Unsupervised Learning and Clustering

Mingmin Chi

SCS Fudan University, Shanghai, China

Outline

Introduction

Previously, all our training samples were labeled: these samples were said “supervised"

Why unsupervised learning

- Collecting and Labeling a large set of sample patterns can be costly
- We can train with large amounts of (less expensive) unlabeled data, and only then use supervision to label the groupings found
- We can use unsupervised methods to identify features that will then be useful for categorization
- We gain some insight into the nature (or structure) of the data
- ...

Introduction (Cont's)

Assumption: the functional forms for the underlying probability densities are known and the only thing that must be learned is the value of an unknown parameter vector

We make the following assumptions:

- The samples come from a known number of classes, c

Introduction (Cont's)

Assumption: the functional forms for the underlying probability densities are known and the only thing that must be learned is the value of an unknown parameter vector

We make the following assumptions:

- The samples come from a known number of classes, c
- The prior probabilities $P(\omega_j)$ for each class are known ($j = 1, ..., c$)

Introduction (Cont's)

Assumption: the functional forms for the underlying probability densities are known and the only thing that must be learned is the value of an unknown parameter vector

We make the following assumptions:

- The samples come from a known number of classes, c
- The prior probabilities $P(\omega_j)$ for each class are known ($j = 1, ..., c$)
- The form for the class-conditional probability densities $p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j)$ are known

Introduction (Cont's)

Assumption: the functional forms for the underlying probability densities are known and the only thing that must be learned is the value of an unknown parameter vector

We make the following assumptions:

- The samples come from a known number of classes, c
- The prior probabilities $P(\omega_j)$ for each class are known ($j = 1, ..., c$)
- The form for the class-conditional probability densities $p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j)$ are known
- The values for the c parameter vectors $\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_c$ are unknown

Introduction (Cont's)

Assumption: the functional forms for the underlying probability densities are known and the only thing that must be learned is the value of an unknown parameter vector

We make the following assumptions:

- The samples come from a known number of classes, c
- The prior probabilities $P(\omega_j)$ for each class are known ($j = 1, ..., c$)
- The form for the class-conditional probability densities $p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j)$ are known
- The values for the c parameter vectors $\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_c$ are unknown
- The category labels are unknown

Introduction (Cont's)

Assumption: the functional forms for the underlying probability densities are known and the only thing that must be learned is the value of an unknown parameter vector

We make the following assumptions:

- The samples come from a known number of classes, c
- The prior probabilities $P(\omega_j)$ for each class are known ($j = 1, ..., c$)
- The form for the class-conditional probability densities $p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j)$ are known
- The values for the c parameter vectors $\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_c$ are unknown
- The category labels are unknown

Mixture of Densities

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{j=1}^{c} \underbrace{p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j)}_{\text{component densities}} \overbrace{P(\omega_j)}^{\text{mixing parameters}}$$

- This density function is called a mixture density

Mixture of Densities

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{j=1}^{c} \underbrace{p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j)}_{\text{component densities}} \overbrace{P(\omega_j)}^{\text{mixing parameters}}$$

- This density function is called a mixture density
- Our goal $\Rightarrow$ to use samples drawn from this mixture density to estimate the unknown parameter vector $\boldsymbol{\theta}$

Mixture of Densities

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{j=1}^{c} \underbrace{p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j)}_{\text{component densities}} \overbrace{P(\omega_j)}^{\text{mixing parameters}}$$

- This density function is called a mixture density
- Our goal $\Rightarrow$ to use samples drawn from this mixture density to estimate the unknown parameter vector $\boldsymbol{\theta}$
- Once $\boldsymbol{\theta}$ is known, we can decompose the mixture into its components and use a MAP classifier on the derived densities

Identifiability

Question: whether is it possible in principle to recover θ from the mixture or not?

Definition

- A density $P(\mathbf{x}|\theta)$ is said to be identifiable if $\theta \neq \theta'$ implies that there exists an $\mathbf{x}$ such that:

$$P(\mathbf{x}|\theta) \neq P(\mathbf{x}|\theta')$$

- In other words, a density $P(\mathbf{x}|\theta)$ is not identifiable if we cannot recover a unique θ, even from an infinite amount of data

Identifiability - Normal Densities

Mixtures of normal densities are usually not identifiable.

- The parameters in the simple mixture density

$$P(x|\theta) = \frac{P(\omega_1)}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}(x-\theta_1)^2\right] + \frac{P(\omega_2)}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}(x-\theta_2)^2\right]$$

  cannot be uniquely identified if

Identifiability - Normal Densities

Mixtures of normal densities are usually not identifiable.

- The parameters in the simple mixture density

$$P(x|\theta) = \frac{P(\omega_1)}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}(x-\theta_1)^2\right] + \frac{P(\omega_2)}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}(x-\theta_2)^2\right]$$

  cannot be uniquely identified if $P(\omega_1) = P(\omega_2)$

Identifiability - Normal Densities

Mixtures of normal densities are usually not identifiable.

- The parameters in the simple mixture density

$$P(x|\boldsymbol{\theta}) = \frac{P(\omega_1)}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}(x-\theta_1)^2\right] + \frac{P(\omega_2)}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}(x-\theta_2)^2\right]$$

  cannot be uniquely identified if $P(\omega_1) = P(\omega_2)$
- Since $\boldsymbol{\theta} = (\theta_1, \theta_2)$ and $\boldsymbol{\theta} = (\theta_2, \theta_1)$ are two possible vectors that can be interchanged without affecting $P(x|\boldsymbol{\theta})$
- Identifiability can be a problem, but we always assume that the densities we are dealing with are identifiable!

Ambiguous Queries

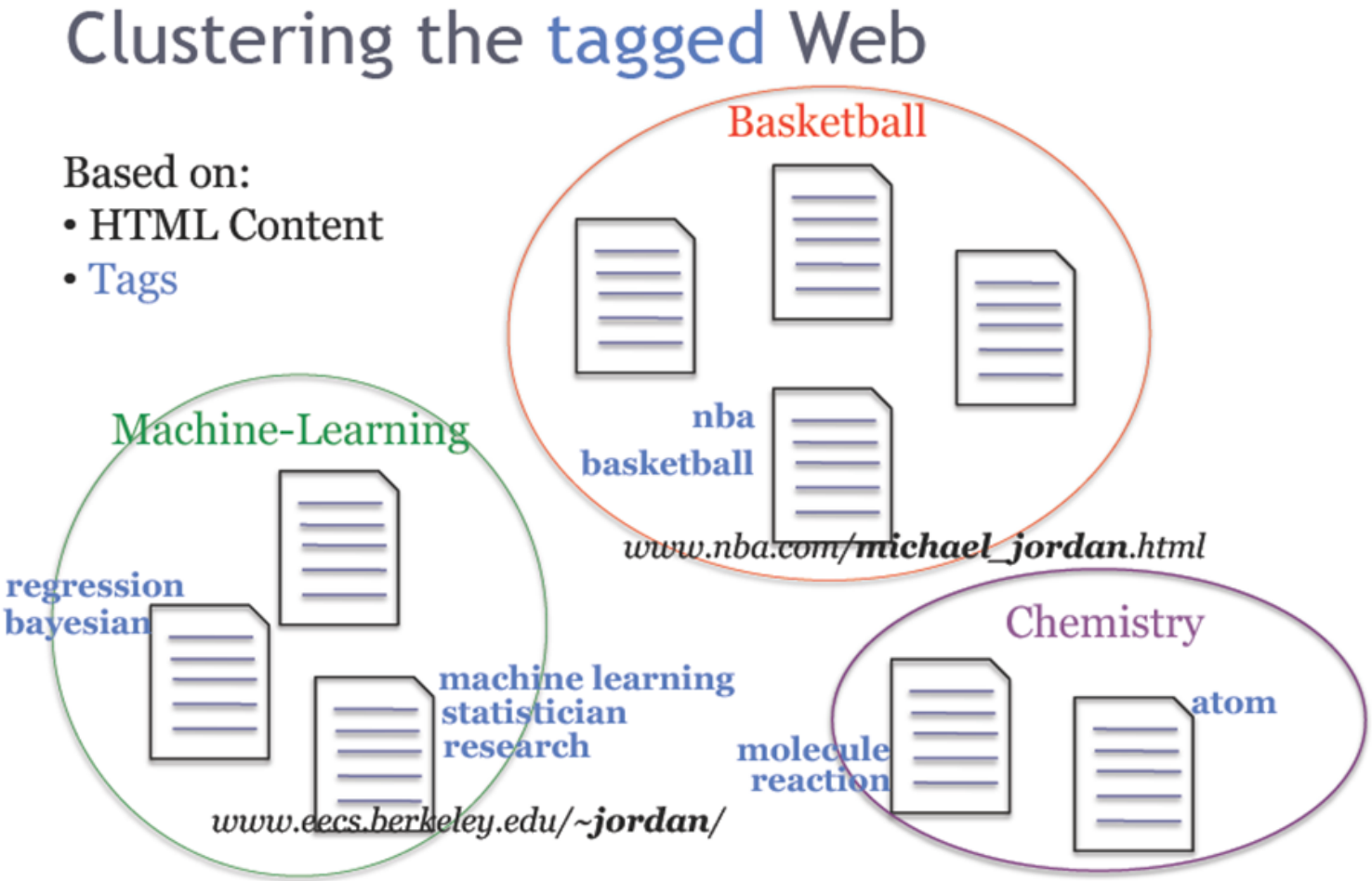
– "Barcelona" (City? Football team? Movie?)

– "Michael Jordan"

Michael *I.* Jordan

Michael *J.* Jordan

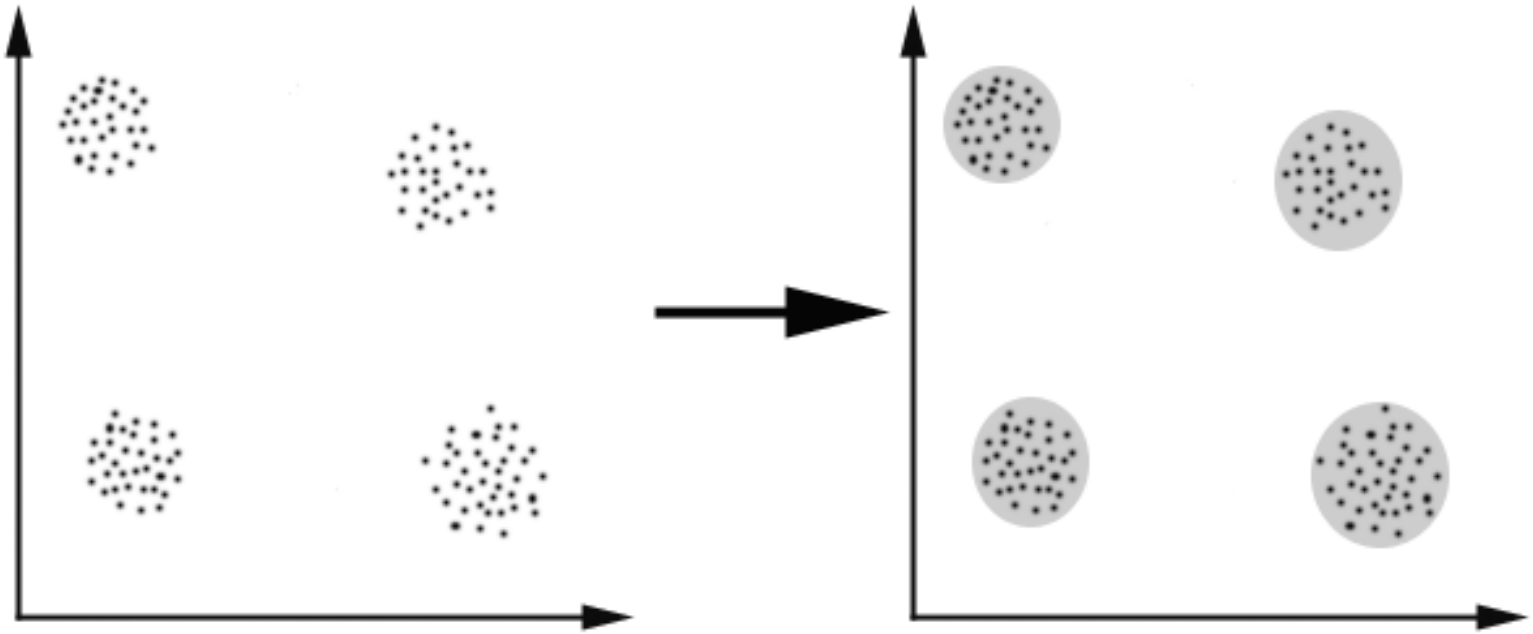
Ambiguous Queries



Overview

- What's clustering?
  - No universal definition
  - internal homogeneity and the extermal separation
  - A cluster is comprised of a number of similar objects collected or grouped together.
  - the process of organizing objects into groups whose members are similar in some way
- Clustering can be considered the most important unsupervised learning problem
- As every other problem of this kind, it deals with finding a structure in a collection of unlabeled data
- A cluster is therefore a collection of objects which are "similar" between them and are "dissimilar" to the objects belonging to other clusters

A Graphical Example



A Graphical Example

Goal of Clustering

The goal of clustering is to determine the intrinsic grouping in a set of unlabeled data

- how to decide what constitutes a good clustering?

