Methods for Intelligent Systems

Lecture Notes on Clustering (III)
2009-2010

Davide Eynard

eynard@elet.polimi.it

Department of Electronics and Information

Politecnico di Milano

Course Schedule [*Tentative*]

Date	Topic
11/03/2010	Clustering: Introduction
18/03/2010	Clustering: K-means & Hierarchical
25/03/2010	Clustering: Fuzzy, Gaussian & SOM
08/04/2010	Clustering: PDDP & Vector Space Model
15/04/2010	Clustering: Limits, DBSCAN & Jarvis-Patrick
29/04/2010	Clustering: Evaluation Measures

Fuzzy C-Means

Fuzzy C-Means (FCM, developed by Dunn in 1973 and improved by Bezdek in 1981) is a method of clustering which allows one piece of data to belong to two or more clusters.

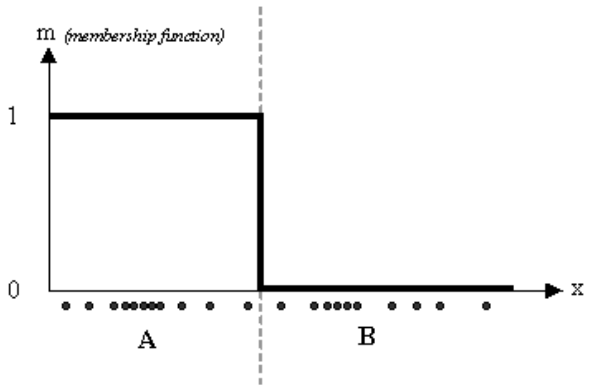
- frequently used in pattern recognition
- based on minimization of the following objective function:

$$J_m = \sum_{i=1}^{N} \sum_{j=1}^{C} u_{ij}^m \|x_i - c_j\|^2, 1 \leq m < \infty$$

where:
m is any real number greater than 1 (*fuzzyness coefficient*),
u_{ij} is the degree of membership of x_i in the cluster j,
x_i is the i-th of d-dimensional measured data,
c_j is the d-dimension center of the cluster,
$\|\cdot\|$ is any norm expressing the similarity between measured data and the center.

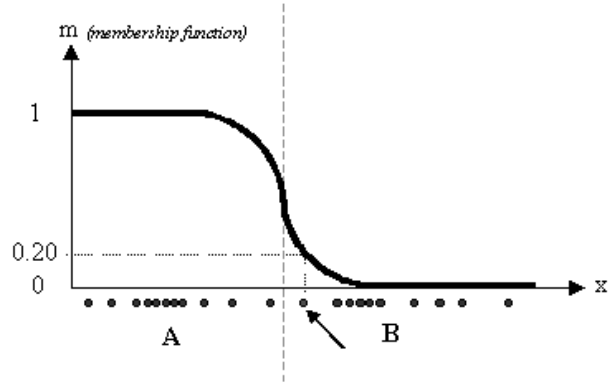
K-Means vs. FCM

- With K-Means, a datum either belongs to centroid A or to centroid B



K-Means vs. FCM

- With FCM, the same datum does not belong exclusively to one cluster, but it may belong to several clusters with different values of the membership coefficient



Data representation

$$(KM)U_{N\times C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ \dots & \dots \\ 0 & 1 \end{bmatrix}$$

$$(FCM)U_{N\times C} = \begin{bmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \\ 0.6 & 0.4 \\ \dots & \dots \\ 0.9 & 0.1 \end{bmatrix}$$

FCM Algorithm

The algorithm is composed of the following steps:

1. Initialize $U = [u_{ij}]$ matrix, $U^{(0)}$
2. At t-step: calculate the centers vectors $C^{(t)} = [c_j]$ with $U^{(t)}$:

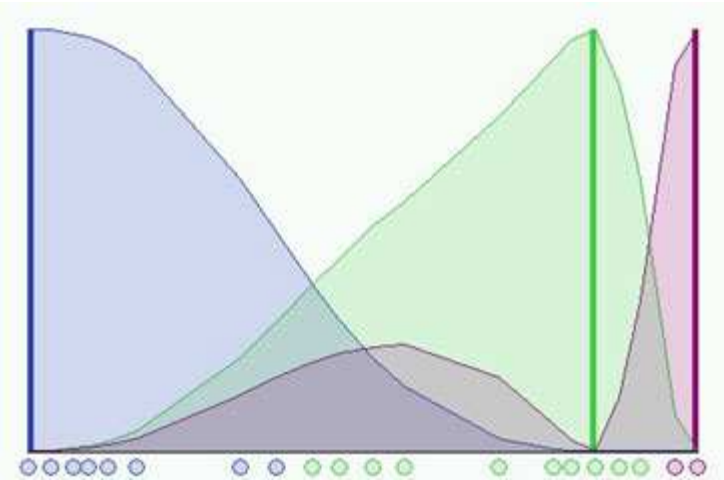
$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^m \cdot x_i}{\sum_{i=1}^{N} u_{ij}^m}$$

