Alternative Clustering Methods

Lecture Overview

Cluster Similarity – Ward Measure

- Similarity of two clusters is based on the increase in squared error when two clusters are merged
    - Similar to group average if distance between points is squared norm
- Less susceptible to noise and outliers
- Biased towards globular clusters
- Hierarchical analogue of K-means
    - Can be used to initialize K-means

Wards Method

Instead of measuring the distance directly, it analyzes the variance of clusters. Ward's is said to be the most suitable method for quantitative variables, not binary variables:

$$\Delta(A,B) =$$

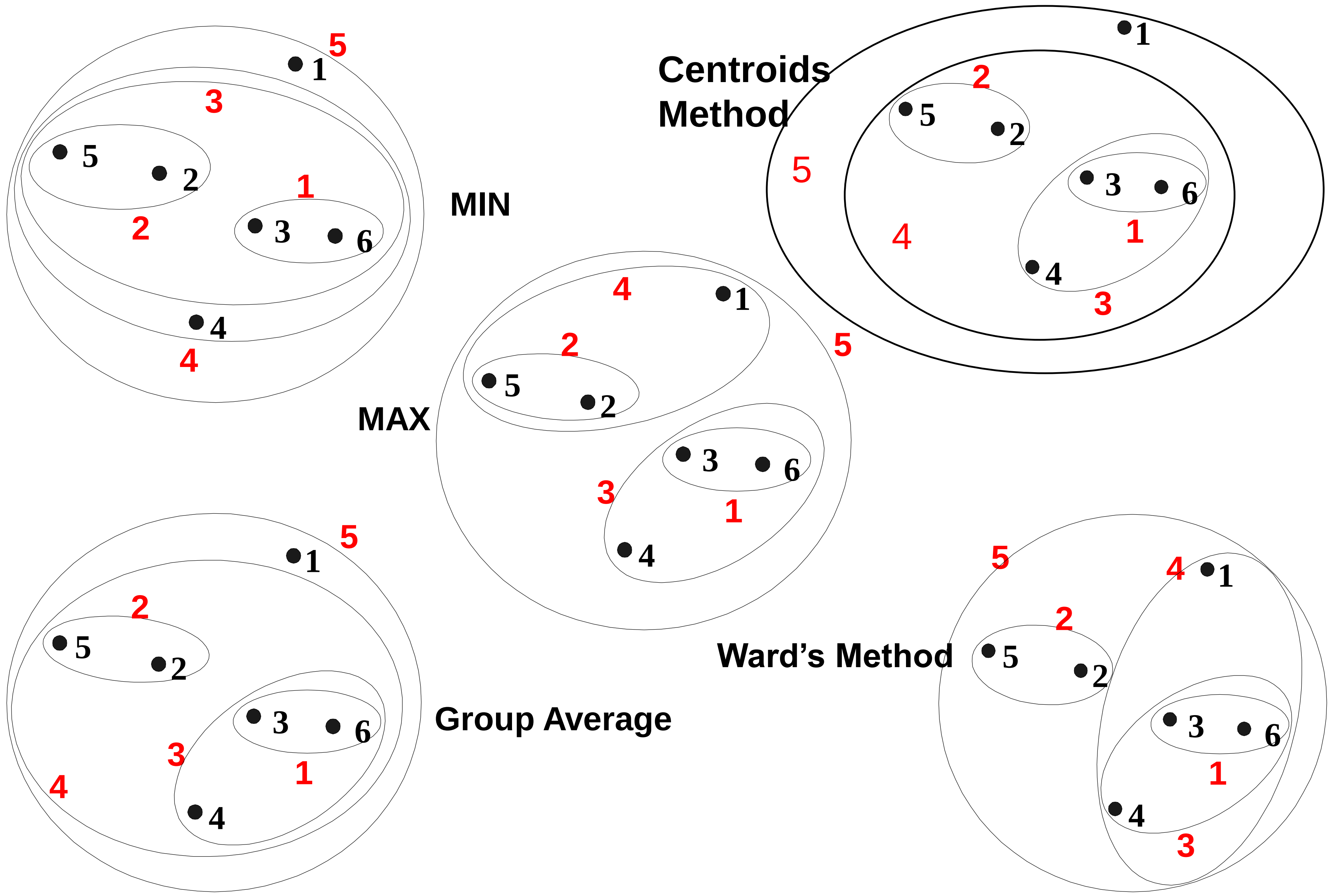
$$= \sum_{i\in A\cup B} ||\vec{x}_i - \vec{m}_{A\cup B}||^2 - \sum_{i\in A} ||\vec{x}_i - \vec{m}_A||^2 - \sum_{i\in B} ||\vec{x}_i - \vec{m}_B||^2$$

$$= \frac{n_A n_B}{n_A + n_B} ||\vec{m}_A - \vec{m}_B||^2$$

– where m's are cluster centroids (means) and n's are sizes of clusters

- Thus in Ward the change in SSE if we join two clusters is used a the distance between clausters
- We can view it as the distance based on measure that is sum of squares

Hierarchical Clustering: Comparison



Agglomerative Clustering: Time and Space

- Straightforward agglomerative clustering algorithm is known as AGNES (AGglomerative NESting – implemented in **R** as $hclust$ or $agnes$ functions)
- $O(n^2)$ space since it uses the proximity matrix, where n is the number of datapoints.
- $O(n^3)$ time in many cases
  - There are n steps and at each step the size, n^2, proximity matrix must be updated and searched
  - Complexity can be reduced to $O(n^2 \log(n)\ )$ time for some approaches

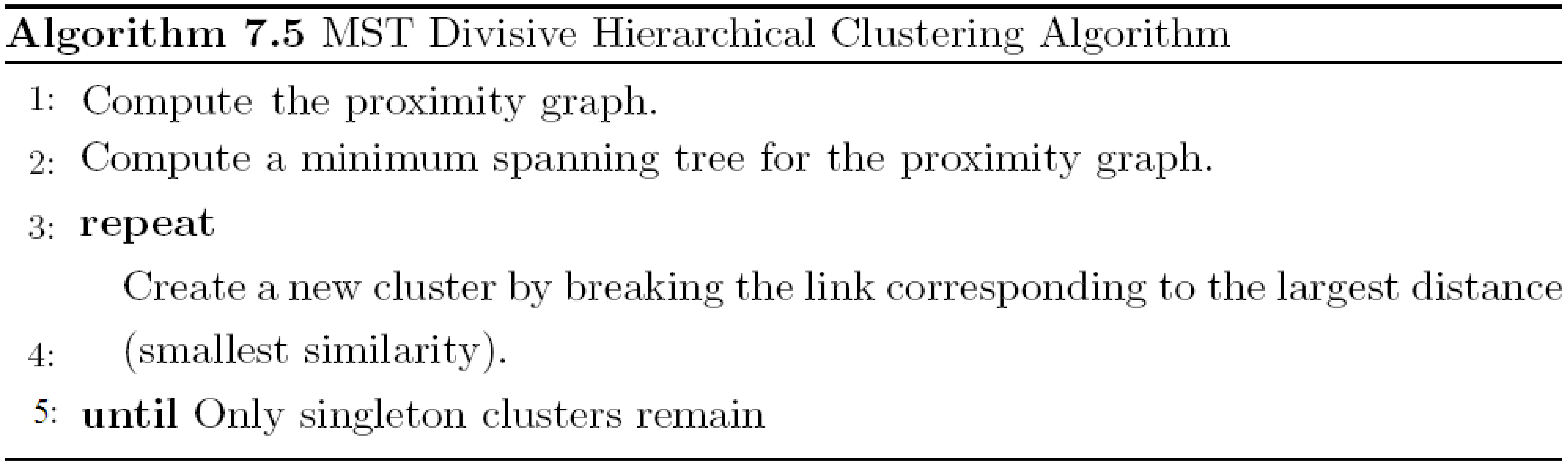
Lecture Overview

Sparcified K —Nearest Neighbor Graph

1. Compute limited distance graph
   - Compute similarity/distance matrix for all datapoints
   - Set the threshold value for the similarity/distance
   - Set all values lover (higher) than threshold to 0
2. Compute K-neighborhood graph from the result
   - Treat the matrix as a weighted matrix of a graph
   - For each datapoint (row) retain only k-nearest (largest/smallest) entries in each row and column, set others to 0
   - Restore symmetry in the graph by adding non-symmetric links back

