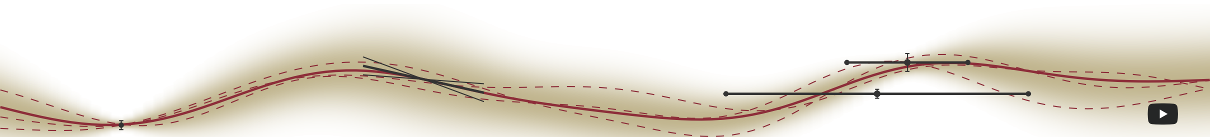
Probabilistic Inference and Learning
Lecture 05
Markov Chain Monte Carlo

Philipp Hennig

04 May 2020

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Faculty of Science
Department of Computer Science
Chair for the Methods of Machine Learning



#	date	content	Ex	#	date	content	Ex
1	20.04.	Introduction	1	14	09.06.	Logistic Regression	8
2	21.04.	Reasoning under Uncertainty		15	15.06.	Exponential Families	
3	27.04.	Continuous Variables	2	16	16.06.	Graphical Models	9
4	28.04.	Monte Carlo		17	22.06.	Factor Graphs	
5	04.05.	Markov Chain Monte Carlo	3	18	23.06.	The Sum-Product Algorithm	10
6	05.05.	Gaussian Distributions		19	29.06.	Example: Topic Models	
7	11.05.	Parametric Regression	4	20	30.06.	Mixture Models	11
8	12.05.	Understanding Deep Learning		21	06.07.	EM	
9	18.05.	Gaussian Processes	5	22	07.07.	Variational Inference	12
10	19.05.	An Example for GP Regression		23	13.07.	Example: Topic Models	
11	25.05.	Understanding Kernels	6	24	14.07.	Example: Inferring Topics	13
12	26.05.	Gauss-Markov Models		25	20.07.	Example: Kernel Topic Models	
13	08.06.	GP Classification	7	26	21.07.	Revision	

$$F := \int f(x)p(x)\,dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i) =: \hat{F} \qquad \text{if } x_i \sim p$$

$$\mathbb{E}_p(\hat{F}) = F \qquad \operatorname{var}_p(\hat{F}) = \frac{\operatorname{var}_p(f)}{N}$$

Recap from last lecture:

- Random numbers can be used to estimate integrals $\rightharpoonup$ **Monte Carlo** algorithms
- although the concept of randomness is fundamentally unsound, Monte Carlo algorithms *are* competitive in high dimensional problems (primarily because the advantages of the alternatives degrade rapidly with dimensionality)

But in High Dimensions, Sampling isn't Easy, Either!

Sampling is harder than global optimization

To produce exact samples:

- need to know cumulative density everywhere
- need to know regions of high density (not just local maxima!)
- a global description of the entire function

Practical Monte Carlo Methods aim to construct samples from

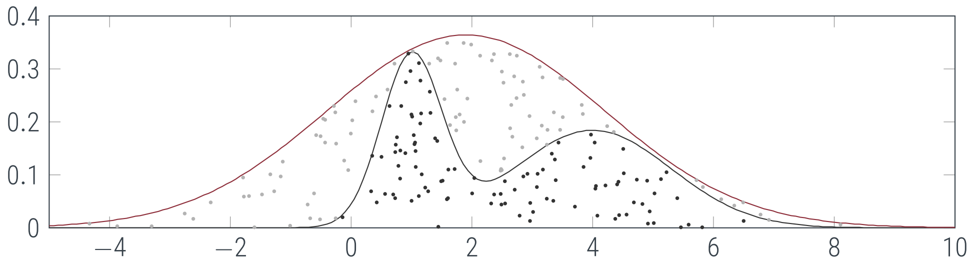
$$p(x) = \frac{\tilde{p}(x)}{Z}$$

assuming that it is possible to *evaluate* the *unnormalized* density $\tilde{p}$ (but not p) at arbitrary points.
Typical example: Compute moments of a posterior

$$p(x \mid D) = \frac{p(D \mid x)p(x)}{\int p(D, x)\, dx} \quad \text{as} \quad \mathbb{E}_{p(x|D)}(x^n) \approx \frac{1}{S}\sum_s x_i^n \quad \text{with } x_i \sim p(x \mid D)$$

Rejection Sampling

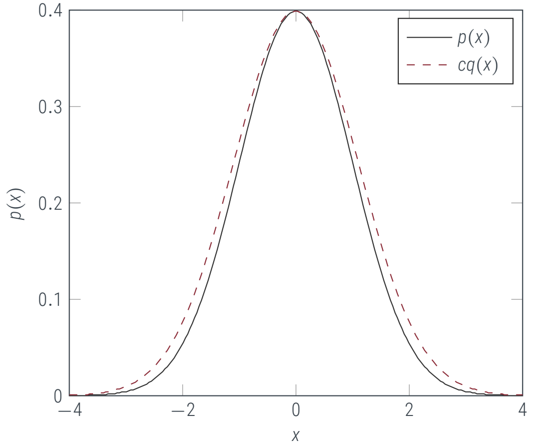
a simple method [Georges-Louis Leclerc, Comte de Buffon, 1707–1788]



