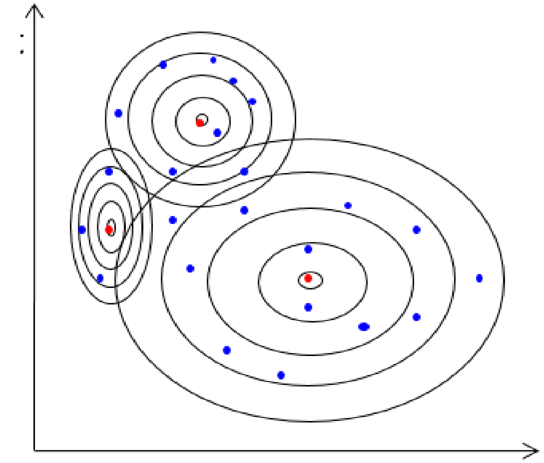
Markov chain Monte Carlo(MCMC)

Jin Young Choi

Outline

- Monte Carlo : Sample from a distribution to estimate the distribution
- Markov Chain Monte Carlo (MCMC)
  - Applied to Clustering, Unsupervised Learning, Bayesian Inference
- Importance Sampling
- Metropolis-Hastings Algorithm
- Gibbs Sampling
- Markov Blanket in Sampling for Bayesian Network
- Example: Estimation of Gaussian Mixture Model



$$p(\mathrm{x}|\theta) = \sum_k p(\mathrm{x}|\theta_k) p(\theta_k|\theta)$$
$$= \sum_Z p(\mathrm{x}, Z|\theta) = \sum_{Z=k} p(\mathrm{x}|Z=k,\theta) p(Z=k|\theta)$$

$$\boxed{p(x|D) = \sum_{z,\theta} p(x,z,\theta|D) = ?\,, p(z|x,\theta) = ?\,, p(\theta|x,z) = ?}$$

Markov chain Monte Carlo(MCMC)

- Monte Carlo : Sample from a distribution
  - to estimate the distribution for GMM estimation, Clustering (Labeling, Unsupervised Learning)
  - to compute max, mean

- Markov Chain Monte Carlo : sampling using “local” information
  - Generic “problem solving technique”
  - decision/inference/optimization/learning problem
  - generic, but not necessarily very efficient

Monte Carlo Integration

- General problem: evaluating
$$\mathbb{E}_P[h(X)] = \int h(x)P(x)dx$$
can be difficult. ($\int |h(x)|P(x)dx < \infty$)

- If we can draw samples $x^{(s)} \sim P(x)$, then we can estimate
$$\mathbb{E}_P[h(X)] \approx \bar{h}_N = \frac{1}{N}\sum_{s=1}^{N} h\left(x^{(s)}\right).$$

- Monte Carlo integration is great if you can sample from the target distribution
  - But what if you can't sample from the target?
  - Importance sampling: Use of a simple distribution

Importance Sampling

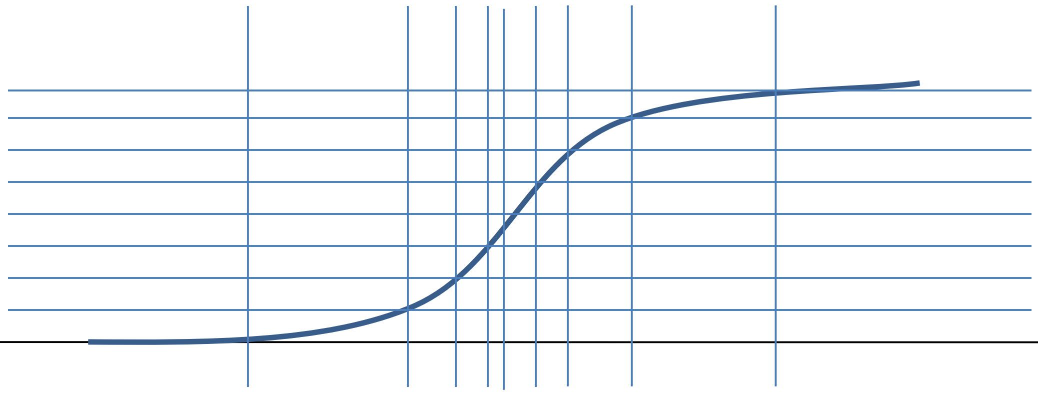
- Idea of importance sampling:

  Draw the sample from a proposal distribution $Q(\cdot)$ and re-weight the integral using importance weights so that the correct distribution is targeted

$$\mathbb{E}_P[h(X)] = \int \frac{h(x)P(x)}{Q(x)} Q(x)dx = \mathbb{E}_Q\left[\frac{h(X)P(X)}{Q(X)}\right].$$

- Hence, given an iid sample $x^{(s)}$ from Q, our estimator becomes

$$E_Q\left[\frac{h(X)P(X)}{Q(X)}\right] = \frac{1}{N}\sum_{s=1}^{N} \frac{h\left(x^{(s)}\right)P\left(x^{(s)}\right)}{Q\left(x^{(s)}\right)}$$



Limitations of Monte Carlo

- Direct (unconditional) sampling
  - Hard to get rare events in high-dimensional spaces → Gibbs sampling

- Importance sampling
  - Do not work well if the proposal $Q(x)$ is very different from target $P(x)$
  - Yet constructing a $Q(x)$ similar to $P(x)$ can be difficult → Markov Chain

- Intuition: instead of a fixed proposal $Q(x)$, what if we could use an **adaptive** proposal?
  - X_{t+1} depends only on X_t, not on $X_0, X_1, \ldots, X_{t-1}$
  - Markov Chain

Markov Chains: Notation & Terminology

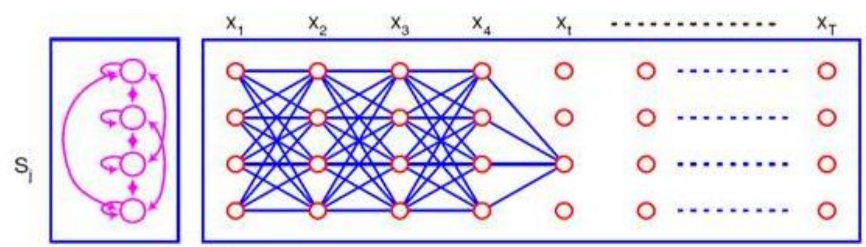
- Countable (finite) state space Ω (e.g. **N**)
- Sequence of random variables $\{X_t\}$ on Ω for $t = 0,1,2,\ldots$

- Definition : $\{X_t\}$ is a Markov Chain if

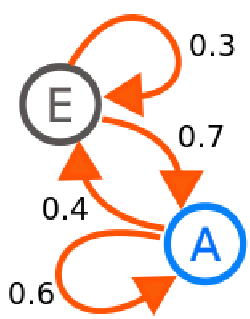
$$P(X_{t+1} = y \mid X_t = x_t, \ldots, X_0 = x_0) = P(X_{t+1} = y \mid X_t = x_t)$$

- Notation : $P(X_{t+1} = i \mid X_t = j) = p_{ji}$

  - Random Works



- Example.



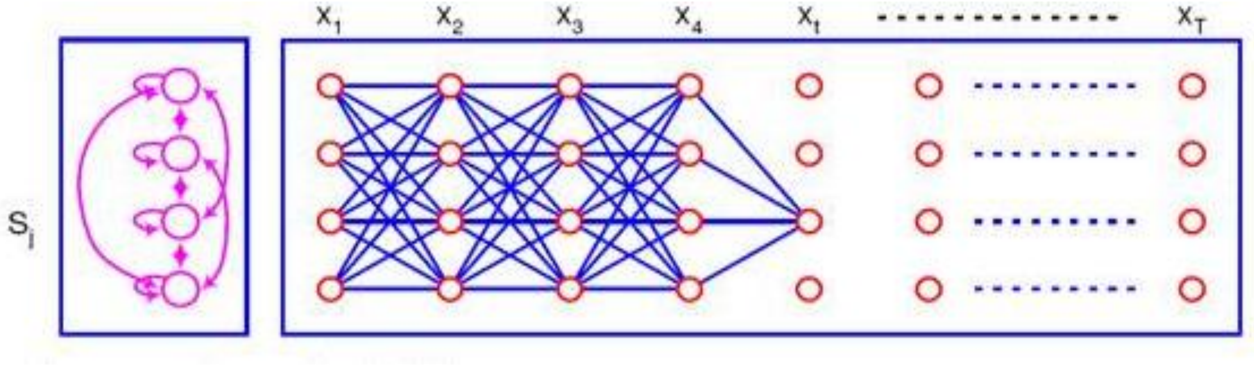
$$p_{AA} = P(X_{t+1} = A \mid X_t = A) = 0.6$$
$$p_{AE} = P(X_{t+1} = E \mid X_t = A) = 0.4$$
$$p_{EA} = P(X_{t+1} = A \mid X_t = E) = 0.7$$
$$p_{EE} = P(X_{t+1} = E \mid X_t = E) = 0.3$$

Markov Chains: Notation & Terminology

- Let $\boldsymbol{P} = (p_{ij})$ - transition probability matrix
  - dimension $|\Omega| \times |\Omega|$

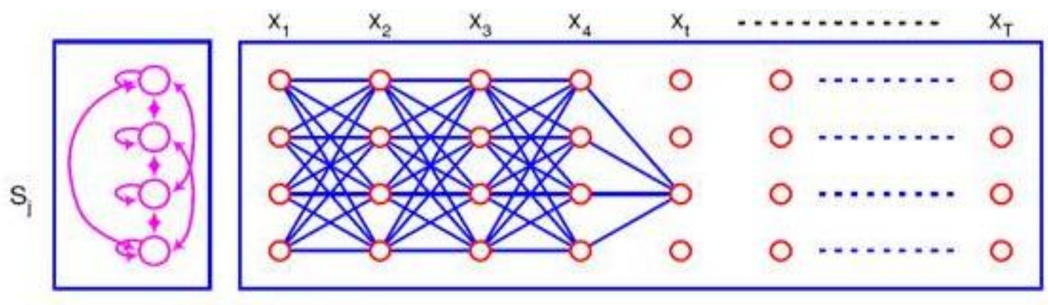
- Let $\pi_t(j) = P(X_t = j)$
  - π_0 : initial probability distribution

