Lecture 7: Markov Chain Monte Carlo

4F13: Machine Learning

Zoubin Ghahramani and Carl Edward Rasmussen

Department of Engineering, University of Cambridge

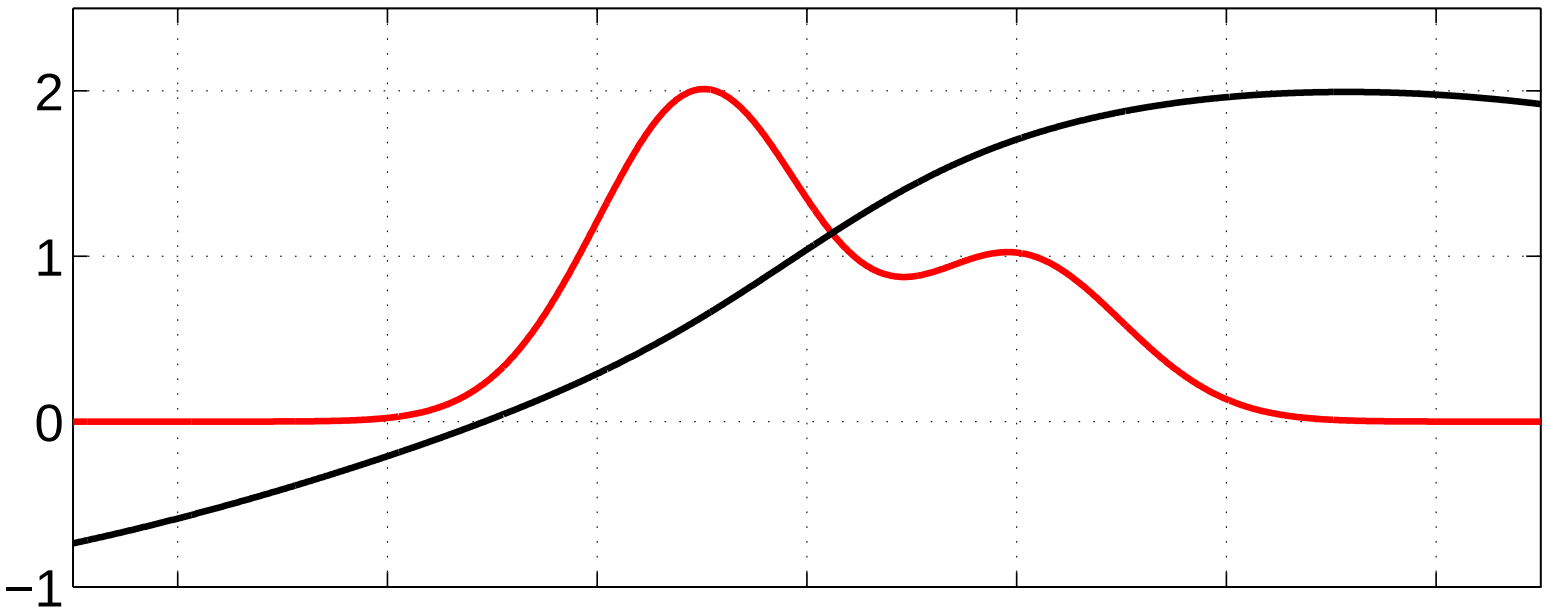
February 8th and 13th, 2008

Objective

Approximate expectations of a function $\phi(\mathbf{x})$ wrt probability $p(\mathbf{x})$:

$$\mathbb{E}_{p(x)}[\phi(x)] = \bar{\phi} = \int \phi(\mathbf{x}) p(\mathbf{x}) d\mathbf{x}, \quad \text{where} \quad \mathbf{x} \in \mathbb{R}^D,$$

when these are not analytically tractable, and typically $D \gg 1$.



Assume that we can evaluate $\phi(x)$ and $p(x)$, or at least $p^*(x)$, an un-normalized version of $p(x)$, for any value of x.

Motivation

Such integrals, or marginalizations, are essential for probabilistic inference and learning:

- Making predictions, e.g. in supervised learning

$$p(y_*|x_*, \mathcal{D}) = \int p(y_*|x_*, \theta) p(\theta|\mathcal{D}) d\theta,$$

  where the *posterior distribution* playes the rôle of $p(x)$
- Approximate marginal likelihoods

$$p(\mathcal{D}|\mathcal{M}_i) = \int p(\mathcal{D}|\theta, \mathcal{M}_i) p(\theta|\mathcal{M}_i) d\theta,$$