MST: Divisive Hierarchical Clustering

- Construct sparcified K-NN graph to use as basic proximity graph
- Use MST for constructing hierarchy of clusters
  - Equivalent to Single link

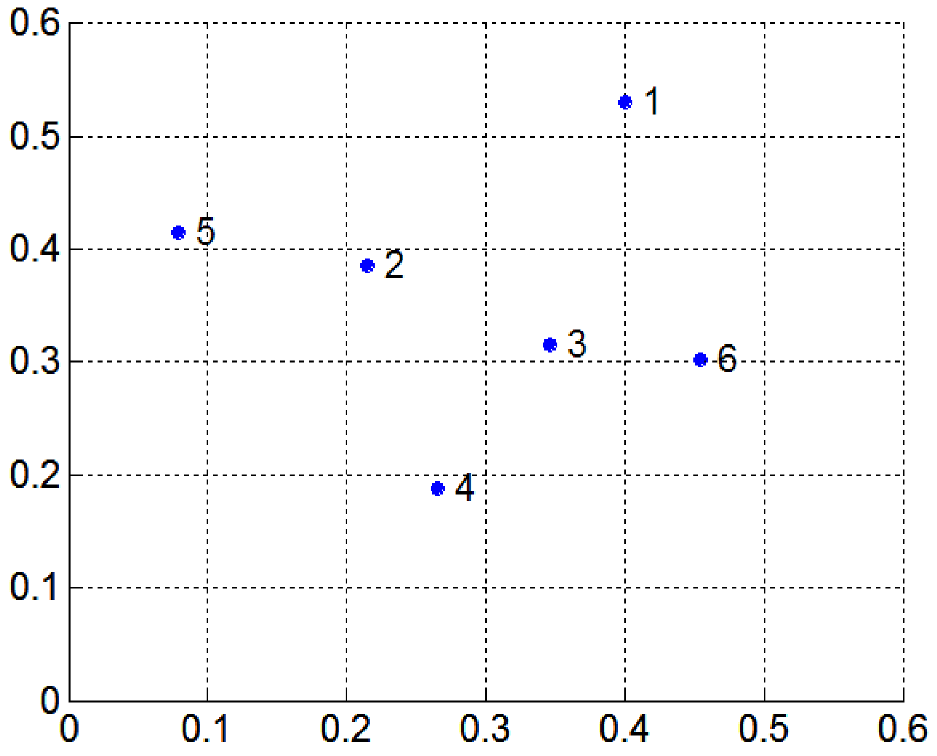


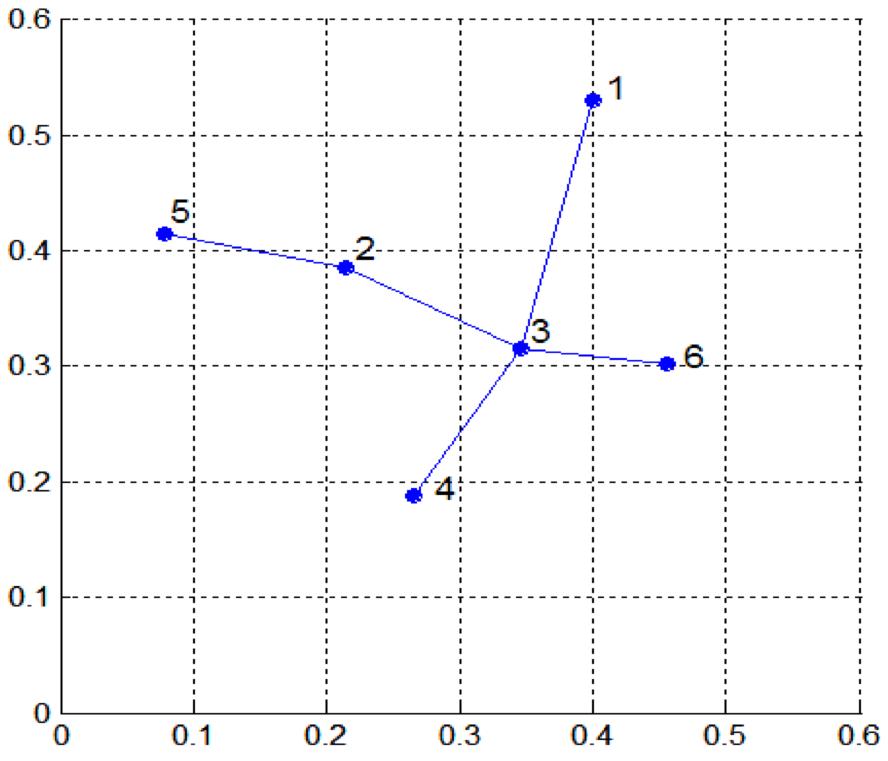
Algorithm 7.5 MST Divisive Hierarchical Clustering Algorithm

```
1: Compute the proximity graph.
2: Compute a minimum spanning tree for the proximity graph.
3: repeat
4:     Create a new cluster by breaking the link corresponding to the largest distance
       (smallest similarity).
5: until Only singleton clusters remain
```

MST: Divisive Hierarchical Clustering

- Build MST (Minimum Spanning Tree)
  - Start with a tree that consists of any point
  - In successive steps, look for the closest pair of points (p, q) such that one point (p) is in the current tree but the other (q) is not
  - Add q to the tree and put an edge between p and q





Divisive Analysis - DIANA

- Starts: single cluster = the set of all objects.
- In each step, the cluster C with largest diameter $\frac{\sum_{x\in C}\sum_{y\in C}\|x-y\|}{|C|\cdot(|C|-1)}$ is selected to be divided into two clusters.
- An data point $\boldsymbol{m}$ in C that has largest average dissimilarity to other objects within C is identified.
- $\boldsymbol{m}$ becomes a medoid of a 'splinter group.' A data point y is reassigned to the splinter group if it is closer to y than to any point in 'old party.'

