Density Estimation

Jin Young Choi

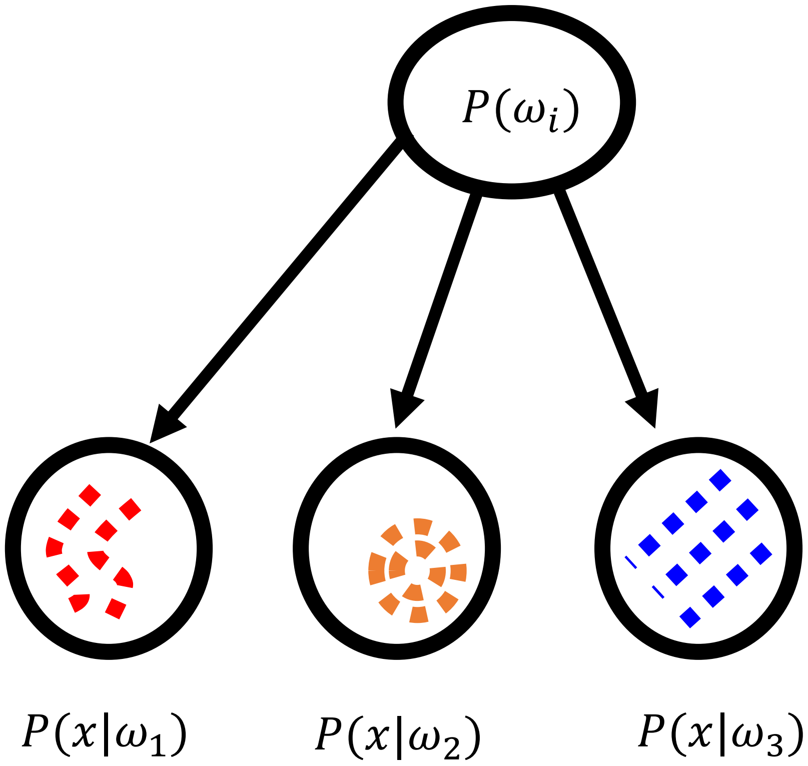
Seoul National University

Outline

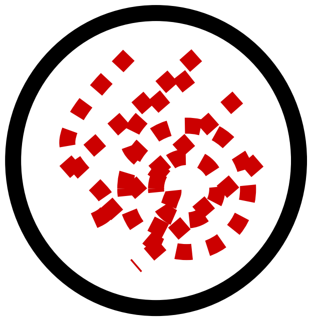
- Parametric Density Estimation
  - Maximum Likelihood Estimation
  - Bayesian Learning
- Nonparametric Density Estimation
  - Histogram
  - K_n Nearest Neighbor Estimation
  - Parzen Window Estimation
  - Gaussian Mixture Estimation
    - Expectation-Maximization
    - Markov-Chain Monte Carlo

Learning From Observed Data

<u>Hidden</u>

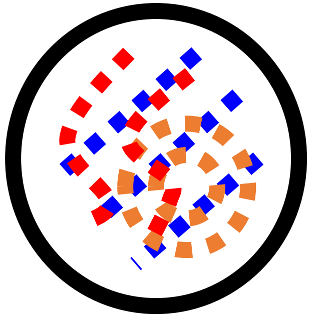


<u>Observed</u>



Unsupervised

Clustering
Auto-encoding



Supervised

Likelihood Estimation
Error backpropagation

Parametric Learning

- Assume specific parametric distributions with parameters:
  $p(x|\omega_i) \approx p(x|\theta_i), \theta_i \in \Theta \subset R^p$
- Estimate parameters $\hat{\theta}(D)$ from training data D
- Replace true value of class-conditional density with approximation and apply the Bayesian framework for decision making.

Parametric Learning

- Suppose we can assume that the relevant (class-conditional) densities are of some parametric form. That is,

$$p(x|\omega) \approx p(x|\theta), where\ \theta \in \Theta \in R^p$$

- Examples of parameterized densities:
  - Binomial: x has m 1's and $(n-m)$ 0's

$$p(x|\theta) = \binom{n}{m} \theta^m (1-\theta)^{n-m}, \Theta = [0,1]$$

  - Exponential: Each data point x is distributed according to

$$p(x|\theta) = \theta e^{-\theta x}, \Theta = (0, \infty)$$

  - Normal:

$$p(x|\theta) \sim N(\mu, \sigma^2)$$

- Multinomial pmf : $\Omega = \{k_1, \cdots, k_m |\ k_i = 0,\ 1,\ 2, \cdots, n\}$

$$p(k_1, \cdots, k_m) = \begin{cases} \frac{n!}{k_1! \cdots k_m!} p_1^{k_1} \cdots p_m^{k_m}, & k_i = 0,1, \cdots, n \\ 0, & \text{else} \end{cases}$$

Parametric Learning

- Bayesian Decision

$$arg \max_i p(\omega_i|x) \propto p(x|\omega_i)p(\omega_i)$$

- Maximum Likelihood Estimation

$$arg \max_{\theta_i} p(D|\theta_i) \leftarrow p(x|\omega_i) \approx p(x|\theta_i), \theta_i \in \Theta \subset R^p$$

- Maximum A-Posteriori Estimation

$$arg \max_{\theta_i} p(\theta_i|D), \theta_i = constant$$

- Bayesian Learning (not Estimation)

Find $p(\theta_i|D), \theta_i = random\ variable$

Maximum Likelihood Estimation

- The samples are i.i.d.

$$j^{th}\ class\ set\ D_j = \{x_l | (x_l, \bar{\omega}_l) \in S_j\},\ S_j \in S = \{(x_l, \bar{\omega}_l) | l = 1, \dots, N\}$$

- The i.i.d. assumption implies that

$$p(D_j|\theta_j) = \prod_{x \in D_j} p(x|\theta_j)$$

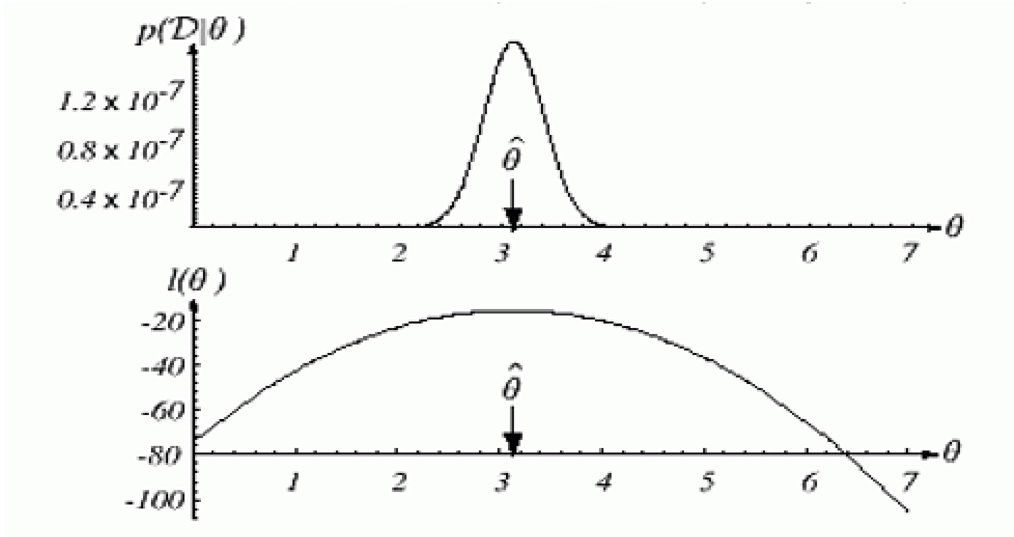
- Let D be a generic sample set of size $n = |D|$
- Log-likelihood function:

$$l(\theta; D) \equiv \ln p(D|\theta) = \sum_{k=1}^{n} \ln p(x_k|\theta)$$

- The log-likelihood function is identical to the logarithm of the probability density (or mass) function, but is interpreted as a function of parameter θ

Log-Likelihood Illustration

- $p(D|\theta)$ is not convex, but $\ln p(D|\theta)$ is convex (quadratic) for normal dist.



$$p(D|\theta) = \prod_{x \in D} p(x|\theta)$$

$$l(\theta; D) \equiv \ln p(D|\theta) = \sum_{k=1}^{n} \ln p(x_k|\theta)$$

Maximum Likelihood Estimation (MLE)

- The “most likely value” for MLE is given by

$$\hat{\theta}(D) = \underset{\theta \in \Theta}{argmax}\, l(\theta; D)$$

- Gradient function

$$\nabla_\theta l(\theta; D) = \left[\frac{\partial l(\theta;D)}{\partial \theta_1}, \ldots, \frac{\partial l(\theta;D)}{\partial \theta_p}\right]$$

- Necessary condition for MLE (if not on border of domain Θ)

$$\nabla_\theta l(\theta; D) = 0$$

Example of Maximum Likelihood

The Gaussian Case:

$$p(x) = \frac{1}{\sqrt{(2\pi)^d}\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

- Log-likelihood is

$$l(\mu, \Sigma; D) = \ln p(D|\mu, \Sigma) = \ln \prod_{k=1}^{n} p(x_k|\mu) = \sum_{k=1}^{n} \ln p(x_k|\mu)$$

where

$$\ln p(x_k|\mu, \Sigma) = -\frac{1}{2}(x_k - \mu)^T\Sigma^{-1}(x_k - \mu) - \frac{d}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma|$$

- For a sample point x_k, we have

$$\nabla_\mu \ln p(x_k|\mu, \Sigma) = \Sigma^{-1}(x_k - \mu)$$

- The maximum likelihood estimate for μ must satisfy

$$\sum_{k=1}^{n} \Sigma^{-1}(x_k - \mu) = 0 \rightarrow \hat{\mu} = \frac{1}{n}\sum_{k=1}^{n} x_k \quad \text{(sample mean)}$$

Example of Maximum Likelihood

The Gaussian Case:

- Log-likelihood is

$$l(\mu, \Sigma; D) = \ln p(D|\mu, \Sigma) = \ln \prod_{k=1}^{n} p(x_k|\mu) = \sum_{k=1}^{n} \ln p(x_k|\mu)$$

where $\ln p(x_k|\mu, \Sigma) = -\frac{1}{2}(x_k - \mu)^t \Sigma^{-1}(x_k - \mu) - \frac{d}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma|$

$$\boxed{\nabla_\Sigma \ln|\Sigma| = \frac{1}{|\Sigma|} adj(\Sigma) = \Sigma^{-1}}$$

- For a sample point x_k, we have

$$\nabla_\Sigma \ln p(x_k|\mu, \Sigma) = \frac{1}{2}\Sigma^{-2}(x_k - \mu)(x_k - \mu)^t - \frac{1}{2}\Sigma^{-1} = 0$$

- The maximum likelihood estimate for Σ must satisfy

$$\hat{\Sigma} = \frac{1}{n}\sum_{k=1}^{n}(x_k - \hat{\mu})(x_k - \hat{\mu})^t$$ (sample covariance)

Example of Maximum Likelihood

The Gaussian case

- For the multivariate case, it is easy to show that the MLE estimates are given by

$$\hat{\mu} = \frac{1}{n}\sum_{k=1}^{n} x_k \,, \qquad \hat{\Sigma} = \frac{1}{n}\sum_{k=1}^{n}(x_k - \hat{\mu})(x_k - \hat{\mu})^t$$

- The MLE for Σ is biased

$$E\left[\frac{1}{n}\sum_{k=1}^{n}(x_k - \hat{\mu})(x_k - \hat{\mu})^t\right] = \frac{n-1}{n}\Sigma \neq \Sigma$$

- Unbiased estimate

$$\hat{\Sigma} = \frac{1}{n-1}\sum_{k=1}^{n}(x_k - \hat{\mu})(x_k - \hat{\mu})^t$$

$$E(\boldsymbol{e}^T\boldsymbol{e}) = Tr(\mathbf{I} - \mathbb{H})E(\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}) = (n-p-1)n\sigma^2$$
$$\rightarrow E(\boldsymbol{e}^T\boldsymbol{e}/(n-p-1)) = n\sigma^2$$
$$E(\boldsymbol{e}\boldsymbol{e}^T) = Tr(\mathbf{I} - \mathbb{H})E(\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T) = (n-p-1)\sigma^2\mathbf{I}$$
$$\rightarrow E(\boldsymbol{e}\boldsymbol{e}^T/(n-p-1)) = \sigma^2\mathbf{I}$$

