

# CS5340

## Uncertainty Modeling in AI

### Lecture 9: Monte Carlo Inference (Sampling)

Asst. Prof. Lee Gim Hee

AY 2020/21

Semester 1

# Course Schedule

Week	Date	Topic	Remarks
1	12 Aug	Introduction to probabilistic reasoning	<b>1830hrs: MS Teams (Live Introduction)</b>
2	19 Aug	Bayesian networks (Directed graphical models)	
3	26 Aug	Markov random Fields (Undirected graphical models)	<b>1830hrs: Zoom discussions</b>
4	02 Sep	Variable elimination and belief propagation	<b>Assignment 1:</b> Belief propagation and maximal probability (15%)
5	09 Sep	Factor graph and the junction tree algorithm	
6	16 Sep	Parameter learning with complete data	<b>Assignment 1:</b> Due <b>Assignment 2:</b> Junction tree and parameter learning (15%) <b>1830hrs: Zoom discussions</b>
-	23 Sep	<b>Recess week</b>	<b>No lecture</b>
7	30 Sep	Mixture models and the EM algorithm	<b>Assignment 2:</b> Due <b>Online quiz 1 (20%)</b>
8	07 Oct	Hidden Markov Models (HMM)	<b>Assignment 3:</b> Hidden Markov model (15%)
9	14 Oct	Monte Carlo inference (Sampling)	<b>1830hrs: Zoom discussions</b>
10	21 Oct	Variational inference	<b>Assignment 3:</b> Due <b>Assignment 4:</b> MCMC Sampling (15%)
11	28 Oct	Variational Auto-Encoder and Mixture Density Networks	
12	04 Nov	Graph-cut and alpha expansion	<b>Assignment 4:</b> Due <b>1830hrs: Zoom discussions</b>
-	11 Nov	--	<b>Online quiz 2 (20%)</b>

# Acknowledgements

- A lot of slides and content of this lecture are adopted from:
  1. "An introduction to MCMC for Machine Learning", Christophe Andrieu et. al.  
<http://www.cs.bham.ac.uk/~axk/mcmc1.pdf>
  2. "Pattern Recognition and Machine Learning", Christopher Bishop, Chapter 11.
  3. <http://www.cs.cmu.edu/~epxing/Class/10708/lectures/lecture16-MC.pdf>  
<http://www.cs.cmu.edu/~epxing/Class/10708/lectures/lecture17-MCMC.pdf>, Eric Xing, CMU.
  4. "Machine Learning – A Probabilistic Perspective", Kevin Murphy, Chapter 23.
  5. "Probabilistic Graphical Models", Daphne Koller and Nir Friedman, chapter 12.

# Learning Outcomes

- Students should be able to:
  1. Explain the **Monte Carlo principle** and its justification for sampling methods.
  2. Apply **Rejection**, **Importance**, **Metropolis-Hasting**, **Metropolis** and **Gibbs** sampling methods to do maximal probability, approximate inference, and expectation.
  3. Use **Markov chain properties**, i.e. homogenous, stationary distribution, irreducibility, aperiodic, ergodic and detail balance, to show validity of MH algorithm.

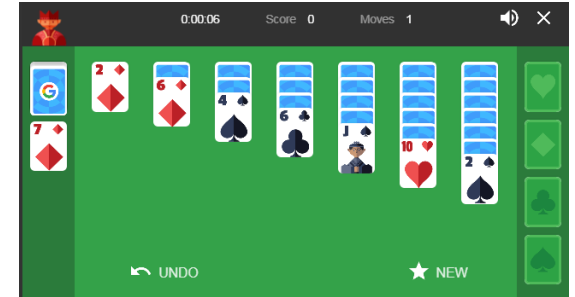
# History of Monte Carlo Sampling

Metropolis algorithm is selected as one of the **top 10 algorithms** that had the greatest influence on science and engineering in the 20<sup>th</sup> century.

[Beichl & Sullivan 2000]

# History of Monte Carlo Sampling

- Invented by Stan Ulam in 1946 when he was **playing solitaire**, while convalescing from an illness.
- Occurred to him to try to **compute the chances** that a particular solitaire laid out with 52 cards would come out successfully.
- He attempted exhaustive combinatorial calculations, but decided to lay out several solitaires **at random** and then observing and counting the number of successful plays.



Stanislaw Ulam  
1909-1984

Image source: [https://en.wikipedia.org/wiki/Stanislaw\\_Ulam](https://en.wikipedia.org/wiki/Stanislaw_Ulam)

# Idea Behind Monte Carlo Sampling

Ulam's idea of selecting a statistical sample to **approximate a hard combinatorial problem by a much simpler problem** is at the heart of modern Monte Carlo simulation.

# Pioneers of Monte Carlo Sampling



Stanislaw Ulam  
1909-1984



John von Neumann  
1903-1957



Nicholas Metropolis  
1915-1999



Marshall Rosenbluth  
1927-2003



Edward Teller  
1908-2003



Augusta H. Teller  
1909-2000



# Why Do We Need Sampling?

## Bayesian inference and learning:

Given some unknown variables  $X \in \mathcal{X}$  and data  $Y \in \mathcal{Y}$ , the following typically **intractable integration problems** are central to Bayesian statistics.

1. **Normalization.** To obtain the posterior  $p(x | y)$  given the prior  $p(x)$  and likelihood  $p(y | x)$ , the **normalizing factor** in Bayes' theorem needs to be computed

$$p(x | y) = \frac{p(y | x)p(x)}{\int_{\mathcal{X}} p(y | x')p(x') dx'}$$

 Intractable to compute in large dimensional space

# Why Do We Need Sampling?

2. **Marginalization:** Given the joint posterior of  $(X, Z) \in \mathcal{X} \times \mathcal{Z}$ , we may often be interested in the **marginal posterior**.

$$p(x | y) = \int_{\mathcal{Z}} p(x, z | y) dz$$

Intractable to compute in large dimensional space

3. **Expectation:** The objective of the analysis is often to obtain **summary statistics** of the form

$$\mathbb{E}_{p(x|y)}(f(x)) = \int_{\mathcal{X}} f(x) p(x | y) dx$$

Intractable to compute in large dimensional space

for some function of interest  $f : \mathcal{X} \rightarrow \mathbb{R}^{n_f}$  integrable with respect to  $p(x | y)$ .

# Why Do We Need Sampling?

## Optimization:

- The goal of optimization is to extract the solution that **minimizes some objective function** from a large set of feasible solutions.
- This set can be continuous and unbounded.
- In general, it is **too computationally expensive** to compare all the solutions to find out which one is optimal.

# The Monte Carlo Principle

- Draw an **i.i.d. set of samples**  $\{x^{(i)}\}_{i=1}^N$  from a target density  $p(x)$  defined on a **high-dimensional space**  $\mathcal{X}$ .
- E.g. the set of possible configurations of a system, the space on which the posterior is defined, or the combinatorial set of feasible solutions.
- These  $N$  samples can be used to **approximate the target density** with the following **empirical point-mass function**:

$$p_N(x) = \frac{1}{N} \sum_{i=1}^N \delta_{x^{(i)}}(x)$$

Delta-Dirac mass located at  $x^{(i)}$

# Weak vs Strong Law of Large Numbers

- **Weak law of large numbers:** the sample average,  $\bar{X}$  **converges in probability** towards the expected value,  $\mu$ , i.e.,

$$\bar{X}_n \xrightarrow{P} \mu \quad \text{when } n \rightarrow \infty.$$

- That is, for any positive number  $\varepsilon$ ,

$$\lim_{n \rightarrow \infty} \Pr\left(|\bar{X}_n - \mu| > \varepsilon\right) = 0.$$

where

$$\bar{X} = \frac{1}{N} \sum_n X_n \quad \text{and} \quad \mu = \int X p(X) dX$$

# Weak vs Strong Law of Large Numbers

- **Strong law of large numbers:** the sample average,  $\bar{X}$  **converges almost surely** to the expected value,  $\mu$ , i.e.,

$$\bar{X}_n \xrightarrow{\text{a.s.}} \mu \quad \text{when } n \rightarrow \infty.$$

- That is,

$$\Pr\left(\lim_{n \rightarrow \infty} \bar{X}_n = \mu\right) = 1.$$

where

$$\bar{X} = \frac{1}{N} \sum_n X_n \quad \text{and} \quad \mu = \int X p(X) dX$$

# The Monte Carlo Principle

- Consequently, we can **approximate the integrals** (or very large sums)  $I(f)$  with tractable sums  $I_N(f)$  that converge as follows:

$$\underbrace{I_N(f) = \frac{1}{N} \sum_{i=1}^N f(x^{(i)})}_{\text{Similar to } \bar{X}} \xrightarrow[N \rightarrow \infty]{a.s.} \underbrace{I(f) = \int_{\mathcal{X}} f(x)p(x) dx}_{\text{Similar to } \mu}$$

- That is, the estimate  $I_N(f)$  is unbiased and by the **strong law of large numbers**, it will almost surely converge to  $I(f)$ .

# The Monte Carlo Principle

- If the variance (in the univariate case for simplicity) of  $f(x)$  satisfies:

$$\sigma_f^2 \triangleq \mathbb{E}_{p(x)}(f^2(x)) - I^2(f) < \infty,$$

- Then the **variance of the estimator**  $I_N(f)$  is equal to:

$$\text{var}(I_N(f)) = \frac{\sigma_f^2}{N},$$



# The Monte Carlo Principle

## Proof:

$$\text{Var}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$$

$$\begin{aligned}\Rightarrow \text{Var}(f) &= \mathbb{E}_{p(x)} \left[ (f - \mathbb{E}_{p(x)}[f])^2 \right] \\ &= \mathbb{E}_{p(x)}[f^2] - \mathbb{E}_{p(x)}[f]^2 = \mathbb{E}_{p(x)}[f^2] - I(f)^2 = \sigma_f^2\end{aligned}$$

$$\text{Var}(\bar{X}) = \text{Var}\left(\frac{1}{N} \sum_n X_n\right) = \frac{1}{N^2} \sum_n \text{Var}(X_n) = \frac{\sigma^2}{N}$$


Let  $\bar{X} := I_N(f)$  and  $X := f$ , we get:

$$\text{Var}(I_N(f)) = \text{Var}\left(\frac{1}{N} \sum_n f_n\right) = \frac{1}{N^2} \sum_n \text{Var}(f_n) = \frac{\sigma_f^2}{N}$$

# The Monte Carlo Principle

- and a **central limit theorem** yields convergence in distribution of the error:

$$\sqrt{N}(I_N(f) - I(f)) \xrightarrow[N \rightarrow \infty]{} \mathcal{N}(0, \sigma_f^2) \quad .$$

Convergence in distribution  


- The  $N$  samples can also be used to obtain a **maximum of the objective function**  $p(x)$  as follows:

$$\hat{x} = \arg \max_{x^{(i)}; i=1, \dots, N} p(x^{(i)})$$

# Non-Parametric Representation

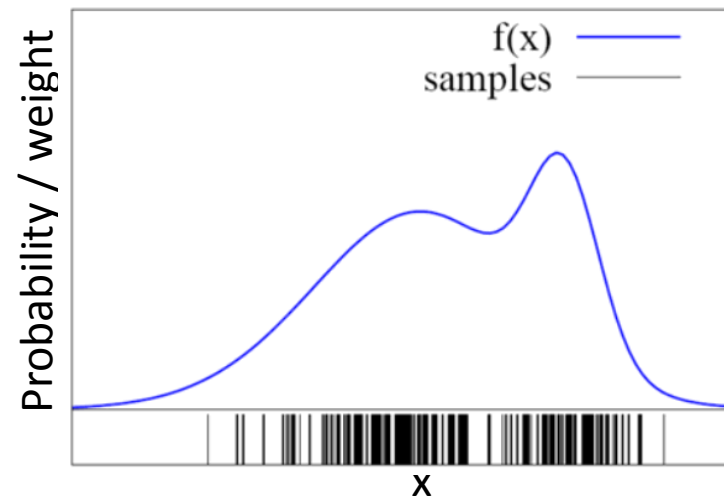
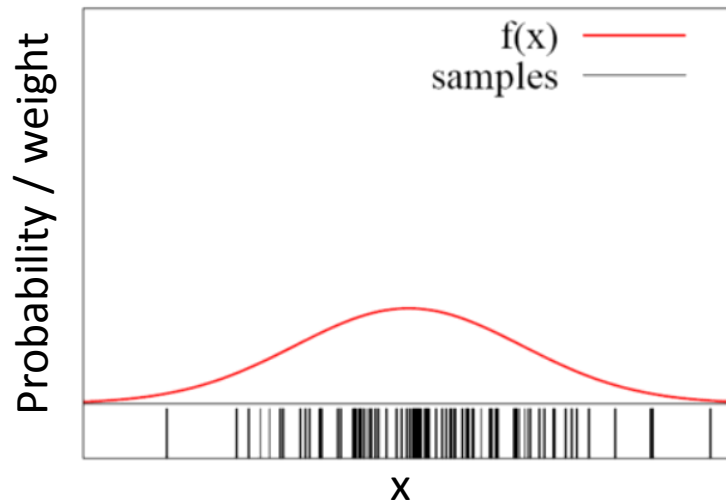
Probability distributions can be represented:

1. **Parametrically**: e.g. using mean and covariance of a Gaussian, or
2. **Non-parametrically**: using a set of *hypotheses* (samples) drawn from the distribution.