DIANA in R (import and print)

```
library(cluster);library(png);library(graphics)
readImage<-readPNG('example.png')
dm <- dim(readImage)
rgbImage <- data.frame(
  x=rep(1:dm[2], each=dm[1]),
  y=rep(dm[1]:1, dm[2]),
  r.value=as.vector(readImage[,,1]),
  g.value=as.vector(readImage[,,2]),
  b.value=as.vector(readImage[,,3]))
plot(y ~ x, data=rgbImage, main="Image for Pattern Analysis",
     col = rgb(rgbImage[c("r.value", "g.value", "b.value")]),
     asp = 1, pch = ".")
```

DIANA in R – Clusterize and Plot -compare

```
mimg<-as.matrix(readImage[,,2])
ind<-which(mimg!= 1, arr.ind=TRUE)
s<-sample(1:dim(ind)[1],2000,F)
sind<-ind[s,c(2,1)]
dv <- diana(sind, metric = "manhattan", stand = TRUE)
plot(dv) #see the tree
dv1 <- cutree(as.hclust(dv), k = 6) #there are 6 clusters
table(dv1)
simg<-data.frame(sind) #convert back to data frame
names(simg)[names(simg)=="row"] <- "x"
names(simg)[names(simg)=="col"] <- "y"
simg$clust<-dv1[1:2000]
plot(simg$x,simg$y,col=simg$clust)
```

Lecture Overview

Clustering Assuming Generative Model

1. How was the data generated?
   - Data consists of a number of separate component classes $\{1, \ldots, k\}$, i.e. $X = \bigcup_{i=1}^{k} Z_i$ such that $Z_i \cap Z_j \neq \emptyset$
   - A particular data point $\bar{z}$ is generated by randomly picking a component of data according to unknown probability distribution D^0 over the component set $\{1, \ldots, k\}$ with probability $\Pr_{D^0}(Y = i)$
   - Generate data point $\overline{x}$ by sampling from distribution $D^i \sim Z_i$ over data of already picked component i with probability $\Pr_{D^i}(\overline{z}|Y)$
2. Challenge: we need to estimate model parameters for $D^0, D^1, \ldots, D^k$ without having a training set.

Y	X_1	X_2
??	0.1	2.1
??	0.5	-1.1
??	0.0	3.0
??	-0.1	-2.0
??	0.2	1.5
…	…	…

Missing Data

- Let $D = \{\bar{x}_1, \dots \bar{x}_n\}$ be a set of n observations drawn from X.
- Let $\mathrm{H} = \{y_1, \dots, y_n\}$ be a set of n values of a hidden variable Y.

  to $\bar{x}_1$ correspond class y_1

  where $Y = \{1, \dots, k\}$
- If we have all data labelled then in generative model we have exactly the model we used in Naïve Bayesian Classification
  - We used max likelihood to estimate parameters
  - But we do not have training labels. Can we still use max likelihood to estimate parameters? And to estimate prior's at the same time?
- We think of clustering as a problem of estimating missing data.
- The missing data are the cluster labels.
  - Clustering is only one example of a missing data problem. Several other problems can be formulated as missing data problems.

The Optimization Problem

- Let $\bar{\theta}$ be vector of parameters of our model. Want to find to $\arg\max_{\overline{\theta}} \Pr\left(D \middle| \overline{\theta}\right) = \arg\max_{\overline{\theta}} \sum_H \Pr\left(D, H \middle| \overline{\theta}\right)$
- Instead we maximize log-likelihood of the observed data $\arg\max_{\overline{\theta}} l(\overline{\theta})$ where $l\left(\overline{\theta}\right) = \log \sum_H \Pr(D, H|\overline{\theta})$ is log likelihood of $\overline{\theta}$
- Not only do we have to estimate $\overline{\theta}$ (model parameters vector) but also H.
- Hard to maximize directly (need to differentiate w.r.t. many variables, in most cases closed form solution is non-existent
- Can we do it some other way?

Lecture Overview

EM Algorithm – the Idea

- If we knew assignments of classes we could learn component models easily
  - We did so when constructing Naïve Bayesian Classifier
- If we knew the model we could assign label easily
  - This is just classification (i.e. applying Naïve Bayesian classifier)
- We deal with missing labels by iteratively alternating between two steps:
  1. Expectation: Fix model and estimate missing labels
  2. Maximization: Fix missing labels (more often than not it means fix a distribution over the missing labels) and find the model that maximizes the expected log-likelihood of the data

Simple Illustration of the Idea

- On the exam some people got A, some people B, some people got C, and some people D. We know that
  - probability of getting high grade and low grade is about equal
  - probability of getting C twice as much as probability of getting B
  - The total number of people who got high grade is $h = 9$ out of $m = 20$ students
- What should we expect to be the grade of a given person?
- The model: let a, b, c, d denote the number of people who got grades A,B,C,D respectively, so that $a + b = h$. Let g be grade random variable.

$$\Pr(g = \mathrm{A} \vee g = \mathrm{B}) = \Pr(g = \mathrm{C} \vee g = D) = \frac{1}{2}$$

Let p denote the probability of getting B. Then $\Pr(g = A) = \frac{1}{2} - p$, $\Pr(g = C) = 2p$, $\Pr(g = D) = \frac{1}{2} - 2p$

Simple Illustration of the Idea

- Expectation step: fix probability $p = \frac{1}{8}$ of B (system parameter)

$$a = \frac{\Pr(\mathrm{A})}{\Pr(\mathrm{A}) + \Pr(\mathrm{B})} h = \frac{\frac{1}{2} - p}{\frac{1}{2}} h = (1 - 2\mathrm{p})h = 6.75$$

$$d = \frac{\frac{1}{2} - 2p}{\frac{1}{2}} (m - h) = (1 - 4p)(m - h) = 5.5$$

$$b = 2ph = 2.25; \qquad c = 4p(m - h) = 5.5$$

- Maximization step: fix values of a, b, c, d (numbers of respective labels)

$$\Pr(a, b, c, d | p) = \frac{m!}{a!\, b!\, c!\, d!} \left(\frac{1}{2} - p\right)^a p^b \, (2p)^c \left(\frac{1}{2} - 2p\right)^d$$

$$\frac{\partial}{\partial p} \log \Pr(a, b, c, d | p) = -\frac{a}{1 - 2p} + \frac{b}{p} + \frac{c}{p} - \frac{d}{1 - 4p} = 0$$

From which $p \approx 0.2$

Repeat with new values

EM Algorithm – How to Implement

Let $Q(H)$ be the prior probability distribution on the missing data

$$
\begin{aligned}
\ell(\theta) &= log \sum_{H} p(D, H|\theta) \\
&= log \sum_{H} Q(H) \frac{P(D, H|\theta)}{Q(H)} \\
&\geq \sum_{H} Q(H) log \frac{P(D, H|\theta)}{Q(H)} \\
&= \sum_{H} Q(H) log\, P(D, H|\theta) + \sum_{H} Q(H) log \frac{1}{Q(H)} \\
&= F(Q, \theta)
\end{aligned}
$$

Inequality is because of Jensen's inequality: $\varphi(E[f(X)]) \leq E[\varphi(f(X))]$ for convex φ

- This means that the $F(Q, \theta)$ is a lower bound on $l(\theta)$, so we can maximize $F(Q, \theta)$ instead of $l(\theta)$
- Notice that the log of sum became a sum of logs

EM Algorithm – How to Implement (cont.)