3. Update $U^{(t)}$, $U^{(t+1)}$:

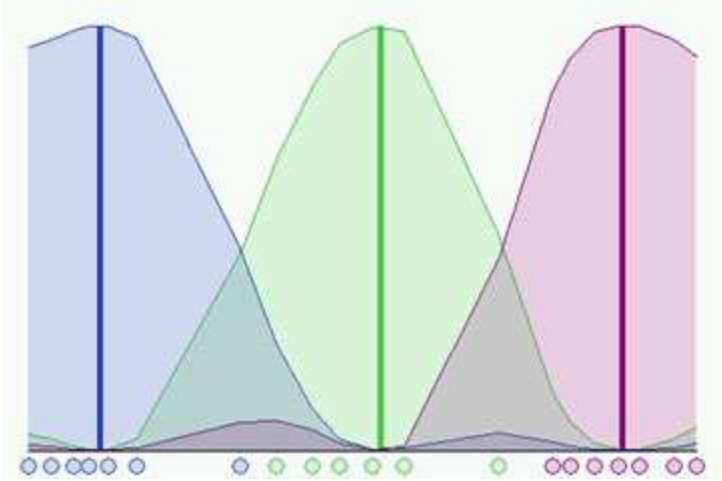
$$u_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}}}$$

4. If $\|U^{(k+1)} - U^{(k)}\| < \varepsilon$ then STOP; otherwise return to step 2.

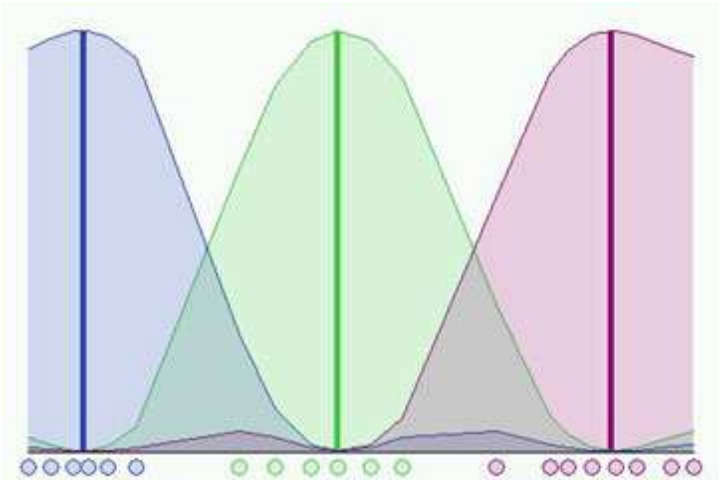
An Example



An Example



An Example



Clustering as a Mixture of Gaussians

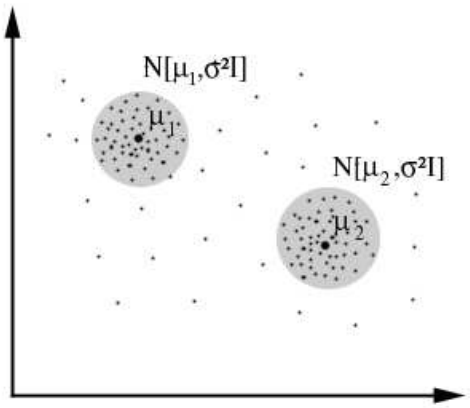
- Gaussians Mixture is a *model-based* clustering approach
  - It uses a statistical model for clusters and attempts to optimize the fit between the data and the model.
  - Each cluster can be mathematically represented by a parametric distribution, like a Gaussian (continuous) or a Poisson (discrete)
  - The entire data set is modelled by a *mixture* of these distributions
- A mixture model with high likelihood tends to have the following traits:
  - Component distributions have high "peaks" (data in one cluster are tight)
  - The mixture model "covers" the data well (dominant patterns in data are captured by component distributions)

Advantages of Model-Based Clustering

- well studied statistical inference techniques available
- flexibility in choosing the component distribution
- obtain a density estimation for each cluster
- a "soft" classification is available

Mixture of Gaussians

It is the most widely used model-based clustering method: we can actually consider clusters as Gaussian distributions centered on their barycentres (as we can see in the figure, where the grey circle represents the first variance of the distribution).



How does it work?

- it chooses the component (the Gaussian) at random with probability $P(\omega_i)$
- it samples a point $N(\mu_i, \sigma^2 I)$
  - Let's suppose we have $x_1, x_2, \ldots, x_n$ and $P(\omega_1), \ldots, P(\omega_K), \sigma$
  - We can obtain the likelihood of the sample: $P(x|\omega_i, \mu_1, \mu_2, \ldots, \mu_K)$ (probability that an observation from class ω_i would have value x given class means $\mu_1, \ldots, \mu_K$)
  - What we really want is to maximize $P(x|\mu_1, \mu_2, \ldots, \mu_K)$

... Can we do it? How?

