Gaussian distribution centered on current  $\nu$ .
- Then you simulate Hamiltonian dynamics from this new point.
  - Following updates on previous slide → yields an updated  $(x, \nu)$  with similar PDF value.
- Then you do an accept/reject step (Metropolis-Hastings) to convert this into a valid MCMC algorithm (asymptotically exact samples).

# Gradient Based MCMC

**Animation:**

HMC in practice!

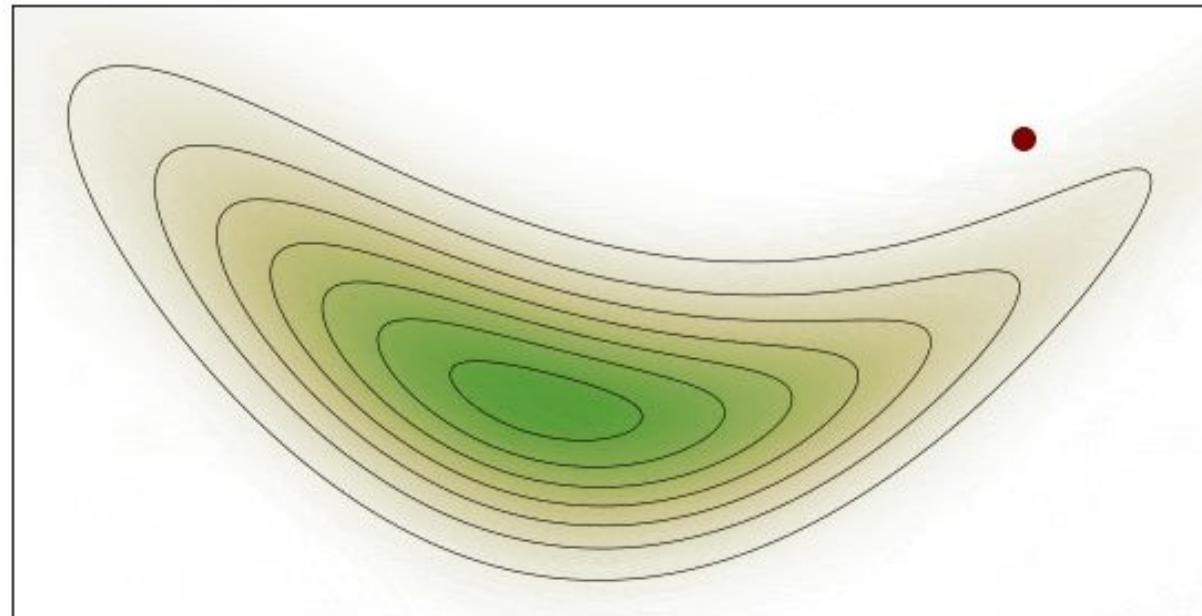


Source: Wikipedia, "Hamiltonian Monte Carlo"

# Gradient Based MCMC

**Animation:**

HMC in practice!



Source: Wikipedia, "Hamiltonian Monte Carlo"

Also, check out the 3d visualization here:

[https://arogozhnikov.github.io/2016/12/19/markov\\_chain\\_monte\\_carlo.html](https://arogozhnikov.github.io/2016/12/19/markov_chain_monte_carlo.html)

# MCMC Demo Website

Check out this MCMC Demo by Chi Feng

- <https://chi-feng.github.io/mcmc-demo/app.html>
- <https://github.com/chi-feng/mcmc-demo>

# Stochastic Gradient MCMC

Very quickly – a note on stochastic gradient MCMC.

# Stochastic Gradient MCMC

Very quickly – a note on stochastic gradient MCMC.

- Similar to how SGD (*stochastic gradient descent*) is a stochastic/scalable version of gradient descent.
  - (I.e., stochastic gradient equal to the gradient in expectation, but is noisy and calculator using a subsample of the data.)

# Stochastic Gradient MCMC

Very quickly – a note on stochastic gradient MCMC.

- Similar to how SGD (*stochastic gradient descent*) is a stochastic/scalable version of gradient descent.
  - (I.e., stochastic gradient equal to the gradient in expectation, but is noisy and calculator using a subsample of the data.)
- There exists a stochastic/scalable version of gradient-based MCMC methods.
  - ... which use stochastic gradients instead of full/deterministic gradients.

# Stochastic Gradient MCMC

Very quickly – a note on stochastic gradient MCMC.

- Similar to how SGD (*stochastic gradient descent*) is a stochastic/scalable version of gradient descent.
  - (I.e., stochastic gradient equal to the gradient in expectation, but is noisy and calculator using a subsample of the data.)
- There exists a stochastic/scalable version of gradient-based MCMC methods.
  - ... which use stochastic gradients instead of full/deterministic gradients.
- This is particularly useful when running MCMC for posterior inference given a large dataset, since you only use a subsample of data at each step.

# Stochastic Gradient MCMC

A few papers of note:

# Stochastic Gradient MCMC

A few papers of note:

2011

**Bayesian Learning via Stochastic Gradient Langevin Dynamics**

---

Max Welling [WELLING@ICS.UCL.AC.UK](mailto:WELLING@ICS.UCL.AC.UK)  
D. Bren School of Information and Computer Science, University of California, Irvine, CA 92697-3425, USA

Yee Whye Teh [YWTBTR@GATSBY.UCL.AC.UK](mailto:YWTBTR@GATSBY.UCL.AC.UK)  
Gatsby Computational Neuroscience Unit, UCL, 17 Queen Square, London WC1N 3AR, UK

**Abstract**

In this paper we propose a new framework for learning from large scale datasets based on iterative learning from small mini-batches. By adding the right amount of noise to a standard stochastic gradient optimization algorithm we show that the iterates will converge to samples from the true posterior distribution as we anneal the stepsize. This seamless transition between optimization and Bayesian inference provides an in-built protection against overfitting. We also propose a practical method for Monte Carlo estimates of posterior statistics which monitors a "sampling threshold" and collects samples after it has been passed. We apply the method to three models: a mixture of Gaussians, logistic regression and ICA with natural gradients.

**1. Introduction**

In recent years there has been an increasing amount of very large scale machine learning data, ranging from internet traffic and network data, computer vision, natural language processing, to bioinformatics. More and more advances in machine learning are now driven by these large scale data, which offers the opportunity to learn large and complex models for solving important real world tasks. Recent advances in large scale machine learning have mostly been optimization based approaches. While there are sophisticated algorithms designed specifically for certain types of models, one of the most successful class of algorithms are stochastic optimization, or Robbins-Monro, algorithms. These algorithms process small (mini-)batches of data at each iteration, updating model parameters by taking small gradient steps in a cost function. Often these algorithms are run in an online setting, where the data batches are discarded after processing and only one pass through the data is performed, reducing memory requirements drastically. One class of methods "left-behind" by the recent advances in large scale machine learning are the Bayesian methods. This has partially to do with the negative results in Bayesian online parameter estimation (Andrieu et al., 1999), but also the fact that each iteration of a typical Markov chain Monte Carlo (MCMC) algorithm requires re-computation of the posterior. Nevertheless, Bayesian methods are appealing in their ability to capture uncertainty in learned parameters and avoid overfitting. Arguably with large datasets there will be little overfitting. Alternatively, as we have access to larger datasets and more computational resources, we become interested in building more complex models, so that there will always be a need to quantify the amount of parameter uncertainty.

In this paper, we propose a method for Bayesian learning from large scale datasets. Our method combines Robbins-Monro type algorithms which stochastically optimize a likelihood, with Langevin dynamics which injects noise into the parameter updates in such a way that the trajectory of the iterates will converge to the full posterior distribution rather than just the maximum a posteriori mode. The resulting algorithm starts off being similar to stochastic optimization, then automatically transitions to one that simulates samples from the posterior using Langevin dynamics.

In Section 2 we introduce the two ingredients of our method: stochastic optimization and Langevin dynamics. Section 3 describes our algorithm and how it converges to the posterior distribution. Section 4 describes a practical method of estimating when our algorithm will transition from stochastic optimization to Langevin dynamics. Section 5 demonstrates our al-

Appearing in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011.  
Copyright 2011 by the author(s)/owner(s).

# Stochastic Gradient MCMC

A few papers of note:

2011

## Bayesian Learning via Stochastic Gradient Langevin Dynamics

Max Welling

WELLING@ICS.UCL.AC.UK  
D. Bren School of Information and Computer Science, University of California, Irvine, CA 92697-3425, USA

Yee Whye Teh

YWTBTR@GATSBY.UCL.AC.UK  
Gatsby Computational Neuroscience Unit, UCL, 17 Queen Square, London WC1N 3AR, UK

### Abstract

In this paper we propose a new framework for learning from large scale datasets based on iterative learning from small mini-batches. By adding the right amount of noise to a standard stochastic gradient optimization algorithm we show that the iterates will converge to samples from the true posterior distribution as we anneal the stepsize. This seamless transition between optimization and Bayesian inference provides an built-in protection against overfitting. We also propose a practical method for Monte Carlo estimates of posterior statistics which monitors a "sampling threshold" and collects samples after it has been passed. We apply the method to three models: a mixture of Gaussians, logistic regression and ICA with natural gradients.

### 1. Introduction

In recent years there has been an increasing amount of very large scale machine learning datasets, ranging from internet traffic and network data, computer vision, natural language processing, to bioinformatics. More and more advances in machine learning are now driven by these large scale data, which offers the opportunity to learn large and complex models for solving problems at scale. Recent advances in large scale machine learning have mostly been optimization based approaches. While there are sophisticated algorithms designed specifically for certain types of models, one of the most successful class of algorithms are stochastic optimization, or Robbins-Monro, algorithms. These algorithms process small (mini-

Appearing in *Proceedings of the 28<sup>th</sup> International Conference on Machine Learning*, Bellevue, WA, USA, 2011. Copyright 2011 by the author(s)/owner(s).

## Stochastic Gradient Hamiltonian Monte Carlo

Tianqi Chen

Emily B. Fox

Carlos Guestrin

MODE Lab, University of Washington, Seattle, WA.

2014

TQCHEN@CS.WASHINGTON.EDU  
EBFOX@STAT.WASHINGTON.EDU  
GUESTRIN@CS.WASHINGTON.EDU

### Abstract

Hamiltonian Monte Carlo (HMC) sampling methods provide a mechanism for defining discrete proposals with high acceptance probabilities in a Metropolis-Hastings framework, enabling more efficient exploration of the state space than standard random-walk proposals. The popularity of such methods has grown significantly in recent years. However, a limitation of HMC methods is the required gradient computation for simulation of the Hamiltonian dynamical system—such computation is infeasible in problems involving a large sample size or streaming data. Instead, we must rely on a noisy gradient estimate computed from a subset of the data. In this paper, we explore the properties of such a stochastic gradient HMC approach. Surprisingly, the natural implementation of the stochastic approximation can be arbitrarily bad. To address this problem we introduce a variant that uses second-order Langevin dynamics with a friction term that counteracts the effects of the noisy gradient, maintaining the desired rate of convergence in a noisy environment. Results on simulated data validate our theory. We also provide an application of our methods to a classification task using neural networks and to online Bayesian matrix factorization.

