Unsupervised Learning: Optimization-based clustering

Universitat de les Illes Balears
Departament de Ciències Matemàtiques i Informàtica

11752 Aprendizaje Automático
11752 Machine Learning
Máster Universitario
en Sistemas Inteligentes

Alberto ORTIZ RODRÍGUEZ

Contents

- Introduction
- Membership-based clustering
- Possibilistic clustering
- Supplementary material: Mixture models

Introduction

- These clustering approaches (most popular in unsupervised learning) can be stated as the optimization of a **cost function** J using differential calculus techniques
- The cost function J is defined in terms of the data set $X = \{x_1, x_2, \ldots, x_N\}$ and the clustering $\mathcal{R} = \{C_1, C_2, \ldots, C_M\}$, for a **predefined number of clusters M**:

$$\mathcal{R} = \{C_1, C_2, \ldots, C_M\} \text{ such that } \bigcup_{i=1}^{M} C_i = X \text{ and } J(X;\mathcal{R}) = \min_{\mathcal{R}'}\{J(X;\mathcal{R}')\}$$

- Each cluster C_j is defined in terms of a set of **parameters** θ_j, which in turn depend on the features of the cluster we look for
  - e.g. θ_j can be a point in L-dimensional space corresponding to the centroid of the cluster
- Therefore, the problem becomes into the determination of the **optimum set of parameters** θ:

$$\theta = \{\theta_1, \theta_2, \ldots, \theta_M) = \underset{\theta'}{\operatorname{argmin}}\, J(X;\theta')$$

- Due to this optimization-based nature, with these algorithms, **all the samples of the data set X are involved** in the computation of the parameters of the clusters.

Contents

- Introduction
- Membership-based clustering
- Possibilistic clustering
- Supplementary material: Mixture models

Membership-based clustering

- In a first version, these algorithms define the cost function in the following terms:

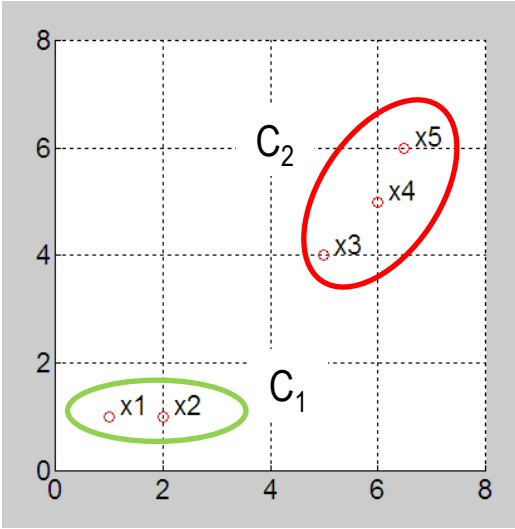
$$J(X;U,\theta)=\sum_{i=1}^{N}\sum_{j=1}^{M}u_{ij}\wp(x_i,\theta_j)$$

  - where $u_{ij}=1$ if sample x_i is assigned to cluster C_j, otherwise $u_{ij}=0$
  - J is optimized with respect to $U=\{u_{ij}\}$ and $\theta=\{\theta_j\}$.

  $\Rightarrow$ The clustering problem becomes into the determination of optimum U and θ.

    - if $\wp$ is DM, the problem is $\min J$; if $\wp$ is SM, the problem is $\max J$

- Example: given $N=5$ and $M=2$, the following describes an optimum clustering



$\wp(x_i,\theta_j)=d_2(x_i,\theta_j),\theta_j$ is the centroid of C_j

$$\theta_1=\frac{x_1+x_2}{2}=(1.5,1.0),\theta_2=\frac{x_3+x_4+x_5}{3}=(5.83,5.0)$$

u_{ij}	C_1	C_2
x_1	1	0
x_2	1	0
x_3	0	1
x_4	0	1
x_5	0	1

i.e. $u_{11}=1, u_{12}=0$

$$J=d_2(x_1,C_1)+d_2(x_2,C_1)+d_2(x_3,C_2)+d_2(x_4,C_2)+d_2(x_5,C_2)$$
$$=0.5+0.5+1.3017+0.1667+1.2019=3.6702$$

Membership-based clustering

- The previous problem, in which each sample belongs to exclusively **one single cluster**, is known as **hard** or **crisp clustering**:

$$\min / \max \; J(X; U, \theta) = \sum_{i=1}^{N} \sum_{j=1}^{M} u_{ij} \wp(x_i, \theta_j)$$

$$\text{subject to } u_{ij} \in \{0, 1\}, \; i = 1, \ldots, N, \; j = 1, \ldots, M$$

$$\text{with } \sum_{j=1}^{M} u_{ij} = 1, \; i = 1, \ldots, N$$

- To find the optimum $U = \{u_{ij}\}$ and $\theta = \{\theta_j\}$, we can use differential calculus for θ_j, but not for u_{ij}, because J is not differentiable with respect to $u_{ij} \in \{0,1\}$.
- Instead, we adopt the following (sub-optimum) rule:
  - Keeping θ_j constant, $j = 1, \ldots, M$, since for each x_i only one u_{ij} is 1 and the others are 0, we set

$$u_{ij} = \begin{cases} 1 & \text{if } \wp(x_i, \theta_j) = \min_k \wp(x_i, \theta_k) \quad (\wp \text{ is DM}) \\ & \text{if } \wp(x_i, \theta_j) = \max_k \wp(x_i, \theta_k) \quad (\wp \text{ is SM}) \\ 0 & \text{otherwise} \end{cases}$$

  since $J(X; U, \theta)$ is minimized if we assign each x_i to its closest/most similar cluster

Membership-based clustering

- If we now keep constant $U = \{u_{ij}\}$, we can find $\theta = \{\theta_j\}$ using differential calculus and solving the resulting equation once $\wp$ is chosen:

$$J(X;U,\theta) = \sum_{i=1}^{N}\sum_{j=1}^{M} u_{ij}\wp(x_i,\theta_j) \Rightarrow \frac{\partial J(X;U,\theta)}{\partial \theta_j} = \sum_{i=1}^{N} u_{ij}\frac{\partial \wp(x_i,\theta_j)}{\partial \theta_j} = 0$$

- We can now state the **Generalized Clustering Hard Algorithmic Scheme** (GCHAS):

