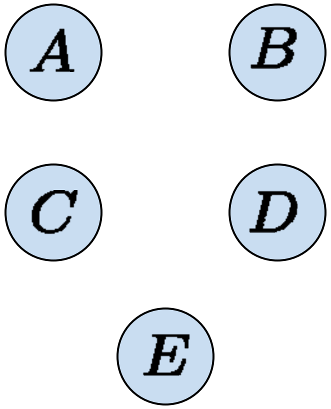
Learning Graphical Models

Adrian Barbu

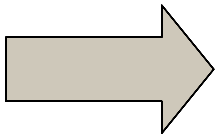
Learning Graphical Models

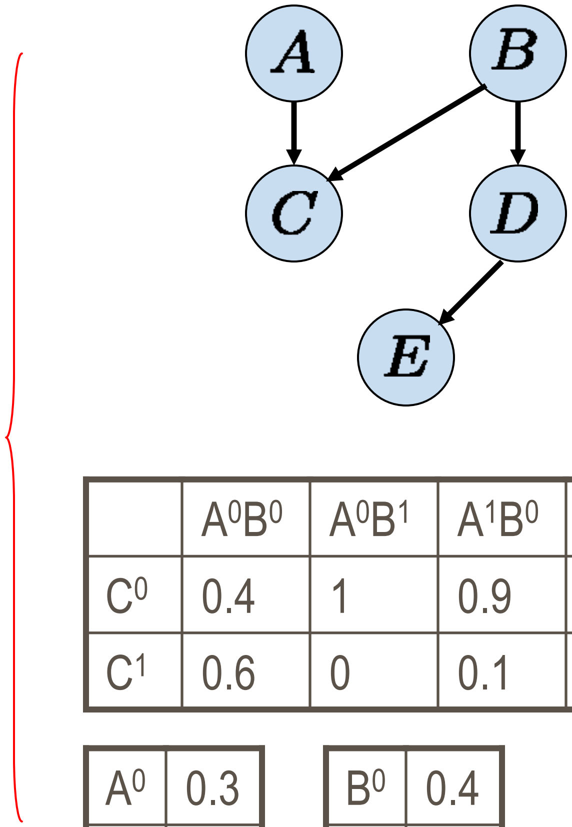
Task:

- Given training data
- Find "best" DAG and CPD



$$(A, B, C, D, E) = (00110)$$
$$(A, B, C, D, E) = (00010)$$
$$(A, B, C, D, E) = (00111)$$
$$\dots$$
$$(A, B, C, D, E) = (01110)$$





	A^0B^0	A^0B^1	A^1B^0	A^1B^1
C^0	0.4	1	0.9	0.8
C^1	0.6	0	0.1	0.2

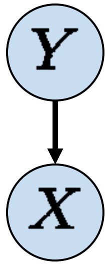
A^0	0.3
A^1	0.7

B^0	0.4
B^1	0.6

Maximum Likelihood Estimation

Case: completely observed data

Example:



- Know D={(x$_1$,y$_1$),…,(x$_n$,y$_n$)}
- Each y$_i$ is an indicator vector

$$y_i = [y_i^1, ..., y_i^M], \sum_k y_i^k = 1, y_i^k \in \{0,1\}$$

- Prior on class labels is

$$p(y) = \prod_{k=1}^{M} \pi_k^{y^k}$$

- Assume x is Gaussian with class specific mean

$$P(x|y^k = 1) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-(x-\mu_k)^2/2\sigma^2)$$

different miu, same sigma

- Then

$$P(x|y) = \prod_{k=1}^{M} N(x, \mu_k, \sigma)^{y^k}$$

MLE Example

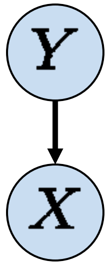
- Obtain the total log-likelihood

$$l(\mu,\sigma,\pi|D) = \log \prod_{i=1}^{n} P(x_i,y_i) = \log \prod_{i=1}^{n} P(x_i|y_i)P(y_i)$$

$$= \sum_{i=1}^{n} \log P(y_i) + \sum_{i=1}^{n} \log P(x_i|y_i)$$

$$= \sum_{i=1}^{n} \log \prod_{k=1}^{M} \pi_k^{y_i^k} + \sum_{i=1}^{n} \log \prod_{k=1}^{M} N(x_i,\mu_k,\sigma)^{y_i^k}$$

$$= \sum_{i=1}^{n}\sum_{k=1}^{M} y_i^k \log \pi_k - \sum_{i=1}^{n}\sum_{k=1}^{M} y_i^k \frac{1}{2\sigma^2}(x_i-\mu_k)^2 + C$$



