

**CS 2750 Machine Learning**  
**Lecture 19**

**Expectation Maximization (EM)**  
**Mixtures of Gaussians.**

Milos Hauskrecht  
[milos@cs.pitt.edu](mailto:milos@cs.pitt.edu)  
5329 Sennott Square

---

CS 2750 Machine Learning

**Learning probability distribution**

**Basic learning settings:**

- A set of random variables  $\mathbf{X} = \{X_1, X_2, \dots, X_n\}$
- **A model of the distribution** over variables in  $\mathbf{X}$   
with parameters  $\Theta$
- **Data**  $D = \{D_1, D_2, \dots, D_N\}$  **s.t.**  $D_i = (x_1^i, x_2^i, \dots, x_n^i)$

**Objective:** find parameters  $\hat{\Theta}$  that describe  $p(\mathbf{X})$

**Assumptions considered so far:**

- Known parameterizations
- No hidden variables
- No-missing values

**This lecture:** consider hidden variables and missing values

---

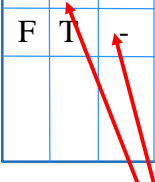
CS 2750 Machine Learning

## Learning with hidden variables and missing values


**Problem:** we want to learn  $P(\mathbf{X})$  from data  $\mathbf{D}$

- Data consists of examples with values for variables
- But sometimes the values are missing

Data	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	Extended Data	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$h$
	T	T	F	T	T		T	T	F	T	T	-
	T	-	T	F	F		T	-	T	F	F	-
	F	T	-	F	T		F	T	-	F	T	-



Missing values



Hidden variable  $h$   
(all values missing)

CS 2750 Machine Learning

## Density estimation

**Goal:** Find the set of parameters  $\hat{\Theta}$

**Estimation criteria:**

- ML  $\max_{\Theta} p(D | \Theta, \xi)$
- Bayesian  $p(\Theta | D, \xi)$

**Optimization methods for ML:** gradient-ascent, conjugate gradient, Newton-Rhapson, etc.

**Problem:** No or very small advantage from the structure of the corresponding belief network when there are unobserved values

**Expectation-maximization (EM) method**

- An alternative optimization method
- Suitable when there are missing or hidden values
- **Takes advantage of the structure of the belief network**

CS 2750 Machine Learning

## General EM

**The key idea of a method:**

**Compute the parameter estimates** iteratively by performing the following two steps:

**Two steps of the EM:**

1. **Expectation step.** For all hidden and missing variables (and their possible value assignments) calculate their expectations for the current set of parameters  $\Theta'$
2. **Maximization step.** Compute the new estimates of  $\Theta$  by considering the expectations of the different value completions

**Stop when no improvement possible**

## General EM

**Current parameters  $\Theta'$**

**(1) Expectation step.** For all hidden and missing variables (and their possible value assignments) calculate their expectations for the current set of parameters

$x_1$	$x_2$		$x_5$	$h$
T	T	F	T	T
T	-	T	F	F
F	T	-	F	T

Calculate expectations for all missing value assignments from  $\Theta'$

$$P(h^{(1)} = T \mid D^{(1)}, \Theta')$$

$$P(h^{(1)} = F \mid D^{(1)}, \Theta')$$

$$P(x_2^{(2)} = T, h^{(2)} = T \mid D^{(2)}, \Theta')$$

$$P(x_2^{(2)} = T, h^{(2)} = F \mid D^{(2)}, \Theta')$$

$$P(x_2^{(2)} = F, h^{(2)} = T \mid D^{(2)}, \Theta')$$

$$P(x_2^{(2)} = F, h^{(2)} = F \mid D^{(2)}, \Theta')$$

## General EM

**Current parameters**  $\Theta'$

**(2) Maximization step.** Compute the new estimates of parameters  $\Theta$  by considering the expectations of the different value completions