- Then $\pi_t(j) = \sum_i \pi_{t-1}(i)\, p_{ij} = (\pi_{t-1}\boldsymbol{P})(j) = (\pi_0 \boldsymbol{P}^t)(j)$
  $\pi_t = \pi_{t-1}\boldsymbol{P} = \pi_{t-2}\boldsymbol{P}^2 = \cdots = \pi_0 \boldsymbol{P}^t$



Markov Chains: Fundamental Properties

- Theorem:
  - If the limit $\left(\lim_{t\to\infty} P^t\right) = P$ exists and Ω is finite, then
    $$(\pi P)(j) = \pi(j) \text{ and } \sum_j \pi(j) = 1$$
    and such π is an **unique** solution to $\pi \boldsymbol{P} = \pi$ (π is called a **stationary distribution**)
  - No matter where we start, after some time, we will be in any state j with probability $\sim \pi(j)$



Markov Chain Monte Carlo

MCMC algorithm feature adaptive proposals

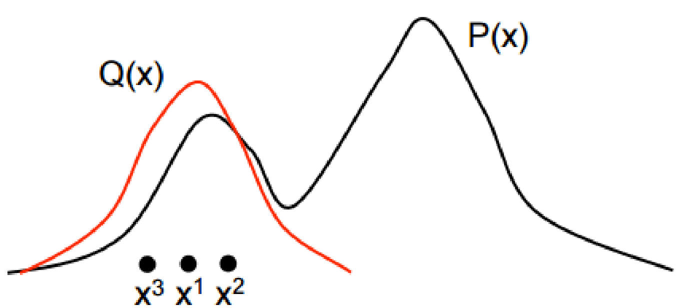
- Instead of $Q(x')$, they use $Q(x'|x)$ where x' is the new state being sampled, and x is the previous sample
- As x changes, $Q(x'|x)$ can also change (as a function of x')

importance

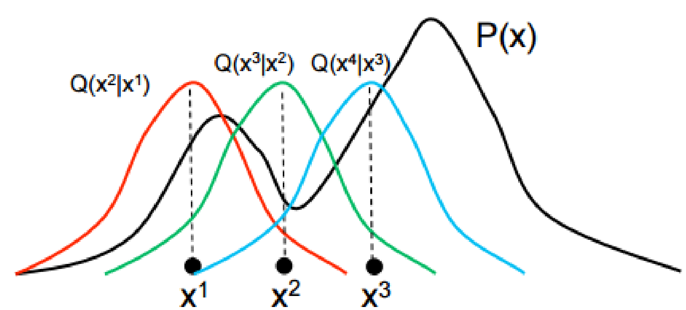
- The acceptance probability is set to $A(x'|x) = \min\left(1, \frac{P(x')/Q(x'|x)}{P(x)/Q(x|x')}\right)$
- No matter where we start, after some time, we will be in any state j with probability $\sim \pi(j)$

$Q(x'|x) = Q(x'|x)$ for Gaussian Why?

Importance sampling with a (bad) proposal Q(x)



MCMC with adaptive proposal Q(x'|x)



p_{11} $\rightarrow p_{12}$ p_{22}
$\leftarrow p_{21}$

p_{11} $\rightarrow p_{12}$ p_{22}
$\leftarrow p_{21}$

Metropolis-Hastings

- Draws a sample x' from $Q(x'|x)$, where x is the previous sample
- The new sample x' is accepted or rejected with some probability $A(x'|x)$
  - This acceptance probability is $A(x'|x) = \min\left(1, \frac{P(x')/Q(x'|x)}{P(x)/Q(x|x')}\right)$
  - $A(x'|x)$ is like a ratio of importance sampling weights
    - $\frac{P(x')}{Q(x'|x)}$ is the importance weight for x', $\frac{P(x)}{Q(x|x')}$ is the importance weight for x
    - We divide the importance weight for x' by that of x
    - Notice that we only need to compute $P(x')/P(x)$ rather than $P(x')$ or $P(x)$ separately
  - $A(x'|x)$ ensures that, after sufficiently many draws, our samples will come from the true distribution $P(x)$

$Q(x'|x) = Q(x'|x)$ for Gaussian Why?

$$\mathbb{E}_P[h(X)] = \int \frac{h(x)P(x)}{Q(x)} Q(x)dx = \mathbb{E}_Q\left[\frac{h(X)P(X)}{Q(X)}\right]$$

The MH Algorithm

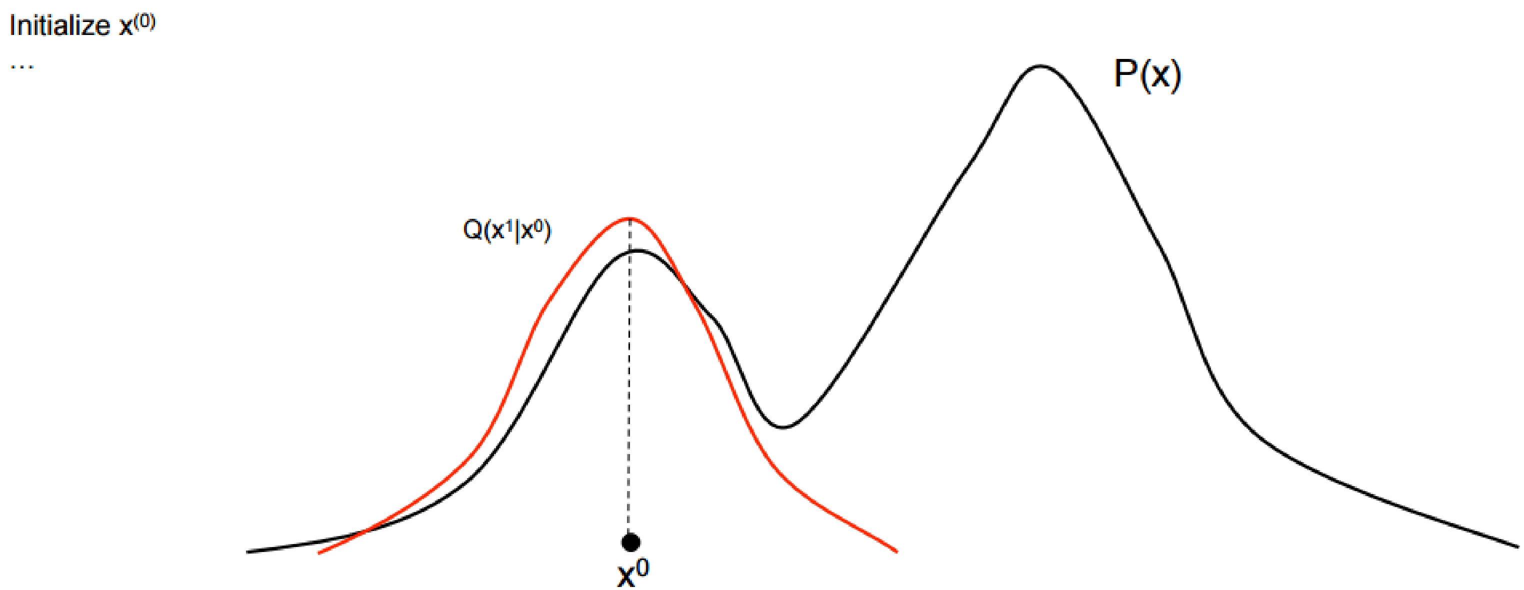
- Initialize starting state $x^{(0)}$,
- Burn-in: while samples have “not converged”
  - $x = x^{(t)}$
  - $t = t + 1$
  - Sample $x^* \sim Q(x^*|x)$ // draw from proposal
  - Sample $u \sim \mathrm{Uniform}(0,1)$ // draw acceptance threshold
    - If $u < A(x^*|x) = \min\left(1, \frac{P(x^*)Q(x|x^*)}{P(x)Q(x^*|x)}\right)$, $x^{(t)} = x^*$ // transition
    - Else $x^{(t)} = x$ // stay in current state
  - Repeat until converging

The MH Algorithm

$$A(x'|x) = \min\left(1, \frac{P(x')/Q(x'|x)}{P(x)/Q(x|x')}\right)$$

Example:

- Let $Q(x'|x)$ be a Gaussian centered on x
- We're trying to sample from a bimodal distribution $P(x)$

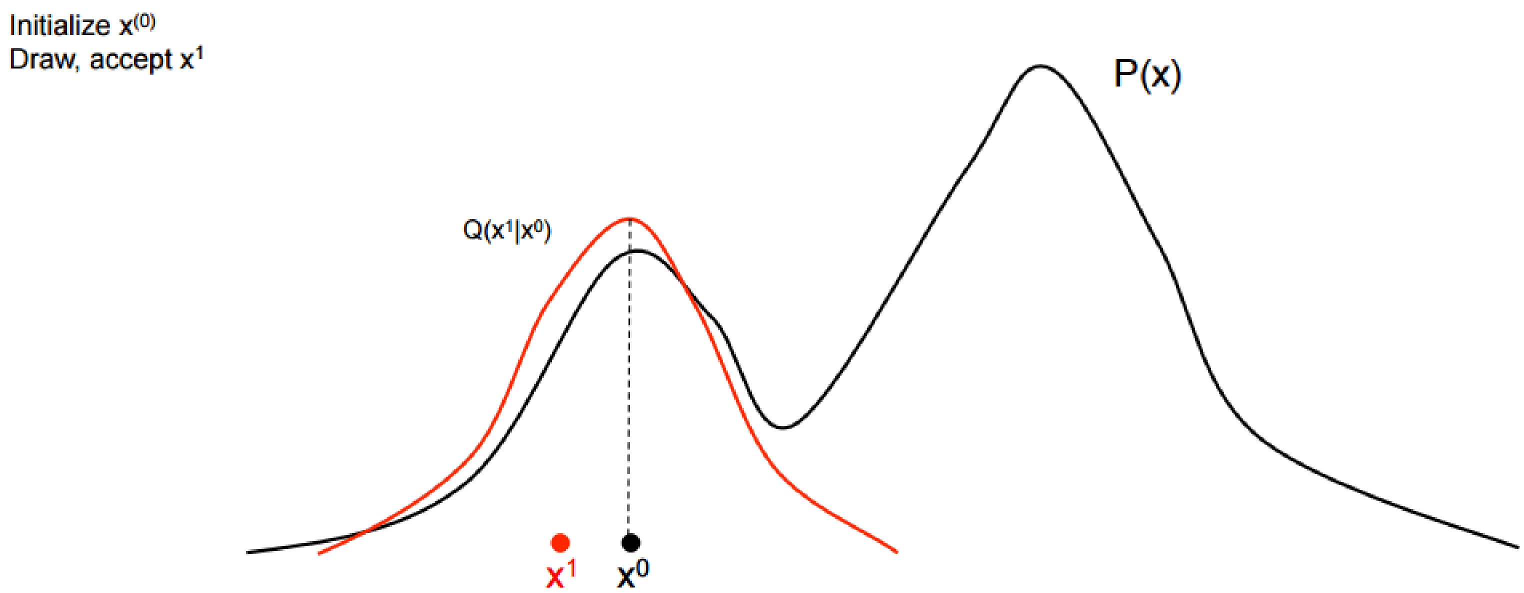


The MH Algorithm

$$A(x'|x) = \min\left(1, \frac{P(x')/Q(x'|x)}{P(x)/Q(x|x')}\right)$$

Example:

- Let $Q(x'|x)$ be a Gaussian centered on x
- We're trying to sample from a bimodal distribution $P(x)$

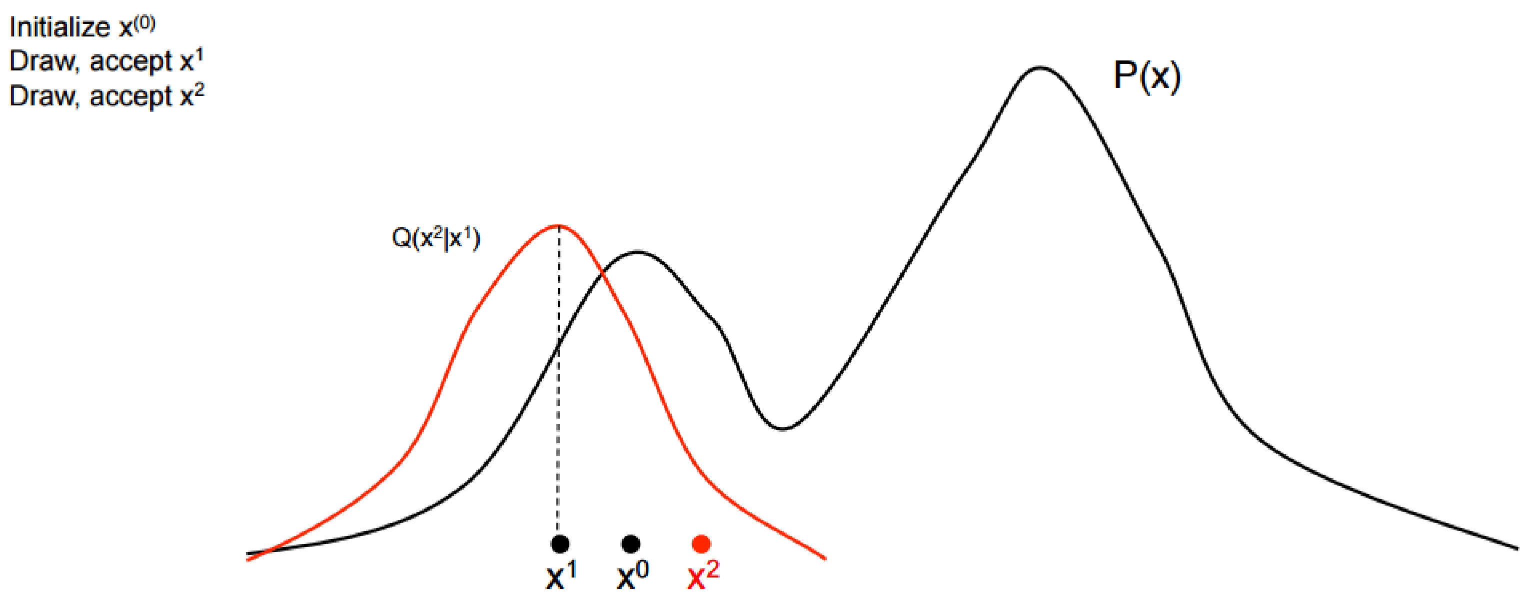


The MH Algorithm

$$A(x'|x) = \min\left(1, \frac{P(x')/Q(x'|x)}{P(x)/Q(x|x')}\right)$$

Example:

- Let $Q(x'|x)$ be a Gaussian centered on x
- We're trying to sample from a bimodal distribution $P(x)$

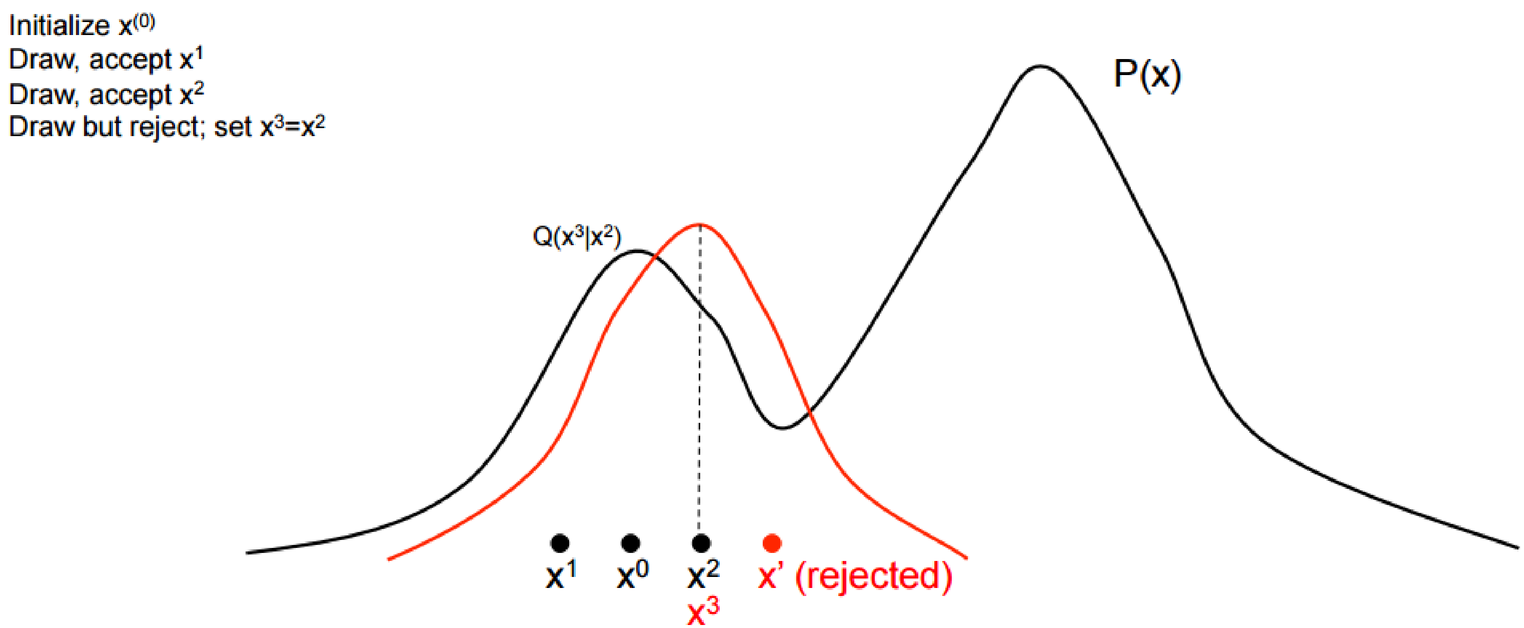


The MH Algorithm

$$A(x'|x) = \min\left(1, \frac{P(x')/Q(x'|x)}{P(x)/Q(x|x')}\right)$$

Example:

- Let $Q(x'|x)$ be a Gaussian centered on x
- We're trying to sample from a bimodal distribution $P(x)$

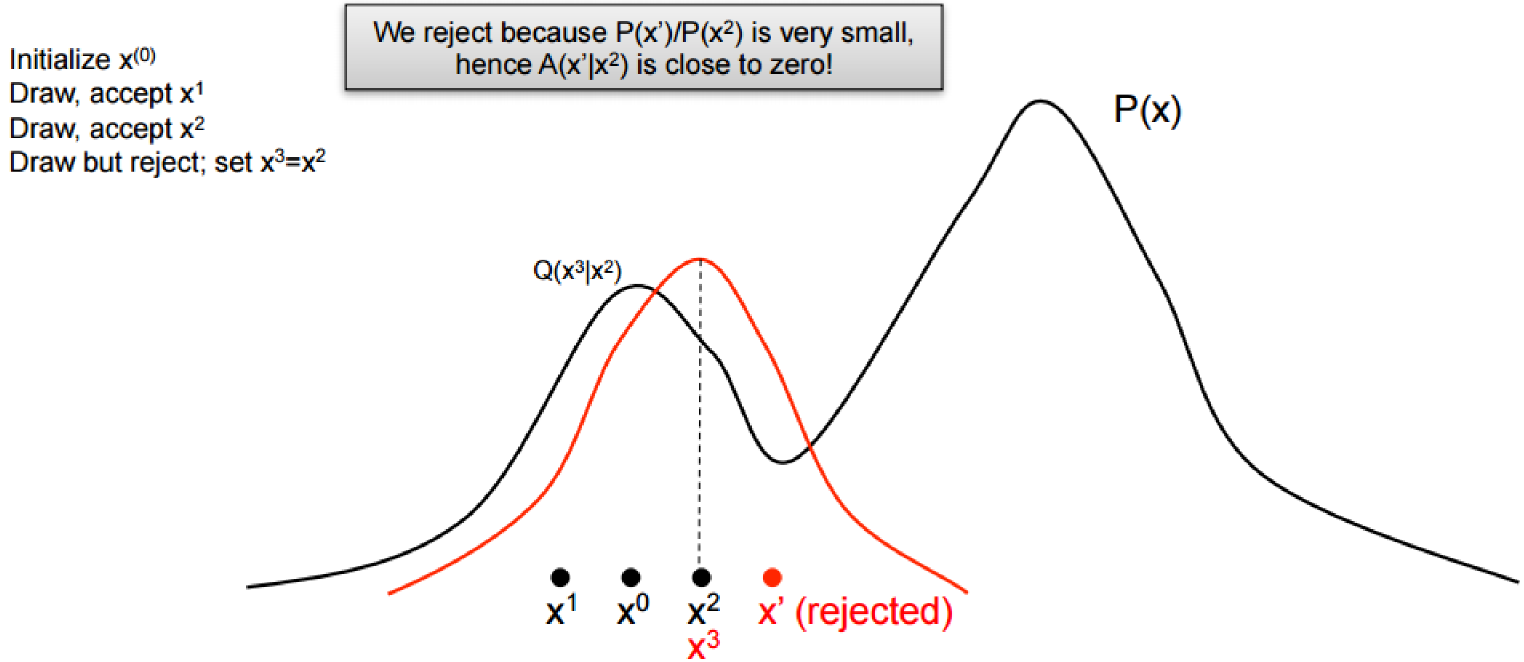


The MH Algorithm

$$A(x'|x) = \min\left(1, \frac{P(x')/Q(x'|x)}{P(x)/Q(x|x')}\right)$$

Example:

- Let $Q(x'|x)$ be a Gaussian centered on x
- We're trying to sample from a bimodal distribution $P(x)$

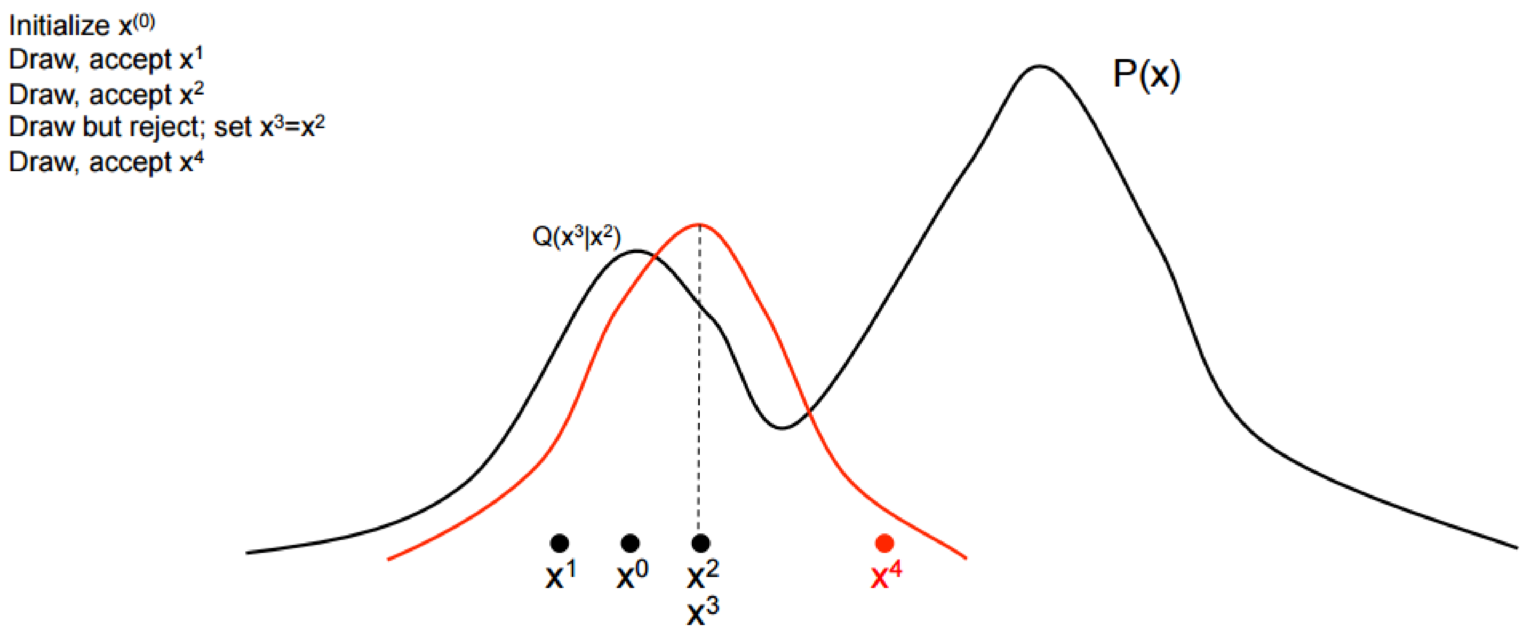


The MH Algorithm

$$A(x'|x) = \min\left(1, \frac{P(x')/Q(x'|x)}{P(x)/Q(x|x')}\right)$$

Example:

- Let $Q(x'|x)$ be a Gaussian centered on x
- We're trying to sample from a bimodal distribution $P(x)$

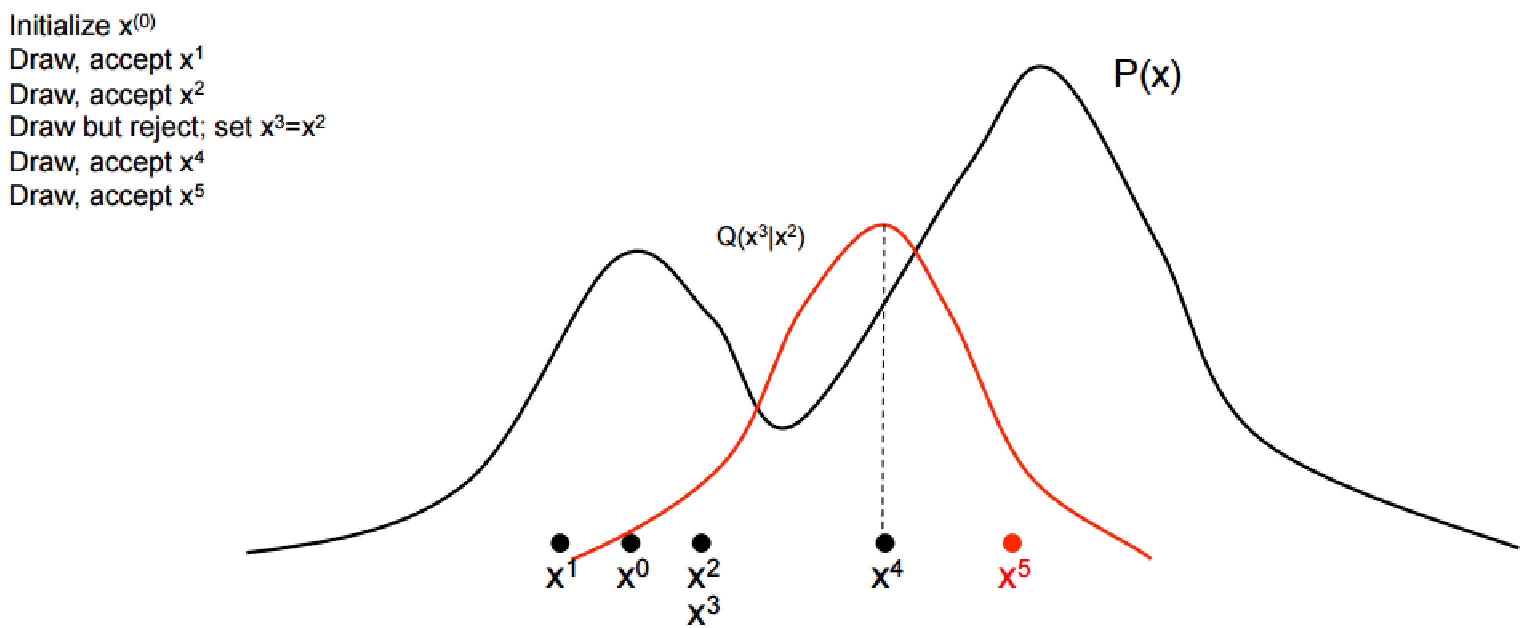


The MH Algorithm

$$A(x'|x) = \min\left(1, \frac{P(x')/Q(x'|x)}{P(x)/Q(x|x')}\right)$$

Example:

- Let $Q(x'|x)$ be a Gaussian centered on x
- We're trying to sample from a bimodal distribution $P(x)$

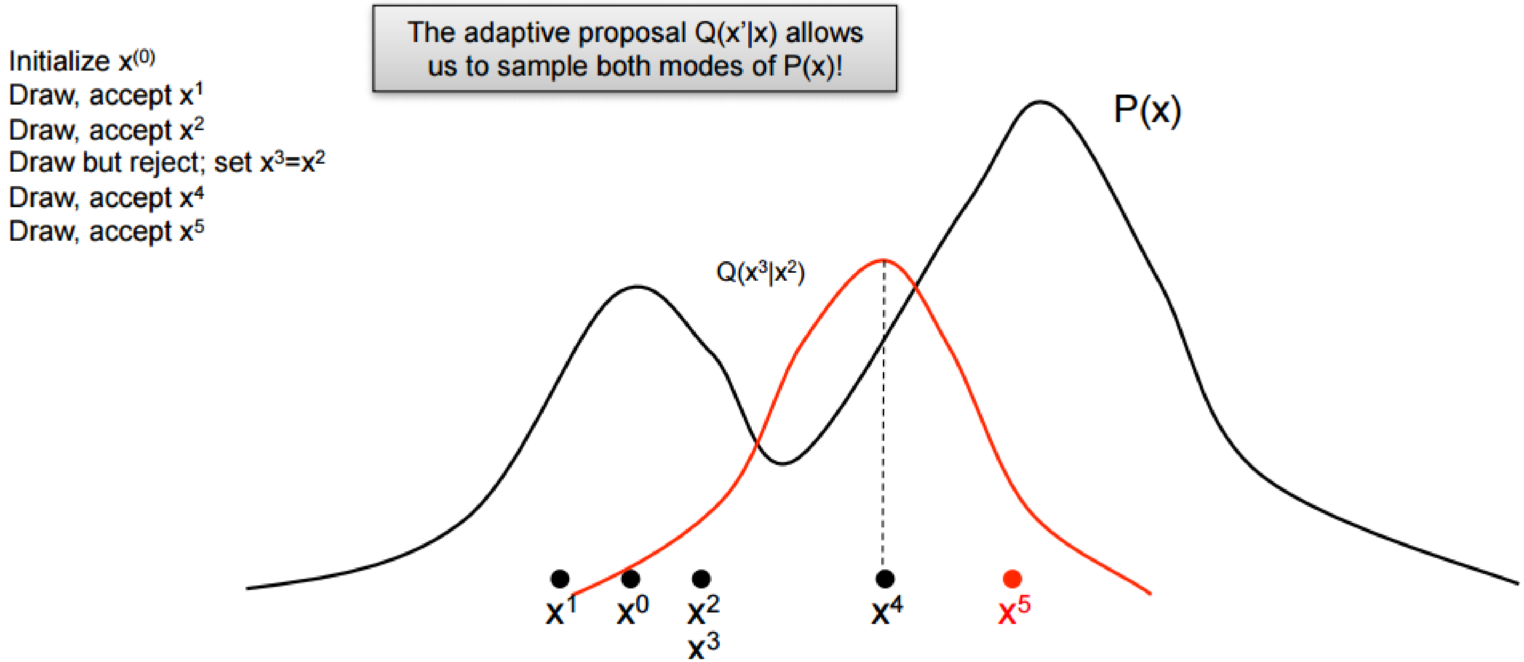


The MH Algorithm

$$A(x'|x) = \min\left(1, \frac{P(x')/Q(x'|x)}{P(x)/Q(x|x')}\right)$$

Example:

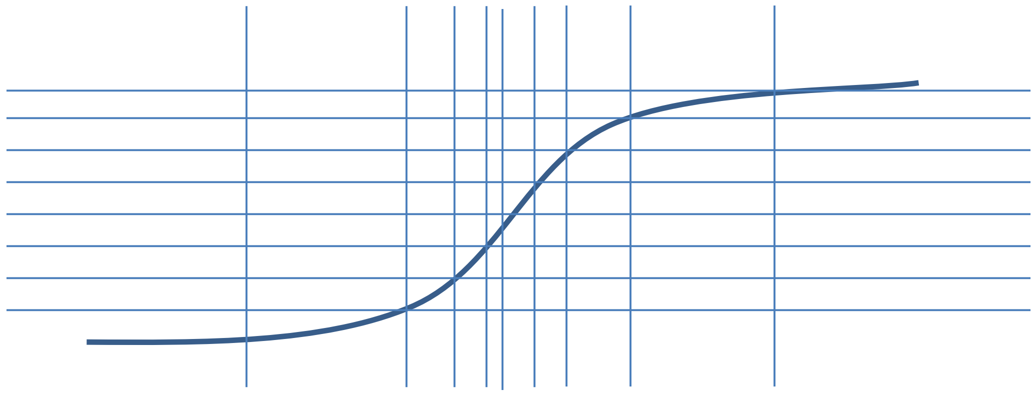
- Let $Q(x'|x)$ be a Gaussian centered on x
- We're trying to sample from a bimodal distribution $P(x)$



Gibbs Sampling

- Gibbs Sampling is an MCMC algorithm that samples each random variable of a graphical model, one at a time
  - GS is a special case of the MH algorithm

- Consider a factored state space
  - $x \in \Omega$ is a vector $x = (x_1, \dots, x_m)$
  - Notation: $x_{-i} = \{x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_m\}$



Gibbs Sampling

$$A(x'|x) = \min\left(1, \frac{P(x')/Q(x'|x)}{P(x)/Q(x|x')}\right)$$

- The GS algorithm:

1. Suppose the graphical model contains variables $x_1, \ldots, x_n$
2. Initialize starting values for $x_1, \ldots, x_n$
3. Do until convergence:
   1. Pick a component $i \in \{1, \ldots, n\}$
   2. Sample value of $z \sim P(x_i|x_{-i})$, and update $x_i \leftarrow z$

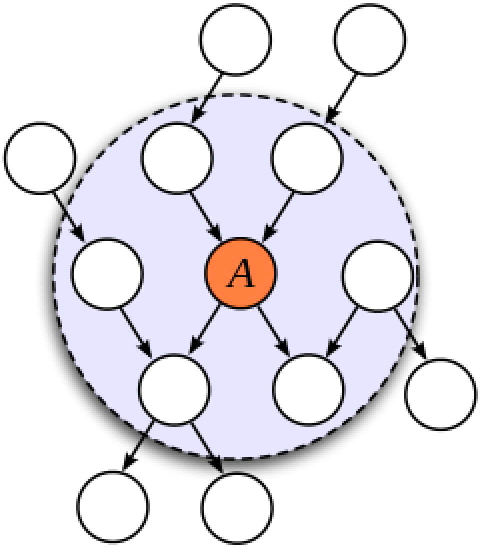
- When we update x_i, we <u>immediately</u> use its new value for sampling other variables x_j
- $P(x_i|x_{-i})$ achieves the acceptance probability in MH algorithm.

$$A(x_i', x_{-i}|x_i, x_{-i}) = min(1, \frac{P(x_i'|x_{-i})Q(x_i, x_{-i}|x_i', x_{-i})}{P(x_i|x_{-i})Q(x_i', x_{-i}|x_i, x_{-i})})$$
$$= min(1, \frac{P(x_i'|x_{-i})P(x_i|x_{-i})}{P(x_i|x_{-i})P(x_i'|x_{-i})})$$

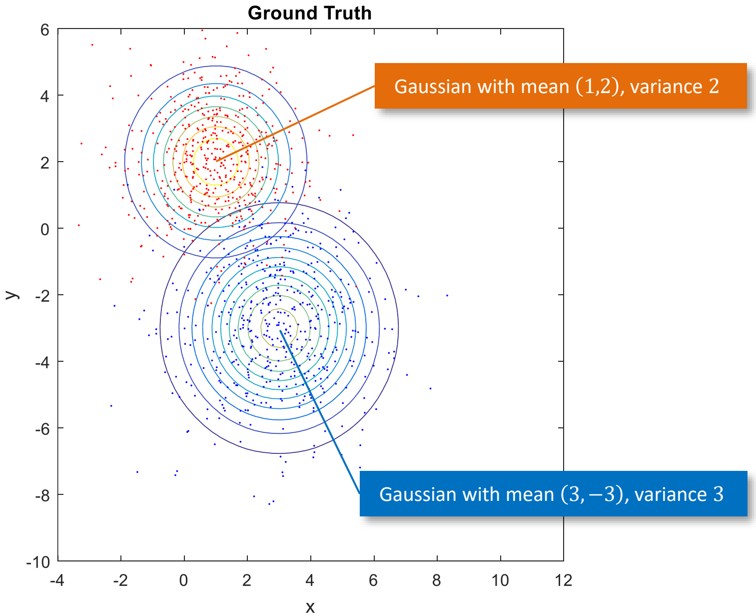
Markov Blankets

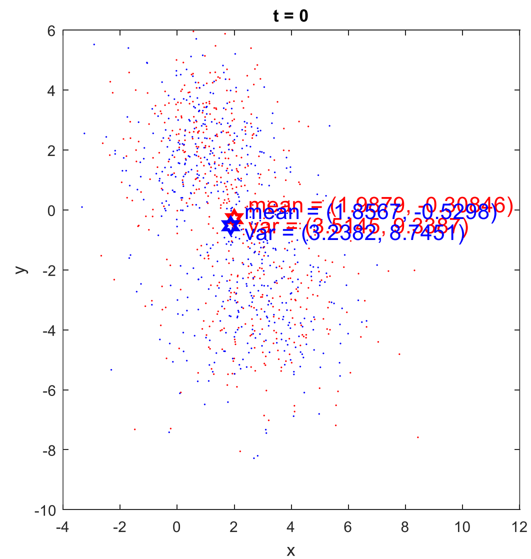
- The conditional $P(x_i|x_{-i})$ can be obtained using Markov Blanket
  - Let $MB(x_i)$ be the Markov Blanket of x_i, then
$$P(x_i \mid x_{-i}) = P\big(x_i|\mathrm{MB}(x_i)\big)$$

- For a Bayesian Network, the Markov Blanket of x_i is the set containing its parents, children, and co-parents



Gibbs Sampling: An Example



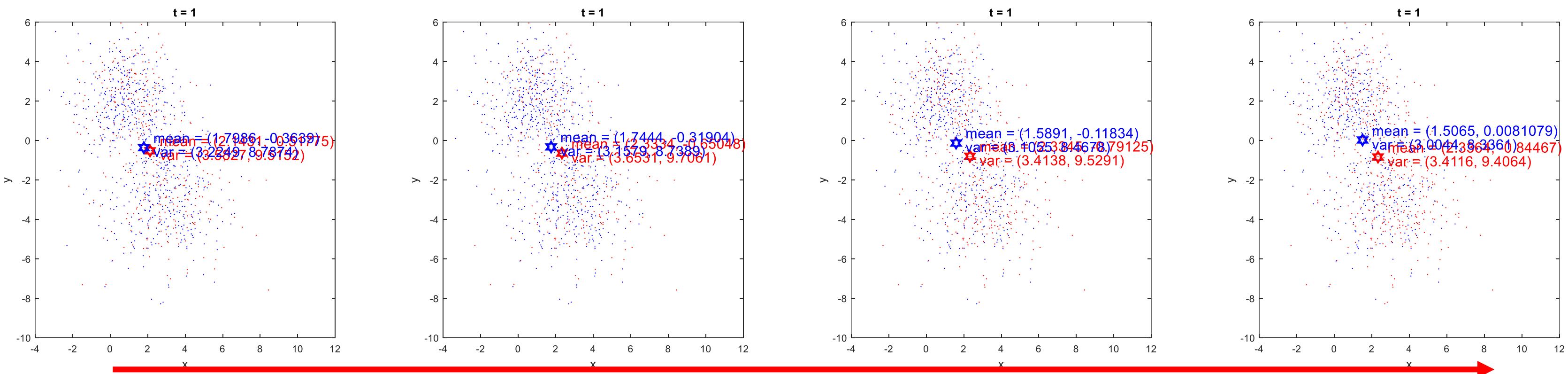


- Consider the GMM
  - The data x (position) are extracted from two Gaussian distribution
  - We do NOT know the class y of each data, and information of the Gaussian distribution
  - Initialize the class of each data at $t = 0$ to randomly

$$p(\mathrm{x}|\theta) = \sum_k p(\mathrm{x}|\theta_k)p(\theta_k|\theta)$$

$$= \sum_Z p(\mathrm{x}, Z|\theta) = \sum_{Z=k} p(\mathrm{x}|Z = k, \theta)p(Z = k|\theta)$$