- for any $p(x) = \tilde{p}(x)/Z$ (normalizer Z not required)
- choose $q(x)$ s.t. $cq(x) \geq \tilde{p}(x)$
- draw $s \sim q(x), u \sim \text{Uniform}[0, cq(s)]$
- reject if $u > \tilde{p}(s)$

The Problem with Rejection Sampling

the curse of dimensionality [MacKay, §29.3]



Example:

- $p(x) = \mathcal{N}(x; 0, \sigma_p^2)$
- $q(x) = \mathcal{N}(x; 0, \sigma_q^2)$
- $\sigma_q > \sigma_p$
- optimal c is given by

$$c = \frac{(2\pi\sigma_q^2)^{D/2}}{(2\pi\sigma_p^2)^{D/2}} = \left(\frac{\sigma_q}{\sigma_p}\right)^D = \exp\left(D \ln \frac{\sigma_q}{\sigma_p}\right)$$

- acceptance rate is ratio of volumes: $1/c$
- rejection rate rises **exponentially** in D
- for $\sigma_q/\sigma_p = 1.1, D = 100, 1/c < 10^{-4}$

Importance Sampling

a slightly less simple method

- computing $\tilde{p}(x), q(x)$, then throwing them away seems wasteful
- instead, rewrite (assume $q(x) > 0$ if $p(x) > 0$)

$$\phi = \int f(x)p(x)\,\mathrm{d}x = \int f(x)\frac{p(x)}{q(x)}q(x)\,\mathrm{d}x$$
$$\approx \frac{1}{S}\sum_s f(x_s)\frac{p(x_s)}{q(x_s)} =: \frac{1}{S}\sum_s f(x_s)w_s \qquad \text{if } x_s \sim q(x)$$

- this is just using a new function $g(x) = f(x)p(x)/q(x)$, so it is an unbiased estimator
- w_s is known as the **importance (weight)** of sample s
- if normalization unknown, can also use $\tilde{p}(x) = Zp(x)$

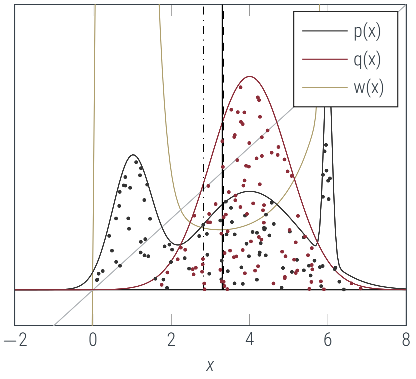
$$\int f(x)p(x) = \frac{1}{Z}\frac{1}{S}\sum_s f(x_s)\frac{\tilde{p}(x_s)}{q(x_s)}\,\mathrm{d}x$$
$$= \frac{1}{S}\sum_s f(x_s)\frac{\tilde{p}(x_s)/q(x_s)}{\frac{1}{S}\sum_{s'} 1\tilde{p}(x_s)/q(x_s)} =: \sum_s f(x_s)\tilde{w}_s$$

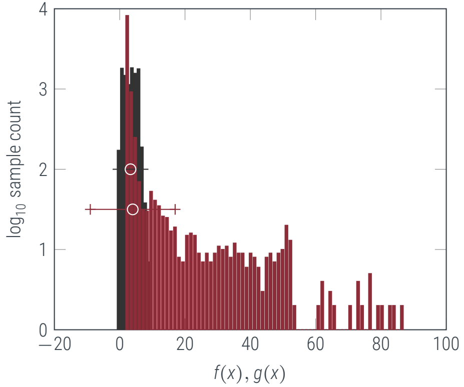
- this is consistent, but biased

What's wrong with Importance Sampling?

the curse of dimensionality, revisited

- recall that $\operatorname{var}\hat{\phi} = \operatorname{var}(f)/S$ – importance sampling replaces $\operatorname{var}(f)$ with $\operatorname{var}(g) = \operatorname{var}\left(f\frac{p}{q}\right)$
- $\operatorname{var}\left(f\frac{p}{q}\right)$ can be very large if $q \ll p$ somewhere. In many dimensions, usually all but everywhere!
- if p has "undiscovered islands", some samples have $p(x)/q(x) \to \infty$





Summary: Simple Practical Monte Carlo Methods

1. Producing exact samples is just as hard as high-dimensional integration. Thus, practical MC methods sample from a unnormalized density $\tilde{p}(x) = Z \cdot p(x)$
2. even this, however, is hard. Because it is hard to build a *globally* useful approximation to the integrand

Markov-Chain Monte Carlo

random walks drawing random numbers

- problem of importance sampling: samples generated independently, requires q good approximation to p everywhere.
- instead: generate samples **iteratively**, approximation q only needs to be good *locally*

Definition (Markov Chains)

A joint distribution $p(X)$ over a sequence of random variabels $X := [x_1, \dots, x_N]$ is said to have **the Markov property** if

$$p(x_i \mid x_1, x_2, \dots, x_{i-1}) = p(x_i \mid x_{i-1}).$$

The sequence is then called a **Markov chain**.