Advantage of non-parametric representation:

- **No restriction** on the *type* of distribution (e.g. can be multi-modal, non-Gaussian, etc)

# Non-Parametric Representation



The more samples are in an interval, the higher the probability of that interval.

**But:**

How to draw samples from a function/distribution?

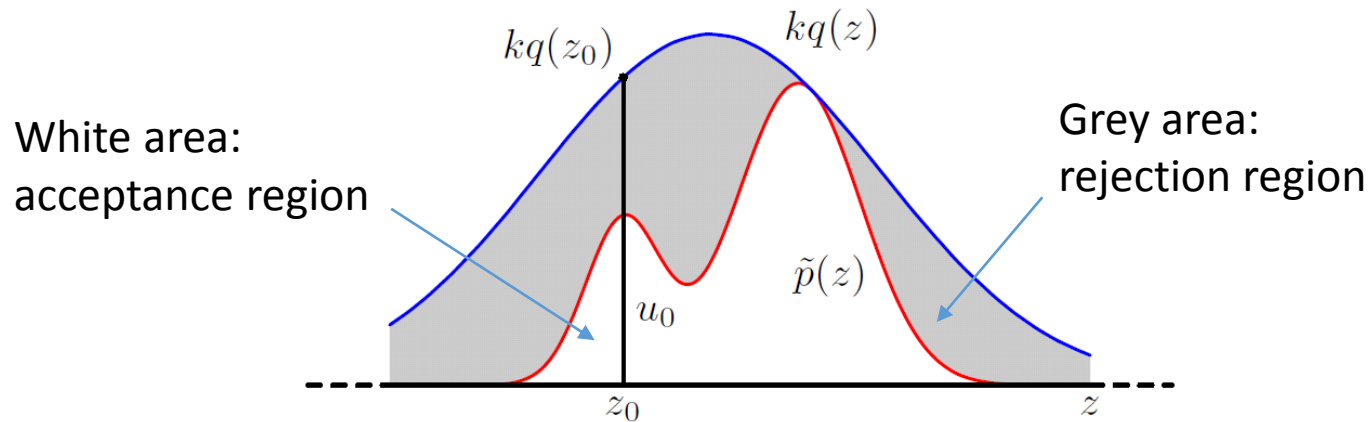
# Rejection Sampling

- Suppose we wish to sample from a **distribution**  $p(z)$ , where **direct sampling is difficult**.
- Furthermore, suppose that we are **unable to easily evaluate**  $p(z)$  due to an **unknown normalizing constant**  $Z_p$ , so that:

$$p(z) = \frac{1}{Z_p} \tilde{p}(z)$$

- Where  $\tilde{p}(z)$  can **readily be evaluated**.

# Rejection Sampling



---

## Algorithm : Rejection Sampling

---

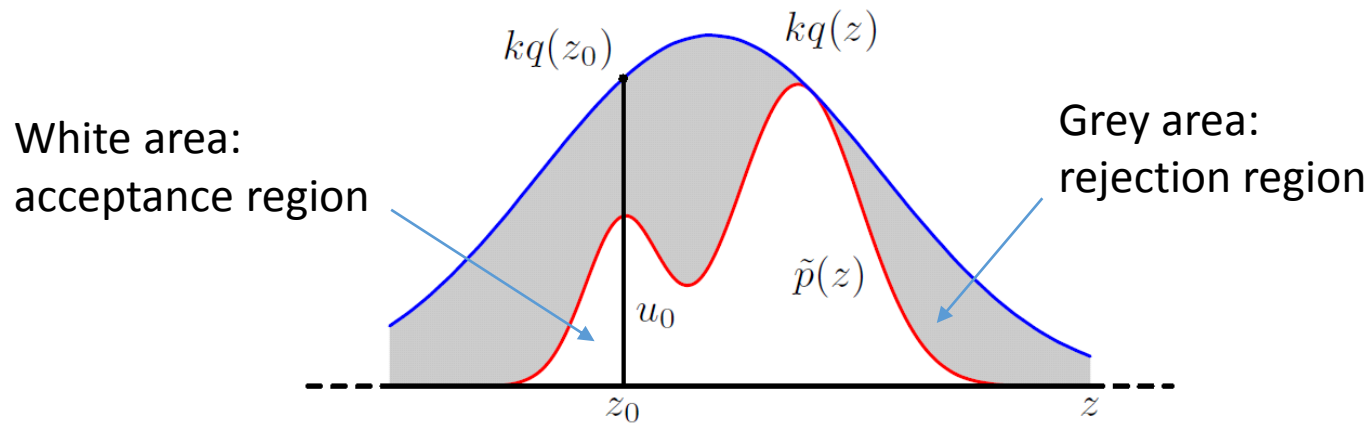
Set  $i = 1$

Repeat until  $i = N$     *// draw  $N$  samples*

Proposal distribution  $q(z)$  is an easier-to-sample distribution e.g. Gaussian!

1. Sample  $z^{(i)} \sim q(z)$  and  $u \sim U_{(0,1)}$     *// sample from proposal distribution  $q(z)$   
// sample from uniform distribution  $U_{(0,1)}$*
2. If  $u < \frac{\tilde{p}(z^{(i)})}{kq(z^{(i)})}$ , then accept  $z^{(i)}$  and increment the counter  $i$  by 1.
3. Otherwise, reject.

# Rejection Sampling



---

## Algorithm : Rejection Sampling

---

Set  $i = 1$

Repeat until  $i = N$     *// draw  $N$  samples*

Accept proposal  $z^{(i)}$  when  $u$  falls in the acceptance region.

1. Sample  $z^{(i)} \sim q(z)$  and  $u \sim U_{(0,1)}$     *// sample from proposal distribution  $q(z)$   
// sample from uniform distribution  $U_{(0,1)}$*

2. If  $u < \frac{\tilde{p}(z^{(i)})}{kq(z^{(i)})}$ , then accept  $z^{(i)}$  and increment the counter  $i$  by 1.

3. Otherwise, reject.    *// accept proposal  $z^{(i)}$  if  $u < \frac{\tilde{p}(z^{(i)})}{kq(z^{(i)})}$ ,  
// constant  $k$  is chosen such that  $\tilde{p}(z^{(i)}) \leq kq(z)$  for all values of  $z$*

# Rejection Sampling: Limitations

- It is **not always possible** to bound  $\frac{\tilde{p}(z)}{q(z)}$  with a reasonable constant  $k$  over the whole space  $\mathcal{Z}$ .
- If  $k$  is **too large**, the acceptance probability:

$$\Pr(z \text{ accepted}) = \Pr\left(u < \frac{\tilde{p}(z)}{kq(z)}\right) = \frac{1}{k},$$

will be **too small**.

- This makes the method **impractical in high dimensional space scenarios**.



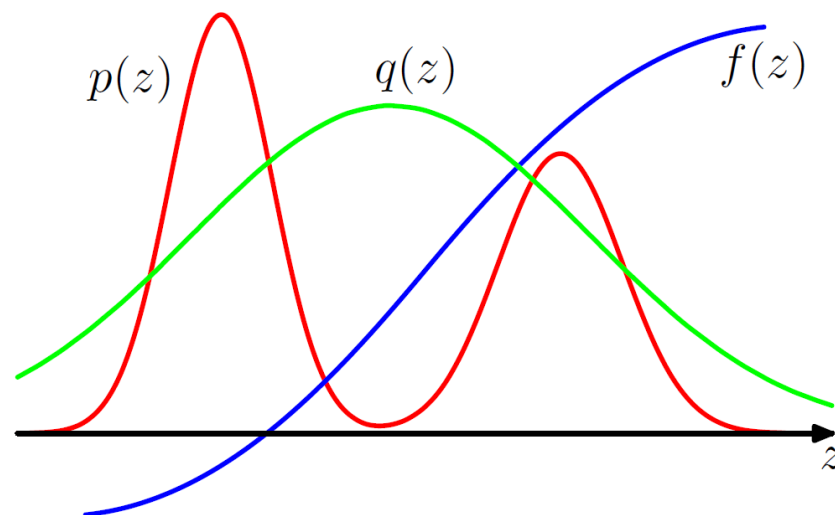
# Importance Sampling

- Given a **target distribution**  $p(z)$  which is difficult to draw samples directly.
- Importance sampling provides a framework for **approximating expectations** of a function  $f(z)$  w.r.t.  $p(z)$ .
- Samples  $\{z^{(l)}\}$  are drawn from a simpler distribution  $q(z)$ , i.e. **proposal distribution**.

# Importance Sampling

- Express expectation in the form of a finite sum over samples  $\{z^{(l)}\}$  **weighted by the ratios**  $p(z^{(l)})/q(z^{(l)})$ :

$$\begin{aligned}\mathbb{E}[f] &= \int f(\mathbf{z})p(\mathbf{z}) d\mathbf{z} \\ &= \int f(\mathbf{z}) \frac{p(\mathbf{z})}{q(\mathbf{z})} q(\mathbf{z}) d\mathbf{z} \\ &\simeq \frac{1}{L} \sum_{l=1}^L \frac{p(\mathbf{z}^{(l)})}{q(\mathbf{z}^{(l)})} f(\mathbf{z}^{(l)}).\end{aligned}$$



Recall:

$$I_N(f) = \frac{1}{N} \sum_{i=1}^N f(x^{(i)}) \xrightarrow[N \rightarrow \infty]{a.s.} I(f) = \int_{\mathcal{X}} f(x)p(x) dx$$

Image source: “Machine Learning and Pattern Recognition”, Christopher Bishop

# Importance Sampling

- The quantities  $r_l = p(z^{(l)})/q(z^{(l)})$  are known as **importance weights**.
- And they **correct the bias** introduced by sampling from the wrong distribution.
- Note that, unlike rejection sampling, **all of the samples generated are retained**.

# Importance Sampling

- Often the case that  $\tilde{p}(z)$  can be evaluated easily, but not  $p(z) = \tilde{p}(z)/Z_p$ , where  $Z_p$  is unknown.
- Let us define the **proposal distribution** in similar form, i.e.  $q(z) = \tilde{q}(z)/Z_q$ .

- We then have:

$$\begin{aligned}\mathbb{E}[f] &= \int f(\mathbf{z})p(\mathbf{z}) \, d\mathbf{z} \\ &= \frac{Z_q}{Z_p} \int f(\mathbf{z}) \frac{\tilde{p}(\mathbf{z})}{\tilde{q}(\mathbf{z})} q(\mathbf{z}) \, d\mathbf{z} \\ &\simeq \frac{Z_q}{Z_p} \frac{1}{L} \sum_{l=1}^L \tilde{r}_l f(\mathbf{z}^{(l)}) ,\end{aligned}$$

where  $\tilde{r}_l = \frac{\tilde{p}(\mathbf{z}^{(l)})}{\tilde{q}(\mathbf{z}^{(l)})}$ .