Bayesian Learning
– Univariate Normal Distribution

- Let μ be the only unknown parameter

$$p(x|\mu) \sim N(\mu, \sigma^2)$$

- Goal: to find posterior $p(\mu|D)$ from $i.i.d.\, D = \{x_1, \dots, x_n\}$, where

$$p(\mu|D) = \frac{p(D|\mu)p(\mu)}{\int p(D|\mu)p(\mu)d\mu}$$

$$= \alpha \prod_{k=1}^{n} p(x_k|\mu)p(\mu).$$

- conjugate prior probability for μ is given by

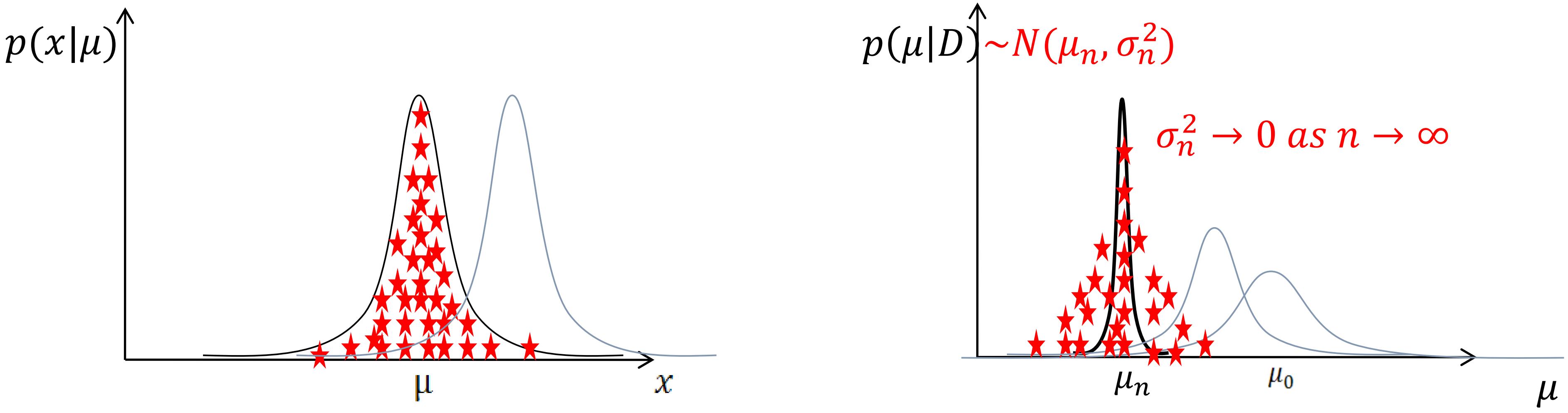
$$p(\mu) \sim N(\mu_0, \sigma_0^2) \rightarrow p(\mu|D) \sim N(\mu_n, \sigma_n^2)$$

Maximum Likelihood vs Bayesian Learning

- Maximum likelihood estimation: find $\hat{\mu}(D)$ to maximize $p(x|D)$

$$p(x|D) \approx p(x|\hat{\mu}(D)), \hat{\mu}(D) = arg \max_{\mu} p(D|\mu)$$

- Bayesian learning: find $p(\theta|D)$.



Bayesian Learning

– Univariate Normal Distribution

- Computing the posterior distribution

$$p(\mu|D) = \alpha\, p(D|\mu)p(\mu)$$

$$p(\mu|D) = \alpha \prod_{k=1}^{n} p(x_k|\mu)p(\mu)$$

$$p(\mu) \sim N(\mu_0, \sigma_0^2)$$

$$= \alpha' \exp\left[-\frac{1}{2}\left(\sum_{k=1}^{n}\left(\frac{(x_k-\mu)}{\sigma}\right)^2 + \left(\frac{(\mu-\mu_0)}{\sigma_0}\right)^2\right)\right]$$

$$= \alpha'' \exp\left[-\frac{1}{2}\left(\left(\frac{n}{\sigma^2} + \frac{1}{{\sigma_0}^2}\right)\mu^2 - 2\left(\frac{1}{\sigma^2}\sum_{k=1}^{n} x_k + \frac{\mu_0}{\sigma_0^2}\right)\mu\right)\right]$$

- Since $p(\mu|D)$ should be Gaussian

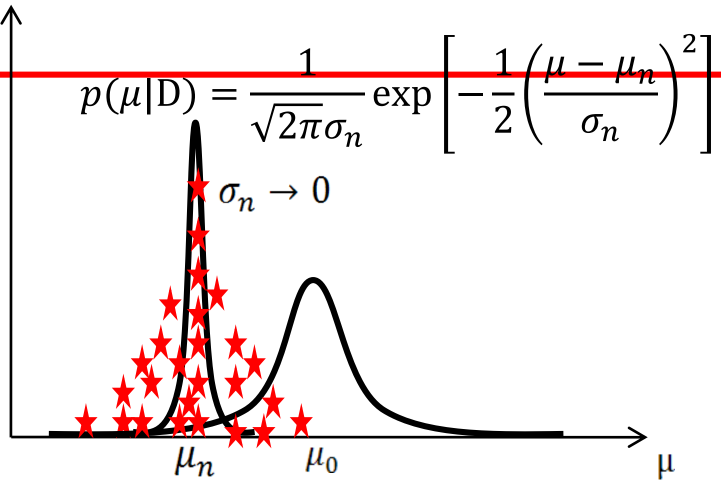
$$p(\mu|D) \sim N(\mu_n, \sigma_n^2)$$

$$p(\mu|D) = \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left[-\frac{1}{2}\left(\frac{(\mu-\mu_n)}{\sigma_n}\right)^2\right] = \alpha''' \exp\left[-\frac{1}{2\sigma_n^2}\mu^2 - \frac{\mu_n}{\sigma_n}\mu\right]$$

- $\mu_n = f(x_k, \mu_0, \sigma_0) = ?, \quad \sigma_n = h(x_k, \mu_0, \sigma_0) = ?$

Bayesian Learning

– Univariate Normal Distribution



- Solution: $\frac{1}{\sigma_n^2} = \frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}$, $\frac{\mu_n}{\sigma_n^2} = \frac{n}{\sigma^2}\hat{\mu}_n + \frac{\mu_0}{\sigma_0^2}$

  where $\hat{\mu}_n = \frac{1}{n}\sum_{k=1}^{n} x_k$ is the sample mean.

- Solving explicitly for μ_n and σ_n^2 we obtain

$$\boxed{\sigma_n^2 = \frac{\sigma_0^2\sigma^2}{n\sigma_0^2+\sigma^2}, \qquad \mu_n = \left(\frac{n\sigma_0^2}{n\sigma_0^2+\sigma^2}\right)\hat{\mu}_n + \frac{\sigma^2}{n\sigma_0^2+\sigma^2}\mu_0}$$

- μ_n represents our best guess for μ after observing n samples.
- σ_n^2 measures our uncertainty about this guess.
- σ_n^2 decreases monotonically with n

  (approaching $\sigma_n^2 \to \frac{\sigma^2}{n} \to 0$ as n approaches infinity)

Bayesian Learning

– Univariate Normal Distribution

- Likelihood Estimation:

$$p(\mathbf{x}|\omega) \approx p(x|D) = \int p(x|\mu)p(\mu|D)d\mu$$

$$= \int \frac{1}{2\pi\sigma\sigma_n} \exp\left[-\frac{1}{2}\left(\frac{(x-\mu)}{\sigma}\right)^2 - \frac{1}{2}\left(\frac{(\mu-\mu_n)}{\sigma_n}\right)^2\right] d\mu$$

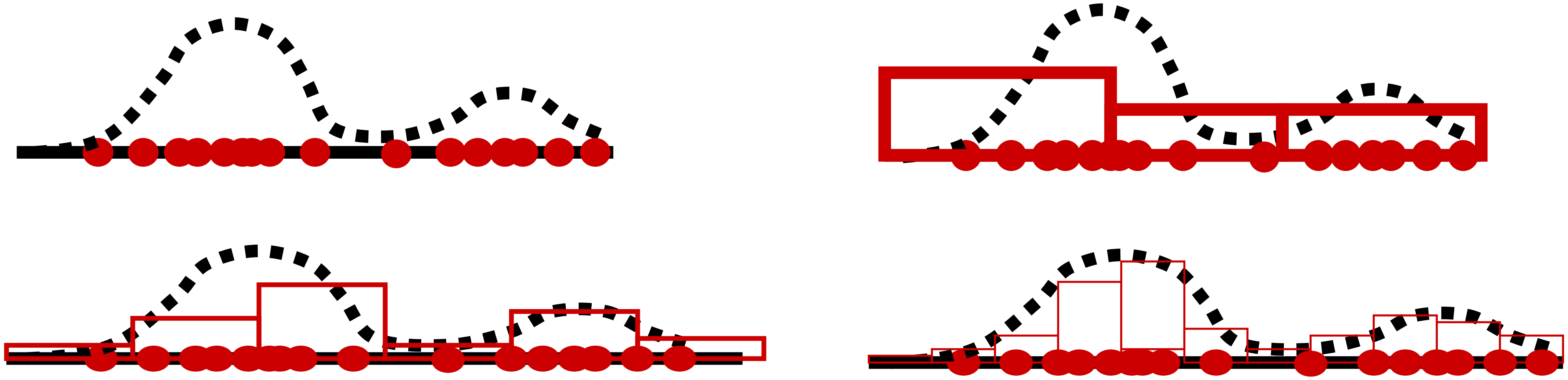
$$= \frac{1}{2\pi\sigma\sigma_n} \exp\left[-\frac{1}{2}\frac{(x-\mu_n)^2}{\sigma^2+\sigma_n^2}\right] \frac{\sqrt{2\pi}\sigma\sigma_n}{\sqrt{\sigma^2+\sigma_n^2}}$$

$$= \frac{1}{\sqrt{2\pi}\sqrt{\sigma^2+\sigma_n^2}} \exp\left[-\frac{1}{2}\frac{(x-\mu_n)^2}{\sigma^2+\sigma_n^2}\right]$$

$$\xrightarrow[n\to\infty]{} \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right]$$

Nonparametric Density Estimation

- The form of $p(x|\omega_i)$ to be estimated is not assumed.
- Naïve approach is Histogram



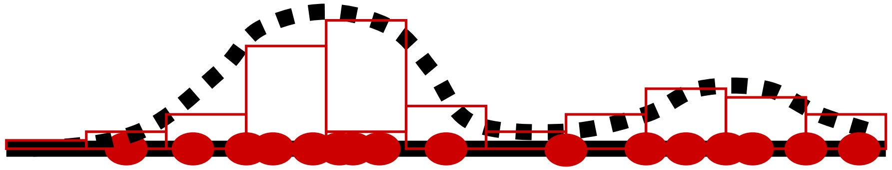
$$p = \int_R p(\mathbf{x}')d\,\mathbf{x}' \rightarrow P_k = \binom{n}{k} p^k (1-p)^{n-k} \rightarrow E[k] = np \quad \rightarrow \mathrm{var}(k) = np(1-p)$$

$$E\left[\frac{k}{n}\right] = p, \qquad \mathrm{var}\left[\frac{k}{n}\right] = \frac{p(1-p)}{n} \qquad p = \int_R p(\mathbf{x}')d\,\mathbf{x}' \approx p(x)V \rightarrow p(x) = p/V \approx \frac{k/n}{V}$$

Three Conditions for Density Estimation

- Reducing the region by increasing the samples
- Let us take a growing sequence of samples $n = 1, 2, 3 \ldots$
- We take regions R_n with reduced volumes $V_1 > V_2 > V_3 > \cdots$
- Let k_n be the number of samples falling in R_n
- Let $p_n(x)$ be the n^{th} estimate for $p(x)$
- If $p_n(x)$ is to converge to $p(x)$, 3 conditions must be required:
  - $\lim_{n\to\infty} V_n = 0,$ resolution as big as possible (for smoothing)
  - $\lim_{n\to\infty} k_n = \infty,$ to preserve $\int p(x) d\,x = 1$
  - $\lim_{n\to\infty} \frac{k_n}{n} = 0$ to guarantee convergence of $p(x) \approx p_n(x) = \frac{k/n}{V}$

$$\int \hat{p}(\mathbf{x}|\omega_i)d\mathbf{x} = \sum_{j=1}^{m}\int_{b_j}\frac{k_j}{n_i V}d\mathbf{x} = \frac{1}{n_i}\sum_{j=1}^{m}k_j = 1$$



