Stochastic Processes

Week 10 (Version 1.0)

Sampling Methods

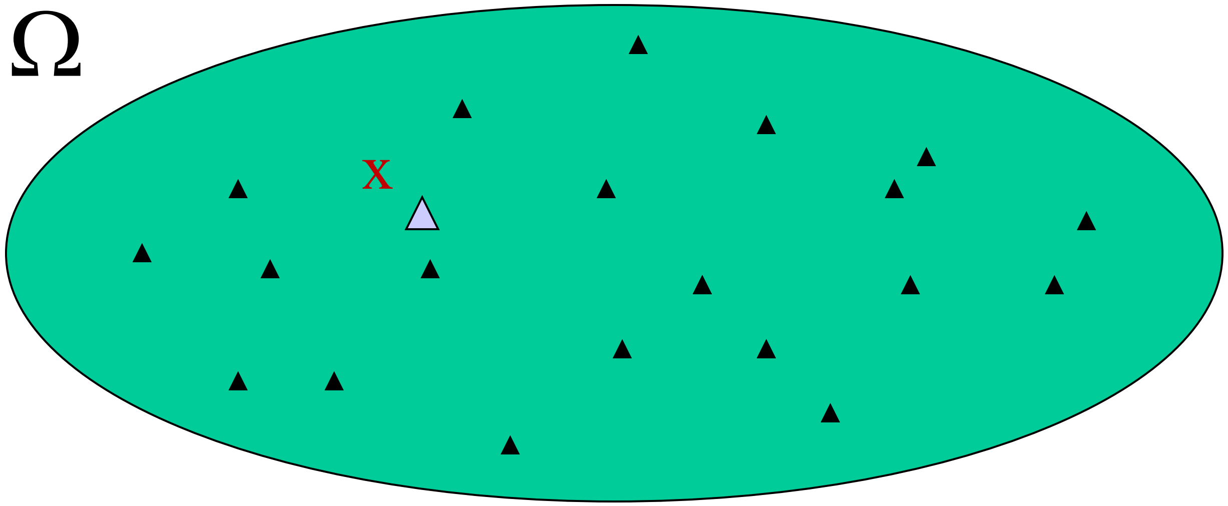
Hamid R. Rabiee

Fall 2025

Overview

- Random Sampling
- Monte Carlo Principle
- Monte Carlo Markov Chain
- Metropolis Hasting
- Gibbs Sampling
- Monte Carlo & Nonparametric Bayesian models

Random Sampling



- Ω - "very large" sample set.
- π - probability distribution over Ω.

<u>Goal:</u> Sample points $x \in \Omega$ at random from distribution π.

The Probability Distribution

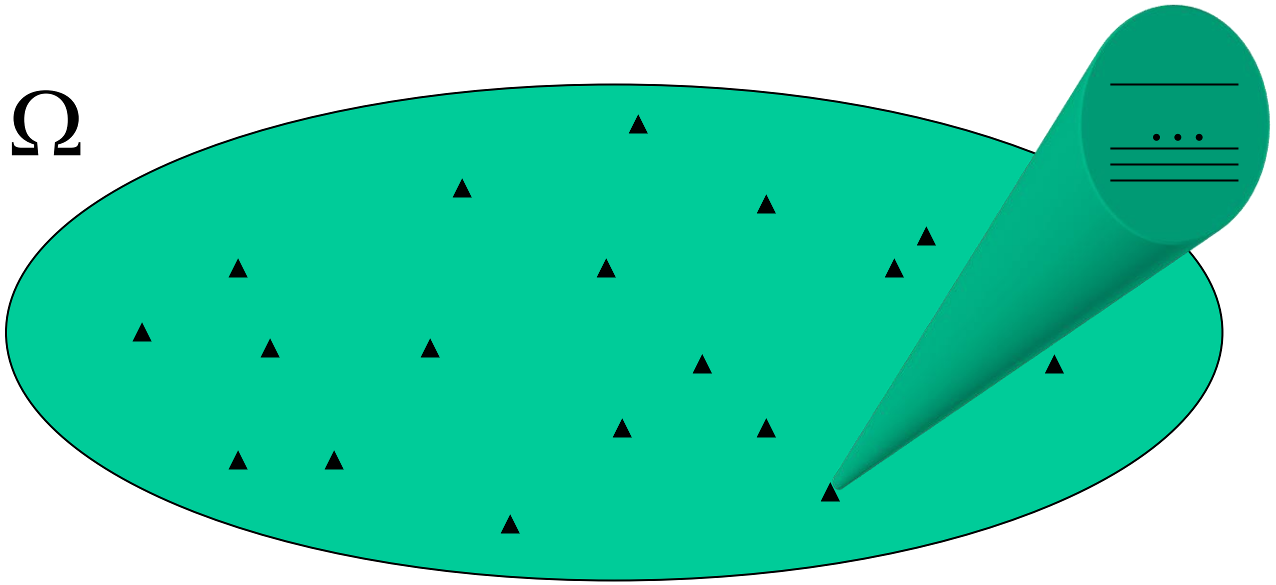
Typically,

$w:\Omega\rightarrow R^+$ is an easily-computed weight function

$$\pi(x) = \frac{w(x)}{Z}$$

$Z=\Sigma_x w(x)$ is an <u>unknown</u> normalization factor

Example: Permutation of N Distinct Object



- Ω - all $N!$ permutations of N distinct objects.
- π - uniform distribution [$\forall x\ w(x)=1$].

Goal: pick a permutation uniformly at random.

Why Sampling?

- The use of samples allows us to conduct studies with more manageable data and in a timely manner.
- Randomly drawn samples do not have much bias if they are large enough, but achieving such a sample may be expensive and time-consuming.
- We often need to compute statistics of “typical” configurations: estimating mean of a stochastic process or mean energy, …
- Estimating the statistics of a posterior density function in Bayesian inference.

Example: Estimating the Mean of f(X)

- Want to compute $E[f(\boldsymbol{X})]$ for function $f(\cdot)$.
- Standard method for approximating $E[f(\boldsymbol{X})]$ is to generate many independent sample values of $\boldsymbol{X}$ and compute sample mean of $f(\boldsymbol{X})$.
- Only useful in "trivial" cases where $\boldsymbol{X}$ can be generated directly.
- Many practical problems have non-trivial distribution for $\boldsymbol{X}$
  - E.g., state in nonlinear/non-Gaussian state-space model, Bayesian inference, …

The Monte Carlo Principle

- p(x): a target density defined over a high-dimensional space (samples, the space of all possible configurations of a system under study)
- The idea of Monte Carlo techniques is to draw a set of (iid) samples $\{x^{(i)}\}$ for i = 1, … , N, from p(x) in order to approximate p(x) with the empirical distribution:

$$p(x) \approx \frac{1}{N}\sum_{i=1}^{N} \delta(x = x^{(i)})$$

- Using these samples, we can approximate integrals I(f) with tractable sums that converge (as the number of samples grows) to I(f):

$$I(f) = \int f(x)p(x)dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)}) \xrightarrow[N\to\infty]{} I(f)$$

The Monte Carlo Core Concept

- The idea behind Monte Carlo sampling is to use randomness to approximate deterministic values.
- For example, If you want to estimate the value of a definite integral:

$$I = \int_a^b f(x)dx$$

- you can approximate this integral using Monte Carlo sampling by:

$$I \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$

- where x_i are random samples drawn uniformly from $[a,b]$.

Steps & Benefits of Monte Carlo Sampling

Steps in Monte Carlo Sampling:

- Define the Problem: Represent the problem in terms of probabilities or expectations.
- Generate Samples: Draw random samples from the relevant distribution or space.
- Compute Function Values: Evaluate the target function for each sample.
- Aggregate Results: Combine the results (e.g., averaging, summing) to estimate the desired quantity.

Benefits of Monte Carlo Sampling

- **Scalability**: Works well in high-dimensional problems.
- **Flexibility**: Can handle complex, irregularly shaped domains or distributions.
- **Accuracy**: Improves with the number of samples, based on the law of large numbers.