Goal of Clustering

The goal of clustering is to determine the intrinsic grouping in a set of unlabeled data

- how to decide what constitutes a good clustering? no absolute “best" criterion which would be independent of the final aim of the clustering
- user must supply this criterion in such a way that the result of the clustering will suit one’s needs
- For instance, we could be interested
  1. in finding representatives for homogeneous groups (data reduction)
  2. in finding “natural clusters" and describe their unknown properties (“natural" data types)
  3. in finding useful and suitable groupings (“useful" data classes)
  4. in finding unusual data objects (outlier detection)

General Procedure

General Procedure

- Feature selection or extraction. (Elegant selection of features can greatly decrease the workload and simplify the subsequent design process)
- Clustering algorithm design or selection. (The proximity measure)
- Cluster validation. (external indices, internal indices, relative indices.)
- Results interpretation. (meaningful insights from the original data)

Clustering criterion

- To minimize the within-cluster coherence at each step
- To maximize the likelihood of observing documents

Possible Applications

Clustering algorithms can be applied in many fields, for instance:

1. Marketing: finding groups of customers with similar behavior given a large database of customer data containing their properties and past buying records
2. Biology: classification of plants and animals given their features
3. Libraries: book ordering
4. Insurance: identifying groups of motor insurance policy holders with a high average claim cost; identifying frauds
5. City-planning: identifying groups of houses according to their house type, value and geographical location
6. Earthquake studies: clustering observed earthquake epicenters to identify dangerous zones
7. WWW: document classification; clustering weblog data to discover groups of similar access patterns

Requirements

The main requirements that a clustering algorithm should satisfy are:

1. scalability
2. dealing with different types of attributes
3. discovering clusters with arbitrary shape
4. minimal requirements for domain knowledge to determine input parameters
5. ability to deal with noise and outliers
6. insensitivity to order of input records
7. high dimensionality
8. interpretability and usability

Problems

There are a number of problems with clustering. Among them:

1. current clustering techniques do not address all the requirements adequately (and concurrently)
2. dealing with large number of dimensions and large number of data items can be problematic because of time complexity
3. the effectiveness of the method depends on the definition of “distance" (for distance-based clustering)
4. if an obvious distance measure doesn’t exist we must “define" it, which is not always easy, especially in multi-dimensional spaces
5. the result of the clustering algorithm (that in many cases can be arbitrary itself) can be interpreted in different ways

Categories

Clustering algorithms may be classified as listed below:

1. Partitional Clustering
   - Exclusive Clustering, K-means
   - Overlapping Clustering, fuzzy K-means
   - Probabilistic Clustering, Mixture of Gaussians
2. Hierarchical Clustering
   - Agglomerative
   - Divisive

Essentials

A good clustering method

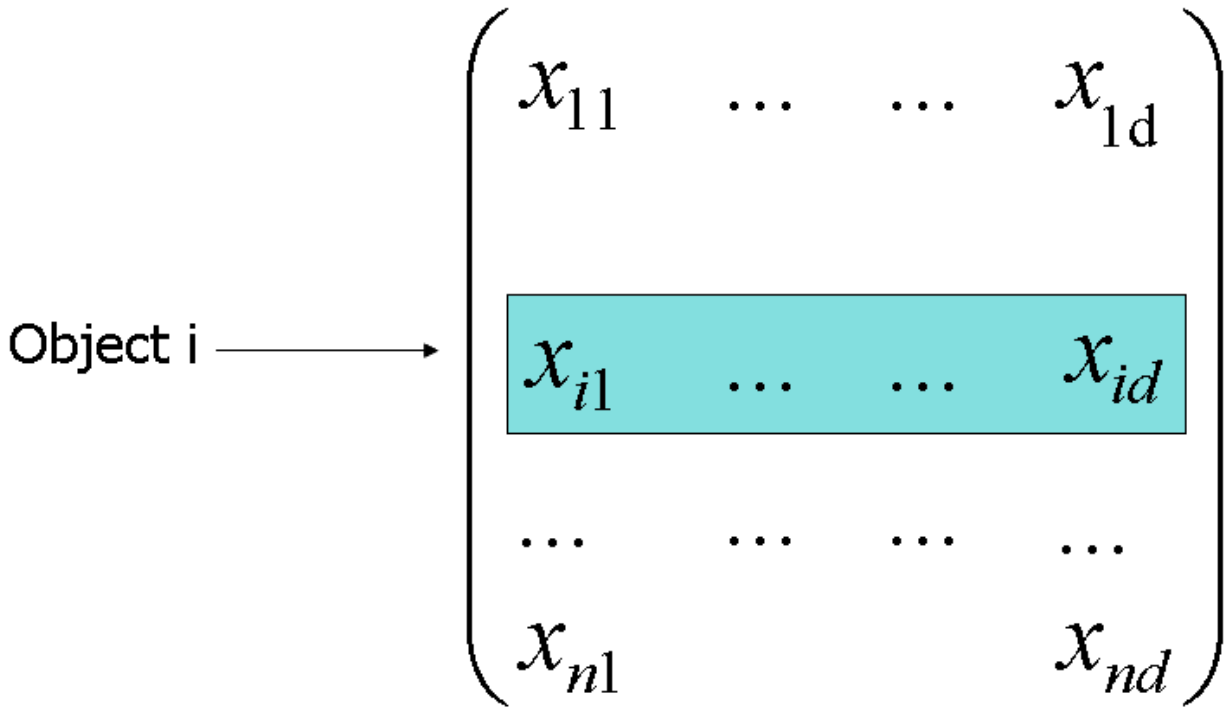
high on intra-class similarity and low on inter-class similarity

What is similarity?

Based on computation of distance

1. Between two numerical attributes
2. Between two nominal attributes
3. Mixed attributes

Database



Numerical Attributes

Distances are normally used to measure the similarity or dissimilarity between two data objects

Minkowski Metric

$$d_p(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_{k=1}^{d} |x_{ik} - x_{jk}|^p \right)^{\frac{1}{p}}$$

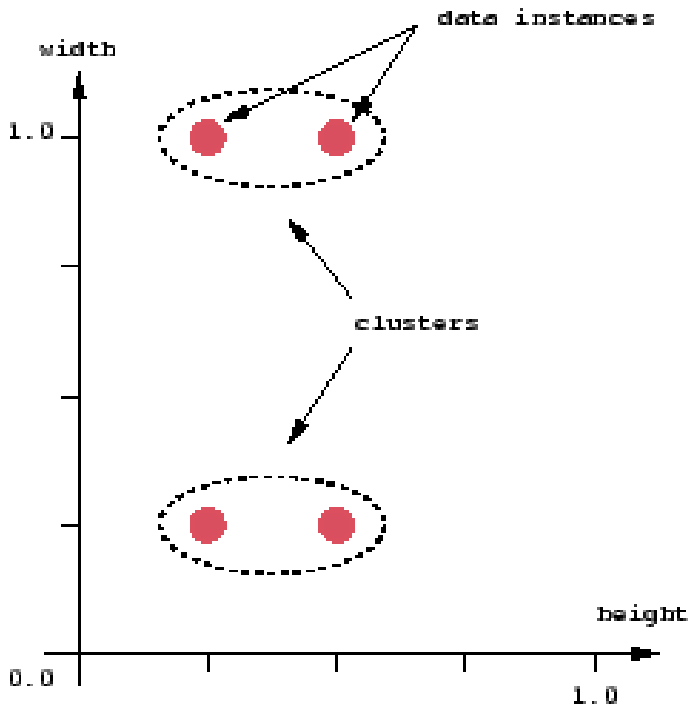
Euclidean metric

$$d_2(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\left(|x_{i1} - x_{j1}|^2 + \cdots + |x_{id} - x_{jd}|^2 \right)}$$

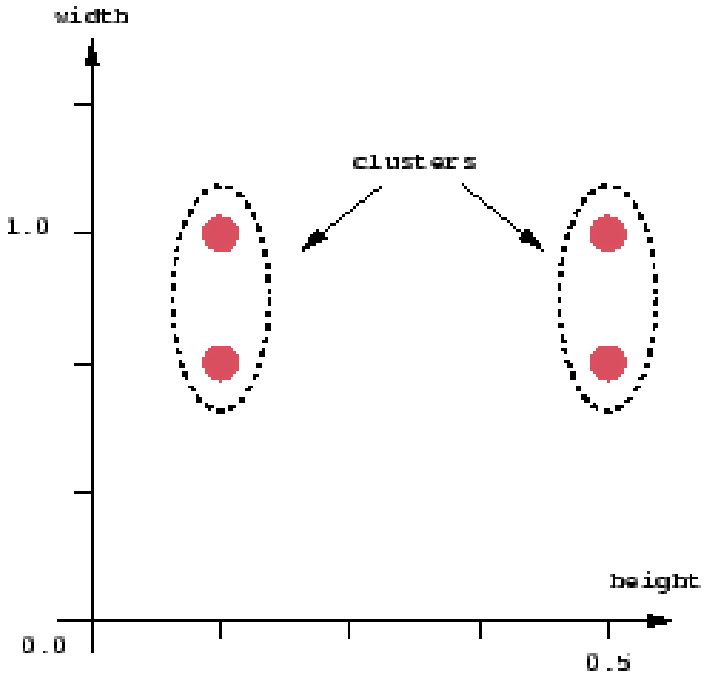
Manhattan metric

$$d_1(\mathbf{x}_i, \mathbf{x}_j) = |x_{i1} - x_{j1}| + \cdots + |x_{id} - x_{jd}|$$

Distance Measure



before scaling



after scaling

Other Measures of Cluster Distance

Minimum distance

$$d(C_i, C_j) = \min_{p \in C_i, p' \in C_j} |P - P'|$$

Max distance

$$d(C_i, C_j) = \max_{p \in C_i, p' \in C_j} |P - P'|$$

Mean distance

$$d(C_i, C_j) = \mu_i - \mu_j$$

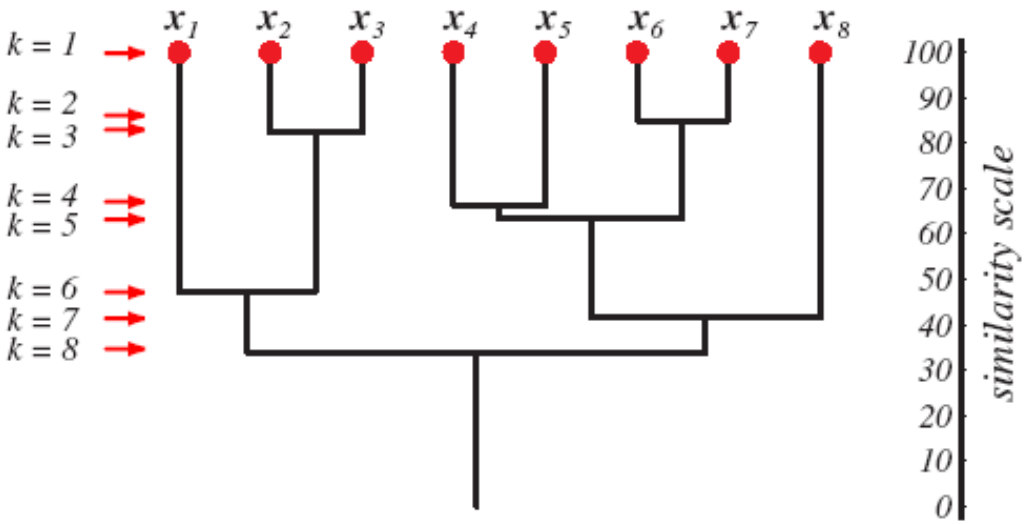
Average distance

$$d(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i} \sum_{p' \in C_j} |P - P'|$$

Categories

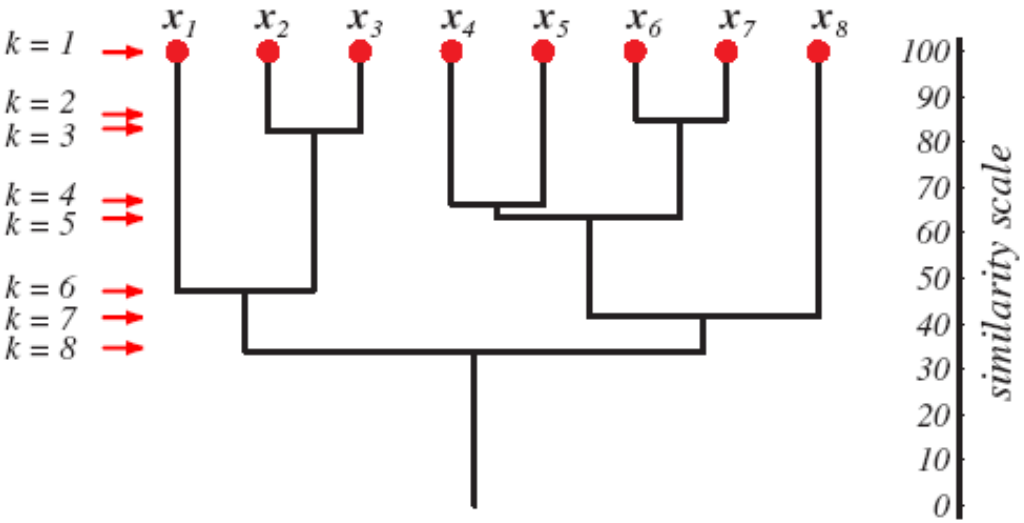
- Agglomerative (bottom-up, clumping): n singleton clusters and then form the sequence by successively merging clusters
- Divisive (top-down, splitting): one cluster and then form the sequence by successively splitting clusters