# Importance Sampling

- We can use the same sample set to evaluate the ratio  $Z_p/Z_q$  with the result:

$$\begin{aligned}\frac{Z_p}{Z_q} &= \frac{1}{Z_q} \int \tilde{p}(\mathbf{z}) \, d\mathbf{z} = \int \frac{\tilde{p}(\mathbf{z})}{\tilde{q}(\mathbf{z})} q(\mathbf{z}) \, d\mathbf{z} \\ &\simeq \frac{1}{L} \sum_{l=1}^L \tilde{r}_l \quad \text{where} \quad \tilde{r}_l = \frac{\tilde{p}(\mathbf{z}^{(l)})}{\tilde{q}(\mathbf{z}^{(l)})}.\end{aligned}$$

- And hence

$$\mathbb{E}[f] \simeq \sum_{l=1}^L w_l f(\mathbf{z}^{(l)})$$

where

$$w_l = \frac{\tilde{r}_l}{\sum_m \tilde{r}_m} = \frac{\tilde{p}(\mathbf{z}^{(l)})/q(\mathbf{z}^{(l)})}{\sum_m \tilde{p}(\mathbf{z}^{(m)})/q(\mathbf{z}^{(m)})}.$$

Important weight which is easy to compute!

# Importance Sampling

## Proof:

Substituting  $\frac{Z_p}{Z_q} = \frac{1}{L} \sum_m \tilde{r}_m$  into  $\mathbb{E}[f] = \frac{Z_q}{Z_p} \frac{1}{L} \sum_l \tilde{r}_l f(z^{(l)})$ ,  
we get:

$$\begin{aligned}\mathbb{E}[f] &= \frac{L}{\sum_m \tilde{r}_m} \frac{1}{L} \sum_l \tilde{r}_l f(z^{(l)}) \\ &= \sum_l \frac{\tilde{r}_l}{\sum_m \tilde{r}_m} f(z^{(l)}) \\ &= \sum_l w_l f(z^{(l)})\end{aligned}$$

□

# Importance Sampling: Limitations

- Success of the importance sampling approach depends crucially on **how well  $q(z)$  matches  $p(z)$** .
- A strongly varying  $p(z)f(z)$  has a significant proportion of its mass **concentrated over relatively small regions** of  $z$  space.
- Set of importance weights  $\{r_l\}$  may be **dominated by a few weights** having large values, with the remaining weights being relatively insignificant.

# Importance Sampling: Example on a Bayesian Network

$$p(x_1)$$

$X_1 = 0$	$X_1 = 1$
0.6	0.4

$$p(x_2)$$

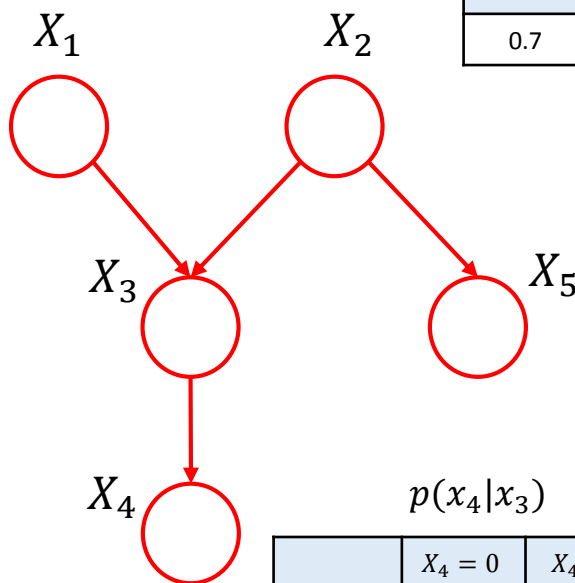
$X_2 = 0$	$X_2 = 1$
0.7	0.3

$$p(x_3|x_1, x_2)$$

	$X_3 = 0$	$X_3 = 1$	$X_3 = 2$
$X_1 = 0, X_2 = 0$	0.3	0.4	0.3
$X_1 = 0, X_2 = 1$	0.05	0.25	0.7
$X_1 = 1, X_2 = 0$	0.9	0.08	0.02
$X_1 = 1, X_2 = 1$	0.5	0.3	0.2

$$p(x_5|x_2)$$

	$X_5 = 0$	$X_5 = 1$
$X_2 = 0$	0.95	0.05
$X_2 = 1$	0.2	0.8



$$p(x_4|x_3)$$

	$X_4 = 0$	$X_4 = 1$
$X_3 = 0$	0.1	0.9
$X_3 = 1$	0.4	0.6
$X_3 = 2$	0.99	0.01

How do we compute  
 $p(x_1, x_4, x_5 \mid x_2 = 1, x_3 = 1)$ ?

$X_1$ : Difficulty,  $X_2$ : Intelligence,  $X_3$ : Grade,  $X_4$ : Letter,  $X_5$ : SAT score





# Importance Sampling: Example on a Bayesian Network

- How do we compute  $p(x_1, x_4, x_5 | x_2 = 1, x_3 = 1)$ ?

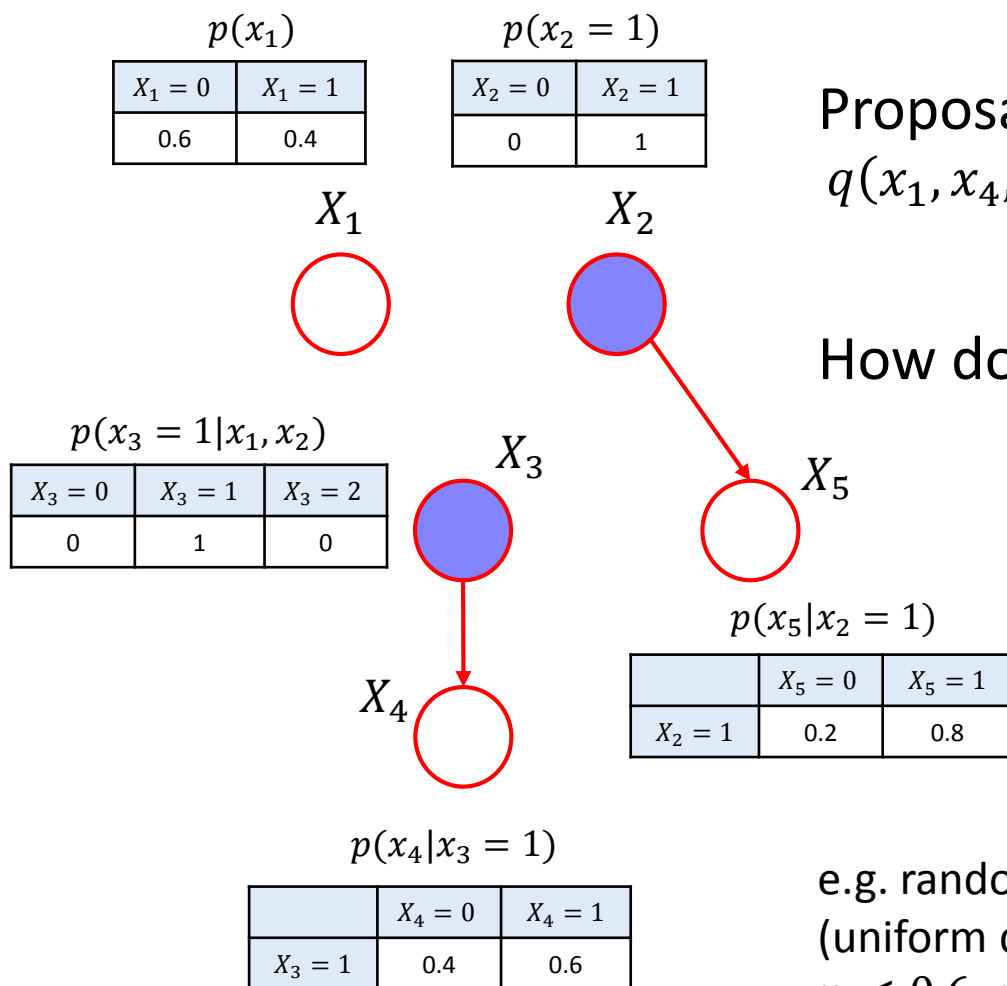
Importance Sampling!!!

$$\begin{aligned} p(x_1, x_4, x_5 | x_2 = 1, x_3 = 1) &= \frac{p(x_1, x_4, x_5, x_2 = 1, x_3 = 1)}{p(x_2 = 1, x_3 = 1)} \\ &= \frac{p(x_F, x_E)}{p(x_E)} \\ &= \frac{1}{Z_p} \tilde{p}(x) \end{aligned}$$

We don't want to evaluate  $Z_p$    Target distribution

- What should we use as the **proposal distribution**  $q(x)$ ?

# Importance Sampling: Example on a Bayesian Network



Proposal distribution:

$$q(x_1, x_4, x_5) = p(x_1)p(x_4|x_3 = 1)p(x_5|x_2 = 1)$$

How do we sample from  $q(x_1, x_4, x_5)$ ?

$$x_1 \sim p(x_1)$$

$$x_4 \sim p(x_4|x_3 = 1)$$

$$x_5 \sim p(x_5|x_2 = 1)$$

e.g. randomly generate a number within  $[0,1]$  (uniform distribution), i.e.  $n = \text{rand}$ ;  $x_1 = 0$  if  $n < 0.6$ ,  $x_1 = 1$  otherwise.

# Importance Sampling: Example on a Bayesian Network

- For each sample  $x^{(l)}$ , we **evaluate the weight** as:

$$w_l = \frac{\tilde{r}_l}{\sum_m \tilde{r}_m} = \frac{\tilde{p}(x^{(l)})/q(x^{(l)})}{\sum_m \tilde{p}(x^{(m)})/q(x^{(m)})}.$$

- Example:

$x^{(l)}: \{x_1 = 0, x_4 = 1, x_5 = 1\}$  obtained from sampling, we have

$$\begin{aligned}\tilde{p}(x^{(l)}) &= p(x_1 = 0, x_4 = 1, x_5 = 1, x_2 = 1, x_3 = 1) \\ &= p(x_1 = 0)p(x_2 = 1)p(x_3 = 1|x_2 = 1, x_1 = 0) \\ &\quad p(x_4 = 1 | x_3 = 1)p(x_5 = 1 | x_2 = 1) \\ &= (0.6)(0.3)(0.08)(0.6)(0.8) \\ &= 0.006912\end{aligned}$$

# Importance Sampling: Example on a Bayesian Network

- For each sample  $x^{(l)}$ , we **evaluate the weight** as:

$$w_l = \frac{\tilde{r}_l}{\sum_m \tilde{r}_m} = \frac{\tilde{p}(x^{(l)})/q(x^{(l)})}{\sum_m \tilde{p}(x^{(m)})/q(x^{(m)})}.$$

- Example:

$x^{(l)}: \{x_1 = 0, x_4 = 1, x_5 = 1\}$  obtained from sampling, we have

$$\begin{aligned} q(x^{(l)}) &= p(x_1 = 0)p(x_4 = 1|x_3 = 1)p(x_5 = 1|x_2 = 1) \\ &= (0.6)(0.6)(0.8) = 0.288 \end{aligned}$$

$$\Rightarrow \frac{\tilde{p}(x^{(l)})}{q(x^{(l)})} = \frac{0.006912}{0.288} = 0.024$$

- Finally, **denominator** (hence each weight  $w_l$ ) can be computed from all  $M$  samples.