- MLE means max likelihood, so partial derivatives are zero

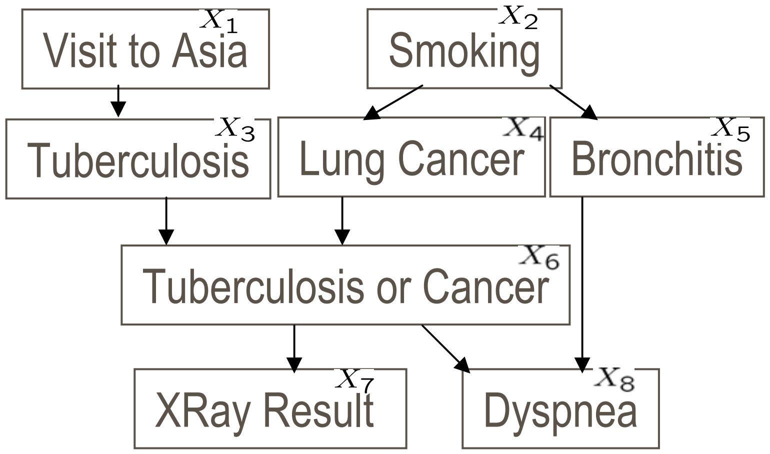
$$\frac{\partial}{\partial \pi_k} l(\mu,\sigma,\pi|D) = 0, \sum_k \pi_k = 1 \Rightarrow \pi_k = \frac{\sum_{i=1}^{n} y_i^k}{n} = \frac{n_k}{n}$$ Percent of samples in class k

so pi_{M}=1-\sum_j pi_j

$$\frac{\partial}{\partial \mu_k} l(\mu,\sigma,\pi|D) = 0 \Rightarrow \mu_k = \frac{\sum_{i=1}^{n} y_i^k x_i}{\sum_{i=1}^{n} y_i^k} = \frac{\sum_{i=1}^{n} y_i^k x_i}{n_k}$$ Average of samples in class k

MLE for Completely Observed BN

- Assume parameters of local CPDs are independent
- Assume all nodes are fully observed (known) for each training sample



- Factorized probability

$$P(\mathbf{x}) = \prod_{k=1}^{d} P(x_k|\mathbf{x}_{\pi_k})$$

- Log-likelihood:

$$l(\theta|D) = \log \prod_{i=1}^{n} P(\mathbf{x}_i) = \sum_{i=1}^{n} \log \prod_{k=1}^{d} P(x_{k,i}|\mathbf{x}_{\pi_{k},i})$$

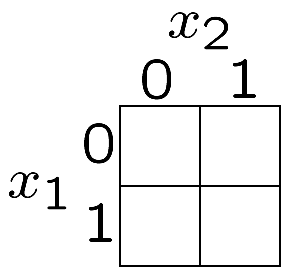
$$= \sum_{k=1}^{d} [\sum_{i=1}^{n} \log P(x_{k,i}|\mathbf{x}_{\pi_{k},i})]$$

MLE for Completely Observed BN

- Assume each CPD is a table with entries

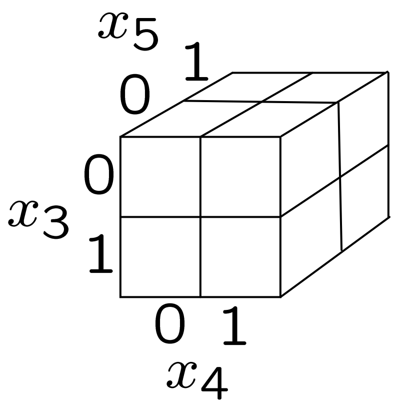
$$\theta_{kjm} = P(X_k = j|\mathbf{x}_{\pi_k} = m)$$

for all the examples



- Define the counts for each table entry

$$n_{kjm} = \sum_{i=1}^{n} \delta(X_{k,i} = j|\mathbf{x}_{\pi_{k,i}} = m)$$



- Obtain the log-likelihood

$$l(\theta|D) = \sum_{k=1}^{d} \sum_{j,m} n_{kjm} \log \theta_{kjm}$$

- Using a Lagrange multiplier to enforce $\sum_j \theta_{kjm} = 1$, we get

$$\theta_{kjm} = \frac{n_{kjm}}{\sum_l n_{klm}}$$

Partially Observed GM

- Speech Recognition
  - Each phoneme is ambiguous
    - Hear X_i and want to find Y_i
  - Disambiguation follows from the nearby phonemes (context)