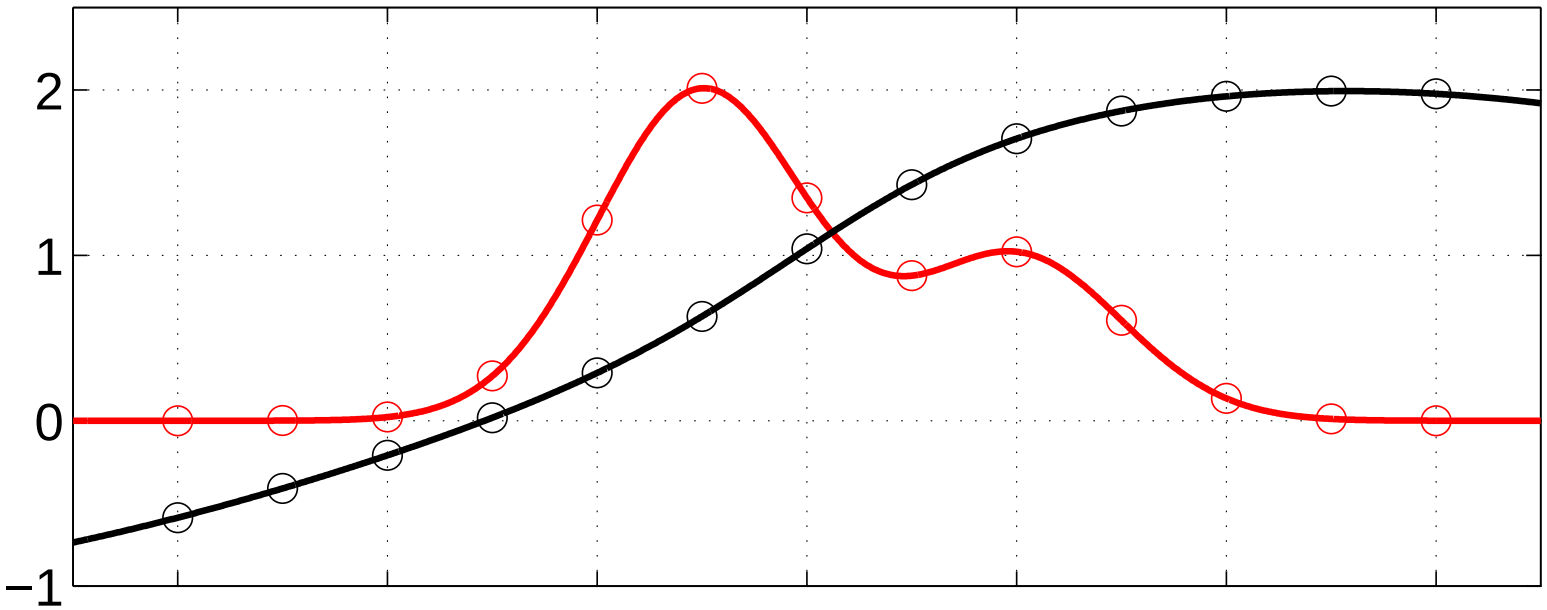
  where the *prior distribution* playes the rôle of $p(x)$.

Numerical integration on a grid

Approximate the integral by a sum of products

$$\int \phi(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \simeq \frac{1}{T} \sum_{\tau=1}^{T} \phi(\mathbf{x}^{(\tau)}) p(\mathbf{x}^{(\tau)}),$$

where the $\mathbf{x}^{(\tau)}$ lie on an equidistant grid (or fancier versions of this).



Problem: the number of grid points required, k^D, grows exponentially with the dimension D.

Monte Carlo

The foundation identity for Monte Carlo approximations is

$$\mathbb{E}[\phi(\mathbf{x})] \simeq \hat{\phi} = \frac{1}{T}\sum_{\tau=1}^{T}\phi(\mathbf{x}^{(\tau)}). \text{ where } \mathbf{x}^{(\tau)} \sim p(\mathbf{x}).$$

Under mild conditions, $\hat{\phi} \to \mathbb{E}[\phi(\mathbf{x})]$ as $T \to \infty$.

For moderate T, $\hat{\phi}$ may still be a good approximation. In fact it is an *unbiased* estimate with

$$\mathbb{V}[\hat{\phi}] = \frac{\mathbb{V}[\phi]}{T}, \text{ where } \mathbb{V}[\phi] = \int (\phi(\mathbf{x}) - \bar{\phi})^2 p(\mathbf{x}) d\mathbf{x}.$$

Note, that this variance in independent of the dimension D of $\mathbf{x}$.

This is great, but how do we generate random samples from $p(\mathbf{x})$?

Random number generation

There are (pseudo-) random number generators for the uniform $[0; 1]$ distribution.