# Importance Sampling:

## Example on a Bayesian Network

- We can compute  $p(x_1, x_4, x_5 | x_2 = 1, x_3 = 1)$  from all the weights and samples:

Sum of all weights from samples at  
 $\{x_1 = 0, x_4 = 0, x_5 = 0\}$

$$p(x_1 = 0, x_4 = 0, x_5 = 0 | x_2 = 1, x_3 = 1) = \frac{\sum_m w_m \delta(x^{(m)} = \{x_1 = 0, x_4 = 0, x_5 = 0\})}{\sum_m w_m}$$

normalizer: ensure probability sums to 1

⋮

$$p(x_1 = 1, x_4 = 1, x_5 = 1 | x_2 = 1, x_3 = 1) = \frac{\sum_m w_m \delta(x^{(m)} = \{x_1 = 1, x_4 = 1, x_5 = 1\})}{\sum_m w_m}$$

- In summary, we get:

$$p(x_F | x_E) = \frac{\sum_m w_m \delta(x^{(m)})}{\sum_m w_m}$$

# Sampling and the EM Algorithm

- Sampling methods can be used to **approximate the E step** of the EM algorithm for models in which the E step **cannot be performed analytically**.

$$Q(\theta, \theta^{\text{old}}) = \int \underbrace{p(\mathbf{Z}|\mathbf{X}, \theta^{\text{old}})} \ln p(\mathbf{Z}, \mathbf{X}|\theta) d\mathbf{Z}$$

Cannot be computed analytically!

# Sampling and the EM Algorithm

- Approximate integral by a **finite sum over samples**  $\{Z^l\}$ , which are drawn from the current estimate for  $p(Z | X, \theta^{old})$ , so that:

$$Q(\theta, \theta^{old}) \simeq \frac{1}{L} \sum_{l=1}^L \ln p(\mathbf{Z}^{(l)}, \mathbf{X} | \theta)$$

- The  $Q$  function is then **optimized in the usual way** in the M step.
- This procedure is called the **Monte Carlo EM algorithm**.

# Markov Chain Monte Carlo (MCMC)

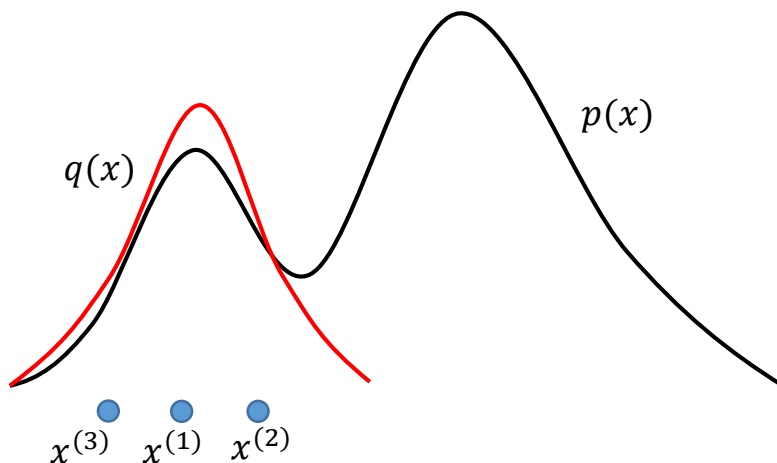
- MCMC is a strategy for generating samples  $x^{(i)}$  while exploring the state space  $\mathcal{X}$  using a **Markov chain**.
- The chain is constructed to **spend more time in the most important regions**.
- In particular, it is constructed so that the samples  $x^{(i)}$  **mimic samples drawn from the target distribution  $p(x)$** .



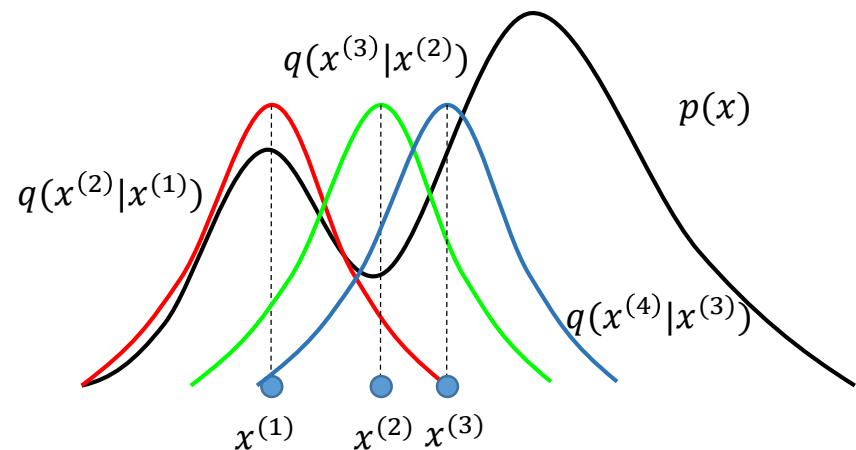
# Markov Chain Monte Carlo (MCMC)

- MCMC algorithms feature **adaptive proposals**:
  - Instead of  $q(x')$ , we use  $q(x'|x)$  where  $x'$  is the new state being sampled, and  $x$  is the previous sample.
  - As  $x$  changes,  $q(x'|x)$  can also change (as a function of  $x'$ ).

**Importance sampling**  
with **bad proposal**  $q(x)$



**MCMC with adaptive proposal**  $q(x'|x)$



# MCMC: Metropolis-Hasting Algorithm

---

## Algorithm : Metropolis-Hasting

---

1. Initialize  $x^{(0)}$
  2. For  $i = 0$  to  $N - 1$
  3.     Sample  $u \sim \mathcal{U}_{[0,1]}$      // draw acceptance threshold
  4.     Sample  $x' \sim q(x' | x^{(i)})$      // draw from proposal
  5.     If  $u < \mathcal{A}(x', x^{(i)}) = \min \left\{ 1, \frac{\tilde{p}(x')q(x^{(i)} | x')}{\tilde{p}(x^{(i)})q(x' | x^{(i)})} \right\}$      // acceptance probability
  6.          $x^{(i+1)} = x'$      // new sample is accepted
  7.     else
  8.          $x^{(i+1)} = x^{(i)}$      // new sample is rejected  
       // we create a duplicate of the previous sample
-

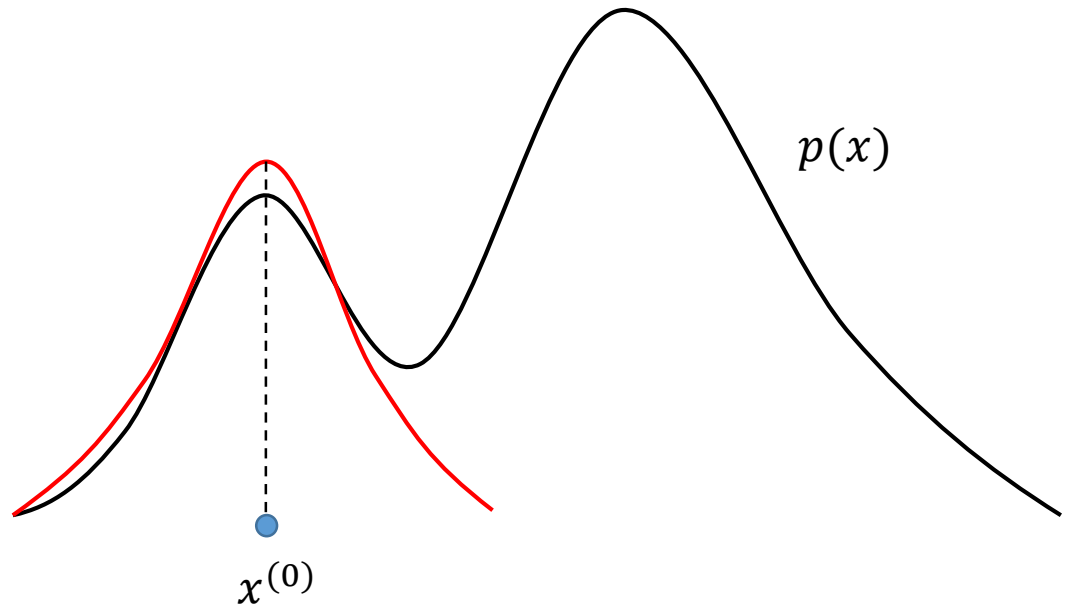
# MCMC: Metropolis-Hasting Algorithm

## Example:

- Our goal is to sample from a bimodal distribution  $p(x)$ .
- Let  $q(x'|x)$  be a Gaussian centered on  $x$ .

Initialize  $x^{(0)}$

$$\mathcal{A}(x', x^{(i)}) = \min \left\{ 1, \frac{\tilde{p}(x')q(x^{(i)}|x')}{\tilde{p}(x^{(i)})q(x'|x^{(i)})} \right\}$$



# MCMC: Metropolis-Hasting Algorithm

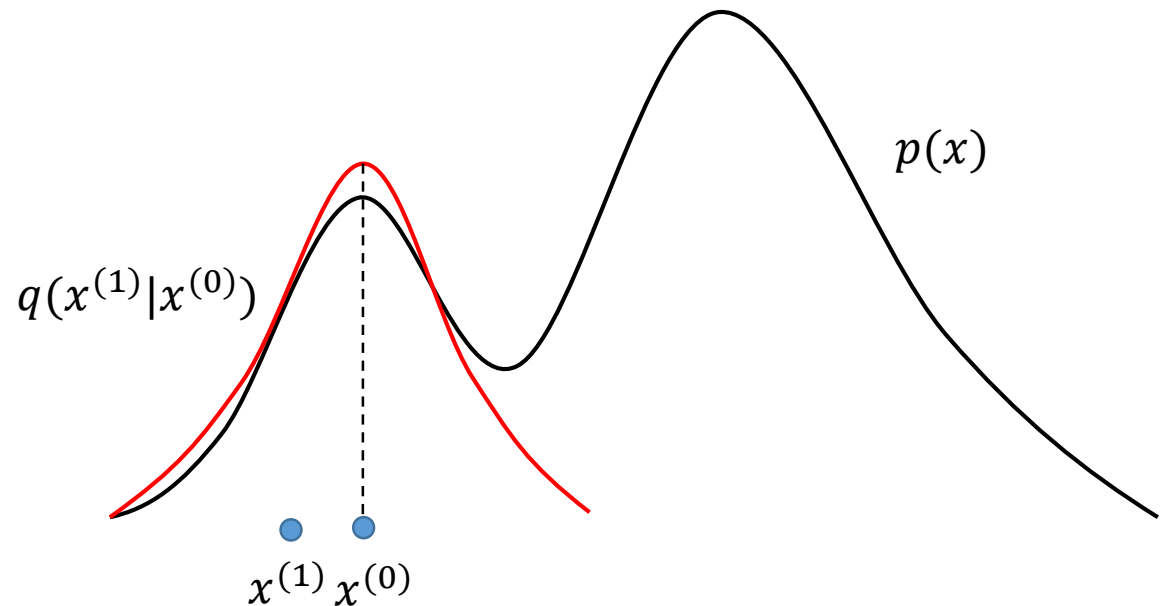
## Example:

- Our goal is to sample from a bimodal distribution  $p(x)$ .
- Let  $q(x'|x)$  be a Gaussian centered on  $x$ .