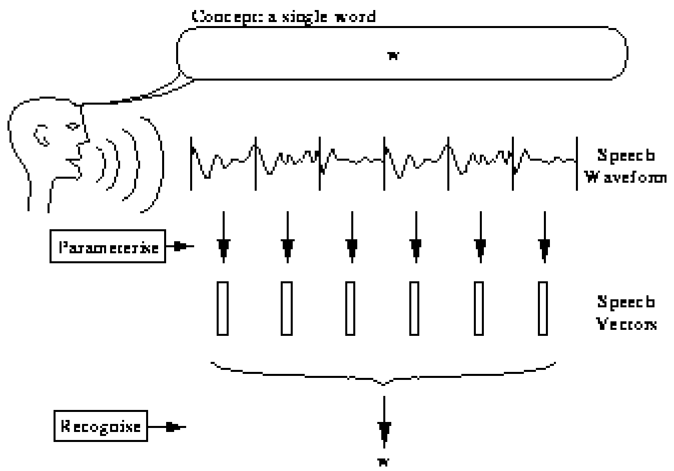
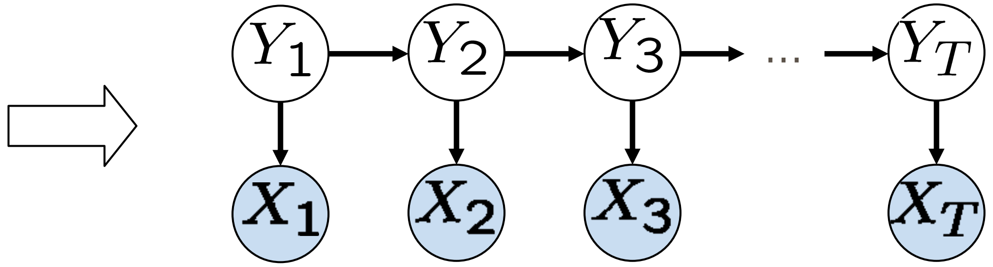
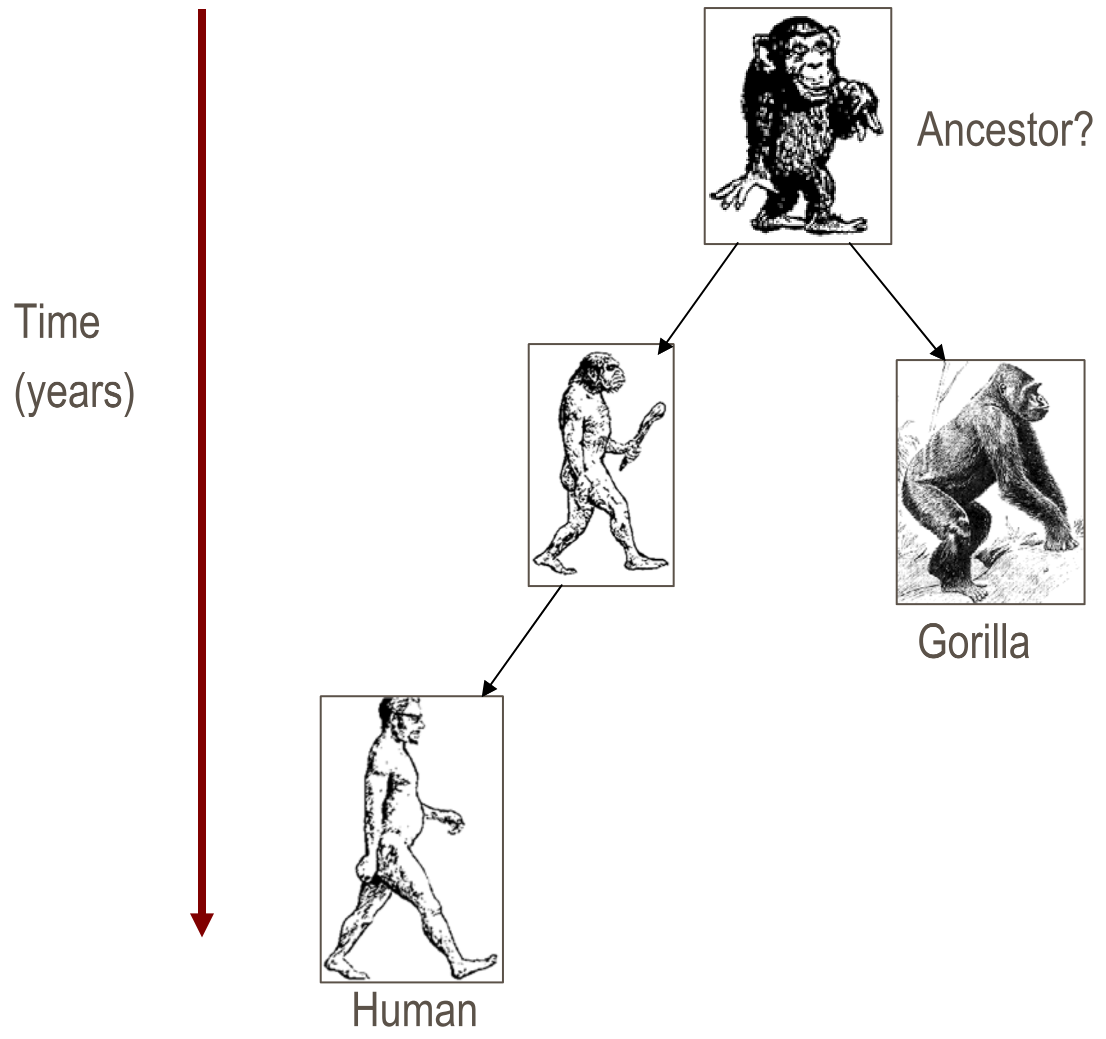