(two little examples on *Expectation Maximization*)

The Algorithm

The algorithm is composed of the following steps:

1. Initialize parameters:

$$\lambda_0 = \{\mu_1^{(0)}, \mu_2^{(0)}, \ldots, \mu_k^{(0)}, p_1^{(0)}, p_2^{(0)}, \ldots, p_k^{(0)}\}$$

   where $p_i^{(t)}$ is shorthand for $P(\omega_i)$ at t-th iteration

2. E-step:

$$P(\omega_j|x_k, \lambda_t) = \frac{P(x_k|\omega_j, \lambda_t)P(\omega_j|\lambda_t)}{P(x_k|\lambda_t)} = \frac{P(x_k|\omega_i, \mu_i^{(t)}, \sigma^2)p_i(t)}{\sum_k P(x_k|\omega_j, \mu_j^{(t)}, \sigma^2)p_j^{(t)}}$$

3. M-step:

$$\mu_i^{(t+1)} = \frac{\sum_k P(\omega_i|x_k, \lambda_t)x_k}{\sum_k P(\omega_i|x_k, \lambda_t)}$$

$$p_i^{(t+1)} = \frac{\sum_k P(\omega_i|x_k, \lambda_t)}{R}$$

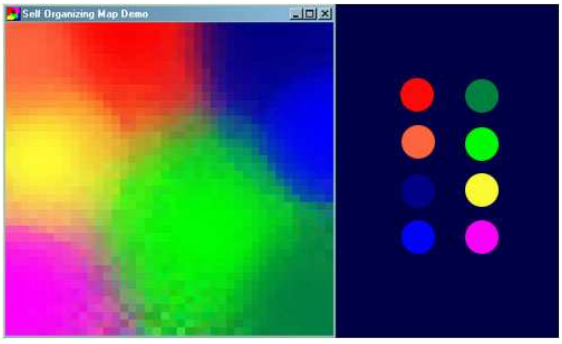
   where R is the number of records

Self Organizing Features Maps

Kohonen Self Organizing Features Maps (a.k.a. SOM) provide a way to represent multidimensional data in much lower dimensional spaces.

- They implement a data compression technique similar to *vector quantization*
- They store information in such a way that any topological relationships within the training set are maintained

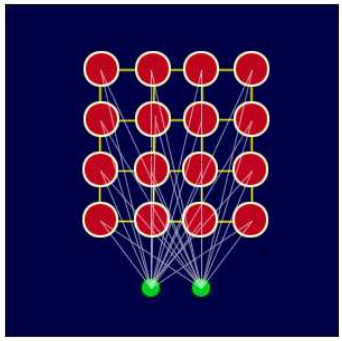
Example: Mapping of colors from their three dimensional components (i.e., red, green and blue) into two dimensions.



Self Organizing Feature Maps: The Topology

- The network is a lattice of "nodes", each of which is fully connected to the input layer
- Each node has a specific topological position and contains a vector of weights of the same dimension as the input vectors
- There are no lateral connections between nodes within the lattice

A SOM does not need a target output to be specified; instead, where the node weights match the input vector, that area of the lattice is selectively optimized to more closely resemble the data vector



Self Organizing Features Maps: The Algorithm

Training occurs in several steps over many iterations:

1. Initialize each node's weights
2. Presented a random vector from the training set to the lattice
3. Examinate every node to calculate which one's weights are most like the input vector (the winning node is commonly known as the Best Matching Unit)
4. Calculate the radius of the neighborhood of the BMU (this is a value that starts large, typically set to the 'radius' of the lattice, but diminishes each time-step), any nodes found within this radius are deemed to be inside the BMU's neighborhood
5. Each neighboring node's weights are adjusted to make them more like the input vector. The closer a node is to the BMU, the more its weights get altered
6. Repeat step 2 for N iterations

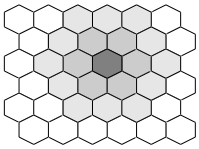
Practical Learning of Self Organizing Features Maps

There are few things that have to be specified in the previous algorithm:

- Choosing the weights initialization
- We select the Best Matching Unit according to the distance between its weights and the input vector:

$$||\mathbf{x} - \mathbf{w}_i|| = \sqrt{\sum_{k=1}^{p}(\mathbf{x}[k] - \mathbf{w}_i[k])^2}$$

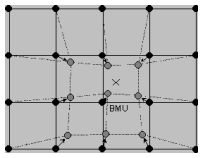
- Select the neighborhood according to some decreasing function



$$h_{ij} = e^{-\frac{(i-j)^2}{2\sigma^2}}$$

- Define the updating rule

$$\mathbf{w}_i(t+1) = \begin{cases} \mathbf{w}_i + \alpha(t)[\mathbf{x}(t) - \mathbf{w}_i(t)], & i \in N_i(t) \\ \mathbf{w}_i, & i \notin N_i(t) \end{cases}$$



Bibliography

- Clustering with gaussian mixtures
  Andrew W. Moore
- A Tutorial on Clustering Algorithms
  Online tutorial by M. Matteucci
- As usual, more info on del.icio.us