Agglomerative Hierarchical Clustering

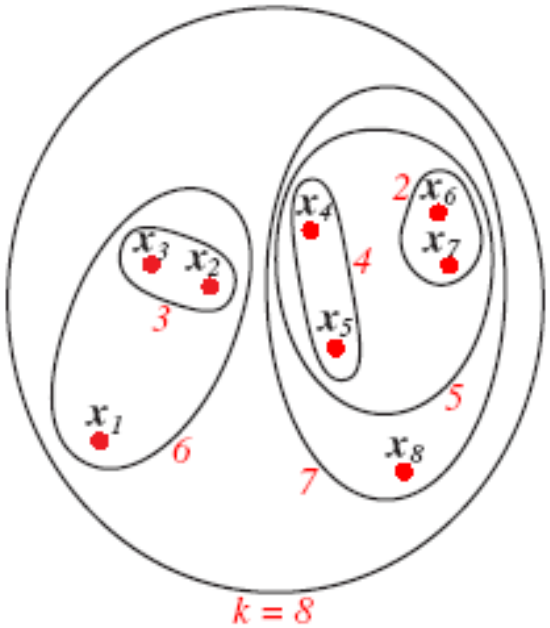


Dendrogram

Agglomerative Hierarchical Clustering



Dendrogram



Venn diagram representation

Revisit: Measures of Cluster Distance

Minimum distance - Agglomerative single linkage

$$d(C_i, C_j) = \min_{p \in C_i, p' \in C_j} |p - p'|$$

Max distance - Agglomerative complete linkage

$$d(C_i, C_j) = \max_{p \in C_i, p' \in C_j} |p - p'|$$

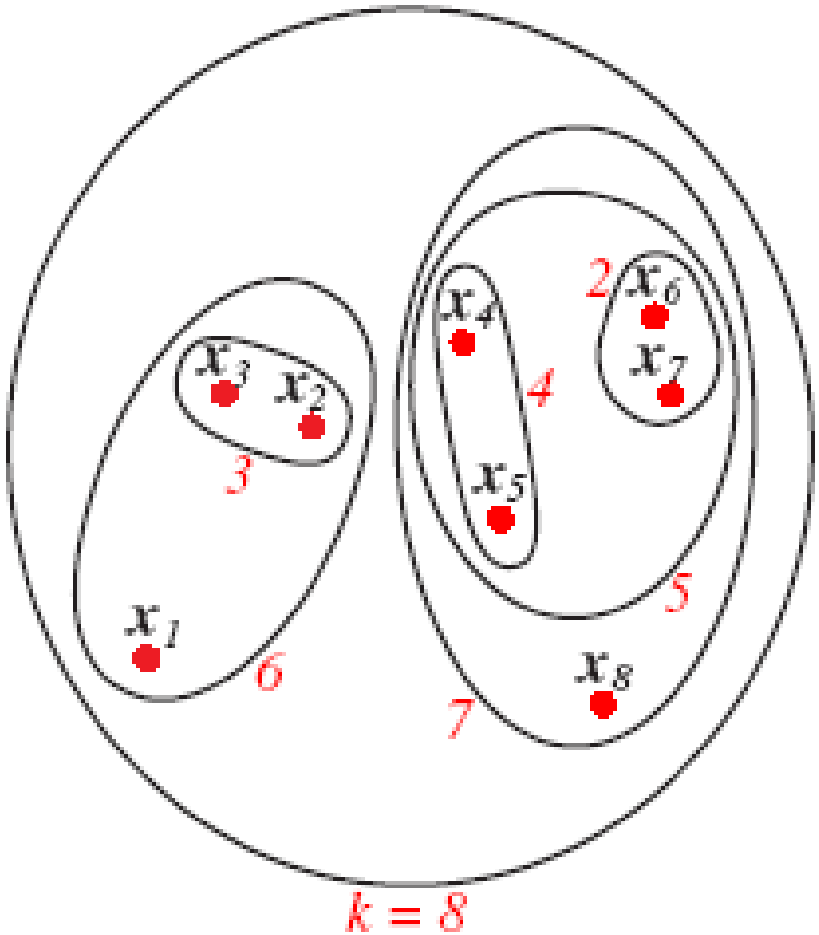
Mean distance - Agglomerative median linkage

$$d(C_i, C_j) = \boldsymbol{\mu}_i - \boldsymbol{\mu}_j$$

Average distance - Agglomerative centroid linkage

$$d(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i} \sum_{p' \in C_j} |p - p'|$$

Divisive Hierarchical Clustering



Objective

Suppose we have a set of data $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}, \mathbf{x}_i \in \mathbb{R}^d$

Our goal is to partition the dataset into some number K of clusters

- Each cluster is parameterized by a prototype vector $\boldsymbol{\mu}_k$
- Our object is to assign data points to clusters such that the sum of the squares of the distance of each data point to its closest cluster is a minimum

$$J = \min \sum_{i=1}^{n} \sum_{k=1}^{K} r_i^{(k)} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2$$

where assignment

$$r_i^{(k)} = \begin{cases} 1, & \text{if } \mathbf{x}_i \text{ is in the } k^{\text{th}} \text{ cluster} \\ 0, & \text{otherwise} \end{cases}$$

Optimization (1)

The k-means is to put n data points in a d-dimensional space into K clusters by minimizing

$$J = \sum_{i=1}^{n} \sum_{k=1}^{K} r_i^{(k)} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2$$

Optimization (1)

The k-means is to put n data points in a d-dimensional space into K clusters by minimizing

$$J = \sum_{i=1}^{n} \sum_{k=1}^{K} r_i^{(k)} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2$$

Our goal is to find $r_i^{(k)}, \boldsymbol{\mu}_k$ by minimizing J

Iterative procedure

- fixing $\boldsymbol{\mu}_k$, minimizing J w.r.t. $r_i^{(k)}$
- fixing $r_i^{(k)}$, minimizing J w.r.t. $\boldsymbol{\mu}_k$
- K-means is then an iterative two-step algorithm
- Repeat the two steps until the assignment does not change

Optimization (2)

- Updating $r_i^{(k)}$

$$r_i^{(k)} = \begin{cases} 1, & \text{if } k = \arg\min_k \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2 \\ 0, & \text{otherwise.} \end{cases}$$

- Updating $\boldsymbol{\mu}_k$ by setting the gradient of J wrt $\boldsymbol{\mu}_k$ to zero

$$2\sum_{i=1}^{n} r_i^{(k)} \|\mathbf{x}_i - \boldsymbol{\mu}_k\| = 0$$

$$\Rightarrow \boldsymbol{\mu}_k = \frac{\sum_i r_i^{(k)} \mathbf{x}_i}{\sum_i r_i^{(k)}}$$

Optimization (2)

- Updating $r_i^{(k)}$

$$r_i^{(k)} = \begin{cases} 1, & \text{if } k = \arg\min_k \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2 \\ 0, & \text{otherwise.} \end{cases}$$

- Updating $\boldsymbol{\mu}_k$ by setting the gradient of J wrt $\boldsymbol{\mu}_k$ to zero

$$2\sum_{i=1}^{n} r_i^{(k)} \|\mathbf{x}_i - \boldsymbol{\mu}_k\| = 0$$

$$\Rightarrow \boldsymbol{\mu}_k = \frac{\sum_i r_i^{(k)} \mathbf{x}_i}{\sum_i r_i^{(k)}}$$

$\boldsymbol{\mu}_k$ is the mean of the data points assigned to the cluster k. For the reason, the process is known as K-means algorithm.

The Algorithm: K-Means

Set K means $\{\mu_k\}$ to random values

Iterative procedure

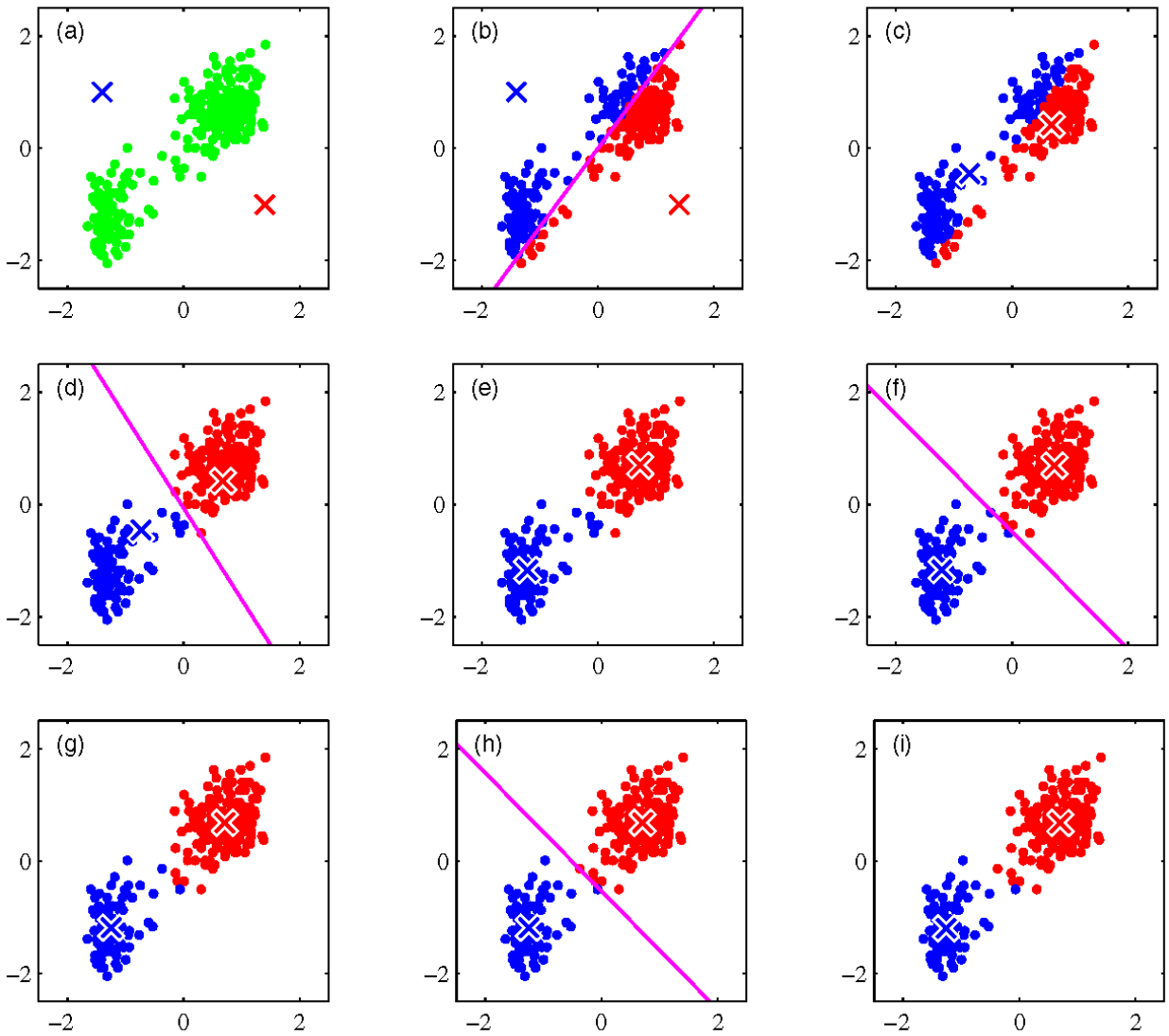
- E step: Updating $r_i^{(k)}$ by assigning each data point $\mathbf{x}_i$ to the nearest mean

$$r_i^{(k)} = \begin{cases} 1, & \text{if } k = \arg\min_k \|\mathbf{x}_i - \mu_k\|^2 \\ 0, & \text{otherwise.} \end{cases}$$