$$\begin{tikzcd} x_1 \arrow[r] & x_2 \arrow[r] & x_3 \arrow[r] & x_4 \arrow[r] & {} \end{tikzcd}$$

An analogy

optimization

assume we wanted to find the maximum of $\tilde{p}(x)$

- given current estimate x_t
- draw proposal $x' \sim q(x' \mid x_t)$
- evaluate

$$a = \frac{\tilde{p}(x')}{\tilde{p}(x_t)}$$

- if $a \geq 1$, accept: $x_{t+1} \leftarrow x'$
- else stay: $x_{t+1} \leftarrow x_t$

Usually, throw away estimates at the end, only keep "best guess". But the estimates do contain information about the shape of $\tilde{p}$!

The Metropolis-Hastings* Method

* Authorship controversial. Likely inventors: M. Rosenbluth, A. Rosenbluth & E. Teller, 1953

we want to find representers (**samples**) of $\tilde{p}(x)$

- given current sample x_t
- **draw proposal** $x' \sim q(x' \mid x_t)$ (for example, $q(x' \mid x_t) = \mathcal{N}(x'; x_t, \sigma^2)$)
- **evaluate**

$$a = \frac{\tilde{p}(x')}{\tilde{p}(x_t)} \frac{q(x_t \mid x')}{q(x' \mid x_t)}$$

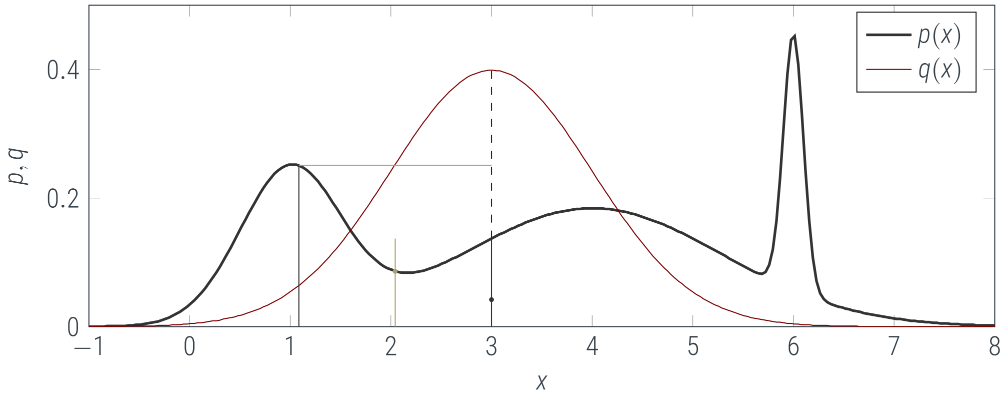
- if $a \geq 1$, **accept**: $x_{t+1} \leftarrow x'$
- else
  - **accept** with probability a: $x_{t+1} \leftarrow x'$
  - **stay** with probability $1 - a$: $x_{t+1} \leftarrow x_t$

Usually, assume symmetry $q(x_t \mid x') = q(x' \mid x_t)$ (the Metropolis method)

- no rejection. Every sample counts!
- like optimization, but with a chance to move downhill

Metropolis-Hastings in pictures

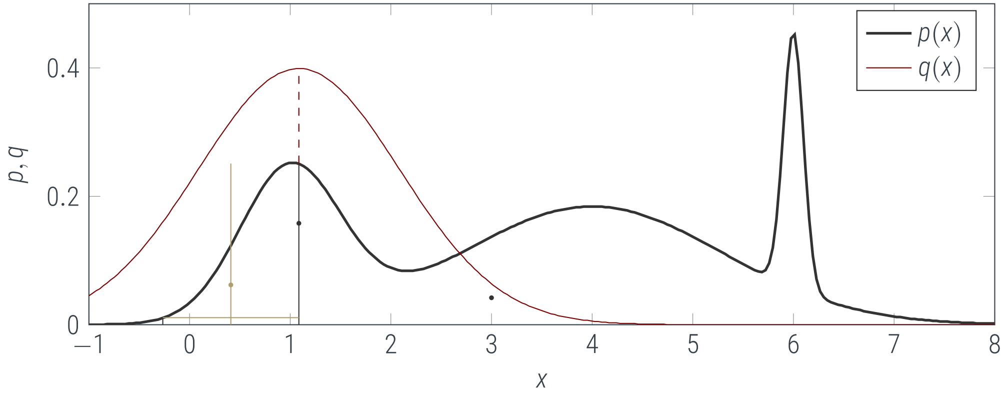
$t = 1$



$$a = \frac{\tilde{p}(x')}{\tilde{p}(x_t)} \frac{q(x_t \mid x')}{q(x' \mid x_t)} \qquad \text{accept with } p = \min(1, a)$$

