CS 1675 Introduction to Machine Learning
Lecture 17

Learning complex distributions:
Hidden variables and missing values

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Learning probability distribution

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 the data

Assumptions considered so far:

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

Hidden variables

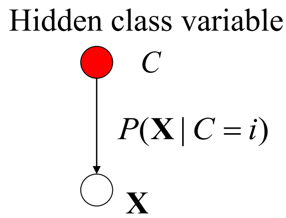
Modeling assumption:

Observed Variables $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$

- Additional **(hidden) variables** may added to the model
  – they are never observed in data

Why to add hidden variables?

- **More flexibility in describing the distribution** $P(\mathbf{X})$
- **Smaller parameterization of** $P(\mathbf{X})$
  – **New independences can be introduced via hidden variables**



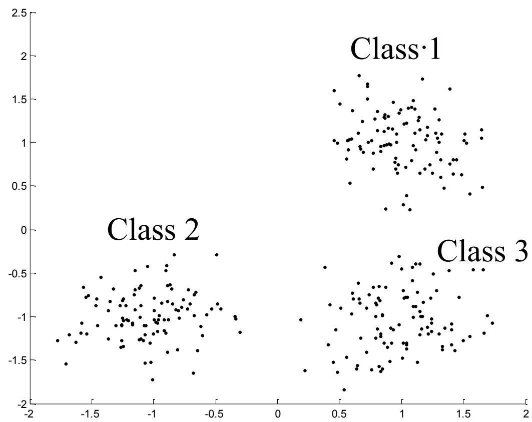
Example:

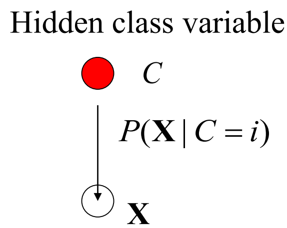
- Latent variable models
  – hidden classes (categories)

Gaussian mixture model

Assumption: **data are coming from multiple Gaussians**

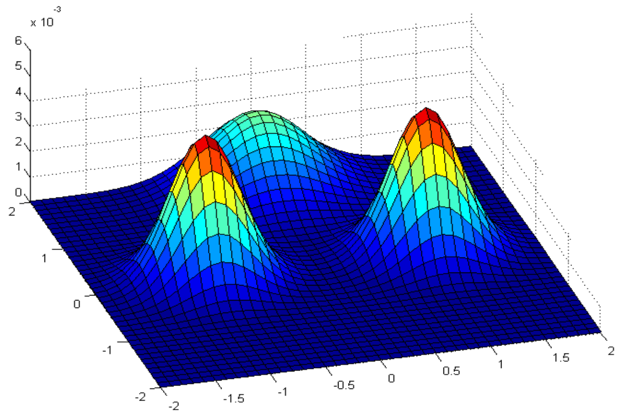
- **Hidden variable: models the different Gaussians**





Mixture of Gaussians

- Density function for the Mixture of Gaussians model



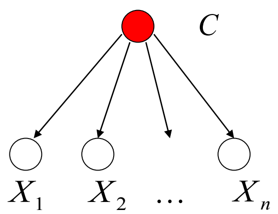
Naïve Bayes with a hidden class variable

Introduction of a hidden variable can reduce the number of parameters defining $P(\mathbf{X})$

Example:

- Naïve Bayes model with a hidden class variable

Hidden class variable



Attributes are independent given the class

- **Useful in customer profiles**
  - Class value = type of customers

Missing values

A set of random variables $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$

- **Data** $D = \{D_1, D_2, .., D_N\}$
- **But some values are missing**

$D_i = (x_1^i, x_3^i, \ldots x_n^i)$

Missing value of x_2^i

$D_{i+1} = (x_3^{i+1}, \ldots x_n^{i+1})$

Missing values of x_1^{i+1}, x_2^{i+1}

Etc.

- **Example: medical records**
- **We still want to estimate parameters of** $P(\mathbf{X})$

Density estimation

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

Estimation criterion:

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

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

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

Expectation-maximization (EM) method

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

General EM

The key idea of a method: **parameter estimates** iteratively

Pick initial set of model parameters Θ

Repeat

Set $\Theta' = \Theta$

Expectation step. For all hidden and missing variables (and their possible value assignments) calculate their expectations for all data instances given the parameters Θ'

Maximization step. Compute the new estimates of Θ by considering the expectations of the different value completions (for hidden variables and missing values for all instances)

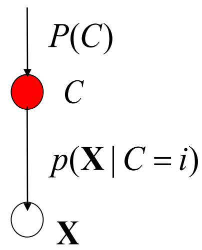
Till no improvement possible

EM advantages

Key advantages:

- In many problems (e.g. Bayesian belief networks)
  - the maximization step can be carried out in the closed form
  - We directly optimize using quantities corresponding to expected counts
- Climbs the gradient, but it does not need a learning rate, automatically renormalized update

Example: Gaussian mixture model



Probability of occurrence of a data point $\boldsymbol{x}$ is modeled as

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

where

$p(C = i)$

= 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>**Special feature:**</u> ***C*** **is hidden !!!!**

Example: Gaussian mixture model

Assume a generative classifier model based on the QDA:

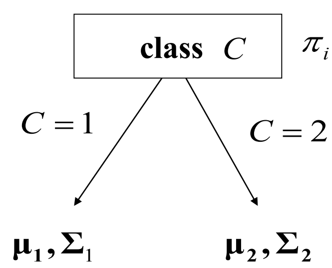
- **The class labels are known**. The ML estimate is

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

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

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

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



Example: Gaussian mixture model

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

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

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

Count replaced with the expected count

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

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

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

CS 1675 Introduction to Machine Learning
Lecture 17

Clustering

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Clustering

Groups together "similar" instances in the data sample

Basic clustering problem:

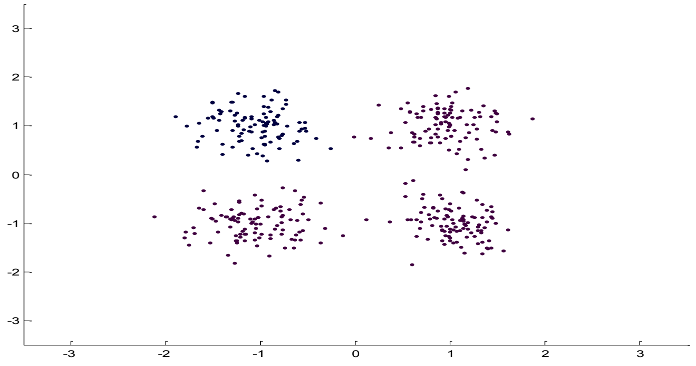
- distribute data into *k* different groups such that data points similar to each other are in the same group
- Similarity between data points is defined in terms of some distance metric (can be chosen)

Clustering is useful for:

- **Similarity/Dissimilarity analysis**
  Analyze what data points in the sample are close to each other
- **Dimensionality reduction**
  High dimensional data replaced with a group (cluster) label

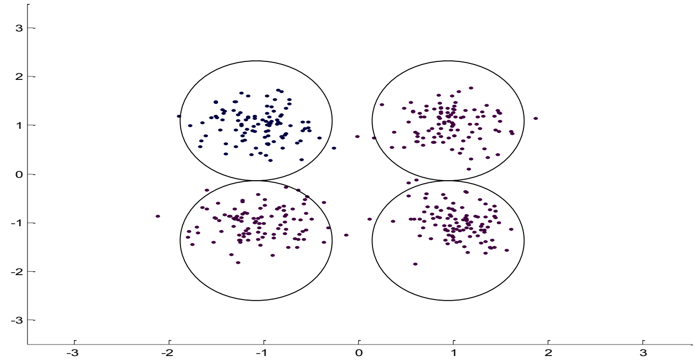
Clustering example

- We see data points and want to partition them into groups
- Which data points belong together?



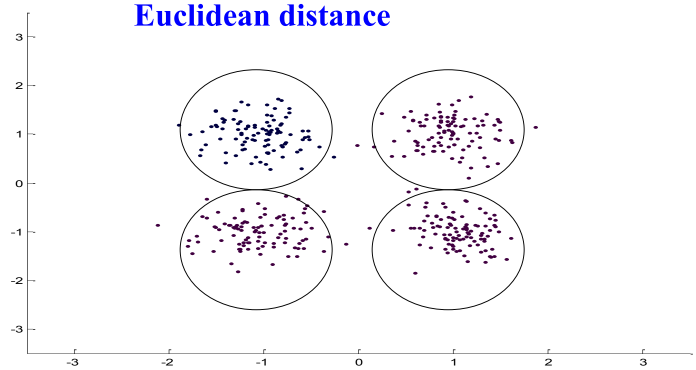
Clustering example

- We see data points and want to partition them into the groups
- Which data points belong together?



