Stochastic Processes

Week 10 (Version 1.2)

Sampling Methods

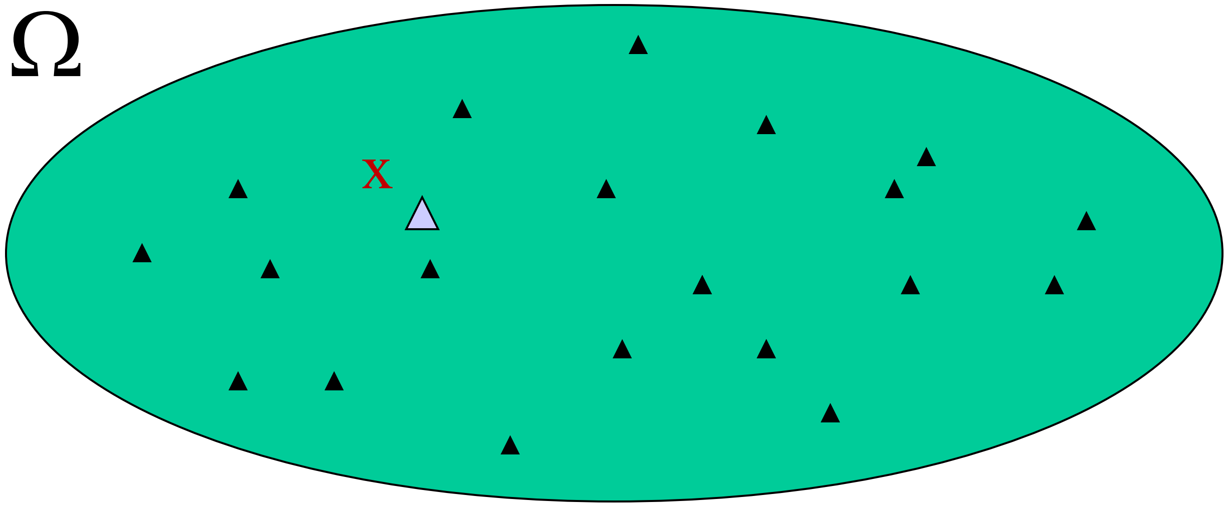
Hamid R. Rabiee

Fall 2023

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.

Goal: Sample points $x \in \Omega$ at random from distribution π.

The Probability Distribution

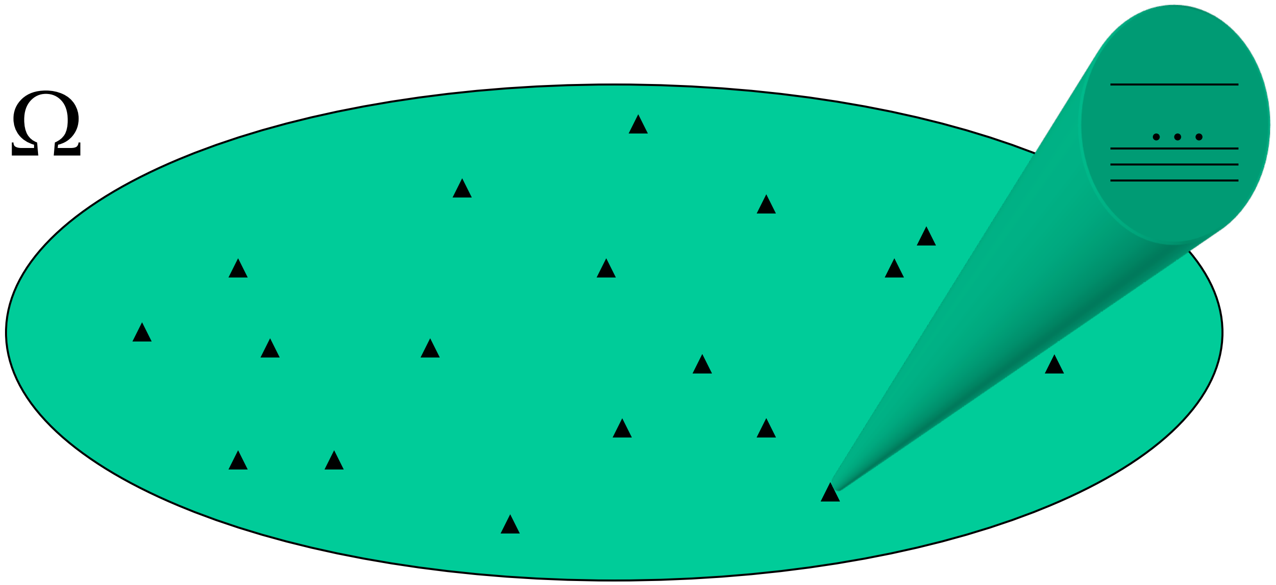
Typically,

$w:\Omega\rightarrow R^+$ 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 [$\forall x\ w(x)=1$].

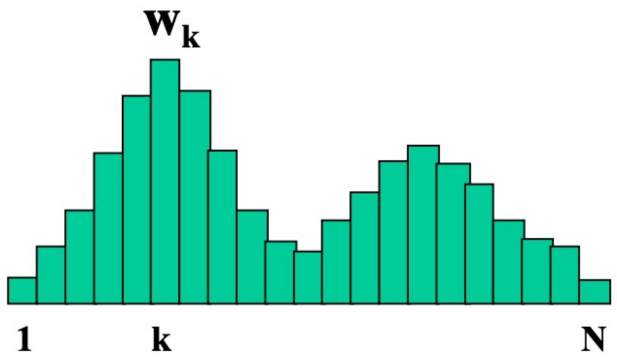
Goal: pick a permutation uniformly at random.

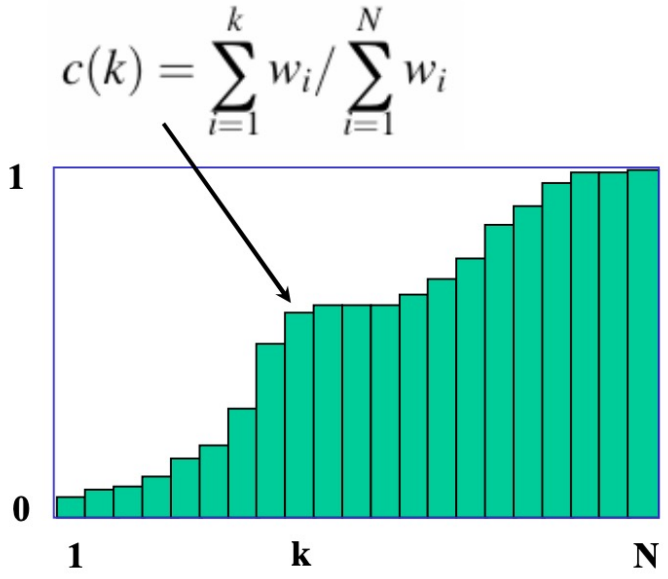
Why Sampling?

- 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.