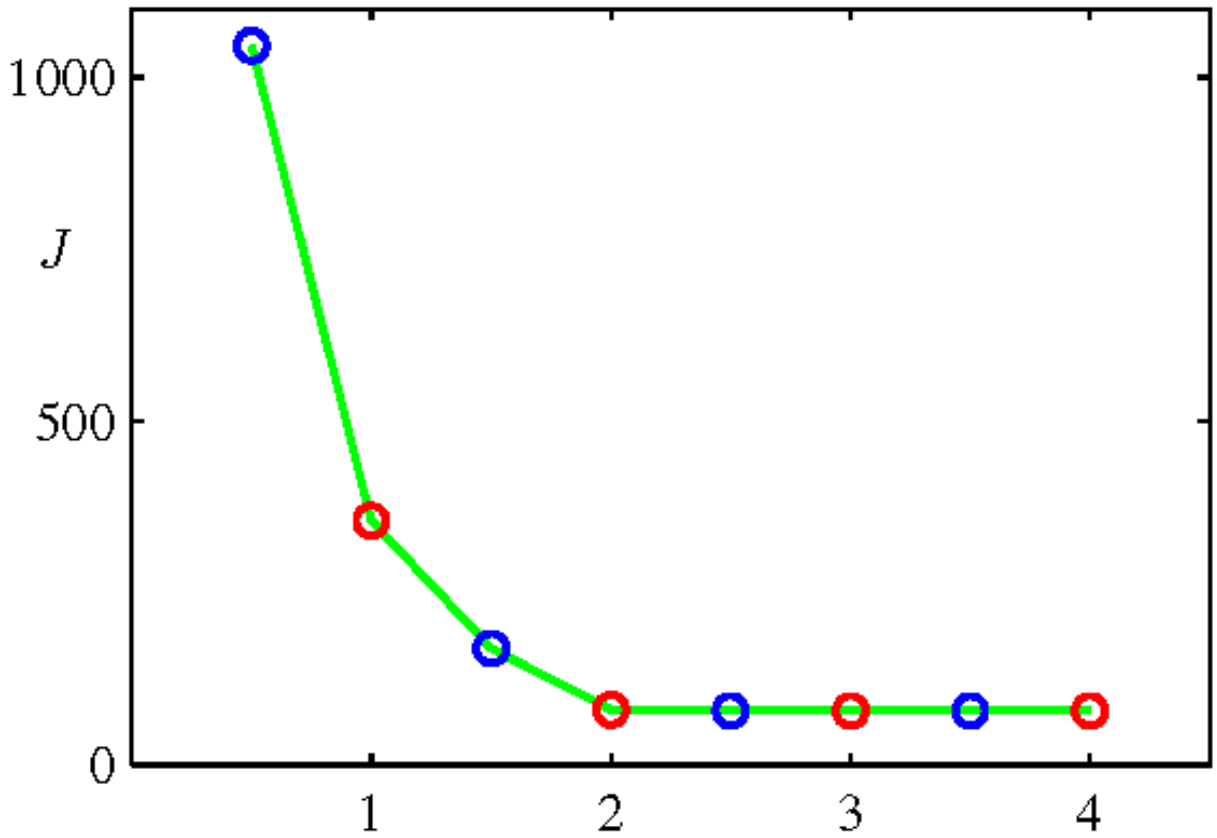
- M step: The model parameters, the means, are adjusted to match the sample means of the data points that they are responsible for:

$$\mu_k = \frac{\sum_i r_i^{(k)} \mathbf{x}_i}{\sum_i r_i^{(k)}}$$

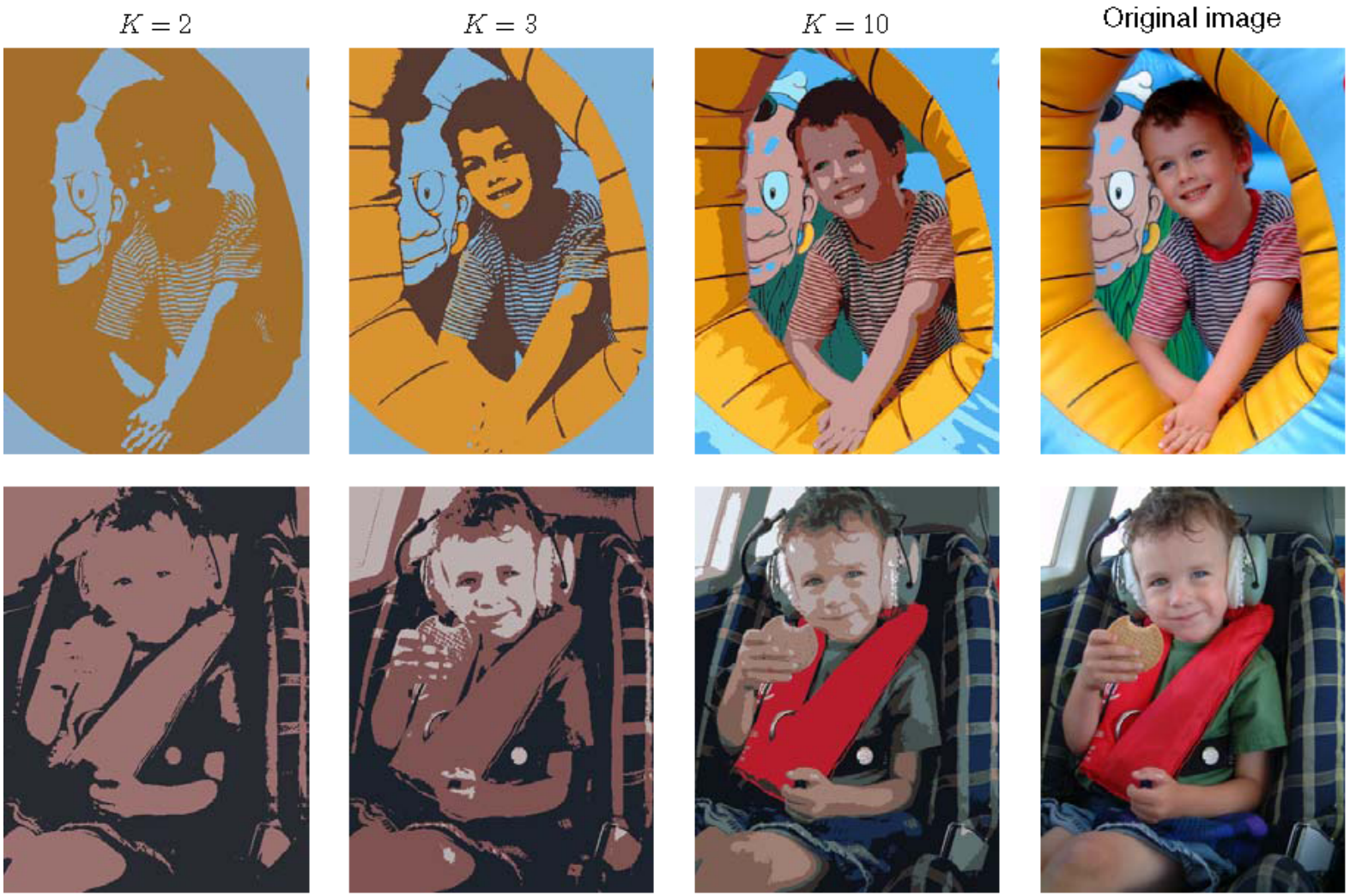
An Example



An Example: cost



Application: image segmentation



Application: Lossy Data Compression

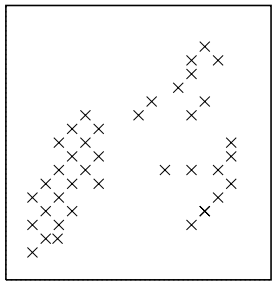
Vector quantization

- Store K cluster centers instead of original data points
- each data point is approximated by its nearest center
- number of clusters $K \ll N$ the number of original data

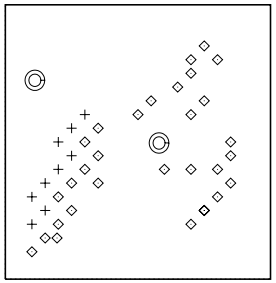
The center vectors are called code-book vectors

Example2

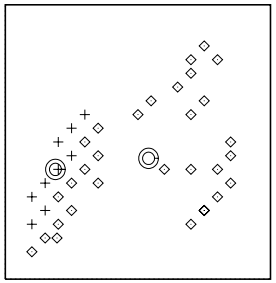
Data:



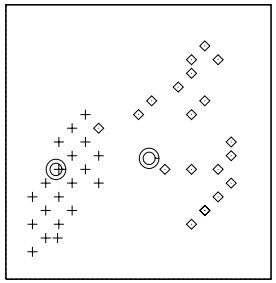
Assignment



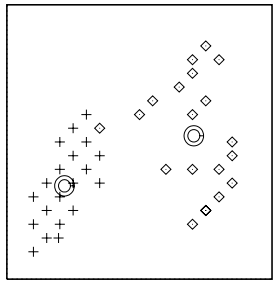
Update



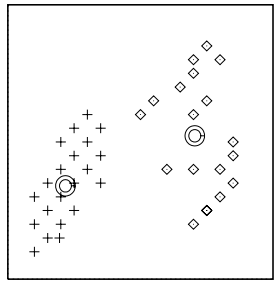
Assignment



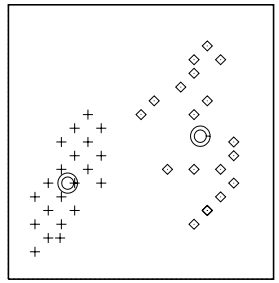
Update



Assignment

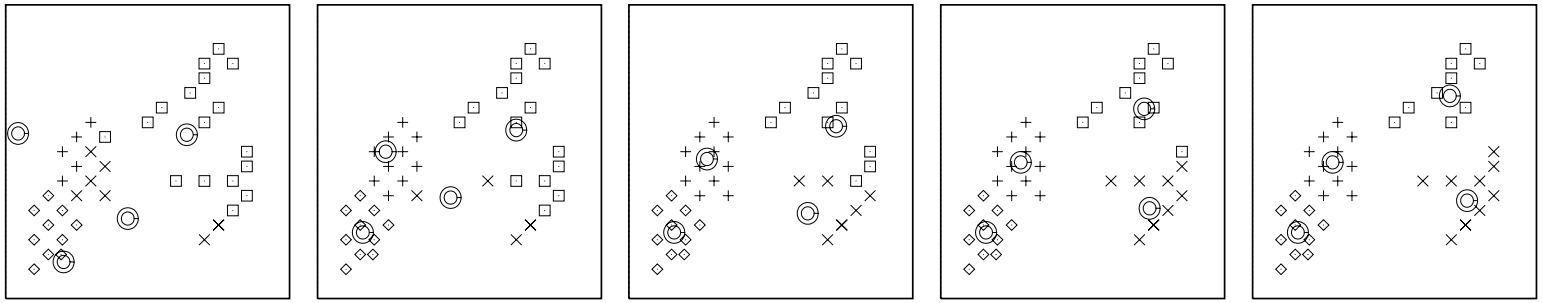


Update

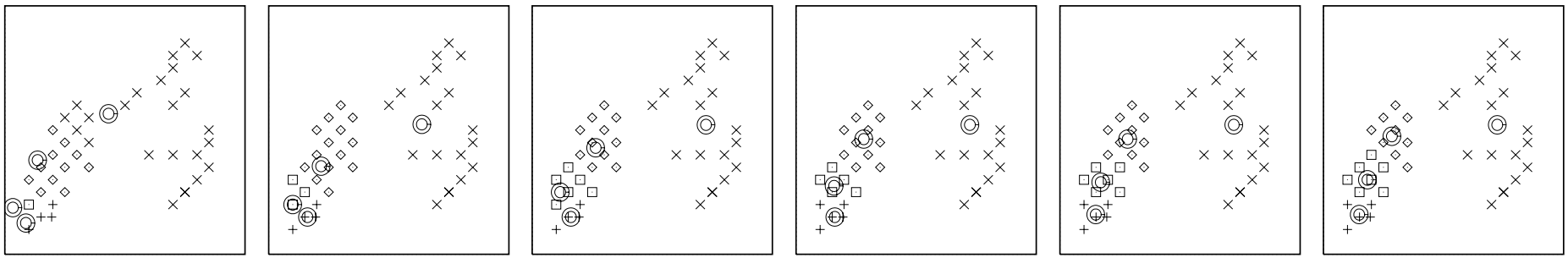


An Example: Different Initial Guess

Run 1

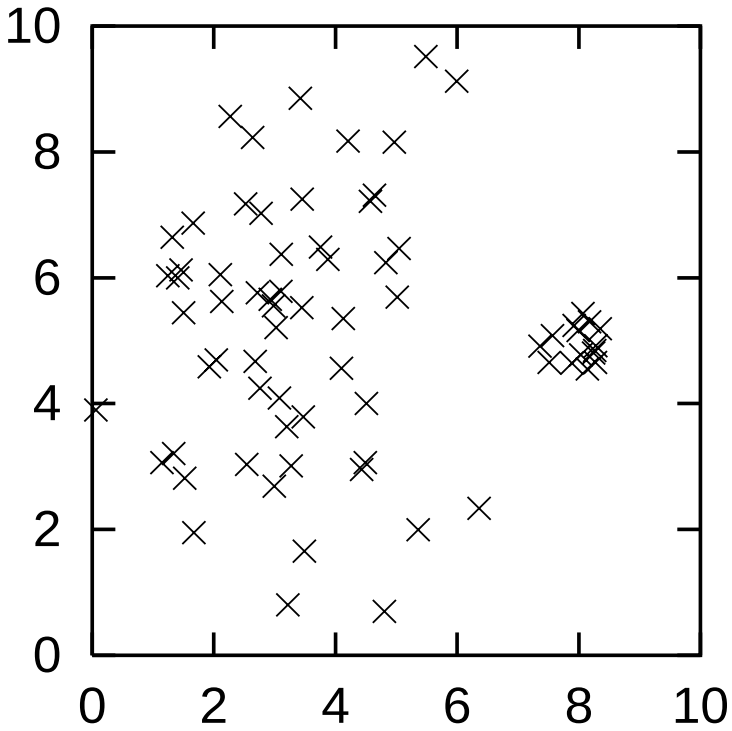


Run 2

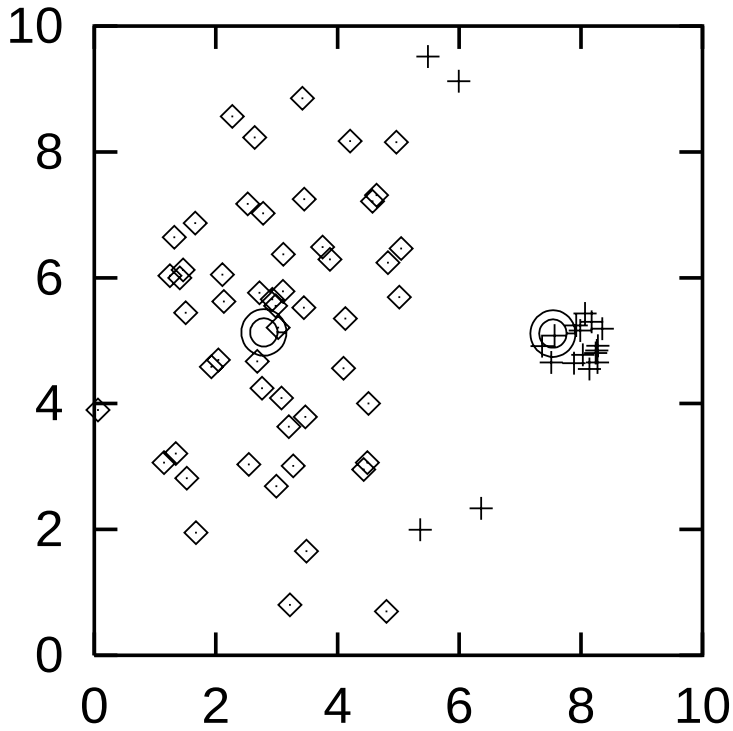


An Example: Two Gaussian

(a)

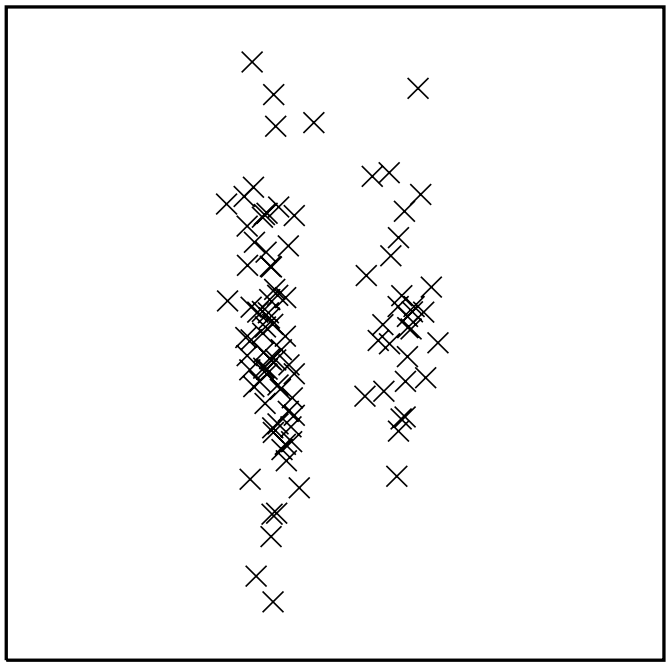


(b)

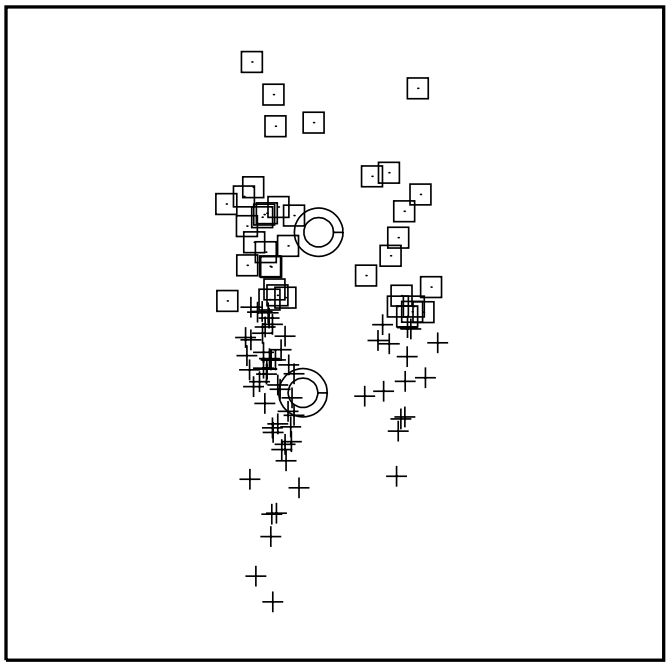


An Example: Bad Case

(a)



(b)



Problems

1. Two examples show that there is something wrong with the distance $d(\cdot, \cdot)$ in the K-means algorithm
   - The K-means algorithm is very sensitive to outliers

Problems

1. Two examples show that there is something wrong with the distance $d(\cdot,\cdot)$ in the K-means algorithm
   - The K-means algorithm is very sensitive to outliers (medoid in stead of centroid) so-called *K-medoid algorithm* with the general distance function:

$$\widetilde{J} = \min \sum_{i=1}^{n} \sum_{k=1}^{K} r_i^{(k)} \mathcal{V}(\mathbf{x}_i, \boldsymbol{\mu}_k)$$

   - A centroid is not necessarily an element of the k^{th} cluster
   - A medoid is defined similarly but with the added constraint that it must be a member of the kth cluster

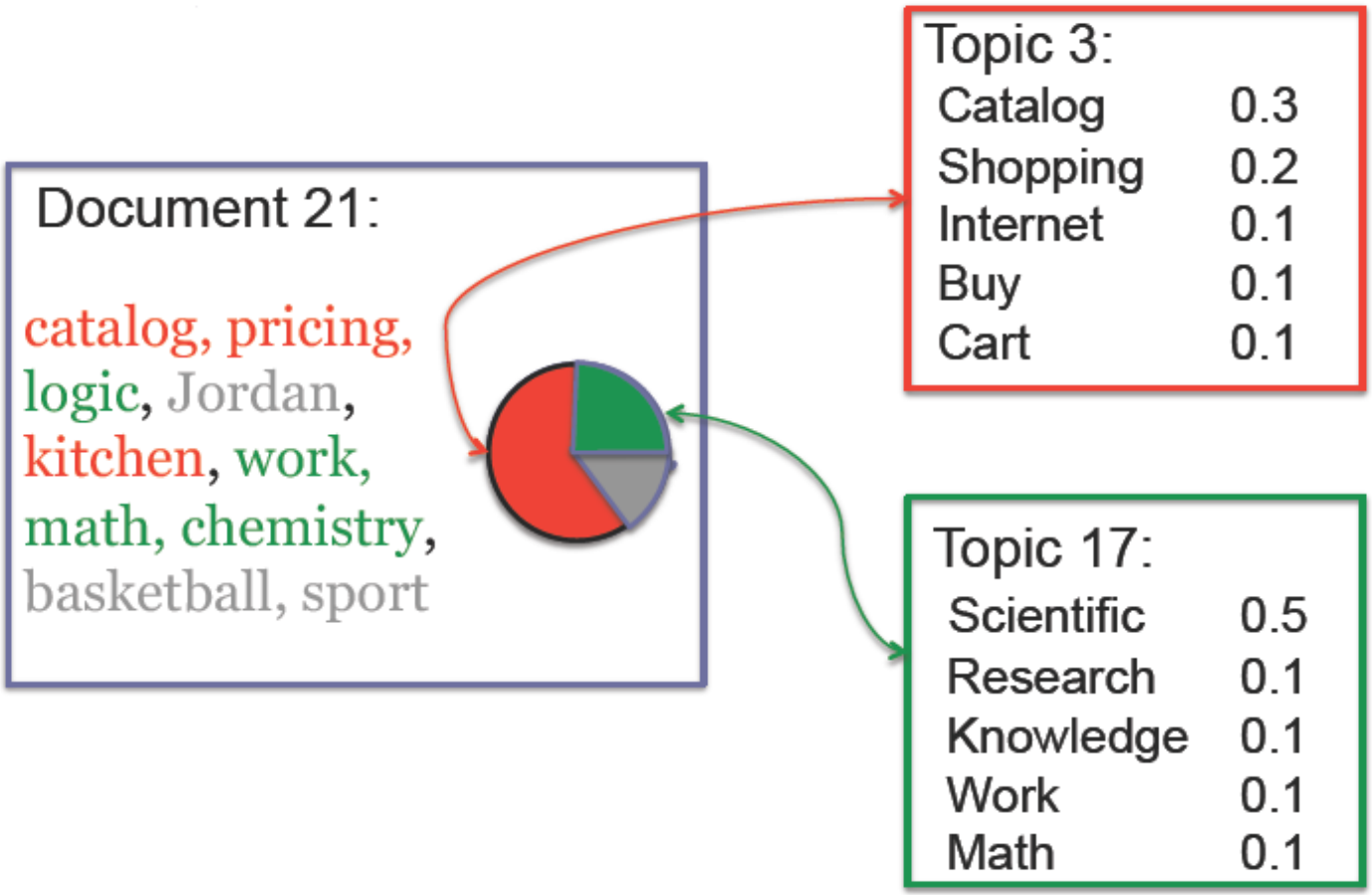
Problems

1. Two examples show that there is something wrong with the distance $d(\cdot,\cdot)$ in the K-means algorithm
   - The K-means algorithm is very sensitive to outliers (medoid in stead of centroid) so-called *K-medoid algorithm* with the general distance function:

$$\widetilde{J} = \min \sum_{i=1}^{n} \sum_{k=1}^{K} r_i^{(k)} \mathcal{V}(\mathbf{x}_i, \boldsymbol{\mu}_k)$$

   - A centroid is not necessarily an element of the k^{th} cluster
   - A medoid is defined similarly but with the added constraint that it must be a member of the kth cluster
2. A final criticism of k-means is that it is a 'hard' rather than a 'soft' algorithm

Hard K-means?


Document 21:
catalog, pricing,
logic, Jordan,
kitchen, work,
math, chemistry,
basketball, sport
Topic 3:
Catalog 0.3
Shopping 0.2
Internet 0.1
Buy 0.1
Cart 0.1
Topic 17:
Scientific 0.5
Research 0.1
Knowledge 0.1
Work 0.1
Math 0.1

Soft K-Means

The criticisms of k-means motivate the 'soft K-means' algorithm

- The algorithm has one parameter, β, which we could term the *stiffness*
- The stiffness β is an inverse-length-squared, so we can associate a lengthscale, $\sigma \equiv 1/\sqrt{\beta}$ with it

The Algorithm: Soft K-Means

Set K means $\{\boldsymbol{\mu}_k\}$ to random values

E step

- Each data point $\mathbf{x}_i$ is given a soft 'degree of assignment' to each of the means
- We call the degree to which $\mathbf{x}_i$ is assigned to cluster k the responsibility $r_i^{(k)}$ (the responsibility of cluster k for point $\mathbf{x}_i$:

$$r_i^{(k)} = \frac{\exp\left(-\beta d(\boldsymbol{\mu}_k, \mathbf{x}_i)\right)}{\sum_{k'} \exp\left(-\beta d(\boldsymbol{\mu}^{(k')}, \mathbf{x}_i)\right)}$$