Fig. 1.2 Isolated Word Problem



Partially Observed GM

- Biological Evolution



Partially Observed GM

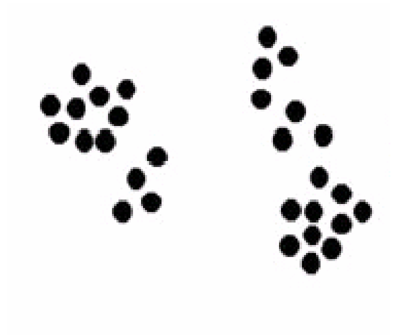
Unobserved (Latent) Variables

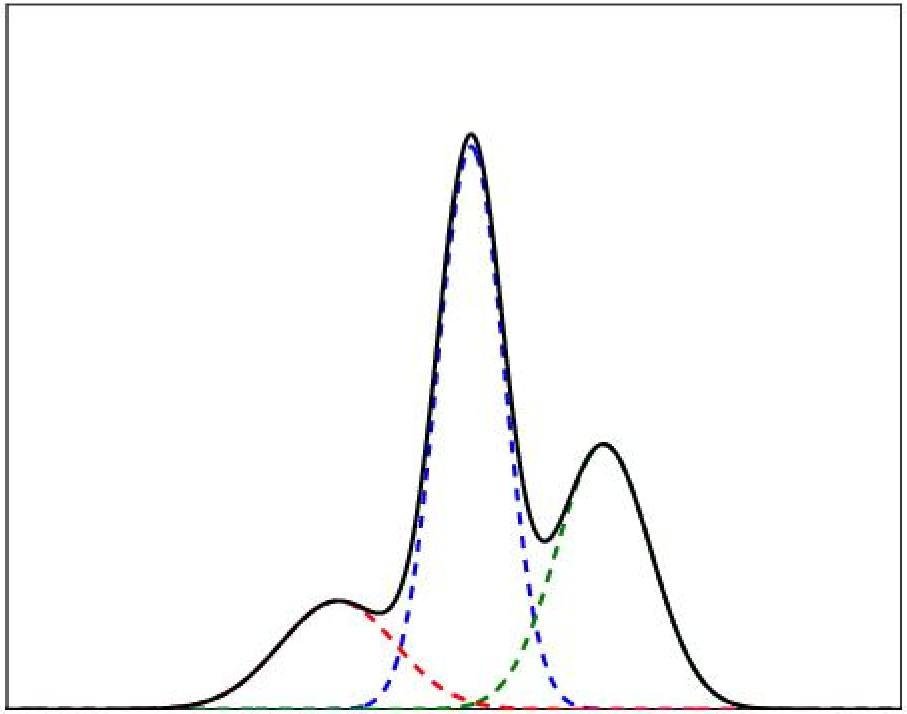
- Exist for many reasons:
  - Imaginary quantities used to model a phenomenon
    - speech recognition models, mixture models …
  - Hard to measure real-world variables
    - the temperature of a star, causes of a disease, evolutionary ancestors …
  - Was not measured all the time, because of faulty sensors, noisy channel, etc.
    - traffic radio, aircraft signal on a radar screen
- Types of latent variables:
  - Discrete latent variables can be used to partition/cluster data into sub-groups (mixture models).
  - Continuous latent variables (factors) can be used for dimensionality reduction (factor analysis, etc.).

Mixture Models

- If the label Y is not known, X will be observed as coming from a mixture model
  - E.g. mixture of Gaussians
    - Label=which Gaussian is the sample coming from

$$\begin{tikzcd} Y \arrow[d] \\ X \end{tikzcd}$$

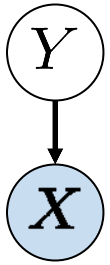




Gaussian Mixture Models

- Mixture of K Gaussian Models
  - Y=class indicator vector (unknown)

$$p(y) = \prod_{k=1}^{K} \pi_k^{y^k}$$



  - X is a conditional Gaussian with class specific mean and covariance

$$P(x|y^k = 1) = \frac{1}{(2\pi)^{m/2}|\Sigma_k|^{1/2}} \exp(-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k))$$