### 1. Introduction

Hamiltonian Monte Carlo (HMC) (Duane et al., 1987; Neal, 2010) sampling methods provide a powerful Markov chain Monte Carlo (MCMC) sampling algorithm. The methods define a Hamiltonian function in terms of the target distribution from which we desire samples—the potential energy—and a kinetic energy term parameterized by a set of “momentum” auxiliary variables. Based on

*Proceedings of the 31<sup>st</sup> International Conference on Machine Learning*, Beijing, China, 2014. JMLR: W&CP volume 32. Copyright 2014 by the author(s).

simple updates to the momentum variables, one simulates from a Hamiltonian dynamical system that enables proposals of distant states. The target distribution is invariant under these dynamics; in practice, a discretization of the continuous-time system is needed necessitating a Metropolis-Hastings (MH) correction, though still with high acceptance probability. Based on the attractive properties of HMC in terms of rapid exploration of the state space, HMC methods have grown in popularity recently (Neal, 2010; Hoffman & Gelman, 2011; Wang et al., 2013). A limitation of HMC, however, is the necessity to compute the gradient of the potential energy function in order to simulate the Hamiltonian dynamical system. We are increasingly faced with datasets that are massive in terms of observations or where data come in as a stream and we need to make inferences online, such as in online advertising or recommender systems. In these ever-more-common scenarios of massive batch or streaming data, such gradient computations are infeasible since they utilize the entire dataset, and thus are not applicable to “big data” problems. Recently, in a variety of machine learning algorithms, we have witnessed the many successes of utilizing a noisy estimate of the gradient as a *noisefilter* in order to scale the algorithm (Robbins & Monro, 1951; Hoffman et al., 2013; Welling & Teh, 2011). A majority of these developments have been in optimization-based algorithms (Robbins & Monro, 1951; Nemirovski et al., 2009), and a question is whether similar efficiencies can be garnered by sampling-based algorithms that maintain many desirable theoretical properties for Bayesian inference. One attempt at applying such methods in a sampling context is the recently proposed stochastic gradient Langevin dynamics (SGLD) (Welling & Teh, 2011; Ahn et al., 2012; Paterson & Teh, 2013). This method builds on first-order Langevin dynamics that do not include the crucial momentum term of HMC.

In this paper, we explore the possibility of marrying the efficiencies in state space exploration of HMC with the big-data computational efficiencies of stochastic gradients. Such an algorithm would enable a large-scale and online

# Stochastic Gradient MCMC

A few papers of note:

2011

## Bayesian Learning via Stochastic Gradient Langevin Dynamics

Max Welling

D. Bren School of Information and Computer Science, University of California, Irvine, CA 92697-3425, USA

Yee Whye Teh

Gatsby Computational Neuroscience Unit, UCL, 17 Queen Square, London WC1N 3AR, UK

### Abstract

In this paper we propose a new framework for learning from large scale datasets based on iterative learning from small mini-batches. By adding the right amount of noise to a standard stochastic gradient optimization algorithm we show that the iterates will converge to samples from the true posterior distribution as we anneal the stepsize. This seamless transition between optimization and Bayesian posterior sampling provides an in-built protection against overfitting. We also propose a practical method for Monte Carlo estimates of posterior statistics which monitors a “sampling threshold” and collects samples after it has been surpassed. We apply the method to three models: a mixture of Gaussians, logistic regression and ICA with natural gradients.

### 1. Introduction

In recent years there has been an increasing amount of very large scale machine learning datasets, ranging from internet traffic and network data, computer vision, natural language processing, to bioinformatics. More and more advances in machine learning are now driven by these large scale data, which offers the opportunity to learn large and complex models for solving important real world tasks. Recent advances in large scale machine learning have mostly been optimization-based approaches. While there are sophisticated algorithms designed specifically for certain types of models, one of the most successful class of algorithms are stochastic optimization, or Robbins-Monro, algorithms. These algorithms process small (mini-

Appearing in *Proceedings of the 28<sup>th</sup> International Conference on Machine Learning*, Bellevue, WA, USA, 2011. Copyright 2011 by the author(s)/owner(s).

batches of data at each iteration, updating model parameters by taking small gradient steps in a cost function. Often these algorithms are run in an online setting, where the data batches are discarded after processing and only one pass through the data is performed, reducing memory requirements drastically. One class of methods ‘left-behind’ by the recent advances in large scale machine learning are the Bayesian methods. This has partially to do with the negative results in Bayesian online parameter estimation (Andrieu et al., 1999), but also the fact that each iteration of a typical Markov chain Monte Carlo (MCMC) algorithm requires full posterior sampling of the state vector. Nevertheless, Bayesian methods are appealing in their ability to capture uncertainty in learned parameters and avoid overfitting. Arguably with large datasets there will be little overfitting. Alternatively, as we have access to larger datasets and more computational resources, we become interested in building more complex models, so that these will always be a need to quantify the amount of parameter uncertainty.

In this paper, we propose a method for Bayesian learning from large scale datasets. Our method combines Robbins-Monro type algorithms which stochastically optimize a likelihood, with Langevin dynamics which injects noise into the parameter updates in such a way that the trajectory of the iterates will converge to the full posterior distribution rather than just the maximum a posteriori mode. The resulting algorithm starts off being similar to stochastic optimization, then automatically transitions to one that simulates samples from the posterior using Langevin dynamics.

In Section 2 we introduce the two ingredients of our method: stochastic optimization, and Langevin dynamics. Section 3 describes our algorithm and how it converges to the posterior distribution. Section 4 describes a practical method of estimating when our algorithm will transition from stochastic optimization to Langevin dynamics. Section 5 demonstrates our al-

## Stochastic Gradient Hamiltonian Monte Carlo

Tianqi Chen

Emily B. Fox

Carlos Guestrin

MODE Lab, University of Washington, Seattle, WA.

TQCHEN@CS.WASHINGTON.EDU  
EBFOX@STAT.WASHINGTON.EDU  
GUESTRIN@CS.WASHINGTON.EDU

### Abstract

Hamiltonian Monte Carlo (HMC) sampling methods provide a mechanism for defining discrete proposals with high acceptance probabilities in a Metropolis-Hastings framework, enabling more efficient exploration of the state space than standard random-walk proposals. The popularity of such methods has grown significantly in recent years. However, a limitation of HMC methods is the required gradient computation for simulation of the Hamiltonian dynamical system—such computation is expensive in problems involving a large sample size or streaming data. Instead, we must rely on a noisy gradient estimate computed from a subset of the data. In this paper, we explore the properties of such a stochastic gradient HMC approach. Surprisingly, the natural implementation of the stochastic approximation can be arbitrarily bad. To address this problem we introduce a variant that uses second-order Langevin dynamics with a friction term that counteracts the effects of the noisy gradient, maintaining the desired rate of convergence in a much more stable manner. Results on simulated data validate our theory. We also provide an application of our methods to a classification task using neural networks and to online Bayesian matrix factorization.

### 1. Introduction

Hamiltonian Monte Carlo (HMC) (Duane et al., 1987; Neal, 2010) sampling methods provide a powerful Markov chain Monte Carlo (MCMC) sampling algorithm. The methods define a Hamiltonian function in terms of the target distribution from which we desire samples—the potential energy—and a kinetic energy term parameterized by a set of “momentum” auxiliary variables. Based on

In this paper, we explore the possibility of marrying the efficiencies in state space exploration of HMC with the big-data computational efficiencies of stochastic gradients. Such an algorithm would enable a large-scale and online

2014

## A Complete Recipe for Stochastic Gradient MCMC

Yi-An Ma, Tianqi Chen, and Emily B. Fox

University of Washington {yianma@,tqchen@cs,ebfox@stat}.washington.edu

### Abstract

Many recent Markov chain Monte Carlo (MCMC) samplers leverage continuous dynamics to define a transition kernel that efficiently explores a target distribution. In tandem, a focus has been on devising variants that sample the data and update the model in place of full data passes in the discrete-time iterations. However, such stochastic gradient MCMC samplers have lagged behind their full-data counterparts in terms of the complexity of dynamics considered since proving convergence in the presence of the stochastic gradient noise is non-trivial. Even with simple dynamics, significant physical intuition is often required to modify the dynamical system to account for the stochastic gradient noise. In this paper, we propose a complete recipe for constructing such samplers—labeled “integrating stochastic gradient versions”—based on continuous Markov processes specified via two metrics. We constructively prove that the framework is complete. That is, any continuous Markov process that provides samples from the target distribution can be written in our framework. We show how previous continuous-dynamic samplers can trivially “rethink” in our framework, avoid the complicated sampling logic, and provide a flexible template for straight-forwardly proposing a new state-adaptive sampler: *stochastic gradient Riemann Hamiltonian Monte Carlo* (SGRHMC). Our experiments on simulated data and a streaming Wikipedia analysis demonstrate that the proposed SGRHMC sampler inherits the benefits of Riemann HMC, with the scalability of stochastic gradient methods.

### 1. Introduction

Markov chain Monte Carlo (MCMC) has become a defacto tool for Bayesian posterior inference. However, these methods naturally scale poorly in the context of large datasets. Such large-scale posterior inference challenges have seen a rise in MCMC methods that provide more efficient exploration of the posterior, such as Hamiltonian Monte Carlo (HMC) [8, 12] and its Riemann manifold variant [10]. This class of samplers is based on defining a *potential energy* function in terms of the target posterior distribution and then devising various continuous dynamics to explore the energy landscape, enabling proposals of distant states. The gain in efficiency of exploration often comes at the cost of a significant computational burden in large datasets.

Recently, stochastic gradient variants of such continuous-dynamic samplers have proven quite useful in scaling the methods to large datasets [17, 1, 6, 2, 7]. At each iteration, these samplers use data subsamples or *mini-batches* of the data. Stochastic gradient Langevin dynamics (SGLD) [17] improved this approach by combining stochastic optimization with a continuous dynamic MCMC technique, showing that adding the “right amount” of noise to stochastic gradient ascent iterates leads to samples from the target posterior as the step size is annealed. Stochastic gradient Hamiltonian Monte Carlo (SGHMC) [6] builds on this idea, but importantly incorporates the efficient exploration provided by the HMC momentum term. A key insight in that paper was that the naive stochastic gradient variant of HMC actually leads to an incorrect stationary distribution (also see [4]); instead a modification to the dynamics underlying HMC is needed to account for

2015

# **Variational Inference**

# Variational Inference – Overview

To begin, a few potential shortcomings of sampling-based approximate inference:

# Variational Inference – Overview

To begin, a few potential shortcomings of sampling-based approximate inference:

1. Although they are guaranteed to perform *correct inference* given enough time, it is difficult to tell the *approximation quality* after running for a finite amount of time.