1. choose $\theta_j(0)$ as initial estimates for $\theta_j, j = 1,\ldots,M$
2. $t = 0$

repeat

the process can be reversed: set θ and next calculate U

stage 1: estimate u_{ij} keeping θ_j at their previous values

3.1 **for** i = 1 **to** N
  for j = 1 **to** M

$$u_{ij}(t) = \begin{cases} 1 & \text{if } \wp(x_i,\theta_j) = \min_k / \max_k \wp(x_i,\theta_k) \quad (\wp \text{ is DM/SM}) \\ 0 & \text{otherwise} \end{cases}$$

  end
  end

stage 2: estimate θ_j keeping u_{ij} at their previous values

3.2 **for** j = 1 **to** M
  solve for $\theta_j(t+1)$ in $\sum_{i=1}^{N} u_{ij}(t)\frac{\partial \wp(x_i,\theta_j)}{\partial \theta_j} = 0$
  end

4. $t = t + 1$

until a termination criterion is met, e.g. $\|\theta(t) - \theta(t-1)\| < \epsilon$

Membership-based clustering

- The most popular instance of GCHAS is the so-called **K-means algorithm** (also known as **C-means** or **Isodata**)
- Each cluster is represented by its centroid, $\theta_j = \mu_j$, and $\wp = d_2(x_i, \mu_j)^2$

$$\sum_{i=1}^{N} u_{ij}(t) \frac{\partial \wp(x_i, \theta_j)}{\partial \theta_j} = 0$$

$$\Rightarrow \sum_{i=1}^{N} u_{ij}(t) \frac{\partial (x_i - \mu_j)^T (x_i - \mu_j)}{\partial \mu_j}$$

$$= 2 \sum_{i=1}^{N} u_{ij}(t)(x_i - \mu_j) = 0$$

$$\Rightarrow \mu_j(t+1) = \frac{\sum_{i=1}^{N} u_{ij}(t) x_i}{\sum_{i=1}^{N} u_{ij}(t)} \quad \Rightarrow$$

1. choose $\mu_j(0)$ as initial estimates for $\mu_j, j = 1, \ldots, M$
2. $t = 0$

repeat

3.1 **for** i = 1 **to** N
for j = 1 **to** M

$$u_{ij}(t) = \begin{cases} 1 & \text{if } d_2(x_i, \mu_j) = \min_k d_2(x_i, \mu_k) \\ 0 & \text{otherwise} \end{cases}$$

end
end

3.2 **for** j = 1 **to** M

$$\mu_j(t+1) = \frac{\sum_{x_i \in C_j} x_i}{n_j}$$

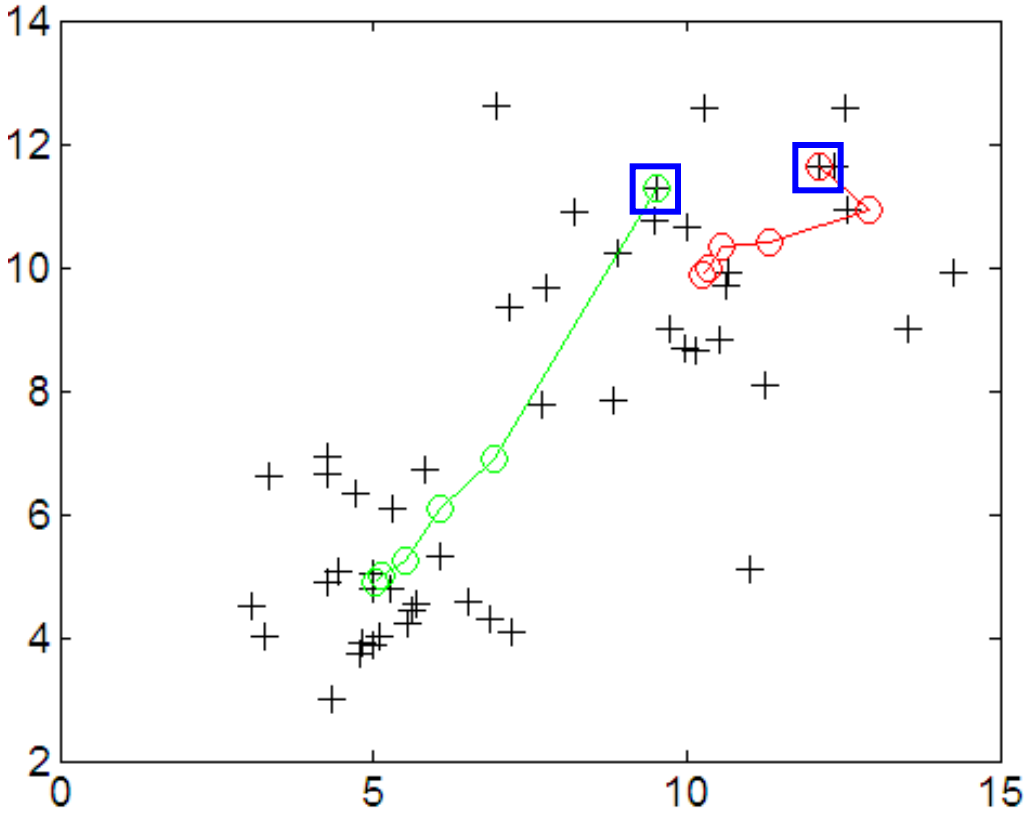
end

4. $t = t + 1$

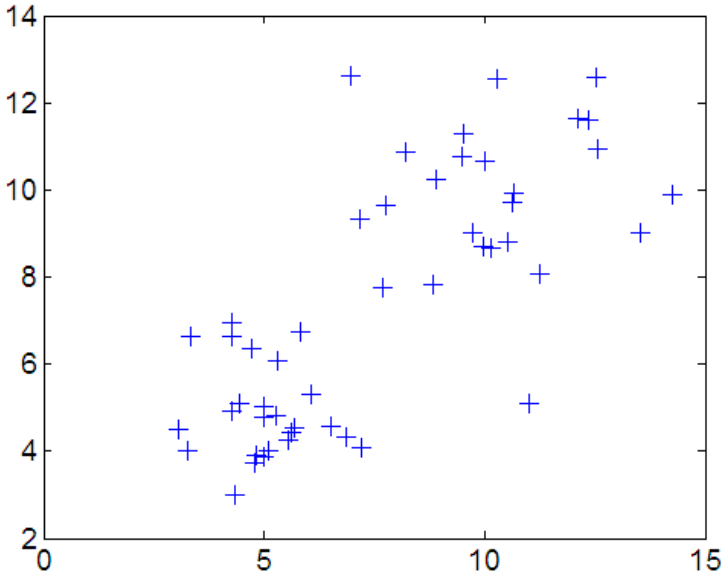
until a termination criterion is met, e.g. $\|\mu(t) - \mu(t-1)\| < \epsilon$