Transformation methods exist to turn these into other simple univariate distributions, such as Gaussian, Gamma, . . . , as well as some multivariate distributions e.g. Gaussian and Dirichlet.

Example: $p(x)$ is uniform $[0; 1]$. The transformation $y = -\log(x)$ yields:

$$p(y) = p(x)\left|\frac{dx}{dy}\right| = \exp(-y),$$

i.e. an *exponential* distribution.

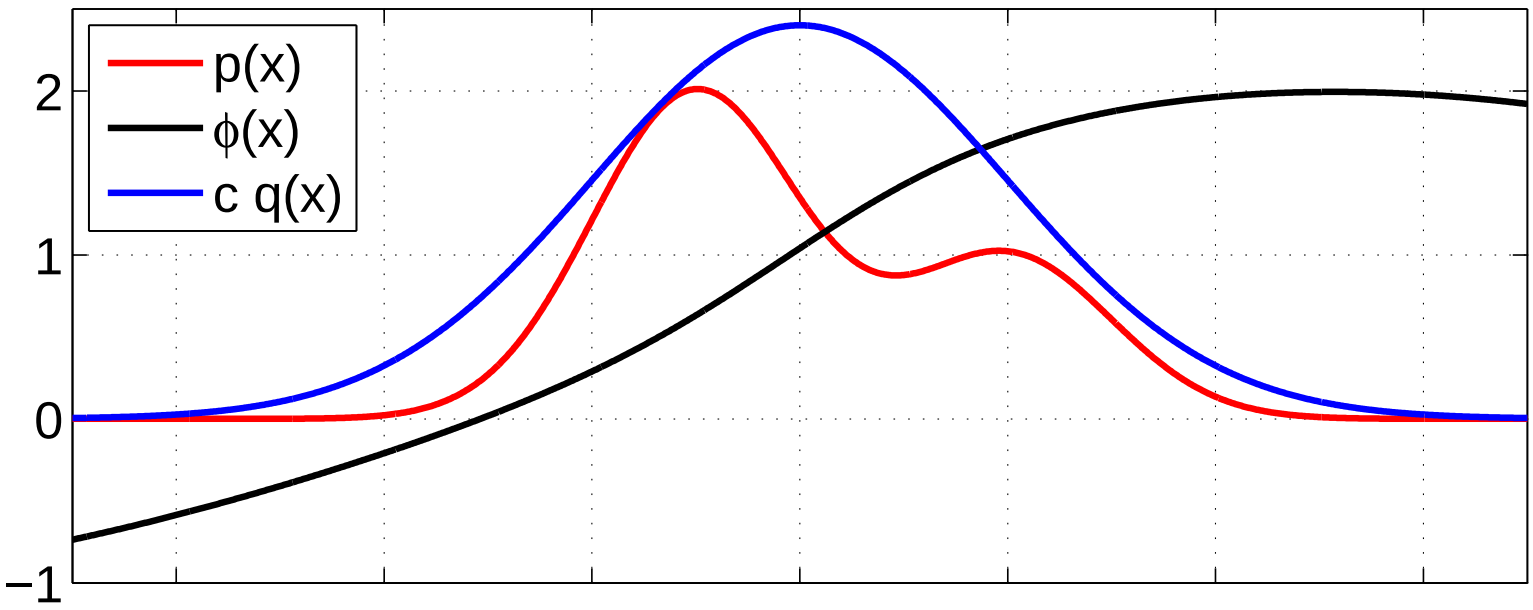
Typically $p(\mathbf{x})$ may be more complicated.

How can we generate samples from a (multivariate) distribution $p(\mathbf{x})$?

Idea: we can discretize $p(\mathbf{x})$, and sample from the discrete multinomial distribution. Will this work?

Rejection sampling

Find a tractable distribution $q(\mathbf{x})$ and $c \geqslant 1$, such that $\forall \mathbf{x}$, $cq(\mathbf{x}) \geqslant p(\mathbf{x})$.