Example of Monte Carlo Sampling

Example: Estimating π Using Monte Carlo

- Imagine a circle inscribed in a square. The area of the circle (πr^2) and the area of the square ($4r^2$) give:

$$\frac{\text{Area of Circle}}{\text{Area of Square}} = \frac{\pi}{4}$$

To estimate π:

1. Generate random points within the square.
2. Count how many points fall inside the circle.
3. Use the ratio of points in the circle to total points to estimate π:

$$\pi \approx 4 \times \frac{\text{Number of Points in Circle}}{\text{Total Number of Points}}$$

Importance Sampling

- **Importance sampling** is a statistical technique for estimating properties of a particular distribution using samples from a different distribution.
- It is commonly used in situations where direct sampling from the target distribution is difficult or when some areas of the distribution contribute more significantly to the desired out

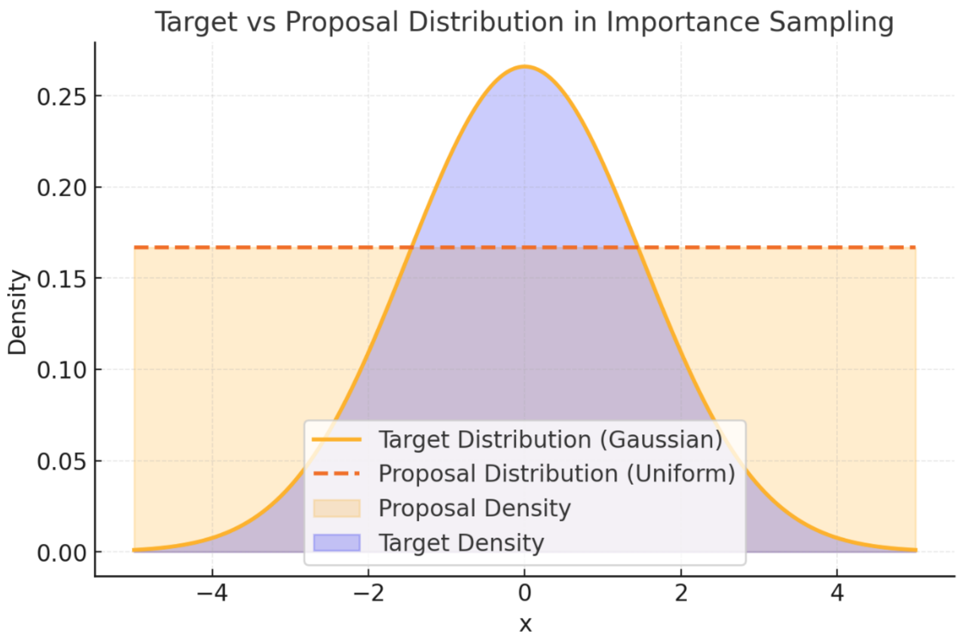
Key Concepts

1. **Target Distribution** (p(x)): The distribution you want to analyze or estimate properties for, such as mean or variance.
2. **Proposal Distribution** (q(x)): The distribution from which samples are drawn because it is easier to sample from than the target distribution.

Importance Sampling

Core Idea

- Instead of sampling directly from p(x), you sample from q(x) and reweight the samples to reflect their importance relative to p(x). The weights correct for the fact that q(x) is not the target distribution.



Importance Sampling

- **Estimating an Expectation**
- Suppose you want to estimate the expectation of a function f(x) under the target distribution p(x):

$$\mathbb{E}_p[f(x)] = \int f(x)p(x)dx$$

- If sampling directly from p(x) is difficult, but you can sample from q(x), the expectation can be rewritten as:

$$\mathbb{E}_p[f(x)] = \int f(x)\frac{p(x)}{q(x)}q(x)dx$$

- This implies:

$$\mathbb{E}_p[f(x)] \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i)w(x_i)$$

Importance Sampling

where:

- $x_i \sim q(x)$
- $w(x_i) = \frac{p(x_i)}{q(x_i)}$ are the importance weights.
- Normalized Weights to ensure stability and better convergence:

$$\mathbb{E}_p[f(x)] \approx \frac{\sum_{i=1}^N f(x_i) w(x_i)}{\sum_{i=1}^N w(x_i)}$$

Applications

- Monte Carlo Integration: Estimate integrals where direct evaluation is difficult.
- Bayesian Inference: Approximating posterior distributions in cases with complex likelihoods.
- Reinforcement Learning: Adjusting for the difference between behavior and target policies.

Importance Sampling

Challenges

- **Choice of Proposal Distribution**: q(x) must adequately cover the regions where p(x) is significant. Poor choices can lead to large variances in the estimates.
- **Weight Calculation**: Large differences between p(x) and q(x) can result in extreme weights, leading to numerical instability.

Tips

- Ensure q(x) has heavier tails than p(x) to avoid zero weights.
- Normalize weights to improve stability.

Sequential Monte Carlo

- Advantages of Sequential Sampling:
  - Real time processing
  - Dealing with non-stationarity
  - Not having to store the data
- Goal: Estimate the distribution of 'hidden' trajectories:
  - We observe y_t at each time *t:* $p(x_{0:t} \mid y_{1:t})$, where
  - We have a model:
    - Initial distribution: $p(x_0)$
    - Dynamic model: $p(x_t | x_{0:t-1}, y_{1:t-1})$ for $t \geq 1$
    - Measurement model: $p(y_t | x_{0:t}, y_{1:t-1})$ for $t \geq 1$

Sequential Monte Carlo

- Can define a proposal distribution:

$$q(\widetilde{x}_{0:t}|y_{1:t}) = p(x_{0:t-1}|y_{1:t-1})q(\widetilde{x}_t|x_{0:t-1}, y_{1:t})$$

- Then the importance weights are:

$$\begin{aligned} w_t &= \frac{p(\widetilde{x}_{0:t}|y_{1:t})}{q(\widetilde{x}_{0:t}|y_{1:t})} = \frac{p(x_{0:t-1}|y_{1:t})}{p(x_{0:t-1}|y_{1:t-1})} \frac{p(\widetilde{x}_t|x_{0:t-1}, y_{1:t})}{q(\widetilde{x}_t|x_{0:t-1}, y_{1:t})} \\ &\propto \frac{p\left(y_t|\widetilde{x}_t\right) p\left(\widetilde{x}_t|x_{0:t-1}, y_{1:t-1}\right)}{q_t\left(\widetilde{x}_t|x_{0:t-1}, y_{1:t}\right)}. \end{aligned}$$

- Simplifying the choice for proposal distribution:
  Then: $q(\widetilde{x}_t|x_{0:t-1}, y_{1:t}) = p\left(\widetilde{x}_t|x_{0:t-1}, y_{1:t-1}\right)$

  $w_t \propto p\left(y_t|\widetilde{x}_t\right)$ ‘fitness’

Sequential Monte Carlo

<u>*Sequential importance sampling step*</u>

– For $i = 1, ..., N$, sample from the transition priors

$$\widetilde{x}_t^{(i)} \sim q_t\left(\widetilde{x}_t | x_{0:t-1}^{(i)}, y_{1:t}\right)$$

and set

$$\widetilde{x}_{0:t}^{(i)} \triangleq \left(\widetilde{x}_t^{(i)}, x_{0:t-1}^{(i)}\right)$$