The sum of the K responsibilities for the ith point is 1

The Algorithm: Soft K-Means

M step

The model parameters, the means, are adjusted to match the sample means of the data points that they are responsible for:

$$\boldsymbol{\mu}_k = \frac{\sum_i r_i^{(k)} \mathbf{x}_i}{R^{(k)}}$$

where $R^{(k)}$ is the total responsibility of mean k:

$$R^{(k)} = \sum_i r_i^{(k)}$$

Repeat the E step and M step

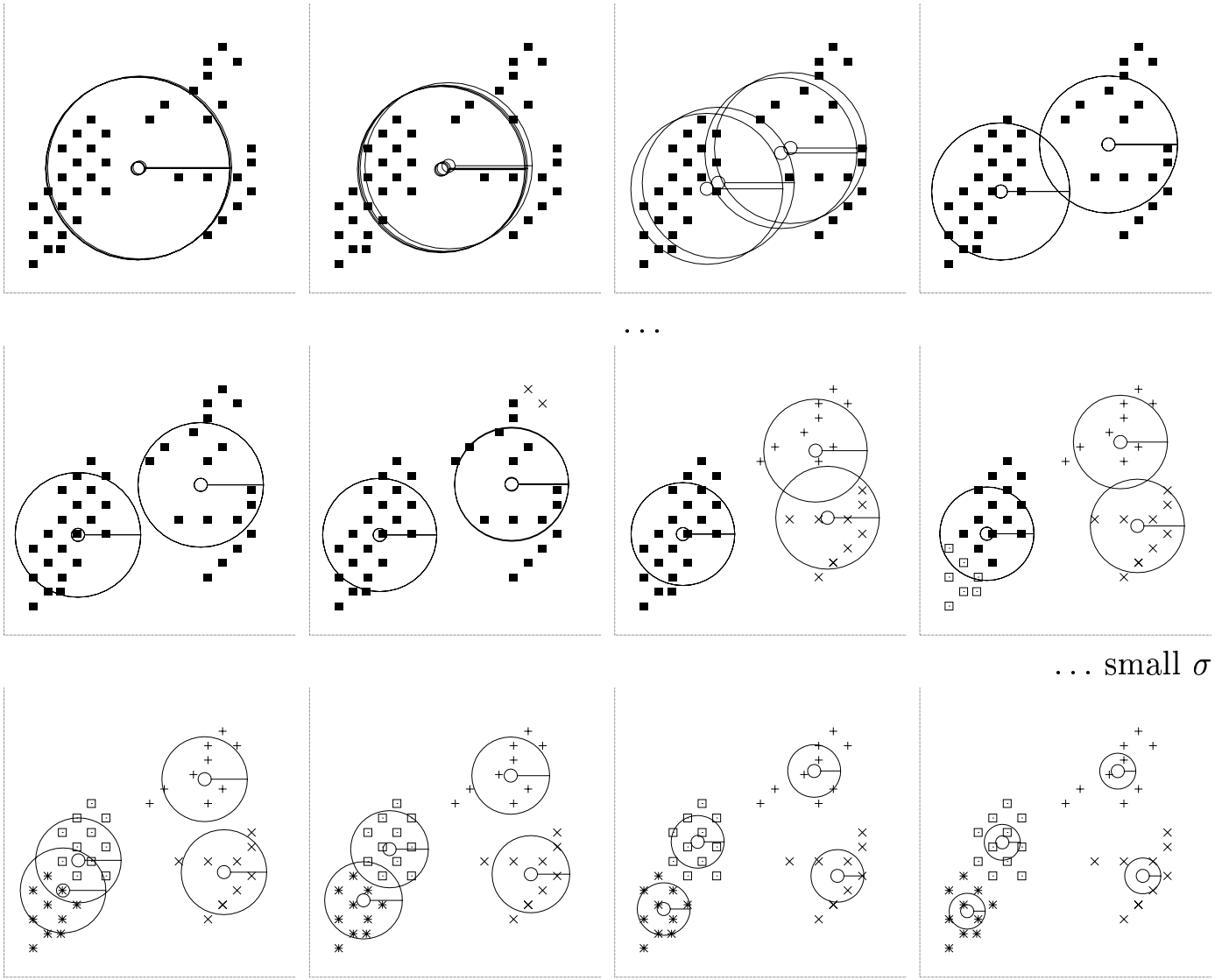
until the assignment does not change

Comparison between Hard and Soft K-Means

- The update step is identical
- the only difference is that:
  1. the responsibilities $r_i^{(k)}$ can take on values between 0 and 1
  2. the assignment in the hard k-means algorithm involved a 'min' over the distance
  3. the rule for assigning the responsibilities of the soft k-means algorithm is a 'soft-min'

An Example

Large σ ...



... small σ

Gaussian mixture density

A random variable $\mathbf{x}$ is assumed to have a probability distribution that is a mixture of K Gaussians

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k)$$

General Form

- So far, Gaussians as a common special case, the density of the data point $\mathbf{x}_i$ is

$$p(\mathbf{x}_i|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k p(\mathbf{x}_i|\boldsymbol{\theta}_k)$$

where $\boldsymbol{\theta} \equiv \{\{\boldsymbol{\mu}_k\}, \{\Sigma_k\}\}$

General Form

- So far, Gaussians as a common special case, the density of the data point $\mathbf{x}_i$ is

$$p(\mathbf{x}_i|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k p(\mathbf{x}_i|\boldsymbol{\theta}_k)$$

  where $\boldsymbol{\theta} \equiv \{\{\boldsymbol{\mu}_k\}, \{\Sigma_k\}\}$

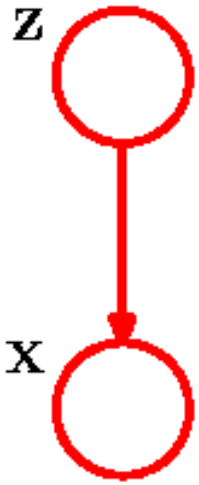
- A different way to think about mixture models is to consider them as latent variable models

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

  where $P(z_k = 1) = \pi_k$ is the prior for the latent variable taking on value $z_k = 1$, and $p(\mathbf{x}_i|z_k = 1, \boldsymbol{\theta}) = p(\mathbf{x}_i|\boldsymbol{\theta}_k)$ is the density under the component k

Latent variable model for MoG

- $\mathbf{z} \in \mathbb{R}^K, z_k \in \{0,1\}, \sum_k z_k = 1$
- $p(z_k = 1) = \pi_k$
- $\sum_k \pi_k = 1$
- $p(\mathbf{z}) = \prod_{k=1}^K \pi_k^{z_k}$
- $p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k)$



Joint and Marginal Probabilities

- $p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k)^{z_k}$
- joint distribution

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} (\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k))^{z_k}$$

- z_k is latent variable

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k)$$

Joint and Marginal Probabilities

- $p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k)^{z_k}$
- joint distribution

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} (\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k))^{z_k}$$

- z_k is latent variable

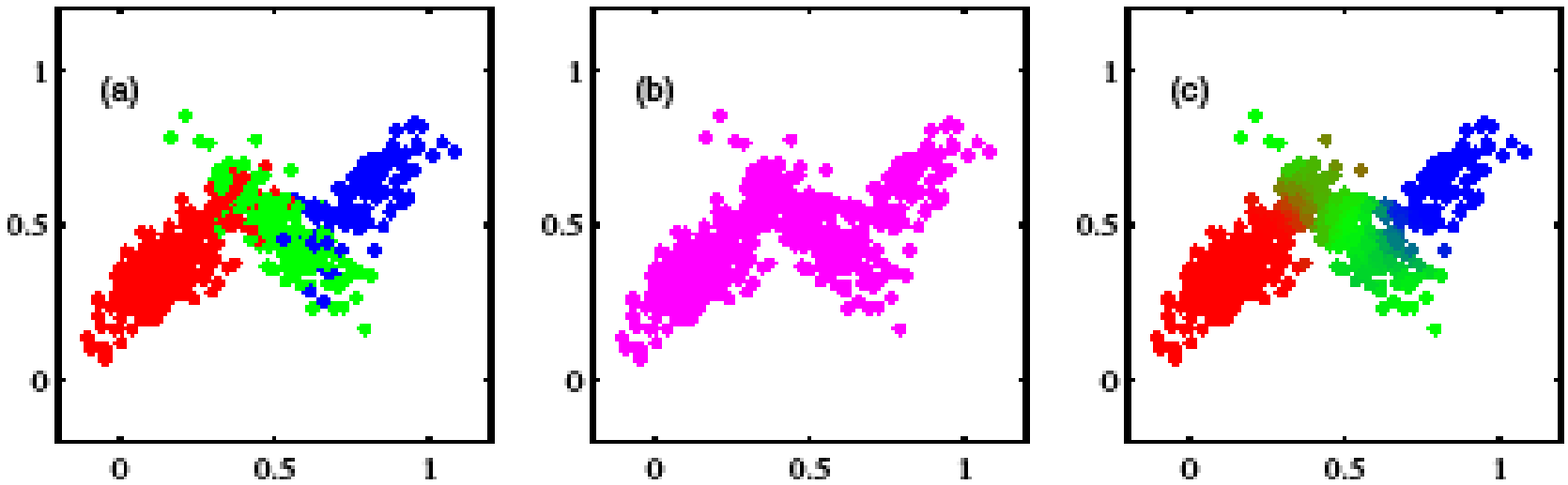
$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k)$$

An equivalent formation of the Gaussian mixture involving an explicit latent variable

Responsibility

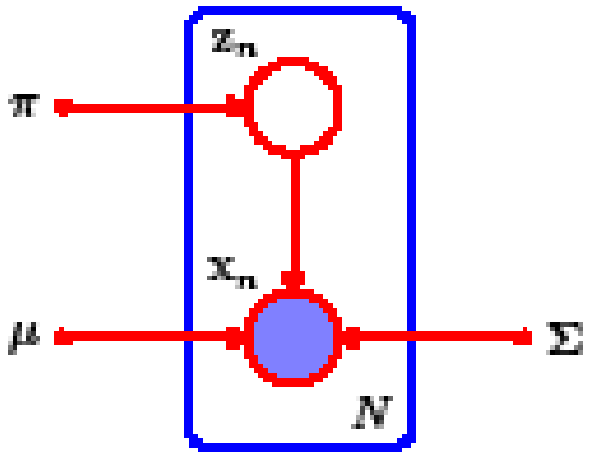
Responsibility (posteriori probability)

$$\gamma(z_k) \equiv p(z_k = 1|\mathbf{x}) = \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{k=1}^{K} p(z_k = 1)p(\mathbf{x}|z_k = 1)}$$
$$= \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \Sigma_j)}$$



Maximum Likelihood for MoG

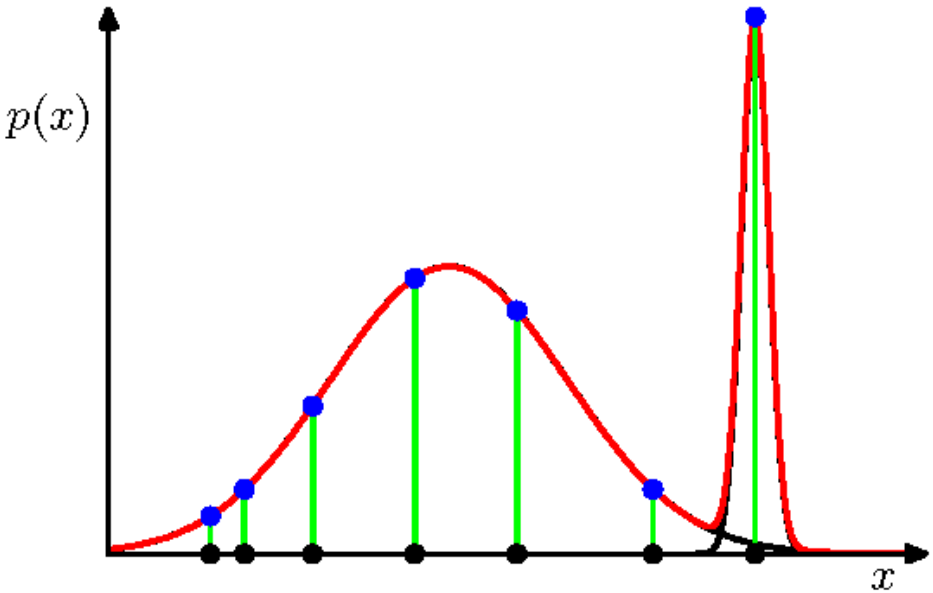
$$l(\mathbf{X}) = \ln p(\mathbf{X}|\pi, \boldsymbol{\mu}, \Sigma)$$
$$= \sum_{i=1}^{n} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \Sigma_k) \right\}$$



Singularity of ML

Problem

- Assume $\Sigma_k = \sigma_k^2 \mathbf{I}$
- $\boldsymbol{\mu}_k = \mathbf{x}_i$
- $\mathcal{N}(\mathbf{x}_i | \mathbf{x}_i, \sigma_k^2) = \frac{1}{(2\pi)^{1/2}} \frac{1}{\sigma_k}$
- if $\sigma_k \propto 0, l(\mathbf{X}) \propto \infty$



One of solutions is an iterative strategy, expectation-maximization (EM) algorithm

$r_i^{(k)}$

Responsibility (posteriori probability)

$$r_i^{(k)} = \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \Sigma_j)}$$

μ_k

Since

$$l(\mathbf{X}) = \ln p(\mathbf{X}|\pi, \mu, \Sigma) = \sum_{i=1}^{n} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_i|\mu_k, \Sigma_k) \right\}$$

$$\begin{aligned}
&\frac{\partial \ln p(\mathbf{X}|\pi, \mu, \Sigma)}{\partial \mu_k} = 0 \\
&= -\sum_{i=1}^{n} \frac{\pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}|\mu_j, \Sigma_j)} \Sigma_k^{-1}(\mathbf{x}_i - \mu_k) \\
&\Rightarrow \mu_k = \frac{1}{N_k} \sum_{i=1}^{n} r_i^{(k)} \mathbf{x}_i
\end{aligned}$$

where $N_k = \sum_{i=1}^{n} r_i^{(k)}$

Σ_k

Since

$$l(\mathbf{X}) = \ln p(\mathbf{X}|\pi, \boldsymbol{\mu}, \Sigma) = \sum_{i=1}^{n} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \Sigma_k) \right\}$$

$$\frac{\partial \ln p(\mathbf{X}|\pi, \boldsymbol{\mu}, \Sigma)}{\partial \Sigma_k} = 0$$

$$\Rightarrow \Sigma_k = \frac{1}{N_k} \sum_{i=1}^{n} r_i^{(k)} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^\top$$