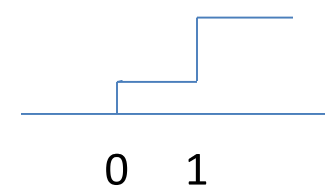
Gibbs Sampling: An Example



Iteration of i at the same t

Sampling $P(y_i \mid x_{-i}, y_{-i})$ at $t = 1$, we compute:

$$P(y_i = 0 \mid x_{-i}, y_{-i}) \propto \mathcal{N}\left(x_i \mid \mu_{x_{-i},0}, \sigma_{x_{-i},0}\right)$$
$$P(y_i = 1 \mid x_{-i}, y_{-i}) \propto \mathcal{N}\left(x_i \mid \mu_{x_{-i},1}, \sigma_{x_{-i},1}\right)$$

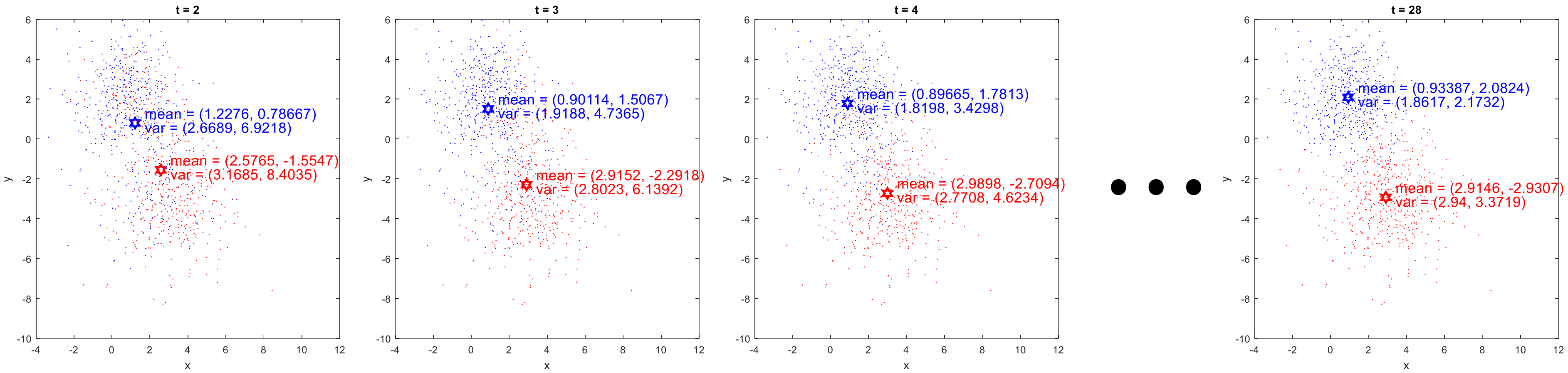


where

$$\mu_{x_{-i},K} = MEAN(X_{iK}), \sigma_{x_{-i},K} = VAR(X_{iK})$$
$$X_{iK} = \{x_j \mid x_j \in x_{-i}, y_j = K\}$$

And update y_i with $P(y_i \mid x_{-i}, y_{-i})$ and repeat for all data

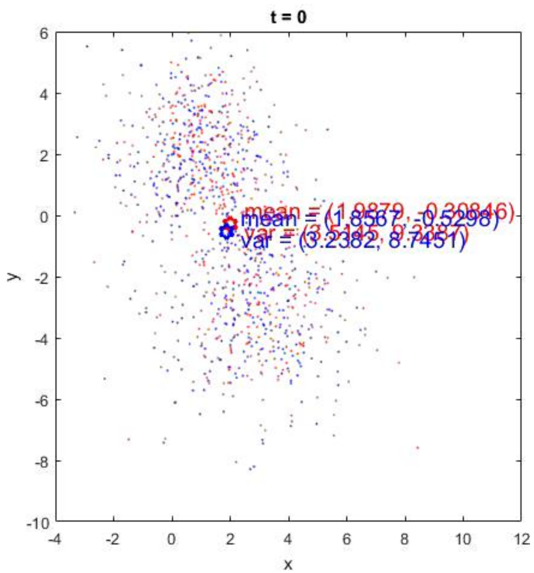
Gibbs Sampling: An Example

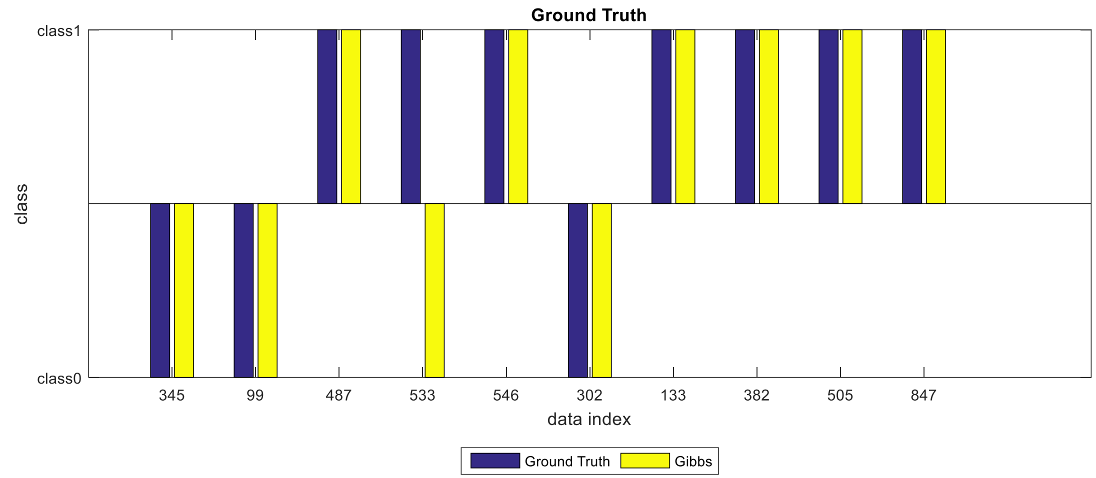


Now $t = 2$, and we repeat the procedure to sample new class of each data

And similarly for $t = 3, 4, \ldots$

Gibbs Sampling: An Example





- Data i's class can be chosen with tendency of y_i
  - The classes of the data can be oscillated after the sufficient sequences
  - We can assume the class of datum as more frequently selected class

- In the simulation, the final class is correct with the probability of 94.9% at $t = 100$

Interim Summary

Markov Chain Monte Carlo methods use adaptive proposals $Q(x'|x)$ to sample from the true distribution $P(x)$

Metropolis-Hastings allows you to specify any proposal $Q(x'|x)$

- But choosing a good $Q(x'|x)$ requires care

Gibbs sampling sets the proposal $Q(x_i'|x_{-1})$ to the conditional distribution $P(x_i'|x_{-1})$

- Acceptance rate always 1.
- But remember that high acceptance usually entails slow exploration
- In fact, there are better MCMC algorithms for certain models

Bolzmann Machine

Jin Young Choi

Overview

- Unsupervised Modelling of Binary Data
- What is Boltzmann Machine ?
- Restricted Boltzmann Machine (RBM)
- RBM Learning
- Contrast Divergence (CD)
- Example

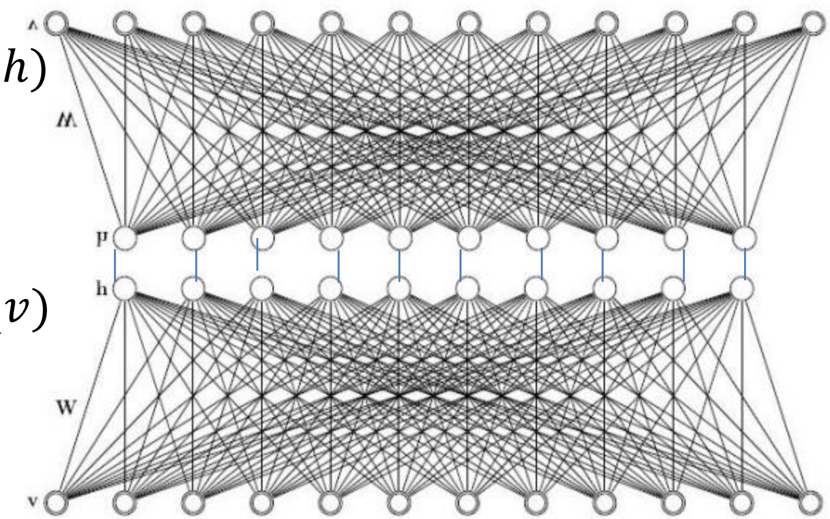
Unsupervised Modelling of Binary Data

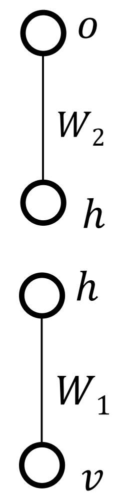
1 0 0 1 1 0 0 1 0 0 0 1

If no desired outputs ?

$o = \sigma(W_2 h)$

$h = \sigma(W_1 v)$





1 0 0 1 1 0 0 1 0 0 0 1

0 1 0 1 0 0 1 1 0 1 0 1

....