Initialize  $x^{(0)}$

Draw, accept  $x^{(1)}$

$$\mathcal{A}(x', x^{(i)}) = \min \left\{ 1, \frac{\tilde{p}(x')q(x^{(i)}|x')}{\tilde{p}(x^{(i)})q(x'|x^{(i)})} \right\}$$



# MCMC: Metropolis-Hasting Algorithm

## Example:

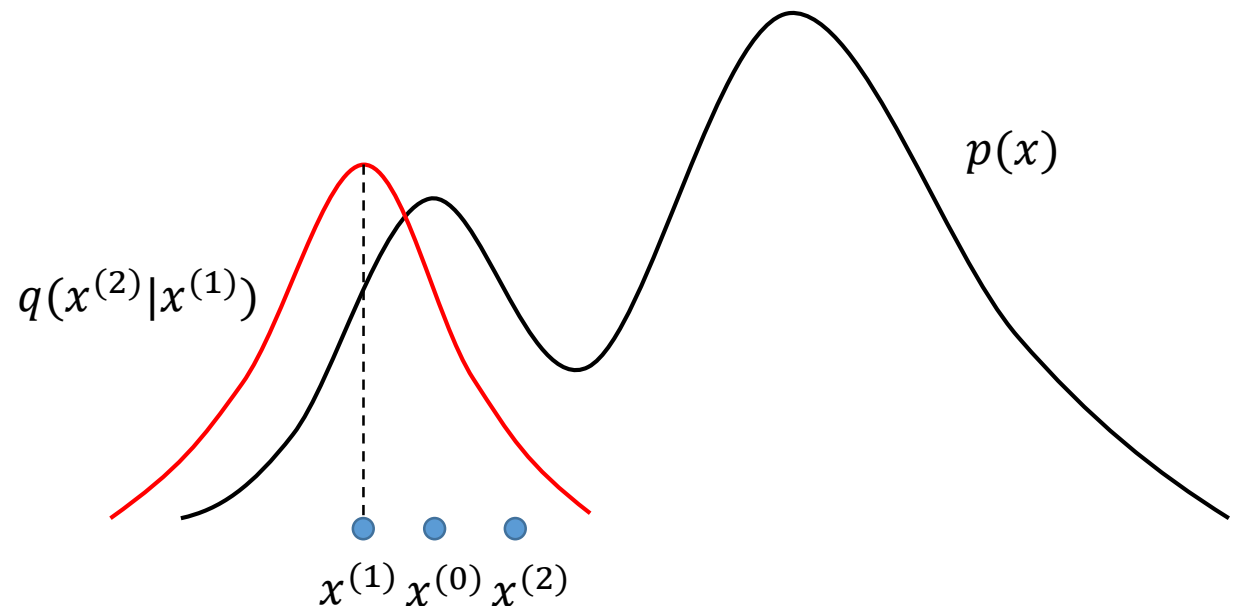
- Our goal is to sample from a bimodal distribution  $p(x)$ .
- Let  $q(x'|x)$  be a Gaussian centered on  $x$ .

Initialize  $x^{(0)}$

Draw, accept  $x^{(1)}$

Draw, accept  $x^{(2)}$

$$\mathcal{A}(x', x^{(i)}) = \min \left\{ 1, \frac{\tilde{p}(x')q(x^{(i)}|x')}{\tilde{p}(x^{(i)})q(x'|x^{(i)})} \right\}$$



# MCMC: Metropolis-Hasting Algorithm

## Example:

- Our goal is to sample from a bimodal distribution  $p(x)$ .
- Let  $q(x'|x)$  be a Gaussian centered on  $x$ .

Initialize  $x^{(0)}$

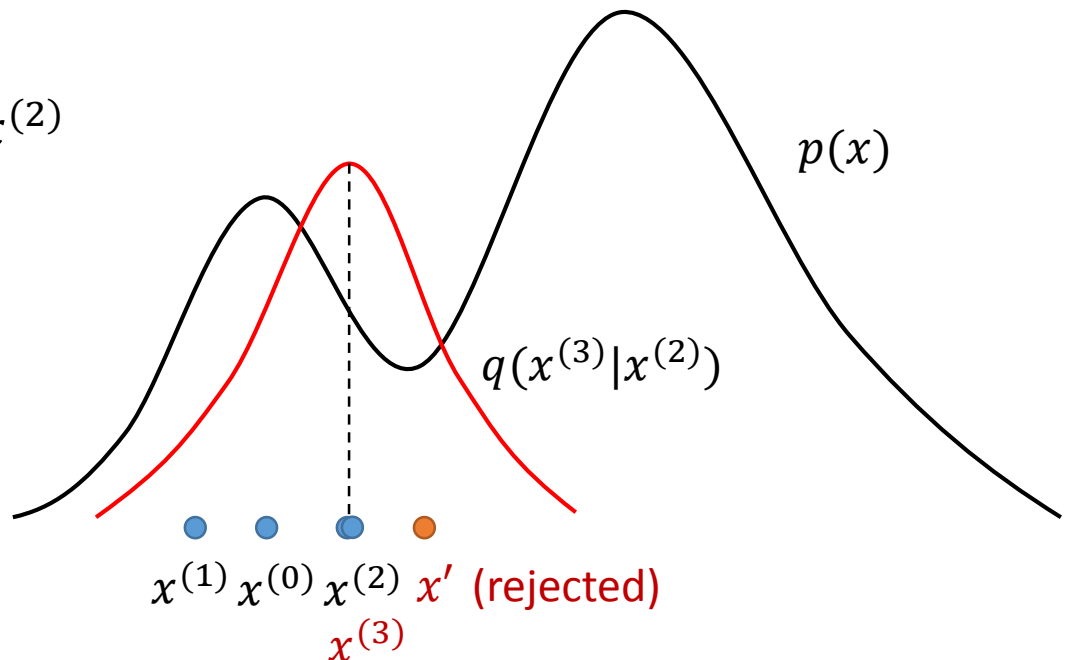
Draw, accept  $x^{(1)}$

Draw, accept  $x^{(2)}$

Draw but reject; set  $x^{(3)} = x^{(2)}$

Reject because  $\frac{\tilde{p}(x')}{q(x'|x^{(2)})} < 1$  and  $\frac{\tilde{p}(x^{(2)})}{q(x^{(2)}|x')} > 1$ , hence  $\mathcal{A}(x', x^{(2)})$  is close to zero!

$$\mathcal{A}(x', x^{(i)}) = \min \left\{ 1, \frac{\tilde{p}(x')q(x^{(i)}|x')}{\tilde{p}(x^{(i)})q(x'|x^{(i)})} \right\}$$



# MCMC: Metropolis-Hasting Algorithm

## Example:

- Our goal is to sample from a bimodal distribution  $p(x)$ .
- Let  $q(x'|x)$  be a Gaussian centered on  $x$ .

Initialize  $x^{(0)}$

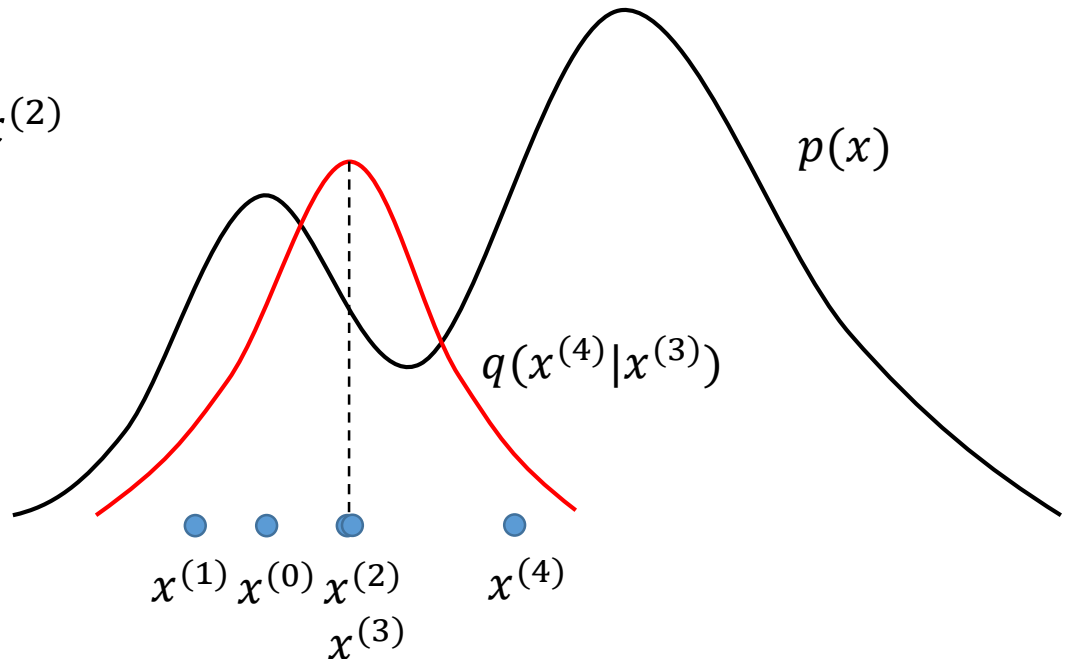
Draw, accept  $x^{(1)}$

Draw, accept  $x^{(2)}$

Draw but reject; set  $x^{(3)} = x^{(2)}$

Draw, accept  $x^{(4)}$

$$\mathcal{A}(x', x^{(i)}) = \min \left\{ 1, \frac{\tilde{p}(x')q(x^{(i)}|x')}{\tilde{p}(x^{(i)})q(x'|x^{(i)})} \right\}$$



# MCMC: Metropolis-Hasting Algorithm

## Example:

- Our goal is to sample from a bimodal distribution  $p(x)$ .
- Let  $q(x'|x)$  be a Gaussian centered on  $x$ .

Initialize  $x^{(0)}$

Draw, accept  $x^{(1)}$

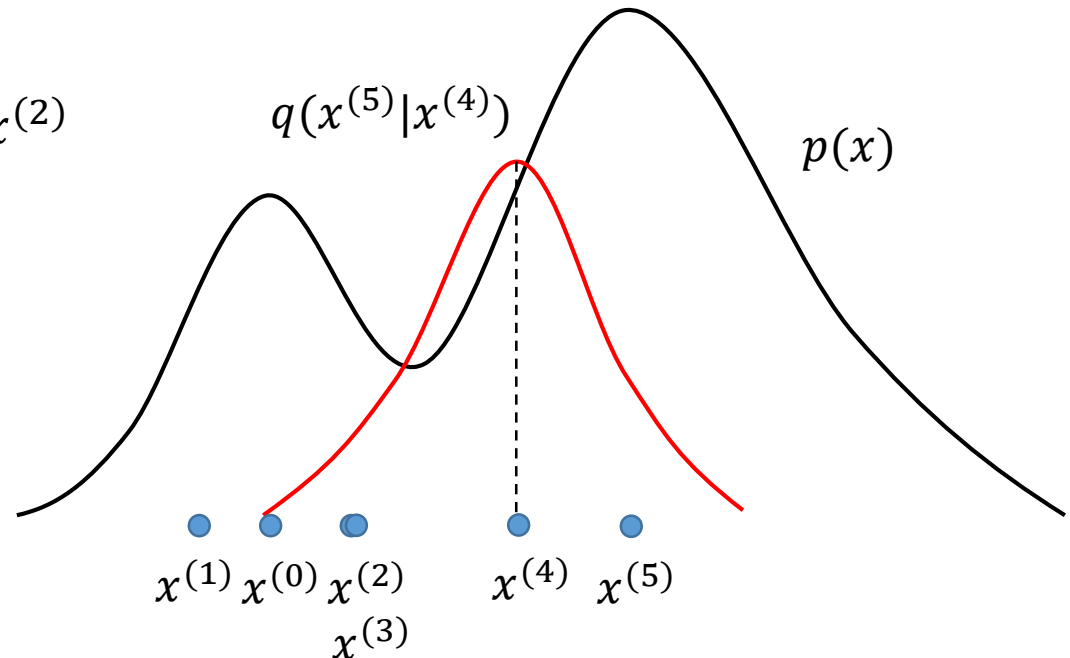
Draw, accept  $x^{(2)}$

Draw but reject; set  $x^{(3)} = x^{(2)}$

Draw, accept  $x^{(4)}$

Draw, accept  $x^{(5)}$

$$\mathcal{A}(x', x^{(i)}) = \min \left\{ 1, \frac{\tilde{p}(x')q(x^{(i)}|x')}{\tilde{p}(x^{(i)})q(x'|x^{(i)})} \right\}$$



The adaptive proposal  $q(x'|x^{(i)})$  allows us to sample both modes of  $p(x)$ !



# Burn-In Period

- The **initial samples** may follow a very different distribution, especially if the starting point is in a region of low density.
- As a result, a **burn-in period** is typically necessary, where an initial number of samples (e.g. the first 1,000 or so) are thrown away.