π_k

Since $\sum_{k=1}^{K} \pi_k = 1$, we have

$$L(\pi_k) = \ln p(\mathbf{X}|\pi, \boldsymbol{\mu}, \Sigma) + \lambda(\sum_{k=1}^{K} \pi_k - 1)$$

$$\frac{\partial L(\pi_k)}{\partial \pi_k} = 0$$

$$\sum_{i=1}^{n} \frac{\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \Sigma_j)} + \lambda = 0$$

$$\Rightarrow \pi_k = \frac{N_k}{N}$$

Mixture of Gaussians: Enhancement to Soft K-Means

In the mixture of Gaussians:

- each cluster is a spherical Gaussian having its own width (each cluster has its own $\beta^{(k)} = 1/\sigma_k^2$)
- the algorithm updates the lengthscale σ_k for itself
- the algorithm also includes cluster weight parameters $w_1, \cdots, w_K$ which also update themselves

The Algorithm: MoG

Set K means $\{\boldsymbol{\mu}_k\}$ to random values

E step

The responsibilities are

$$r_i^{(k)} = \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \Sigma_j)}$$

$$= \frac{\pi_k \frac{1}{(2\pi\sigma_k^2)^{d/2}} \exp\left(-\frac{1}{2\sigma_k^2}\|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2\right)}{\sum_{k'} \pi_{k'} \frac{1}{(2\pi\sigma_{k'}^2)^{d/2}} \exp\left(-\frac{1}{2\sigma_{k'}^2}\|\mathbf{x}_i - \boldsymbol{\mu}_{k'}\|^2\right)}$$

The Algorithm: MoG

M step

Each cluster's parameters, μ_k, w_k and σ_k^2 are adjusted to match the data points that it is responsible for

$$\mu_k = \frac{\sum_i r_i^{(k)} \mathbf{x}_i}{R^{(k)}}$$

$$\sigma_k^2 = \frac{\sum_i r_i^{(k)} (\mathbf{x}_i - \mu_k)^2}{\mathrm{d} \times R^{(k)}}$$

$$\pi_k = \frac{R^{(k)}}{\sum_k R^{(k)}}, \quad \sum_{k=1}^{K} \pi_k = 1$$

where $R^{(k)}$ is the total responsibility of mean k:

$$R^{(k)} = \sum_i r_i^{(k)}$$

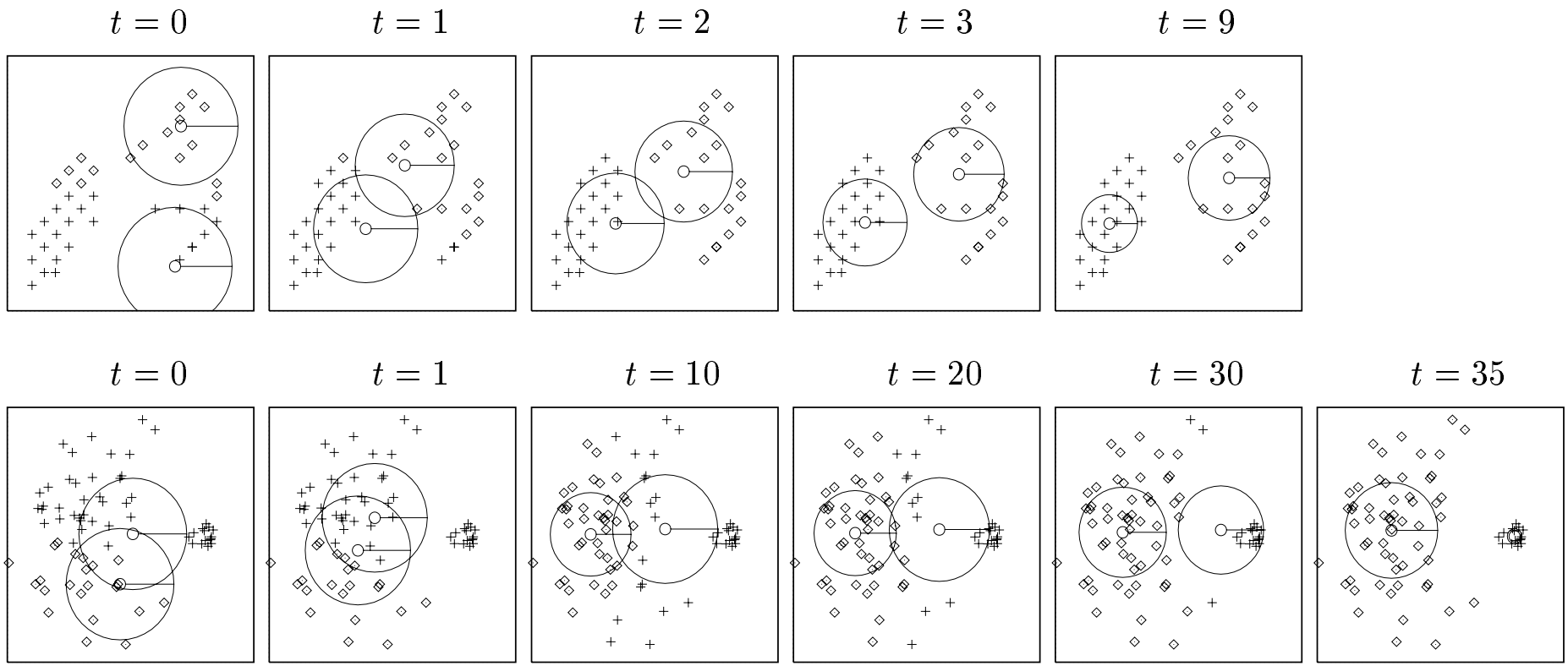
The Algorithm: Evaluation

$$l(\mathbf{X}) = \ln P(\mathbf{X}|\pi, \boldsymbol{\mu}, \Sigma) = \sum_{i=1}^{n} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \Sigma_k) \right\}$$

Check convergence

No, go to E-Step

An Example



Summary

In the version 1 of MoG

- The MoG, version 1, is a maximum-likelihood algorithm for fitting a mixture of spherical Gaussian to data

Summary

In the version 1 of MoG

- The MoG, version 1, is a maximum-likelihood algorithm for fitting a mixture of spherical Gaussian to data
- 'spherical' meaning that the variance of the Gaussian is the same in all directions

Summary

In the version 1 of MoG

- The MoG, version 1, is a maximum-likelihood algorithm for fitting a mixture of spherical Gaussian to data
- 'spherical' meaning that the variance of the Gaussian is the same in all directions
- the density of each data point in a mixture model:

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

where $\boldsymbol{\theta} \equiv \{\{\boldsymbol{\mu}_k\}, \{\sigma_k\}\}$

- Can we model cigar-shaped clusters by the MoG version 1?
- If no, how to solve it?

- Can we model cigar-shaped clusters by the MoG version 1?
- If no, how to solve it?
  - Can we model clusters by axis-aligned Gaussian with possibly-unequal variances?

- Can we model cigar-shaped clusters by the MoG version 1?
- If no, how to solve it?
  - Can we model clusters by axis-aligned Gaussian with possibly-unequal variances?
  - If yes, what is the density for the data point $\mathbf{x}_i$ in a mixture model?

- Can we model cigar-shaped clusters by the MoG version 1?
- If no, how to solve it?
  - Can we model clusters by axis-aligned Gaussian with possibly-unequal variances?
  - If yes, what is the density for the data point $\mathbf{x}_i$ in a mixture model?

$$p(x_{ij}|\theta) =$$

- Can we model cigar-shaped clusters by the MoG version 1?
- If no, how to solve it?
  - Can we model clusters by axis-aligned Gaussian with possibly-unequal variances?
  - If yes, what is the density for the data point $\mathbf{x}_i$ in a mixture model?

$$p(x_{ij}|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k p(x_{ij}|\boldsymbol{\theta}_j^{(k)})$$

where $\boldsymbol{\theta} \equiv \{\{\boldsymbol{\mu}_j^{(k)}\}, \{\sigma_j^{(k)}\}\}$

Responsibilities

we replace the assignment rule by:

$$r_i^{(k)} = \frac{\pi_k \frac{1}{(2\pi)^{d/2}\prod_{j=1}^{d}\sigma_{jk}} \exp\left(-\sum_{j=1}^{d}\frac{(\boldsymbol{\mu}_{jk}-\mathbf{x}_{ij})^2}{2\sigma_{jk}^2}\right)}{\sum_{k'}(\text{numerator, with } k' \text{ in place of } k)}$$

and the the variance update rule by:

$$\sigma_{jk}^2 = \frac{\sum_i r_i^{(k)}(\mathbf{x}_{ji}-\boldsymbol{\mu}_{jk})^2}{R_k}$$

The Algorithm: Initialization

Initialization

- Set K means $\{\boldsymbol{\mu}_k\}$ to random values or be initialized by K-means algorithm for fast convergence.
- Compute the number of the $k-$ th cluster N_k
- Compute variance by

$$\sigma_{jk} = \frac{1}{N_k} \sum_{k=1}^{K} (\mathbf{x}_{jk} - \boldsymbol{\mu}_{jk})^2$$

The Algorithm: E-Step

E step

The responsibilities (assignments) are

$$r_i^{(k)} = \frac{\pi_k \frac{1}{(2\pi)^{d/2}\prod_{j=1}^{d}\sigma_{jk}} \exp\left(-\sum_{j=1}^{d}\frac{(\mu_{jk}-\mathbf{x}_{ij})^2}{2\sigma_{jk}^2}\right)}{\sum_{k'}(\text{numerator, with } k' \text{ in place of } k)}$$

The Algorithm: M-Step

M step

Each cluster's parameters, $\boldsymbol{\mu}_k, \pi_k$ and σ_k^2 are adjusted to match the data points that it is responsible for

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_i r_i^{(k)} \mathbf{x}_i$$

$$\sigma_{jk}^2 = \frac{1}{N_k} \sum_i r_i^{(k)} (\mathbf{x}_{ji} - \boldsymbol{\mu}_{jk})^2$$

$$\pi_k = \frac{N_k}{\sum_k N_k}, \quad \sum_{k=1}^{K} \pi_k = 1$$

where R_k is the total responsibility of mean k:

$$N_k = \sum_i r_i^{(k)}, N = \sum_k N_k$$

The Algorithm: Evaluation

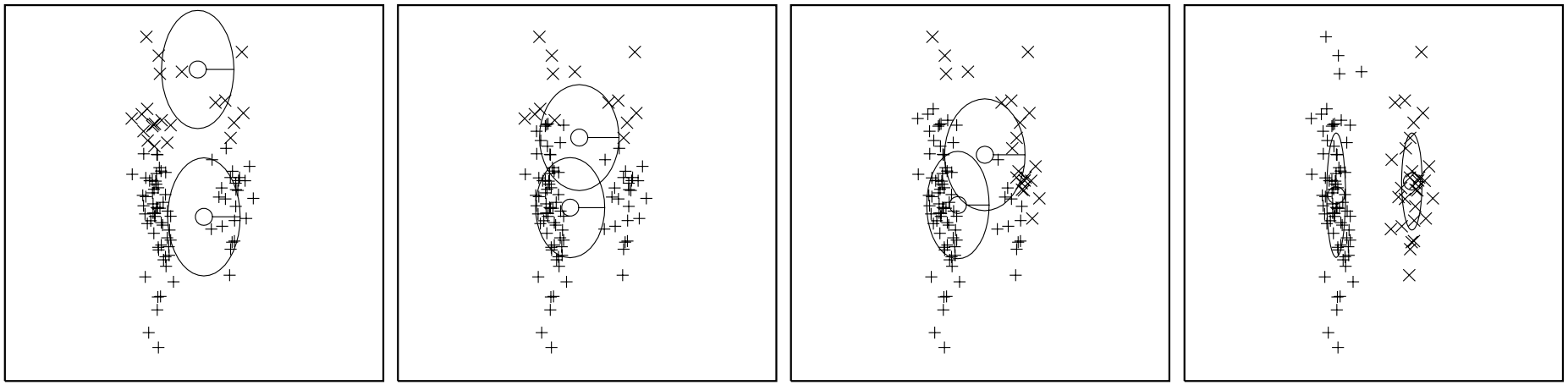
$$l(\mathbf{X}) = \ln P(\mathbf{X}|\pi, \boldsymbol{\mu}, \Sigma) = \sum_{i=1}^{n} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k) \right\}$$