- Want to find θ to maximize $\Pr(D\,|\theta)$
- Instead, find θ, Q to maximize

$$F(\theta, Q) \; = \sum\nolimits_H Q(H) \log \Pr(D, H\,|\theta) - \sum\nolimits_H Q(H) \log Q(H)$$
$$= E_Q[\,\log \Pr(D, H\,|\theta) - \log Q(\mathrm{H})]$$

- Note that as we have seen $F(\theta, Q)$ has the same local and global optimums as $l(\theta)$, and hence same local and global optimums as $\Pr(D|\theta)$
- We can find (at least local) optimum of $F(\theta, Q)$ by alternating between
  - holding Q fixed and optimizing θ
  - holding θ fixed and optimizing Q

EM Algorithm

Repeat

1. E-step: maximizing F with respect to Q (when $\overline{\theta}$ is assumed constant)

$$Q^{k+1} = \operatorname{argmax}_Q F(Q^k, \theta^k)$$

and then

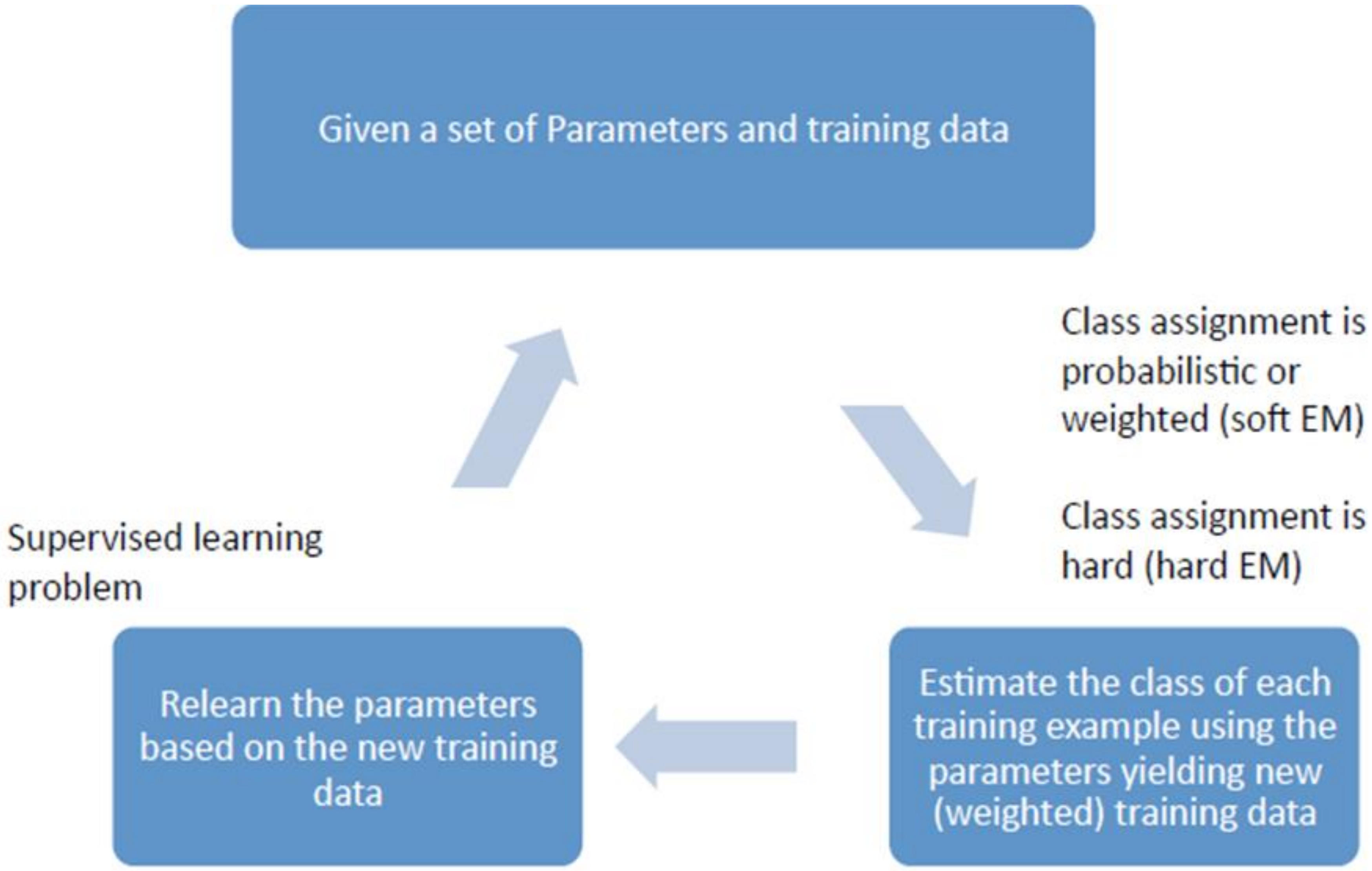
2. M-step: maximizing F with respect to $\overline{\theta}$ (when Q is assumed constant)

$$\theta^{k+1} = \operatorname{argmax}_Q F(Q^{k+1}, \theta^k)$$

until we obtain local extremum (i.e. until next step produces lower values)

- We need to pick the initial values of θ^0, Q^0
- Loop apparently converges since at each step F value increases.
- EM corresponds to gradient ascent on F. Thus, maximizes lower bound on marginal log likelihood

EM-algorithm – Pictorial View


Given a set of Parameters and training data
Class assignment is probabilistic or weighted (soft EM)
Class assignment is hard (hard EM)
Estimate the class of each training example using the parameters yielding new (weighted) training data
Relearn the parameters based on the new training data
Supervised learning problem

Lecture Overview

General Mixture Model of Data

- Data (random multivariate variable X) comes from (unknown) classes $C_1, \ldots, C_k$ (i.e. class random variale Y takes values from $\{1, \ldots, k\}$
- Each class C_i has known pdf f_i with unknown vector $\overline{\theta}_i$ of parameters
- Each class has known prior probability $P(Y = i) = P(C_i)$, and $\sum_i^k P(C_i) = 1$
- We assume that the probability density function of X is given as a *mixture model* over k clusters:

$$f(X) = \sum_{j=1}^{k} f_j(X)P(C_i) = \sum_{j=1}^{k} f_j\left(X|\overline{\theta}_j\right)P(C_i)$$

and each data point $\overline{x}_i$ is i.i.d. sampled from X so

$$f(\overline{x}_i) = f(X)$$

What is the Best Mixture Model - Intuition

- Say we have data points $\overline{x}_1, \ldots, \overline{x}_m$ that are independent identically distributed variables from mixture of classes $C_1, C_2, \ldots, C_k$.
- Each parameter vector $\overline{\theta}_i$ and prior probability $P(C_i)$ can take any value. So fix values of parameters one way, the model is $\varphi_0 = \left\{\left(\overline{\theta}_1^0, P^0(C_1)\right), \left(\overline{\theta}_2^0, P^0(C_2)\right), \ldots, \left(\overline{\theta}_k^0, P^0(C_k)\right)\right\}$, fix parameters differently and then $\varphi_1 = \{(\overline{\theta}_i^1, P^1(C_i)); i = 1, \ldots, k\}$ is another model.
- We obviously want the "best" model. $Which\ model\ is\ better?$
- Given the dataset D, define *the likelihood of a model* φ as the conditional probability $P(D|\varphi)$ of the data D given the model parameter set φ. Of course the model that has the highest likelihood must be the best.