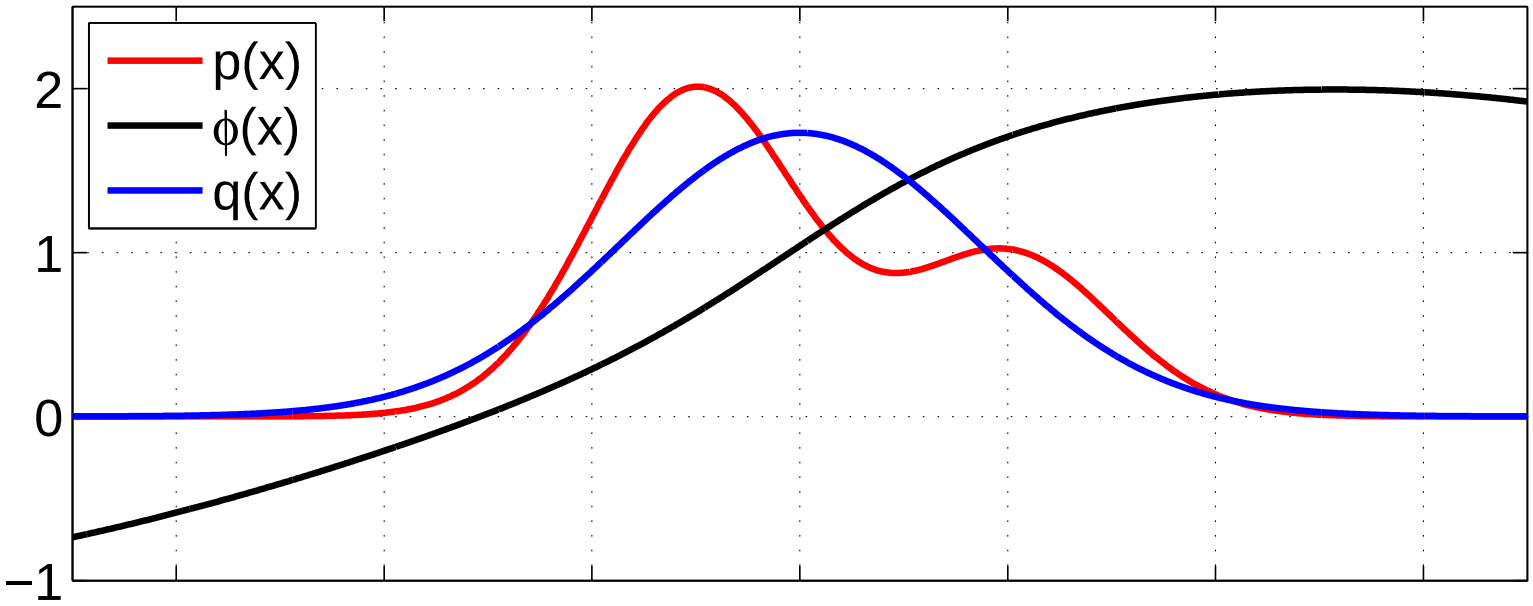
Rejection sampling algorithm:

- Generate samples independently from $q(\mathbf{x})$
- Accept samples with probability $p(\mathbf{x})/(cq(\mathbf{x}))$, otherwise reject
- Form a Monte Carlo estimate from the accepted samples.

This estimate with be exactly unbiased. How effective will it be?

Importance sampling

Find a tractable $q(\mathbf{x}) \simeq p(\mathbf{x})$, such that $q(\mathbf{x}) > 0$ whenever $p(\mathbf{x}) > 0$.



Form the Importance sampling estimate:

$$\int \phi(\mathbf{x}) \frac{p(\mathbf{x})}{q(\mathbf{x})} q(\mathbf{x}) d\mathbf{x} \simeq \hat{\phi} = \frac{1}{T} \sum_{\tau=1}^{T} \phi(\mathbf{x}^{(\tau)}) \underbrace{\frac{p(\mathbf{x}^{(\tau)})}{q(\mathbf{x}^{(\tau)})}}_{w(\mathbf{x}^{(\tau)})}, \quad \text{where } x^{(\tau)} \sim q(\mathbf{x}),$$

where $w(\mathbf{x}^{(\tau)})$ are called the *importance weights*.

There is also a version of importance sampling which doesn't require knowing how $q(\mathbf{x})$ normalizes:

$$\hat{\phi} = \frac{\sum_\tau \phi(\mathbf{x}^{(\tau)}) w(\mathbf{x}^{(\tau)})}{\sum_\tau w(\mathbf{x}^{(\tau)})}, \quad \text{where} \quad w(\mathbf{x}^{(\tau)}) = \frac{p(x^{(\tau)})}{q^*(\mathbf{x}^{(\tau)})}, \quad \text{and} \quad \mathbf{x}^{(\tau)} \sim q(\mathbf{x}).$$

How fast is importance sampling? This depends on the variance of the importance weights.

Example: $p(x) = \mathcal{N}(0, 1)$ and $q(x) = \mathcal{N}(0, \sigma^2)$. The variance of the weights is:

$$\begin{aligned}\mathbb{V}[w] = \mathbb{E}[w^2] - \mathbb{E}^2[w] &= \int \frac{p(x)^2}{q(x)^2} q(x) dx - \left(\int \frac{p(x)}{q(x)} q(x) dx \right)^2 \\ &= \sigma \int \exp(-x^2 + \frac{x^2}{2\sigma^2}) dx - 1.\end{aligned}$$

which only exists when $\sigma > \sqrt{1/2}$. Morale: always chose $q(\mathbf{x})$ to be wider, have 'heavier tails' than $p(\mathbf{x})$ to avoid this problem.