PARZEN WINDOW and KNN

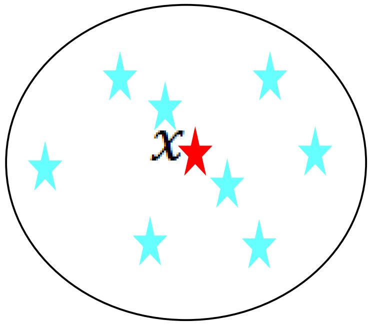
- How to obtain the sequence $R_1, R_2, ..$?
- There are 2 common approaches of obtaining sequences of regions that satisfy the convergence conditions:
  - Specify k_n as some function of n, such as $k_n = \sqrt{n}$. Here the volume V_n is grown until it encloses k_n neighbors of x.
  - This is k_n –nearest-neighbor method.
  - Shrink an initial region by specifying the volume V_n as some function of n, such as $V_n = 1/\sqrt{n}$ and show that k_n and k_n/n behave properly i.e. $p_n(x)$ converges to $p(x)$.
  - This is Parzen-window (or kernel) method.

$$\lim_{n\to\infty} V_n = 0,$$
$$\lim_{n\to\infty} k_n = \infty,$$
$$\lim_{n\to\infty} \frac{k_n}{n} = 0$$

K_n-Nearest-Neighbor Estimation

- To estimate $p(x)$ from n training samples, we center a cell about x and let it grow until it captures k_n samples, where k_n is some specified function of n.
- These samples are the k_n nearest-neighbors of x.
- If the density is high near x, the cell will be relatively small good resolution.

$$\boxed{\lim_{n\to\infty} V_n = 0\,,\ \lim_{n\to\infty} k_n = \infty,\ \lim_{n\to\infty} k_n/n = 0} \to p(\mathbf{x}) \approx \frac{k_n/n}{V_n}$$

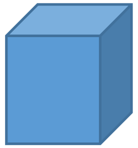


Let $k_n = \sqrt{n} \to V_n \approx 1/(\sqrt{n}p(x)) \to 0$

PARZEN WINDOWS

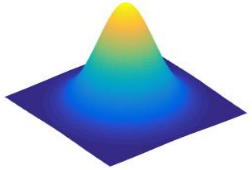
- Assume that the region R_n is a d –dimensional hypercube.
- If h_n is the length of an edge of that hypercube $\mathcal{H}(x)$ centered at x, then its volume is given by

$$V_n = h_n^d$$



$$V_n = 1/\sqrt{n}$$

$$h_n = 1/\sqrt[d]{n} \to 0$$



- Define the following window function:

$$\varphi(\mathbf{u}) = \begin{cases} 1 & |u_j| \le 1/2\,; \qquad j = 1, \dots, d \\ 0 & \text{otherwise.} \end{cases}$$

- $\varphi(\mathbf{u})$ defines a unit hypercube centered at the origin.

$$\varphi((\mathbf{x}-\mathbf{x}_i)/h_n) = \begin{cases} 1 & \mathbf{x}_i \in \mathcal{H}(x) \\ 0 & \text{otherwise.} \end{cases}$$

$$\boxed{\begin{gathered} -1/2 \le (\mathbf{x}_i - \mathbf{x})/h_n \le 1/2 \to -h_n/2 \le \mathbf{x}_i - \mathbf{x} \le h_n/2 \\ \to \mathbf{x} - h_n/2 \le \mathbf{x}_i \le \mathbf{x} + h_n/2 \end{gathered}}$$

- The number of samples in this hypercube is given by:

$$k_n = \sum_{i=1}^{n} \varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right) \to p_n(\mathbf{x}) = \frac{k_n}{nV_n} = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{V_n} \varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right).$$

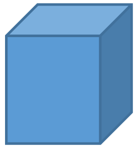
$$\boxed{\lim_{n\to\infty} V_n = 0\,, \ \lim_{n\to\infty} k_n = \infty, \ \lim_{n\to\infty} k_n/n = 0}$$

$$\because\ k_n = nV_n p_n(\mathbf{x}) = \sqrt{n}\, p_n(\mathbf{x}), \quad k_n/n = p_n(\mathbf{x})/\sqrt{n}$$

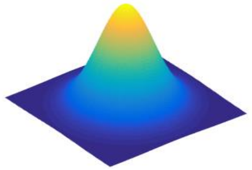
Gaussian Mixture Estimation

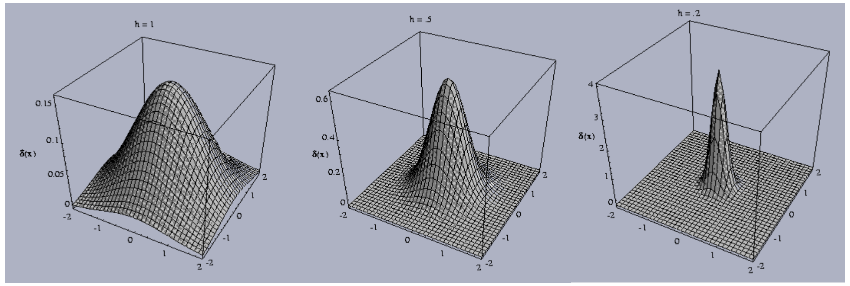
- Parzen Window

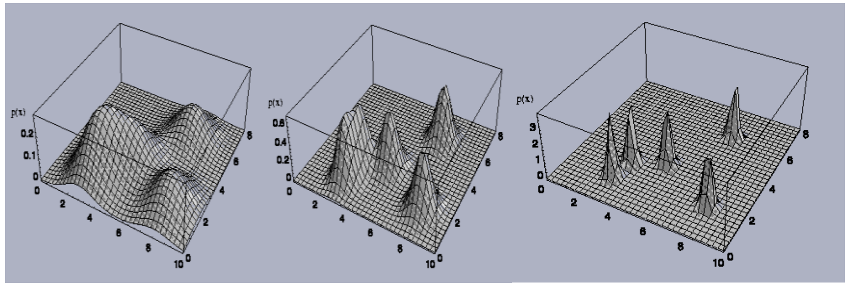
$$p_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{V_n}\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)$$



$$\varphi(\mathbf{u}) \geq 0 \quad \text{and} \quad \int \varphi(\mathbf{u})\ d\mathbf{u} = 1$$





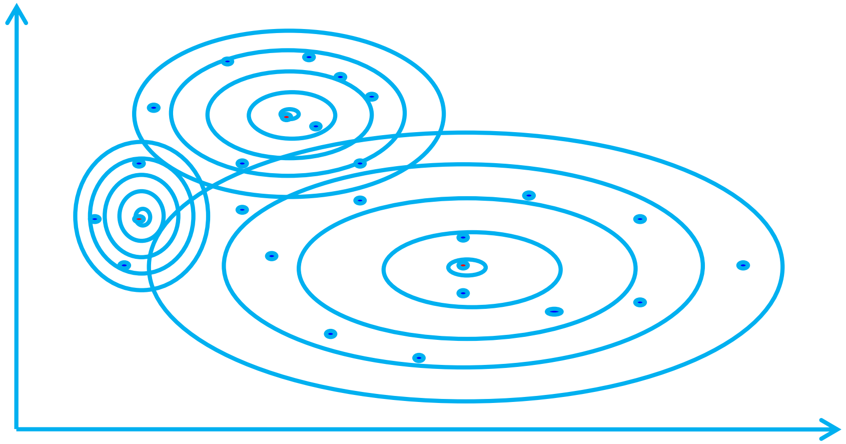


- Gaussian Mixture

$$p(\mathbf{x}|\theta) = \sum_{k=1}^{K} w_k \varphi\left(\frac{\mathbf{x}-\mu_k}{\sigma_k}\right)$$

$$p(\mathbf{x}|\theta) = \sum_{k=1}^{K} p(\mathbf{x}|\theta_k)p(\theta_k|\theta)$$

$$= \sum_{k=1}^{K} p(\mathbf{x}|k)p(k|\theta) = \sum_{k=1}^{K} p(\mathbf{x},k|\theta)$$



Expectation-Maximization (EM)

- EM aims to find parameter values that maximize likelihood,

$$L(\theta;\mathbf{x}) = p(\mathbf{x}|\theta) = \Sigma_k p(\mathbf{x},k|\theta) = \Sigma_k\ L(\theta\,;\mathbf{x}\,,k)$$

where k is a latent variable.

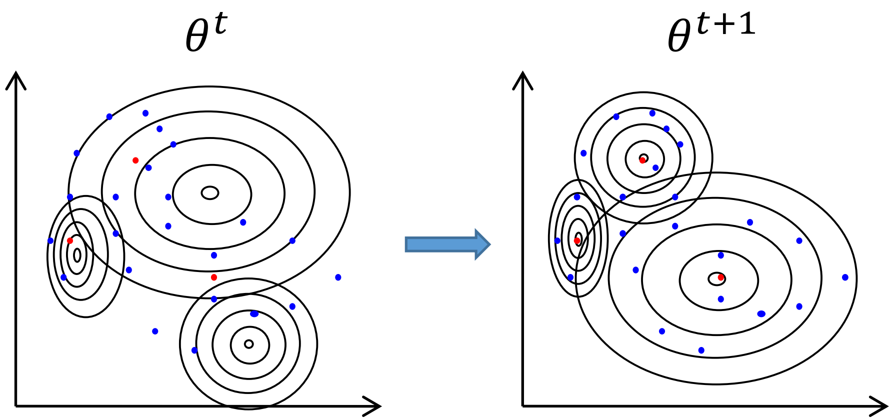
- **E-step:** For given θ^t, $\mathbf{x}$, find expectation of the likelihood on the conditional distribution of k.

$$Q(\theta|\theta^t) = E_{k|\mathbf{x},\theta^t}[\log L\,(\theta\,;\mathbf{x}\,,k)] = \Sigma_k p(k|\mathbf{x},\theta^t)\log L\,(\theta\,;\mathbf{x}\,,k)$$

- **M-step:** Find θ^{t+1} maximizing Q.

$$\theta^{t+1} = \underset{\theta}{\operatorname{argmax}} Q(\theta|\theta^t)$$

- Repeat E-step and M-step.



Expectation-Maximization (EM)

- **E-step**

$$Q(\theta|\theta^t) = E_{k|\mathbf{x},\theta^t}[\log L(\theta\,;\mathbf{x}\,,k)] = \Sigma_k p(k|\mathbf{x},\theta^t)\log L(\theta\,;\mathbf{x}\,,k)$$

$$T_{k,m}^t := p(k|\mathbf{x}=x_m,\theta^t) = \frac{p(x_m\,|k_k^t,\Sigma_k^t)\tau_k^t}{\Sigma_k\, p(x_m\,;\mu_k^t,\Sigma_k^t)\tau_k^t},$$

$$p(x_m\,|\mu_k^t,\Sigma_k^t) = \frac{1}{(2\pi)^{d/2}\sqrt{|\Sigma_k^t|}}\exp\left(-\frac{(x_m-\mu_k^t)^T{\Sigma_k^t}^{-1}(x_m-\mu_k^t)}{2}\right)$$

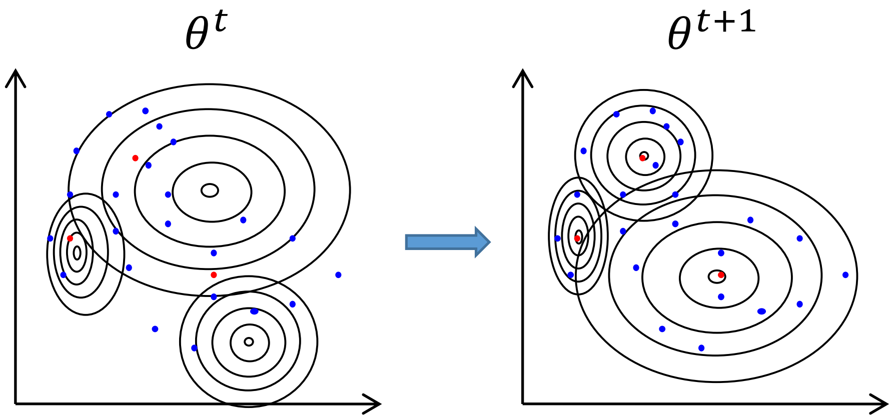
$$Q(\theta|\theta^t) = \Sigma_m\Sigma_k T_{k,m}^t\left(\log\tau_k - \frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x_m-\mu_k)^T\Sigma_k^{-1}(x_m-\mu_k)\right)$$