Clustering example

- We see data points and want to partition them into the groups
- Requires a distance metric to tell us what points are close to each other and are in the same group



Clustering example

- A set of patient cases
- We want to partition them into groups based on similarities

Patient #	Age	Sex	Heart Rate	Blood pressure …
Patient 1	55	M	85	125/80
Patient 2	62	M	87	130/85
Patient 3	67	F	80	126/86
Patient 4	65	F	90	130/90
Patient 5	70	M	84	135/85

Clustering example

- A set of patient cases
- We want to partition them into the groups based on similarities

Patient #	Age	Sex	Heart Rate	Blood pressure …
Patient 1	55	M	85	125/80
Patient 2	62	M	87	130/85
Patient 3	67	F	80	126/86
Patient 4	65	F	90	130/90
Patient 5	70	M	84	135/85

How to design the distance metric to quantify similarities?

Clustering example. Distance measures.

In general, one can choose an arbitrary distance measure.

Properties of distance metrics:

Assume 2 data entries *a, b*

Positiveness: $d(a,b) \geq 0$

Symmetry: $d(a,b) = d(b,a)$

Identity: $d(a,a) = 0$

Triangle inequality: $d(a,c) \leq d(a,b) + d(b,c)$

Distance measures.

Assume pure real-valued data-points:

12	34.5	78.5	89.2	19.2
23.5	41.4	66.3	78.8	8.9
33.6	36.7	78.3	90.3	21.4
17.2	30.1	71.6	88.5	12.5

…

What distance metric to use?

Distance measures

Assume pure real-valued data-points:

12	34.5	78.5	89.2	19.2
23.5	41.4	66.3	78.8	8.9
33.6	36.7	78.3	90.3	21.4
17.2	30.1	71.6	88.5	12.5

…

What distance metric to use?

Euclidian: works for an arbitrary k-dimensional space

$$d(a,b) = \sqrt{\sum_{i=1}^{k} (a_i - b_i)^2}$$

Distance measures

Assume pure real-valued data-points:

12	34.5	78.5	89.2	19.2
23.5	41.4	66.3	78.8	8.9
33.6	36.7	78.3	90.3	21.4
17.2	30.1	71.6	88.5	12.5

What distance metric to use?

Squared Euclidian: works for an arbitrary k-dimensional space

$$d^2(a,b) = \sum_{i=1}^{k} (a_i - b_i)^2$$

Distance measures

Assume pure real-valued data-points:

12	34.5	78.5	89.2	19.2
23.5	41.4	66.3	78.8	8.9
33.6	36.7	78.3	90.3	21.4
17.2	30.1	71.6	88.5	12.5

Manhattan distance:

works for an arbitrary k-dimensional space

$$d(a,b) = \sum_{i=1}^{k} | a_i - b_i |$$

Etc. ..

Γ^{-1}

Distance measures

Generalized distance metric:

$$d^2(\mathbf{a},\mathbf{b}) = (\mathbf{a}-\mathbf{b})\mathbf{\Gamma}^{-1}(\mathbf{a}-\mathbf{b})^T$$

Γ semi-definite positive matrix

Γ^{-1} is a matrix that weights attributes proportionally to their importance. Different weights lead to a different distance metric.

If $\Gamma = I$ we get squared Euclidean

$\Gamma = \Sigma$ (covariance matrix) – we get the **Mahalanobis distanc**e that takes into account correlations among attributes

Distance measures.

Assume pure binary values data:

0	1	1	0	1
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

…

What distance metric to use?

Distance measures.

Assume pure binary values data:

0	1	1	0	1
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

…

What distance metric to use?

Hamming distance: The number of bits that need to be changed to make the entries the same

How about Euclidean distance?

Distance measures.

Assume pure categorical data:

0	1	1	0	0
1	0	3	0	1
2	1	1	0	2
1	1	1	1	2
…				

What distance metric to use?

Hamming distance: The number of values that need to be changed to make them the same

Distance measures.

Combination of real-valued and categorical attributes

Patient #	Age	Sex	Heart Rate	Blood pressure …
Patient 1	55	M	85	125/80
Patient 2	62	M	87	130/85
Patient 3	67	F	80	126/86
Patient 4	65	F	90	130/90
Patient 5	70	M	84	135/85

What distance metric to use?

Distance measures.

Combination of real-valued and categorical attributes

Patient #	Age	Sex	Heart Rate	Blood pressure …
Patient 1	55	M	85	125/80
Patient 2	62	M	87	130/85
Patient 3	67	F	80	126/86
Patient 4	65	F	90	130/90
Patient 5	70	M	84	135/85

What distance metric to use?