Rejection and Importance sampling

In the multivariate case with Gaussian $q(\mathbf{x})$ we must be able to guarantee that no direction exist when we underestimate the width by more than a factor of $\sqrt{2}$.

In fact, both rejection and importance sampling have severe problems in high dimensions, since it is virtually impossible to get $q(\mathbf{x})$ close enough to $p(\mathbf{x})$, see MacKay chapter 29 for details.

Independent Sampling vs. Markov Chains

So far, we've considered two methods, Rejection and Importance Sampling, which were both based on independent samples from $q(\mathbf{x})$.

However, for many problems of practical interest it is difficult or impossible to find $q(\mathbf{x})$ with the necessary properties.

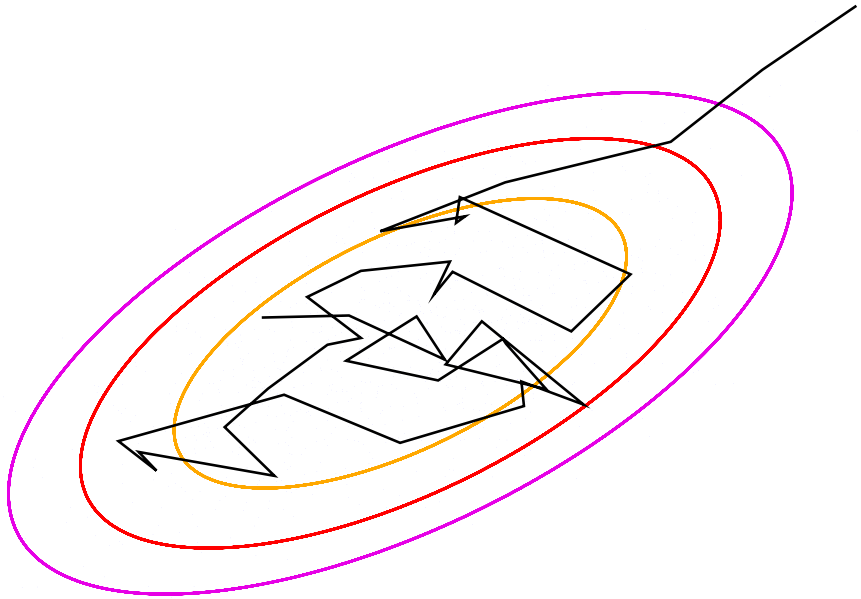
Instead we can abandon the idea of independent sampling, and instead rely on a Markov Chain to generate dependent samples from the target distribution.

Independence would be a nice thing, but it is not necessary for the Monte Carlo estimate to be valid.

Markov Chain Monte Carlo

We want to construct a Markov Chain that explores $p(\mathbf{x})$.

Markov Chain: $\mathbf{x}^{(t)} \sim q(\mathbf{x}^{(t)}|\mathbf{x}^{(t-1)})$.



MCMC gives approximate, correlated samples from $p(x)$.

Challenge: how do we find transition probabilities $q(\mathbf{x}^{(t)}|\mathbf{x}^{(t-1)})$, which give rise to the correct stationary distribution $p(\mathbf{x})$?

Discrete Markov Chains

Consider

$$\mathbf{p} = \begin{bmatrix} 3/5 \\ 1/5 \\ 1/5 \end{bmatrix}, \qquad Q = \begin{bmatrix} 2/3 & 1/2 & 1/2 \\ 1/6 & 0 & 1/2 \\ 1/6 & 1/2 & 0 \end{bmatrix}, \qquad Q_{ij} = Q(x_i \leftarrow x_j)$$

where Q is a stochastic (or transition) matrix.

To machine precision: $Q^{100} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \mathbf{p}$.

$\mathbf{p}$ is called a stationary distribution of Q, since $Q\mathbf{p} = \mathbf{p}$.

Ergodicity is also a requirement.

In Continuous Spaces

In continuous spaces transitions are governed by $q(\mathbf{x}'|\mathbf{x})$.