# Variational Inference – Overview

To begin, a few potential shortcomings of sampling-based approximate inference:

1. Although they are guaranteed to perform *correct inference* given enough time, it is difficult to tell the *approximation quality* after running for a finite amount of time.
2. MCMC methods may require choosing an appropriate proposal distribution (or various hyperparameters). This choice can be difficult.

# Variational Inference – Overview

Instead, we could take the strategy/philosophy of *inference as optimization*.

This method of approximate inference is known as **variational inference (VI)**.

# Variational Inference – Overview

Instead, we could take the strategy/philosophy of *inference as optimization*.

This method of approximate inference is known as **variational inference (VI)**.

The main idea:

# Variational Inference – Overview

Instead, we could take the strategy/philosophy of *inference as optimization*.

This method of approximate inference is known as **variational inference (VI)**.

The main idea:

- Suppose we are given an intractable (*complex*) probability distribution  $p(x)$ .

# Variational Inference – Overview

Instead, we could take the strategy/philosophy of *inference as optimization*.

This method of approximate inference is known as **variational inference (VI)**.

The main idea:

- Suppose we are given an intractable (*complex*) probability distribution  $p(x)$ .
- VI aims to solve an optimization problem over a set of tractable distributions  $Q$ .

# Variational Inference – Overview

Instead, we could take the strategy/philosophy of *inference as optimization*.

This method of approximate inference is known as **variational inference (VI)**.

The main idea:

- Suppose we are given an intractable (*complex*) probability distribution  $p(x)$ .
- VI aims to solve an optimization problem over a set of tractable distributions  $Q$ .
- And, after optimization, return a  $q(x) \in Q$  that is *most similar* to  $p(x)$ .  
Typically: in terms  
of KL divergence

# Variational Inference – Overview

Instead, we could take the strategy/philosophy of *inference as optimization*.

This method of approximate inference is known as **variational inference (VI)**.

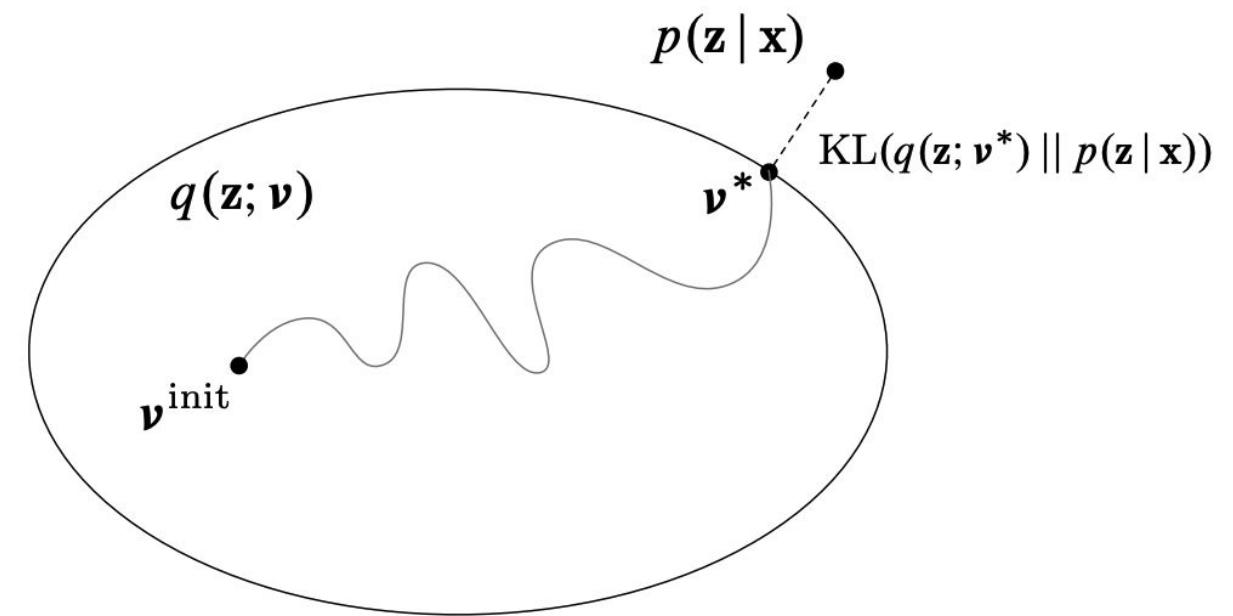
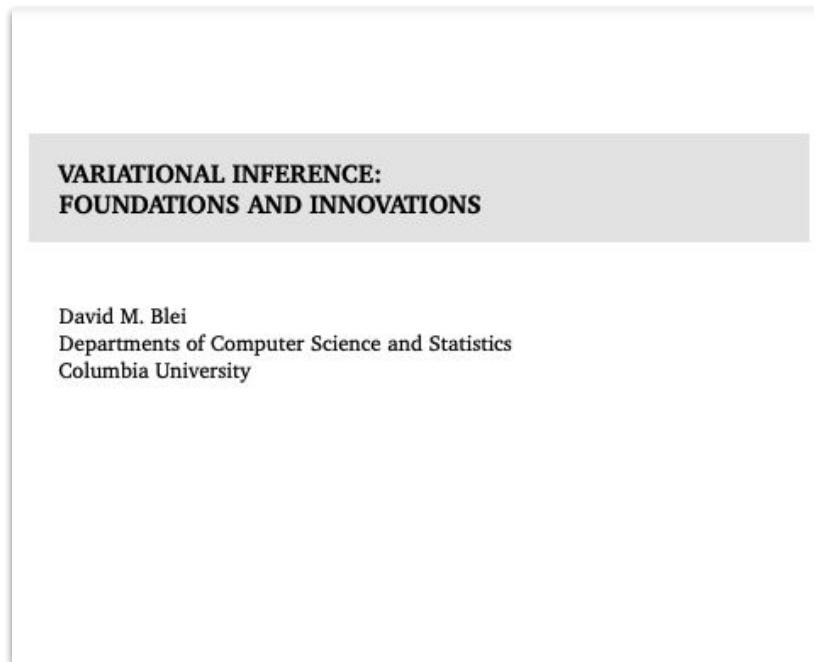
The main idea:

- Suppose we are given an intractable (*complex*) probability distribution  $p(x)$ .
- VI aims to solve an optimization problem over a set of tractable distributions  $Q$ .
- And, after optimization, return a  $q(x) \in Q$  that is *most similar* to  $p(x)$ .
- We can then make an inference query on  $q(x)$  instead of  $p(x)$ .

Typically: in terms  
of KL divergence

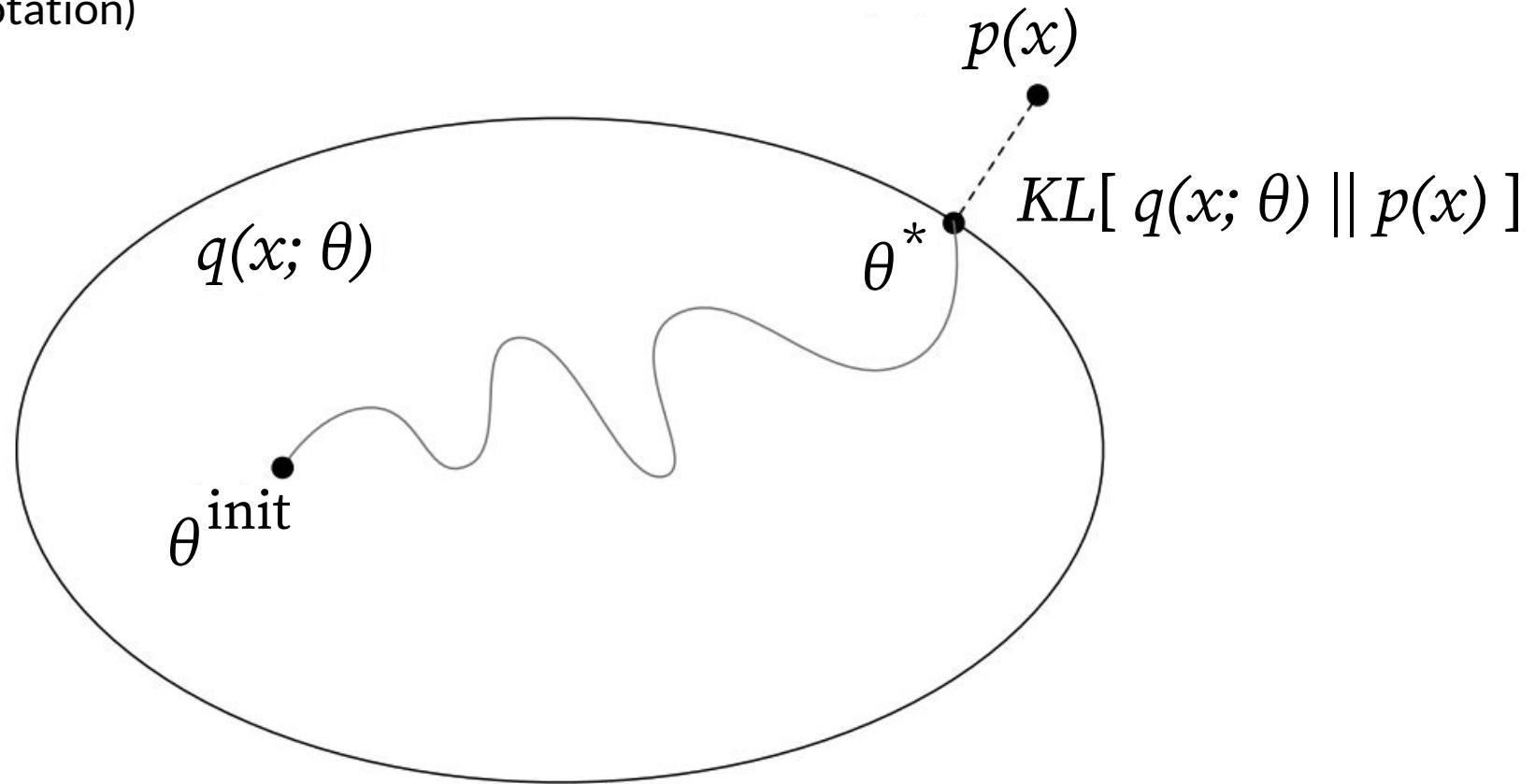
# Variational Inference – Overview

An illustration from Dave Blei's [Variational Inference Tutorial](#):



# Variational Inference – Overview

An illustration from Dave Blei's Variational Inference Tutorial:  
(replaced with our notation)



# Variational Inference – Overview

Main differences with sampling/MCMC methods:

# Variational Inference – Overview

Main differences with sampling/MCMC methods:

- Unlike sampling, variational methods will almost never find the exact/globally optimal solution (even in the limit as  $T \rightarrow \infty$ ).