A weighted sum approach: **e.g. a mix of Euclidian and Hamming distances for subsets of attributes**

Clustering

Clustering is useful for:

- **Similarity/Dissimilarity analysis**
  Analyze what data points in the sample are close to each other
- **Dimensionality reduction**
  High dimensional data replaced with a group (cluster) label
- **Data reduction:** Replaces many datapoints with the point representing the group mean

Problems:

- Pick the correct similarity measure (problem specific)
- Choose the correct number of groups
  - Many clustering algorithms require us to provide the number of groups ahead of time

Clustering algorithms

- **K-means algorithm**
  - **suitable** only when data points have continuous values; groups are defined in terms of cluster centers (also called **means**). Refinement of the method to categorical values: **K-medoids**
- **Probabilistic methods (with EM) = soft clustering**
  - **Latent variable models**: class (cluster) is represented by a latent (hidden) variable value
  - Every point goes to the class with the highest posterior
  - **Examples:** mixture of Gaussians, Naïve Bayes with a hidden class
- **Hierarchical methods**
  - **Agglomerative**
  - **Divisive**

K-means

K-Means algorithm:

Initialize randomly *k* values of means (centers)

Repeat two steps until no change in the means:

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

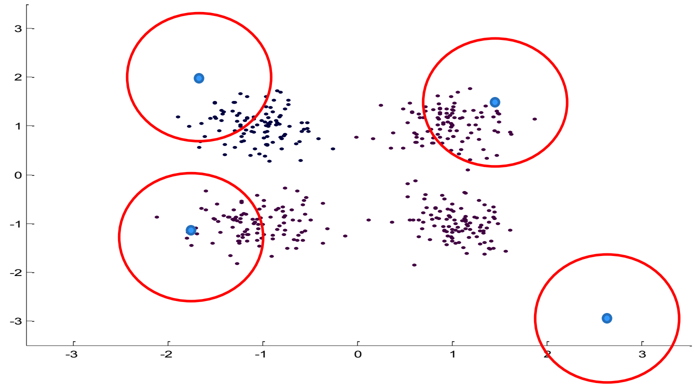
Stop when no change in the means

Properties:

- Minimizes th [redacted] for all clusters
- The algorithm always converges (to the local optima).