Modeling binary data

- Given a training set of binary vectors, fit a model that will assign a probability to other binary vectors

Name	Harry Potter	Avatar	LOTR3	Gladiator	Titanic	Glitter
Alice	1	1	1	0	0	0
Bob	1	0	1	0	0	0
Carol	1	1	1	0	0	0
David	0	0	1	1	1	0
Eric	0	0	1	1	0	0
Fred	0	0	1	1	1	0

Prefer SF/fantasy

Prefer Oscar winner

$$p(x) = \prod_j \left(\boxed{x_j p_j} + \boxed{(1 - x_j)(1 - p_j)} \right)$$

If component j of vector x is on

If component j of vector x is off

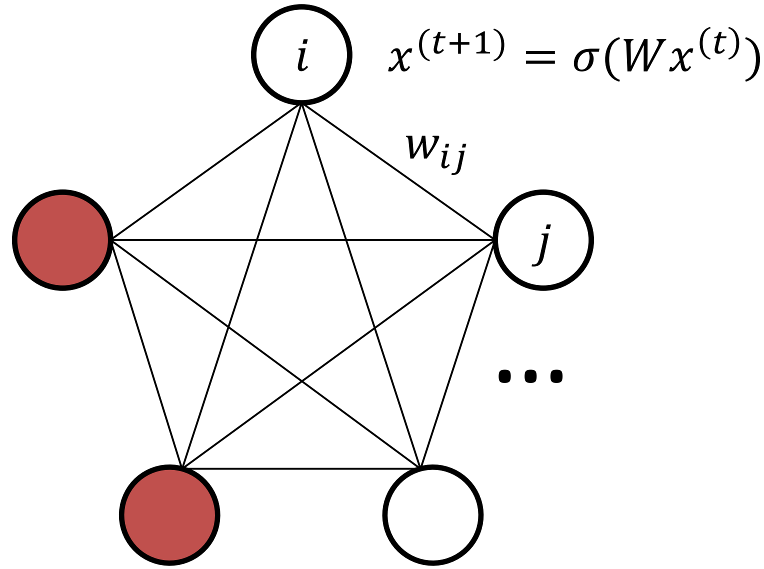
Modeling binary data

- Modelling with Boltzmann Machine

Name	Harry Potter	Avatar	LOTR3	Gladiator	Titanic	*Glitter*
Alice	1	1	1	0	0	0
Bob	1	0	1	0	0	0
Carol	1	1	1	0	0	0
David	0	0	1	1	1	0
Eric	0	0	1	1	0	0
Fred	0	0	1	1	1	0

Prefer SF/fantasy

Prefer Oscar winner



Hidden nodes

Visible nodes

- w_{ij} represents a correlation between nodes
- $p(v) = \sum_h p(h)p(v|h)$

Boltzmann Machine

- Probability distribution on binary vectors x

$$P(x) = \frac{\exp\big(-E(x)\big)}{Z}$$

$$E(x) = -\frac{1}{2}x^T W x - \theta^T x$$

$$= -\sum_{k<j} x_k w_{kj} x_j - \sum_k \theta_k x_k$$

  - From the entropy maximization

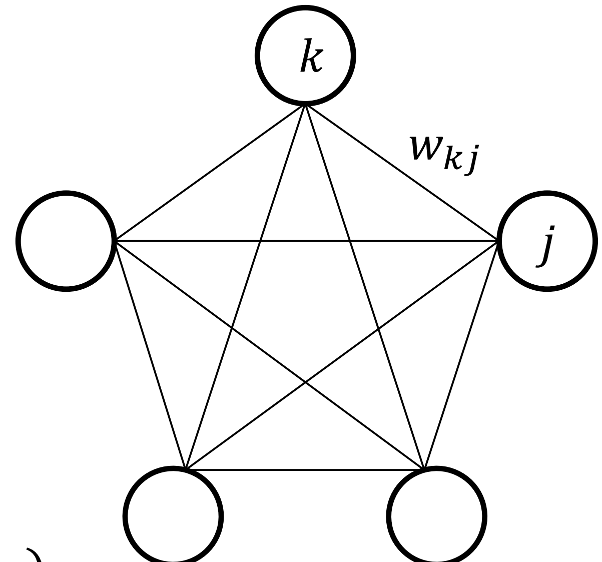
$$\max_{P(x)} - \sum_x P(x) \ln P(x)$$

$$s.t \sum_x P(x) = 1, \alpha = \sum_x P(x) E(x)$$

  - Z is the partition function that ensures $\sum_x P(x) = 1$

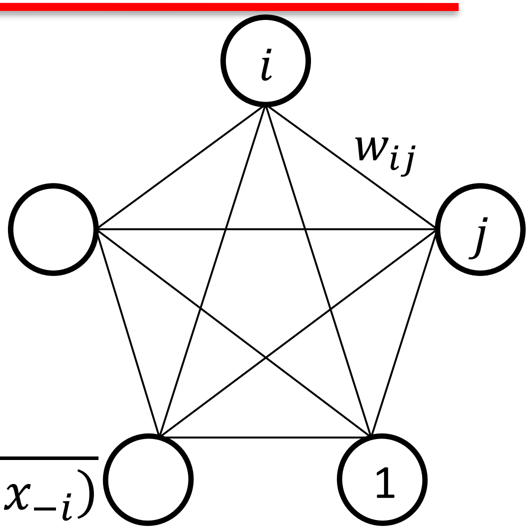
$$Z = \sum_x \exp\big(-E(x)\big)$$

$$x^{(t+1)} = \sigma(W x^{(t)})$$



Boltzmann Machine

$$x^{(t+1)} = \sigma(Wx^{(t)})$$



- Probability distribution on binary vectors x

$$P(x) = \frac{\exp(-E(x))}{Z}$$

- $E(x) = -\sum_{k<j} x_k w_{kj} x_j - \sum_k \theta_k x_k$

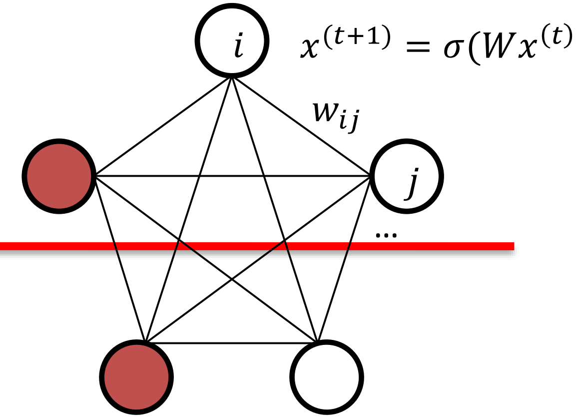
- Gibbs Sampling

$$P(x_i = 1 | x_{-i}) = \frac{P(x_i = 1,\ x_{-i})}{P(x_i = 1,\ x_{-i}) + P(x_i = 0,\ x_{-i})}$$

$$= \frac{exp\big(-E(x_i = 1,\ x_{-i})\big)}{exp\big(-E(x_i = 1,\ x_{-i})\big) + exp\big(-E(x_i = 0,\ x_{-i})\big)}$$

$$= \frac{1}{1 + exp\big(-E(x_i = 0,\ x_{-i}) + E(x_i = 1,\ x_{-i})\big)}$$

$$= \frac{1}{1 + \exp(-\sum_{j\neq i} w_{ij} x_j - \theta_i)} = \sigma\Big(\sum_{j\neq i} w_{ij} x_j + \theta_i\Big)$$



Hidden nodes Visible nodes

Restricted Boltzmann Machine

- Variant of Boltzmann Machine
- Restrict the connectivity to make **learning easier**
  - There is a hidden layer and visible layer
  - No hidden-to-hidden or visible-to-visible connections
  - Hidden units extends the class of distributions that can be modeled

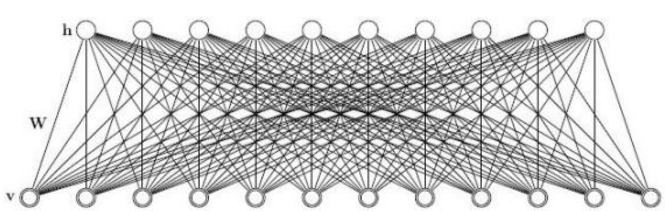
Bias of RBM

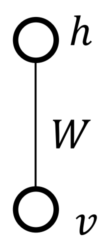
- Energy function

$$E(v,h) = -\sum_{\forall i,j} v_i w_{ij} h_j - \sum_i b_i^v v_i - \sum_j b_j^h h_j = -v^T W h - v^T b^v - h^T b^h$$

  - Vectors h, v are of dimension $J \times 1$ and $I \times 1$
  - W is of dimension $I \times J$

- $p(v) = \sum_h p(h) p(v|h)$





$$h = \sigma(Wv)$$

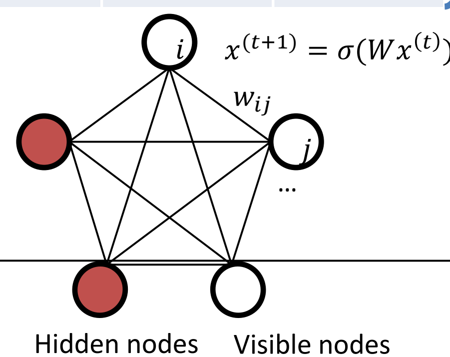
Modeling binary data

- Given a training set of binary vectors, fit a model that will assign a probability to other binary vectors

Name	Harry Potter	Avatar	LOTR3	Gladiator	Titanic	Glitter
Alice	1	1	1	0	0	0
Bob	1	0	1	0	0	0
Carol	1	1	1	0	0	0
David	0	0	1	1	1	0
Eric	0	0	1	1	0	0
Fred	0	0	1	1	1	0

Prefer SF/fantasy

Prefer Oscar winner



Restricted Boltzmann Machine

- Marginal distribution $P(v)$

$$P(v) = \sum_h P(h)P(v|h) = \sum_h P(v,h) = \frac{\sum_h \exp\big(-E(v,h)\big)}{Z}$$

  - $P(v,h)$ is a Boltzmann distribution with energy function $E(v,h)$

  - And $P(v)$ is a Boltzmann distribution with a *energy* $F(v)$

$$P(v) = \frac{\exp\big(-F(v)\big)}{Z}$$

$$F(v) = -\ln \sum_h \exp\big(-E(v,h)\big)$$

  - the energy $F(v)$ cannot be represented as a quadratic form in v (Why?)