# Variational Inference – Overview

Main differences with sampling/MCMC methods:

- Unlike sampling, variational methods will almost never find the exact/globally optimal solution (even in the limit as  $T \rightarrow \infty$ ).
- However, it is easy to determine if they have converged (at least to a local optima).

# Variational Inference – Overview

Main differences with sampling/MCMC methods:

- Unlike sampling, variational methods will almost never find the exact/globally optimal solution (even in the limit as  $T \rightarrow \infty$ ).
- However, it is easy to determine if they have converged (at least to a local optima).
- In practice, variational inference methods often scale better.

# Variational Inference – Overview

Main differences with sampling/MCMC methods:

- Unlike sampling, variational methods will almost never find the exact/globally optimal solution (even in the limit as  $T \rightarrow \infty$ ).
- However, it is easy to determine if they have converged (at least to a local optima).
- In practice, variational inference methods often scale better.
- ⇒ Amenable to techniques like stochastic gradient optimization, parallelization over multiple processors, and acceleration using GPUs.
  - (While it's easier for VI, this is also a popular field of study for MCMC).

# Kullback-Leibler Divergence

Quick background on Kullback-Leibler (KL) Divergence.

# Kullback-Leibler Divergence

Quick background on Kullback-Leibler (KL) Divergence.

To optimize for an optimal approximation  $q(x)$ , we must choose an approximating family  $Q$ , and an objective  $J(q)$ .

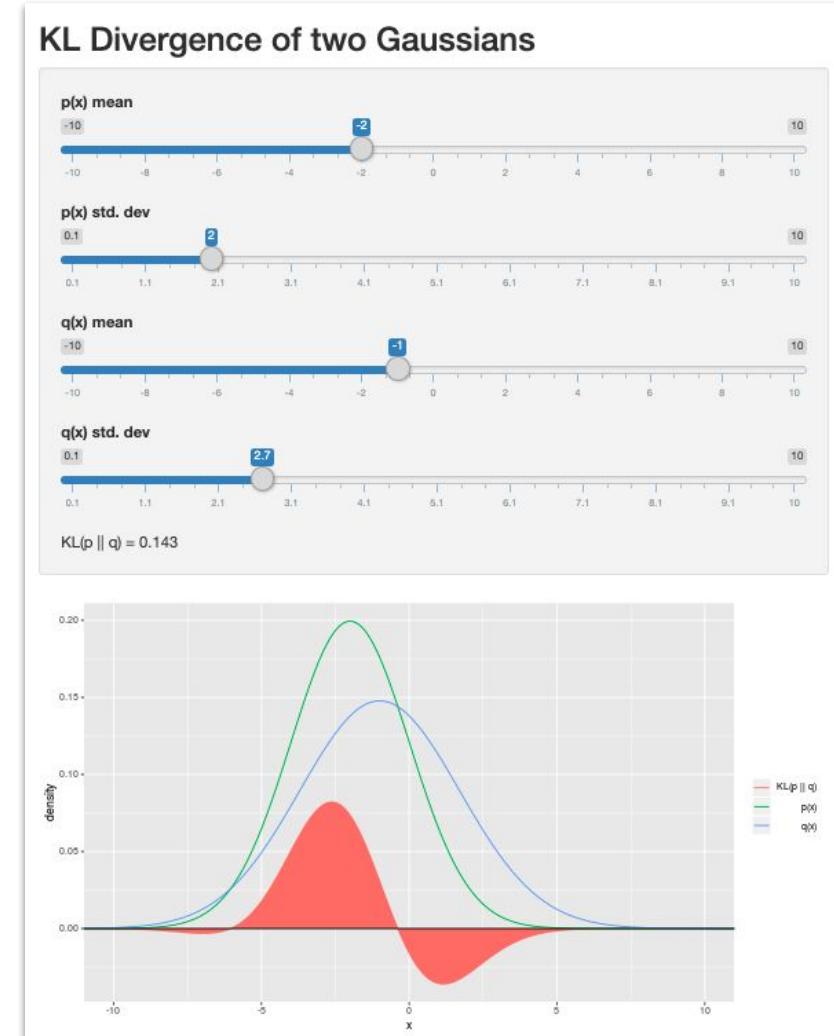
The objective should capture the similarity between  $q(x)$  and  $p(x)$ .

The field of information theory provides a tool: **Kullback-Leibler (KL) divergence**.

# Kullback-Leibler Divergence

*Intuitively + visually:*

It is a “divergence” (~distance) between distributions  $q(x)$  and  $p(x)$ :



*KL divergence online demo:*

<https://gnarlyware.com/blog/kl-divergence-online-demo/>

# Kullback-Leibler Divergence

More formally:

# Kullback-Leibler Divergence

More formally:

The KL divergence between distributions  $q$  and  $p$  with *discrete support* is:

$$\text{KL} [q \parallel p] = \sum_x q(x) \log \left( \frac{q(x)}{p(x)} \right) = \mathbb{E}_{q(x)} \left[ \log \left( \frac{q(x)}{p(x)} \right) \right]$$

# Kullback-Leibler Divergence

More formally:

The KL divergence between distributions  $q$  and  $p$  with *discrete support* is:

$$\text{KL} [q \parallel p] = \sum_x q(x) \log \left( \frac{q(x)}{p(x)} \right) = \mathbb{E}_{q(x)} \left[ \log \left( \frac{q(x)}{p(x)} \right) \right]$$

And the KL divergence between distributions  $q$  and  $p$  with *continuous support* is:

$$\text{KL} [q \parallel p] = \int_{x \in \mathcal{X}} q(x) \log \left( \frac{q(x)}{p(x)} \right) dx = \mathbb{E}_{q(x)} \left[ \log \left( \frac{q(x)}{p(x)} \right) \right]$$

# Kullback-Leibler Divergence

The KL divergence has a couple of properties that make it useful in this setting:

# Kullback-Leibler Divergence

The KL divergence has a couple of properties that make it useful in this setting:

- $\text{KL} [q \parallel p] \geq 0$  for all  $q, p$ .

# Kullback-Leibler Divergence

The KL divergence has a couple of properties that make it useful in this setting:

- $\text{KL}[q \parallel p] \geq 0$  for all  $q, p$ .
- $\text{KL}[q \parallel p] = 0$  if and only if  $q = p$ .

# Kullback-Leibler Divergence

The KL divergence has a couple of properties that make it useful in this setting:

- $\text{KL}[q \parallel p] \geq 0$  for all  $q, p$ .
- $\text{KL}[q \parallel p] = 0$  if and only if  $q = p$ .

Importantly, note that  $\text{KL}[q \parallel p] \neq \text{KL}[p \parallel q]$ .

$\Rightarrow$  i.e., KL divergence is **not symmetric**.

$\Rightarrow$  (this is why we say that it is a “divergence” and not a “distance”).

# Kullback-Leibler Divergence

Intuitively, when minimizing KL divergence in variational inference:  
(in the words of Dave Blei)

# Kullback-Leibler Divergence

$$\text{KL} [q \parallel p] = \sum_x q(x) \log \left( \frac{q(x)}{p(x)} \right)$$

Intuitively, when minimizing KL divergence in variational inference:  
(in the words of Dave Blei)

# Kullback-Leibler Divergence

$$\text{KL} [q \parallel p] = \sum_x q(x) \log \left( \frac{q(x)}{p(x)} \right)$$

Intuitively, when minimizing KL divergence in variational inference:

(in the words of Dave Blei)

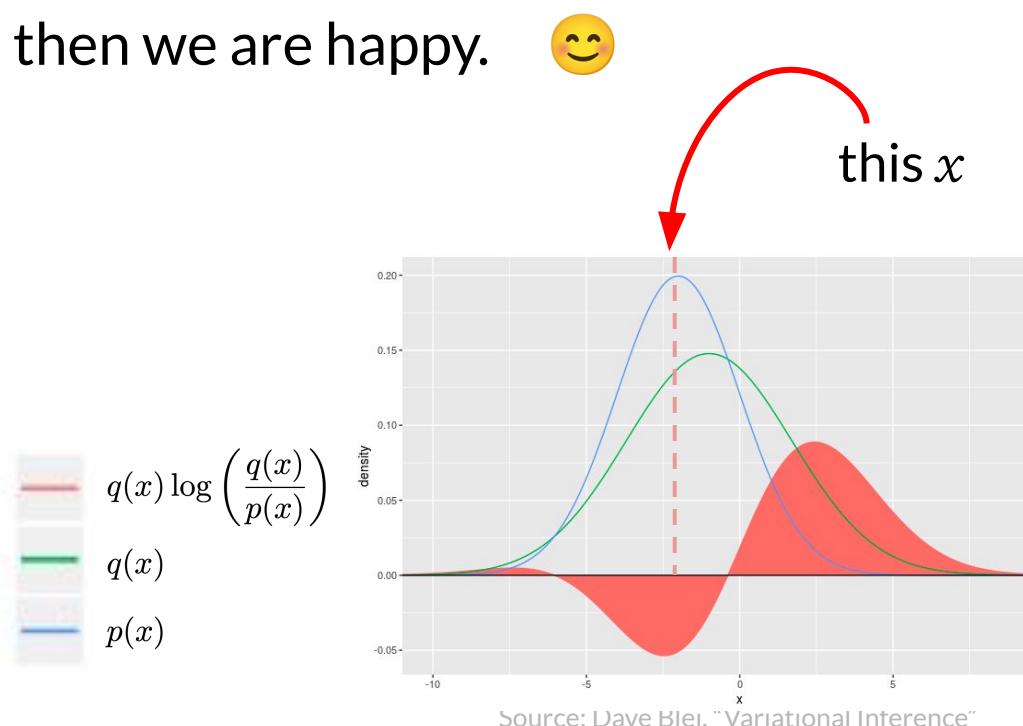
- If  $q(x)$  is high and  $p(x)$  is also high (or higher) then we are happy. 😊

# Kullback-Leibler Divergence

$$\text{KL} [q \parallel p] = \sum_x q(x) \log \left( \frac{q(x)}{p(x)} \right)$$

Intuitively, when minimizing KL divergence in variational inference:  
(in the words of Dave Blei)

- If  $q(x)$  is high and  $p(x)$  is also high (or higher) then we are happy.