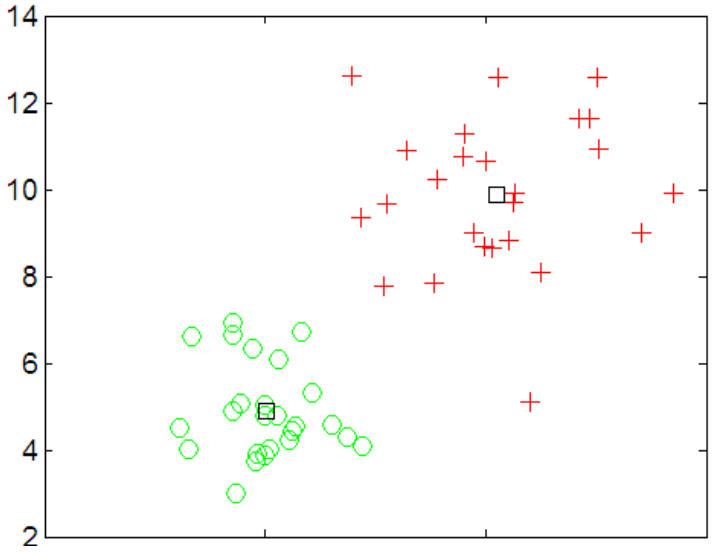
Membership-based clustering

- **Example**: Two Gaussian classes such that
  $n_1 = n_2 = 50$ – $\mu_1 = (10,10)$, $\sigma_1 = 2$ – $\mu_2 = (5,5)$, $\sigma_2 = 1$



initial centroids: blue squares
6 iterations
final centroids: $\mu_1 = (10.24, 9.89)$, $\mu_2 = (5.02, 4.90)$





Membership-based clustering

- K-means is **used widely** and frequently **finds reasonable solutions** quickly
  - It is conceptually simple as well as its implementation
  - Its time complexity is $\mathcal{O}(NMq)$, where q is the number of iterations until convergence
    - if $M <<< N$ and $q <<< N$, K-means becomes eligible for processing large datasets
- However, there are also major **shortcomings**:
  - Although K-means has been proved not to increase J between iterations, **convergence to the global optimum cannot be ensured** (solve for the optimum is NP-hard)
    - The outcome depends on the initial centroids
  - K-means is **sensitive to outliers and noise**
  - One or more **clusters can be empty** because of centroids too far away from any sample, i.e. the cluster has a representative but comprises no samples (due to bad initialization)
- To face these drawbacks:
  - One can **run K-means several times** (with different randomly chosen initial centroids) and keep the clustering leading to the lowest value of the cost function J
  - **K-means++** modifies the setup stage of K-means by a smarter initialization stage
  - **K-medoids** faces the outlier-sensitivity and naturally avoids empty clusters

- **K-means++** as initialization of K-means
  - The intuition behind K-means++ is that **spreading** out the M initial cluster centroids is beneficial, e.g. at least decreases the number of iterations until convergence
  - The following pseudo-code assumes the use of a **distance as the proximity measure**:

1. $t = 1$
2. Initialize the set of centroids S with a sample $\mu_t = x \in X$ chosen at random
3. $t = t + 1$
4. Compute the minimum distance of samples x_i in X to the centroids in S:

$$d(x_i, S)^2 = \min\{d(x_i, \mu_j)^2, x_i \notin S, \mu_j \in S\}$$

5. Randomly select the new centroid $\mu_t = x_i$ according to a probability proportional to $d(x_i, S)^2$, i.e. the larger the distance the higher the probability:

$$p(\mu_t = x_i) = \frac{d(x_i, S)^2}{\sum_{i=1}^{N} d(x_i, S)^2}, \quad d(x_i, S) = \min_{\mu_j \in S}\{d(x_i, \mu_j)\}$$

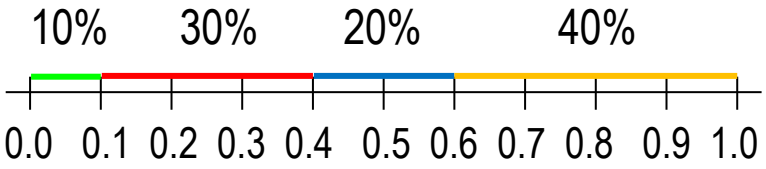
6. **if** $t < M$ **go to** step 3

Membership-based clustering

- **K-means++** initialization: Python code

```
def initialize(X, K):
    S = [X[0]]
    for k in range(1, K):
        D2 = numpy.array([min([numpy.inner(s-x,s-x) for s in S]) for x in X])
        probs = D2/D2.sum() # x in S will lead to prob = 0 and will not be chosen
        cumprobs = probs.cumsum()
        r = scipy.rand()
        for j,p in enumerate(cumprobs):
            if r < p:
                i = j
                break
        S.append(X[i])
    return S
```