- **M-step**

$$\tau_k^{t+1} = \frac{\Sigma_m T_{k,m}^t}{\Sigma_k\Sigma_m T_{k,m}^t},$$

$$\mu_k^{t+1} = \frac{\Sigma_m T_{k,m}^t x_m}{\Sigma_k\Sigma_m T_{k,m}^t},$$

$$\Sigma_k^{t+1} = \frac{\Sigma_m T_{k,m}^t(x_m-\mu_k^{t+1})(x_m-\mu_k^{t+1})^T}{\Sigma_k\Sigma_m T_{k,m}^t}$$



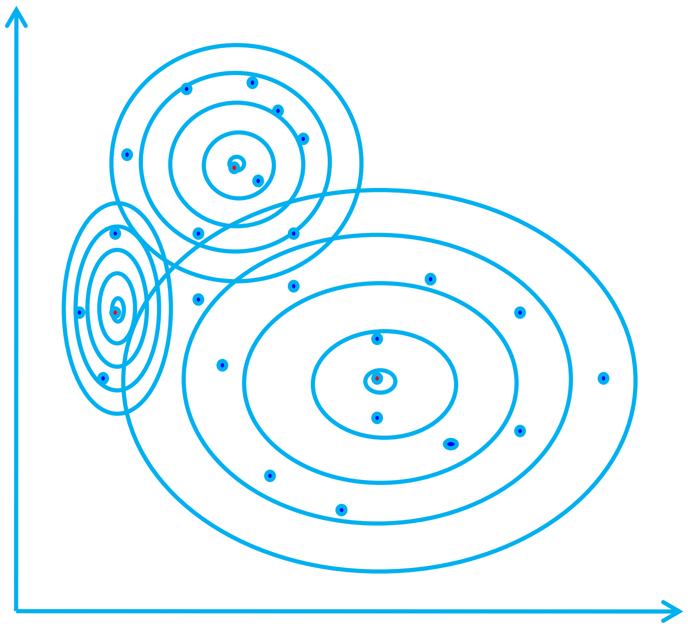
Markov Chain Monte Carlo(MCMC)

- Monte Carlo : Sample from a distribution to estimate the distribution
- Markov Chain Monte Carlo (MCMC)

– Applied to Clustering, Unsupervised Learning, Bayesian Inference

- Example: Estimation of Gaussian Mixture Model

$$p(\mathbf{x}|\theta) = \sum_{k=1}^{K} p(\mathbf{x}|\theta_k)p(\theta_k|\theta)$$



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

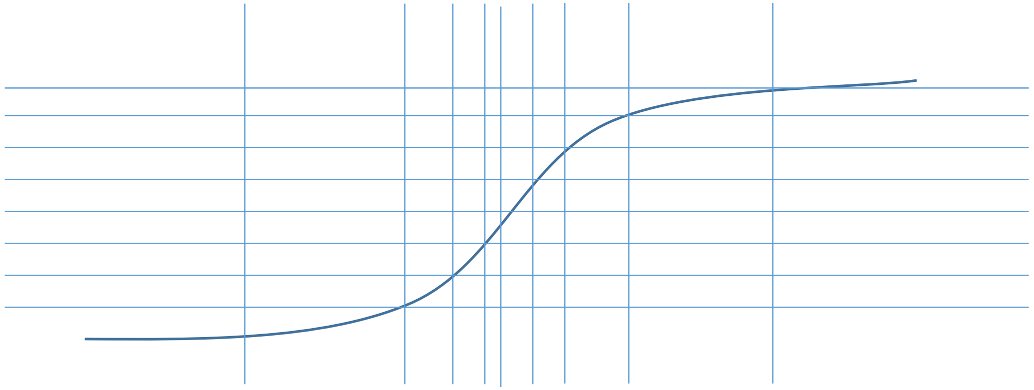
- Idea of importance sampling:

  Draw the sample from a proposal distribution $Q(\cdot)$ and re-weight by importance weights

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

- 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

- 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

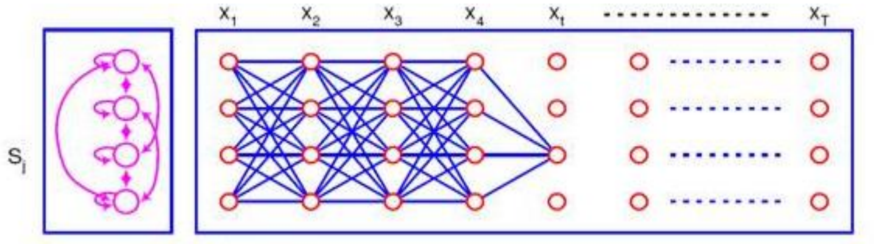
- Intuition: instead of a fixed proposal $Q(x)$, what if we could use an **adaptive** proposal?
  - X_{t+1} depends only on X_t, not on $X_0, X_1, \dots, X_{t-1}$
  - Markov Chain

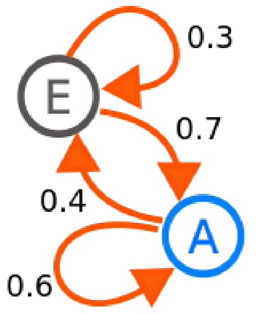
Markov Chains: Notation & Terminology

- 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.





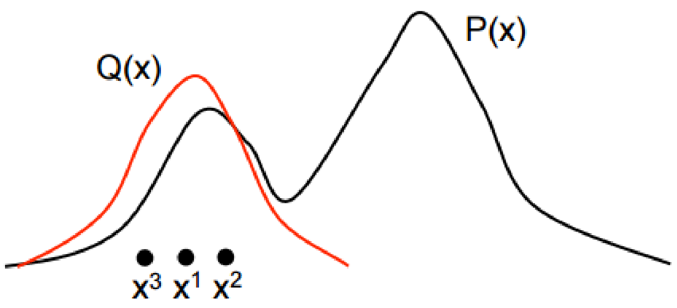
$$p_{AA} = P(X_{t+1} = A \mid X_t = A) = 0.6$$
$$p_{AE} = P(X_{t+1} = E \mid X_t = A) = 0.4$$
$$p_{EA} = P(X_{t+1} = A \mid X_t = E) = 0.7$$
$$p_{EE} = P(X_{t+1} = E \mid X_t = E) = 0.3$$

Markov 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')
  - The acceptance probability is $\text{set to } A(x'|x) = \min\left(1, \frac{P(x')/Q(x'|x)}{P(x)/Q(x|x')}\right)$ importance
  - 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)

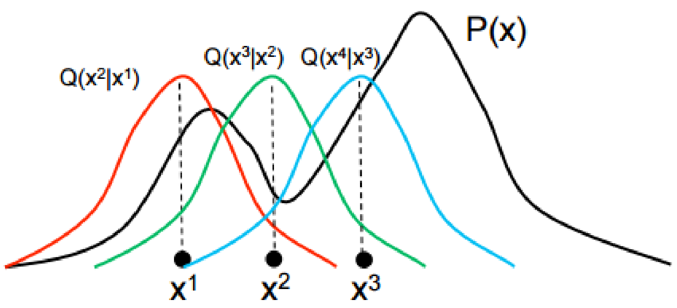


$\pi(1)$ $\pi(2)$

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

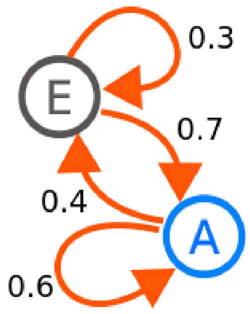
$\leftarrow p_{21}$

MCMC with adaptive proposal Q(x'|x)



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

$\leftarrow p_{21}$



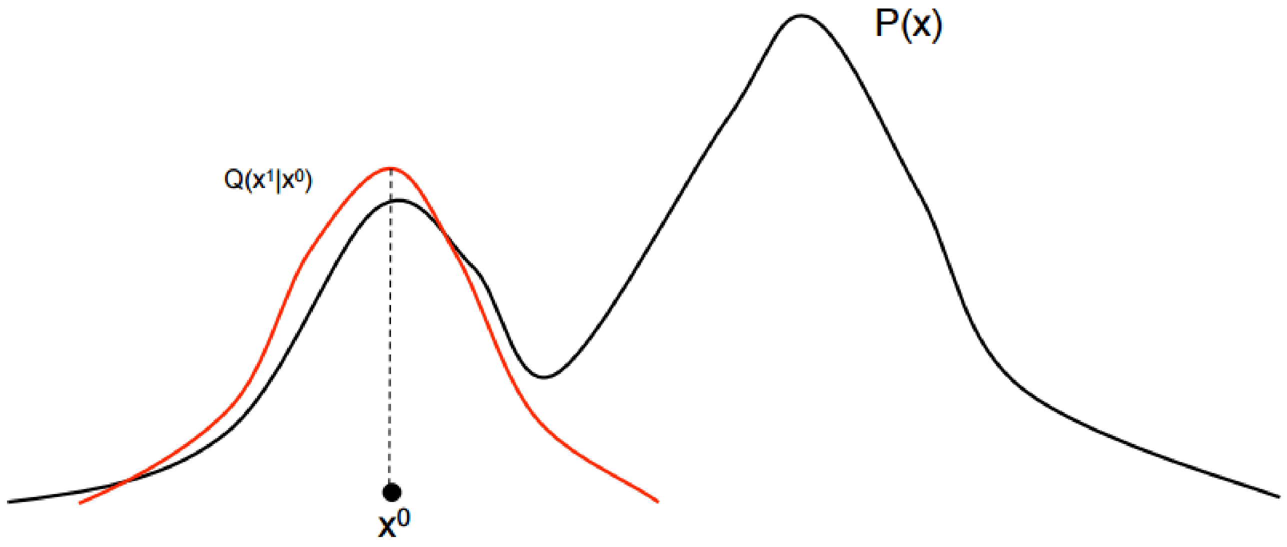
The MH Algorithm

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

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

Initialize $x^{(0)}$

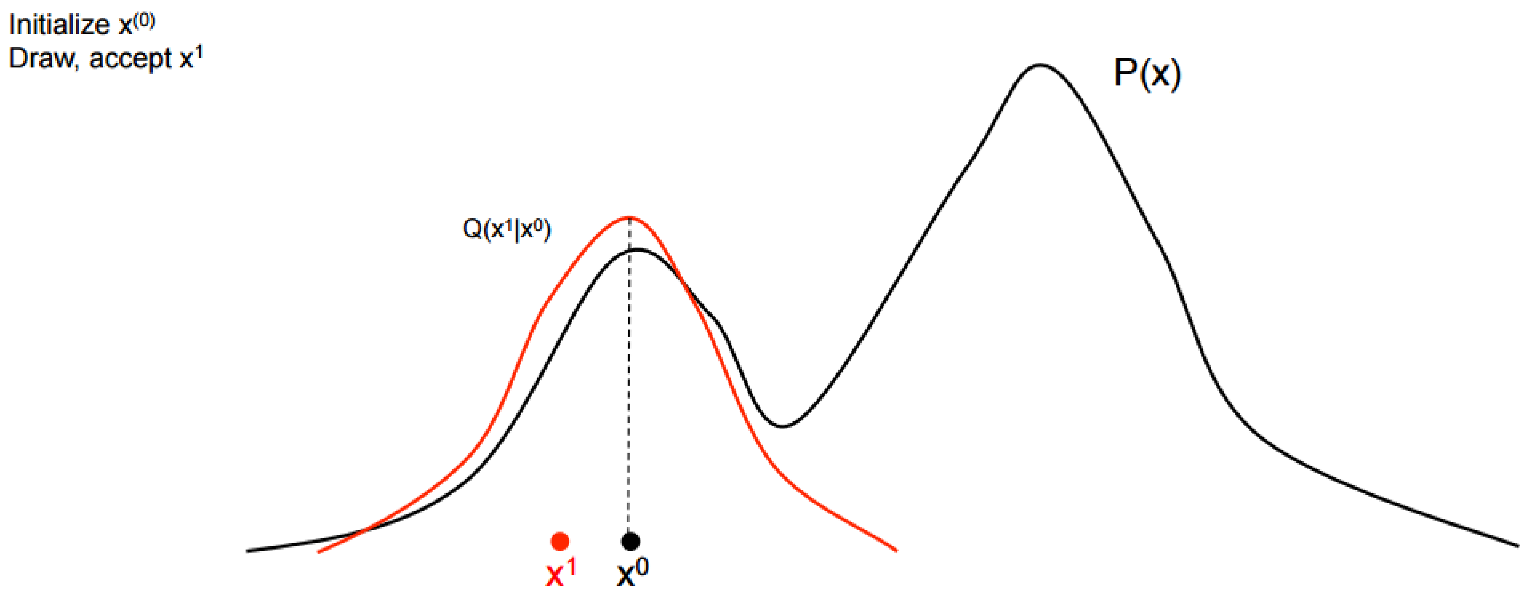
...