Now, $p(\mathbf{x})$ is a stationary distribution for $q(\mathbf{x}'|\mathbf{x})$ if

$$\int q(\mathbf{x}'|\mathbf{x})p(\mathbf{x})d\mathbf{x} \;=\; p(\mathbf{x}').$$

Detailed Balance

Detailed balance means

$$q(\mathbf{x}'|\mathbf{x})p(\mathbf{x}) \;=\; q(\mathbf{x}|\mathbf{x}')p(\mathbf{x}').$$

Now, integrating both sides wrt $\mathbf{x}$, we get

$$\int q(\mathbf{x}'|\mathbf{x})p(\mathbf{x})d\mathbf{x} \;=\; \int q(\mathbf{x}|\mathbf{x}')p(\mathbf{x}')d\mathbf{x} \;=\; p(\mathbf{x}').$$

Thus, detailed balance implies the existence of a stationary distribution

The Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm:

- propose a new state $\mathbf{x}^*$ from $q(\mathbf{x}^*|\mathbf{x}^{(\tau)})$
- compute the acceptance probability a

$$a = \frac{p(\mathbf{x}^*)}{p(\mathbf{x}^{(\tau)})} \frac{q(\mathbf{x}^{(\tau)}|\mathbf{x}^*)}{q(\mathbf{x}^*|\mathbf{x}^{(\tau)})}$$

- if $a > 1$ then the proposed state is accepted,
  otherwise the proposed state is accepted with probability a.
  If the proposed state is accepted, then $\mathbf{x}^{(\tau+1)} = \mathbf{x}^*$ otherwise $\mathbf{x}^{(\tau+1)} = \mathbf{x}^{(\tau)}$.

This Markov chain has $p(\mathbf{x})$ as a stationary distribution. This holds trivially if $\mathbf{x}^{(\tau+1)} = \mathbf{x}^{(\tau)}$, otherwise

$$\begin{aligned} p(\mathbf{x})Q(\mathbf{x}' \leftarrow \mathbf{x}) &= p(\mathbf{x})q(\mathbf{x}'|\mathbf{x}) \min\left(1, \frac{p(\mathbf{x}')q(\mathbf{x}|\mathbf{x}')}{p(\mathbf{x})q(\mathbf{x}'|\mathbf{x})}\right) \\ &= \min\left(p(\mathbf{x})q(\mathbf{x}'|\mathbf{x}), p(\mathbf{x}')q(\mathbf{x}|\mathbf{x}')\right) \\ &= p(\mathbf{x}')q(\mathbf{x}|\mathbf{x}') \min\left(1, \frac{p(\mathbf{x})q(\mathbf{x}'|\mathbf{x})}{p(\mathbf{x}')q(\mathbf{x}|\mathbf{x}')}\right) = p(\mathbf{x}')Q(\mathbf{x} \leftarrow \mathbf{x}'). \end{aligned}$$

Some properties of Metropolis Hastings

- The Metropolis algorithm has $p(\mathbf{x})$ as its stationary distribution
- If $q(\mathbf{x}^*|\mathbf{x}^{(\tau)})$ is symmetric, then
  - the expression for a simplifies to $a = p(\mathbf{x}^*)/p(\mathbf{x}^{(\tau)})$
  - the algorithm then always accepts if the proposed state has higher probability than the current state and sometimes accepts a state with lower probability.
- we only need the ratio of $p(\mathbf{x})$'s, so we don't need the normalization constant. This is important, e.g. when sampling from a posterior distribution.

The Metropolis algorithm can be widely applied, you just need to specify a proposal distribution.

The proposal distribution must satisfy some (mild) constraints (related to ergodicity).

The Proposal Distribution

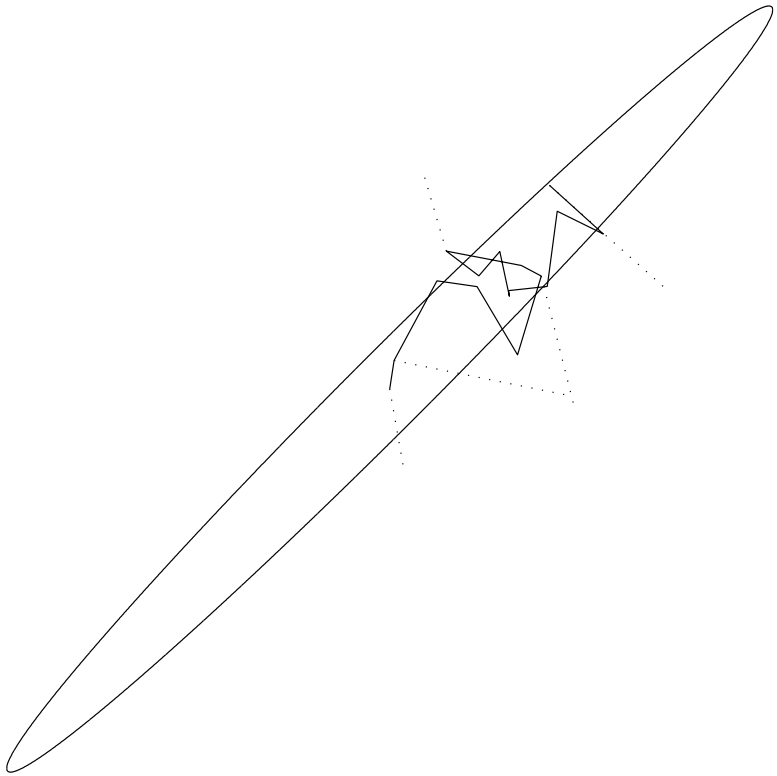
Often, Gaussian proposal distributions are used, centered on the current state. You need to specify the width of the proposal distribution.