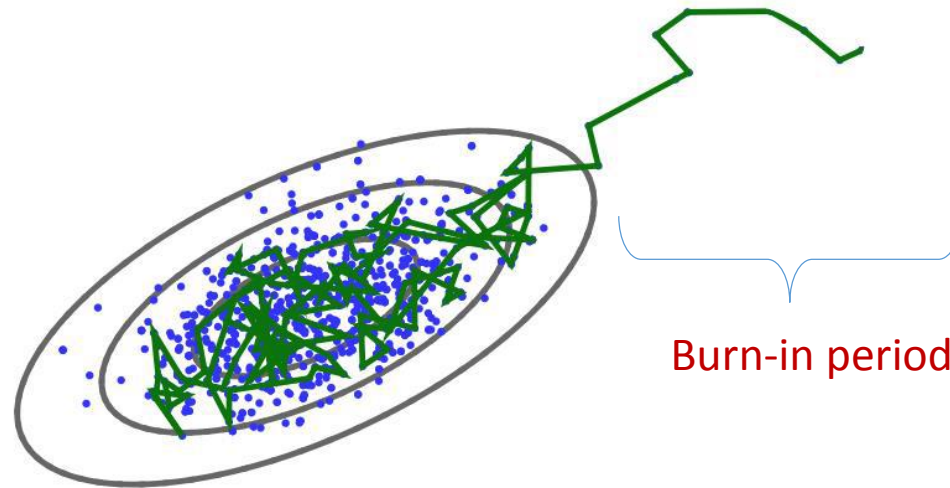


Image source: <http://wiki.ubc.ca/Course:CPSC522/MCMC>

What is the connection between Markov chains and MCMC?

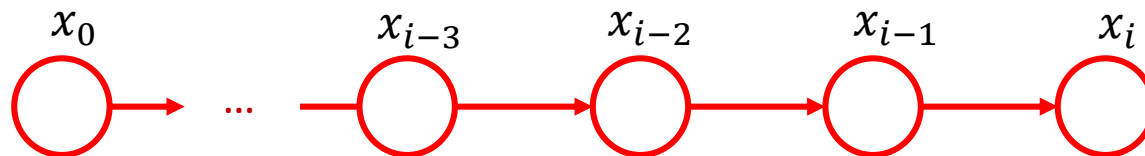
Why does the Metropolis-Hasting algorithm work?

# What is a Markov Chain?

- Intuitive to introduce Markov chains on **finite state spaces**, where  $x^{(i)}$  can only take  $s$  **discrete values**  $x^{(i)} \in \mathcal{X} = \{x_1, x_2, \dots, x_s\}$ .
- The stochastic process  $x^{(i)}$  is called a **Markov chain** if:

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

←  $s \times s$  matrix



- Current state  $x^{(i)}$  is **conditionally independent** of all previous states given most recent state  $x^{(i-1)}$ .

# Properties of a Markov Chain

## 1. Homogeneous chain:

- Chain is homogeneous if  $T \triangleq T(x^{(i)} \mid x^{(i-1)})$  **remains invariant**  $\forall i$ , with  $\sum_{x^{(i)}} T(x^{(i)} \mid x^{(i-1)}) = 1$  for any  $i$ .

Sum of each row in  $T$  equals to 1

- That is, the evolution of the chain in a space  $\mathcal{X}$  depends solely on the current state of the chain and a **fixed transition matrix**.

# Properties of a Markov Chain

## 2. Stationary and limiting distributions:

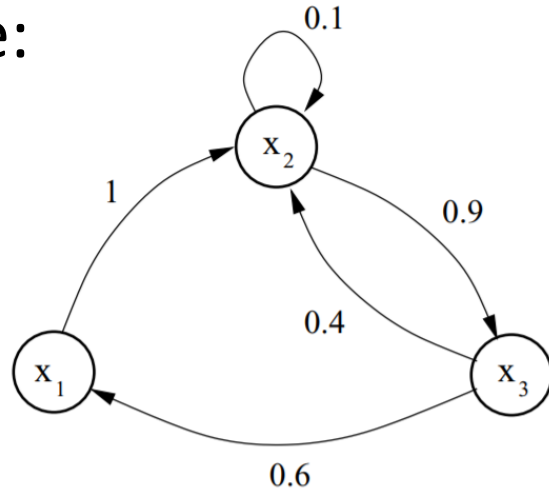
- A probability vector  $\pi = p(x)$  defined on  $\mathcal{X}$  is a **stationary (invariant) distribution** (w.r.t  $T$ ) if

$$\pi T = \pi.$$

- A **limiting distribution**  $\pi$ , is a distribution over the states such that whatever the starting distribution  $\pi_0$ , the Markov chain converges to  $\pi$ .

# Properties of a Markov Chain

Example:



Transition graph for the Markov chain example with  $\mathcal{X} = \{x_1, x_2, x_3\}$ .

$$T = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0.1 & 0.9 \\ 0.6 & 0.4 & 0 \end{bmatrix}$$

If initial state is  $\mu(x^{(1)}) = (0.5, 0.2, 0.3)$  (can be any state), it follows that:

$$\mu(x^{(2)}) = \mu(x^{(1)})T = (0.18, 0.64, 0.18)$$

$\vdots$

$$\mu(x^{(t)}) = \mu(x^{(t-1)})T = (0.2213, 0.4098, 0.3689)$$

$$\mu(x^{(t+1)}) = \mu(x^{(t+2)})T = (0.2213, 0.4098, 0.3689)$$

Converges to  
stationary distribution!

Image source: "An introduction to MCMC for Machine Learning", Christophe Andrieu et al.

# Properties of a Markov Chain

## 3. Irreducibility:

- A Markov chain is irreducible if for any state of the Markov chain, there is a **positive probability** of visiting all other states, i.e.

$$\text{if } \forall a, b \in \mathcal{X}, \quad \exists t \geq 0$$

$$\text{s.t. } p(x_t = b \mid x_0 = a) > 0$$

# Properties of a Markov Chain

## 4. Aperiodicity:

- The Markov chain **should not get trapped in cycles**, i.e.

$$\gcd\{t : p(x_t = a \mid x_0 = a) > 0\} = 1, \quad \forall a \in \mathcal{X}$$

greatest common divisor



# Ergodic Theorem for Markov Chains

- A Markov chain is ergodic if it is **irreducible and aperiodic**.
- **Ergodicity is important**: it implies we can reach the stationary/limiting distribution  $\pi$ , no matter the initial distribution  $\pi_0$ .
- All good MCMC algorithms **must satisfy ergodicity**, so that we cannot initialize in a way that will never converge.

# Detailed Balance (Reversibility)

- A probability vector  $\pi = p(x)$  defined on  $\mathcal{X}$  **satisfies detailed balance** w.r.t  $T$  if:

$$\pi_a T_{ab} = \pi_b T_{ba}, \quad \forall a, b \in \mathcal{X}$$

**Remark 1:** Detailed balance  $\implies$  stationary distribution, i.e.  $\pi T = \pi$ .

**Proof:**

$$\begin{aligned} \pi_b &= \sum_a \pi_a T_{ab} \\ &= \sum_a \pi_b T_{ba} && \text{(detailed balance)} \\ &= \pi_b \boxed{\sum_a T_{ba}} = 1 && \text{(sum over row of } T_{ba}) \\ &= \pi_b, \quad \forall b \in \mathcal{X} && \text{(stationary distribution)} \end{aligned}$$

# Detailed Balance (Reversibility)

- A probability vector  $\pi = p(x)$  defined on  $\mathcal{X}$  **satisfies detailed balance** w.r.t  $T$  if:

$$\pi_a T_{ab} = \pi_b T_{ba}, \quad \forall a, b \in \mathcal{X}$$

**Remark 2:** Detailed balance = “reversibility”

- Just a **terminology**: we say that a Markov chain is “reversible” if it had a stationary distribution  $\pi$  that satisfies detailed balance w.r.t  $T$ .

# Why does Metropolis-Hastings work?

- Recall that we draw a sample  $x'$  according to  $q(x'|x)$ , and then accept/reject according to  $\mathcal{A}(x', x)$ .
- In other words, the **transition kernel** is:

$$T(x' | x) = q(x' | x) \mathcal{A}(x' | x)$$

- We shall prove that the Metropolis-Hasting algorithm **satisfies** detailed balance!

# Why does Metropolis-Hastings work?

- Recall that:

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

- Notice this implies the following:

$$\text{if } \mathcal{A}(x', x) \leq 1, \text{ then } \frac{\tilde{p}(x)q(x'|x)}{\tilde{p}(x')q(x|x')} \geq 1$$

$$\text{and thus } \mathcal{A}(x, x') = 1$$

# Why does Metropolis-Hastings work?

- Now suppose  $\mathcal{A}(x', x) < 1$  and  $\mathcal{A}(x, x') = 1$ , we have:

$$\mathcal{A}(x', x) = \frac{\tilde{p}(x')q(x|x')}{\tilde{p}(x)q(x'|x)}$$

$$\mathcal{A}(x', x)\tilde{p}(x)q(x'|x) = \tilde{p}(x')q(x|x')$$

$$\mathcal{A}(x', x)\tilde{p}(x)q(x'|x) = \mathcal{A}(x, x')\tilde{p}(x')q(x|x')$$

$$\tilde{p}(x)T(x' | x) = \tilde{p}(x')T(x | x')$$

**This is the detailed balance condition!**

# Why does Metropolis-Hastings work?

$$\tilde{p}(x)T(x' | x) = \tilde{p}(x')T(x | x')$$

- In other words, the Metropolis-Hasting algorithm leads to a **stationary distribution**  $\tilde{p}(x)$ !
- Recall we defined  $\tilde{p}(x)$  to be the (un-normalized) **true distribution** of  $x$ !
- Thus, the Metropolis-Hasting **eventually converges** to the true distribution!

# Metropolis Algorithm

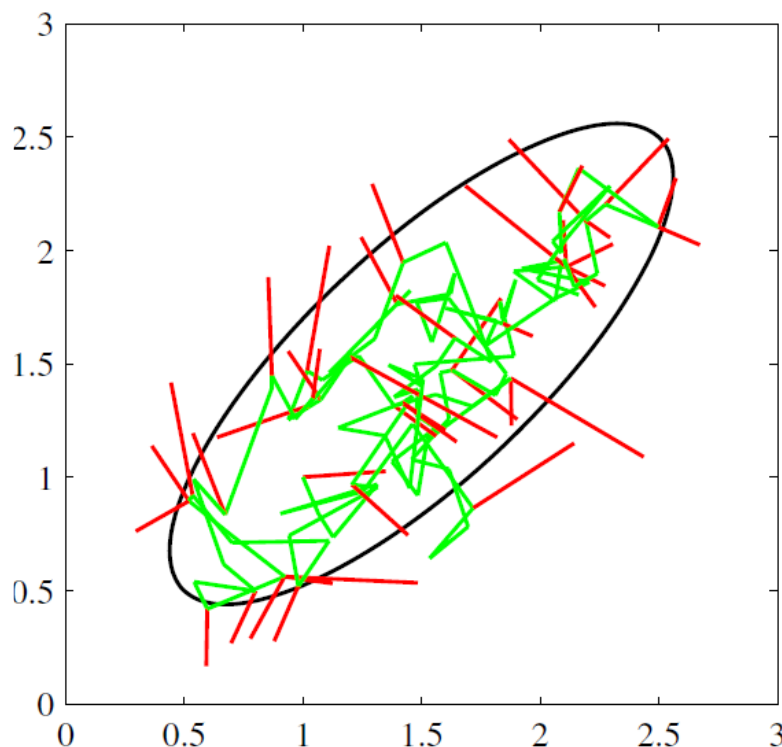
- Metropolis algorithm is a **special case** of the Metropolis-Hasting algorithm.
- Proposal distribution is a **random walk**, i.e.  $q(x|x') = q(x'|x)$ , e.g. an isotropic Gaussian distribution.
- **Acceptance probability** of Metropolis algorithm is given by:

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



# Metropolis Algorithm

- Illustration of using Metropolis algorithm (proposal distribution: isotropic Gaussian) to sample from a Gaussian distribution:



— Accepted sample  
— Rejected sample

150 candidate samples are generated, 43 are rejected.

