CS 2750 Machine Learning
Lecture 18

Expectation Maximization (EM)
Mixture of Gaussians

Milos Hauskrecht

milos@cs.pitt.edu

5329 Sennott Square

Learning probability distributions

Basic learning settings:

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

Objective: find parameters $\hat{\Theta}$ that describe p($\boldsymbol{X}$) based on D

Assumptions considered so far:

– Known parameterizations

– No hidden variables

– No-missing values

This lecture: learning with hidden variables and missing values

Learning with hidden variables and missing values

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

- **Data consists of instances with values assigned to variables**
- **But sometimes the values are hidden or missing**

Data

x_1	x_2			x_5
T	T	F	T	T
T	-	T	F	F
F	T	-	F	T

Missing values

Extended Data

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

Hidden variable h (all values missing)

Density estimation

Goal: Find the best set of parameters $\hat{\Theta}$ based on data

Estimation criteria:

- **ML** $\max_{\Theta} p(D \mid \mathbf{\Theta}, \xi)$
- **Bayesian** $p(\mathbf{\Theta} \mid D, \xi)$

Why hidden vars and missing values are a problem? The parameter estimation for models defined by a set of parameters does not decompose to a set of independent estimation tasks

Approaches for ML estimates: general optimization methods: gradient-ascent, conjugate gradient, Newton-Rhapson, etc

An alternative optimization method:

- **Expectation-maximization (EM) method**
  - Suitable when there are missing or hidden values
  - **Takes advantage of the structure of the belief network**

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 Θ'
2. **Maximization step**. Compute the new estimates of Θ by considering the expectations of the different value completions

Stop when no improvement possible

General EM

Current parameters Θ' estimating P(x)

(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 unknown value assignments using Θ'

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

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

General EM

Current parameters Θ' estimating P(x)

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

$x_1 \quad x_2 \quad \quad \quad x_5 \quad h$

T	T	F	T	T	-
T	-	T	F	F	-
F	T	-	F	T	-

Calculate expectations for all unknown value assignments using Θ'

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

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

General EM

Current parameters Θ'

(2) Maximization step. Compute the new improved estimates of parameters Θ 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 Θ

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

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

$$\Rightarrow \Theta$$

EM details

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

Derivation

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

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

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

Log-likelihood of data

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

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

$$\underbrace{\log P(D\mid\Theta,\xi)} = Q(\Theta\mid\Theta') + H(\Theta\mid\Theta')$$

Log-likelihood of data

EM algorithm

Algorithm (general formulation)

Initialize parameters Θ

Repeat

Set $\Theta' = \Theta$

1. **Expectation step**

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

2. **Maximization step**

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

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

Questions: Why this leads to the ML estimate ?

What is the advantage of the algorithm?

EM algorithm

Question: Why is the EM algorithm correct?

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

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

Difference in log-likelihoods (current and next step)

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

≥ 0

Kullback-Leibler (KL) divergence

- distance between 2 distributions)

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

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

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

EM algorithm

Difference in log-likelihoods

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

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

Thus

by **maximizing Q we improve (!!!) the log-likelihood**

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

EM is a first-order optimization procedure

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

No need to adjust the learning rate !!!!

EM advantages

Key advantages:

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

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

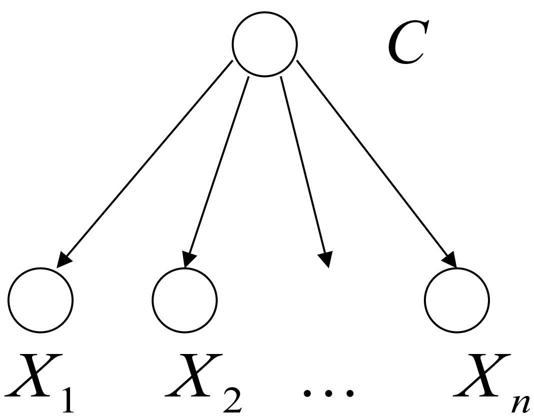
  – has a nice form and the maximization of $Q(\Theta \mid \Theta')$ can be carried out in **the closed form**
- No need to compute Q before maximizing
- We directly optimize
  – using quantities corresponding to expected counts

Naïve Bayes with a hidden class and missing values

Assume:

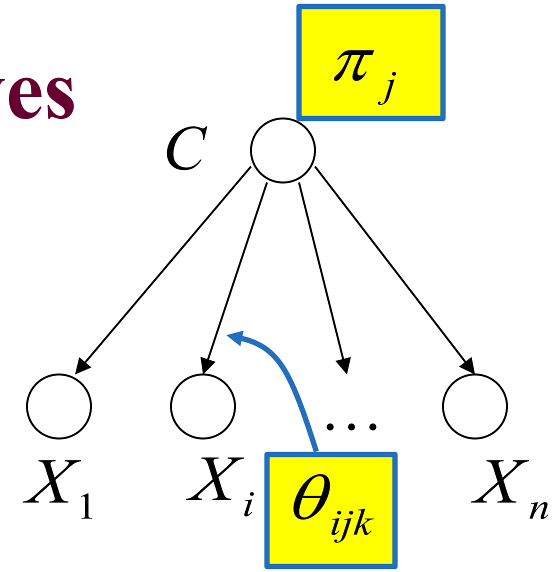
- $P(\mathbf{X})$ is modeled using a Naïve Bayes model with a hidden class variable C
- Missing entries (values) for attributes in the dataset D

Hidden class variable



Attributes are independent given the class

EM for the Naïve Bayes



- **Use EM to iteratively optimize**

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

- **Parameters:**

  π_j prior on class j

  θ_{ijk} probability of an attribute i having value k given class j

- **Indicator variables :**

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

  δ_{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)