What happens if the proposal distribution is

- too wide?
- too narrow?

Metropolis Hastings Example

20 iterations of the Metropolis Hastings algorithm for a bivariate Gaussian



The proposal distribution was Gaussian centered on the current state.

Rejected states are indicated by dotted lines.

Random walks

When exploring a distribution with the Metropolis algorithm, the proposal has to be narrow to avoid too many rejections.

If a typical step moves a distance ε, then one needs an expected $T = (L/\varepsilon)^2$ steps to travel a distance of L. This is a problem with random walk behavior.

Notice that strong correlations can slow down the Metropolis algorithm, because the proposal width should be set according to the tightest direction.

Variants of the Metropolis algorithm

Instead of proposing a new state by changing simultaneously all components of the state, you can concatenate different proposals changing one component at a time.

For example, $q_j(\mathbf{x}'|\mathbf{x})$, such that $x'_i = x_i$, $\forall i \neq j$.

Now, cycle through the $q_j(\mathbf{x}'|\mathbf{x})$ in turn.

This is valid as

- each iteration obeys detailed balance
- the *sequence* guarantees ergodicity

Gibbs sampling

Updating one coordinate at a time, and choosing the proposal distribution to be the conditional distribution of that variable given all other variables $q(x_i'|\mathbf{x}) = p(x_i'|x_{\neq i})$ we get

$$a = \min\left(1, \frac{p(x_{\neq i})p(x_i'|x_{\neq i})p(x_i|x_{\neq i}')}{p(x_{\neq i})p(x_i|x_{\neq i})p(x_i'|x_{\neq i})}\right) = 1,$$

i.e., we get an algorithm which always accepts. This is called the Gibbs sampling algorithm.

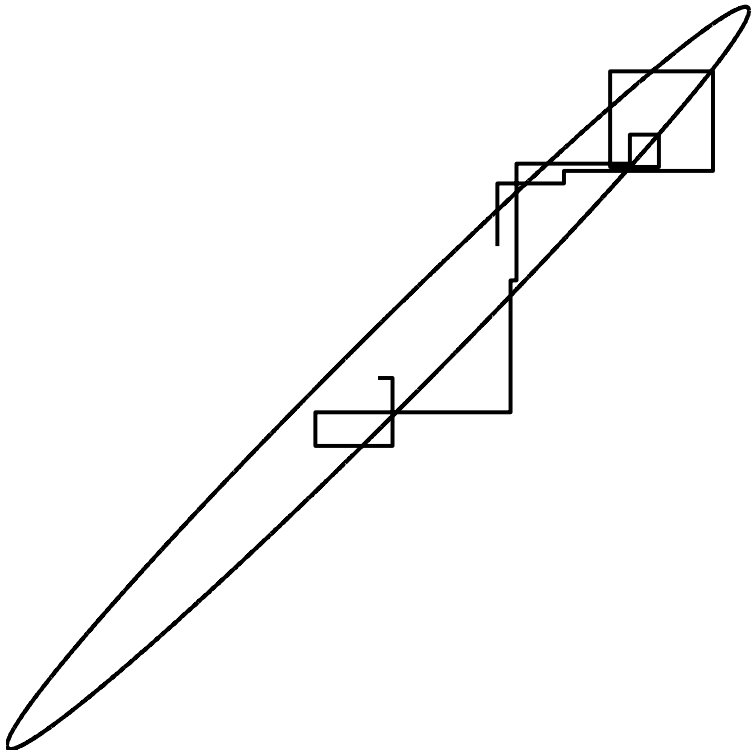
If you can compute (and sample from) the conditionals, you can apply Gibbs sampling.

This algorithm is completely parameter free.

Can also be applied to subsets of variables.

Example: Gibbs Sampling

20 iterations of Gibbs sampling on a bivariate Gaussian



Notice that strong correlations can slow down Gibbs sampling.