Metropolis-Hastings in pictures

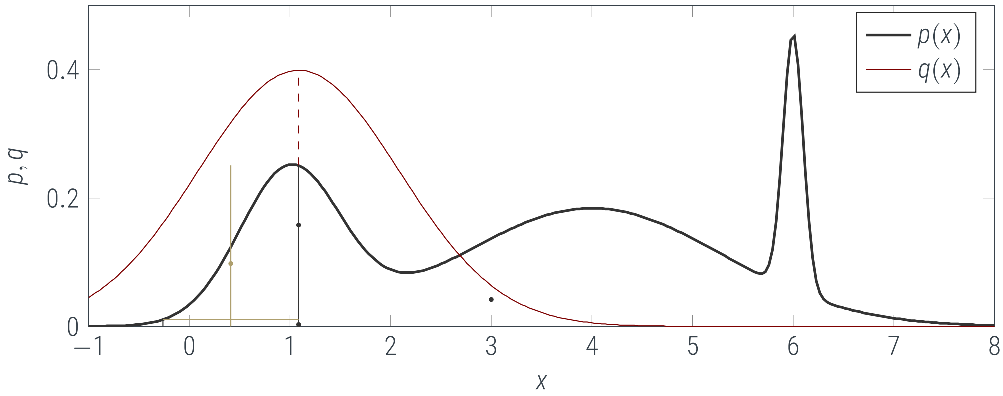
$t = 2$



$$a = \frac{\tilde{p}(x')}{\tilde{p}(x_t)} \frac{q(x_t \mid x')}{q(x' \mid x_t)} \qquad \text{accept with } p = \min(1, a)$$

Metropolis-Hastings in pictures

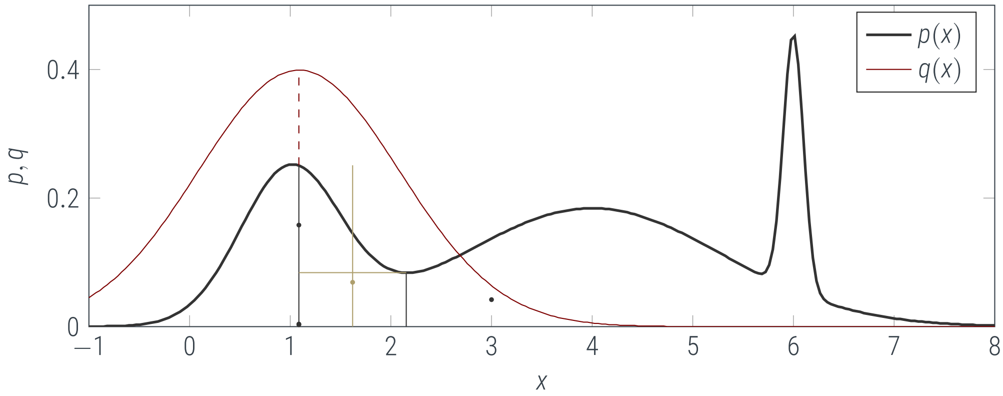
$t = 3$



$$a = \frac{\tilde{p}(x')}{\tilde{p}(x_t)} \frac{q(x_t \mid x')}{q(x' \mid x_t)} \qquad \text{accept with } p = \min(1, a)$$

Metropolis-Hastings in pictures

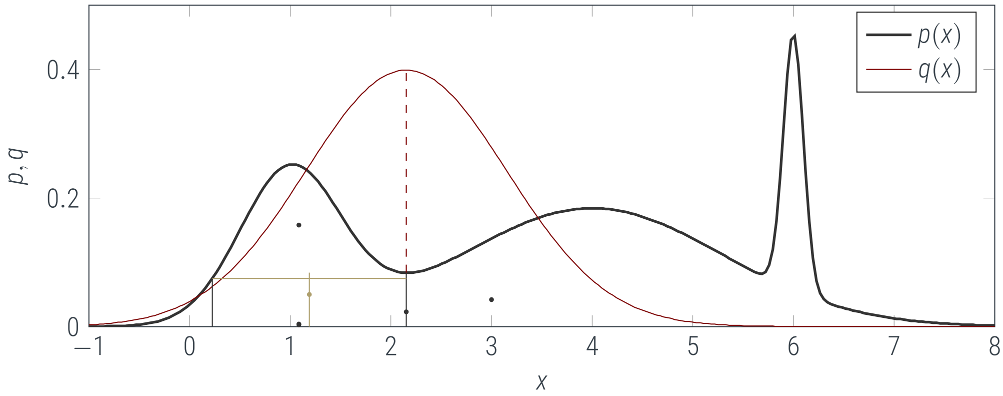
$t = 4$



$$a = \frac{\tilde{p}(x')}{\tilde{p}(x_t)} \frac{q(x_t \mid x')}{q(x' \mid x_t)} \qquad \text{accept with } p = \min(1, a)$$