The MH Algorithm

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

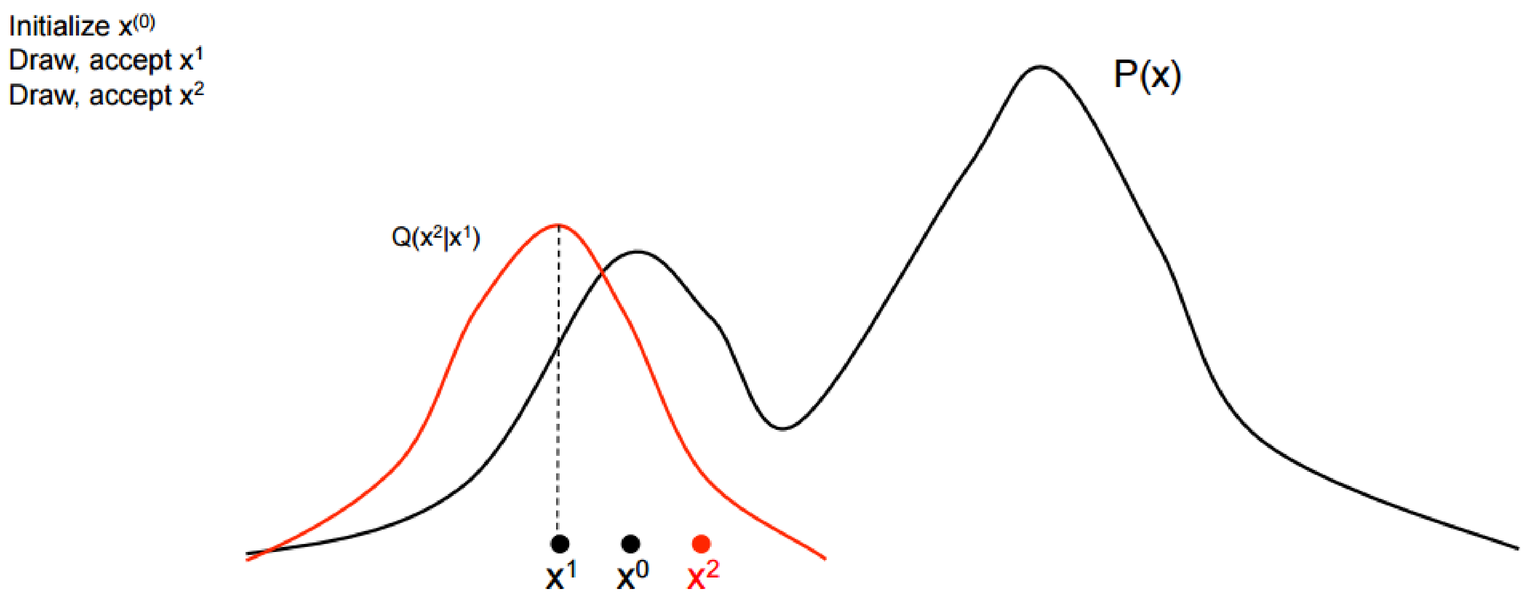
$$A(x'|x) = \min\left(1, \frac{P(x')/Q(x'|x)}{P(x)/Q(x|x')}\right)$$



The MH Algorithm

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

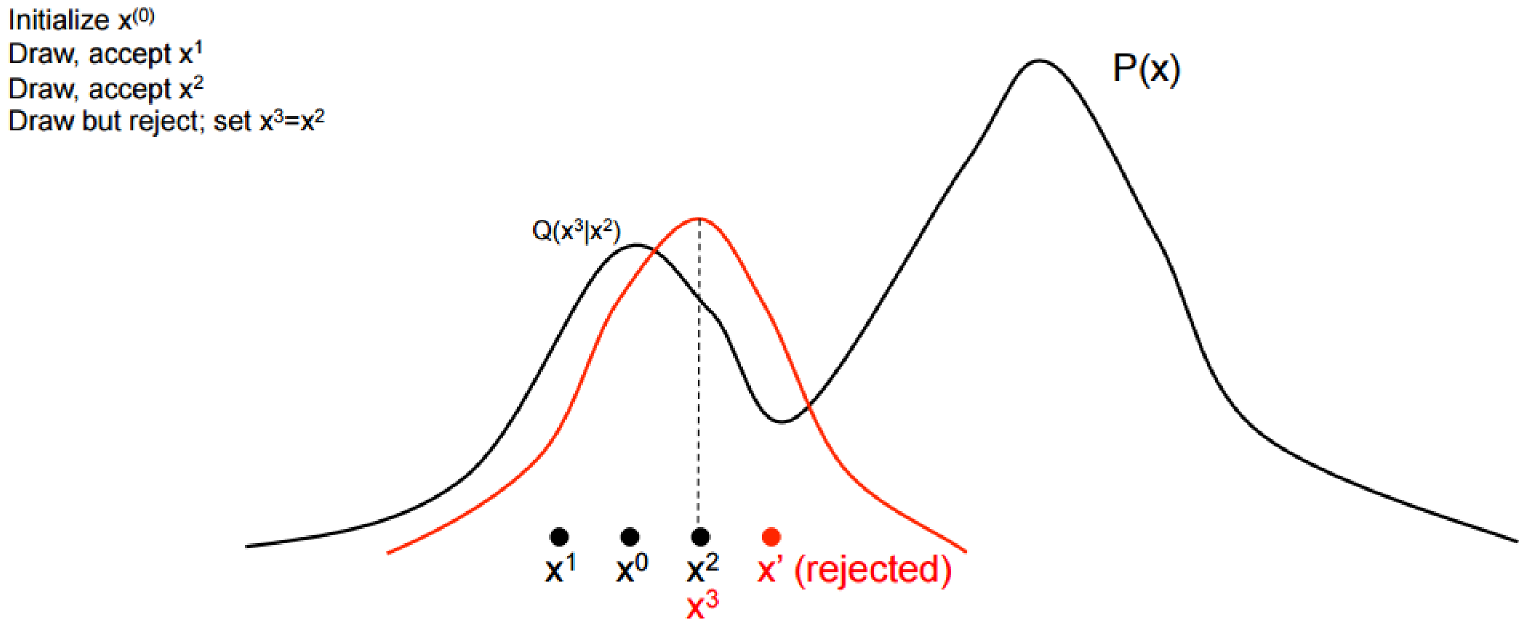
$$\boxed{A(x'|x) = \min\left(1, \frac{P(x')/Q(x'|x)}{P(x)/Q(x|x')}\right)}$$



The MH Algorithm

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

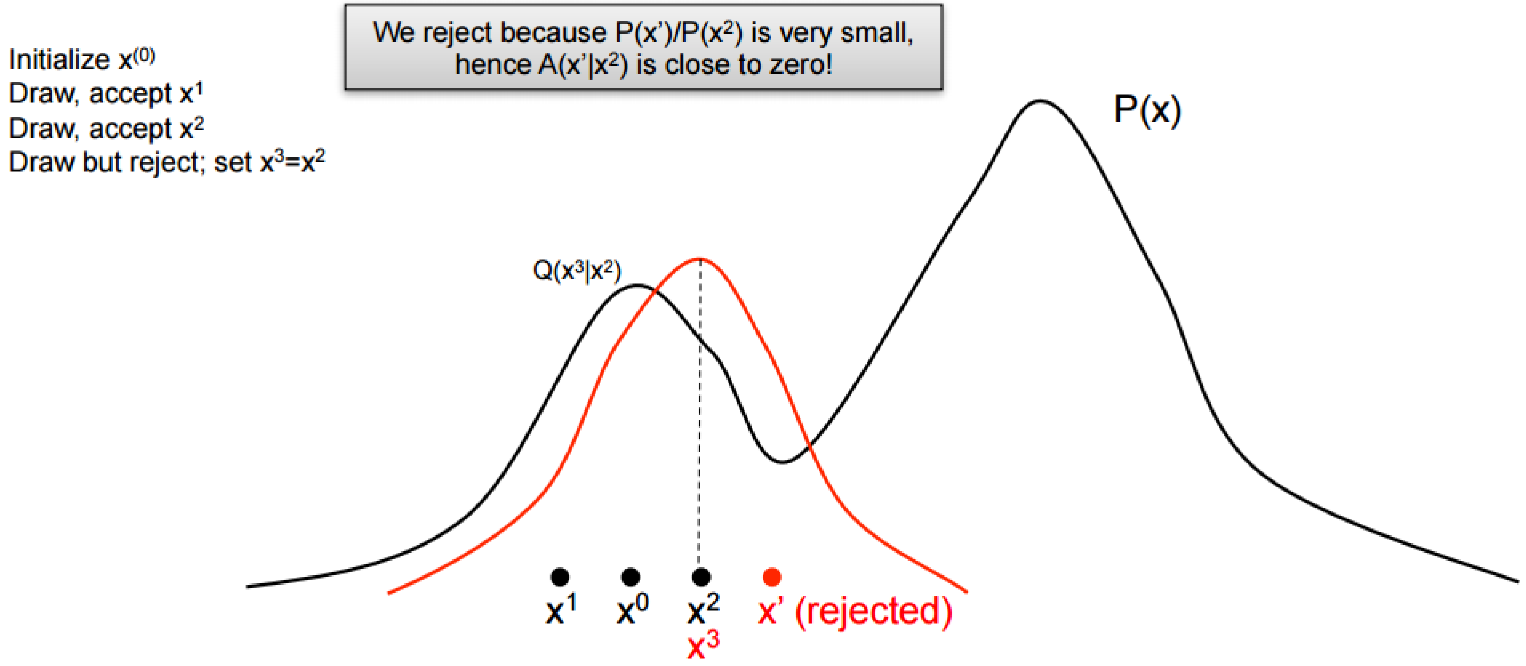
$$A(x'|x) = \min\left(1, \frac{P(x')/Q(x'|x)}{P(x)/Q(x|x')}\right)$$