Inverse Transform Sampling

- It is easy to sample from a discrete 1D distribution, using the cumulative distribution function (CDF).





cumulative distribution function

$$F(x) = P(X \leq x)$$

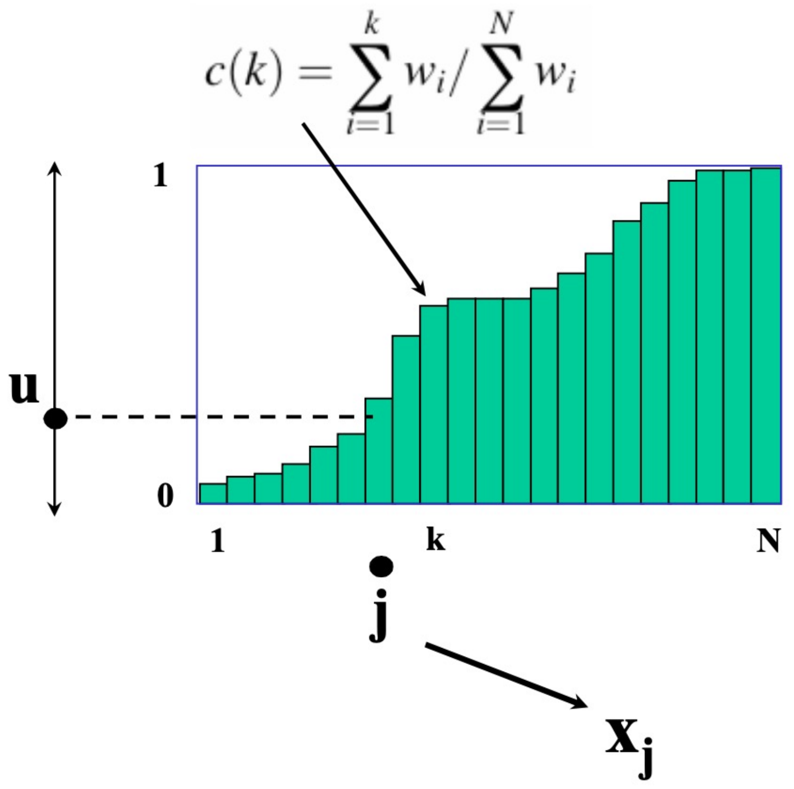
Inverse Transform Sampling

- It is easy to sample from a discrete 1D distribution, using the cumulative distribution function (CDF).

1) Generate uniform u in the range [0,1]

2) Visualize a horizontal line intersecting bars

3) If index of intersected bar is j, output new sample $\mathbf{x_j}$



Inverse Transform Sampling

Why it works:

cumulative distribution function

$$F(x) = P(X \le x)$$

inverse cumulative distribution function

$$F^{-1}(t) = \min\{x : F(x) = t, 0 < t < 1\}$$

Claim: if U is a uniform random variable on (0,1) then X=$F^{-1}(U)$ has distribution function F.
Proof:

$$\begin{aligned} & P(F^{-1}(U) \le x) \\ = \quad & P(\min\{x : F(x) = U\} \le x) && \text{(def of } F^{-1}\text{)} \\ = \quad & P(U \le F(x)) && \text{(applied } F \text{ to both sides)} \\ = \quad & F(x) && \text{(def of distribution function of } U\text{)} \end{aligned}$$

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 (e.g. 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:

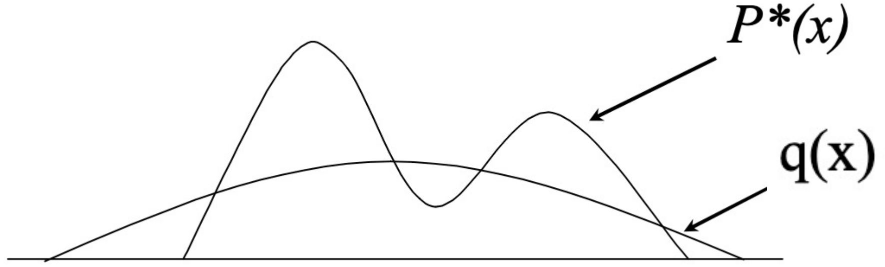
$$p(x) \approx \frac{1}{N} \sum_{i=1}^{N} \delta(x = x^{(i)})$$

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

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

Importance sampling

- Not a method for generating samples. It is a method for estimating expected value of functions $f(x_i)$.
- Target density $p(x)$ is known up to a constant
- Use a Proposal density that includes the support of $p(x)$
- An empirical estimate of $E_q(f(x))$, the expected value of $f(x)$ under distribution $q(x)$, then can be find.
- However, we want $E_P(f(x))$, which is the expected value of $f(x)$ under distribution $P(x)$.
- When we generate from $q(x)$, values of x where $q(x)$ is greater than $P^*(x)$ are overrepresented, and values where $q(x)$ is less than $P^*(x)$ are underrepresented.



Importance sampling at a glance

- Target density p(x) is known up to a constant
- Task: compute $I(f) = \int f(x) p(x) dx$

Idea:

- Introduce an arbitrary proposal density that includes the support of p(x). Then:

$$I(f) = \int f(x) \underbrace{p(x)/q(x)}_{w(x)\ \text{'importance weight'}} q(x)dx \approx \sum_{i=1}^{N} f(x^{(i)}) w(x^{(i)})$$

  - Sample from q instead of p
  - Weight the samples according to their ‘importance’

- It also implies that p(x) is approximated by:

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

Efficiency depends on a ‘good’ choice of q(x).

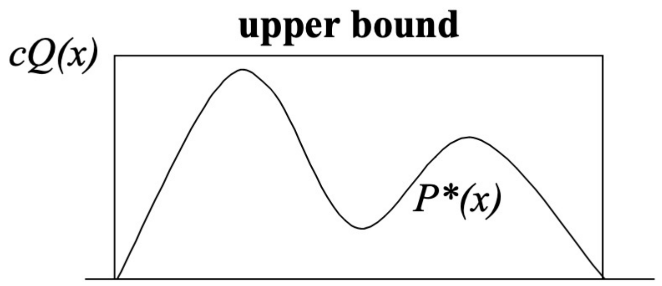
Importance sampling at a glance

- Computational efficiency is best if the proposal distribution looks a lot like the desired distribution (area between curves is small).
- These methods can fail badly when the proposal distribution has 0 density in a region where the desired distribution has non-negligeable density.
- For this reason, it is said that the proposal distribution should have heavy tails.

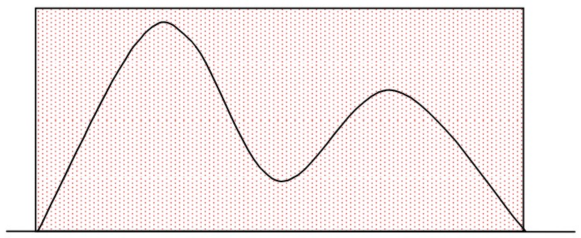
Rejection Sampling

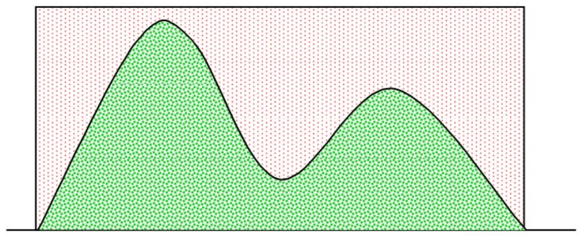
- Need a proposal density Q(x) [e.g. uniform or Gaussian], and a constant c such that c(Qx) is an upper bound for P*(x).