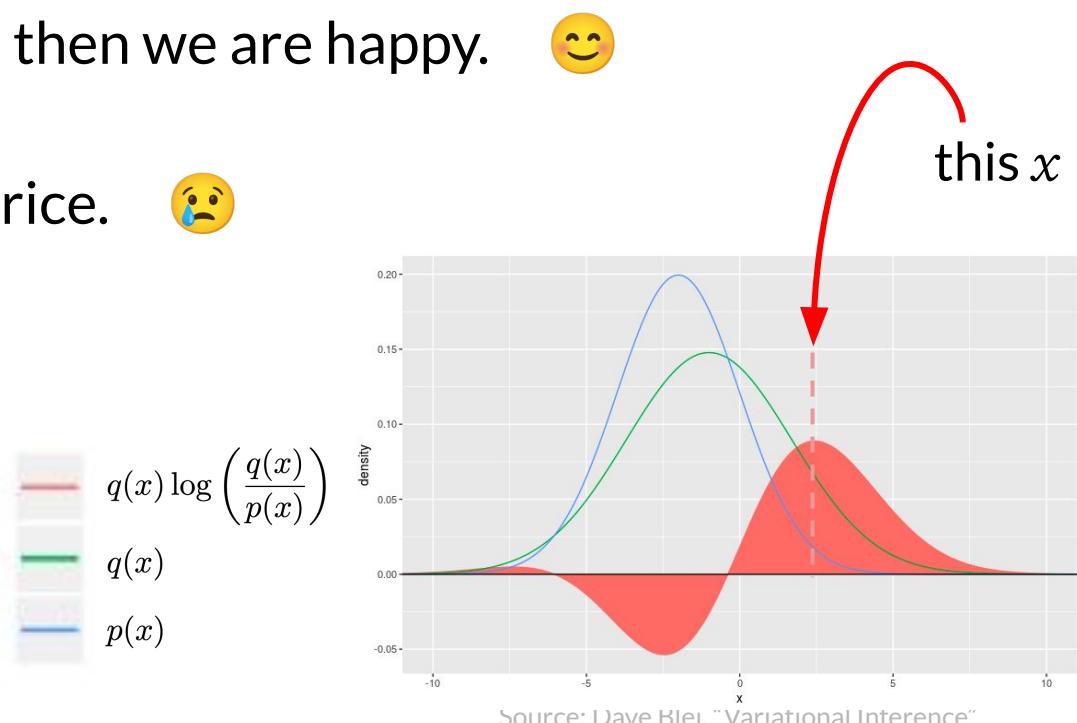
# Kullback-Leibler Divergence

$$\text{KL} [q \parallel p] = \sum_x q(x) \log \left( \frac{q(x)}{p(x)} \right)$$

Intuitively, when minimizing KL divergence in variational inference:

(in the words of Dave Blei)

- If  $q(x)$  is high and  $p(x)$  is also high (or higher) then we are happy. 😊
- If  $q(x)$  is high and  $p(x)$  is low, then we pay a price. 😢

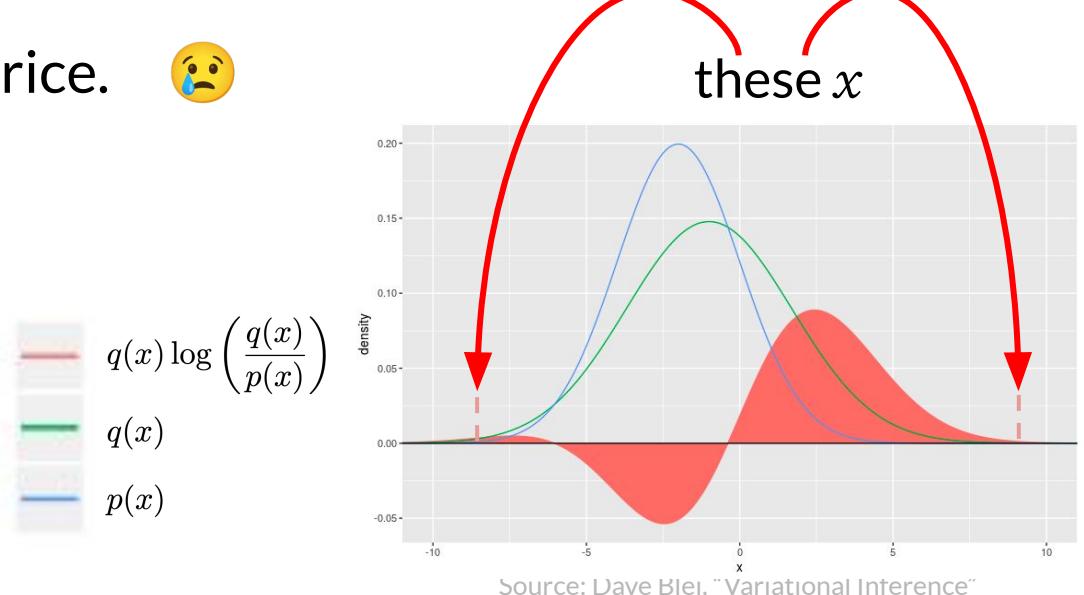
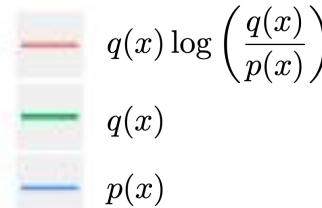


# Kullback-Leibler Divergence

$$\text{KL} [q \parallel p] = \sum_x q(x) \log \left( \frac{q(x)}{p(x)} \right)$$

Intuitively, when minimizing KL divergence in variational inference:  
(in the words of Dave Blei)

- If  $q(x)$  is high and  $p(x)$  is also high (or higher) then we are happy. 😊
- If  $q(x)$  is high and  $p(x)$  is low, then we pay a price. 😢
- If  $q(x)$  is low... then we don't care. 😐



# The Evidence Lower Bound

How do we perform variational inference with the KL divergence?

Next we will go through an overview of the main strategy.

# The Evidence Lower Bound

Suppose we have an arbitrary probabilistic model of  $n$  variables:  $x = (x_1, \dots, x_n)$

# The Evidence Lower Bound

Suppose we have an arbitrary probabilistic model of  $n$  variables:  $x = (x_1, \dots, x_n)$

Let's write this in a general way, as a (normalized) product of factors:

$$p_\theta(x_1, \dots, x_n) = \frac{1}{Z(\theta)} \tilde{p}_\theta(x_1, \dots, x_n) = \frac{1}{Z(\theta)} \prod_k \phi_k(x_k; \theta)$$

# The Evidence Lower Bound

Suppose we have an arbitrary probabilistic model of  $n$  variables:  $x = (x_1, \dots, x_n)$

Let's write this in a general way, as a (normalized) product of factors:

$$p_\theta(x_1, \dots, x_n) = \frac{1}{Z(\theta)} \tilde{p}_\theta(x_1, \dots, x_n) = \frac{1}{Z(\theta)} \prod_k \phi_k(x_k; \theta)$$

Assume  $p(x)$  has  
a parameter  $\theta$

# The Evidence Lower Bound

Suppose we have an arbitrary probabilistic model of  $n$  variables:  $x = (x_1, \dots, x_n)$

Let's write this in a general way, as a (normalized) product of factors:

$$p_\theta(x_1, \dots, x_n) = \frac{1}{Z(\theta)} \tilde{p}_\theta(x_1, \dots, x_n) = \frac{1}{Z(\theta)} \prod_k \phi_k(x_k; \theta)$$

Assume  $p(x)$  has  
a parameter  $\theta$

$Z$  is a normalization  
constant

# The Evidence Lower Bound

Suppose we have an arbitrary probabilistic model of  $n$  variables:  $x = (x_1, \dots, x_n)$

Let's write this in a general way, as a (normalized) product of factors:

$$p_\theta(x_1, \dots, x_n) = \frac{1}{Z(\theta)} \tilde{p}_\theta(x_1, \dots, x_n) = \frac{1}{Z(\theta)} \prod_k \phi_k(x_k; \theta)$$

Assume  $p(x)$  has  
a parameter  $\theta$

$Z$  is a normalization  
constant

Recall: can write most  
directed / undirected  
PGMs in this way.

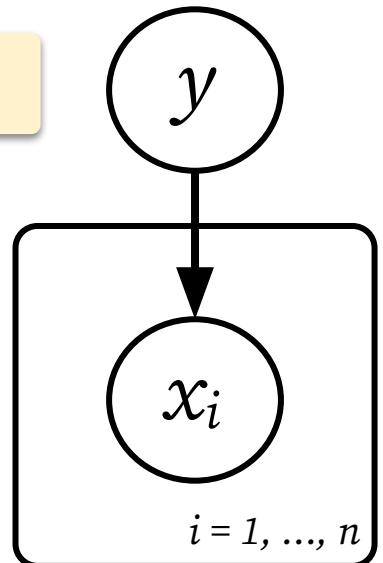
# The Evidence Lower Bound

Example prototypical PGM (Bayesian network):

# The Evidence Lower Bound

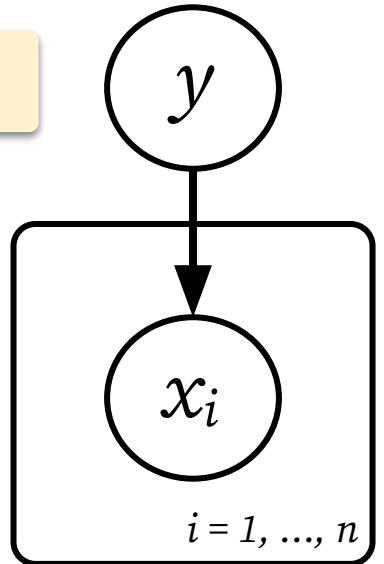
Example PGM

Example prototypical PGM (Bayesian network):



# The Evidence Lower Bound

Example PGM



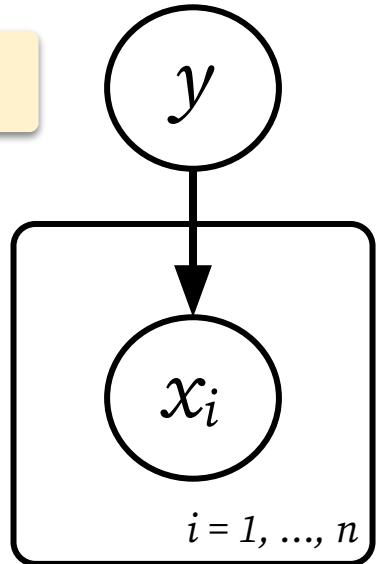
Example prototypical PGM (Bayesian network):

Can can write out the joint PDF as:

$$p(y, x_1, \dots, x_n) = p(y) \prod_i p(x_i | y)$$

# The Evidence Lower Bound

Example PGM



Example prototypical PGM (Bayesian network):

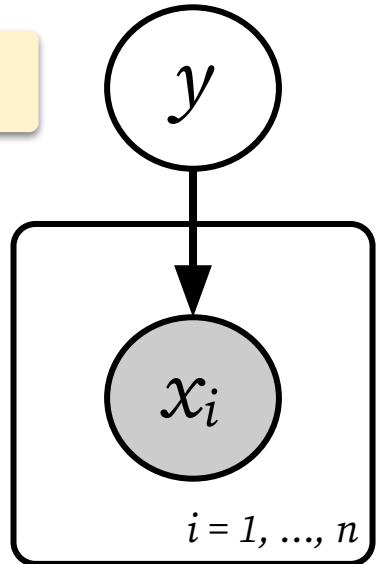
Can can write out the joint PDF as:

$$p(y, x_1, \dots, x_n) = p(y) \underbrace{\prod_i p(x_i | y)}$$

Product of factors

# The Evidence Lower Bound

Example PGM



Example prototypical PGM (Bayesian network):