Image Source: "Pattern Recognition and Machine Learning", Christopher Bishop

# Gibbs Sampling

- Suppose we have an  $N$ -dimensional vector  $x$  and the expressions for the **full conditionals**:

$$p(x_j | x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_N).$$

- In this case, we use the following **proposal distribution** for  $j = 1, \dots, N$ :

$$q(x' | x^{(i)}) = \begin{cases} p(x'_j | x_{\setminus j}^{(i)}) & \text{if } x'_{\setminus j} = x_{\setminus j}^{(i)} \\ 0 & \text{Otherwise} \end{cases}$$

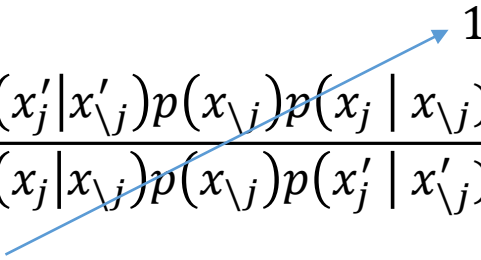
# Gibbs Sampling

- Gibbs sampling is a special case of the Metropolis-Hasting algorithm where the **acceptance probability is always one**.

**Proof:**  $\mathcal{A}(x', x) = \min \left\{ 1, \frac{p(x')q(x|x')}{p(x)q(x'|x)} \right\}$ , where  $\begin{matrix} p(x) = p(x_j|x_{\setminus j})p(x_{\setminus j}) \\ p(x') = p(x'_j|x'_{\setminus j})p(x'_{\setminus j}) \end{matrix}$

$$= \min \left\{ 1, \frac{p(x'_j|x'_{\setminus j})p(x'_{\setminus j})p(x_j | x'_{\setminus j})}{p(x_j|x_{\setminus j})p(x_{\setminus j})p(x'_j | x_{\setminus j})} \right\}$$

We use  $x'_{\setminus j} = x_{\setminus j}$  because these components are **kept fixed** during the sampling step:

$$\Rightarrow \mathcal{A}(x', x) = \min \left\{ 1, \frac{p(x'_j|x'_{\setminus j})p(x_{\setminus j})p(x_j | x_{\setminus j})}{p(x_j|x_{\setminus j})p(x_{\setminus j})p(x'_j | x_{\setminus j})} \right\} = 1$$


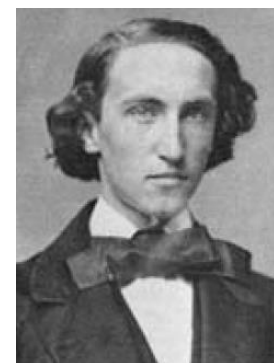
# Gibbs Sampling

---

## Algorithm : Gibbs Sampling

---

1. Initialize  $\{x_i : i = 1, \dots, M\}$
2. For  $\tau = 1, \dots, T$  :
  3. Sample  $x_1^{\tau+1} \sim p(x_1 | x_2^{(\tau)}, x_3^{(\tau)}, \dots, x_M^{(\tau)})$ .
  4. Sample  $x_2^{\tau+1} \sim p(x_2 | x_1^{(\tau+1)}, x_3^{(\tau)}, \dots, x_M^{(\tau)})$ .
  - $\vdots$
  5. Sample  $x_j^{\tau+1} \sim p(x_j | x_1^{(\tau+1)}, \dots, x_{j-1}^{(\tau+1)}, x_{j+1}^{(\tau)}, \dots, x_M^{(\tau)})$ .
  - $\vdots$
  6. Sample  $x_M^{\tau+1} \sim p(x_M | x_1^{(\tau+1)}, x_2^{(\tau+1)}, \dots, x_{M-1}^{(\tau+1)})$



Josiah Willard Gibbs  
1839–1903

Image Source: "Pattern Recognition and Machine Learning", Christopher Bishop

# Gibbs Sampling: Markov Blankets

- The conditional  $p(x_j | x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_N)$  looks intimidating, but recall **Markov Blankets**:

$$p(x_j | x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_N) = p(x_j | \underbrace{\text{MB}(x_j)}_{\text{Markov blanket of } x_j}) .$$

- Bayesian network**: the Markov blanket of  $X_j$  is the set containing its parents, children, and co-parents.
- MRF**: the Markov Blanket of  $X_j$  is its immediate neighbors.

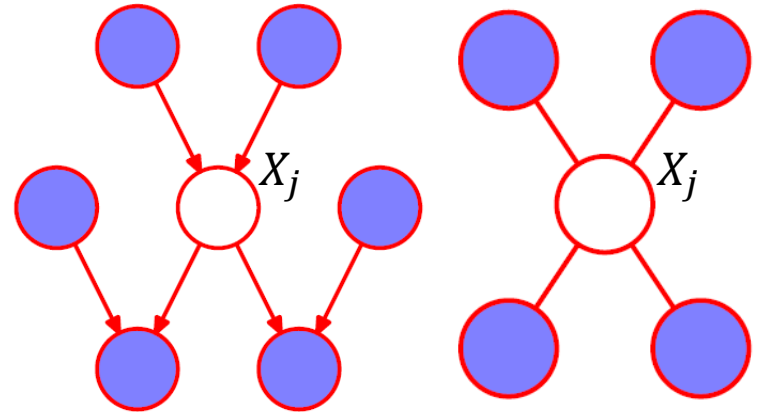
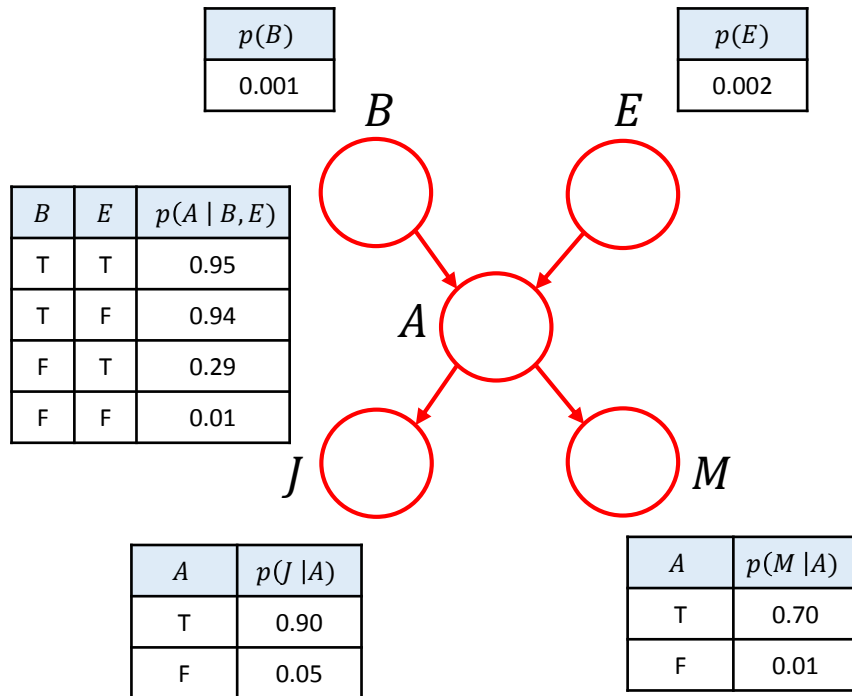


Image Source: "Pattern Recognition and Machine Learning", Christopher Bishop

# Gibbs Sampling:

## Example on a Bayesian Network

$B$ : Burglary,  $E$ : Earthquake,  $A$ : Alarm,  $J$ : John Calls,  $M$ : Mary Calls



- Assume we sample variables in the order  $B, E, A, J, M$ .
- Initialize all variables at  $t = 0$  to False.

$t$	$B$	$E$	$A$	$J$	$M$
0	F	F	F	F	F
1					
2					
3					
4					

# Gibbs Sampling:

## Example on a Bayesian Network

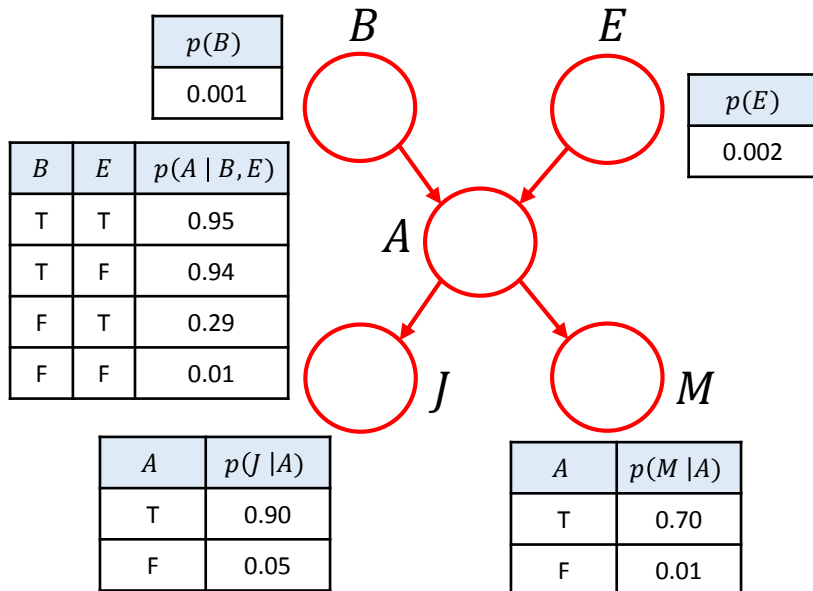
- Sampling  $p(B | A, E)$  at  $t = 1$ : using Bayes rule, we have

$$p(B | A, E) \propto p(A | B, E) p(B)$$

- $(A, E) = (F, F)$ , we compute the following, and sample  $B = F$

$$p(B = T | A = F, E = F) \propto (0.06)(0.001) = 0.00006$$

$$p(B = F | A = F, E = F) \propto (0.99)(0.999) = 0.98901$$



$t$	$B$	$E$	$A$	$J$	$M$
0	F	F	F	F	F
1	F				
2					
3					
4					

# Gibbs Sampling:

## Example on a Bayesian Network

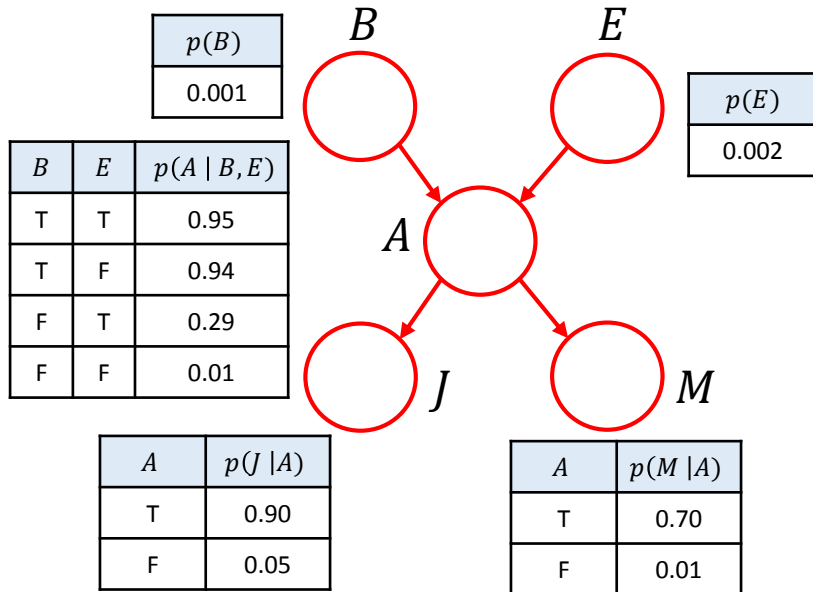
- Sampling  $p(E|A, B)$  at  $t = 1$ : using Bayes rule, we have