Example with Q(x) uniform:



generate uniform random samples in upper bound volume





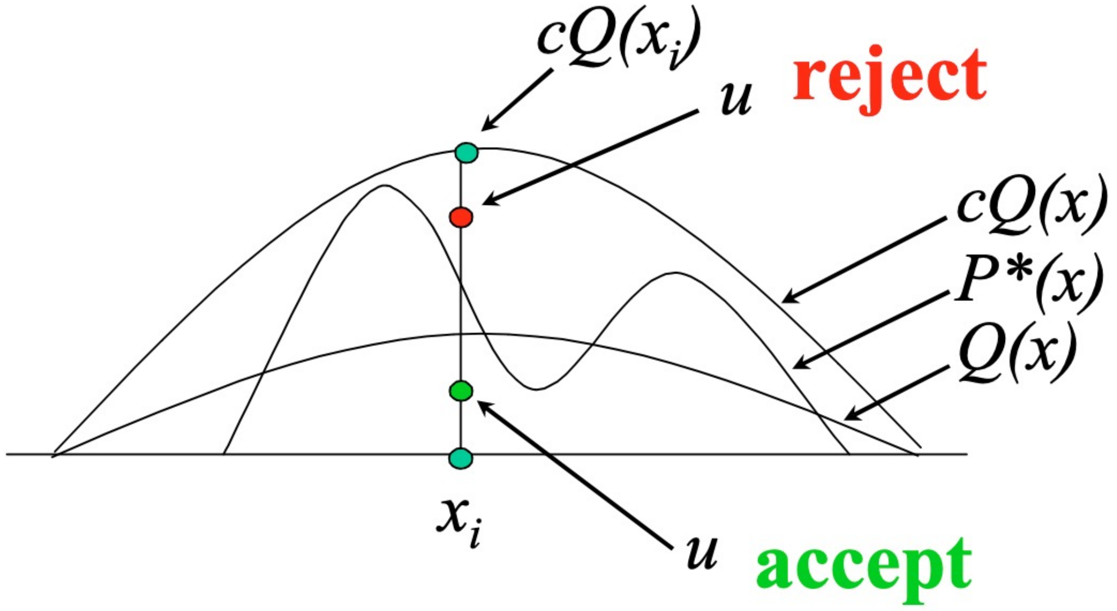
accept samples that fall below the P*(x) curve

the marginal density of the x coordinates of the points is then proportional to P*(x)

Note: this very related to Monte Carlo integration.

Rejection Sampling

- generally:
- 1) generate sample xi from a proposal density Q(x)
- 2) generate sample u from uniform [0,cQ(x_i)]
- 3) if u <= P*(x_i) accept x_i ; else reject



Sequential Monte Carlo

- Sequential Importance Sampling (SIS) and the closely related algorithm Sampling Importance Sampling (SIR) are known by various names in the literature:
  - bootstrap filtering
  - particle filtering
  - Condensation algorithm
  - survival of the fittest
- General idea: Importance sampling on time series data, with samples and weights updated as each new data term is observed. Well-suited for simulating recursive Bayes.

Sequential Monte Carlo

- Sequential:
  - 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

Sequential importance sampling step

– For $i = 1, \ldots, N$, 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

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

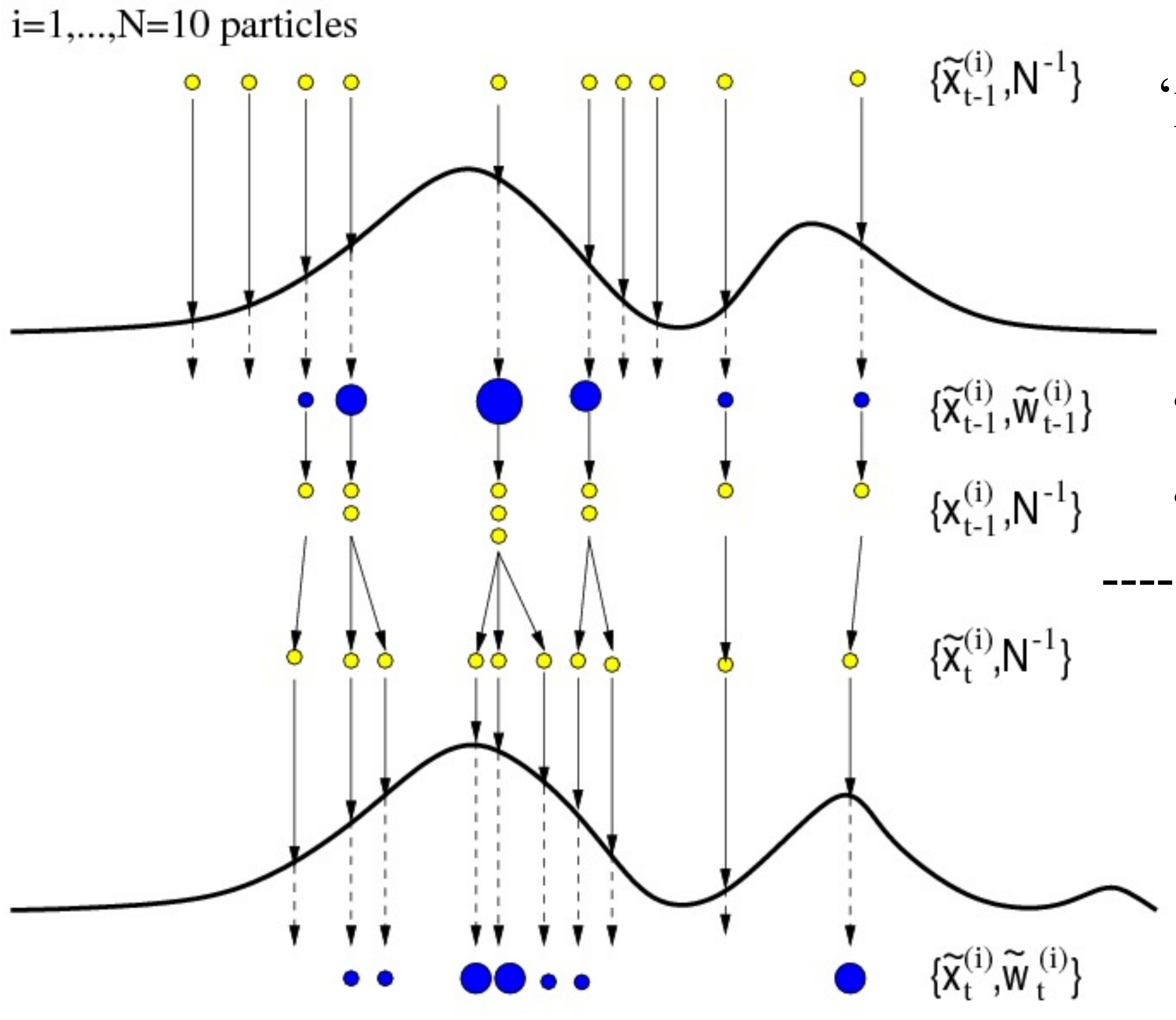
– For $i = 1, \ldots, N$, evaluate and normalize the importance weights

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

Selection step

– 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
$\{\tilde{x}_{t-1}^{(i)}, N^{-1}\}$
$\{\tilde{x}_{t-1}^{(i)}, \tilde{w}_{t-1}^{(i)}\}$
$\{x_{t-1}^{(i)}, N^{-1}\}$
$\{\tilde{x}_{t}^{(i)}, N^{-1}\}$
$\{\tilde{x}_{t}^{(i)}, \tilde{w}_{t}^{(i)}\}$

⋮

'proposed'

'weighted'

're-sampled'

'proposed'

'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

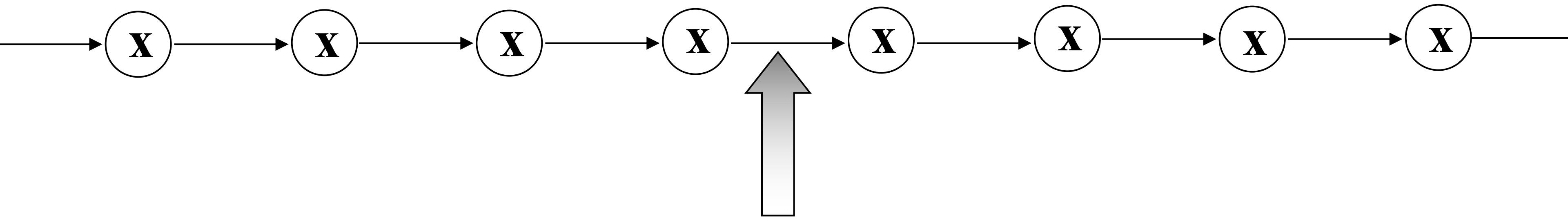
Applications on Monte Carlo Sampling

- 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 $\mathbf{T}$ indicates how likely one permutation will become another
- The transition probabilities are determined by the shuffling procedure
  - riffle shuffle
  - overhand
  - one card