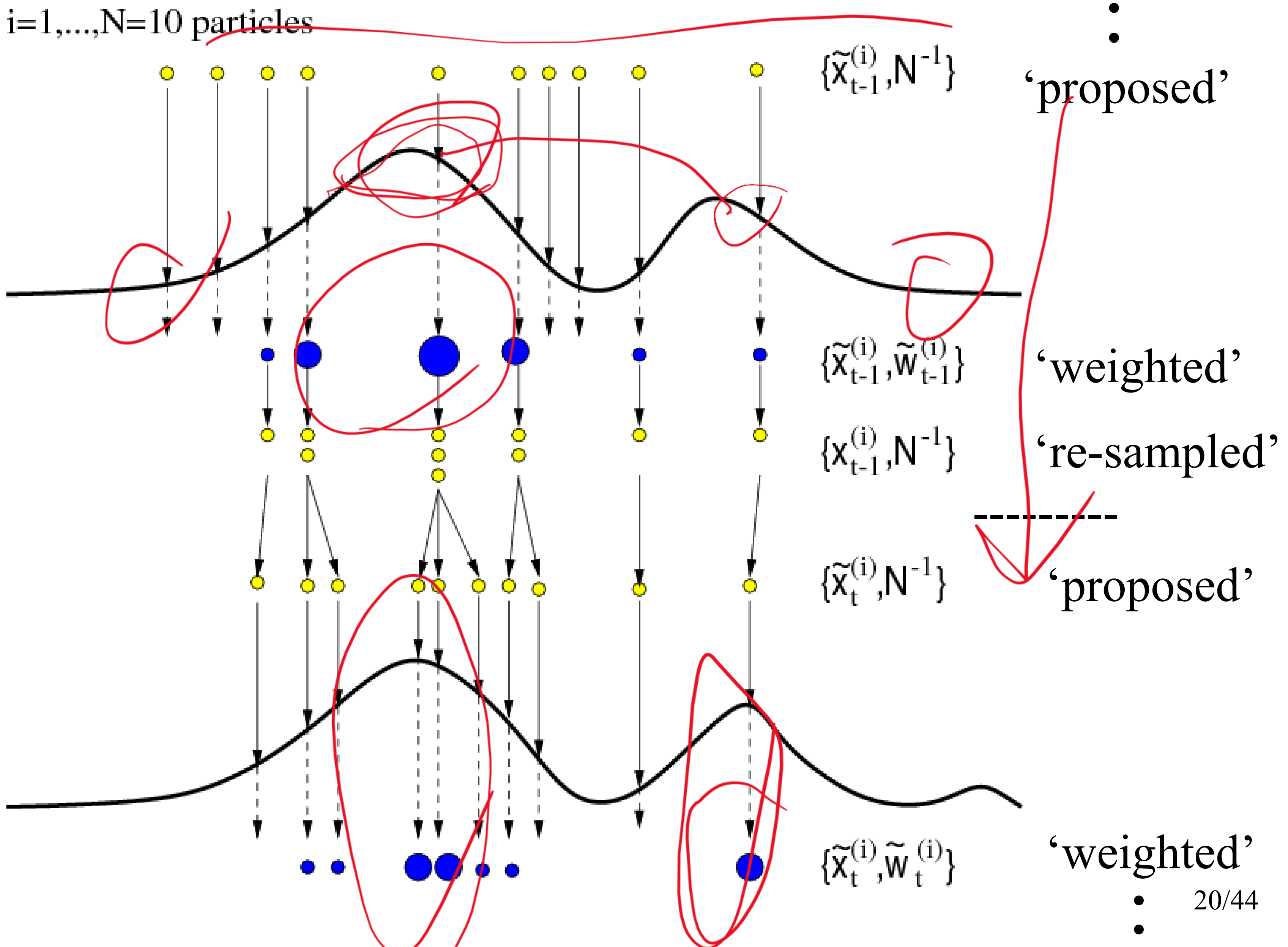
– For $i = 1, ..., N$, evaluate and normalize the importance weights

$$w_t^{(i)} \propto \frac{p\left(y_t | \widetilde{x}_t^{(i)}\right) p\left(\widetilde{x}_t^{(i)} | x_{0:t-1}^{(i)}, y_{1:t-1}\right)}{q_t\left(\widetilde{x}_t^{(i)} | x_{0:t-1}^{(i)}, y_{1:t}\right)}.$$

<u>*Selection step*</u>

– Multiply/Discard particles $\left\{\widetilde{x}_{0:t}^{(i)}\right\}_{i=1}^{N}$ with high/low importance weights $w_t^{(i)}$ to obtain N particles $\left\{x_{0:t}^{(i)}\right\}_{i=1}^{N}$.

Sequential Monte Carlo


i=1,...,N=10 particles
$\{\tilde{x}_{t-1}^{(i)}, N^{-1}\}$
'proposed'
$\{\tilde{x}_{t-1}^{(i)}, \tilde{w}_{t-1}^{(i)}\}$
'weighted'
$\{x_{t-1}^{(i)}, N^{-1}\}$
're-sampled'
$\{\tilde{x}_{t}^{(i)}, N^{-1}\}$
'proposed'
$\{\tilde{x}_{t}^{(i)}, \tilde{w}_{t}^{(i)}\}$
'weighted'

Three uses of Monte Carlo methods

1. For solving problems of probabilistic inference involved in developing computational models

2. As a source of hypotheses about how the mind might solve problems of probabilistic inference

3. As a way to explore people’s subjective probability distributions

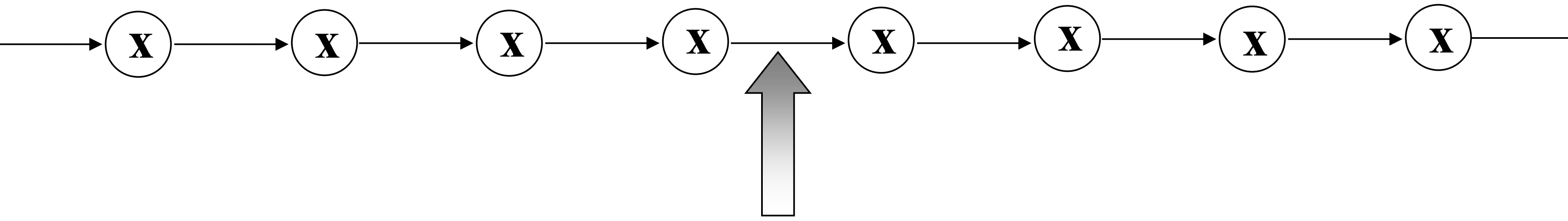
Applications of Monte Carlo Sampling

- Computer vision
- Speech & audio enhancement
- Web statistics estimation
- Regression & classification
- Bayesian networks
- Genetics & molecular biology
- Robotics, etc.
- …

Markov Chain Monte Carlo

- Basic idea: Construct a *Markov chain* that will converge to the target distribution and draw samples from that chain.
- Just uses something proportional to the target distribution (good for Bayesian inference!).
- Can work in state spaces of arbitrary (including unbounded) size (good for nonparametric Bayes).

Markov Chains



Transition matrix

$\mathbf{T} = P(\mathbf{x}^{(t+1)} | \mathbf{x}^{(t)})$

Variables $\mathbf{x}^{(t+1)}$ independent of all previous variables given immediate predecessor $\mathbf{x}^{(t)}$

An example: Card Shuffling

- Each state $\mathbf{x}^{(t)}$ is a permutation of a deck of cards (there are 52! permutations)
- Transition matrix **T** indicates how likely one permutation will become another
- The transition probabilities are determined by the shuffling procedure:
  - Riffle Shuffle
  - Overhand
  - One Card

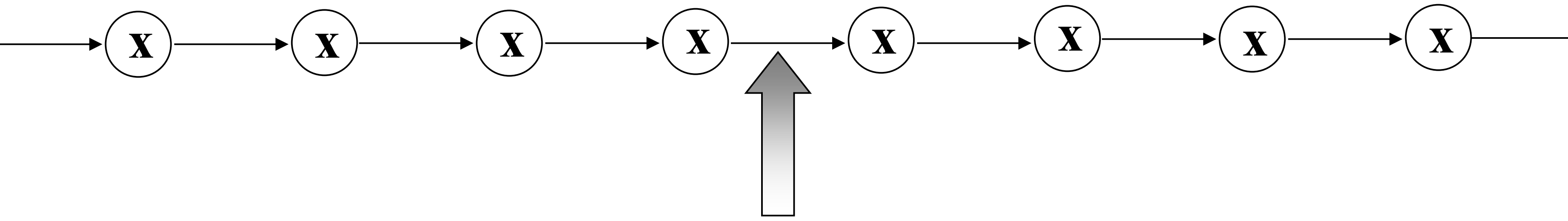
Convergence of Markov Chains

- Why do we shuffle cards?
- Convergence to a uniform distribution takes only 7 riffle shuffles…
- Other Markov chains will also converge to a *stationary distribution*, if certain simple conditions are satisfied (called “ergodicity”)
  - e.g. every state can be reached in some number of steps from every other state

Modern Monte Carlo methods

- Sampling schemes for distributions with large state spaces known up to a multiplicative constant
- Two approaches:
  - Importance Sampling (particle filters)
  - Markov Chain Monte Carlo

Markov chain Monte Carlo



Transition matrix

$\mathbf{T} = P(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)})$

- States of chain are variables of interest
- Transition matrix chosen to give target distribution as stationary distribution

The Markov Chain Monte Carlo (MCMC)