Hybrid Monte Carlo

Define a joint distribution over position and velocity

- $p(\mathbf{x}, \mathbf{v}) = p(\mathbf{v})p(\mathbf{x}) = e^{-E(\mathbf{x})-K(\mathbf{v})} = e^{-H(\mathbf{x},\mathbf{v})}$.
- $\mathbf{v}$ independent of $\mathbf{x}$ and $\mathbf{v} \sim \mathcal{N}(0, 1)$.
- sample in the joint space, then ignore the velocities to get the positions.

Markov chain:

- Gibbs sample velocities using $\mathcal{N}(0, 1)$
- Simulate Hamiltonian dynamics through fictitious time, then negate velocity
  - 'proposal' is deterministic and reversible
  - conservation of energy guarantees $p(\mathbf{x}, \mathbf{v}) = p(\mathbf{x}', \mathbf{v}')$
  - Metropolis acceptance probability is 1.

But we can not usually simulate Hamiltonian dynamics exactly.

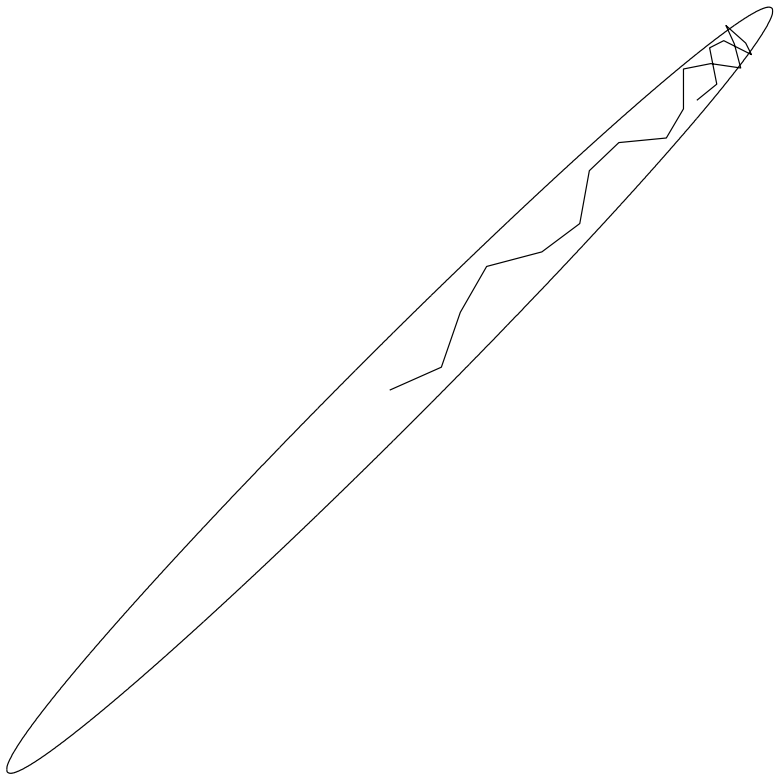
Leap-frog dynamics

a discrete approximation of Hamiltonian dynamics:

$$
\begin{aligned}
v_i(t + \tfrac{\varepsilon}{2}) &= v_i(t) - \frac{\varepsilon}{2} \frac{\partial E(\mathbf{x}(t))}{\partial x_i} \\
x_i(t + \varepsilon) & \; x_i(t) + \varepsilon v_i(t + \tfrac{\varepsilon}{2}) \\
v_i(t + \varepsilon) &= v_i(t + \tfrac{\varepsilon}{2}) - \frac{\varepsilon}{2} \frac{\partial E(\mathbf{x}(t + \varepsilon))}{\partial x_i}
\end{aligned}
$$

- using L steps of Leap-frog dynamics
- H is no-longer exactly conserved
- dynamics still deterministic and reversible
- acceptance probability $\min(1, \exp(H(\mathbf{v}, \mathbf{x}) - H(\mathbf{v}', \mathbf{x}')))$

Hybrid Monte Carlo avoids random walks



The advantage of introducing velocity variables is that random walks are avoided.

Summary

Rejection and Importance sampling are based on independent samples from $q(\mathbf{x})$.

These methods are not suitable for high dimensional problems.

The Metropolis method, does not give independent samples, but can be used successfully in high dimensions.

Gibbs sampling is used extensively in practice.

- parameter-free
- requires simple conditional distributions

Hybrid Monte Carlo is used extensively in continuous systems

- avoids random walks
- requires setting of ε and L.

Monte Carlo in Practice

Although very general, and can be applied in systems with 1000's of variables.

Care must be taken when using MCMC

- high dependency between variables may cause slow exploration
  - how long does 'burn-in' take?
  - how correlated are the samples generated?
  - slow exploration may be hard to detect – but you might not be getting the right answer
- diagnosing a convergence problem may be difficult
- curing a convergence problem may be even more difficult