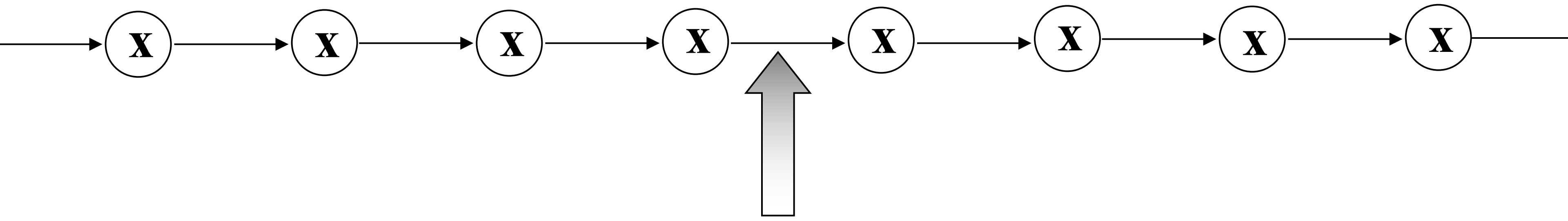
Convergence of Markov chains

- 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 (and 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)

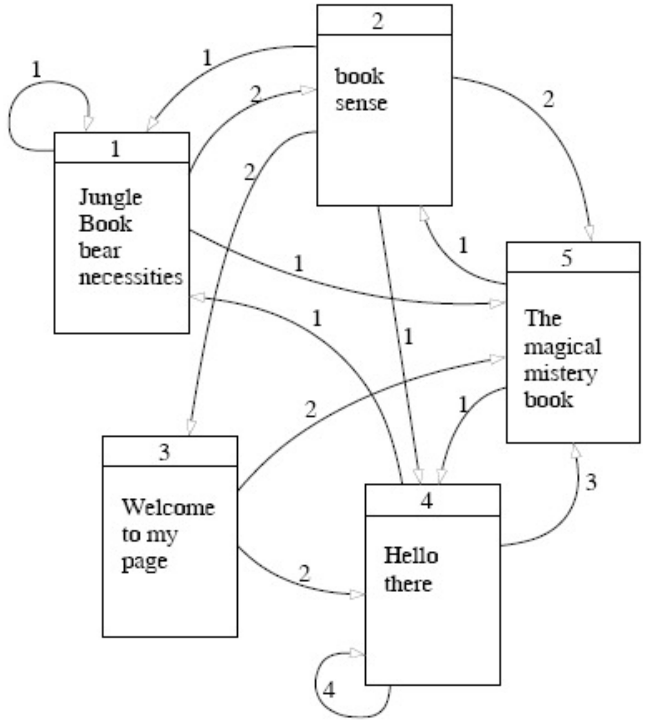
- Design a Markov Chain on finite state space:

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

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

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 this for infinitely long time, no matter where you started off:
- What is the probability to be 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

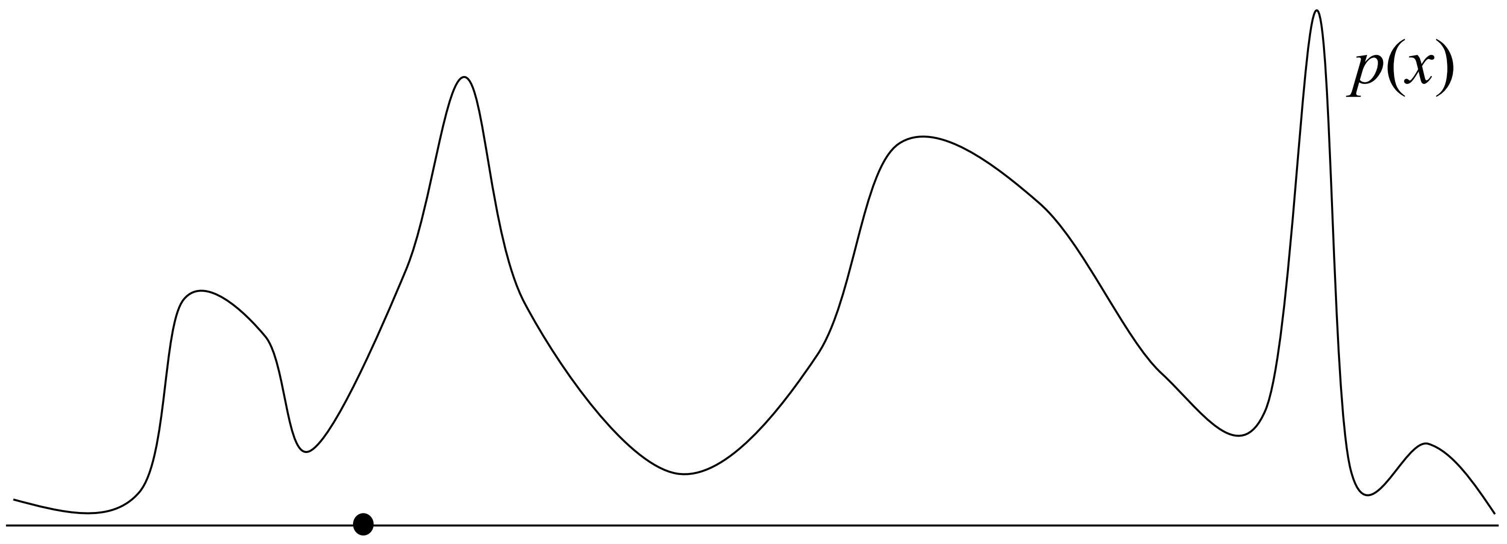
- Metropolis-Hastings algorithm
- Metropolis algorithm
  - Mixtures and blocks
- Gibbs sampling
- other
- Sequential Monte Carlo & Particle Filters

Metropolis-Hastings algorithm

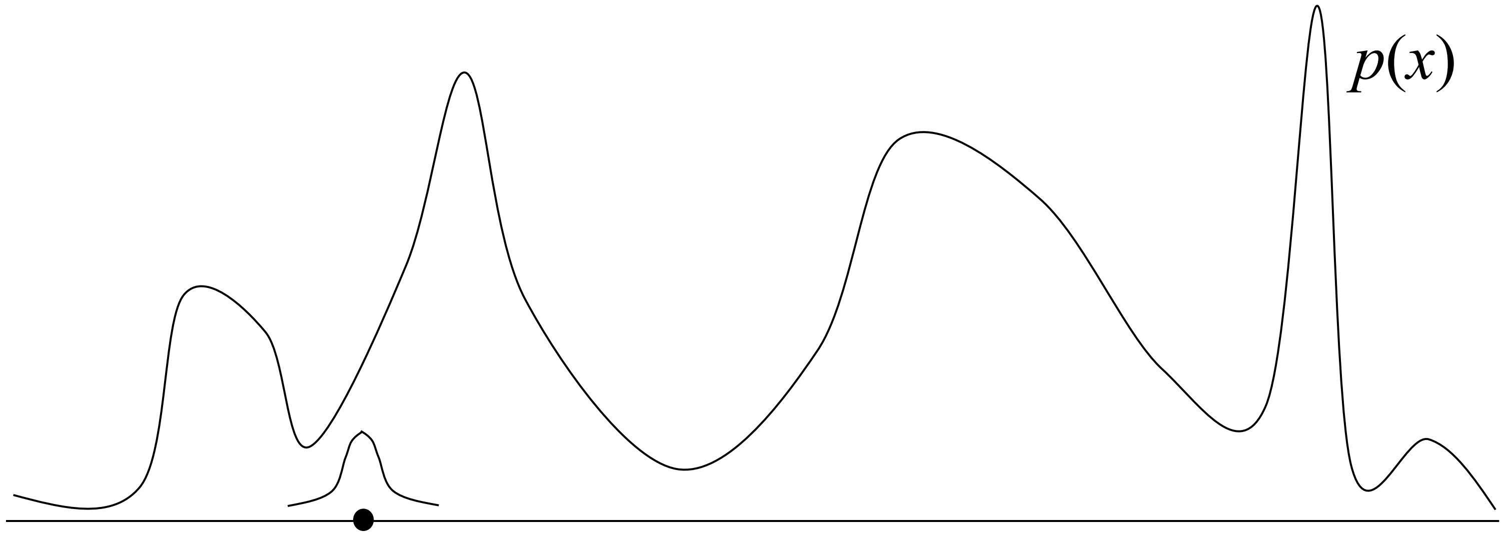
- Transitions have two parts:
  - proposal distribution: $q(\mathbf{x}^{(t+1)}|\mathbf{x}^{(t)})$
  - acceptance: take proposals with probability