Metropolis-Hastings in pictures

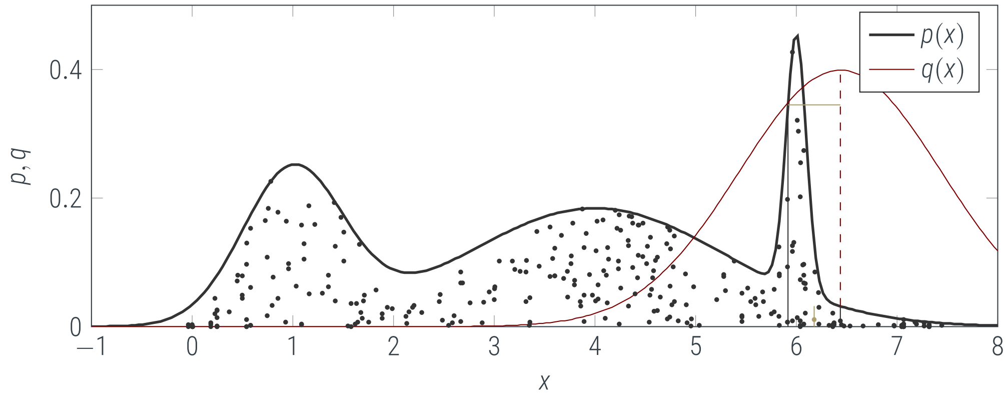
$t = 5$



$$a = \frac{\tilde{p}(x')}{\tilde{p}(x_t)} \frac{q(x_t \mid x')}{q(x' \mid x_t)} \qquad \text{accept with } p = \min(1, a)$$

Metropolis-Hastings in pictures

$t = 300$



$$a = \frac{\tilde{p}(x')}{\tilde{p}(x_t)} \frac{q(x_t \mid x')}{q(x' \mid x_t)} \qquad \text{accept with } p = \min(1, a)$$

Visualization

by Chi Feng

https://github.com/chi-feng

https://chi-feng.github.io/mcmc-demo/app.html#RandomWalkMH

Why is this a Monte Carlo Method?

MH draws from $p(x)$ in the limit of ∞ samples

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Theorem (convergence of Metropolis-Hastings, simplified)

If $q(x' \mid x_t) > 0 \; \forall (x', x_t)$, then, for any x_0, the distribution of x_t approaches $p(x)$ as $t \to \infty$.

proof (sketch) existence of stationary distribution: detailed balance

- MH satisfies detailed balance

$$\begin{aligned} p(x)T(x \to x') &= p(x) \cdot q(x' \mid x) \min\left[1, \frac{p(x')q(x \mid x')}{p(x)q(x' \mid x)}\right] \\ &= \min[p(x)q(x' \mid x), p(x')q(x \mid x')] \\ &= p(x') \cdot q(x \mid x') \min\left[\frac{p(x)q(x' \mid x)}{p(x')q(x \mid x')}, 1\right] \\ &= p(x')T(x' \to x) \end{aligned}$$

- Markov Chains satisfying detailed balance have **at least one *stationary distribution***

$$\int p(x)T(x \to x')\,dx = \int p(x')T(x' \to x)\,dx = p(x') \int T(x' \to x)\,dx = p(x')$$

Why is this a Monte Carlo Method?

MH draws from $p(x)$ in the limit of ∞ samples

proof (sketch) uniqueness of stationary distribution:

Definition (Ergodicity)

A sequence $\{x_t\}_{t\in\mathbb{N}}$ is called **ergodic** if it

1. is *a-periodic* (contains no recurring sequence)
2. has *positive recurrence*: $x_t = x_*$ implies there is a $t' > t$ such that $p(x_{t'} = x_*) > 0$

→ for MH, $\{x_t\}_{t\in\mathbb{N}}$ is ergodic (by definition)

- ergodic Markov Chains have **at most one stationary distribution**

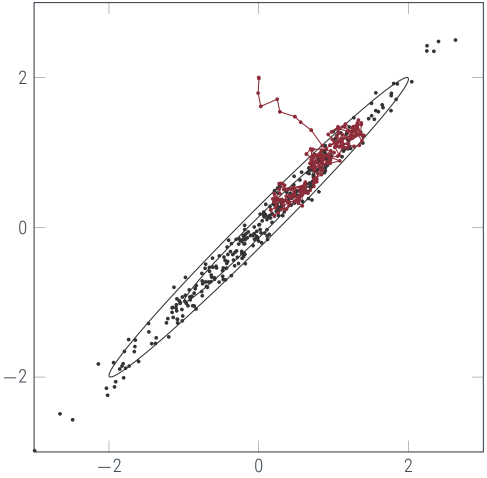
Theorem (convergence of Metropolis-Hastings, simplified)

If $q(x' \mid x_t) > 0\ \forall (x', x_t)$, then, for any x_0, the density of $\{x_t\}_{t\in\mathbb{N}}$ approaches $p(x)$ as $t \to \infty$.

- this is not a statement about convergence rate!