The MH Algorithm

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

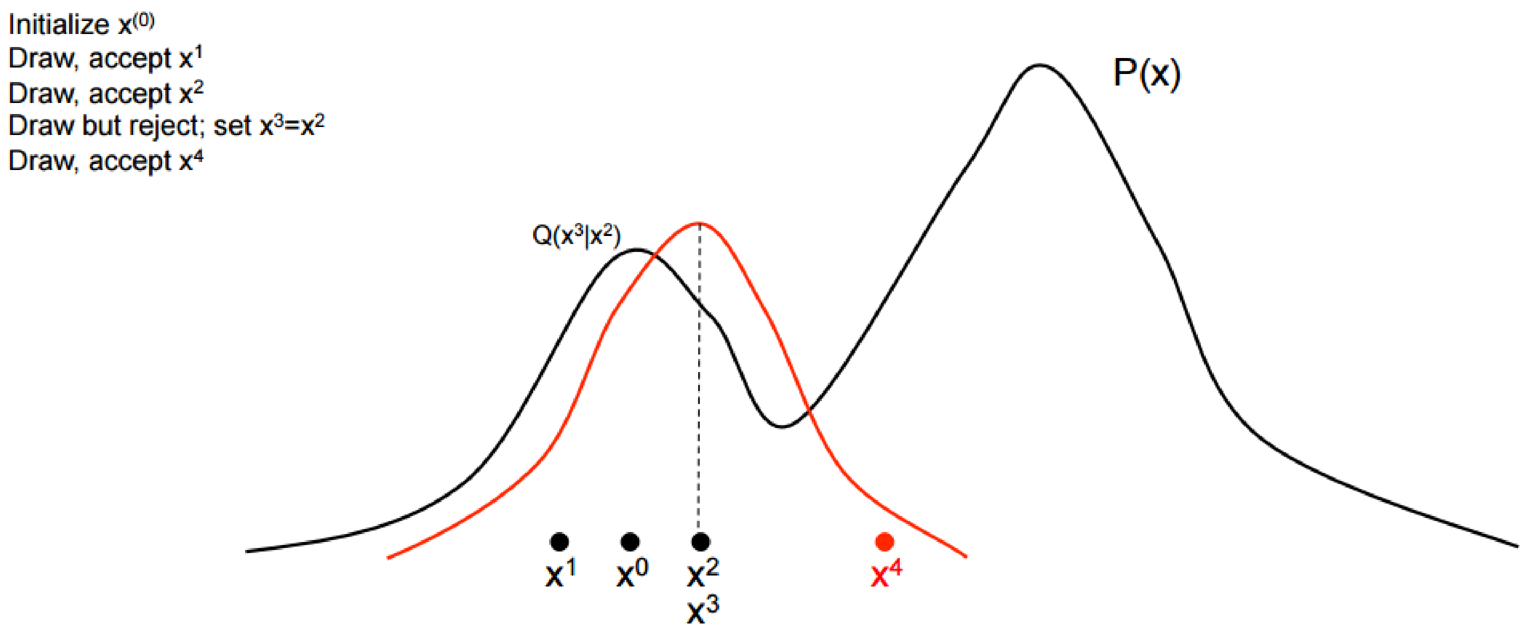
$$A(x'|x) = \min\left(1, \frac{P(x')/Q(x'|x)}{P(x)/Q(x|x')}\right)$$



The MH Algorithm

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

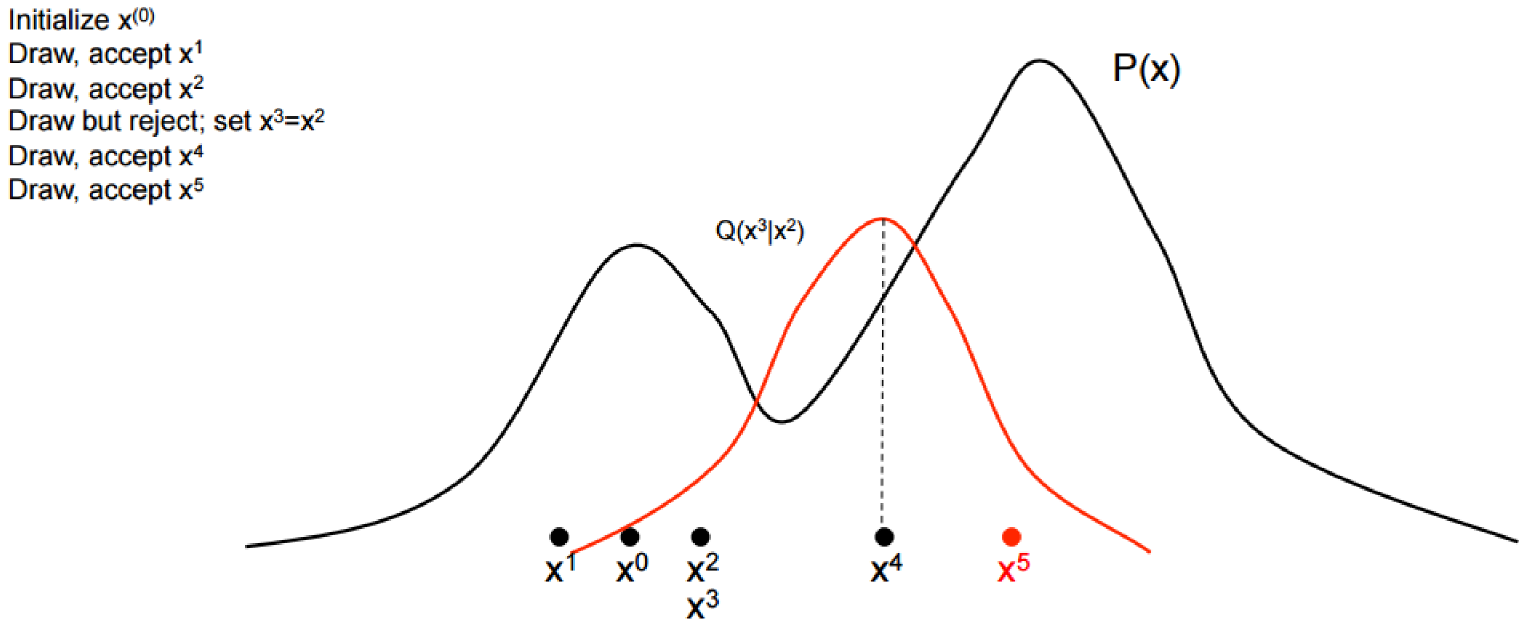
$$A(x'|x) = \min\left(1, \frac{P(x')/Q(x'|x)}{P(x)/Q(x|x')}\right)$$



The MH Algorithm

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

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

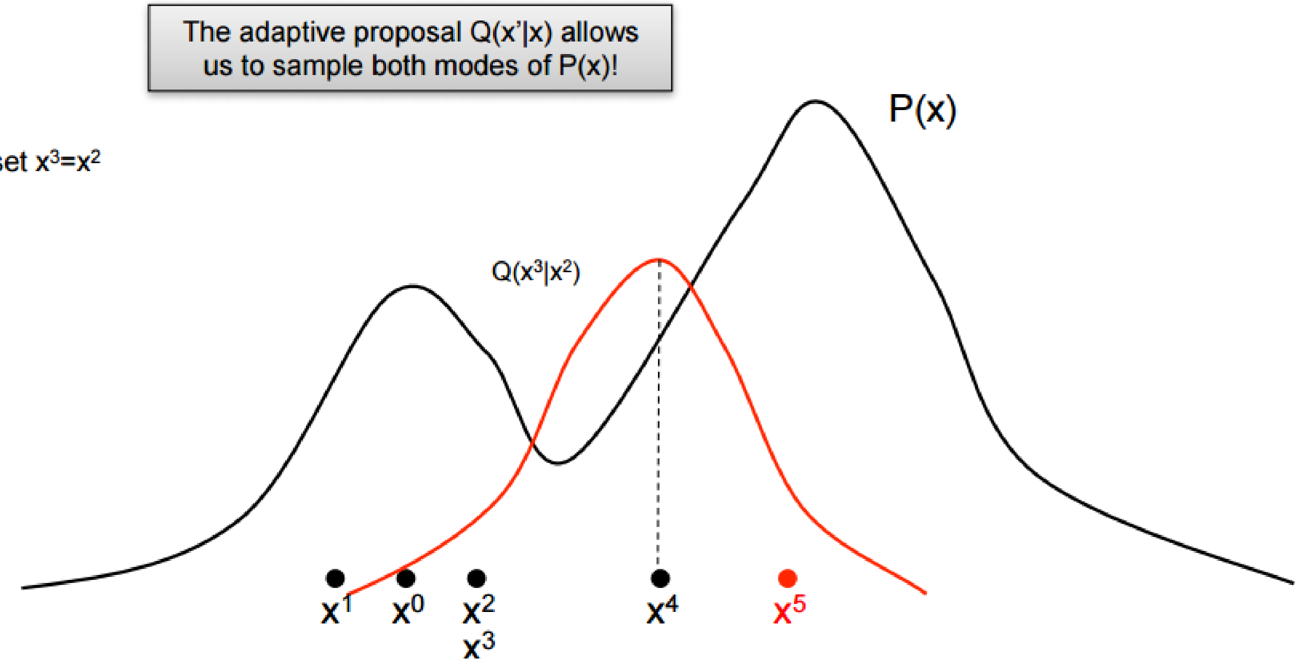


The MH Algorithm

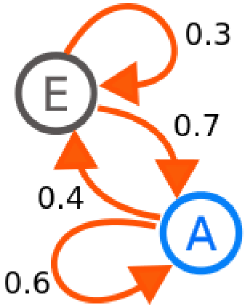
- Example:
  - Let $Q(x'|x)$ be a Gaussian centered on x
  - We're trying to sample from a bimodal distribution $P(x)$

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

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



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



The MH Algorithm

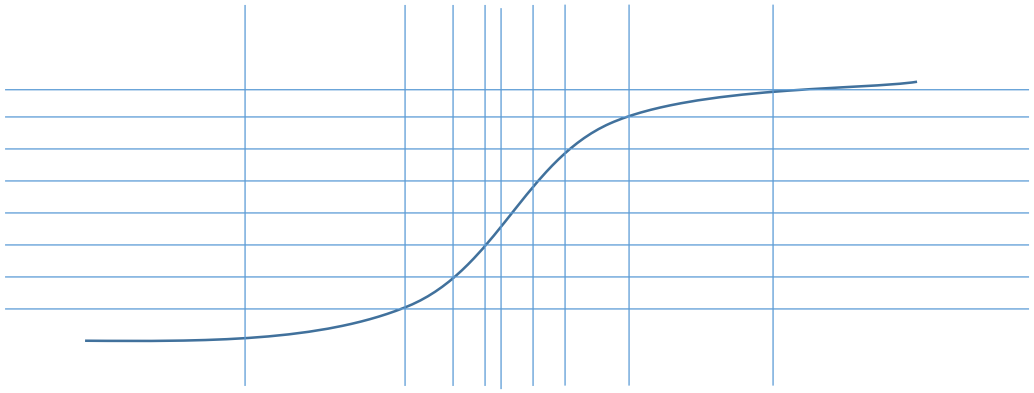
- Initialize starting state $x^{(0)}$,
- Burn-in: while samples have "not converged"
  - $x = x^{(t)}$
  - $t = t + 1$
  - Sample $x^* \sim Q(x^*|x)$ // draw from proposal
  - Sample $u \sim \text{Uniform}(0,1)$ // draw acceptance threshold
    - If $u < A(x^*|x) = \min\left(1, \frac{P(x^*)Q(x|x^*)}{P(x)Q(x^*|x)}\right)$,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

Gibbs Sampling

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

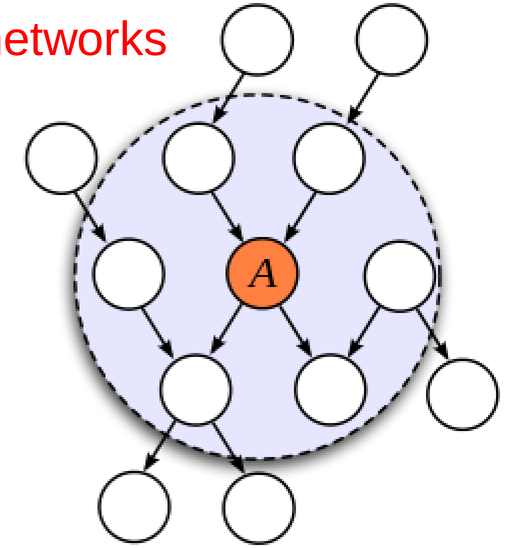
- Gibbs Sampling is an MCMC algorithm that is a special case of the MH algorithm

- 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

Bayesian networks



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

- When we update x_i, we immediately use its new value for sampling other variables x_j

  $P(x_i | x_{-i})$ achieves the acceptance probability in MH algorithm.