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

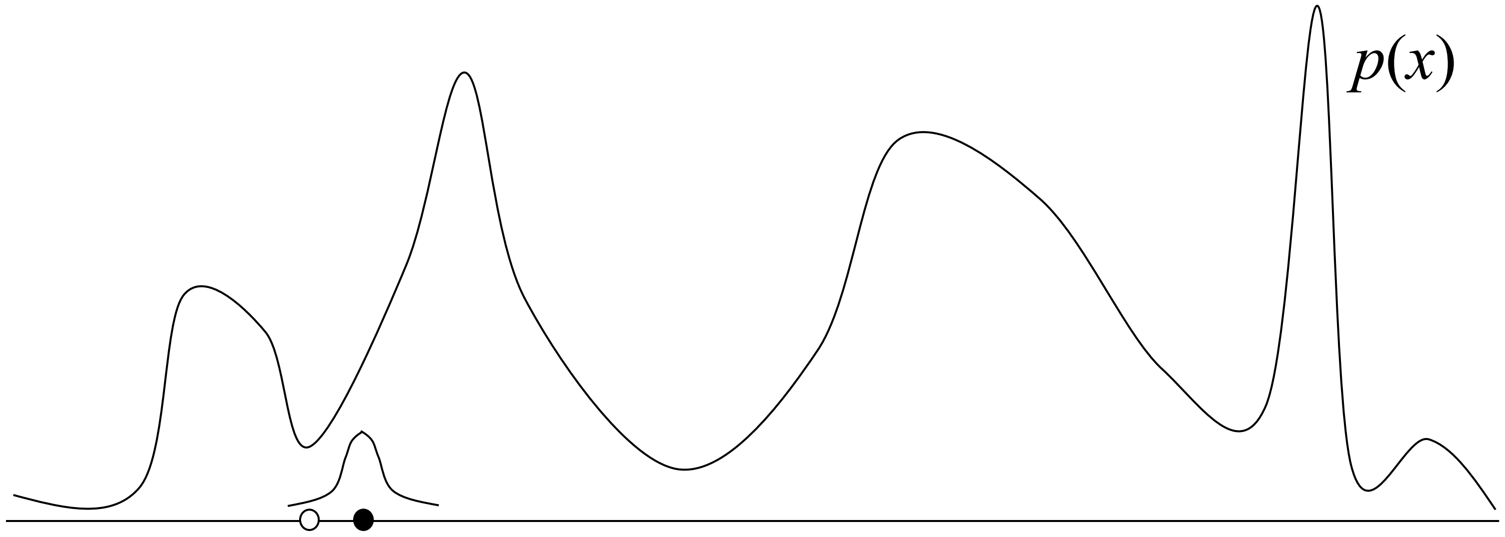
Metropolis-Hastings algorithm



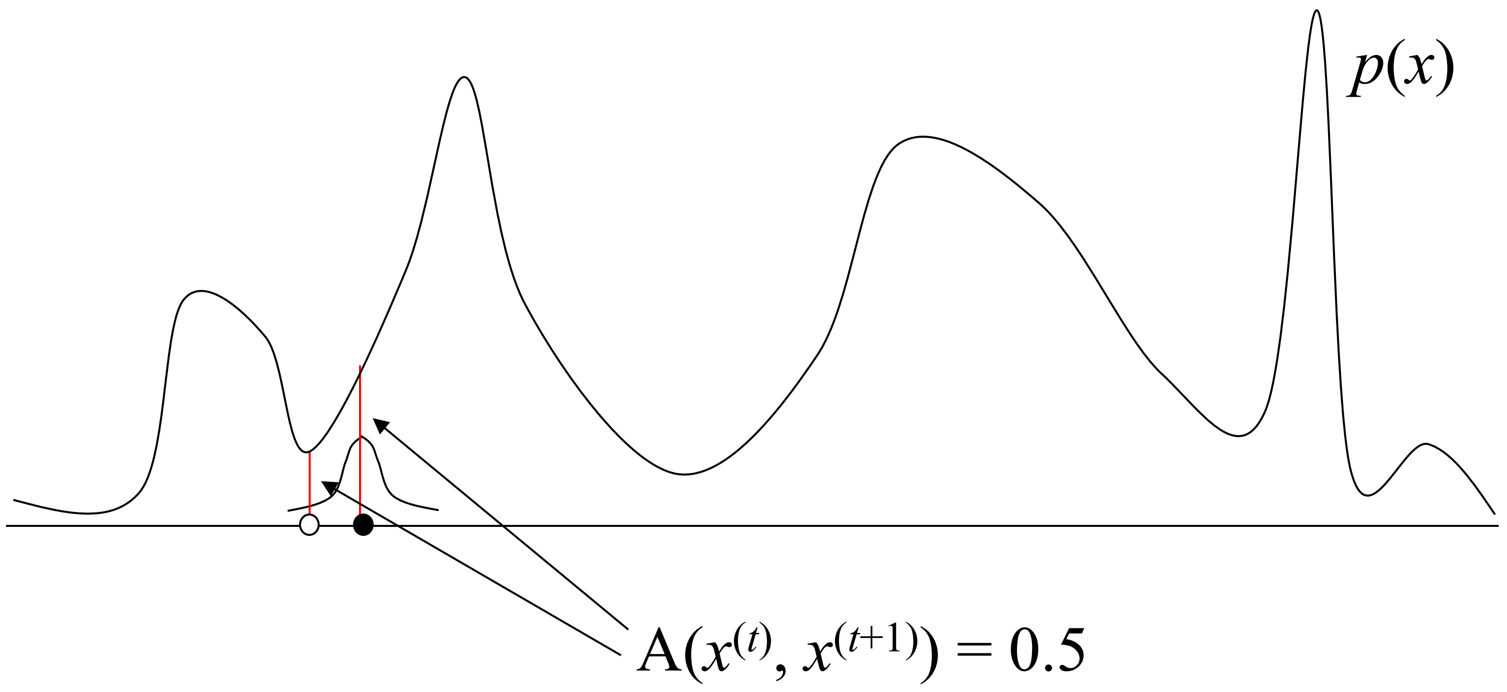
Metropolis-Hastings algorithm



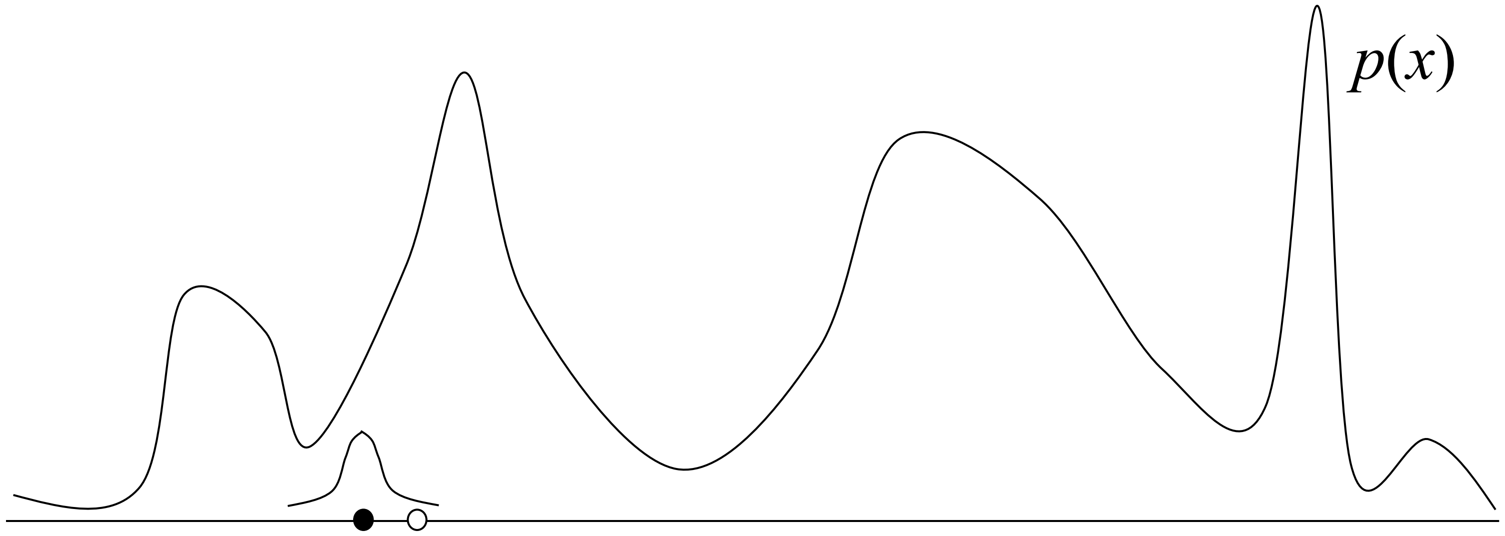
Metropolis-Hastings algorithm



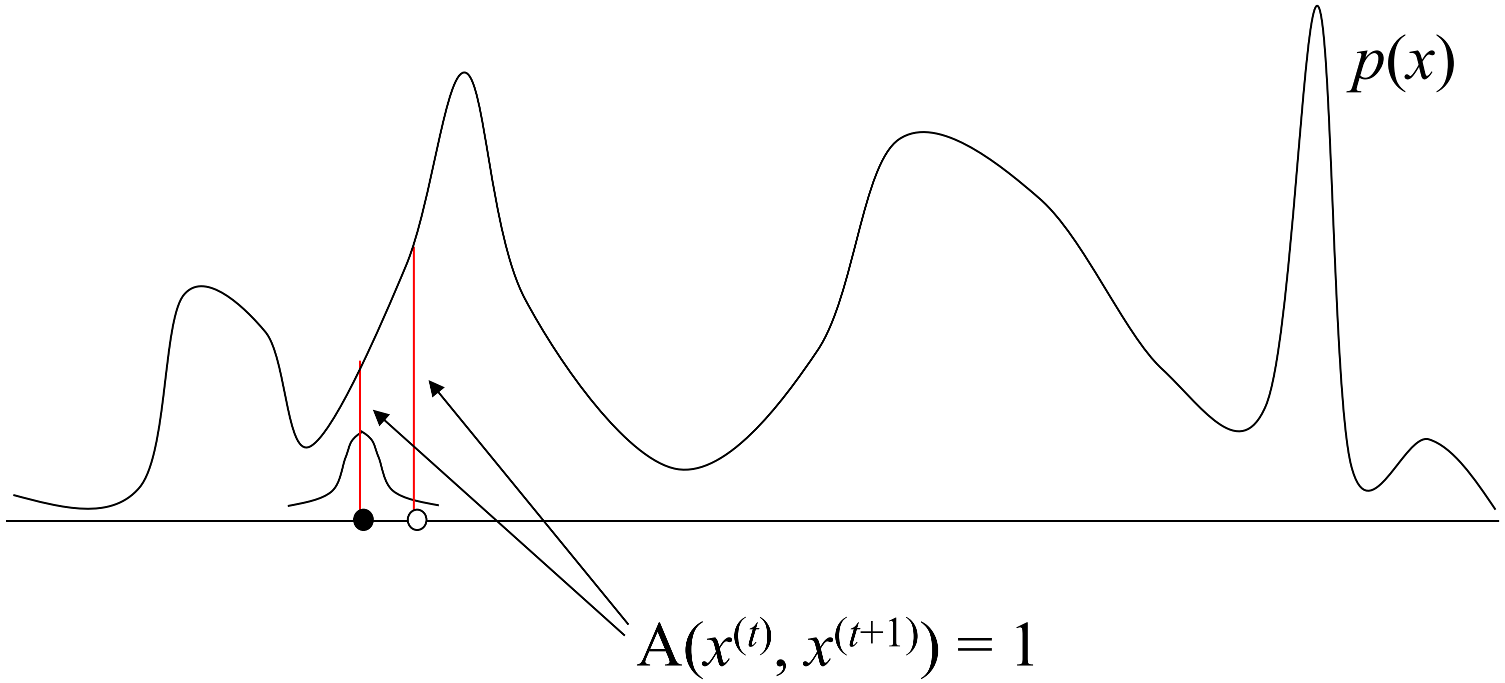
Metropolis-Hastings algorithm



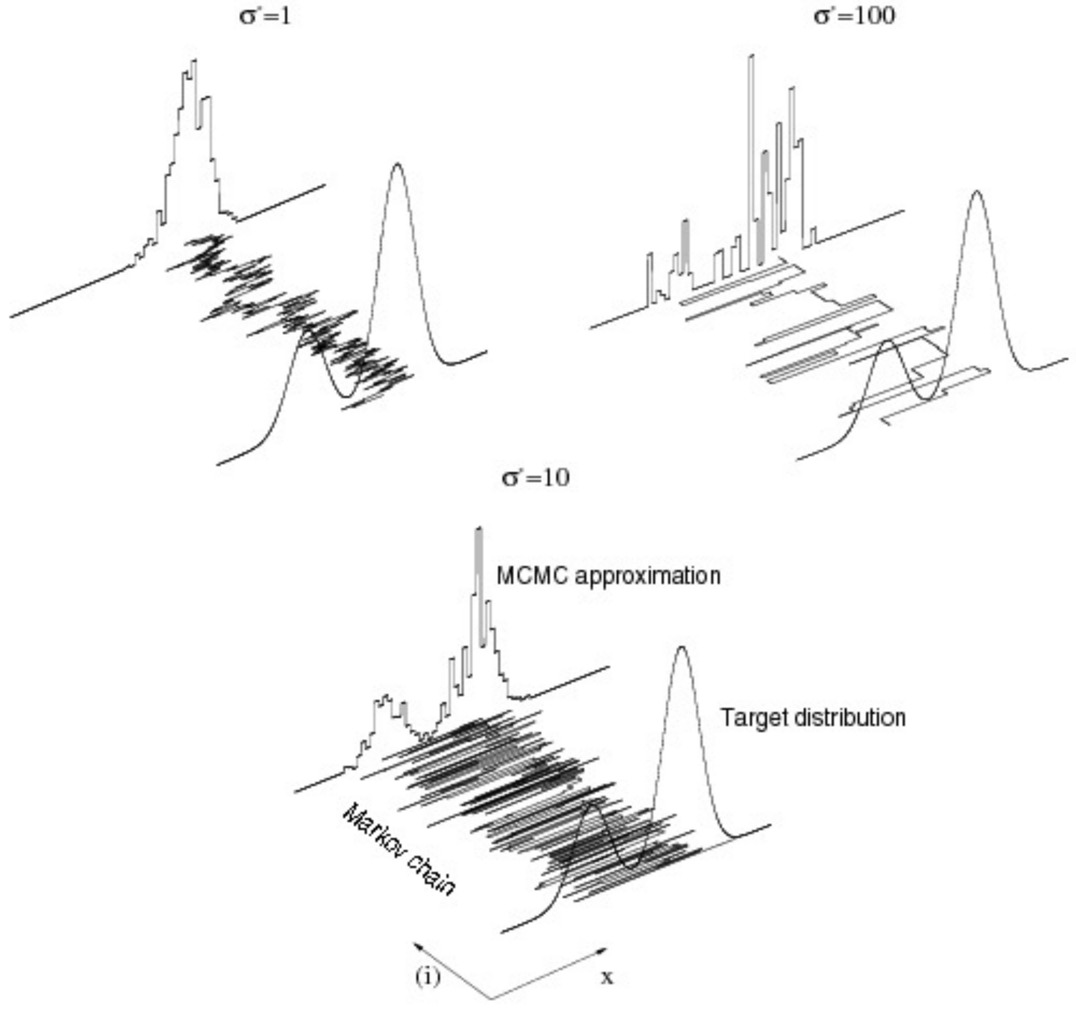