Metropolis-Hastings performs a (biased) random walk

hence diffuses $\mathcal{O}(s^{1/2})$



Rule of Thumb: [MacKay, (29.32)]

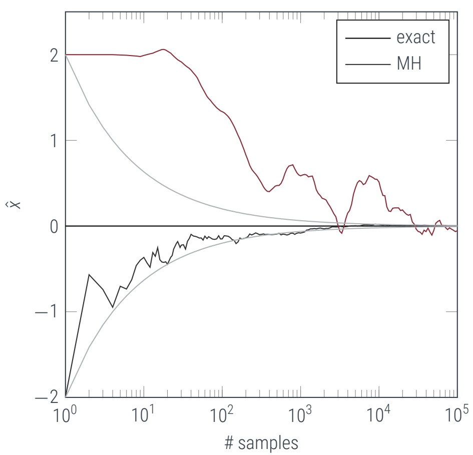
- typical use-case: high-dimensional D problem of largest length-scale L, smallest ε, isotropic proposal distribution
- have to set width of q to $\approx \varepsilon$, otherwise acceptance rate r will be very low.
- then Metropolis-Hastings does a **random walk** in D dimensions, moving a distance of $\sqrt{\mathbb{E}[\|x_t - x_0\|^2]} \sim \epsilon\sqrt{rt}$
- so, to create **one** independent draw at distance L, MCMC has to run for **at least**

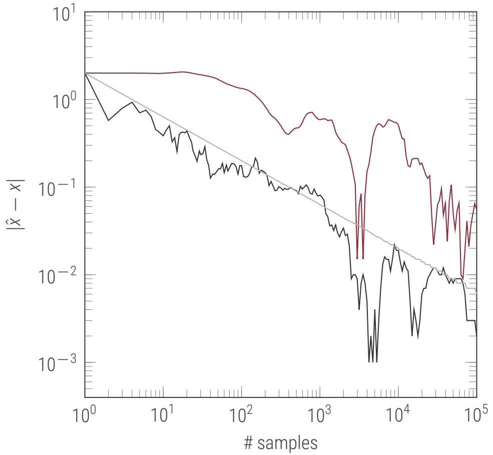
$$t \sim \frac{1}{r}\left(\frac{L}{\epsilon}\right)^2$$

iterations. In practice (e.g. if the distribution has *islands*), the situation can be **much** worse.

Metropolis-Hastings performs a (biased) random walk

estimating the mean of a correlated Gaussian





Summary: Practical Monte Carlo Methods

1. Producing exact samples is just as hard as high-dimensional integration. Thus, practical MC methods sample from a unnormalized density $\tilde{p}(x) = Z \cdot p(x)$
2. even this, however, is hard. Because it is hard to build a *globally* useful approximation to the integrand
3. **Markov Chain Monte Carlo** circumvents this problem by using *local* operations. It only converges well on the scale in which the local models cover the global problem. Thus the local behaviour has to be *tuned*.

Gibbs Sampling

Preparation for Exercise Sheet

[D. & S. Geman, 1984]

- $x_t \leftarrow x_{t-1}; x_{ti} \sim p(x_{ti} \mid x_{t1}, x_{t2}, \dots, x_{t(i-1)}, x_{t(i+1)}, \dots)$
- a special case of Metropolis-Hastings:
  - $q(x' \mid x_t) = \delta(x'_{\setminus i} - x_{t,\setminus i}) p(x'_i \mid x_{t,\setminus i})$
  - $p(x') = p(x'_i \mid x'_{\setminus i}) p(x'_{\setminus i}) = p(x'_i \mid x_{t,\setminus i}) p(x_{t,\setminus i})$
  - acceptance rate:

$$a = \frac{p(x')}{p(x_t)} \cdot \frac{q(x_t \mid x')}{q(x' \mid x_t)} = \frac{p(x'_i \mid x_{t,\setminus i}) p(x_{t,\setminus i})}{p(x_{ti} \mid x_{t,\setminus i}) p(x_{t,\setminus i})} \cdot \frac{q(x_t \mid x')}{\delta(x'_{\setminus i} - x_{t,\setminus i}) p(x'_i \mid x_{t,\setminus i})}$$

$$= \frac{q(x_t \mid x')}{p(x_{ti} \mid x_{t,\setminus i}) \delta(x'_{\setminus i} - x_{t,\setminus i})} = 1$$

Visualization

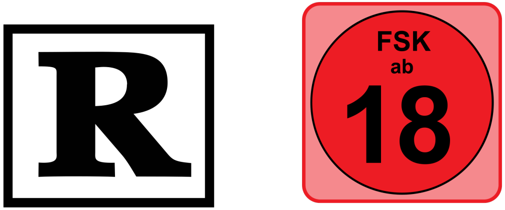
by Chi Feng

https://github.com/chi-feng

https://chi-feng.github.io/mcmc-demo/app.html#GibbsSampling,banana

Proceed with Confidence!

and don’t worry, it’ll be fine ...



- you don’t *need* to understand the following slides
- but a good engineer knows their tools

Hamiltonian Monte Carlo

reduce randomness by smoothing

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

[e.g. DJC MacKay, 2003, §30]

- consider **Boltzmann** distributions $P(x) = Z^{-1} \exp(-E(x))$
- augment the state-space by auxiliary **momentum** variables $p = \dot{x}$. Define **Hamiltonian** (“potential and kinetic energy”)

$$H(x,p) = E(x) + K(p) \qquad \text{with, e.g. } K(p) = \frac{1}{2} p^\intercal p$$

- do Metropolis-Hastings with p, x coupled by to **Hamiltonian dynamics**

$$\dot{x} := \frac{\partial x}{\partial t} = \frac{\partial H}{\partial p} \qquad \dot{p} := \frac{\partial p}{\partial t} = -\frac{\partial H}{\partial x}$$

nb: need to solve an ODE!