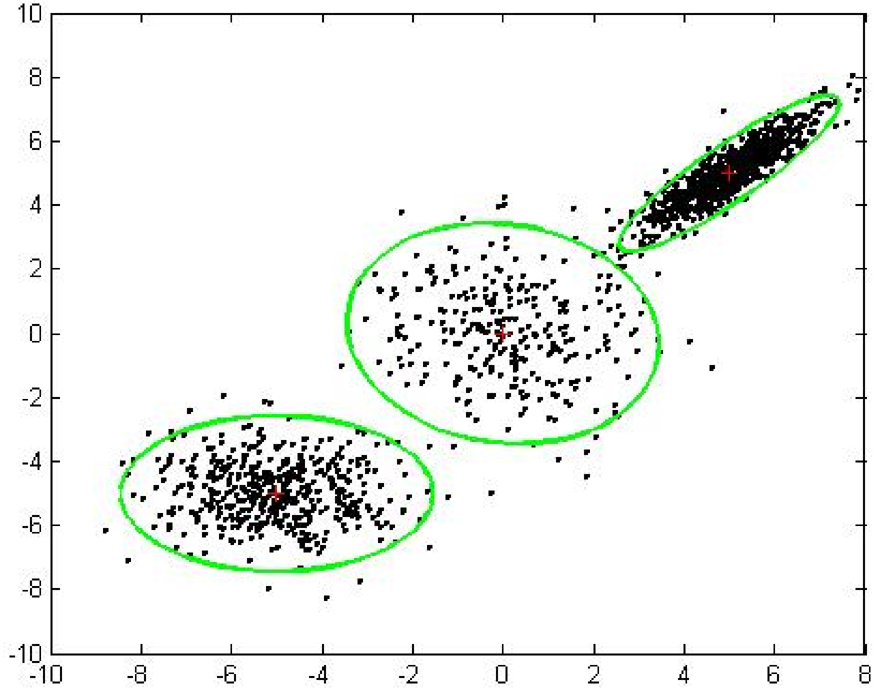
  - The likelihood of a sample integrates y

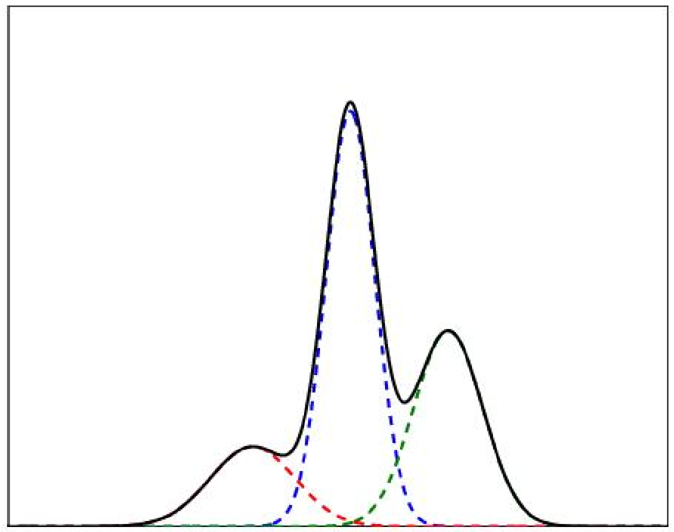
$$p(x) = \sum_{k=1}^{K} p(x|y^k = 1) = \sum_{k=1}^{K} \pi_k N(x, \mu_k, \Sigma_k)$$

Gaussian Mixture Models

- Gaussian Mixture Model (GMM)

$$p(x) = \sum_{k=1}^{K} p(x|y^k = 1) = \sum_{k=1}^{K} \underbrace{\pi_k}_{\text{Mixture proportion}} \underbrace{N(x, \mu_k, \Sigma_k)}_{\text{Mixture component}}$$





- Can be used for unsupervised clustering.
  - Has been used to discover new kinds of stars in astronomical data

Why is Learning Harder for GMM?

- For fully observed models, we have

$$l(\theta|D) = \sum_{i=1}^{n} \log P(y_i) + \sum_{i=1}^{n} \log P(x_i|y_i)$$

and we can independently learn π and the model for each label

- For Gaussian Mixture Models

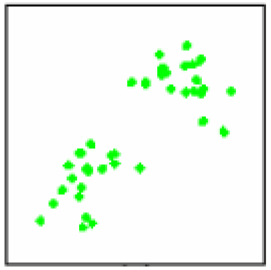
$$l(\theta|x) = \sum_{i=1}^{n} \log[\sum_{k=1}^{K} \pi_k N(x_i, \mu_k, \Sigma_k)]$$

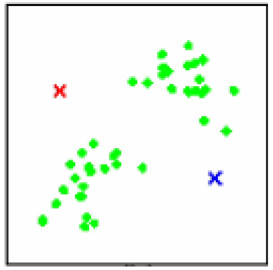
and all the parameters are coupled together

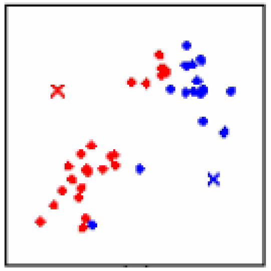
The EM algorithm

- A greedy iterative optimization algorithm for likelihoods with unobserved variables.
- Simpler than gradient descent:
  - No need to choose step size.
  - Enforces constraints automatically.
  - Calls inference and fully observed learning as subroutines.
- EM alternates two steps:
  - E-step: find hidden variables using inference, $p(y|x,\theta)$.
  - M-step: update parameters using standard MLE/MAP method applied to fully observed data
- At each step, the likelihood improves or remains unchanged.
  - Thus it always converges to a local maximum likelihood.