$x_1$	$x_2$		$x_5$	$h$	
T	T	F	T	T	-
T	-	T	F	F	-
F	T	-	F	T	-

Compute the new estimates of  $\Theta$

$$\begin{aligned}
 &P(h^{(1)} = T \mid D^{(1)}, \Theta') \\
 &P(h^{(1)} = F \mid D^{(1)}, \Theta') \\
 &P(x_2^{(2)} = T, h^{(2)} = T \mid D^{(2)}, \Theta') \\
 &P(x_2^{(2)} = T, h^{(2)} = F \mid D^{(2)}, \Theta') \\
 &P(x_2^{(2)} = F, h^{(2)} = T \mid D^{(2)}, \Theta') \\
 &P(x_2^{(2)} = F, h^{(2)} = F \mid D^{(2)}, \Theta')
 \end{aligned}$$



$\Theta$

## EM details

Let  $H$  – be a set of hidden or missing variable values in data

**Derivation**

$$P(H, D \mid \Theta, \xi) = P(H \mid D, \Theta, \xi) P(D \mid \Theta, \xi)$$

$$\log P(H, D \mid \Theta, \xi) = \log P(H \mid D, \Theta, \xi) + \log P(D \mid \Theta, \xi)$$

$$\log P(D \mid \Theta, \xi) = \log P(H, D \mid \Theta, \xi) - \log P(H \mid D, \Theta, \xi)$$

**Log-likelihood of data**

**Average both sides** with  $P(H \mid D, \Theta', \xi)$  **for some**  $\Theta'$

$$\begin{aligned}
 E_{H \mid D, \Theta'} \log P(D \mid \Theta, \xi) &= \underbrace{E_{H \mid D, \Theta'} \log P(H, D \mid \Theta, \xi)}_Q(\Theta \mid \Theta') - \underbrace{E_{H \mid D, \Theta'} \log P(H \mid \Theta, \xi)}_H(\Theta \mid \Theta') \\
 \log P(D \mid \Theta, \xi) &= Q(\Theta \mid \Theta') + H(\Theta \mid \Theta')
 \end{aligned}$$

**Log-likelihood of data**

## EM algorithm

**Algorithm** (general formulation)

Initialize parameters  $\Theta$

Repeat

Set  $\Theta' = \Theta$

1. **Expectation step**

$$Q(\Theta | \Theta') = E_{H|D, \Theta'} \log P(H, D | \Theta, \xi)$$

2. **Maximization step**

$$\Theta = \arg \max_{\Theta} Q(\Theta | \Theta')$$

until no or small improvement in  $\Theta$  ( $\Theta = \Theta'$ )

**Questions:** Why this leads to the ML estimate ?

What is the advantage of the algorithm?

CS 2750 Machine Learning

## EM algorithm

**Question:** Why is the EM algorithm correct?

• **Claim: maximizing Q improves the log-likelihood**

$$l(\Theta) = Q(\Theta | \Theta') + H(\Theta | \Theta')$$

**Difference in log-likelihoods (current and next step)**

$$l(\Theta) - l(\Theta') = Q(\Theta | \Theta') - Q(\Theta' | \Theta') + H(\Theta | \Theta') - H(\Theta' | \Theta')$$

**Subexpression**  $H(\Theta | \Theta') - H(\Theta' | \Theta') \geq 0$

**Kullback-Leibler (KL) divergence** (distance between 2 distributions)

$$KL(P | R) = \sum_i P_i \log \frac{P_i}{R_i} \geq 0 \quad \text{Is always positive !!!}$$

$$H(\Theta | \Theta') = -E_{H|D, \Theta'} \log P(H | \Theta, D, \xi) = -\sum_i p(H | D, \Theta') \log P(H | \Theta, D, \xi)$$

$$H(\Theta | \Theta') - H(\Theta' | \Theta') = \sum_i P(H | D, \Theta') \log \frac{P(H | \Theta', D, \xi)}{P(H | \Theta, D, \xi)} \geq 0$$