$$p(E | A, B) \propto p(A | B, E) p(E)$$

- $(A, B) = (F, F)$ , we compute the following, and sample  $E = T$

$$p(E = T | A = F, B = F) \propto (0.71)(0.002) = 0.00142$$

$$p(E = F | A = F, B = F) \propto (0.99)(0.998) = 0.98802$$



$t$	$B$	$E$	$A$	$J$	$M$
0	F	F	F	F	F
1	F	T			
2					
3					
4					



# Gibbs Sampling:

## Example on a Bayesian Network

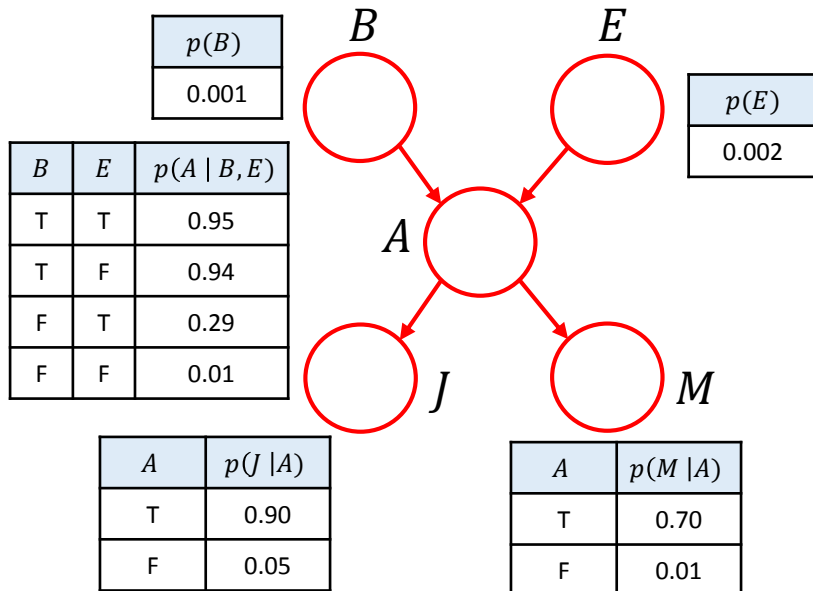
- Sampling  $p(A|B, E, J, M)$  at  $t = 1$ : using Bayes rule

$$p(A | B, E, J, M) \propto p(J|A)p(M|A)p(A|B, E)$$

- $(B, E, J, M) = (F, T, F, F)$ , we compute the following, and sample  $A = F$

$$p(A = T|B = F, E = T, J = F, M = F) \propto (0.1)(0.3)(0.29) = 0.0087$$

$$p(A = F|B = F, E = T, J = F, M = F) \propto (0.95)(0.99)(0.71) = 0.6678$$



$t$	$B$	$E$	$A$	$J$	$M$
0	F	F	F	F	F
1	F	T	F		
2					
3					
4					

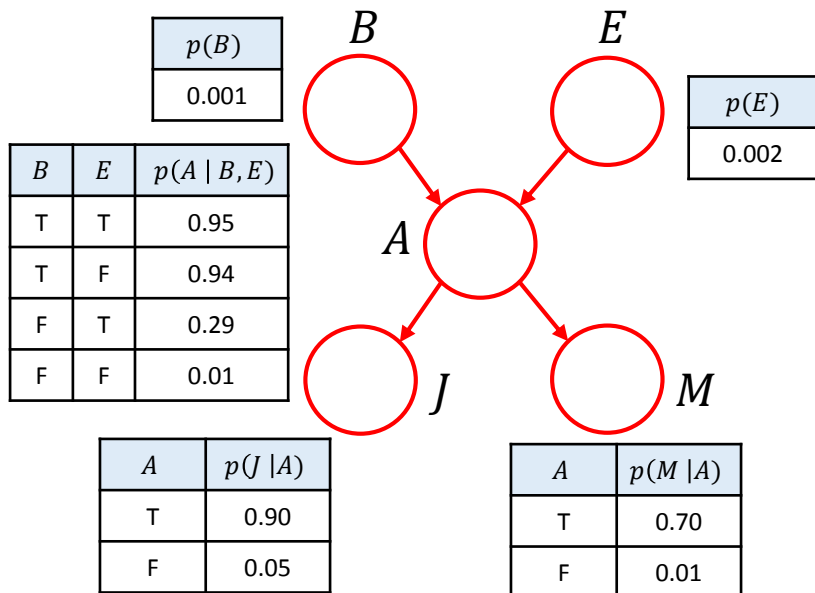
# Gibbs Sampling:

## Example on a Bayesian Network

- Sampling  $p(J|A)$  at  $t = 1$ : no need to apply Bayes rule
- $A = F$ , we compute the following, and sample  $J = T$

$$p(J = T|A = F) \propto 0.05$$

$$p(J = F|A = F) \propto 0.95$$



$t$	$B$	$E$	$A$	$J$	$M$
0	F	F	F	F	F
1	F	T	F	T	
2					
3					
4					

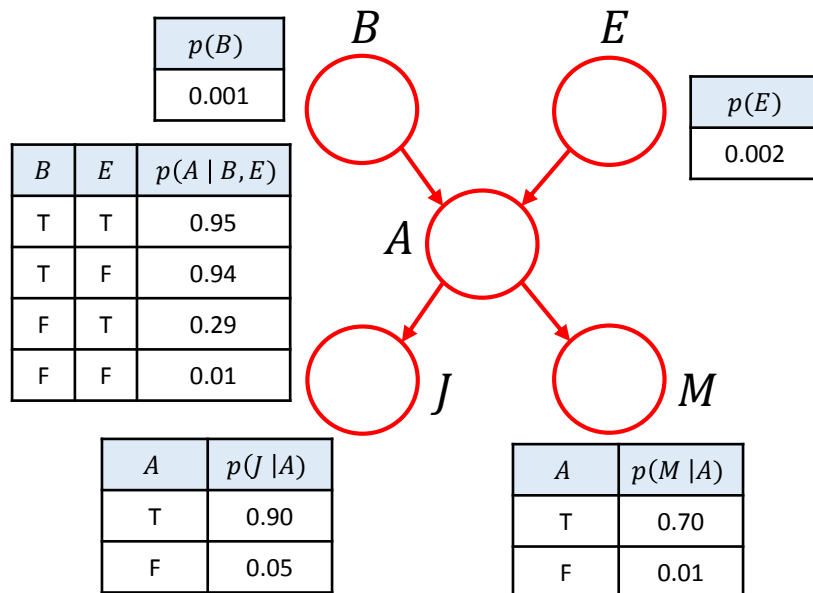
# Gibbs Sampling:

## Example on a Bayesian Network

- Sampling  $p(M|A)$  at  $t = 1$ : no need to apply Bayes rule
- $A = F$ , we compute the following, and sample  $M = F$

$$p(M = T|A = F) \propto 0.01$$

$$p(M = F|A = F) \propto 0.99$$

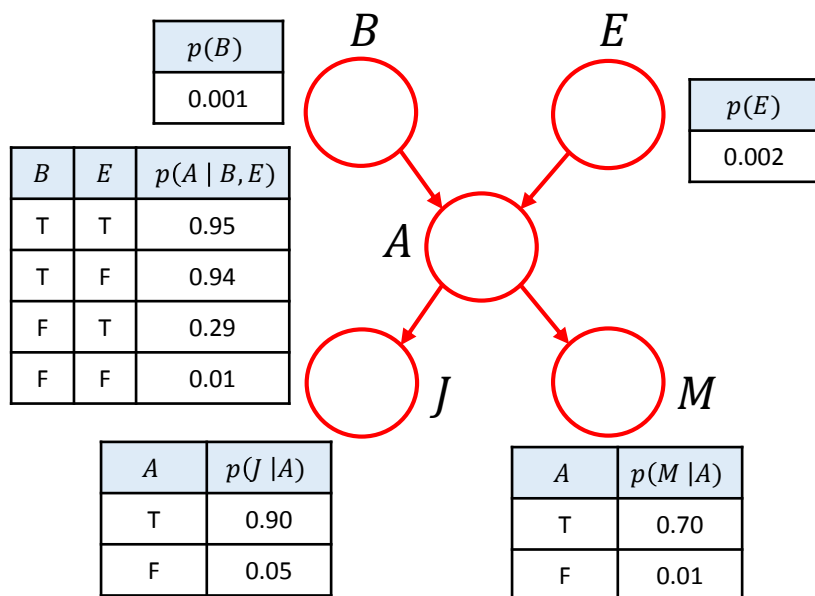


$t$	$B$	$E$	$A$	$J$	$M$
0	F	F	F	F	F
1	F	T	F	T	F
2					
3					
4					

# Gibbs Sampling:

## Example on a Bayesian Network

- Now  $t = 2$ , and we repeat the procedure to sample new values of  $B, E, A, J, M \dots$
- And similarly for  $t = 3, 4$ , etc.



$t$	$B$	$E$	$A$	$J$	$M$
0	F	F	F	F	F
1	F	T	F	T	F
2	F	T	T	T	T
3	T	F	T	F	T
4	T	F	T	F	F

# Gibbs Sampling: Illustration

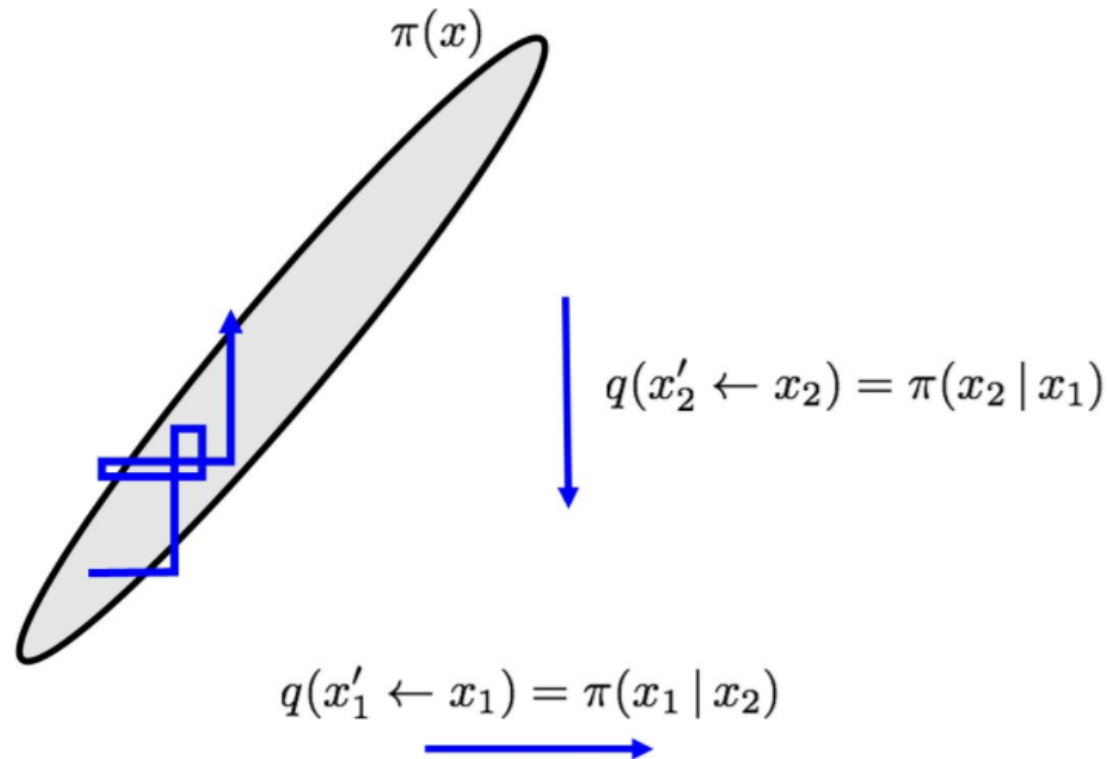


Image source: <http://slideplayer.com/slide/4261639/>

# Summary

- We have looked at how to:
  1. Explain the **Monte Carlo principle** and its justification for sampling methods.
  2. Apply **Rejection, Importance, Metropolis-Hasting, Metropolis** and **Gibbs** sampling methods to do maximal probability, approximate inference, and expectation.
  3. Use **Markov chain properties**, i.e. homogenous, stationary distribution, irreducibility, aperiodic, ergodic and detail balance, to show validity of MH algorithm.