K-means Clustering







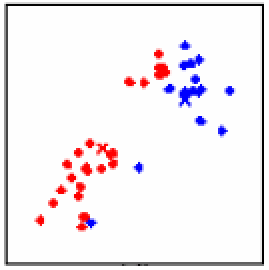
- Initialize
  - K- the number of clusters
  - Cluster center positions μ_k and covariances Σ_k
- Repeat until convergence
  - For each point i=1 to n
    - Find the cluster it most probably belongs to

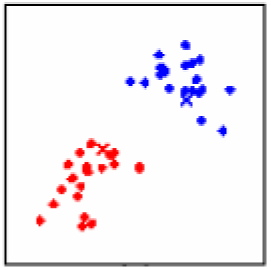
Should be min

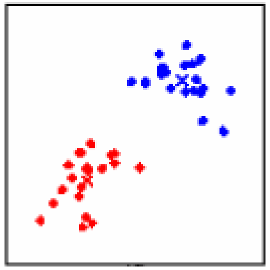
$$y_i = \arg\max_k (x_i - \mu_k)^T \Sigma_k^{-1} (x_i - \mu_k)$$

  - Update the cluster parameters (μ_k, Σ_k)

$$\mu_k = \frac{\sum_{i, y_i = k} x_i}{n_k}, \Sigma_k = \frac{\sum_{i, y_i = k} (x_i - \mu_k)(x_i - \mu_k)^T}{n_k - 1}$$







The EM Algorithm

- Initialize
  - K- the number of clusters
  - Cluster parameters θ (e.g. $\theta=(\mu_k,\Sigma_k)$ for Gaussian models)
- Repeat until convergence
  - Expectation step:
    - For each point i=1 to n, compute the expected value of y_i, given θ

$$y_i^k = p(y = k | x_i, \theta)$$

  - Maximization step:
    - Update the parameters θ to maximize the log-likelihood given the y_i

$$\theta = \arg\max_\theta l(\theta | x, y) = \arg\max_\theta \sum_{i,k} y_i^k \log p(x_i, y = k | \theta)$$

  - y_i are not hard assignments anymore, they are probabilities

EM for Gaussian Models

- Mixture of K Gaussians
  - y is the indicator vector y=(y^1,…,y^K)

$$p(y) = \prod_{k=1}^{K} \pi_k^{y^k}$$

  - X is a conditional Gaussian with class-specific mean and covariance

$$P(x|y^k = 1) = \frac{1}{(2\pi)^{m/2}|\Sigma_k|^{1/2}} \exp(-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k))$$

  - Sample likelihood:

$$p(x) = \sum_{k=1}^{K} p(x|y^k = 1) = \sum_{k=1}^{K} \pi_k N(x, \mu_k, \Sigma_k)$$

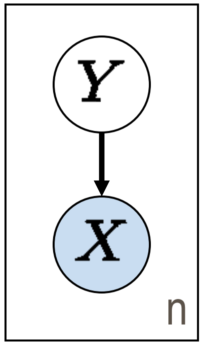
  - Total log-likelihood:

$$l(\mu, \Sigma, \pi|x) = \sum_{i=1}^{n} \log[\sum_{k=1}^{K} \pi_k N(x_i, \mu_k, \Sigma_k)]$$

EM for Gaussian Models

- Expectation step:
  - Considering all π,μ,Σ fixed, compute $p(y_i^k=1|x,\mu,\Sigma,\pi)$ by Bayes rule

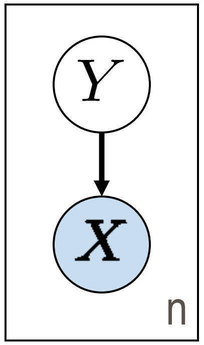
$$p(y_i^k = 1|x_i,\mu,\Sigma,\pi) = \frac{p(x_i|y_i^k = 1,\mu,\Sigma,\pi)p(y_i^k = 1|\mu,\Sigma,\pi)}{\sum_l p(x_i|y_i^l = 1,\mu,\Sigma,\pi)p(y_i^l = 1|\mu,\Sigma,\pi)}$$
$$= \frac{N(x_i,\mu_k,\Sigma_k)\pi_k}{\sum_l \pi_l N(x_i,\mu_l,\Sigma_l)}$$