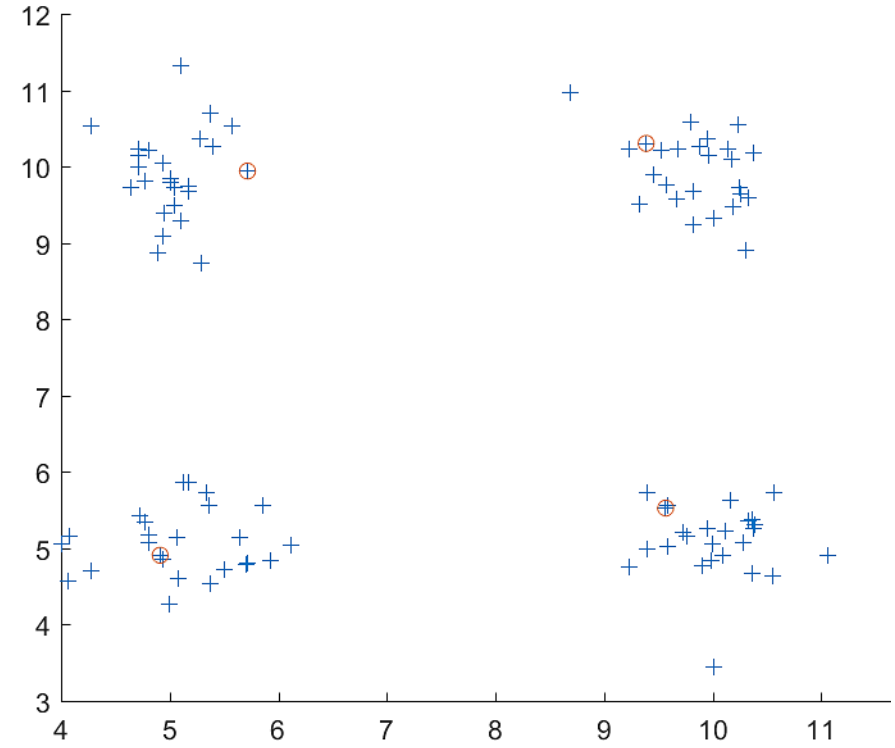
r is generated uniformly in the inteval [0,1] ⇒
10% probability of r falling in 1st interval
30% probability of r falling in 2nd interval
20% probability of r falling in 3rd interval
40% probability of r falling in 4th interval

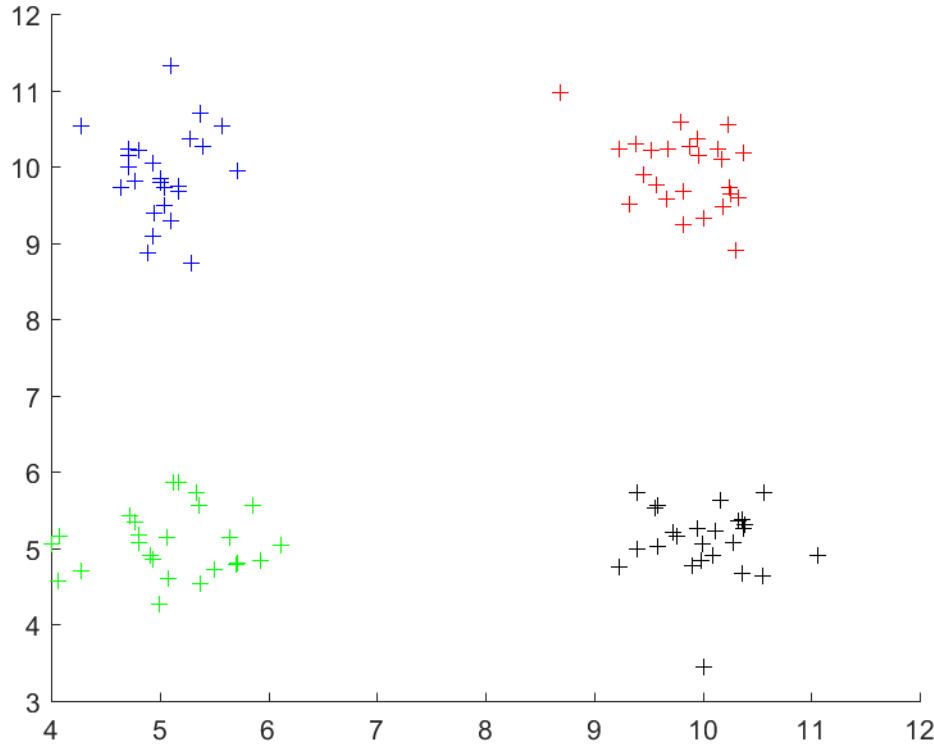



```
example (of centroid selection):
probs = [0.1, 0.3, 0.2, 0.4]
cumprobs = [0.1, 0.4, 0.6, 1.0]
if r < cumprobs[0]:
    # this event has probability 0.1
    i = 0
elif r < cumprobs[1]: # r ≥ cumprobs[0]
    # this event has probability 0.2
    i = 1
elif r < cumprobs[2]: # r ≥ cumprobs[1]
    # this event has probability 0.3
    i = 2
elif r < cumprobs[3]: # r ≥ cumprobs[2]
    # this event has probability 0.4
    i = 3
```

Membership-based clustering

- **Example**: 4 Gaussian clases, 25 samples each





Membership-based clustering

- **K-medoids** – PAM (Partioning Around Medoids) algorithm
  - Each cluster is represented by a sample from X, a **medoid**, instead of its centroid
  - Therefore, the cost function becomes:

$$J(X;U,\theta) = \sum_{x_i \in X-\mathcal{M}} \sum_{x_j \in \mathcal{M}} u_{ij} \wp(x_i, x_j)$$

    where $\mathcal{M}$ is the set of samples which are medoids
  - At **every iteration**:
    1. K-medoids assigns each sample to the closest medoid ($\equiv$ closest cluster)
    2. K-medoids checks whether there exists a sample x_i which can replace a medoid x_j and reduce J

       If this is possible, the medoid is redefined and a new iteration takes place; otherwise, the process stops
  - This procedure:
    - avoids empty clusters, i.e. each cluster comprises at least one sample (= medoid),
    - tends to be less sensitive to outliers and noise
  - CLARA and CLARANS are variants of PAM intended for dealing with large datasets

Membership-based clustering

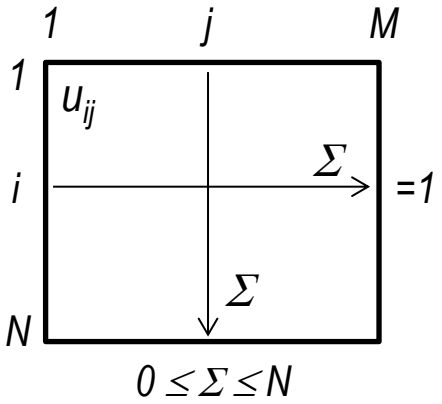
- A relaxation of constraint $u_{ij} \in \{0,1\}$ leads to the so-called **soft clustering problems**
- One of these problems is the **fuzzy clustering problem**, whose formulation is:

$$\min J(X;U,\theta) = \sum_{i=1}^{N}\sum_{j=1}^{M} u_{ij}^{q}\wp(x_i,\theta_j)$$

$$\text{subject to } \sum_{j=1}^{M} u_{ij} = 1,\ i = 1,\ldots,N$$

$$\text{hence } u_{ij} \in [0,1],\ i = 1,\ldots,N,\ j = 1,\ldots,M$$

$$\text{and } 0 \leq \sum_{i=1}^{N} u_{ij} \leq N,\ j = 1,\ldots,M$$



where:

- $\wp$ is a **DM** for fuzzy clustering algorithms
- $u_{ij} \in [0,1]$ represents the **grade of membership** of sample x_i to cluster C_j
  - u_{ij} is related to u_{ik}, $k \neq j$ because of the **constraints**
- q is named as a **fuzzyfier** which gives more or less importance to u_{ij}

  $\wp$ is DM and $q > 1 \Rightarrow u_{ij}^{q} < u_{ij} \Rightarrow u_{ij}^{q}\wp(x_i,\theta_j) < u_{ij}\wp(x_i,\theta_j)$

  $\Rightarrow$ reduces the influence of $\wp$, what can be useful for dealing with **outliers**

Membership-based clustering

- Now, we can find the optimum U and θ using differential calculus:

$$J(X;U,\theta)=\sum_{i=1}^{N}\sum_{j=1}^{M}u_{ij}^{q}\wp(x_i,\theta_j),\quad \text{subject to}\ \sum_{j=1}^{M}u_{ij}=1,\ i=1,\ldots,N$$

  - To find the **optimum** $U=\{u_{ij}\}$ we need to introduce the following **Lagrangian function** because of the constraints on u_{ij} :

$$\mathcal{L}(X;U,\theta)=\sum_{i=1}^{N}\sum_{j=1}^{M}u_{ij}^{q}\wp(x_i,\theta_j)-\sum_{i=1}^{N}\lambda_i\left(\sum_{j=1}^{M}u_{ij}-1\right)$$

  - The partial derivative of $\mathcal{L}(X;\ U,\theta)$ with respect to u_{rs} is

$$\frac{\partial\mathcal{L}(X;U,\theta)}{\partial u_{rs}}=qu_{rs}^{q-1}\wp(x_r,\theta_s)-\lambda_r,\quad \frac{\partial\mathcal{L}(X;U,\theta)}{\partial\lambda_r}=\sum_{j=1}^{M}u_{rj}-1,$$

  - We can now equal to 0 and solve for u_{rs}:

$$qu_{rs}^{q-1}\wp(x_r,\theta_s)-\lambda_r=0\Rightarrow u_{rs}=\left(\frac{\lambda_r}{q\wp(x_r,\theta_s)}\right)^{\frac{1}{q-1}}$$

$$\sum_{j=1}^{M}u_{rj}=1\Rightarrow\sum_{j=1}^{M}\left(\frac{\lambda_r}{q\wp(x_r,\theta_j)}\right)^{\frac{1}{q-1}}=1\Rightarrow\lambda_r=\frac{q}{\left(\sum_{j=1}^{M}\left(\frac{1}{\wp(x_r,\theta_j)}\right)^{\frac{1}{q-1}}\right)^{q-1}}$$

$$\Rightarrow u_{rs}=\frac{1}{\sum_{j=1}^{M}\left(\frac{\wp(x_r,\theta_s)}{\wp(x_r,\theta_j)}\right)^{\frac{1}{q-1}}}$$

Membership-based clustering

- Now, we can find optimum U and θ using differential calculus:

$$J(X;U,\theta)=\sum_{i=1}^{N}\sum_{j=1}^{M}u_{ij}^{q}\wp(x_i,\theta_j),\quad \text{subject to } \sum_{j=1}^{M}u_{ij}=1,\ i=1,\ldots,N$$

  - To find the **optimum** $\theta=\{\theta_j\}$ we calculate the partial derivative with respect to θ_j and equal it to 0: $\dfrac{\partial J(X;U,\theta)}{\partial\theta_j}=\sum_{i=1}^{N}u_{ij}^{q}\dfrac{\partial\wp(x_i,\theta_j)}{\partial\theta_j}=0,\quad j=1,\ldots,M$

- Summing up, we have:

$$u_{rs}=\frac{1}{\sum_{j=1}^{M}\left(\frac{\wp(x_r,\theta_s)}{\wp(x_r,\theta_j)}\right)^{\frac{1}{q-1}}},\quad \sum_{i=1}^{N}u_{ij}^{q}\frac{\partial\wp(x_i,\theta_j)}{\partial\theta_j}=0, j=1,\ldots,M$$

- There are two issues with the expressions we have just obtained:
  - The expression leading to the calculation of $\theta=\{\theta_j\}$ depends on the particular proximity function we are using, so the derivative and the underlying equation cannot be solved in general
  - Besides, the two expressions are coupled, what prevents from obtaining closed-form expressions
- REMARK: Notice that $J(X;U,\theta)=\sum_{i=1}^{N}\sum_{j=1}^{M}u_{ij}^{q}\wp^2(x_i,\theta_j)\Rightarrow u_{rs}=\dfrac{1}{\sum_{j=1}^{M}\left(\frac{\wp(x_r,\theta_s)}{\wp(x_r,\theta_j)}\right)^{\frac{2}{q-1}}}$

Membership-based clustering

- One way of proceeding is to employ a **two-stage iterative algorithm** named GCFAS

1. choose $\theta_j(0)$ as initial estimates for $\theta_j, j = 1, \ldots, M$
2. $t = 0$

repeat

stage 1: estimate u_{ij} keeping θ_j at their previous values

3.1 **for** i = 1 **to** N
for j = 1 **to** M

$$u_{ij}(t) = \frac{1}{\sum_{k=1}^{M} \left(\frac{\wp(x_i, \theta_j(t))}{\wp(x_i, \theta_k(t))} \right)^{\frac{1}{q-1}}}$$

end
end

stage 2: estimate θ_j keeping u_{ij} at their previous values

3.2 **for** j = 1 **to** M
solve for $\theta_j(t+1)$ in $\sum_{i=1}^{N} u_{ij}^q(t) \frac{\partial \wp(x_i, \theta_j)}{\partial \theta_j} = 0$
end