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

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

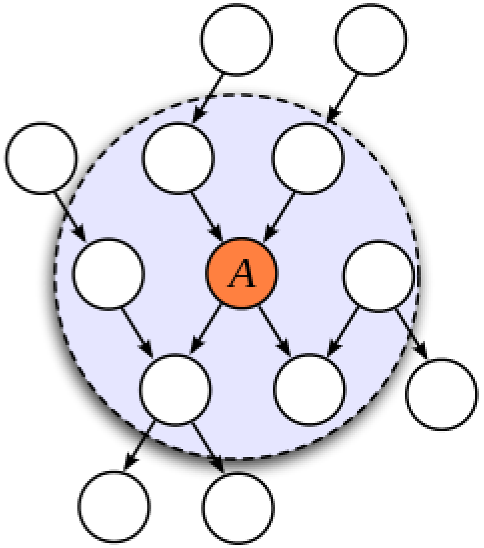
$\because$ x_i', x_i are independent

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

$$P(x_i \mid x_{-i}) = P\big(x_i|\mathrm{MB}(x_i)\big)$$

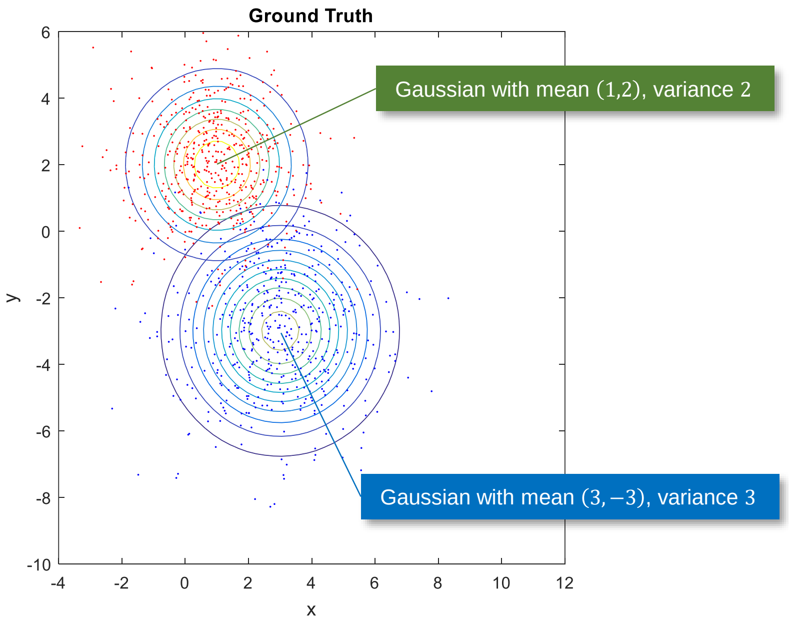
- For a Bayesian Network, the Markov Blanket of x_i is the set containing its parents, children, and co-parents

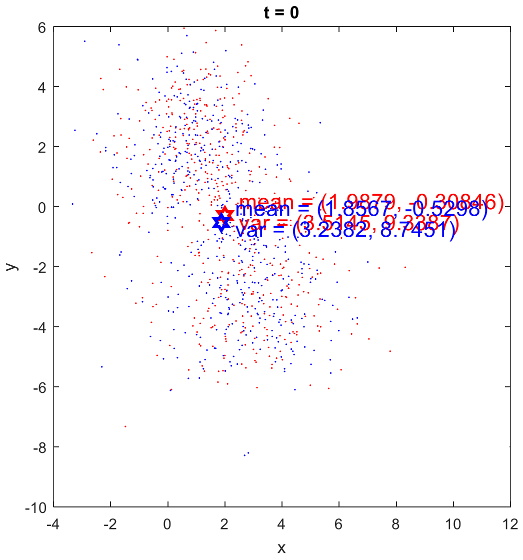


Gibbs Sampling: An Example

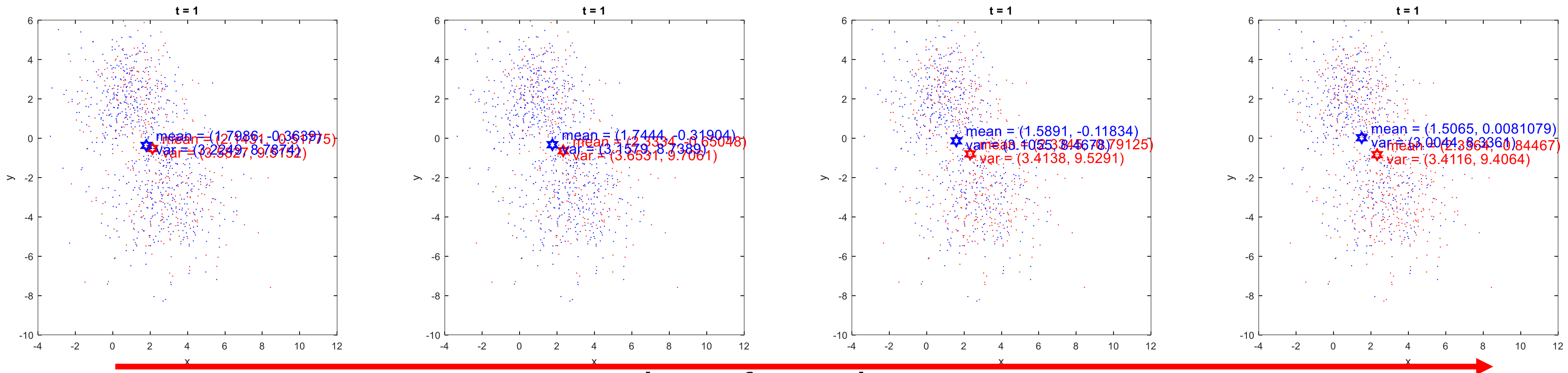
- Consider the GMM
  - The data x (position) are extracted from two Gaussian distribution
  - We do NOT know the class y of each data, and information of the Gaussian distribution
  - Initialize the class of each data at $t = 0$ to randomly

$$p(\mathbf{x}|\theta) = \sum_{k=1}^{K} p(\mathbf{x}|\theta_k)p(\theta_k|\theta) = \sum_{k=1}^{K} p(\mathbf{x}|k)p(k|\theta) = \sum_{k=1}^{K} p(\mathbf{x}, k|\theta)$$





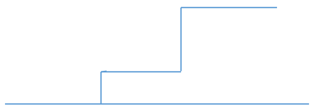
Gibbs Sampling: An Example



Iteration of i at the same t

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

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



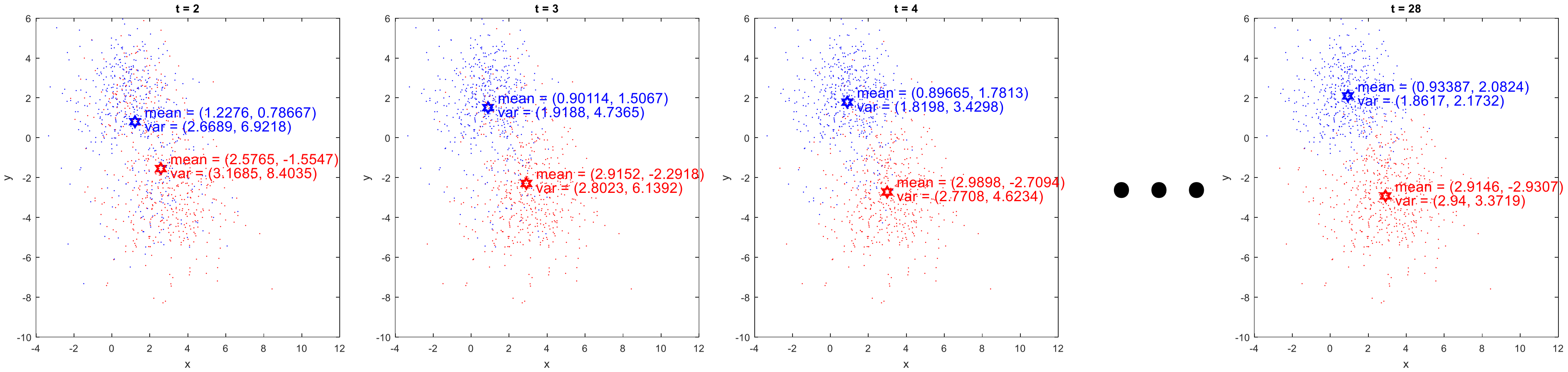
0 1

where

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

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

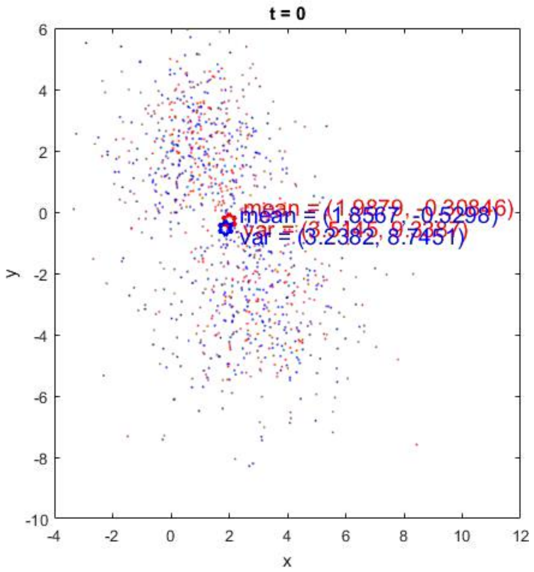
Gibbs Sampling: An Example

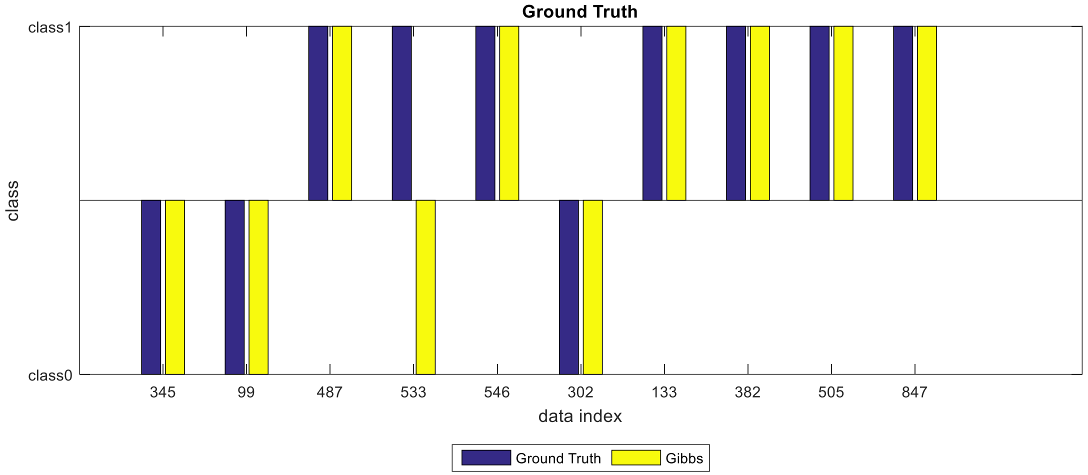


Now $t = 2$, and we repeat the procedure to sample new class of each data

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

Gibbs Sampling: An Example



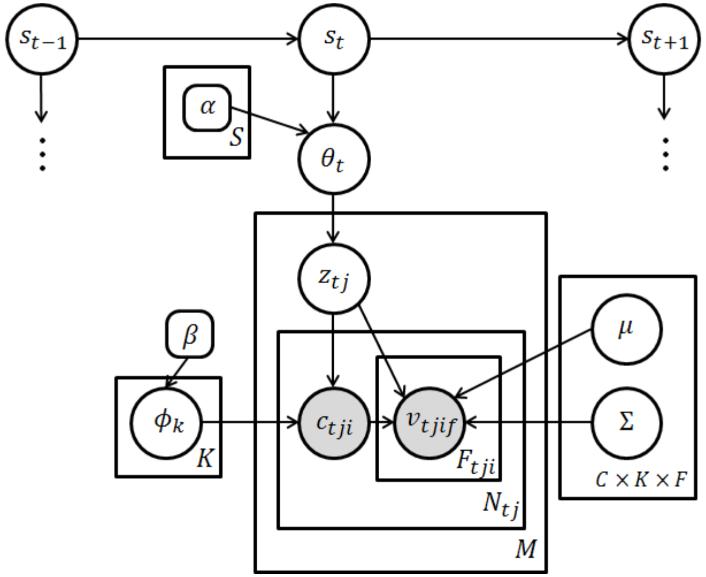


- Data i's class can be chosen with tendency of y_i
  - The classes of the data can be oscillated after the sufficient sequences
  - We can assume the class of datum as more frequently selected class

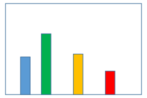
- In the simulation, the final class is correct with the probability of 94.9% at $t = 100$