RBM Learning

- Maximize the product of probabilities assigned to training set V

$$\arg\max_{W} \prod_{v \in \mathrm{V}} P(v)$$

- Or equivalently, maximize the sum of log probability of V:

$$\arg\max_{W} \sum_{v \in \mathrm{V}} \ln P(v)$$

- The model is updated after each training token or in batch mode

$$w_{ij} \leftarrow w_{ij} + \alpha \frac{\partial \ln P(v)}{\partial w_{ij}}\Big|_{v=v^1}$$

RBM Learning

$$P(v) = \frac{\exp\big(-F(v)\big)}{Z}$$

$$F(v) = -\ln \sum_h \exp\big(-E(v,h)\big)$$

- Stochastic gradient ascent
  - Calculate the gradient of the log likelihood, given a training token v^1

$$\left.\frac{\partial \ln P(v)}{\partial w_{ij}}\right|_{v=v^1} = -\left.\frac{\partial F(v)}{\partial w_{ij}}\right|_{v=v^1} - \frac{\partial \ln Z}{\partial w_{ij}}$$

$$= v_i^1 h_j^1 - \frac{\partial}{\partial w_{ij}} \ln \sum_v \exp\big(-F(v)\big)$$

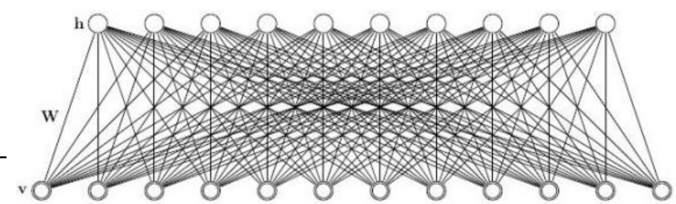
$$= v_i^1 h_j^1 - \frac{1}{\sum_v \exp\big(-F(v)\big)} \sum_v \exp\big(-F(v)\big) \frac{\partial F(v)}{\partial w_{ij}}$$

$$= v_i^1 h_j^1 - \frac{1}{Z} \sum_v \exp\big(-F(v)\big) v_i h_j$$

$$= v_i^1 h_j^1 - \textstyle\sum_v P(v) v_i h_j$$

Expectation of $v_i h_j$

$$= v_i^1 h_j^1 - \langle v_i h_j \rangle_{model}$$



RBM Learning

- Stochastic gradient ascent

$$F(v) = -\ln \sum_h \exp\big(-E(v,h)\big)$$

$$E(v,h) = -\sum_{\forall i,j} v_i w_{ij} h_j$$

$$\frac{\partial F(v)}{\partial w_{ij}} = -\frac{\partial}{\partial w_{ij}} \ln \sum_h \exp\big(-E(v,h)\big)$$

$$= -\frac{1}{\cancel{\sum_h} \exp\big(-E(v,h)\big)} \cancel{\sum_h} \exp\big(-E(v,h)\big)\left(-\frac{\partial E(v,h)}{\partial w_{ij}}\right)$$

$$= -v_i h_j \qquad \text{for fixed } v, h$$

RBM Learning

$$\left.\frac{\partial \ln P(v)}{\partial w_{ij}}\right|_{v=v^1} = v_i^1 h_j^1 - \langle v_i h_j \rangle_{model}$$

- If there are K iid training tokens $v^1, \dots, v^K$

$$\frac{\partial}{\partial w_{ij}} \sum_k \ln P(v^k) = \sum_k \frac{\partial \ln P(v^k)}{\partial w_{ij}}$$
$$= \left(v_i^1 h_j^1 + \cdots + v_i^K h_j^K - K \langle v_i h_j \rangle_{model}\right)$$

- So that…

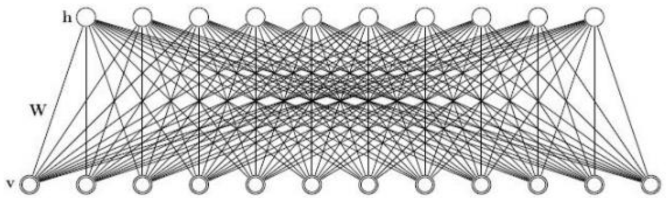
$$\frac{\partial}{\partial w_{ij}} \mathbb{E}_v[\ln P(v)] \approx \frac{\partial}{\partial w_{ij}} \frac{1}{K} \sum_k \ln P(v^k) = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}$$

Data statistics

Model statistics

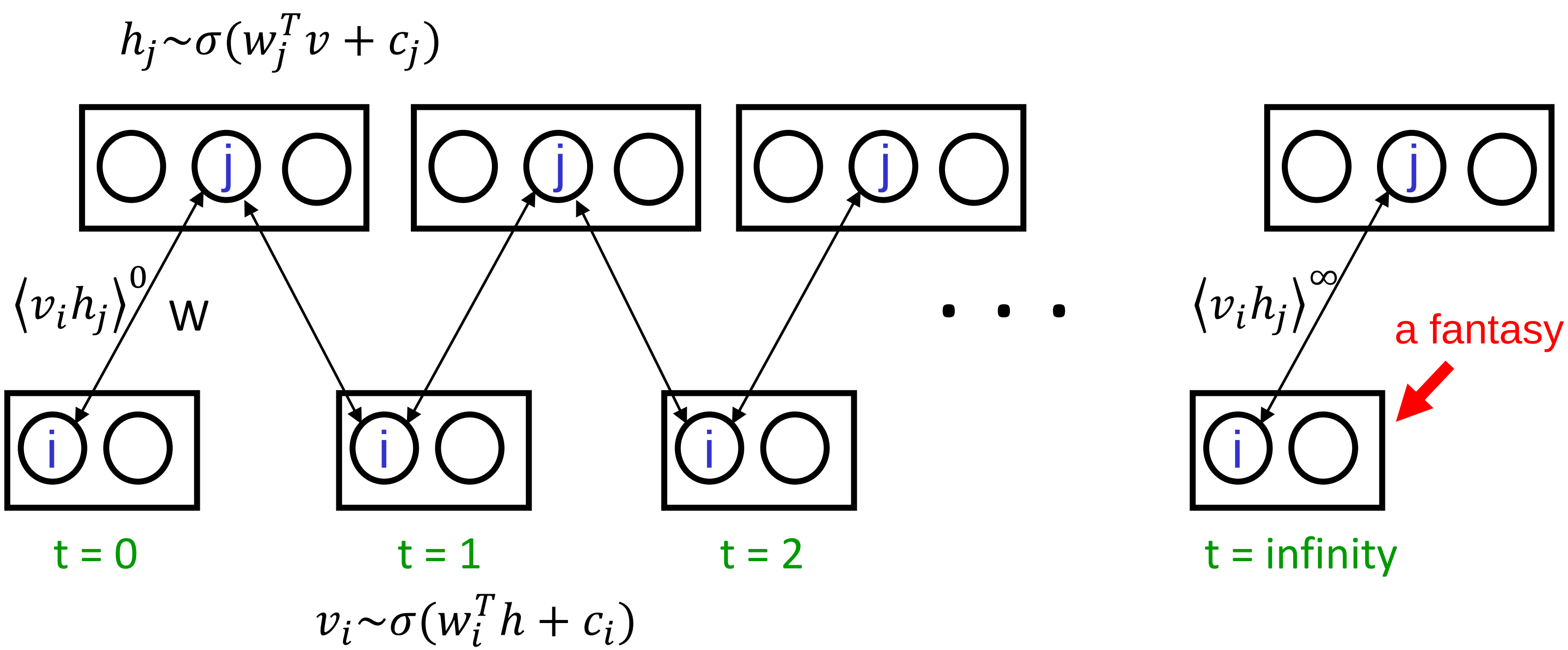
: *unknown*

- $\Delta w_{ij} = \eta(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model})$



Model statistics

- $\langle v_i h_j \rangle_{model}$ can be estimated by using any MCMC algorithm
  - But nobody knows t_{conv} which indicates the step at which $\langle v_i h_j \rangle$ converges



Model statistics

- Contrast Divergence (CD) [Bengio, et al.]: Starting at the given training token $v^{(1)}, h^{(1)}$, run the Markov chain for n steps:
  - $v^{(1)}, h^{(1)} \rightarrow \cdots \rightarrow v^{(n+1)}, h^{(n+1)}$
  - With the edge weight $[w_{ij}]$

- And we can approximate

$$\left.\frac{\partial \ln P(v)}{\partial w_{ij}}\right|_{v=v^1} \approx v_i^{(1)} h_j^{(1)} - \underbrace{v_i^{(n+1)} h_j^{(n+1)}}_{\text{CD-}n}$$

- **CD-1** $\rightarrow$ *weight change* $\rightarrow$ **CD-3** $\rightarrow \ldots \rightarrow$ **CD-5** $\rightarrow \ldots \rightarrow$ **CD-7** $\ldots$ **CD-9**

Example of RBM

- Train the RBM using following data (with CD-1)
  - 6 visible units (each movies) with 2 hidden units

Name	Harry Potter	Avatar	LOTR3	Gladiator	Titanic	Glitter
Alice	1	1	1	0	0	0
Bob	1	0	1	0	0	0
Carol	1	1	1	0	0	0
David	0	0	1	1	1	0
Eric	0	0	1	1	0	0
Fred	0	0	1	1	1	0

Prefer SF/fantasy

Prefer Oscar winner

Example of RBM

- And… the network is trained by the following weights:
  - $W = \begin{bmatrix} 4.97 & 2.27 & 4.11 & -4.01 & -5.60 & -2.92 \\ -7.09 & -5.18 & 2.52 & 6.75 & 3.25 & -2.82 \end{bmatrix}$

Name	Harry Potter	Avatar	LOTR3	Gladiator	Titanic	Glitter	
Alice	1	1	1	0	0	0	Prefer SF/fantasy
Bob	1	0	1	0	0	0	
Carol	1	1	1	0	0	0	
David	0	0	1	1	1	0	Prefer Oscar winner
Eric	0	0	1	1	0	0	
Fred	0	0	1	1	1	0	

- The **first hidden unit** seems to correspond to the **SF/fantasy** , and the **second hidden unit** seems to correspond to the **Oscar winners** movies
- If the RBM is presented to a new user, George, who has $[0,0,0,1,1,0]$ as his preferences, then It turns the second hidden unit on

Persistent CD

- A set of samples $v^1, \dots v^K$ is drawn(observed) from the model distribution
  - The set is maintained and updated whenever the model is updated
  - K Markov chains are run in parallel and, on every update, several steps of Gibbs sampling are performed in each chain
  - The model statistics are derived by averaging over the samples:

$$\langle v_i h_j \rangle_{model} = \frac{1}{K} \sum_k v_i^{k,(n+1)} \, h_j^{k,(n+1)}$$

- Persistent CD generally works better than CD

Interim Summary

- Boltzmann machines try to model a realistic brain learning mechanism (unsupervised model).
- Boltzmann machines and Restricted Boltzmann machines are based on the energy model
- Undirected Graph model such as Markov random field
- The RBM is the simple type of Boltzmann machine, and it can be easily learned
  - We use the Contrastive Divergence (CD) to train the RBM
- Persistent Contrastive Divergence is the improved version of CD, and it lesson the problem that CD does not guarantee the fast convergence