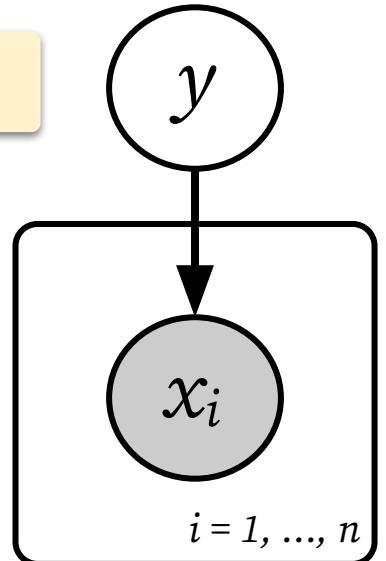
Can can write out the joint PDF as:

$$p(y, x_1, \dots, x_n) = p(y) \prod_i p(x_i | y)$$

Alternatively, we can write out the posterior PDF (prob of latents given observations) as:

# The Evidence Lower Bound

Example PGM



Example prototypical PGM (Bayesian network):

Can we write out the joint PDF as:

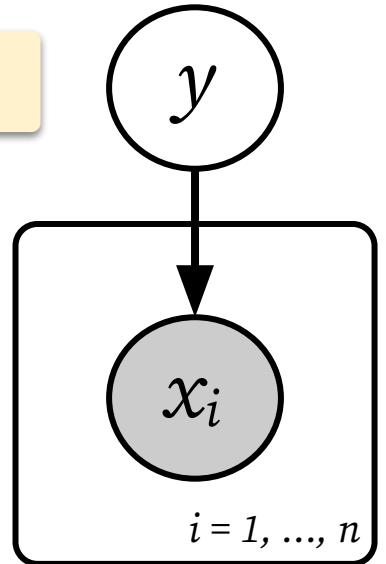
$$p(y, x_1, \dots, x_n) = p(y) \prod_i p(x_i | y)$$

Alternatively, we can write out the posterior PDF (prob of latents given observations) as:

$$p(y | x_1, \dots, x_n) = \frac{p(y, x_1, \dots, x_n)}{p(x_1, \dots, x_n)} = \frac{1}{Z} p(y, x_1, \dots, x_n)$$

# The Evidence Lower Bound

Example PGM



Example prototypical PGM (Bayesian network):

Can we write out the joint PDF as:

$$p(y, x_1, \dots, x_n) = p(y) \prod_i p(x_i | y)$$

Alternatively, we can write out the posterior PDF (prob of latents given observations) as:

Factorize

$$p(y | x_1, \dots, x_n) = \frac{p(y, x_1, \dots, x_n)}{p(x_1, \dots, x_n)} = \frac{1}{Z} p(y) \prod_i p(x_i | y)$$

# The Evidence Lower Bound

Given a probability model such as this, in the form  $p_{\theta}(x) = \frac{1}{Z(\theta)} \prod_k \phi_k(x_k; \theta)$ ,

# The Evidence Lower Bound

Given a probability model such as this, in the form  $p_{\theta}(x) = \frac{1}{Z(\theta)} \prod_k \phi_k(x_k; \theta)$ ,

It is difficult to optimize  $\text{KL}[q \parallel p]$  directly!

For a few reasons:

# The Evidence Lower Bound

Given a probability model such as this, in the form  $p_{\theta}(x) = \frac{1}{Z(\theta)} \prod_k \phi_k(x_k; \theta)$ ,

It is difficult to optimize  $\text{KL}[q \parallel p]$  directly!

For a few reasons:

- Potentially intractable normalization constant.
- $\Rightarrow$  difficult to take gradients due to this.
- In fact, it's difficult even to evaluate  $\text{KL}[q \parallel p]$  (or  $p(x)$  for that matter).

# The Evidence Lower Bound

Therefore, we will not work with the KL divergence directly.

# The Evidence Lower Bound

Therefore, we will not work with the KL divergence directly.

Instead, we will use the following objective, which has a similar form as the KL divergence, but only involves the *unnormalized probability*  $\tilde{p}(x) = \prod_k \phi_k(x_k; \theta)$ .

# The Evidence Lower Bound

Therefore, we will not work with the KL divergence directly.

Instead, we will use the following objective, which has a similar form as the KL divergence, but only involves the *unnormalized probability*  $\tilde{p}(x) = \prod_k \phi_k(x_k; \theta)$ .

For example, if  $p(x)$  is the posterior PDF (of latents given observed),  
then an example of  $\tilde{p}(x)$  is the joint PDF.

# The Evidence Lower Bound

Therefore, we will not work with the KL divergence directly.

Instead, we will use the following objective, which has a similar form as the KL divergence, but only involves the *unnormalized probability*  $\tilde{p}(x) = \prod_k \phi_k(x_k; \theta)$ .

This objective is:

# The Evidence Lower Bound

Therefore, we will not work with the KL divergence directly.

Instead, we will use the following objective, which has a similar form as the KL divergence, but only involves the *unnormalized probability*  $\tilde{p}(x) = \prod_k \phi_k(x_k; \theta)$ .

This objective is:

$$J(q) = \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)} = \mathbb{E}_{q(x)} \left[ \log \frac{q(x)}{\tilde{p}(x)} \right]$$

For discrete distributions

# The Evidence Lower Bound

Therefore, we will not work with the KL divergence directly.

Instead, we will use the following objective, which has a similar form as the KL divergence, but only involves the *unnormalized probability*  $\tilde{p}(x) = \prod_k \phi_k(x_k; \theta)$ .

This objective is:

$$J(q) = \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)} = \mathbb{E}_{q(x)} \left[ \log \frac{q(x)}{\tilde{p}(x)} \right]$$

For discrete distributions

$$J(q) = \int_{\mathcal{X}} q(x) \log \frac{q(x)}{\tilde{p}(x)} dx = \mathbb{E}_{q(x)} \left[ \log \frac{q(x)}{\tilde{p}(x)} \right]$$

For continuous distributions

# The Evidence Lower Bound

This function is *tractable* (we can evaluate it), and it has the following property:

# The Evidence Lower Bound

This function is *tractable* (we can evaluate it), and it has the following property:

$$\begin{aligned} J(q) &= \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)} \\ &= \sum_x q(x) \log \frac{q(x)}{p(x)} - \log Z(\theta) \\ &= \text{KL}[q(x) \parallel p(x)] - \log Z(\theta) \end{aligned}$$

Assuming a discrete distribution WLOG

# The Evidence Lower Bound

This function is *tractable* (we can evaluate it), and it has the following property:

$$J(q) = \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)}$$

Definition of  $J(q)$

Assuming a discrete distribution WLOG

$$\begin{aligned} &= \sum_x q(x) \log \frac{q(x)}{p(x)} - \log Z(\theta) \\ &= \text{KL}[q(x) \parallel p(x)] - \log Z(\theta) \end{aligned}$$

# The Evidence Lower Bound

This function is *tractable* (we can evaluate it), and it has the following property:

$$J(q) = \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)}$$

Definition of  $J(q)$

Assuming a discrete distribution WLOG

$$= \sum_x q(x) \log \frac{q(x)}{p(x)} - \log Z(\theta)$$

Definition of  $p(x)$

$$= \text{KL} [q(x) \parallel p(x)] - \log Z(\theta)$$

# The Evidence Lower Bound

This function is *tractable* (we can evaluate it), and it has the following property:

$$J(q) = \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)}$$

Definition of  $J(q)$

Assuming a discrete distribution WLOG

$$= \sum_x q(x) \log \frac{q(x)}{p(x)} - \log Z(\theta)$$

Definition of  $p(x)$

$$= \text{KL} [q(x) \parallel p(x)] - \log Z(\theta)$$

Definition of  $\text{KL}$

# The Evidence Lower Bound

This function is *tractable* (we can evaluate it), and it has the following property:

$$J(q) = \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)}$$

Definition of  $J(q)$

Assuming a discrete distribution WLOG

$$= \sum_x q(x) \log \frac{q(x)}{p(x)} - \log Z(\theta)$$

Definition of  $p(x)$

$$= \text{KL} [q(x) \parallel p(x)] - \log Z(\theta)$$

Definition of  $\text{KL}$

$\Rightarrow J(q)$  is equal to the KL divergence minus the *log partition function* (aka *log normalizer*).  
(aka “marginal log-likelihood”).

# The Evidence Lower Bound

Also note that KL divergence is non-negative:  $\text{KL} [q \parallel p] \geq 0$

Which implies the following relation:

# The Evidence Lower Bound

Also note that KL divergence is non-negative:  $\text{KL} [q \parallel p] \geq 0$

Which implies the following relation:

$$\begin{aligned}\log Z(\theta) &= \text{KL} [q(x) \parallel p(x)] - J(q) \geq -J(q) \\ \Rightarrow \log Z(\theta) &\geq -J(q)\end{aligned}$$

# The Evidence Lower Bound

Also note that KL divergence is non-negative:  $\text{KL} [q \parallel p] \geq 0$

Which implies the following relation:

$$\begin{aligned}\log Z(\theta) &= \text{KL} [q(x) \parallel p(x)] - J(q) \geq -J(q) \\ \Rightarrow \log Z(\theta) &\geq -J(q)\end{aligned}$$

Therefore,  $-J(q)$  is a *lower bound* on the log-partition function,  $\log Z(\theta)$ .



# The Evidence Lower Bound

Recall that, for posterior distributions (of latents given observed), i.e.,

(Dropped  $\theta$  for convenience).

$$p(y \mid x_1, \dots, x_n) = \frac{p(y, x_1, \dots, x_n)}{p(x_1, \dots, x_n)} = \frac{1}{Z} p(y, x_1, \dots, x_n)$$

# The Evidence Lower Bound

Recall that, for posterior distributions (of latents given observed), i.e.,

(Dropped  $\theta$  for convenience).

$$p(y \mid x_1, \dots, x_n) = \frac{p(y, x_1, \dots, x_n)}{p(x_1, \dots, x_n)} = \frac{1}{Z} p(y, x_1, \dots, x_n)$$

the log-partition function is  $Z = p(x_1, \dots, x_n)$ , often called the **evidence**.

(or “marginal log-likelihood”)

# The Evidence Lower Bound

Recall that, for posterior distributions (of latents given observed), i.e.,

(Dropped  $\theta$  for convenience).

$$p(y | x_1, \dots, x_n) = \frac{p(y, x_1, \dots, x_n)}{p(x_1, \dots, x_n)} = \frac{1}{Z} p(y, x_1, \dots, x_n)$$

the log-partition function is  $Z = p(x_1, \dots, x_n)$ , often called the **evidence**.