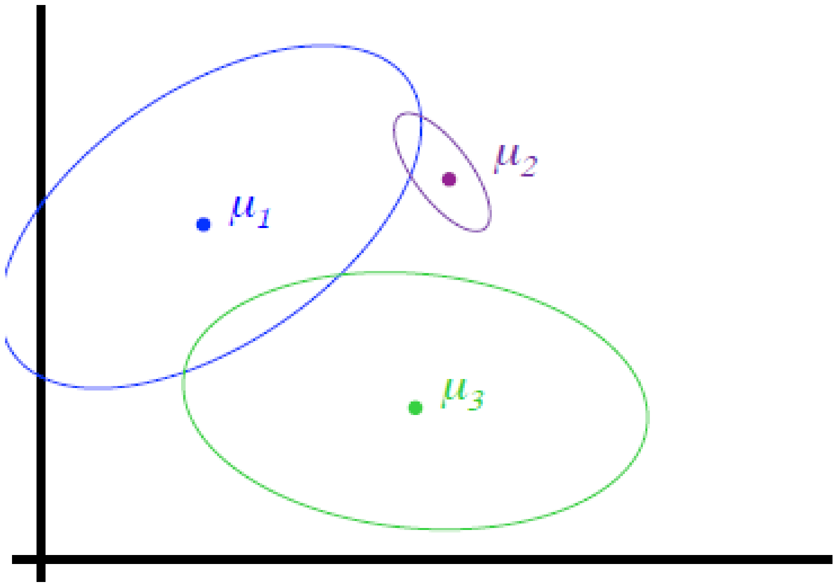
Maximum Likelihood Estimation

- Since each one out of m points $\overline{x}_{\mathrm{j}}$ is a random sample from X (i.i.d. from mixture f), the likelihood of is given as $P(D|\overline{\theta}) = \prod_{i=1}^{n} P(\overline{x}_i) \sim \prod_{i=1}^{n} f(\overline{x}_i)$ where f is the probability density function of X given as a mixture model, i.e. $f(\overline{x}_i) = \sum_{j=1}^{k} f_j(\overline{x}_i|\overline{\theta}_j)P(C_j)$
- As before, the goal of maximum likelihood estimation (MLE) is to choose the parameters that maximize the likelihood, i.e. find $\overline{\theta}^* = \mathrm{argmax}_{\overline{\theta}}\{P(D|\theta)\} = \mathrm{argmax}_{\overline{\theta}}\,\{\ln P(D|\overline{\theta})\}$, but addition is easier to deal with, so the goal is to maximize log-MLE $\mathrm{argmax}_{\overline{\theta}} \sum_{i=1}^{n} \ln\left(\sum_{j=1}^{k} f_j(\overline{x}_i|\overline{\theta}_j)P(C_j)\right)$

Lecture Overview

Gaussian Mixture Model (GMM)

- Each data point $\overline{x}_i$ in data set $D = \{\overline{x}_1, \ldots, \overline{x}_m\}$ is a vector in $\mathbb{R}^d$ where number of dimensions is the number of attributes
- In a subdomain Z_i an attribute X_a is the random variable corresponding to the a^{th} attribute.
- $Z_i \subset \mathbb{R}^d$ where $Z_i \sim N(\overline{\mu}_i, \Sigma_i)$ multivariate random variable normally distributed across attributes
- With this model likelihood of data becomes $p(\overline{x}_j) = \sum_{i=1}^{k} p(\overline{x}_j | Y = i, \overline{\mu}_i, \Sigma_i) \Pr(Y = i)$



Gaussian Mixture Model – Multivariate Gaussian

- Each cluster is normally distributed.
  - If sample values are real numbers (i.e. $d = 1$) then each cluster is distributed as

$$f(x|\mu_i, \sigma_i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} exp\left\{-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right\}$$

  where $x - \mu_i$ is the distance from class center

  - If sample values are vectors in $\mathbb{R}^d$ where $d > 1$ then each cluster is distributed as

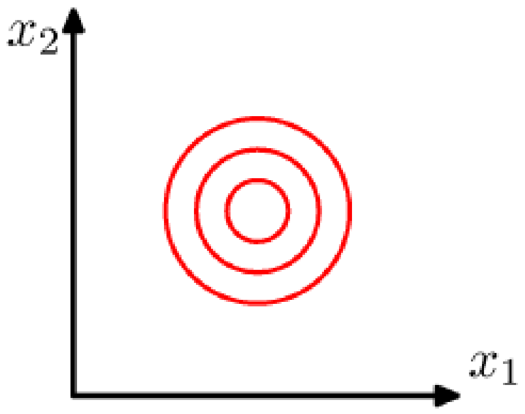
$$f\left(\overline{x}|\overline{\mu}_i, \Sigma_i\right) = \frac{1}{\sqrt{2\pi^d|\Sigma_i\ |}} exp\left\{-\frac{\left(\overline{x}-\overline{\mu}_i\right)^T \Sigma_i^{-1}(\overline{x}-\overline{\mu}_i)}{2}\right\}$$

where $|\Sigma|$ is the determinant of the covariance matrix. As in single variable case $\left(\overline{x}-\overline{\mu}_i\right)^T \Sigma_i^{-1}(\overline{x}-\overline{\mu}_i)$ is a measure of distance (i.e. Mahalanobis distance) that is a generalization of Euclidean distance (if we set $\Sigma = I_d$ then

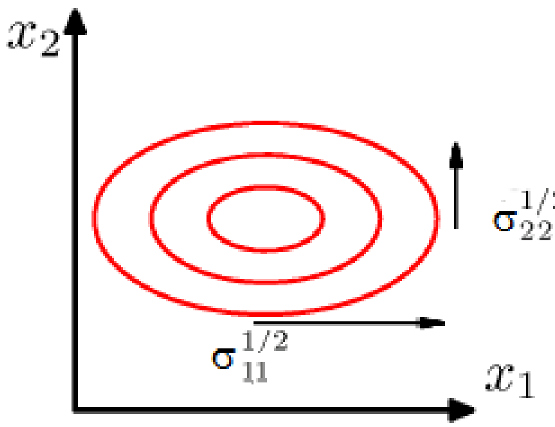
$\left(\overline{x}-\overline{\mu}_i\right)^T I_d^{-1} (\overline{x}-\overline{\mu}_i) = ||x-\mu_i||^2.$)

Multivariate Gaussian in 2 dimensions

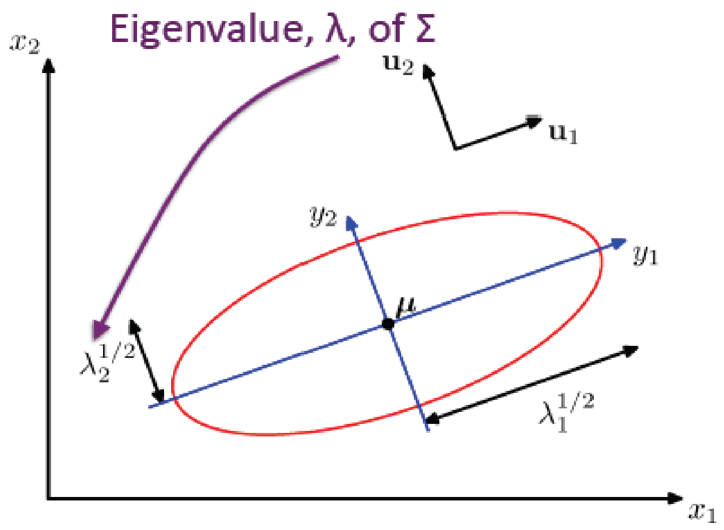
- $\Sigma = I_d$ (identity matrix)



- $\Sigma =$ diagonal matrix and diagonal X_i are independent (*ala* Gaussian NB)



- $\Sigma =$ arbitrary(semidefinite) matrix:
  - Specifies rotation (change of basis)
  - Eigenvalues specify relative elongation



GMM – EM Algorithm

- Directly maximizing the log-likelihood over $\overline{\theta}$ is hard because mixture term occurs under logarithm.
- In Gausian mixed model can use EM approach for finding the maximum likelihood estimates for the parameters $\overline{\theta}$.
  1. [Expectation step] Given the current estimates for $\overline{\theta}$ compute the cluster posterior probabilities $P(C_i|\overline{x}_j)$ using Bayes theorem
  2. [Maximization step] Using the posterior probabilities$P(C_i|\overline{x}_i)$ as weights re-estimate $\overline{\theta}$ (i.e. the parameters $\overline{\mu}_i, \Sigma_i, P(C_i)$ for each cluster C_i).