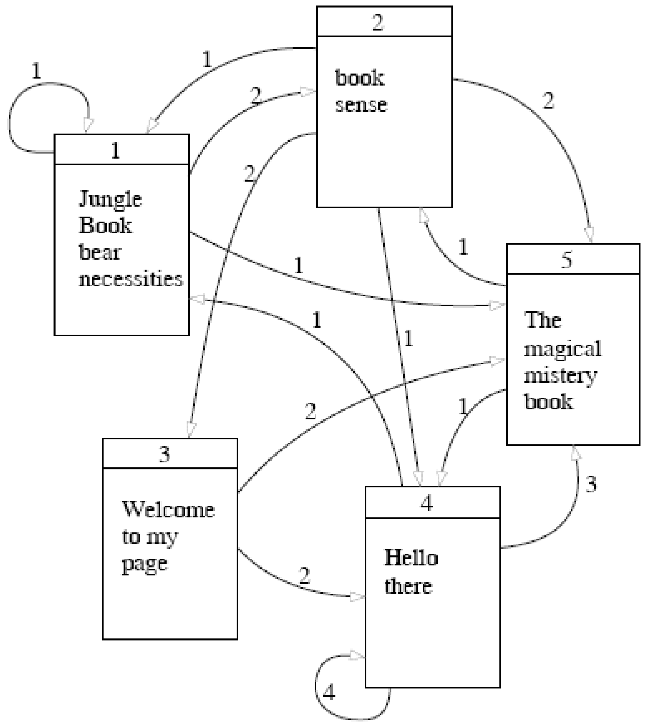
- Design a Markov Chain on finite state space:

$$\text{state space}: x^{(i)} \in \{x_1, x_2, ..., x_s\}$$

$$\text{Markov property}:\ p(x^{(i)} \mid x^{(i-1)}, ..., x^{(1)}) = T(x^{(i)} \mid x^{(i-1)})$$

such that when simulating a trajectory of states from it, it will explore the state space spending more time in the most important regions (i.e. where p(x) is large)

Stationary distribution of a MC



- Suppose you browse the Web for an infinitely long time, no matter where you started:
- What is the probability of being at page x_i.

  =>PageRank (Google)

$$p(x^{(i)} \mid x^{(i-1)}, \ldots, x^{(1)}) = T(x^{(i)} \mid x^{(i-1)}) \equiv \mathbf{T}$$

$$((\mu(x^{(1)})\mathbf{T})\mathbf{T})\ldots\mathbf{T} = \mu(x^{(1)})\mathbf{T}^n = p(x), \quad s.t.\ p(x)\mathbf{T} = p(x)$$

Google vs. MCMC

$$p(x)\mathbf{T} = p(x)$$

- Google: given **T**, finds *p(x)*
- MCMC: given *p(x),* finds **T**
  - But it also needs a ‘proposal (transition) probability distribution’ to be specified.
- Q: Do all MCs have a stationary distribution?
- A: No.

Conditions for existence of a unique stationary distribution

- Irreducibility
  - The transition graph is connected (any state can be reached)
- Aperiodicity
  - State trajectories drawn from the transition don’t get trapped into cycles
- MCMC samplers are irreducible and aperiodic MCs that converge to the target distribution
- These 2 conditions are not easy to impose directly

Reversibility

- Reversibility (also called ‘detailed balance’) is a sufficient (but not necessary) condition for *p(x)* to be the stationary distribution.

$$p(x^{(i)})T(x^{(i-1)}|x^{(i)}) = p(x^{(i-1)})T(x^{(i)}|x^{(i-1)}).$$

Summing both sides over $x^{(i-1)}$, gives us

$$p(x^{(i)}) = \sum_{x^{(i-1)}} p(x^{(i-1)})T(x^{(i)}|x^{(i-1)}).$$

- It is easier to work with this condition.

MCMC Algorithms

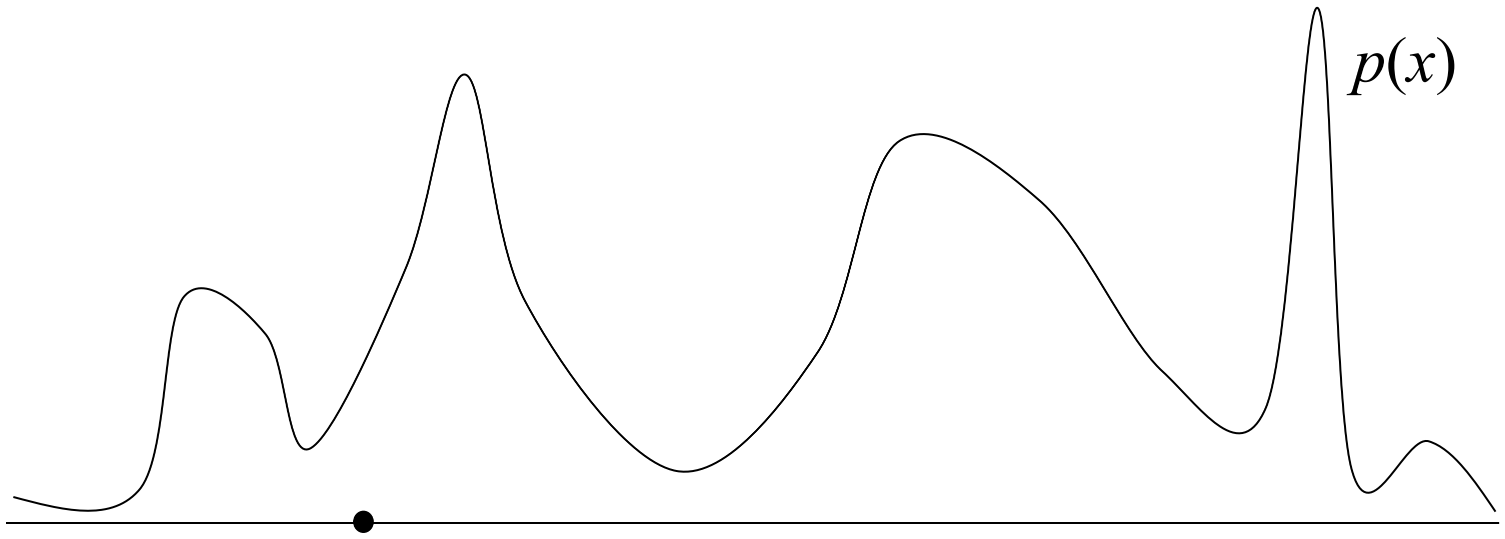
- Metropolis-Hastings algorithm
- Metropolis algorithm
  - Mixtures and blocks
- Gibbs sampling
- Sequential Monte Carlo & Particle Filters

Metropolis-Hastings Algorithm

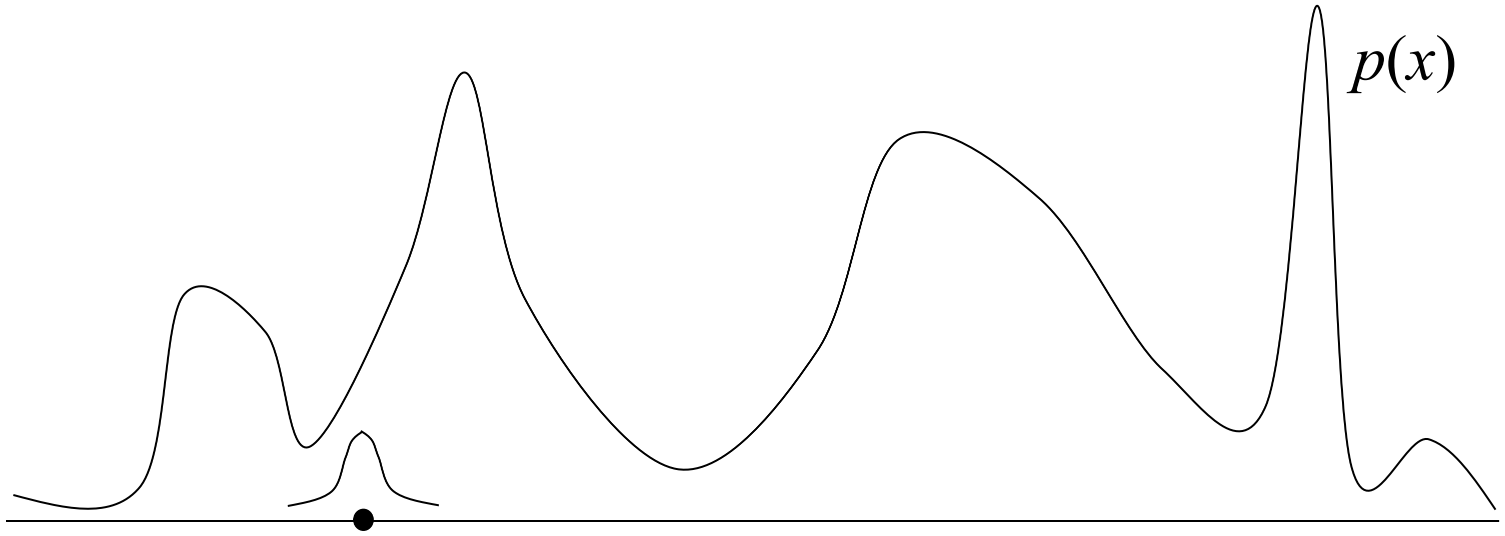
- Transitions have two parts:
  - proposal distribution: $q(\mathbf{x}^{(t+1)}|\mathbf{x}^{(t)})$
  - acceptance: take proposals with probability