  - This is an inference step in the BN

EM for Gaussian Models

- Maximization step:
  - Considering all y_i fixed, re-estimate π,μ,Σ
  - Similar to the fully observed case



$$\pi_k = \frac{\sum_{i=1}^n y_i^k}{n},$$
$$\mu_k = \frac{\sum_i y_i^k x_i}{\sum_i y_i^k},$$
$$\Sigma_k = \frac{\sum_i y_i^k (x_i - \mu_k)(x_i - \mu_k)^T}{\sum_i y_i^k}$$

K-means vs. EM

- The EM algorithm for GMM is a "soft version" of the K-means algorithm.
- Both have E and M steps, but
- E-step
  - In K-means → hard assignment to one class
  $$y_i = \arg\max_k (x_i - \mu_k)^T \Sigma_k^{-1} (x_i - \mu_k)$$
  - In EM → soft assignment (fractional to each class)
  $$p(y_i^k = 1 | x_i, \mu, \Sigma, \pi) = \frac{N(x_i, \mu_k, \Sigma_k)\pi_k}{\sum_l \pi_l N(x_i, \mu_l, \Sigma_l)}$$
- M-step is essentially the same

Partially Hidden Data

- Some variables are missing (hidden) on some examples and not on others.
- In this case the log-likelihood is:

$$l(\theta|D) = \sum_{i \in \text{Complete}} \log p(x_i, y_i|\theta) + \sum_{j \in \text{Missing}} \log p(x_j, y_j|\theta)$$

- y_j can have different missing values for different j
- E-step: estimate the hidden variables on the incomplete examples only.
- M-step: find the MLE parameters given the complete and incomplete data

Theory Underlying EM

- If we knew y, we would have the log-likelihood

$$l(\theta, x, y) = \sum_i \log p(x_i, y_i|\theta)$$

  - Easy to maximize as we saw for MLE of observed data
  - Decomposes in a sum of independent terms

- Since we don't know y, we need to integrate it out

$$l(\theta, x) = \sum_i \log \sum_y p(x_i, y|\theta) = \sum_i \log \sum_y p(y|\theta)p(x_i|y, \theta)$$

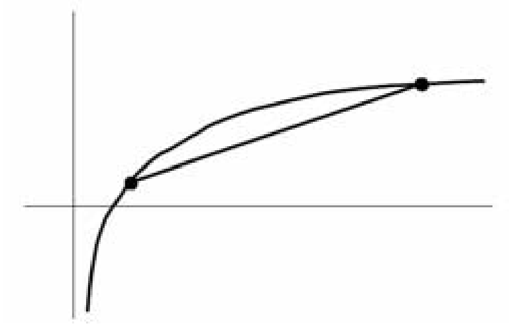
  - No decomposition, hard to maximize

Expected Complete Log-Likelihood

- For any distribution q, we define:

$$\langle l(\theta, x, y)\rangle_q = \sum_i \sum_y q(y|x_i, \theta) \log p(x_i, y|\theta)$$

  - Decomposes into a sum of terms
  - Easy to maximize
- Jensen's inequality
  - Log is concave down



$$\begin{aligned}\log \sum_y p(x_i, y|\theta) &= \log \sum_y q(y|x_i) \frac{p(x_i, y|\theta)}{q(y|x_i)} \\ &\geq \sum_y q(y|x_i) \log \frac{p(x_i, y|\theta)}{q(y|x_i)} \\ &= \sum_y q(y|x_i) \log p(x_i, y|\theta) + H(q|x_i)\end{aligned}$$

  - So we get $l(\theta, x) \geq \langle l(\theta, x, y)\rangle_q + H(q|x)$

Lower Bounds and Free Energy

- For fixed data x, define a functional called the free energy:

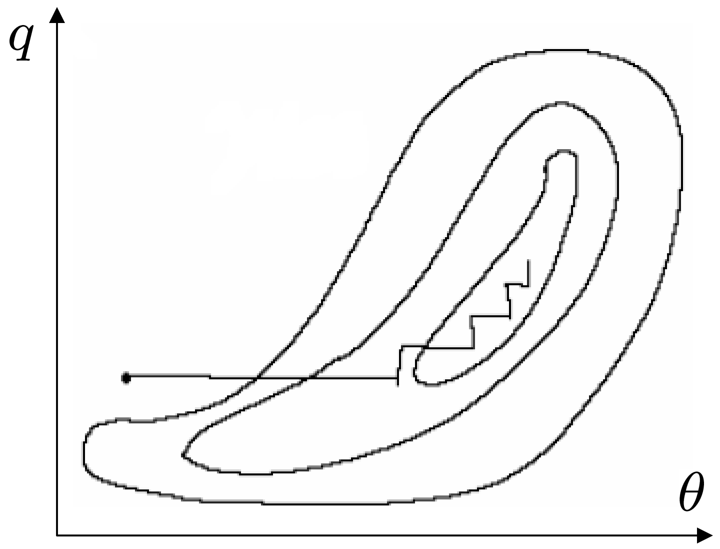
$$F(q,\theta) = \sum_{i,y} q(y|x_i) \log \frac{p(x_i, y|\theta)}{q(y|x_i)} \le l(\theta, x)$$