CS 2750 Machine Learning

## EM algorithm

### Difference in log-likelihoods

$$l(\Theta) - l(\Theta') = Q(\Theta | \Theta') - Q(\Theta' | \Theta') + H(\Theta | \Theta') - H(\Theta' | \Theta')$$

$$l(\Theta) - l(\Theta') \geq Q(\Theta | \Theta') - Q(\Theta' | \Theta')$$

Thus

by **maximizing Q** we improve the log-likelihood

$$l(\Theta) = Q(\Theta | \Theta') + H(\Theta | \Theta')$$

EM is a first-order optimization procedure

- **Climbs the gradient**
- **Automatic learning rate**

**No need to adjust the learning rate !!!!**

---

CS 2750 Machine Learning

## EM advantages

### Key advantages:

- In many problems (e.g. Bayesian belief networks)

$$Q(\Theta | \Theta') = E_{H|D, \Theta'} \log P(H, D | \Theta, \xi)$$

- has a nice form and the maximization of Q can be carried out in the closed form
- No need to compute Q before maximizing
- We directly optimize
  - using quantities corresponding to expected counts

---

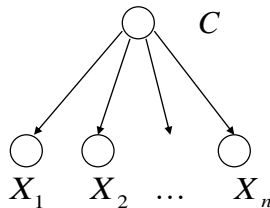
CS 2750 Machine Learning

## Naïve Bayes with a hidden class and missing values

### Assume:

- $P(\mathbf{X})$  is modeled using a Naïve Bayes model with a hidden class variable  $C$
- Missing entries (values) for attributes in the dataset  $D$

### Hidden class variable



Attributes are independent given the class

CS 2750 Machine Learning

## EM for the Naïve Bayes

- We can use EM to iteratively relearning the parameters with

$$Q(\Theta | \Theta') = E_{H|D, \Theta'} \log P(H, D | \Theta, \xi)$$

- **Parameters:**

$\pi_j$  prior on class  $j$

$\theta_{ijk}$  probability of an attribute  $i$  having value  $k$  given class  $j$

- **Indicator variables:**

$\delta_j^l$  for example  $l$ , the class is  $j$ ; if true (=1) else false (=0)

$\delta_{ijk}^l$  for example  $l$ , the class is  $j$  and the value of attrib  $i$  is  $k$

- because the class is hidden and some attributes are missing, the values (0,1) of indicator variables are not known; they are hidden

$H$  – a collection of all indicator variables (that ‘complete’ the data)

CS 2750 Machine Learning

## EM for the Naïve Bayes model

- We can use EM to iteratively learn the parameters of the model

$$Q(\Theta | \Theta') = E_{H|D, \Theta'} \log P(H, D | \Theta, \xi)$$

$$\log P(H, D | \Theta, \xi) = \log \prod_{l=1}^N \prod_j \pi_j^{\delta_j^l} \prod_i \prod_k \theta_{ijk}^{\delta_{ijk}^l}$$

**Completed loglikelihood**  $= \sum_{l=1}^N \sum_j (\delta_j^l \log \pi_j + \sum_i \sum_k \delta_{ijk}^l \log \theta_{ijk})$

$$E_{H|D, \Theta'} \log P(H, D | \Theta, \xi) = \sum_{l=1}^N \sum_j (E_{H|D, \Theta'}(\delta_j^l) \log \pi_j + \sum_i \sum_k E_{H|D, \Theta'}(\delta_{ijk}^l) \log \theta_{ijk})$$

$$E_{H|D, \Theta'}(\delta_j^l) = p(C_l = j | D_l, \Theta')$$

Substitutes 0,1

$$E_{H|D, \Theta'}(\delta_{ijk}^l) = p(X_{il} = k, C_l = j | D_l, \Theta')$$

with the expected value

## EM for the Naïve Bayes model

- Computing derivatives of  $Q$  for parameters and setting it to 0 we get:

$$\pi_j = \frac{\tilde{N}_j}{N} \quad \theta_{ijk} = \frac{\tilde{N}_{ijk}}{\sum_{k=1}^r \tilde{N}_{ijk}}$$

$$\tilde{N}_j = \sum_{l=1}^N E_{H|D, \Theta'}(\delta_j^l) = \sum_{l=1}^N p(C_l = j | D_l, \Theta')$$

$$\tilde{N}_{ijk} = \sum_{l=1}^N E_{H|D, \Theta'}(\delta_{ijk}^l) = \sum_{l=1}^N p(X_{il} = k, C_l = j | D_l, \Theta')$$

- Important:**

- Use expected counts instead of counts !!!
- Re-estimate the parameters using expected counts



## EM for BBNs

- The same result applies to learning of parameters of **any Bayesian belief network** with discrete-valued variables

$$Q(\Theta | \Theta') = E_{H|D, \Theta'} \log P(H, D | \Theta, \xi)$$

$$\theta_{ijk} = \frac{\tilde{N}_{ijk}}{\sum_{k=1} r_i} \leftarrow \text{Parameter value maximizing } Q$$

$$\tilde{N}_{ijk} = \sum_{l=1}^N p(x_i^l = k, pa_i^l = j | D^l, \Theta')$$

may require inference

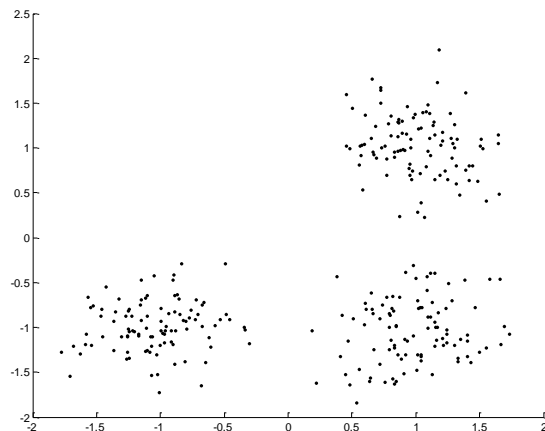
- Again:**
  - Use expected counts instead of counts

CS 2750 Machine Learning

## Gaussian mixture model

**Goal:** We want a model of  $p(\mathbf{x})$

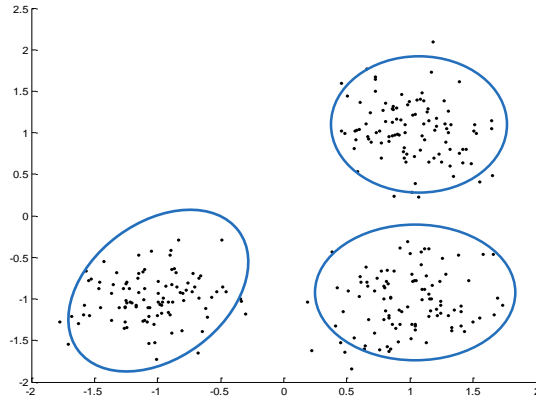
**Problem:** a multidimensional Gaussian not a good fit to data



## Gaussian mixture model

**Goal:** We want a model of  $p(\mathbf{x})$

**Problem:** a multidimensional Gaussian not a good fit to data  
But three different Gaussian may do



## QDA

**A generative classifier model** – model of  $p(\mathbf{x}, c)$

The class labels are known.