  Repeat steps E and M until $P(C_i|\overline{x}_i)$ stabilizes
- This is iterative steepest gradient descent algorithm that estimates parameters $\overline{\theta}$.

Initialization Step

Where do we get initial estimates for the parameters $\overline{\theta}$?

1. Many possibilities: run hierarchical clusterization estimate parameters
2. Use prior knowledge
3. Use no knowledge and no preprocessing:

   For each cluster $C_i, i = 1, \dots, k$, initialize:

   - the mean $\overline{\mu}_i$ by selecting a value μ_a^i for each dimension X_a uniformly at random from the range of the attribute X_a
   - the covariance matrix Σ_i as the identity matrix I_d
   - Initialize cluster prior to $P(C_i) = 1/k$, so that each cluster is equally likely

GMM - Expectation step

- Why we can improve posterior probabilities $P(C_i|\boldsymbol{x}_j)$?
- Using Bayes theorem

$$P(C_i|\overline{x}_j) = \frac{P(C_i,\overline{x}_j)}{P(\overline{x}_j)} = \frac{P(\overline{x}_j|C_i)P(C_i)}{\sum_{s=1}^{k} P(\overline{x}_j|C_s)P(C_s)}$$

Since each cluster is given by a multivariate normal distribution, for a small $\varepsilon > 0$, and an interval of size 2ε centered at $\bar{x}_j$ we have $P(\overline{x}_j|C_s) = 2\varepsilon \cdot f(\overline{x}_j|\overline{\mu}_i, \Sigma_i)$ and then

$$P(C_i|\overline{x}_j) = \frac{P(C_i,\overline{x}_j)}{P(\overline{x}_j)} = \frac{P(\overline{x}_j|C_i)P(C_i)}{\sum_{s=1}^{k} P(\overline{x}_j|C_s)P(C_s)} =$$

$$= \frac{2\varepsilon \cdot f(\overline{x}_j|\overline{\mu}_i, \Sigma_i)P(C_i)}{2\varepsilon \cdot \sum_{s=1}^{k} f(\overline{x}_j|\overline{\mu}_s, \Sigma_s)\,P(C_s)} = \frac{f(\overline{x}_j|\overline{\mu}_i, \Sigma_i)P(C_i)}{\sum_{s=1}^{k} f(\overline{x}_j|\overline{\mu}_s, \Sigma_s)\,P(C_s)}$$

Notice that $\sum_{i=1}^{k} P(C_i|\overline{x}_j) = \sum_{i=1}^{k} \frac{f(\overline{x}_j|\overline{\mu}_i, \Sigma_i)P(C_i)}{\sum_{s=1}^{k} f(\overline{x}_j|\overline{\mu}_s, \Sigma_s)P(C_s)} = 1$

The Maximization Step

- Denote the posterior probability values that are learned in expectation step $w_{ij} = P(C_i|\overline{x}_j)$.
- For each C_i compute the maximum likelihood estimates:

$$\overline{\mu}_i = \frac{\sum_{j=1}^{m} w_{ij}\overline{x}_j}{\sum_{j=1}^{m} w_{ij}} \text{ and } P(C_i) = \frac{\sum_{j=1}^{m} w_{ij}}{m}.$$

- Using pair-wise attribute view, the covariance matrix entry that correspond to features (dimensions) X_a and X_b in class C_i is estimated as

$$\sigma_{a,b}^{i} = \frac{\sum_{j=1}^{m} w_{ij}(x_j^a - \mu_i^a)(x_j^b - \mu_i^b)}{\sum_{j=1}^{m} w_{ij}}$$

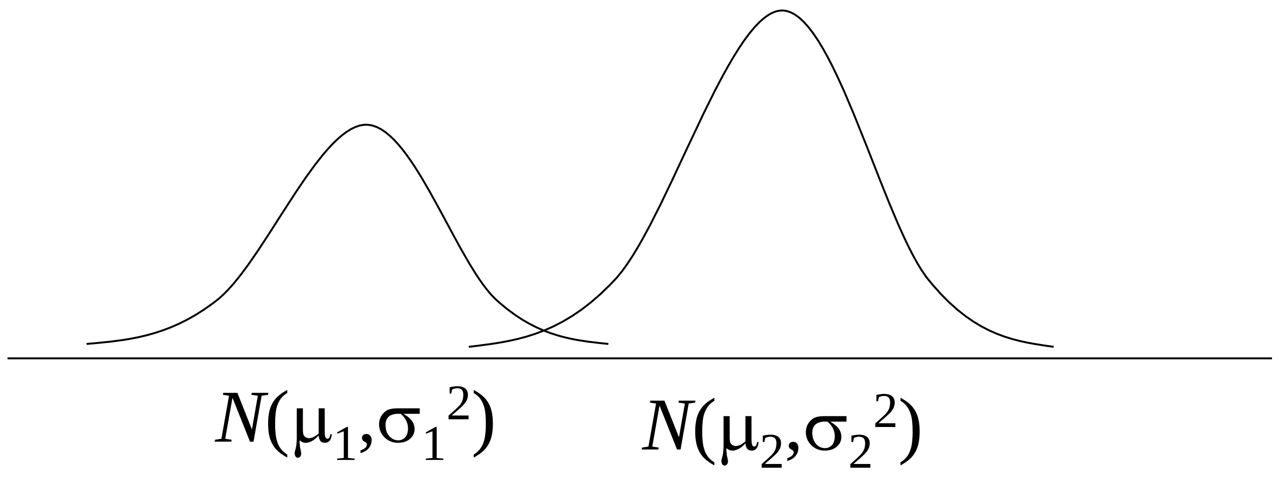
What is known about EM?

- EM: Optimizes a bound on the likelihood
- Is a type of gradient ascent
- Is guaranteed to converge to a (often local) optimum

Lecture Overview

A 2-Cluster 1-dimensional Example

- It is assumed that there are two clusters and the attribute value distributions in both clusters are normal distributions.



- Given data points $D = \left\{ \begin{matrix} x_1 = 1.0, x_2 = 1.3, x_3 = 2.2, x_4 = 2.6, x_5 = 2.8, x_6 = 5.0, \\ x_7 = 7.3, x_8 = 7.4, x_9 = 7.5, x_{10} = 7.7, x_{11} = 7.9 \end{matrix} \right\}$ that we must be clustered into two clusters C_1, C_2
- Initial assignment: means picked uniformly at random from the range [1,10], standard deviations both picked to be 1, probabilities of classes are equal

Computing Conditional Probabilities

- Then, the probabilities that sample x_i belongs to these two clusters are computed as follows:

$$\text{Prob}\left[C_1 \mid x_i\right] = \frac{\text{Prob}\left[x_i \mid C_1\right] P(C_1)}{P(x_i)} = \frac{\frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x_i-\mu_1)^2}{2\sigma_1^2}} P(C_1)}{\left(\frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x_i-\mu_1)^2}{2\sigma_1^2}} P(C_1) + \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(x_i-\mu_2)^2}{2\sigma_2^2}} P(C_2)\right)}$$

$$\text{Prob}\left[C_2 \mid x_i\right] = \frac{\text{Prob}\left[x_i \mid C_2\right] P(C_2)}{P(x_i)} = \frac{\frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(x_i-\mu_2)^2}{2\sigma_2^2}} P(C_2)}{\left(\frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x_i-\mu_1)^2}{2\sigma_1^2}} P(C_1) + \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(x_i-\mu_2)^2}{2\sigma_2^2}} P(C_2)\right)}$$

$$p_i = \frac{\text{Prob}\left[C_1 \mid x_i\right]}{\text{Prob}\left[C_1 \mid x_i\right] + \text{Prob}\left[C_2 \mid x_i\right]}, \quad \longleftarrow$$

This normalization is necessary because of rounding.