4. $t = t + 1$

until a termination criterion is met, e.g. $\|\theta(t) - \theta(t-1)\| < \epsilon$

- the algorithm can also be started from $U(0)$ instead of $\theta(0)$
- u_{ij} cannot be calculated for x_i if $\exists j$ such that $\wp(x_i, \theta_j(t)) = 0$, e.g. $\wp$ is DM
  - check this case and set u_{ij} so that x_i is arbitrarily shared among clusters C_j

$$\sum_{j | \wp(x_i, \theta_j) = 0} u_{ij} = 1$$

- If θ_j is a point representative of cluster C_j, e.g. its (fuzzy) centroid, and $\wp(x_i, \theta_j(t))$ is the squared Euclidean distance then:

$$\wp(x_i, \theta_j) = d_2(x_i, \theta_j)^2 = (x_i - \theta_j)^T (x_i - \theta_j)$$

$$\Rightarrow \sum_{i=1}^{N} u_{ij}^q(t) \frac{\partial \wp(x_i, \theta_j)}{\partial \theta_j} = 0 \Rightarrow \sum_{i=1}^{N} u_{ij}^q(t) 2 \left(x_i - \theta_j(t+1)\right) = 0$$

$$\Rightarrow \sum_{i=1}^{N} u_{ij}^q(t) x_i = \theta_j(t+1) \sum_{i=1}^{N} u_{ij}^q(t)$$

$$\Rightarrow \theta_j(t+1) = \frac{\sum_{i=1}^{N} u_{ij}^q(t) x_i}{\sum_{i=1}^{N} u_{ij}^q(t)}$$

  - The resulting algorithm is named **Fuzzy c-Means** (FCM) or **Soft / Fuzzy k-Means**
  - This result is also valid if $\wp(x_i, \theta_j) = (x_i - \theta_j)^T A (x_i - \theta_j)$ for A symmetric, positive definite (i.e. all eigenvalues are positive) e.g. A^{-1} can be the **fuzzy covariance matrix** of C_j : $A^{-1} = \sum_{x_i} u_{ij}^q (x_i - \theta_j)(x_i - \theta_j)^T$ In this case, $\wp(x_i, \theta_j(t))$ is termed as the **Mahalanobis distance**

- **Fuzzy c-Means** clustering algorithm (FCM):

1. choose $\theta_j(0)$ as initial estimates for $\theta_j, j = 1, \ldots, M$
2. $t = 0$

repeat

3.1 **for** i = 1 **to** N
for j = 1 **to** M

$$u_{ij}(t) = \frac{1}{\sum_{k=1}^{M} \left(\frac{d_2(x_i, \theta_j(t))^2}{d_2(x_i, \theta_k(t))^2} \right)^{\frac{1}{q-1}}}$$

end
end

3.2 **for** j = 1 **to** M

$$\theta_j(t+1) = \frac{\sum_{i=1}^{N} u_{ij}^q(t) x_i}{\sum_{i=1}^{N} u_{ij}^q(t)}$$

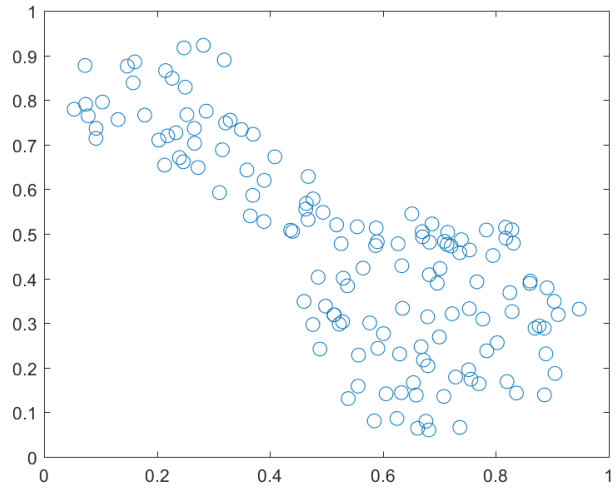
end

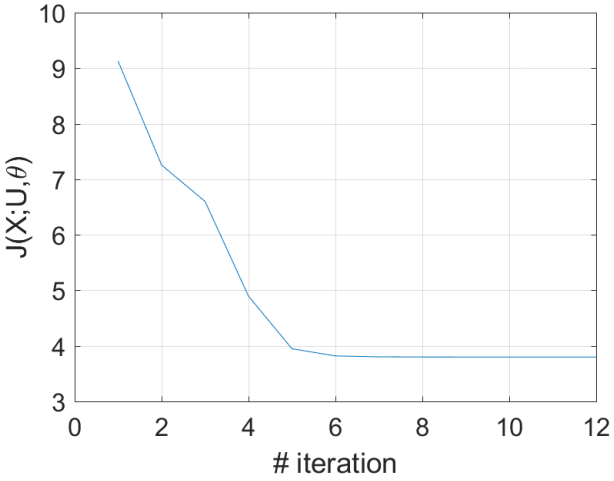
4. $t = t + 1$

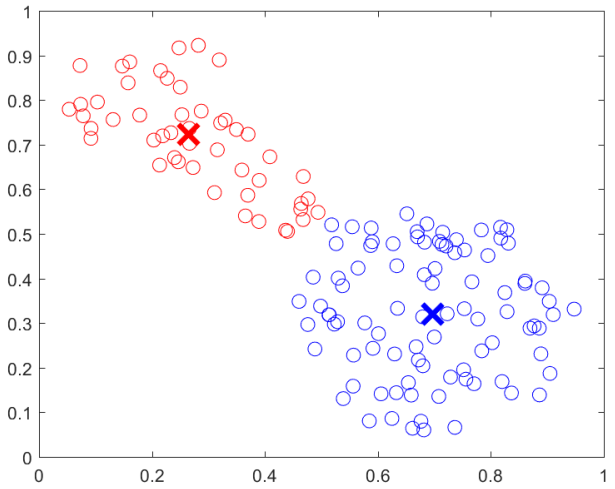
until a termination criterion is met, e.g. $\|\theta(t) - \theta(t-1)\| < \epsilon$