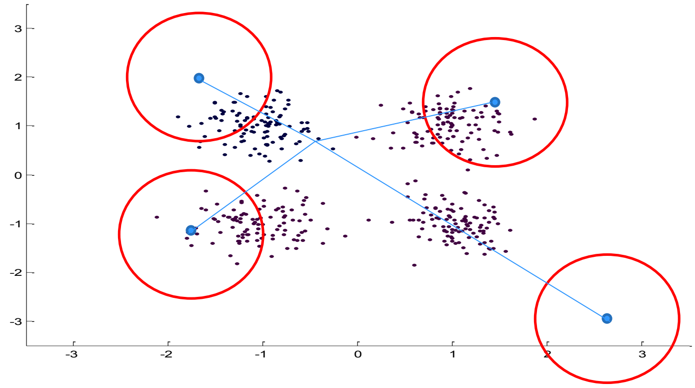
K-means: example

- Pick the cluster centers



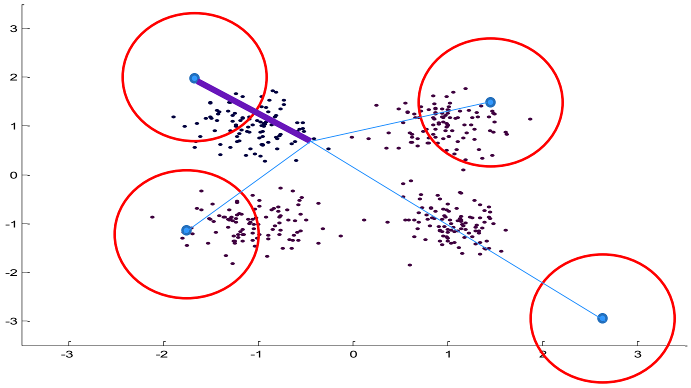
K-means: example

- Calculate the distances to each center



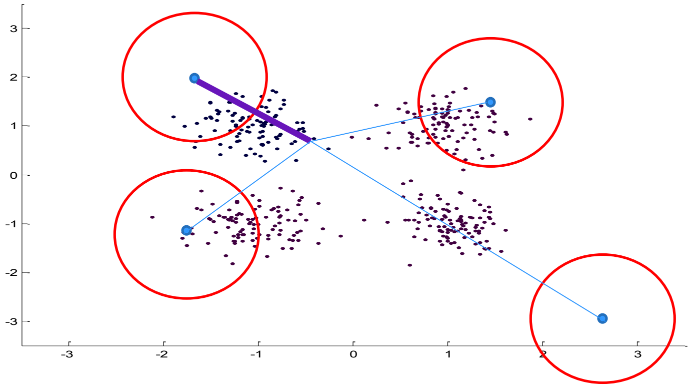
K-means: example

- For each example pick the best center



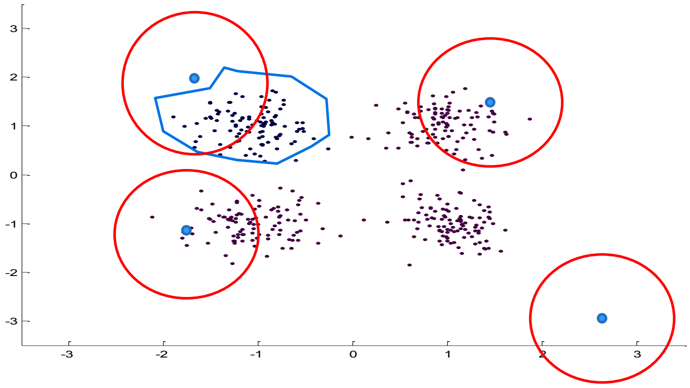
K-means: example

- For each example pick the best center



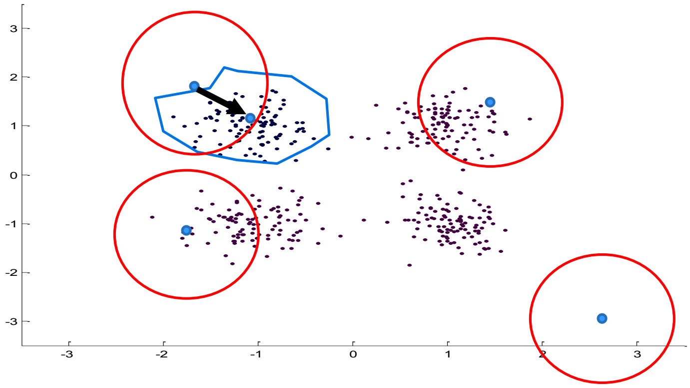
K-means: example

- Recalculate the mean from all data examples assigned to the cluster center



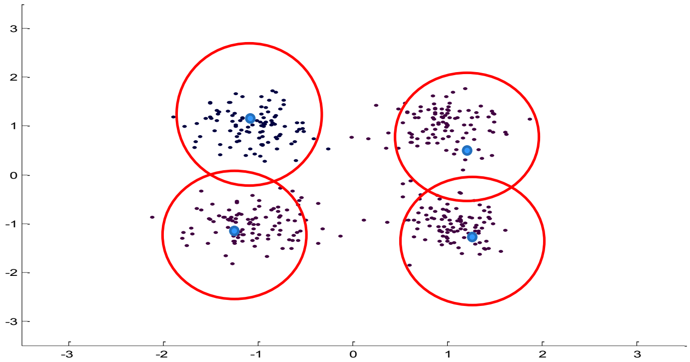
K-means: example

- Shift the cluster center to the new mean



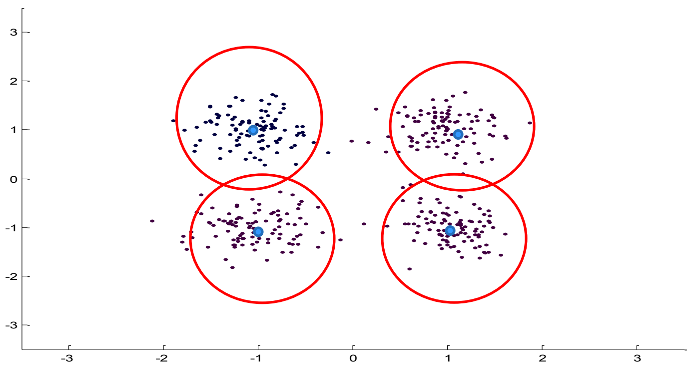
K-means: example

- Shift the cluster center to the new mean



K-means: example

- And repeat the iteration …
- Till no change in the centers



K-means algorithm

- **Properties:**
  - converges to centers minimizing the sum of squared center-point distances (still local optima)
  - The result is sensitive to the initial means' values
- **Advantages:**
  - Simplicity
  - Generality – can work for more than one distance measure
- **Drawbacks:**
  - Can perform poorly with overlapping regions
  - Lack of robustness to outliers
  - Good for attributes (features) with continuous values
    - Allows us to compute cluster means
    - k-medoid algorithm used for discrete data