Computing Parameter Values

- The new estimated values of parameters are computed as follows:

$$\overline{\mu}_1 = \frac{\sum_{i=1}^{11} p_i \overline{x}_i}{\sum_{i=1}^{11} p_i} \text{ and } \overline{\mu}_1 = \frac{\sum_{i=1}^{11} (1-p_i) \overline{x}_i}{\sum_{i=1}^{11} (1-p_i)}$$

$$\sigma_1^2 = \frac{\sum_{i=1}^{11} p_i \left(x_i - \mu_1\right)^2}{\sum_{j=1}^{11} p_i} \text{ and } \sigma_2^2 = \frac{\sum_{i=1}^{11} (1-p_i) \left(x_i - \mu_2\right)^2}{\sum_{j=1}^{n} (1-p_i)}$$

$$P(C_1) = \frac{\sum_{i=1}^{11} p_i}{11} \text{ and } P(C_2) = 1 - P(C_1)$$

- The process is repeated until the clustering results converge

Decision of the Algorithm

- Once we have figured out the parameter values, then we assign sample x_i into C_1, if

$$\frac{\text{Prob}[C_1 \mid x_i]}{\text{Prob}[C_1 \mid x_i] + \text{Prob}[C_2 \mid x_i]} > 0.5, \text{ where}$$

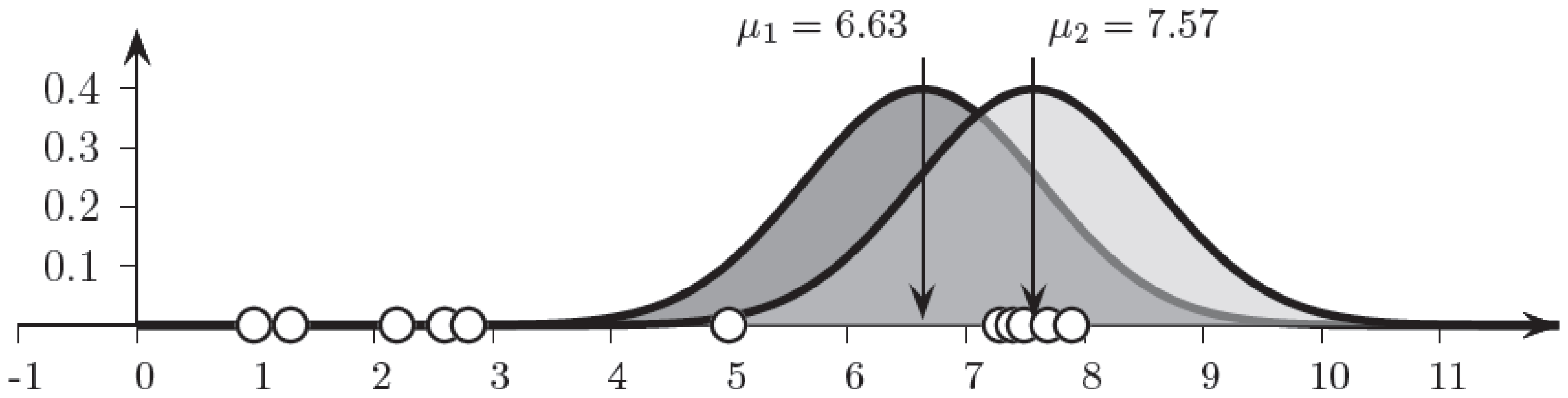
$$\text{Prob}[C_1 \mid x_i] = \frac{\text{Prob}[x_i \mid C_1]\text{Prob}[C_1]}{\text{Prob}[x_i]} = \frac{\text{Prob}[C_1]}{\text{Prob}[x_i]} \cdot \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x_i-\mu_1)^2}{2\sigma_1^2}}$$

$$\text{Prob}[C_2 \mid x_i] = \frac{\text{Prob}[x_i \mid C_2]\text{Prob}[C_2]}{\text{Prob}[x_i]} = \frac{\text{Prob}[C_2]}{\text{Prob}[x_i]} \cdot \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(x_i-\mu_2)^2}{2\sigma_2^2}}.$$

- Otherwise, x_i is assigned into C_2.

Example - Expectation-Maximization in 1D

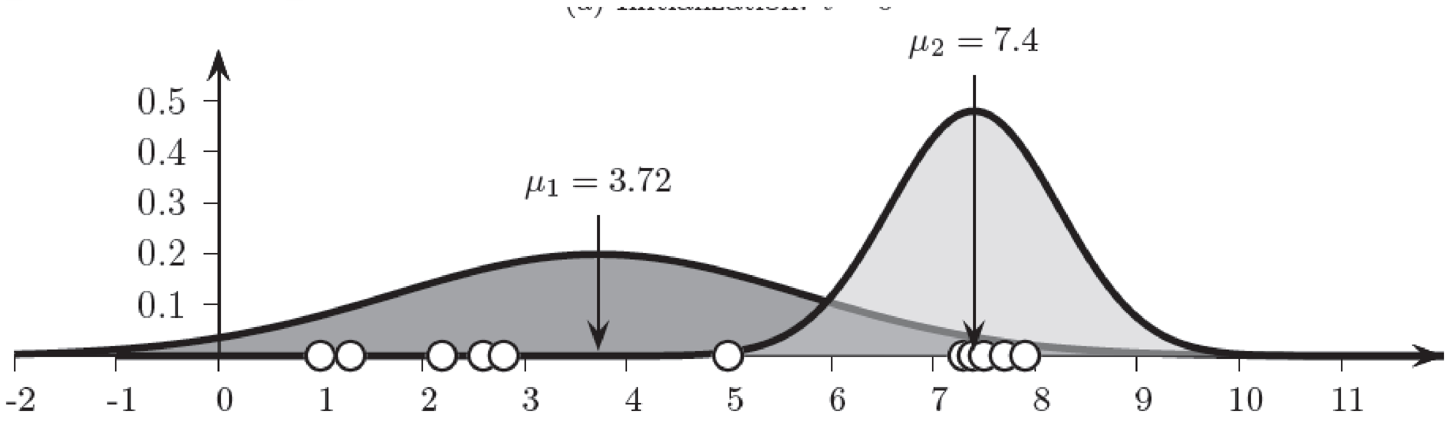
- Figure shows the application of the EM algorithm to our one-dimensional dataset:
  $D = \{x_1 = 1.0, x_2 = 1.3, x_3 = 2.2, x_4 = 2.6, x_5 = 2.8, x_6 = 5.0, x_7 = 7.3, x_8 = 7.4, x_9 = 7.5, x_{10} = 7.7, x_{11} = 7.9\}$.
- The initial random means are shown in Figure 1 with the initial parameters given as $\mu_1 = 6.63, \sigma_1 = 1, P(C_1) = 0.5, \mu_2 = 7.57, \sigma_2 = 1, P(C_2) = 0.5$