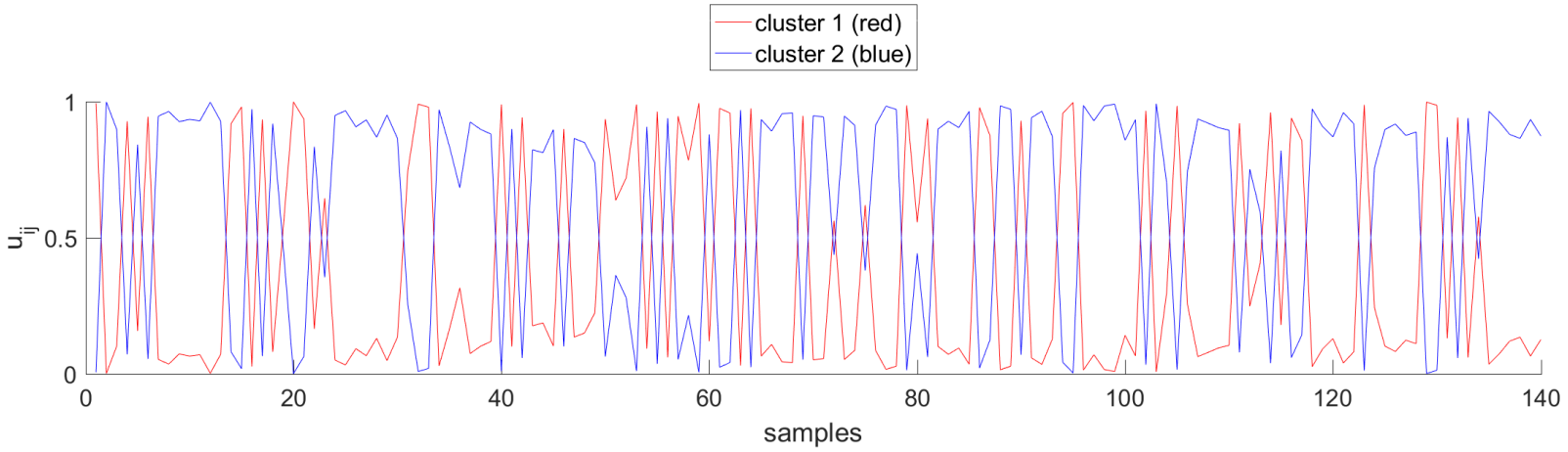
Membership-based clustering

- Example: N = 140 samples, M = 2 clusters, q = 2









Membership-based clustering

- Other FC algorithms: **Gustafson-Kessel** algorithm for **hyperplanes clustering**
  - Planar clusters are represented by centers c_j and covariance matrices Σ_j, i.e. $\theta_j = (c_j, \Sigma_j)$

$$\min J_{GK}(X;U,\theta) = \sum_{i=1}^{N}\sum_{j=1}^{M} u_{ij}^{q} d_{GK}(x_i,\theta_j)$$

$$\text{subject to } \sum_{j=1}^{M} u_{ij} = 1,\ i = 1,\ldots,N$$

$$\text{and } d_{GK}(x_i,\theta_j) = |\Sigma_j|^{1/L}(x_i - c_j)^T \Sigma_j^{-1}(x_i - c_j)$$

  - Following the same derivation procedure as before:

3.1 **for** i = 1 **to** N

for j = 1 **to** M

$$u_{ij}(t) = \frac{1}{\sum_{k=1}^{M}\left(\frac{d_{GK}(x_i,\theta_j(t))}{d_{GK}(x_i,\theta_k(t))}\right)^{\frac{1}{q-1}}}$$

end

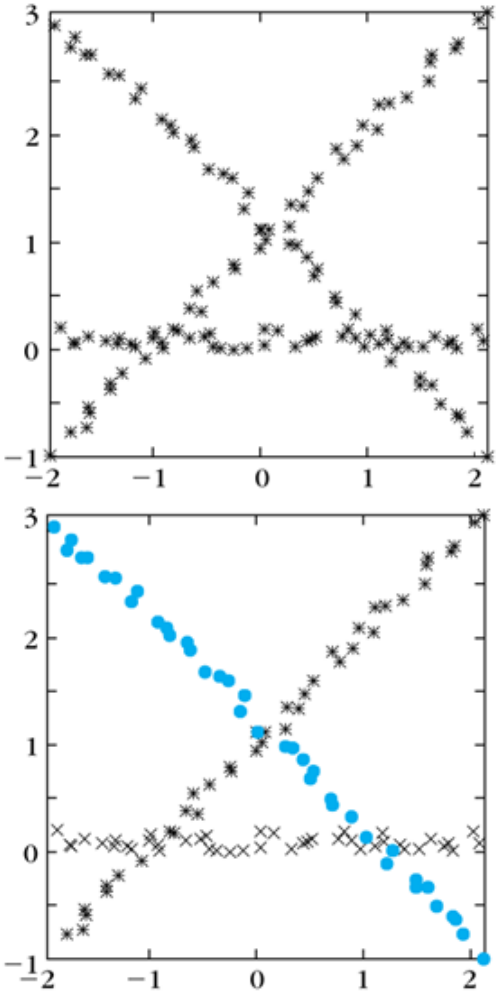
end

3.2 **for** j = 1 **to** M

$$c_j(t+1) = \frac{\sum_{i=1}^{N} u_{ij}^{q}(t) x_i}{\sum_{i=1}^{N} u_{ij}^{q}(t)}$$

$$\Sigma_j(t+1) = \frac{\sum_{i=1}^{N} u_{ij}^{q}(t)(x_i - c_j(t))(x_i - c_j(t))^T}{\sum_{i=1}^{N} u_{ij}^{q}(t)}$$