$$A(\mathbf{x}^{(t)},\mathbf{x}^{(t+1)}) = \min\left(1, \frac{P(\mathbf{x}^{(t+1)})\, q(\mathbf{x}^{(t)}|\mathbf{x}^{(t+1)})}{P(\mathbf{x}^{(t)})\, q(\mathbf{x}^{(t+1)}|\mathbf{x}^{(t)})} \right)$$

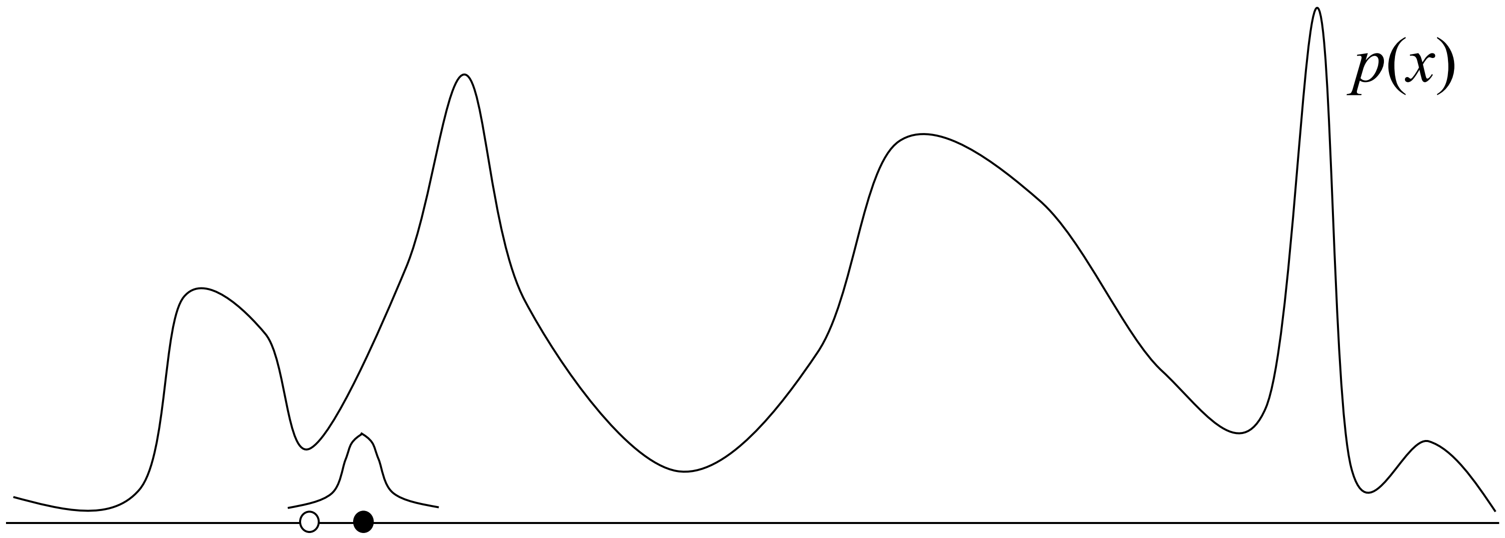
Metropolis-Hastings algorithm



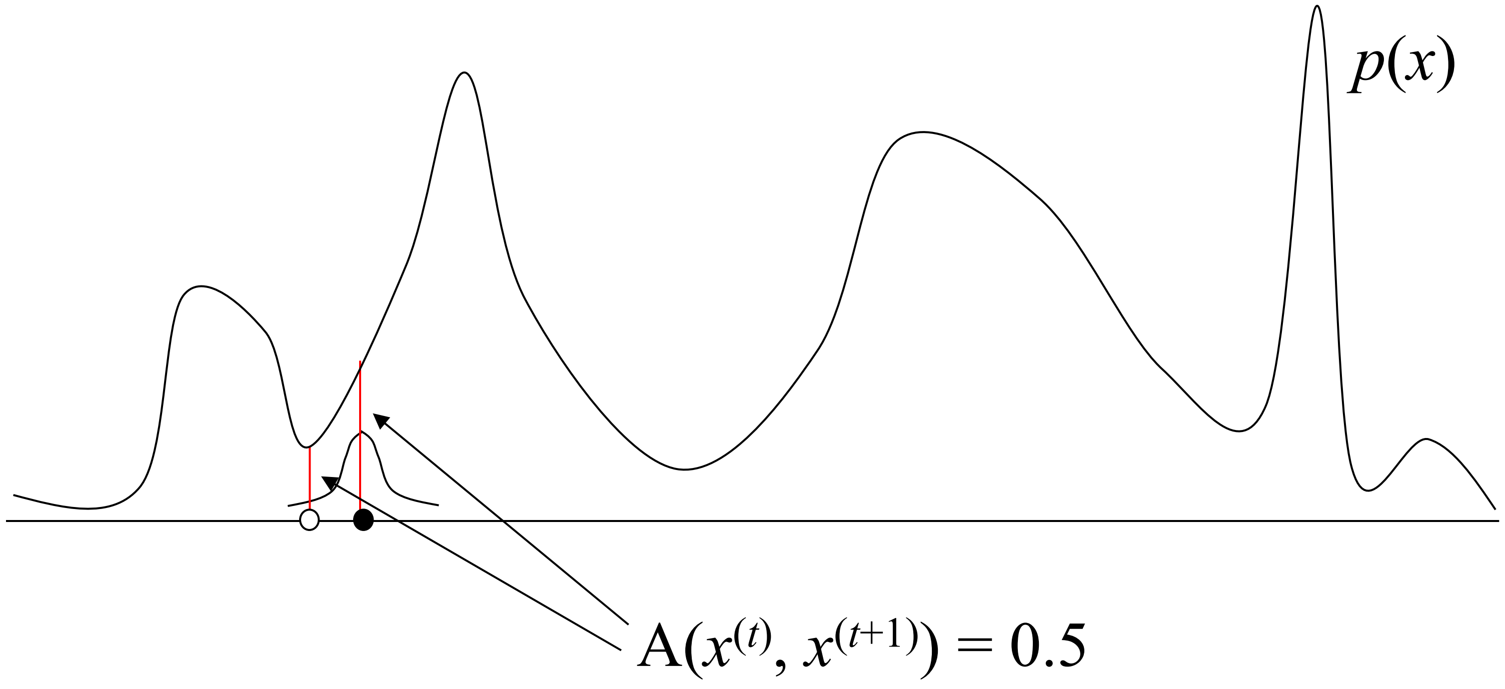
Metropolis-Hastings algorithm



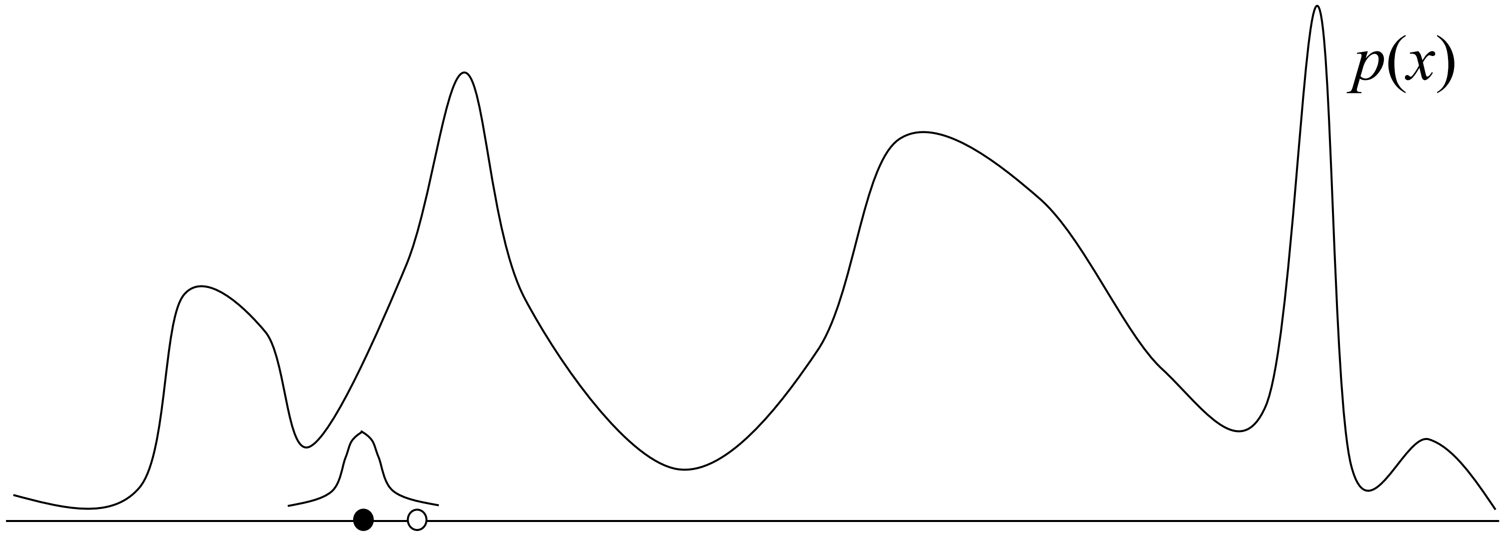
Metropolis-Hastings algorithm



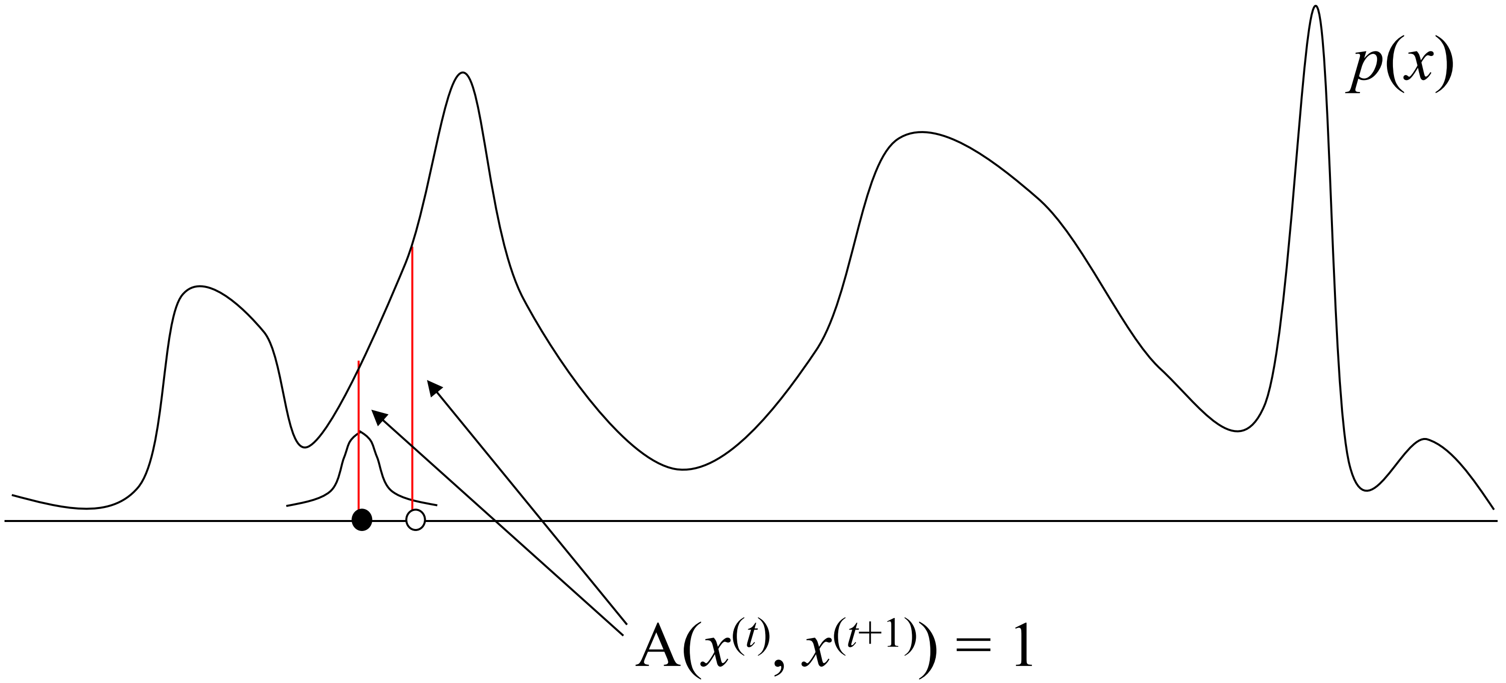
Metropolis-Hastings algorithm



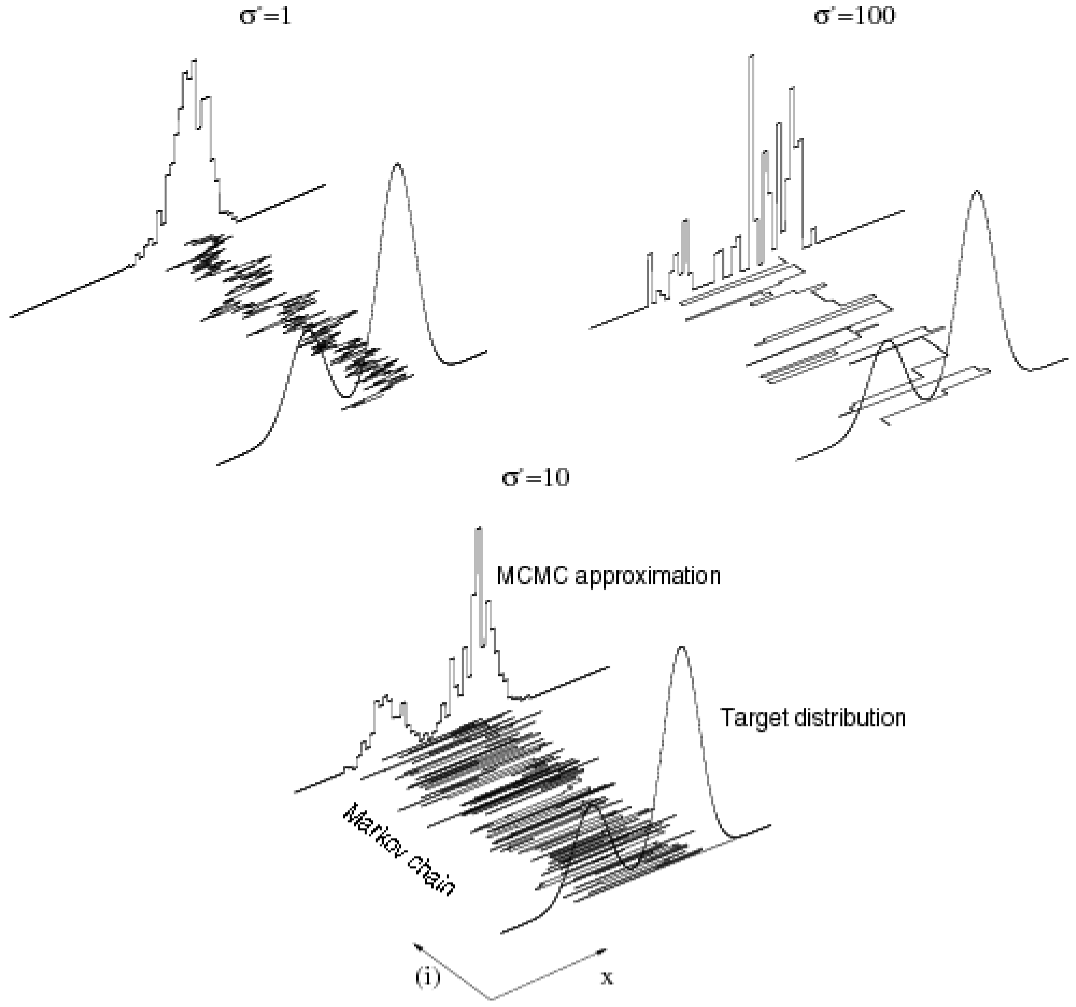
Metropolis-Hastings algorithm



Metropolis-Hastings algorithm



Examples of M-H simulations with q(x) a Gaussian with variance σ



The Metropolis-Hastings and the Metropolis algorithm as a special case

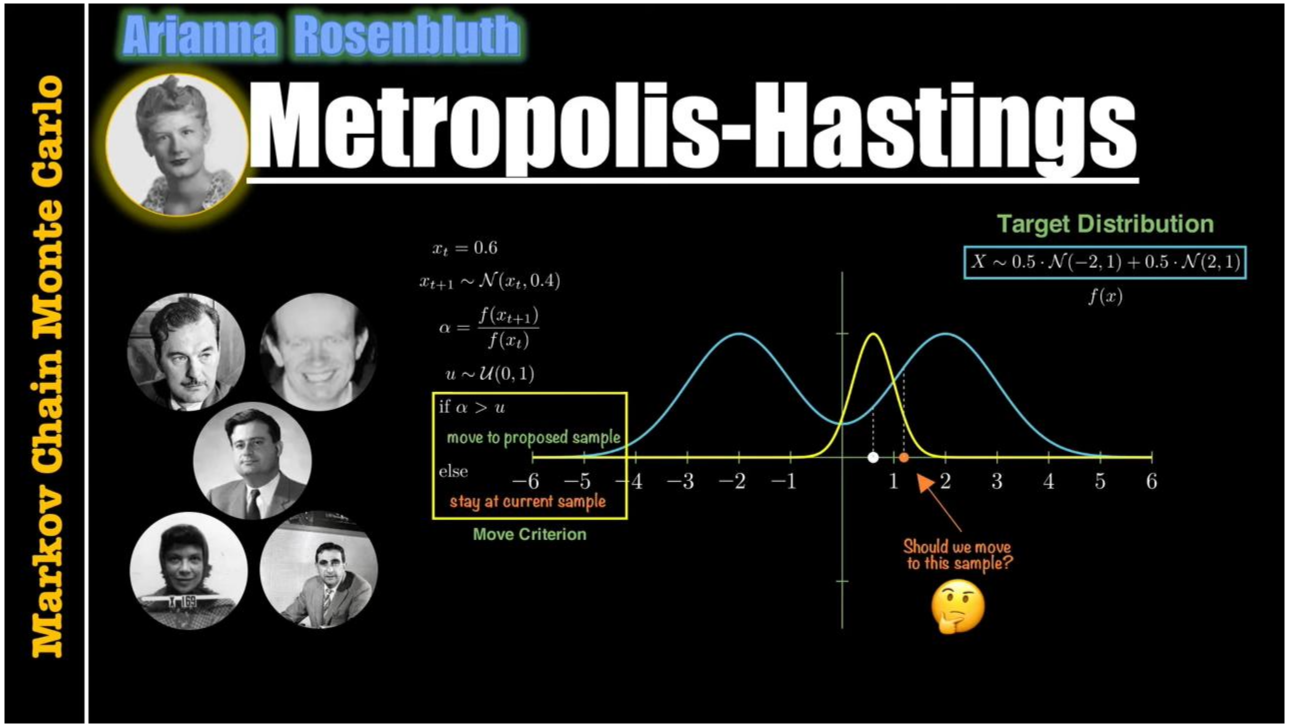
```
1. Initialise $x^{(0)}$.
2. For $i = 0$ to $N-1$
    – Sample $u \sim \mathcal{U}_{[0,1]}$.