EM for the Naïve Bayes model

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

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

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

Complete loglikelihood

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

Expectation over H

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

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

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

Substitutes 0,1
with the expected value

EM for the Naïve Bayes model

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

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

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

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

- **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

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

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

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

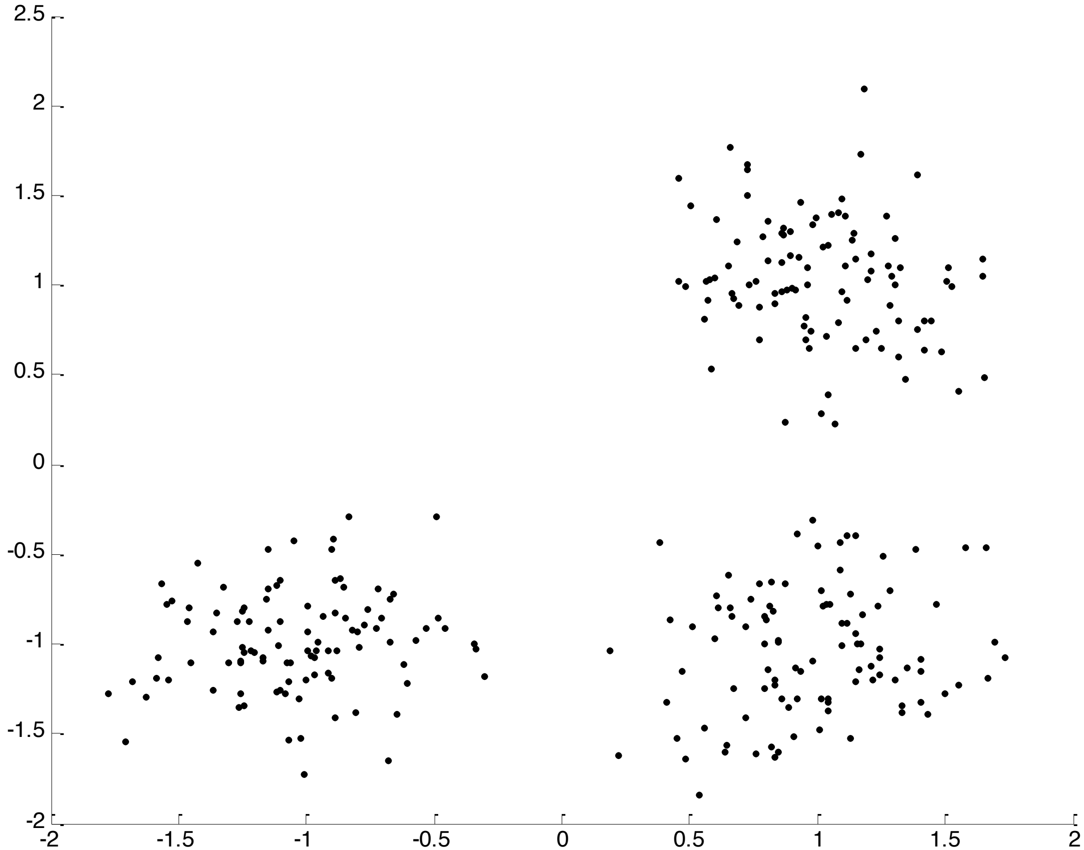
may require inference

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

Gaussian mixture model

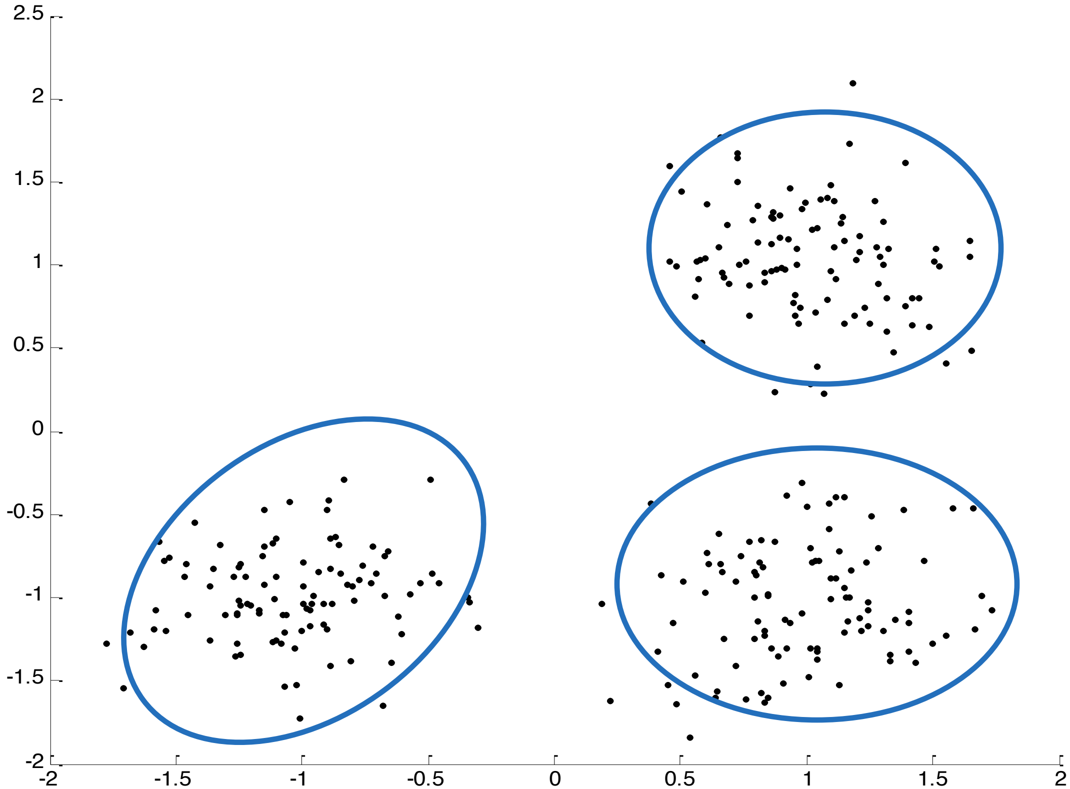
Goal: We want a model of p(**x**)

Problem: a multidimensional Gaussian not a good fit to data



Gaussian mixture model

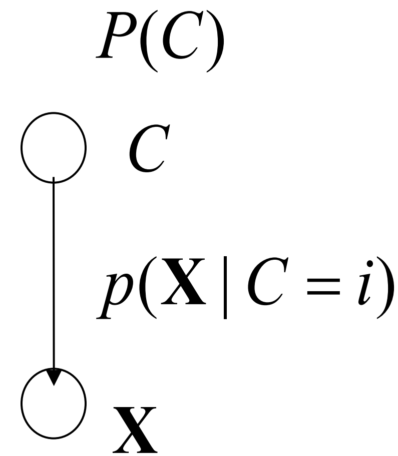
Goal: We want a density model of p(**x**)
Problem: a multidimensional Gaussian not a good fit to data
But three different Gaussians may do well



Recall QDA

Generative classifier model:

- models of p(**x**,C) as p(**x**|C).p(C)
- <u>**Class labels are known**</u>



Model:

$$p(C = i)$$

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

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

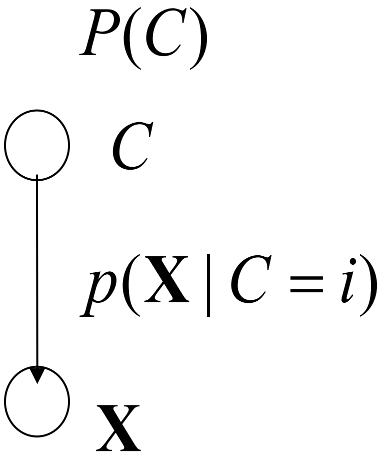
= class conditional density (modeled as a Gaussian) for class i