- The EM algorithm is coordinate-ascent on F
  - E-step:

$$q^{t+1} = \arg\max_q F(q^t, \theta^t)$$

  - M-step:

$$\theta^{t+1} = \arg\max_\theta F(q^t, \theta^t)$$



The Expectation Step

- We need to show that

$$q^{t+1} = \arg\max_q F(q^t, \theta^t) = p(y|x, \theta)$$

- We prove that it attains the bound $F(q, \theta) \le l(\theta, x)$

$$\begin{aligned} F(p(y|x,\theta),\theta) &= \sum_{i,y} p(y|x_i,\theta) \log \frac{p(x_i, y|\theta)}{p(y|x_i,\theta)} \\ &= \sum_{i,y} p(y|x_i,\theta) \log p(x_i|\theta) \\ &= \sum_i \log p(x_i|\theta) = l(\theta, x) \end{aligned}$$

- We can also show this using

$$l(\theta, x) - F(q, \theta) = KL(q, p(y|x, \theta))$$

The Maximization Step

- We have

$$\begin{aligned}
\theta^{t+1} &= \arg\max_{\theta} F(q^t, \theta^t) \\
&= \arg\max_{\theta} \sum_{i,y} q^t(y|x_i) \log \frac{p(x_i, y|\theta)}{q^t(y|x_i)} \\
&= \arg\max_{\theta} [\sum_{i,y} q^t(y|x_i) \log p(x_i, y|\theta) + \sum_{i} H(q^t|x_i)] \\
&= \arg\max_{\theta} \sum_{i,y} q^t(y|x_i) \log p(x_i, y|\theta) \\
&= \arg\max_{\theta} \sum_{i,k} y_i^k \log p(x_i, y = k|\theta)
\end{aligned}$$

which is exactly the MLE with soft assignments

EM: Pros and Cons

- Pros:
  - No learning rate (step-size) parameter
  - Automatically enforces parameter constraints
  - Very fast for low dimensions
  - Each iteration guaranteed to improve likelihood

- Cons:
  - Greedy, can get stuck in local minimum
  - Can be slower than conjugate gradient (especially near convergence)
  - Requires expensive inference step
  - No theoretical guarantees of convergence to true parameters

Provable EM

EM for mixture of isotropic Gaussians [Dasgupta, 2000]

Say we want k clusters in R^M

We will start with l=O(k ln k) clusters

Provable EM Algorithm

1. Initialize μ_i, i=1,…,l, as random data points, π_i=1/l,
$$\sigma_i^2 = \min_j \|\mu_i - \mu_j\|$$
2. One EM step
3. Pruning Step
4. One EM Step

Provable EM

Pruning step:

1. Remove all clusters with π_i<1/4l
2. Selected k centers furthest from each other
    1. Add one random μ_i to S
    2. For j=1 to k-1

       Add to S the center with maximum distance d(μ_i,S)

$$d(\mu_i, S) = \min_{j \in S} \|\mu_i - \mu_j\|$$

Provable EM

Theorem [Dasgupta 2007] If the data points come from a mixture of k Gaussians with centers μ_i^* and S_i are the data points coming from cluster i. Then for any $\varepsilon,\delta > 0$ satisfying

1. Number of clusters is $l = O(\frac{1}{\pi_{min}} \ln \frac{1}{\delta\pi_{min}})$
2. Three other separability conditions (ε,δ)

we have with probability at least $1-\delta$ and up to a permutation

$$\|\mu_i - \mu_i^*\| \leq \|mean(S_i) - \mu_i\| + \epsilon\sigma_i\sqrt{M}$$

- It means we couldn't do much better than EM even if we knew the point labels!

Provable EM Insights

Provable EM gives us advice how to design better EM the right way in general:

1. Start with more cluster centers than k
2. Do one or more EM steps
3. Prune weak cluster centers and keep the best separated k clusters
4. Do one or more EM steps

Conclusions

The EM Algorithm

- A way to maximize the likelihood function for BN with latent variables.
- Finds the MLE parameters when the problem can be broken up into two (easy) pieces:
  1. Estimate the missing data probabilities from observed data and current parameters.
  2. Using the estimates, find the MLE parameters.
- Alternates between estimating the latent variables using the best guess (posterior) and updating the parameters based on this estimation:
  - M-step: optimize a lower bound on the likelihood
  - E-step: close the gap, make bound=likelihood