- note that, due to additive structure of Hamiltonian, this (asymptotically) samples from the **factorizing** joint

$$P_H(x,p) = \frac{1}{Z_H} \exp(-H(x,p)) = \frac{1}{Z_H} \exp(-E(x)) \cdot \exp(-K(p)) \qquad \text{with} \qquad P_H(x) = \int P_H(x,p)\, dp = P(x)$$

William R Hamilton
1805 – 1865
(Dublin)

Why does this improve things?

Hidden gems of Hamiltonian Monte Carlo

$$\dot{x} = \frac{\partial H}{\partial p} \qquad \dot{p} = -\frac{\partial H}{\partial x}$$

- If $p(x)$ is locally flat, then after N steps, x has changed by $x + Nhp$, so $\mathcal{O}(N)$, not $\mathcal{O}(\sqrt{N})$ as for Metropolis Hastings! **Hamiltonian MC mixes faster than Metropolis-Hastings**
- The Hamiltonian is a **conserved quantity**:

$$\frac{dH(p,x)}{dt} = \frac{\partial H}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial H}{\partial p}\frac{\partial p}{\partial t} = \frac{\partial H}{\partial x}\cdot\frac{\partial H}{\partial p} - \frac{\partial H}{\partial p}\cdot\frac{\partial H}{\partial x} = 0$$

So, if we have managed to simulate the dynamics well, then

$$\delta H = 0 \qquad \Rightarrow \qquad P_H(x',p') = P_H(x,p)$$

and the MH step will **always be accepted**!

$$a = \frac{\tilde{p}(x',\mathbf{p}')}{\tilde{p}(x_t,\mathbf{p}_t)}\frac{q(x_t,\mathbf{p}_t \mid x',\mathbf{p}')}{q(x',\mathbf{p}' \mid x_t,\mathbf{p}_t)} = \frac{exp(-H(x',\mathbf{p}'))}{\exp(-H(x_t,\mathbf{p}_t))}\frac{q(x_t,\mathbf{p}_t \mid x',\mathbf{p}')}{q(x',\mathbf{p}' \mid x_t,\mathbf{p}_t)}$$

HMC is a way to construct **really good** MH proposals that are always accepted (up to numerical errors).

Implementing Hamiltonian Monte Carlo ...

Heun's method for the Hamiltonian System

$$H(x,p) = E(x) + \frac{1}{2}p^\intercal p \qquad \dot{x} = \frac{\partial H}{\partial p} = p \qquad \dot{p} = -\frac{\partial H}{\partial x} = -\nabla_x E(x)$$

- We are trying to solve the **ordinary differential equation**

$$\frac{dz(t)}{dt} = f(z(t)) \quad \text{such that} \quad z(t_0) = z_0 \qquad z(t) = \begin{bmatrix} x(t) \\ p(t) \end{bmatrix}, \quad f\begin{pmatrix} x \\ p \end{pmatrix} = \begin{bmatrix} p(t) \\ -\nabla E(x(t)) \end{bmatrix}$$

- Heun's method:

$$z(t_i + h) = z_i + \frac{h}{2}(f(z_i) + f(z_i + hf(z_i)))$$

$$\begin{bmatrix} x_{i+1} \\ p_{i+1} \end{bmatrix} = \begin{bmatrix} x_{i+1} \\ p_{i+1} \end{bmatrix} + \frac{h}{2}\left(\begin{bmatrix} p_i \\ -\nabla E(x_i) \end{bmatrix} + f\left(\begin{bmatrix} x_i + hp_i \\ p_i - h\nabla E(x_i) \end{bmatrix}\right)\right)$$

$$= \begin{bmatrix} x_i + \frac{h}{2}(p_i + p_i - h\nabla E(x_i)) \\ p_i + \frac{h}{2}(-\nabla E(x_i) - \nabla E(x_i + hp_i)) \end{bmatrix} = \begin{bmatrix} x_i + hp_i - \frac{h^2}{2}\nabla E(x_i)) \\ p_i - \frac{h}{2}(\nabla E(x_i) + \nabla E(x_i + hp_i)) \end{bmatrix}$$