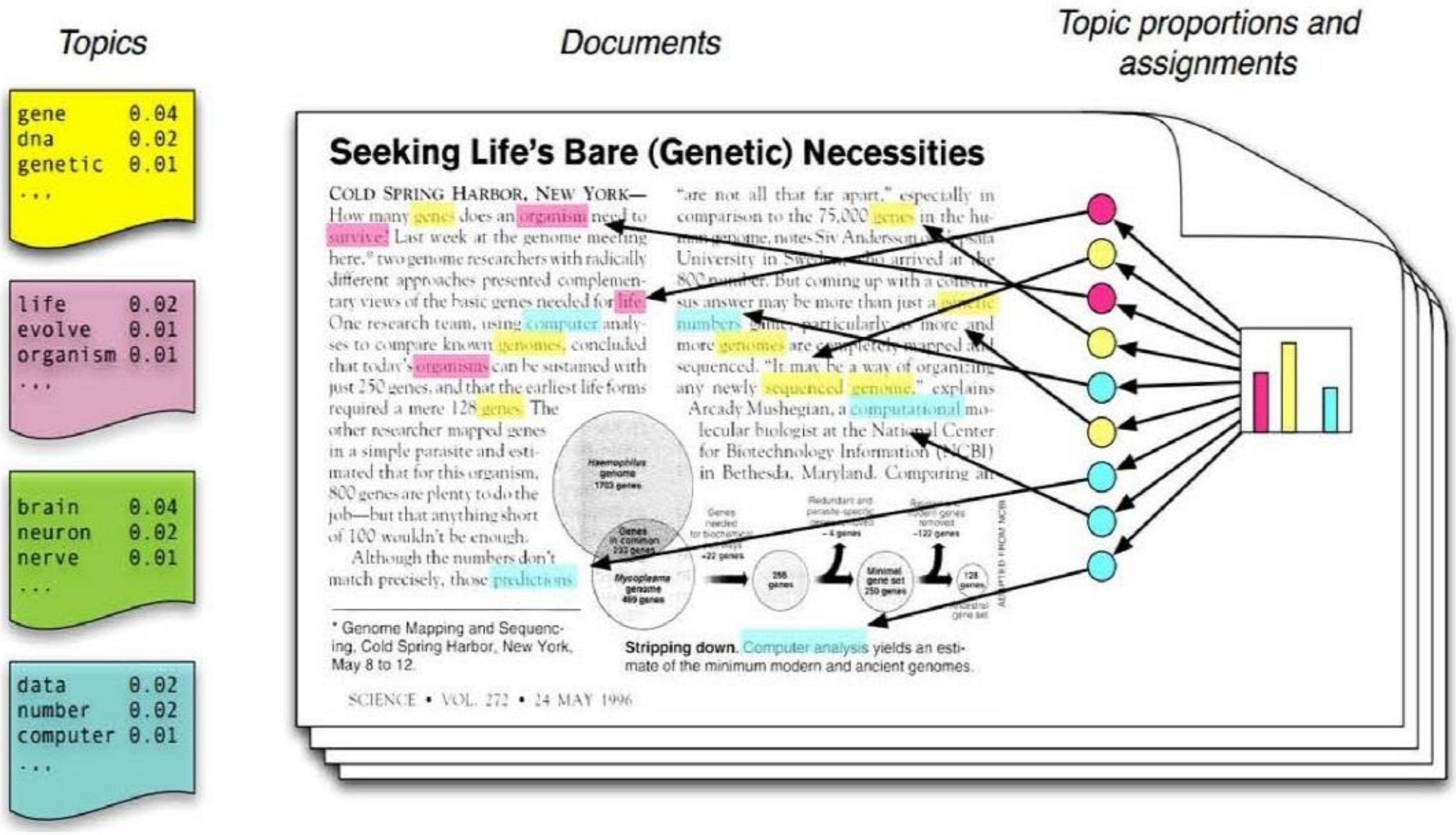
Bayesian networks: Traffic Pattern Analysis

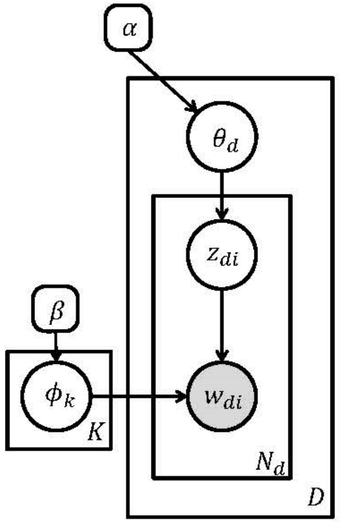
- Surveillance in crowded scenes



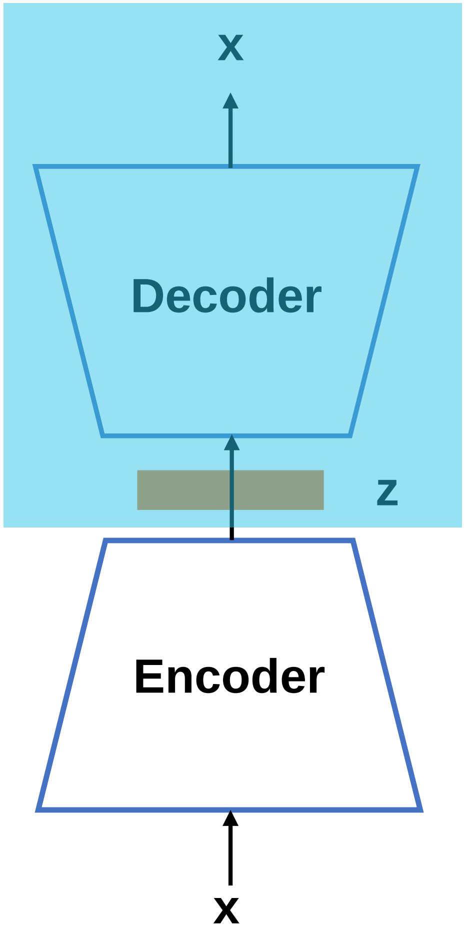
Bayesian networks (Topic Modelling)







Variational Auto-encoder (VAE)



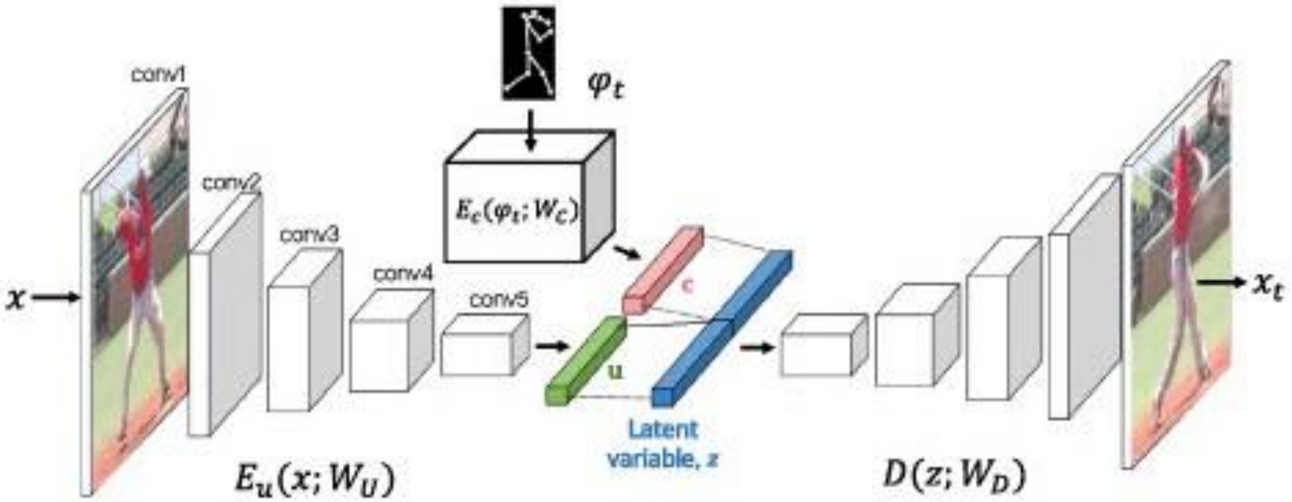
Reconstruction Loss

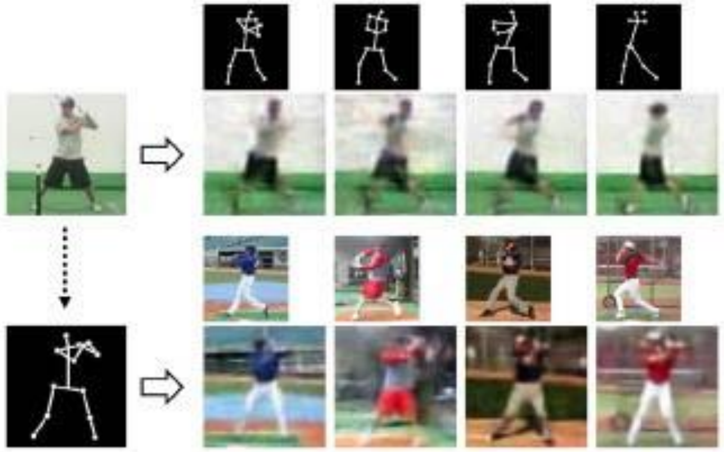
$$Loss = -logP_\theta(x|z) + D_{KL}(q_\varphi(z|x)||P_\theta(z))$$

Variational Inference

$p_\theta(x|z)$: a multivariate Gaussian (real-valued data)
a Bernoulli (binary-valued data)

Pose Transformer





$$\mathcal{L}(\theta, \phi) = \mathcal{L}_{ref} + \mathcal{L}_{pose} + \mathcal{L}_{id}.$$

$$\mathcal{L}_{ref} = -\mathbb{E}_{q_\phi(z|x_a^k, \varphi_a^k)}[\log p_\theta(x_a^k|z)] + D_{KL}\left(q_\phi(z|x_a^k, \varphi_a^k) \parallel p_\theta(z)\right).$$

$$\mathcal{L}_{pose} = -\mathbb{E}_{q_\phi(z|x_a^k, \varphi_a^l)}[\log p_\theta(x_a^l|z)] + D_{KL}\left(q_\phi(z|x_a^k, \varphi_a^l) \parallel p_\theta(z)\right) + \lambda_u \cdot D_{KL}\left(q_\phi(u|x_a^l) \parallel q_\phi(u|x_a^k)\right).$$

$$\mathcal{L}_{id} = -\mathbb{E}_{q_\phi(z|x_b^{k'}, \varphi_a^k)}[\log p_\theta(x_b^{k'}|z)] + D_{KL}\left(q_\phi(z|x_b^{k'}, \varphi_a^k) \parallel p_\theta(z)\right) + \lambda_c \cdot D_{KL}\left(q_\phi(c|\varphi_b^{k'}) \parallel q_\phi(c|\varphi_a^k)\right)$$

Loss Function for Pose Transformer

$$\mathcal{L}(\theta, \phi) = \mathcal{L}_{ref} + \mathcal{L}_{pose} + \mathcal{L}_{id}.$$

$$\mathcal{L}_{ref} = -\mathbb{E}_{q_\phi(z|x_a^k, \varphi_a^k)}[\log p_\theta(x_a^k|z)] + D_{KL}\left(q_\phi(z|x_a^k, \varphi_a^k) \parallel p_\theta(z)\right).$$

$$\mathcal{L}_{pose} = -\mathbb{E}_{q_\phi(z|x_a^k, \varphi_a^l)}[\log p_\theta(x_a^l|z)] + D_{KL}\left(q_\phi(z|x_a^k, \varphi_a^l) \parallel p_\theta(z)\right) + \lambda_u \cdot D_{KL}\left(q_\phi(u|x_a^l) \parallel q_\phi(u|x_a^k)\right).$$

$$\mathcal{L}_{id} = -\mathbb{E}_{q_\phi(z|x_b^{k'}, \varphi_a^k)}[\log p_\theta(x_b^{k'}|z)] + D_{KL}\left(q_\phi(z|x_b^{k'}, \varphi_a^k) \parallel p_\theta(z)\right) + \lambda_c \cdot D_{KL}\left(q_\phi(c|\varphi_b^{k'}) \parallel q_\phi(c|\varphi_a^k)\right)$$

Summary

- Parametric Density Estimation
  - Maximum Likelihood Estimation
  - Bayesian Learning
- Nonparametric Density Estimation
  - Histogram
  - K_n Nearest Neighbor Estimation
  - Parzen Window Estimation
  - Gaussian Mixture Estimation
    - Expectation-Maximization
    - Markov-Chain Monte Carlo