Metropolis-Hastings algorithm



Metropolis-Hastings algorithm



Examples of M-H simulations with q 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 distribution *p(x)* in only needed up to normalisation.

Gibbs sampling

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

- Component-wise proposal q:

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

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

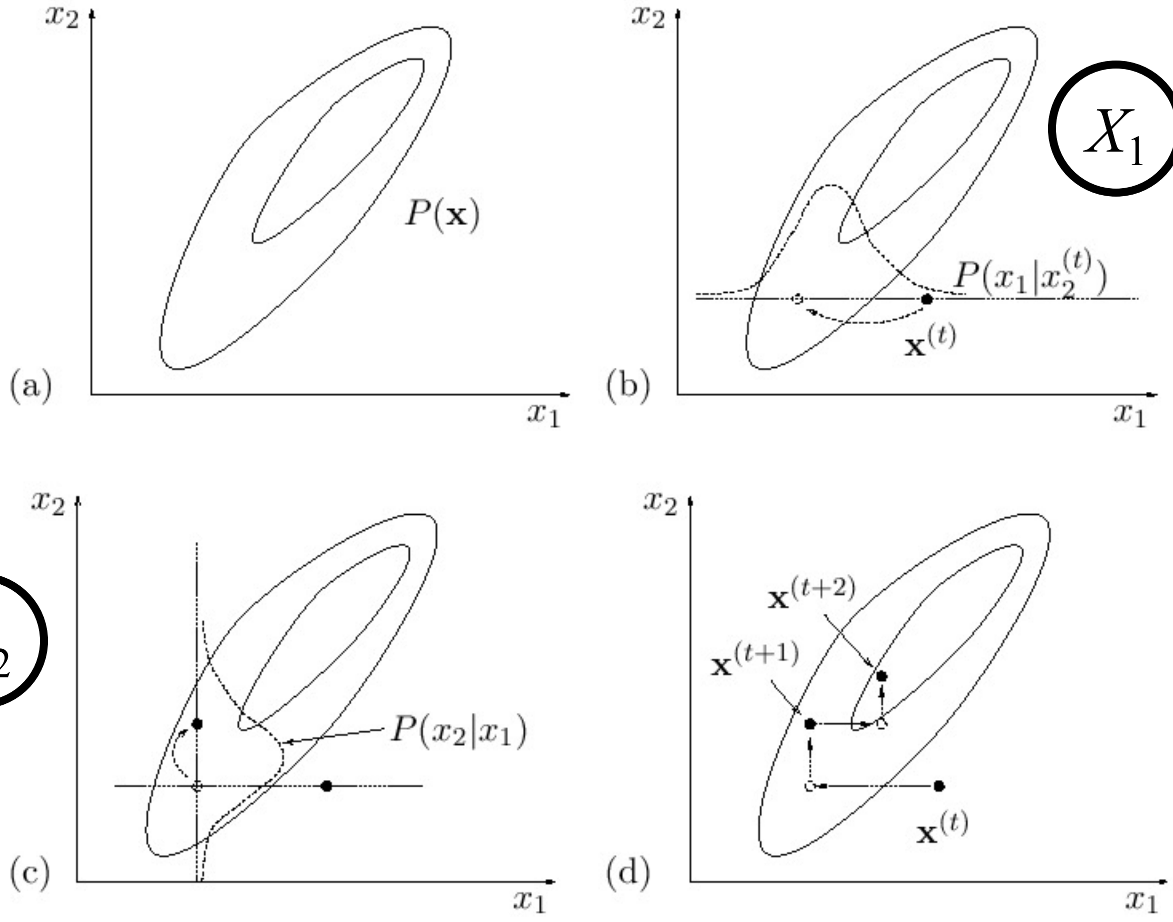
For variables $\mathbf{x} = x_1, x_2, \ldots, x_n$

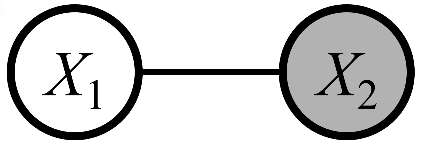
Draw $x_i^{(t+1)}$ from $P(x_i|\mathbf{x}_{-i})$

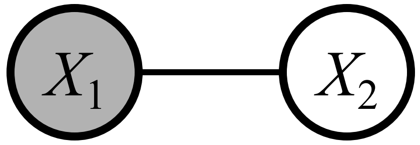
$\mathbf{x}_{-i} = x_1^{(t+1)}, x_2^{(t+1)}, \ldots, x_{i-1}^{(t+1)}, x_{i+1}^{(t)}, \ldots, x_n^{(t)}$

(this is called the *full conditional* distribution)