Hamiltonian Monte Carlo

moving with momentum

```python
import numpy as np; from numpy.random import randn, rand
def HamiltonianMC(findE,gradE,L,Tau,h,x0):
    x       = x0                             # initial sample
    X       = np.zeros([L,x.shape[0]])       # sample storage
    X[0,:]  = x                              # initialize storage
    E = findE(x); g = gradE(x)               # compute initial gradient and objective
    for l in range(L):                       # loop L times
        p = randn(x.shape[0])                # initial momentum is N(0,1)
        H = p.T @ p / 2 + E;                 # evaluate H(x,p)
        xnew = x; gnew = g                   # make temporary copy
        for tau in range(Tau):               # make Tau Heun steps
            p     = p    - h/2 * gnew        # make half-step in p
            xnew  = xnew + h * p             # make step in x
            gnew  = gradE(xnew)              # find new gradient
            p     = p    - h/2 * gnew        # make half-step in p
        Enew = findE(xnew)                   # find new value of H
        Hnew = p.T @ p / 2 + Enew
        dH   = Hnew - H                      # decide whether to accept
        if dH < 0 or rand() < np.exp(-dH): accept = 1
        else:                              accept = 0
        if accept: g = gnew; x = xnew; E = Enew
        X[l,:] = x
    return X
```

Visualization

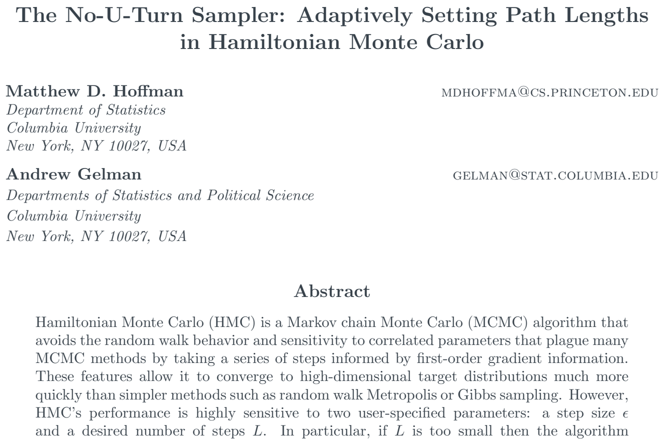
by Chi Feng `https://github.com/chi-feng`

`http://chi-feng.github.io/mcmc-demo/app.html#RandomWalkMH,banana`

How to set the Hyperparameters?

The state of the art in MCMC

[Hoffman & Gelman, JMLR **15** (2014), pp. 1593–1623]



The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo

Matthew D. Hoffman MDHOFFMA@CS.PRINCETON.EDU
Department of Statistics
Columbia University
New York, NY 10027, USA

Andrew Gelman GELMAN@STAT.COLUMBIA.EDU
Departments of Statistics and Political Science
Columbia University
New York, NY 10027, USA

Abstract

Hamiltonian Monte Carlo (HMC) is a Markov chain Monte Carlo (MCMC) algorithm that avoids the random walk behavior and sensitivity to correlated parameters that plague many MCMC methods by taking a series of steps informed by first-order gradient information. These features allow it to converge to high-dimensional target distributions much more quickly than simpler methods such as random walk Metropolis or Gibbs sampling. However, HMC's performance is highly sensitive to two user-specified parameters: a step size ϵ and a desired number of steps L. In particular, if L is too small then the algorithm

Visualization

by Chi Feng

https://github.com/chi-feng

https://chi-feng.github.io/mcmc-demo/app.html#NaiveNUTS,banana

Markov Chain Monte Carlo

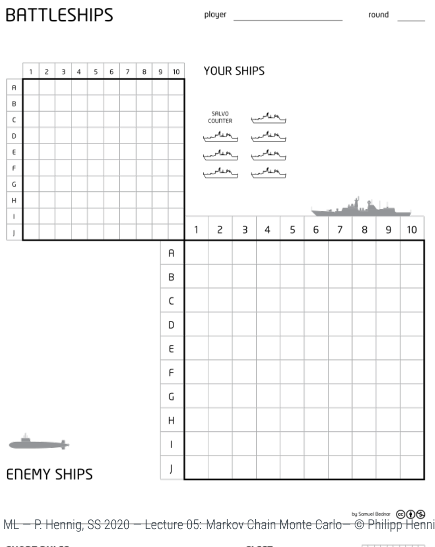
- breaks down sampling into **local dynamics**
- samples correctly in the **asymptotic limit**
- avoiding random walk behaviour (achieving good asymptotic mixing) requires **careful design**
- Hamiltonian MCMC methods (like NUTS) are currently *among* the state of the art (sequantial MC being an alternative).
  - they require the solution of an **ordinary differential equation** (the Hamiltonian dynamics)
  - their hyperparameters are tuned using elaborate subroutines
  - this is typical of all good numerical methods!
- these methods are available in software packages

Reminder: Monte Carlo methods converge stochastically. This stochastic rate is an **optimistic bound** for MCMC, because it has to be scaled by the mixing time. Monte Carlo methods are a powerful, well-developed tool. But they are most likely not the final solution to integration.

Despite centuries of research, integration remains an open problem.

Exercises

Computing with Probabilities, but without tools


BATTLESHIPS player ______ round ____

YOUR SHIPS

SALVO COUNTER

ENEMY SHIPS

by Samuel Bednar

- try to build an *agent* playing the game (with multiple ships)
- Things to think about:
  - how to deal with the combinatorial explosion
  - How is it best implemented *in practice* (in python)
  - how to build an autonomous agent?