    – Sample $x^\star \sim q(x^\star | x^{(i)})$.
    – If $u < \mathcal{A}(x^{(i)}, x^\star) = \min\left\{1, \frac{p(x^\star) q(x^{(i)} | x^\star)}{p(x^{(i)}) q(x^\star | x^{(i)})}\right\}$
            $x^{(i+1)} = x^\star$
      else
            $x^{(i+1)} = x^{(i)}$
```

The Metropolis algorithm assumes a symmetric random walk proposal $q(x^\star | x^{(i)}) = q(x^{(i)} | x^\star)$ and, hence, the acceptance ratio simplifies to

$$\mathcal{A}(x^{(i)}, x^\star) = \min\left\{1, \frac{p(x^\star)}{p(x^{(i)})}\right\}.$$

Obs. The target distrib *p(x)* in only needed up to normalization.

The Metropolis-Hastings Tutorial Clip



Arianna Rosenbluth
Metropolis-Hastings
Markov Chain Monte Carlo
$x_t = 0.6$
$x_{t+1} \sim \mathcal{N}(x_t, 0.4)$
$\alpha = \frac{f(x_{t+1})}{f(x_t)}$
$u \sim \mathcal{U}(0,1)$
if $\alpha > u$
move to proposed sample
else
stay at current sample
Move Criterion
Target Distribution
$X \sim 0.5 \cdot \mathcal{N}(-2,1) + 0.5 \cdot \mathcal{N}(2,1)$
$f(x)$
Should we move to this sample?

Gibbs Sampling

Gibbs sampling is a computationally convenient Bayesian inference algorithm that is a special case of the Metropolis–Hasting's algorithm.

- Component-wise proposal q(x):

$$q(x^\star|x^{(i)}) = \begin{cases} p(x_j^\star|x_{-j}^{(i)}) & \text{If } x^\star_{-j} = x_{-j}^{(i)} \\ 0 & \text{Otherwise.} \end{cases}$$

Where the notation means:

$$p(x_j|x_{-j}) = p(x_j|x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n)$$

- In this case, the acceptance probability is

$$\mathcal{A}(x^{(i)}, x^\star) = 1$$

Gibbs Sampling

Particular choice of proposal distribution

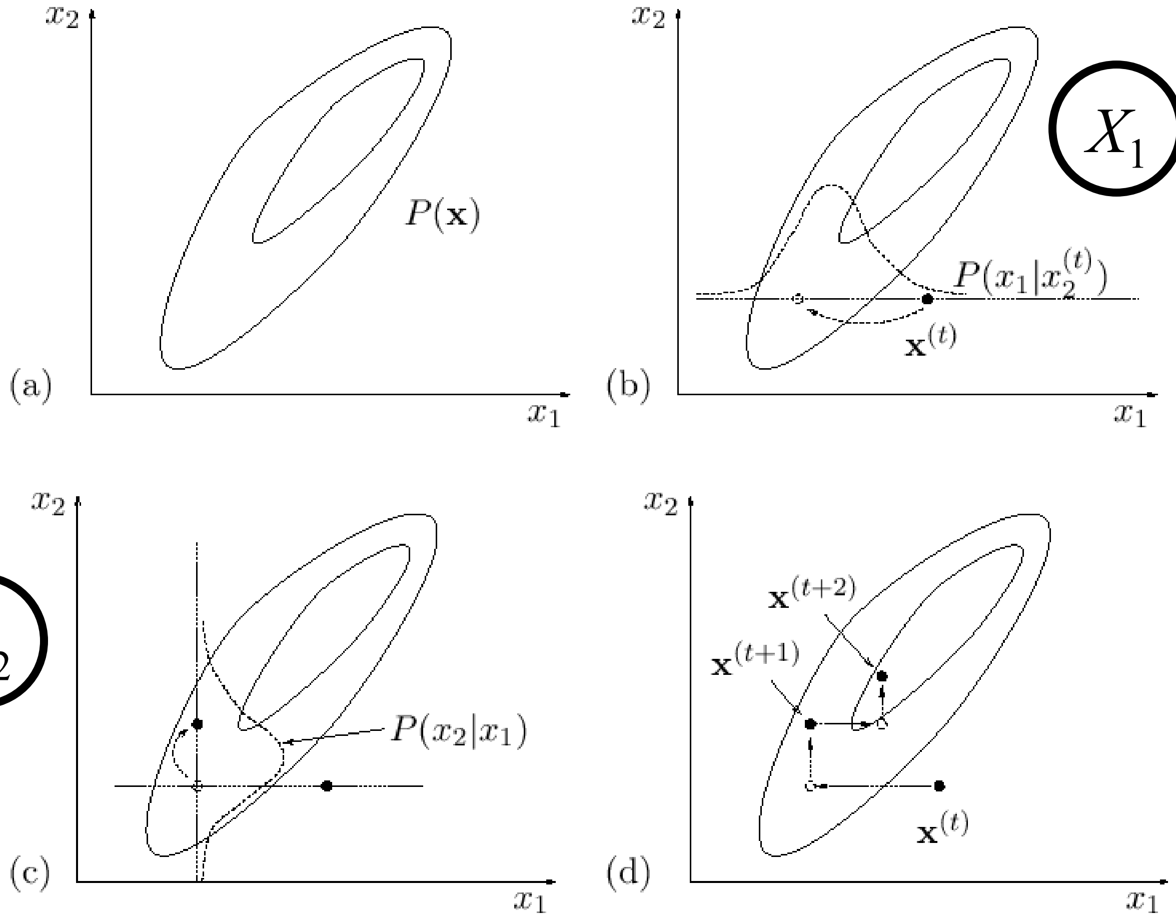
For variables $\mathbf{x} = x_1, x_2, \ldots, x_n$