Gibbs sampling

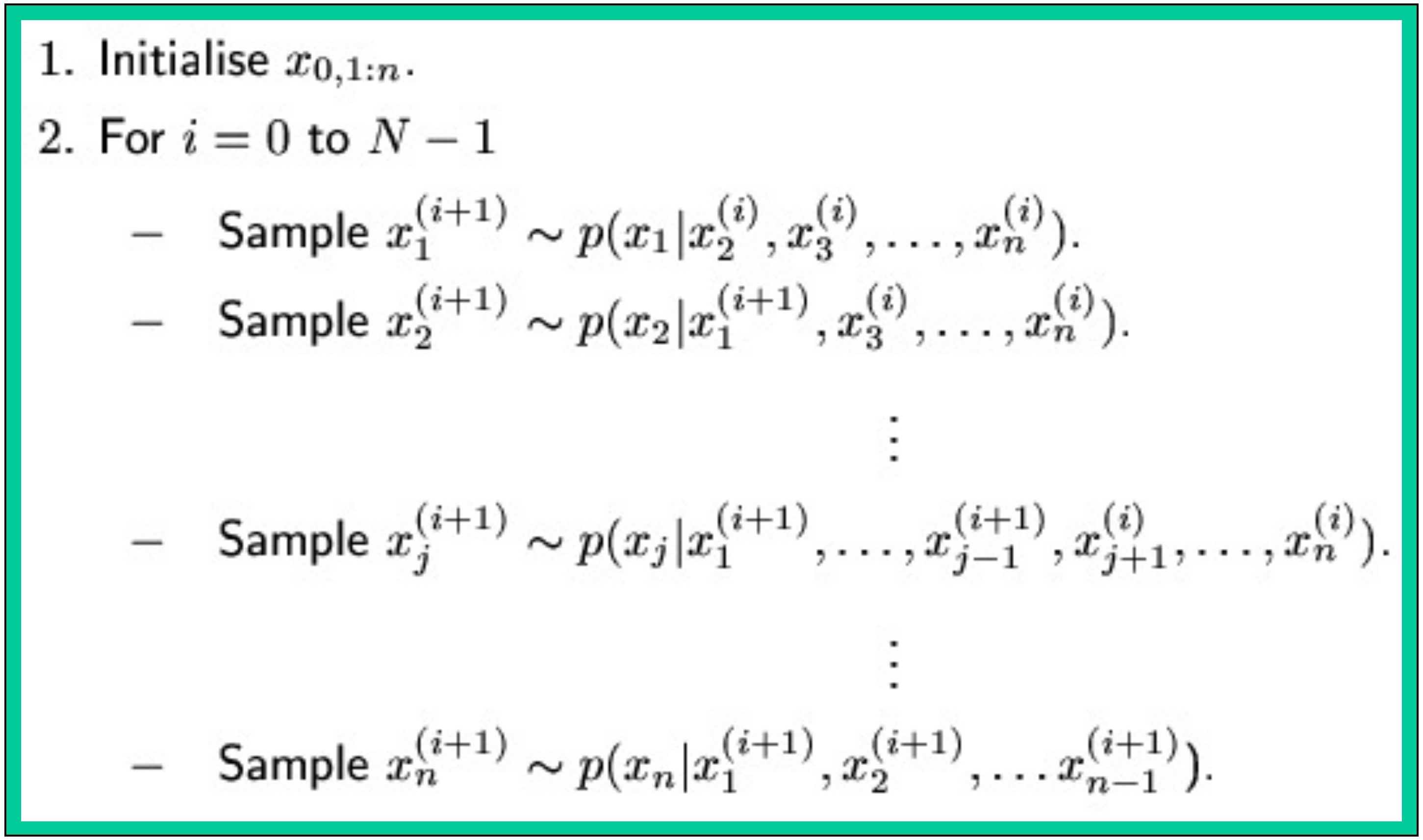






(MacKay, 2002)

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 $x_2^{(i+1)} \sim p(x_2 | x_1^{(i+1)}, x_3^{(i)}, \ldots, x_n^{(i)})$.
                              $\vdots$
    – Sample $x_j^{(i+1)} \sim p(x_j | x_1^{(i+1)}, \ldots, x_{j-1}^{(i+1)}, x_{j+1}^{(i)}, \ldots, x_n^{(i)})$.
                              $\vdots$
    – Sample $x_n^{(i+1)} \sim p(x_n | x_1^{(i+1)}, x_2^{(i+1)}, \ldots x_{n-1}^{(i+1)})$.
```

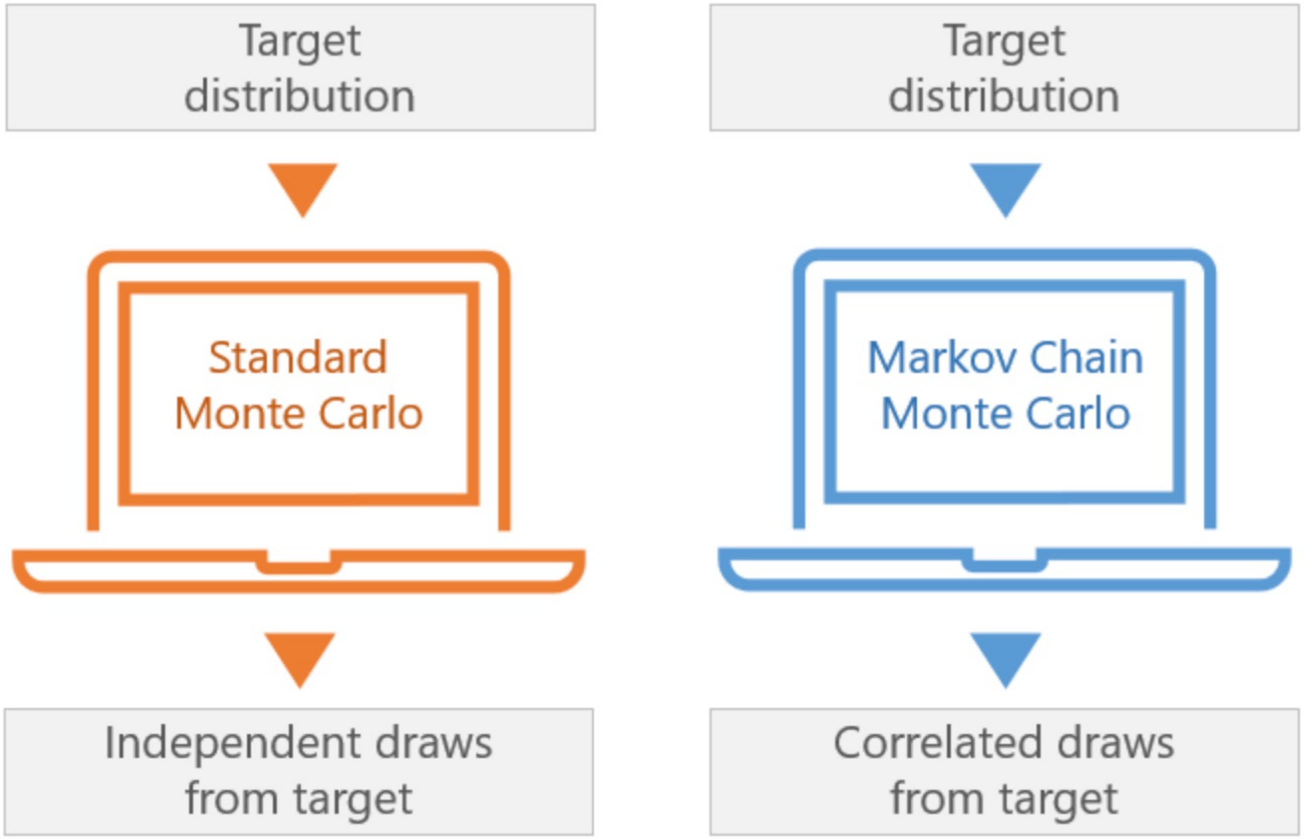
The promise of particle filters

- People need to be able to update probability distributions over large hypothesis spaces as more data become 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

The magic of MCMC Methods



The magic of MCMC Methods

- What are the implications of the lack of independence in MCMC methods?
- The accuracy of a standard MC simulation depends on the sample size: the larger the sample size is, the better the approximation.
- In the case of an MCMC simulation, we need to use the concept of effective sample size: dependent observations are equivalent to a smaller number of independent observations.

The magic of MCMC Methods

- What are the implications of the lack of independence in MCMC methods?
- The higher the correlation between adjacent observations, the smaller the effective sample size, and the less accurate the MCMC approximation is.
- For example, 1000 dependent observations could be equivalent to 100 independent observations. In this case, we say that the effective sample size is equal to 100.
- This is why in an MCMC simulation, most of the efforts are devoted to reducing the correlation as much as possible.

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!