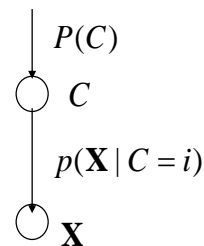
**Model:**

$$p(C = i)$$

= probability of a data point coming  
from class  $C=i$

$$p(\mathbf{x} | C = i) \approx N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$$

= class conditional density (modeled as a Gaussian)  
for class  $i$



## QDA

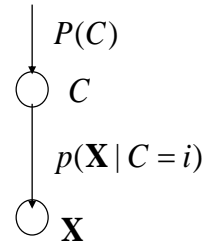
- Generative classifier model – model of  $\mathbf{p}(\mathbf{x},c)$
- **The ML estimate of parameters**

$$N_i = \sum_{j:C_j=i} 1$$

$$\tilde{\pi}_i = \frac{N_i}{N}$$

$$\tilde{\boldsymbol{\mu}}_i = \frac{1}{N_i} \sum_{j:C_j=i} \mathbf{x}_j$$

$$\tilde{\boldsymbol{\Sigma}}_i = \frac{1}{N_i} \sum_{j:C_j=i} (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T$$



CS 2750 Machine Learning

## Gaussian mixture model

The model of  $p(\mathbf{x})$  based on the model of  $\mathbf{p}(\mathbf{x},c)$

- **Class labels are not known**

$$p(\mathbf{x}) = \sum_{i=1}^k p(C = i) p(\mathbf{x} | C = i)$$

**The same model as QDA:**

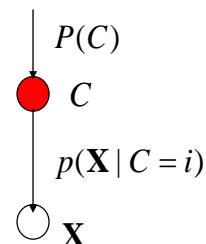
$$p(C = i)$$

= probability of a data point coming  
from class  $C=i$

$$p(\mathbf{x} | C = i) \approx N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$$

= class conditional density (modeled as a Gaussian)  
for class  $i$

**Special feature:  $C$  is hidden !!!!**



CS 2750 Machine Learning

## Gaussian mixture model

- In the Gaussian mixture Gaussians are not labeled
- We can apply **EM algorithm**:
  - re-estimation based on the class posterior

$$h_{il} = p(C_l = i | \mathbf{x}_l, \Theta') = \frac{p(C_l = i | \Theta') p(x_l | C_l = i, \Theta')}{\sum_{u=1}^m p(C_l = u | \Theta') p(x_l | C_l = u, \Theta')}$$

$$N_i = \sum_l h_{il} \quad \leftarrow \text{Count replaced with the expected count}$$

$$\tilde{\pi}_i = \frac{N_i}{N}$$

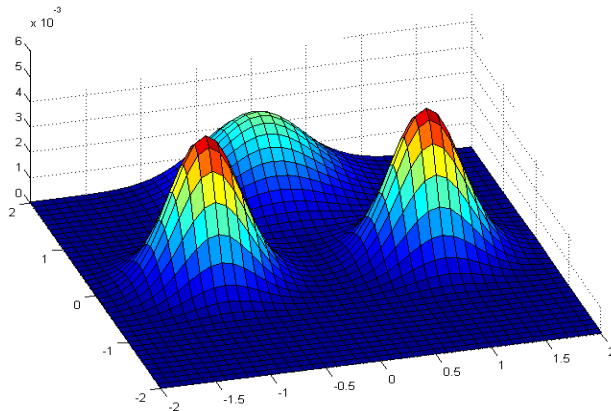
$$\tilde{\boldsymbol{\mu}}_i = \frac{1}{N_i} \sum_l h_{il} \mathbf{x}_l$$

$$\tilde{\boldsymbol{\Sigma}}_i = \frac{1}{N_i} \sum_l h_{il} (\mathbf{x}_l - \boldsymbol{\mu}_i)(\mathbf{x}_l - \boldsymbol{\mu}_i)^T$$

CS 2750 Machine Learning

## Mixture of Gaussians

- Density function  $p(\mathbf{x})$  for the Mixture of Gaussians model



CS 2750 Machine Learning

## Gaussian mixture algorithm

- **Special case:** fixed covariance matrix for all hidden groups (classes) and uniform prior on classes

- **Algorithm:**

Initialize means  $\mu_i$  for all classes  $i$

Repeat two steps until no change in the means:

1. Compute the class posterior for each Gaussian and each point (a kind of responsibility for a Gaussian for a point)

**Responsibility:** 
$$h_{il} = \frac{p(C_l = i | \Theta') p(x_l | C_l = i, \Theta')}{\sum_{u=1}^m p(C_l = u | \Theta') p(x_l | C_l = u, \Theta')}$$

2. Move the means of the Gaussians to the center of the data, weighted by the responsibilities

**New mean:** 
$$\mu_i = \frac{\sum_{l=1}^N h_{il} \mathbf{x}_l}{\sum_{l=1}^N h_{il}}$$

CS 2750 Machine Learning

## Gaussian mixture model: Gradient ascent

- A set of parameters

$$\Theta = \{\pi_1, \pi_2, \dots, \pi_m, \mu_1, \mu_2, \dots, \mu_m\}$$

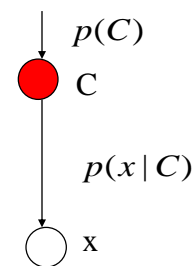
Assume unit variance terms and fixed priors

$$P(\mathbf{x} | C = i) = (2\pi)^{-1/2} \exp\left\{-\frac{1}{2}\|x - \mu_i\|^2\right\}$$

$$P(D | \Theta) = \prod_{l=1}^N \sum_{i=1}^m \pi_i (2\pi)^{-1/2} \exp\left\{-\frac{1}{2}\|x_l - \mu_i\|^2\right\}$$

$$l(\Theta) = \sum_{l=1}^N \log \sum_{i=1}^m \pi_i (2\pi)^{-1/2} \exp\left\{-\frac{1}{2}\|x_l - \mu_i\|^2\right\}$$

$$\frac{\partial l(\Theta)}{\partial \mu_i} = \sum_{l=1}^N h_{il} (x_l - \mu_i) \quad \text{- easy on-line update}$$



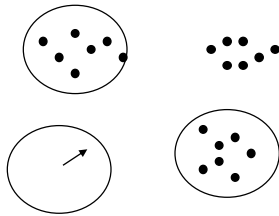
CS 2750 Machine Learning

## EM versus gradient ascent

### Gradient ascent

$$\mu_i \leftarrow \mu_i + \alpha \sum_{l=1}^N h_{il} (x_l - \mu_i)$$

### Learning rate

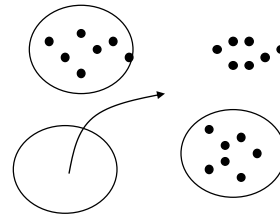


Small pull towards distant uncovered data

### EM

$$\mu_i \leftarrow \frac{\sum_{l=1}^N h_{il} \mathbf{x}_l}{\sum_{l=1}^N h_{il}}$$

### No learning rate



Renormalized – big jump in the first step

CS 2750 Machine Learning

## K-means approximation to EM

### Mixture of Gaussians with the fixed covariance matrix:

- posterior measures the responsibility of a Gaussian for every point

$$h_{il} = \frac{p(C_l = i | \Theta') p(x_l | C_l = i, \Theta')}{\sum_{u=1}^m p(C_l = u | \Theta') p(x_l | C_l = u, \Theta')}$$

- Re-estimation of means:**

$$\mu_i = \frac{\sum_{l=1}^N h_{il} \mathbf{x}_l}{\sum_{l=1}^N h_{il}}$$

- K- Means approximations**

- Only the closest Gaussian is made responsible for a data point**

$$h_{il} = 1 \quad \text{If } i \text{ is the closest Gaussian}$$

$$h_{il} = 0 \quad \text{Otherwise}$$

- Results in moving the means of Gaussians to the center of the data points it covered in the previous step

CS 2750 Machine Learning

## **K-means algorithm**

### **K-Means algorithm:**

Initialize  $k$  values of means (centers)

Repeat two steps until no change in the means:

- Partition the data according to the current means (using the similarity measure)
- Move the means to the center of the data in the current partition

- **Used frequently for clustering data**