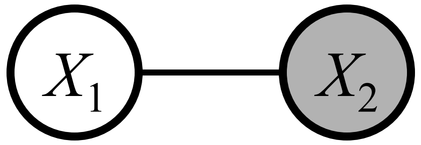
Draw $x_i^{(t+1)}$ from $P(x_i|\mathbf{x}_{-i})$

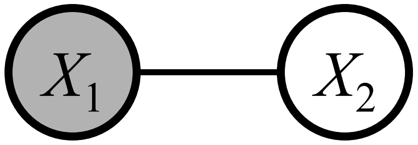
$$\mathbf{x}_{-i} = x_1^{(t+1)}, x_2^{(t+1)}, \ldots, x_{i-1}^{(t+1)}, x_{i+1}^{(t)}, \ldots, x_n^{(t)}$$

(this is called the *full conditional* distribution)

Gibbs sampling

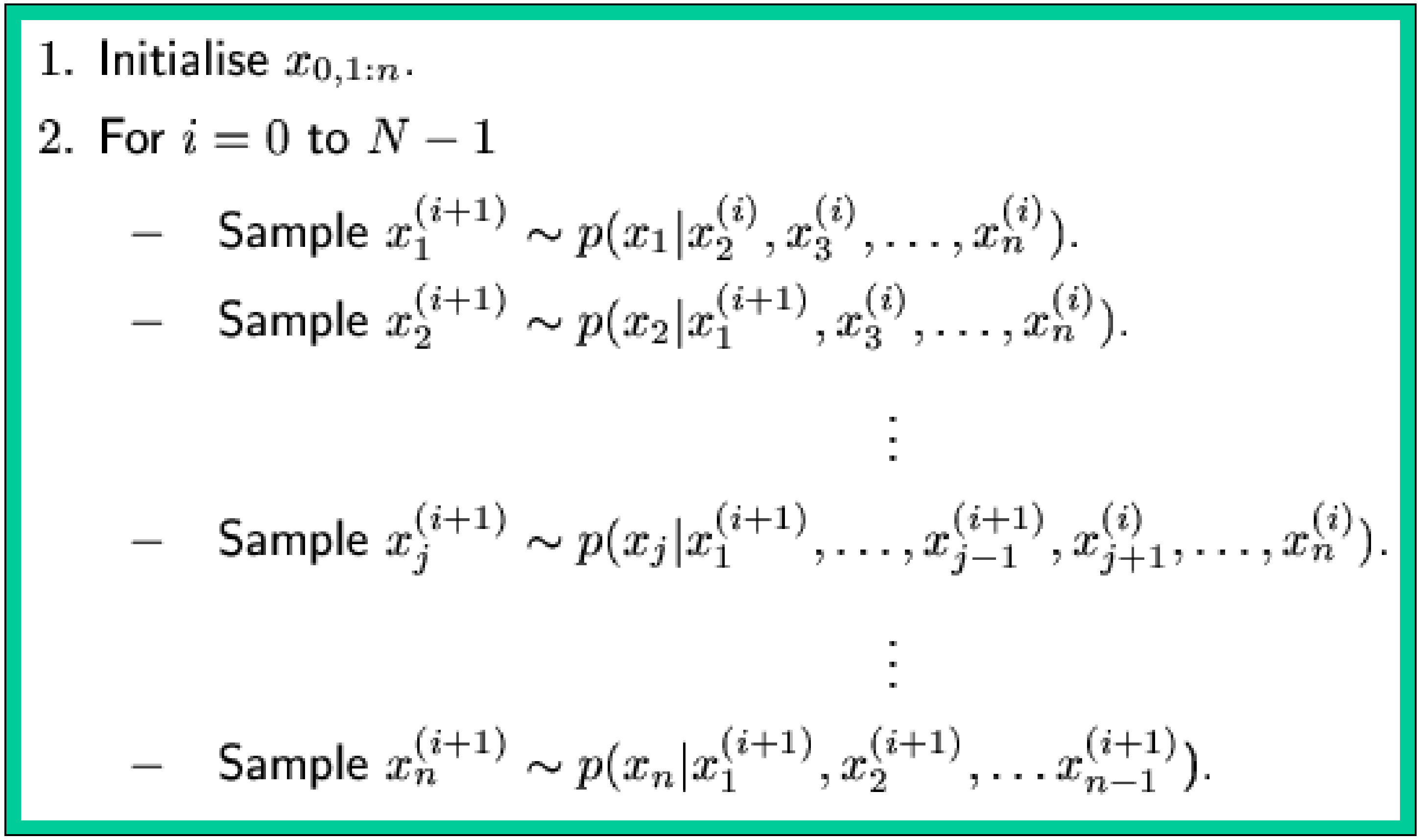






(MacKay, 2002)

Gibbs sampling algorithm



```
1. Initialise $x_{0,1:n}$.
2. For $i = 0$ to $N - 1$
      – Sample $x_1^{(i+1)} \sim p(x_1 | x_2^{(i)}, x_3^{(i)}, \ldots, x_n^{(i)})$.
      – Sample $x_2^{(i+1)} \sim p(x_2 | x_1^{(i+1)}, x_3^{(i)}, \ldots, x_n^{(i)})$.
                                  ⋮
      – Sample $x_j^{(i+1)} \sim p(x_j | x_1^{(i+1)}, \ldots, x_{j-1}^{(i+1)}, x_{j+1}^{(i)}, \ldots, x_n^{(i)})$.
                                  ⋮
      – Sample $x_n^{(i+1)} \sim p(x_n | x_1^{(i+1)}, x_2^{(i+1)}, \ldots x_{n-1}^{(i+1)})$.
```

The Promise of Particle Filters

- People need to be able to update probability distributions over large hypothesis spaces as more data becomes available
- Particle filters provide a way to do this with limited computing resources:
  - Maintain a fixed finite number of samples
- Not just for dynamic models:
  - Can work with a fixed set of hypotheses, although this requires some further tricks for maintaining diversity

The Magic of MCMC Methods

- Since we only ever need to evaluate the relative probabilities of two states, we can have huge state spaces (much of which we rarely reach)
- In fact, our state spaces can be *infinite*
  - Common with nonparametric Bayesian models
- But… the guarantees it provides are asymptotic
  - Making algorithms that converge in practical amounts of time is a significant challenge

References & Resources

[1] M Isard & A Blake: CONDENSATION – conditional density propagation for visual tracking. J of Computer Vision, 1998.

[2] C Andrieu, N de Freitas, A Doucet, M Jordan: An Introduction to MCMC for machine learning. Machine Learning, vol. 50, pp. 5--43, Jan. - Feb. 2003.

[3] MCMC preprint service: http://www.statslab.cam.ac.uk/~mcmc/pages/links.html.

[4] W.R. Gilks, S. Richardson & D.J. Spiegelhalter: Markov Chain Monte Carlo in Practice. Chapman & Hall, 1996.

- Associated demos & further papers: http://www.robots.ox.ac.uk/~misard/condensation.html.
- Nando de Freitas' MCMC papers & sw http://www.cs.ubc.ca/~nando/software.html.

Next Weeks:

I hope you enjoyed this course!

Have a good Final Exam!