Check convergence

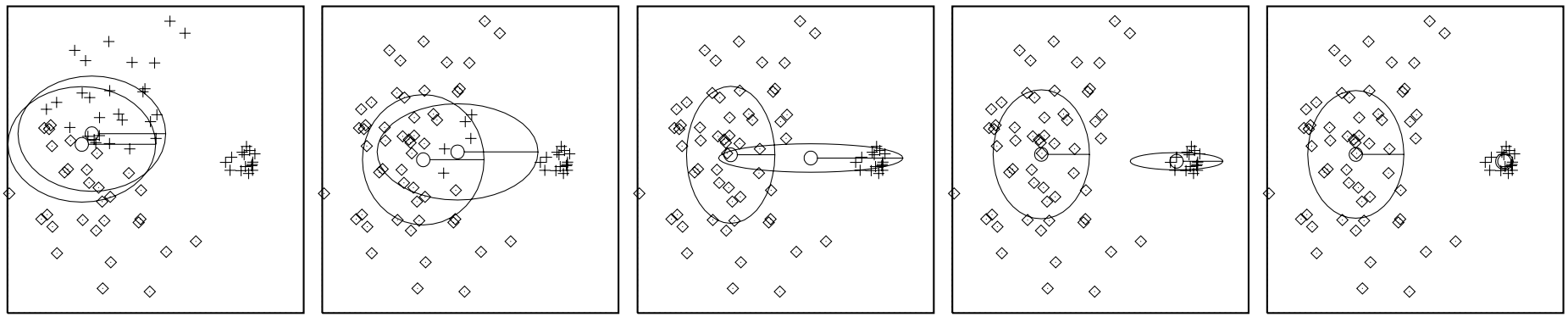
No, go to E-Step

An Example

$t = 0$ $t = 10$ $t = 20$ $t = 30$



$t = 0$ $t = 10$ $t = 20$ $t = 26$ $t = 32$



Density Function

- Non-spherical Gaussians:

$$p(x_{ij}|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k p(x_{ij}|\boldsymbol{\theta}_j^{(k)})$$

where $\boldsymbol{\theta} \equiv \{\{\boldsymbol{\mu}_j^{(k)}\}, \{\sigma_j^{(k)}\}\}$

- Spherical Gaussians:

$$p(\mathbf{x}_i|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k p(\mathbf{x}_i|\boldsymbol{\theta}_k)$$

where $\boldsymbol{\theta} \equiv \{\{\boldsymbol{\mu}_k\}, \{\sigma_k\}\}$

Density Function

- Spherical Gaussians:

$$p(\mathbf{x}_i|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k p(\mathbf{x}_i|\boldsymbol{\theta}_k)$$

where $\boldsymbol{\theta} \equiv \{\{\boldsymbol{\mu}_k\}, \{\sigma_k\}\}$

- Soft k-means:

$$p(\mathbf{x}_i|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k p(\mathbf{x}_i|\boldsymbol{\theta}_k)$$

where $\boldsymbol{\theta} \equiv \{\{\boldsymbol{\mu}_k\}, \sigma\}$

- Hard k-means:

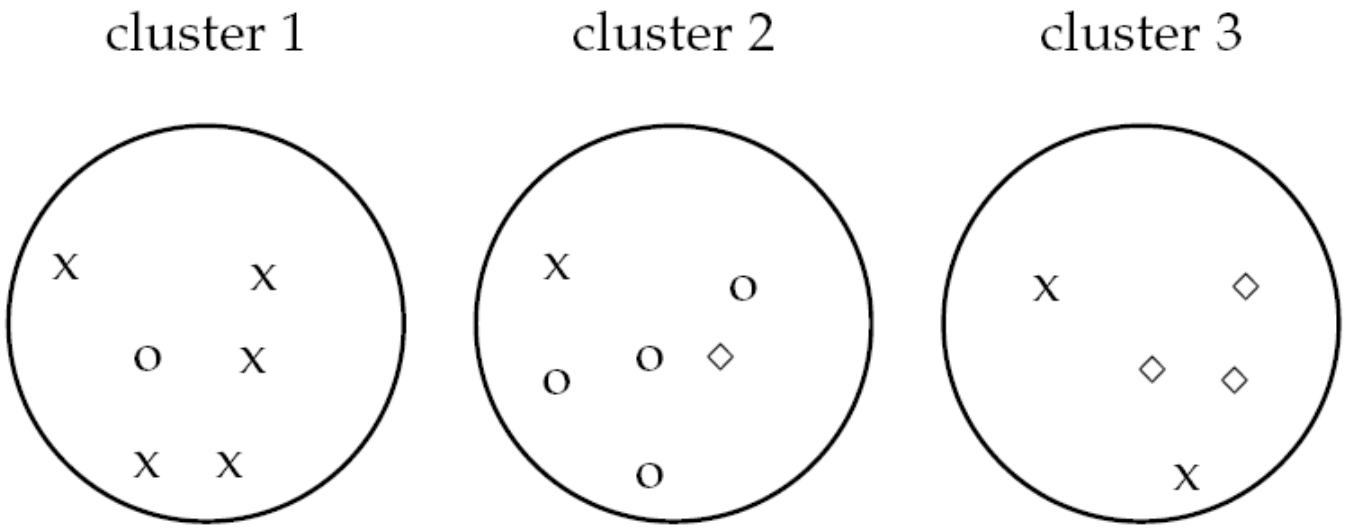
$$p(\mathbf{x}_i|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k p(\mathbf{x}_i|\boldsymbol{\theta}_k)$$

where $\boldsymbol{\theta} \equiv \{\{\boldsymbol{\mu}_k\}, \sigma\}$, $\pi_k = 1$

Criterions for the Quality of a Clustering

- internal criterion: attaining high intracluster similarity and low intercluster similarity
- external criterions:
  - Purity
  - Normalized mutual information (NMI)
  - Rand index
  - F measure

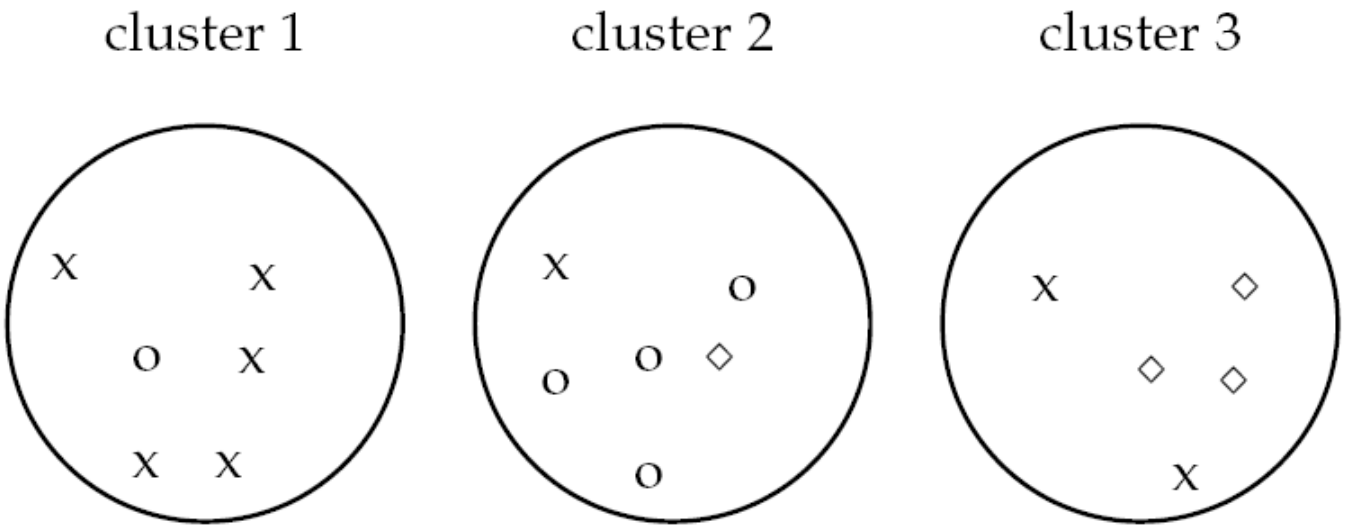
Purity



- the set of classes: $\mathcal{C} = \{c_1, \cdots, c_J\}$
- the set of clusters: $\Omega = \{\omega_1, \cdots, \omega_K\}$

$$\text{purity}(\Omega, \mathcal{C}) = \frac{1}{N} \sum_k \max_j |\omega_k \cap c_j|$$

Purity



- the set of classes: $\mathcal{C} = \{c_1, \cdots, c_J\}$
- the set of clusters: $\Omega = \{\omega_1, \cdots, \omega_K\}$

$$\text{purity}(\Omega, \mathcal{C}) = \frac{1}{N} \sum_k \max_j |\omega_k \cap c_j|$$

drawback?

NMI - Mutual Information

- measuring the dependence of two discrete random variables

$$I(x,y) = \mathrm{KL}(p(x,y)||p(x)p(y)) = \sum_x \sum_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

NMI - Mutual Information

- measuring the dependence of two discrete random variables

$$I(x,y) = \mathrm{KL}(p(x,y)||p(x)p(y)) = \sum_x \sum_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

$$I(\Omega, \mathcal{C}) = \sum_k \sum_j P(\omega_k \cap c_j) \log \frac{P(\omega_k \cap c_j)}{P(\omega_k)P(c_j)}$$

$$= \sum_k \sum_j \frac{|\omega_k \cap c_j|}{N} \frac{N|\omega_k \cap c_j|}{|\omega_k||c_j|}$$

NMI - Mutual Information (con't)

- Here, MI measures the amount of information by which our knowledge about the classes increases when we are told the clusters are
- The minimum of MI is 0, if the clustering is random with respect to class membership
- The maximum of MI is reached for a cluster Ω_{exact} that perfectly recreates the classes - but also if clusters in Ω_{exact} are further subdivided into smaller clusters
- what happens if $K = N$?

NMI - Entropy

- measuring the information content
- depending on the probability distribution $p(x)$
- looking for a quantity $h(x)$ that is a monotonic function of the probability:

$$h(x) = -\log p(x)$$

Entropy is the average amount of information defined as

$$H(x) = -\sum_x p(x) \log p(x)$$

NMI - Entropy

- measuring the information content
- depending on the probability distribution $p(x)$
- looking for a quantity $h(x)$ that is a monotonic function of the probability:

$$h(x) = -\log p(x)$$

Entropy is the average amount of information defined as

$$H(x) = -\sum_x p(x) \log p(x)$$

$$H(\Omega) = -\sum_k P(\omega_k) \log P(\omega_k) = -\sum_k \frac{|\omega_k|}{N} \log \frac{|\omega_k|}{N}$$

$$H(\mathcal{C}) = -\sum_j P(c_j) \log P(c_j)$$

tending to increase with the number of clusters, e.g., $H(\Omega)$ reaches its maximum $\log N$ for $K = N$

NMI

Recall

- The minimum of MI is 0, if the clustering is random with respect to class membership
- The entropy tends to increase with the number of clusters

NMI

Recall

- The minimum of MI is 0, if the clustering is random with respect to class membership
- The entropy tends to increase with the number of clusters

Therefore, we can defined the criterion combining both as

$$\text{NMI}(\Omega, \mathcal{C}) = \frac{I(\Omega, \mathcal{C})}{[H(\Omega) + H(\mathcal{C})]/2}$$

NMI is a number between 0 and 1

Rand Index

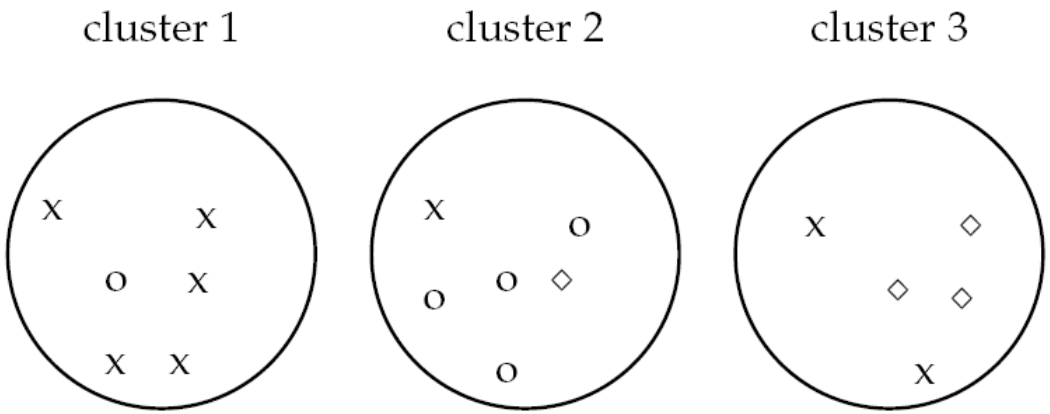
- a series of decisions
- one for each of the $N(N-1)/2$ pairs of patterns in the data set
- TP: true-positive decision - assigns two similar patterns to the same cluster
- TN: true-positive decision - assigns two dissimilar patterns to the different clusters
- FP, FN

Rand index (RI)

measures the RI percentage of decisions that are correct (simply accuracy)

$$\text{RI} = \frac{TP + TN}{TP + FP + TN + FN}$$

Rand Index: Example



F measure

- RI gives equal weights to FP and FN
- F measure - to penalize FNs more strongly than FP by selecting a value $\beta > 1$

Precision (P), Recall(R), F measure

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FP}$$

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$