(or “marginal log-likelihood”)

Thus,  $-J(q) \leq \log Z$  is often referred to as the **evidence lower bound**, or **ELBO**.

# The Evidence Lower Bound

To summarize, the ELBO is:

# The Evidence Lower Bound

To summarize, the ELBO is:

For discrete distributions

$$-J(q) = - \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)} = \sum_x q(x) \log \frac{\tilde{p}(x)}{q(x)} = \mathbb{E}_{q(x)} \left[ \log \frac{\tilde{p}(x)}{q(x)} \right]$$

# The Evidence Lower Bound

To summarize, the ELBO is:

For discrete distributions

$$-J(q) = - \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)} = \sum_x q(x) \log \frac{\tilde{p}(x)}{q(x)} = \mathbb{E}_{q(x)} \left[ \log \frac{\tilde{p}(x)}{q(x)} \right]$$

For continuous distributions

$$-J(q) = - \int_{\mathcal{X}} q(x) \log \frac{q(x)}{\tilde{p}(x)} dx = \int_{\mathcal{X}} q(x) \log \frac{\tilde{p}(x)}{q(x)} dx = \mathbb{E}_{q(x)} \left[ \log \frac{\tilde{p}(x)}{q(x)} \right]$$

# The Evidence Lower Bound

Also, the ELBO is often written in the form:

$$\begin{aligned} -J(q) = \mathbb{E}_{q(x)} [\log \tilde{p}(x) - \log q(x)] &= \log Z(\theta) - \text{KL} [q(x) \parallel p(x)] \\ &\leq \log Z(\theta) \end{aligned}$$

# The Evidence Lower Bound

Also, the ELBO is often written in the form:

$$\begin{aligned} -J(q) = \mathbb{E}_{q(x)} [\log \tilde{p}(x) - \log q(x)] &= \log Z(\theta) - \text{KL} [q(x) \parallel p(x)] \\ &\leq \log Z(\theta) \end{aligned}$$

Importantly, note that:

# The Evidence Lower Bound

Also, the ELBO is often written in the form:

$$\begin{aligned} -J(q) = \mathbb{E}_{q(x)} [\log \tilde{p}(x) - \log q(x)] &= \log Z(\theta) - \text{KL} [q(x) \parallel p(x)] \\ &\leq \log Z(\theta) \end{aligned}$$

Importantly, note that:

- The difference between  $\log Z(\theta)$  and  $-J(q)$  is  $\text{KL} [q \parallel p]$ .

# The Evidence Lower Bound

Also, the ELBO is often written in the form:

$$\begin{aligned} -J(q) = \mathbb{E}_{q(x)} [\log \tilde{p}(x) - \log q(x)] &= \log Z(\theta) - \text{KL} [q(x) \parallel p(x)] \\ &\leq \log Z(\theta) \end{aligned}$$

Importantly, note that:

- The difference between  $\log Z(\theta)$  and  $-J(q)$  is  $\text{KL} [q \parallel p]$ .
- Therefore, by maximizing the ELBO, we are *minimizing*  $\text{KL} [q \parallel p]...$

# The Evidence Lower Bound

Also, the ELBO is often written in the form:

$$\begin{aligned} -J(q) = \mathbb{E}_{q(x)} [\log \tilde{p}(x) - \log q(x)] &= \log Z(\theta) - \text{KL}[q(x) \parallel p(x)] \\ &\leq \log Z(\theta) \end{aligned}$$

Importantly, note that:

- The difference between  $\log Z(\theta)$  and  $-J(q)$  is  $\text{KL}[q \parallel p]$ .
- Therefore, by maximizing the ELBO, we are *minimizing*  $\text{KL}[q \parallel p]...$
- ... by “*squeezing*” it between  $-J(q)$  and  $\log Z(\theta)$ .

## A Few Details on KL Divergence

Recall that  $\text{KL} [q \parallel p] \neq \text{KL} [p \parallel q]$  (i.e., KL divergence is not symmetric).

Both are 0 when  $q=p$ , but assign different penalties when  $q \neq p$ .

So why did we choose one over the other, and how do they differ?

# A Few Details on KL Divergence

Perhaps the most important reason we typically use  $\text{KL}[q \parallel p]$  is because:

# A Few Details on KL Divergence

Perhaps the most important reason we typically use  $\text{KL} [q \parallel p]$  is because:

$$\text{KL} [q(x) \parallel p(x)] = \mathbb{E}_{q(x)} \left[ \log \frac{q(x)}{p(x)} \right]$$

i.e., an expectation with respect to  $q(x)$  ... rather than  $p(x)$ .

# A Few Details on KL Divergence

Perhaps the most important reason we typically use  $\text{KL} [q \parallel p]$  is because:

$$\text{KL} [q(x) \parallel p(x)] = \mathbb{E}_{q(x)} \left[ \log \frac{q(x)}{p(x)} \right]$$

i.e., an expectation with respect to  $q(x)$  ... rather than  $p(x)$ .

Useful because  $q(x)$  is a known/tractable distribution, e.g., possible to sample from.

While  $p(x)$  is unknown/intractable (e.g., normalization unknown).

# A Few Details on KL Divergence

However, choosing  $\text{KL}[q \parallel p]$  in VI affects the optimal distribution!

When the variational family does not contain the true distribution  $p(x)$ .

# A Few Details on KL Divergence

$$\text{KL} [q(x) \parallel p(x)] = \mathbb{E}_{q(x)} \left[ \log \frac{q(x)}{p(x)} \right]$$

However, choosing  $\text{KL} [q \parallel p]$  in VI affects the optimal distribution!

When the variational family does not contain the true distribution  $p(x)$ .

Note that  $\text{KL} [q \parallel p]$  grows unbounded when  $q(x) > 0$  and  $p(x) \rightarrow 0$ .

Therefore, in VI, if  $p(x) = 0$  it means we must have that  $q(x) = 0$ .

# A Few Details on KL Divergence

$$\text{KL} [q(x) \parallel p(x)] = \mathbb{E}_{q(x)} \left[ \log \frac{q(x)}{p(x)} \right]$$

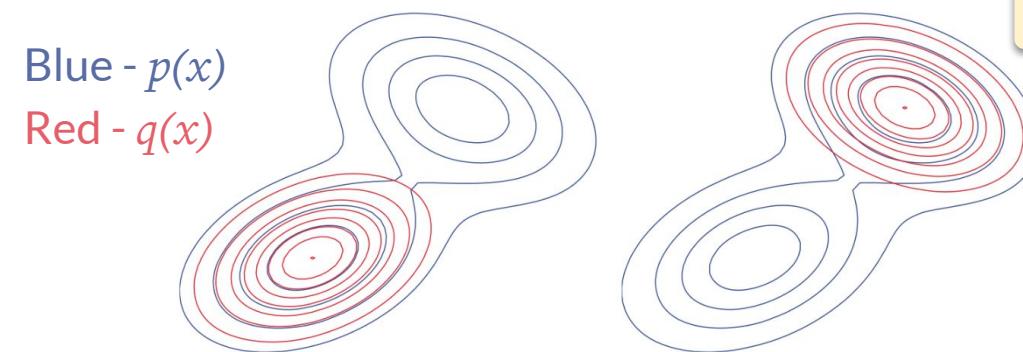
However, choosing  $\text{KL} [q \parallel p]$  in VI affects the optimal distribution!

When the variational family does not contain the true distribution  $p(x)$ .

Note that  $\text{KL} [q \parallel p]$  grows unbounded when  $q(x) > 0$  and  $p(x) \rightarrow 0$ .

Therefore, in VI, if  $p(x) = 0$  it means we must have that  $q(x) = 0$ .

We say that  $\text{KL} [q \parallel p]$  is “zero-forcing for  $q(x)$ ”



# A Few Details on KL Divergence

$$\text{KL} [q(x) \parallel p(x)] = \mathbb{E}_{q(x)} \left[ \log \frac{q(x)}{p(x)} \right]$$

On the other hand...

# A Few Details on KL Divergence

$$\text{KL} [p(x) \parallel q(x)] = \mathbb{E}_{p(x)} \left[ \log \frac{p(x)}{q(x)} \right]$$

On the other hand... consider  $\text{KL} [p \parallel q]$ .

Often called “M-projection”, or “moment projection”, or just the “reverse KL”.

# A Few Details on KL Divergence

$$\text{KL} [p(x) \parallel q(x)] = \mathbb{E}_{p(x)} \left[ \log \frac{p(x)}{q(x)} \right]$$

On the other hand... consider  $\text{KL} [p \parallel q]$ .

Often called “M-projection”, or “moment projection”, or just the “reverse KL”.

Note that  $\text{KL} [p \parallel q]$  grows unbounded when  $q(x) \rightarrow 0$  and  $p(x) > 0$ .

Therefore, in VI, if  $p(x) > 0$  it means we must have that  $q(x) > 0$ .

# A Few Details on KL Divergence

$$\text{KL} [p(x) \parallel q(x)] = \mathbb{E}_{p(x)} \left[ \log \frac{p(x)}{q(x)} \right]$$

On the other hand... consider  $\text{KL} [p \parallel q]$ .

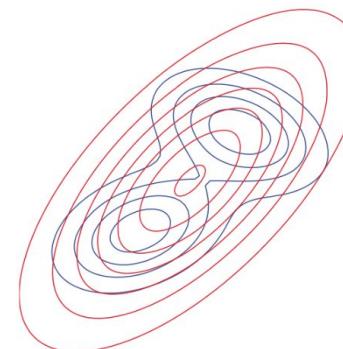
Often called “M-projection”, or “moment projection”, or just the “reverse KL”.

Note that  $\text{KL} [p \parallel q]$  grows unbounded when  $q(x) \rightarrow 0$  and  $p(x) > 0$ .

Therefore, in VI, if  $p(x) > 0$  it means we must have that  $q(x) > 0$ .

We say that  $\text{KL} [p \parallel q]$  is “zero-avoiding for  $q(x)$ ”

Blue -  $p(x)$   
Red -  $q(x)$



$q(x)$  typically overestimates the support of  $p(x)$ .

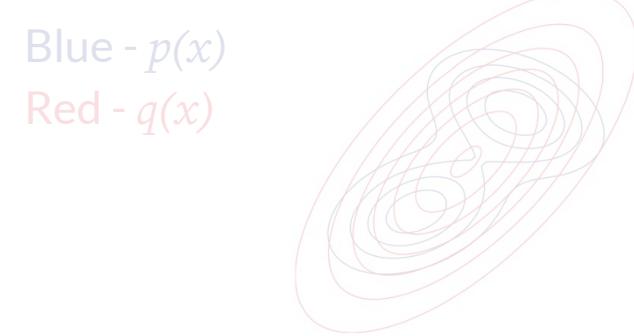
# A Few Details on KL Divergence

Due to these properties, these are also sometimes called:



$$\text{KL}[q \parallel p]$$

The “exclusive” KL divergence.