Recall QDA

Generative classifier model:

- models of p($\mathbf{x}$,C) as p($\mathbf{x}$|C).p(C)
- <u>**Class labels are known**</u>

The ML estimate of parameters



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

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

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

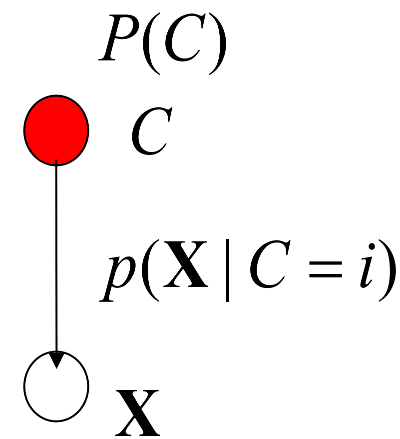
$$\tilde{\boldsymbol{\Sigma}}_i = \frac{1}{N_i} \sum_{j:C_l = i} (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T$$

Gaussian mixture model

The model of p(**x**) based on the model of p(**x**,C)

- **<u>Class labels are not known</u>**

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



The same model as QDA:

$p(C = i)$

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

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

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

<u>Important:</u> ***C* values are hidden !!!!**

Gaussian mixture model

Gaussian mixture: we do not know what Gaussian generates $\mathbf{x}$

- We can apply **EM algorithm**:
  - re-estimation based on the class posterior

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

$$N_i = \sum_l h_{il}$$

Count replaced with the expected count

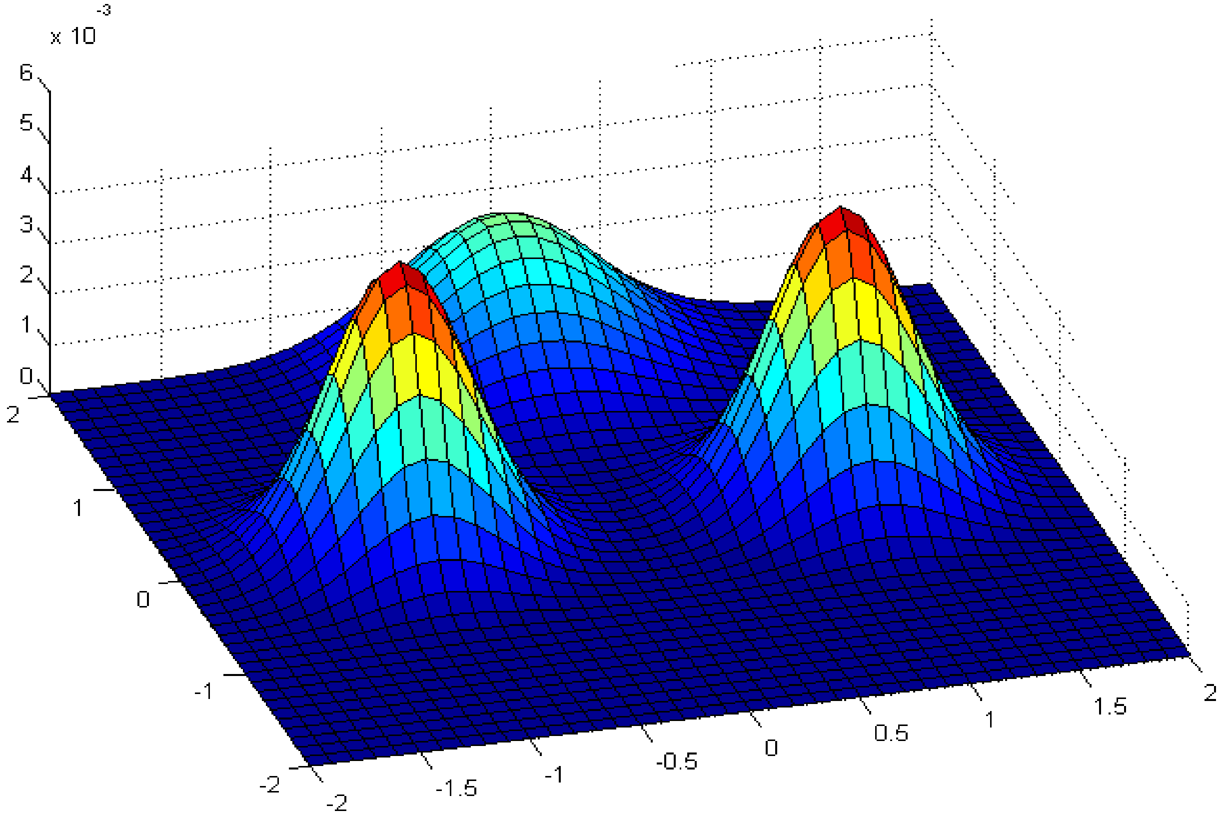
$$\tilde{\pi}_i = \frac{N_i}{N}$$

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

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

Gaussian mixture

- Density function p(**x**) for the Mixture of Gaussians model



GM algorithm

Assume special case of the GM: a fixed and shared covariance matrix for all hidden groups and uniform prior on groups

- **Algorithm**:

  Initialize means $\boldsymbol{\mu}_i$ 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:

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

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

New mean:

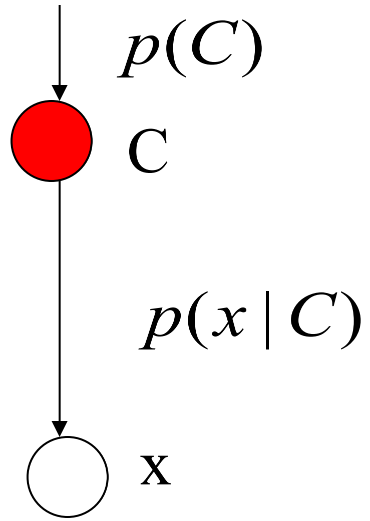
$$\boldsymbol{\mu}_i = \frac{\sum_{l=1}^{N} h_{il} \mathbf{x}_l}{\sum_{l=1}^{N} h_{il}}$$

Gaussian mixture model: Gradient ascent

- A set of parameters

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

Assume unit variance terms and fixed priors



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

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

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

$$\frac{\partial l(\Theta)}{\partial \mu_i} = \sum_{l=1}^{N} h_{il}(x_l - \mu_i)$$

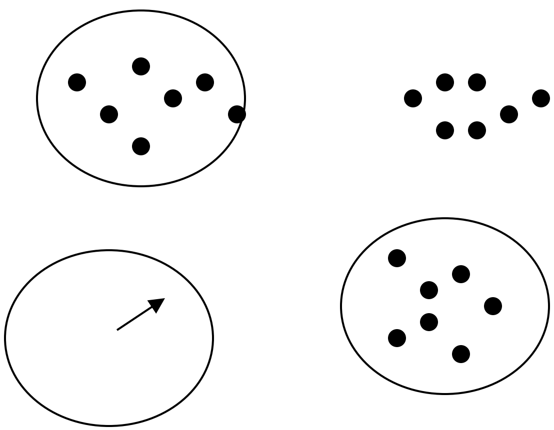
- easy on-line update

EM versus gradient ascent

Gradient ascent

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

Learning rate

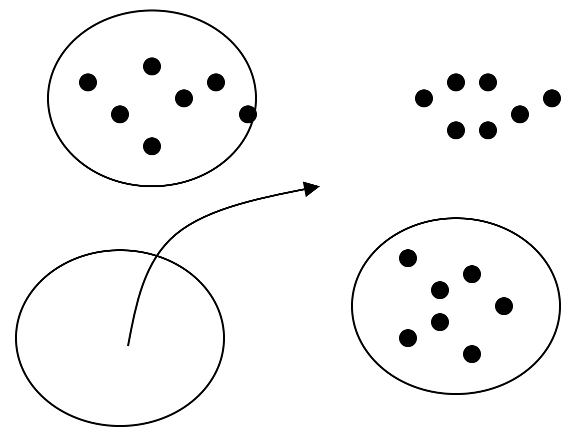


Small pull towards distant uncovered data

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

K-means approximation to EM

Mixture of Gaussians with the fixed covariance matrix:

- posterior measures the responsibility of a Gaussian for every point

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

- **Re-estimation of means:** $\boldsymbol{\mu}_i = \dfrac{\sum_{l=1}^{N} h_{il}\mathbf{x}_l}{\sum_{l=1}^{N} h_{il}}$

- **K- Means approximations (next lecture)**
- **Only the closest Gaussian is made responsible for a data point**

$h_{il} = 1$ If i is the closest Gaussian $\quad h_{il} = 0$ Otherwise

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

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:

Basic clustering problem:

- distribute data into *k* different groups such that data points similar to each other are in the same group