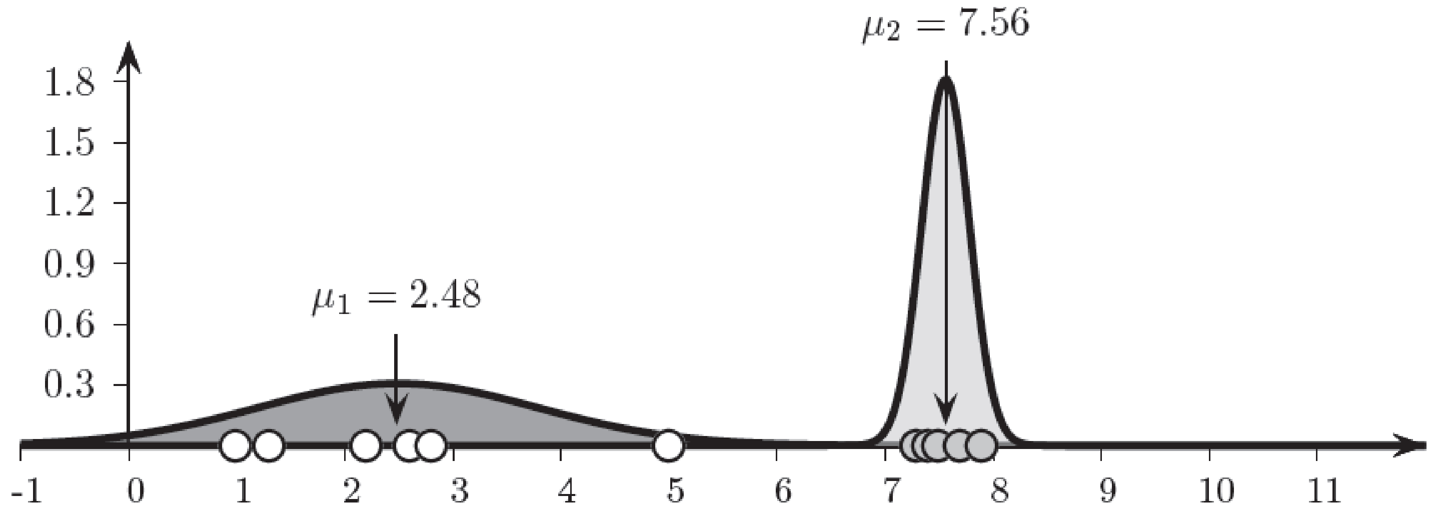
(a) Initialization: $t = 0$

Example (continued)

- After one iteration we have $\mu_1 = 3.72, \sigma_1 = 6.13, P(C_1) = 0.71, \mu_2 = 7.4, \sigma_2 = 0.69, P(C_2) = 0.29$



- After the final iteration $(t = 5)$: $\mu_1 = 2.48, \sigma_1 = 1.69, P(C_1) = 0.55, \mu_2 = 7.56, \sigma_2 = 0.05, P(C_2) = 0.45$



(c) Iteration: $t = 5$ (converged)

Lecture Overview

1. Divisive clustering
2. Clustering in Generative Model
3. EM Algorithm
4. EM in Mixture Model
5. EM in GMM
6. Example in 1-dim GMM
7. **EM for Nominal Data**

EM Algorithm for Nominal Attributes

- Suppose each data point has m nominal attributes and the number of clusters is K. In this case, the total number of parameters is equal to $(\mathrm{k}-1)+\sum_{i=1}^{m}|A_i|$ where $|A_i|$ is the size of the domain of attribute i.
- If two attributes are correlated, then we can merge these two attributes to form an attribute with the domain size $|A_i|\times|A_j|$.

EM for Nominal Attributes: Initialization

- Initialization step:
  - Make initial assignment of points to clusters (for example uniformly at random), set probability of all clusters to be equal
  - Let attribute X_i has domain A_i. For each a in A_i compute conditional probability $P(X_i = a|C_j)$ as in Naïve Bayes, i.e.
    1. Compute $\#(a, C_j) =$ number of times when $X_i = a$ for data points assigned to cluster C_j
    2. Compute size $|C_j|$ of the cluster C_j
    3. Compute $P(X_i = a \mid C_j) = \frac{\#(a, C_j)}{|C_j|}$

EM for Nominal Attributes: Expectation Step

- Expectation step: Compute the probability of a data point $\overline{x} = (a_1, \dots, a_m)$ given a cluster C_j as in Naïve Bayes i.e. $P\left(\overline{x} = (a_1, \dots, a_m) \middle| C_j\right) = \prod_{i=1}^{m} P(X_i = a_i \mid C_j)$
- Compute the absolute probability of a data point $\overline{x}$ as the sum $P(\overline{x}) = \sum_{i=1}^{k} P(\overline{x} | C_i) P(C_i)$. Using Bayes rule compute

$$P(C_i | \overline{x}) = \frac{\prod_{j=1}^{m} P\left(a_j \middle| C_i\right) P(C_i)}{P(\overline{x})} = \frac{\prod_{j=1}^{m} P\left(a_j \middle| C_i\right) P(C_i)}{\sum_{s=1}^{k} P(\overline{x} | C_s) P(C_s)}.$$

EM for Nominal Attributes: Maximization

Maximization Step.

Denote $p_{ji} = P(C_j|\overline{x}_i)$. Let ${X_{ki}}^a$ be an indicator variable

$$ {X_{ki}}^a = \begin{cases} 1 \text{ if } k^{th} \text{ attribute of } \overline{x}_i \text{ is equal } a \\ \quad\quad\quad 0 \text{ otherwise} \end{cases} . $$

Then

- Model parameters are updated as follows:

$$ P\left(X_k = a\middle|C_j\right) = \frac{\sum_{i=1}^{n} X_{ki}^{a} \cdot p_{ij}}{\sum_{i=1}^{n} p_{ij}} \text{ and } \mathrm{P}\left(C_j\right) = \frac{\sum_{i=1}^{n} p_{ij}}{n} $$

- Repeat expectation and maximization step until the algorithm converges

- Assign point $\overline{x}$ to the class $C = max_j \left\{ \frac{P(C_j|\overline{x})}{\sum_{j=1}^{k} P(C_j|\overline{x})} \right\}$

EM in R

```
library(mclust); library(sets)
data(iris)
modelName = "EEE“#when we know the model - supposed diagonal
    covariance and each component has equal volume
data = iris[,-5]
z <- matrix(data = NA, nrow = dim(iris)[1], ncol = 3)
#prepare matrix for initial random assignment of points to clusters
ranclass <- matrix(sample(list(c(0,0,1),c(0,1,0),c(1,0,0)),dim(iris)[1],
    replace=TRUE),dim(iris)[1],3)
#choose random vectors of assignment to classes by sampling of all
    possible assignments
for (i in 1:dim(iris)[1]){#convert the assignment lists to matrix format
for (j in 1:3) {z[i,j] <- (as.numeric(ranclass[[i]])[j])}}
msEst <- mstep(modelName, data, z)
#after random initialization run one maximization step to determine all
    initialization parameters
parameters = msEst$parameters
mix<-em(modelName, data, parameters)
```

EM in R - continued

```
mix$z #we see probabilities of clsses for each datapoint
mix$G #of clusters discovered
clus<-c(1:dim(iris)[1]) #compare cluster assignment with true Iris
    species
for(j in 1:dim(iris)[1])
{clus[j]<-which.max(mix$z[j,])}
table(clus,iris$Species)
#let's try version of em that computes optimal model and nummber of
    classes wrt to covariance
#we have no idea of what the model is so we apply EM that is
    initialized by hclust:
#we do hierarchical clustering, in it determines initial means and priors
    of each class
#and then apply EM with these initial priors and means. The defauult
    for number of clusters is to try 1:9
irisMclust <- Mclust(iris[,-5])
plot(irisMclust,what = "classification")
summary(irisMclust)
table(irisMclust$classification,iris$Species)
```

Reading

- Meira and Zaki, section 13.3
- TSKK pp 631-637