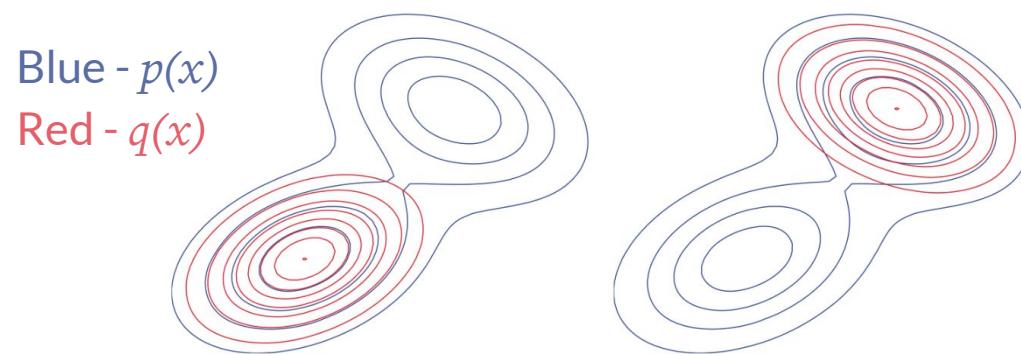


$$\text{KL}[p \parallel q]$$

The “inclusive” KL divergence.

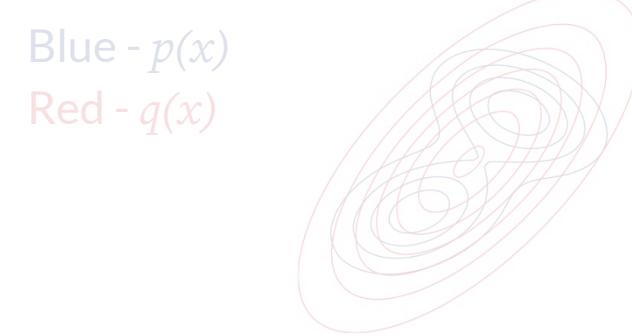
# A Few Details on KL Divergence

Due to these properties, these are also sometimes called:



$$\text{KL} [q \parallel p]$$

The “*exclusive*” KL divergence.

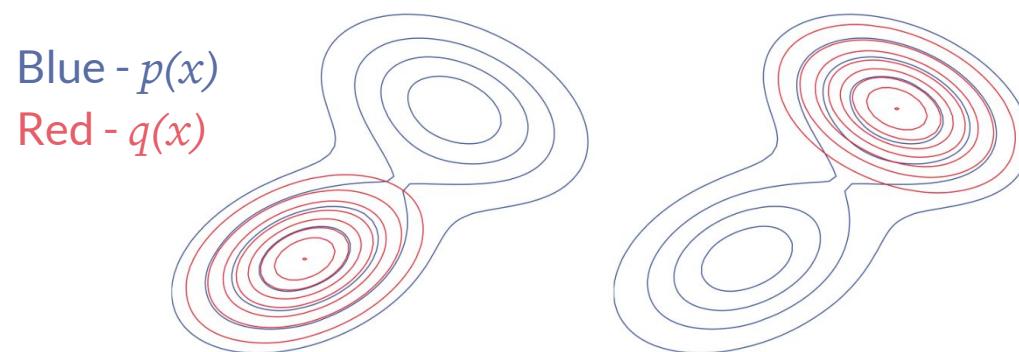


$$\text{KL} [p \parallel q]$$

The “*inclusive*” KL divergence.

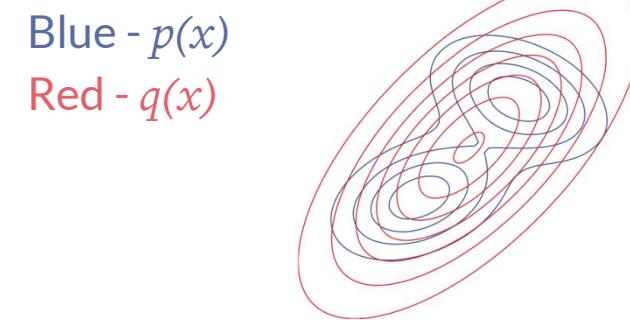
# A Few Details on KL Divergence

Due to these properties, these are also sometimes called:



$$\text{KL} [q \parallel p]$$

The “exclusive” KL divergence.



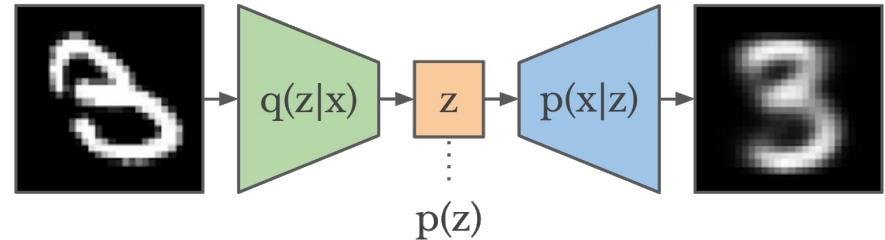
$$\text{KL} [p \parallel q]$$

The “inclusive” KL divergence.

# Next Class

Next class, we will cover:

- Gradient-based optimization via black-box variational inference.
- Neural network-based variational approximations  $q(x)$ .
- Also: amortized inference .
- Using this for Variational Autoencoders (VAE).
  - A classic deep generative model.



**Auto-Encoding Variational Bayes**

---

Diederik P. Kingma  
Machine Learning Group  
Universiteit van Amsterdam  
dipkingma@gmail.com

Max Welling  
Machine Learning Group  
Universiteit van Amsterdam  
welling.max@gmail.com

---

**Abstract**

How can we perform efficient inference and learning in directed probabilistic models with continuous latent variables and/or intractable hidden intermediate distributions? The variational Bayesian (VB) approach uses an optimization of an approximation to the intractable posterior. Unfortunately, the common mean-field approach requires analytical solutions of expectations w.r.t. the approximate posterior, which are also intractable in the general case. We show that a reparameterization of the variational lower bound yields a simple differentiable unbiased estimator of the lower bound; this SGVB (Stochastic Gradient Variational Bayes) estimator can be used for efficient approximate posterior inference in any directed model with continuous latent variables and/or parameters, and is straightforward to optimize using stochastic gradient techniques. For the case of an i.i.d. dataset and continuous latent variables per datapoint, we propose the Auto-Encoding VB (AEVB) algorithm. In the AEVB algorithm we make inference and learning especially efficient by using the SGVB estimator to optimize a recognition model that allows us to perform very efficient approximate posterior inference using simple ancestral sampling, which in turn allows us to efficiently learn the model parameters, without the need of expensive iterative inference schemes (such as MCMC) per datapoint. The learned approximate posterior inference model can also be used for a host of tasks such as recognition, denoising, representation and visualization purposes. When a neural network is used for the recognition model, we arrive at the *variational auto-encoder*.  
**1 Introduction**  
How can we perform efficient approximate inference and learning with directed probabilistic models whose continuous latent variables and/or intermediate hidden intermediate distributions? The variational Bayesian (VB) approach uses an optimization of an approximation to the intractable posterior. Unfortunately, the common mean-field approach requires analytical solutions of expectations w.r.t. the approximate posterior, which are also intractable in the general case. We show that a reparameterization of the variational lower bound yields a simple differentiable unbiased estimator of the lower bound; this SGVB (Stochastic Gradient Variational Bayes) estimator can be used for efficient approximate posterior inference in any directed model with continuous latent variables and/or parameters, and is straightforward to optimize using stochastic gradient techniques. For the case of an i.i.d. dataset and continuous latent variables per datapoint, we propose the Auto-Encoding VB (AEVB) algorithm. In the AEVB algorithm we make inference and learning especially efficient by using the SGVB estimator to optimize a recognition model that allows us to perform very efficient approximate posterior inference using simple ancestral sampling, which in turn allows us to efficiently learn the model parameters, without the need of expensive iterative inference schemes (such as MCMC) per datapoint. The learned approximate posterior inference model can also be used for a host of tasks such as recognition, denoising, representation and visualization purposes. When a neural network is used for the recognition model, we arrive at the *variational auto-encoder*.  
**2 Method**  
The strategy in this section can be used to derive a lower bound estimator (a stochastic objective function) for a variety of directed graphical models with continuous latent variables. We will restrict ourselves here to the common case where we have an i.i.d. dataset with latent variables per datapoint, and where we like to perform maximum likelihood (ML) or maximum a posteriori (MAP) inference on the (global) parameters, and variational inference on the latent variables. It is, for example,





# Paper Presentations

# Paper Presentations – Goal

- During the semester, each student gives **one 20-minute presentation** on a paper relevant to probabilistic and generative models.
- Ideally from a modern machine learning conference (e.g., NeurIPS, ICML, ICLR, AAAI, etc)
  - Could be a “classic” (older) paper relevant to modern models, as well.
- This will accomplish a few things:
  - Gives the class a chance to see a broad set of interesting probabilistic/generative modeling papers.
  - Gives each student (more) experience with distilling key content from a paper & presenting it.

## Paper Presentations – Schedule

Starting at the **end of February** we will go through a few (~4) presentations per class (2nd half of class).

# Paper Presentations – Schedule

Starting at the **end of February** we will go through a few (~4) presentations per class (2nd half of class).

In the spreadsheet, which I will share, students will sign up for a time slot during the semester.

To make it fair, students who sign up for presentations on the first two dates will get slightly more-lenient grading (a point of extra credit on this assignment).

A	B	C	D	E	F	G	H
Date	Presentation ID	Presenter Name ( <a href="#">sign up!</a> )	Paper Title	Link/url to paper	Discussion Lead 1 ( <a href="#">sign up!</a> )	Discussion Lead 2 ( <a href="#">sign up!</a> )	Scribe ( <a href="#">sign up!</a> )
February 28							
	1						
	2						
	3						
	4						
March 7							
	5						

## In-Class Participation and Discussion

There is a grade for in-class participation/discussion (during student presentations).  
Students will sign up for one of three roles (again in the spreadsheet).  
Each student has to do each role once during semester.

# In-Class Participation and Discussion

There is a grade for in-class participation/discussion (during student presentations). Students will sign up for one of three roles (again in the spreadsheet). Each student has to do each role once during semester.

## **Roles 1 and 2 – Discussion Leads**

- Read the paper before the class; Responsible for formulating a short list (~5 questions) for discussion; bring up a couple of these during class; submit all after class.

## **Role 3 – Scribe**

- Read the paper before the class; take notes on paper during presentation; write up a ~1 page summary of the presentation.