end



Membership-based clustering

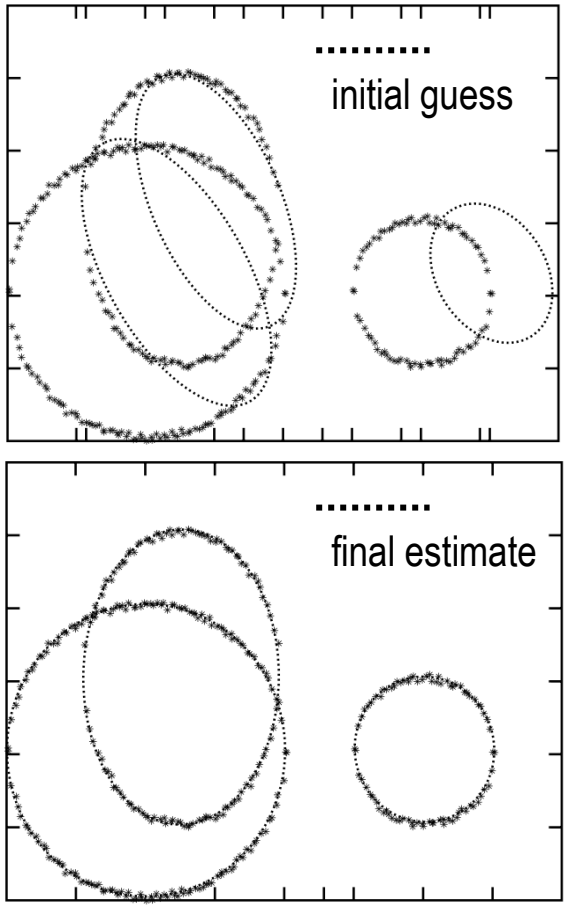
- Other FC algorithms: **hyperellipsoids clustering**
  - *Adaptive Fuzzy C-Shells clustering* (AFCS), where a hyperellipsoid is represented by its center c_j and its shape, defined by a symmetric, positive definite matrix A_j, i.e. $\theta_j = (c_j, A_j)$

$$\min\ J_{nr}(X;U,\theta) = \sum_{i=1}^{N}\sum_{j=1}^{M} u_{ij}^{q} d_{nr}^{2}(x_i,\theta_j)$$

$$\text{subject to } \sum_{j=1}^{M} u_{ij} = 1,\ i = 1,\ldots,N$$

$$\text{and } d_{nr}(x_i,\theta_j) = \sqrt{(x_i - c_j)^T A_j (x_i - c_j)} - 1$$

  - d_{nr} is the **normalized radial distance** (distance to the center of the hyperellipsoid)
  - very similar to hyperplanes clustering: $A_j = \Sigma_j^{-1}$
  - another alternative: *Fuzzy C Ellipsoidal shells* (FCES)



Membership-based clustering

- **Last remarks** about membership-based clustering:
  - K-means is considered as **hard clustering** (HC) against FCM which is considered as **soft clustering** (SC) as a result of the relaxation of the membership-related constraints
    - Actually, K-means can be regarded as a **particular case** of FCM
  - HC algorithms are not as **robust** as their SC counterparts when other than point representatives are used, e.g. the **hard-clustering version of the GK algorithm for hyperplanes clustering** needs an adequate number of samples from all underlying clusters to avoid degenerate cases where Σ_j is not invertible
    - If the descriptor of a sample x_i changes slightly, an HC algorithm might lead to **reassigning** x_i from one cluster to another,
      while, for the case of a SC algorithm, the membership values u_{i*} might be modified but **drastic changes will be more difficult to occur**
  - In general terms, the fuzzy concepts embedded in SC algorithms make them more **flexible** than HC algorithms, i.e. able to **deal better with noise and data uncertainty**
    - By means of FCM, we obtain soft assignments from samples to clusters, what in turn reflects assignments uncertainty (because of the data uncertainty itself)
      over the most appropriate assignment

Contents

- Introduction
- Membership-based clustering
- Possibilistic clustering
- Supplementary material: Mixture models

Possibilistic clustering

- The formulation for the **possibilistic clustering** (PC) problem is similar to FC:

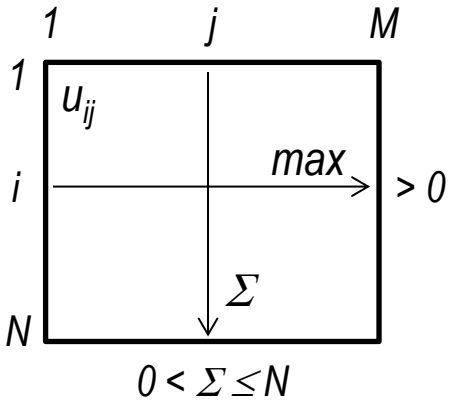
$$\min / \max\ J(X;U,\theta) = \sum_{i=1}^{N}\sum_{j=1}^{M} u_{ij}^{q}\wp(x_i,\theta_j)$$

$$\text{subject to } \max_{j=1,\ldots,M} u_{ij} > 0,\ i = 1,\ldots,N$$

$$\text{where } u_{ij} \in [0,1],\ i = 1,\ldots,N,\ j = 1,\ldots,M$$

$$\text{and } 0 \le \sum_{i=1}^{N} u_{ij} \le N,\ j = 1,\ldots,M$$

$s.t. \sum_{j=1}^{M} u_{ij} \le 1 \rightarrow$ makes u_{ij} depend on each other



where:

- $\wp$ is a **DM** for possibilistic clustering algorithms
- u_{ij} now represents the **grade of compatibility** of sample x_i with cluster C_j or the **possibility** that sample x_i belongs to cluster C_j
  - u_{ij} is no longer coupled to u_{ik}, $k \neq j$, because the **constraints** have changed
  - the possibility that x_i belongs to C_j depends on exclusively x_i and θ_j
  - it is thus independent of the possibilities that x_i belongs to any other cluster
- **q does not play the same role** as in FC, we will clarify this later
- PC behaves better than FC for **noisy datasets** and depends less on a **good election of M**

Possibilistic clustering

- The direct optimization of the previous cost function leads to the **trivial zero-solution**
- In order to avoid that situation, we have to incorporate an **additional term** into J:

$$J(X;U,\theta)=\sum_{i=1}^{N}\sum_{j=1}^{M}u_{ij}^{q}\wp(x_i,\theta_j)+\sum_{j=1}^{M}\eta_j\sum_{i=1}^{N}(1-u_{ij})^{q}=\sum_{j=1}^{M}\left(\sum_{i=1}^{N}u_{ij}^{q}\wp(x_i,\theta_j)+\eta_j\sum_{i=1}^{N}(1-u_{ij})^{q}\right)$$

  - The additional term **depends on u_{ij} to avoid the zero-solution** and also to reduce the effect of outliers
  - η_j are **suitably chosen positive constants**, which will be discussed later
- To find the **optimum** $U=\{u_{ij}\}$ we calculate the partial derivative with respect to u_{ij} and equal it to 0: $\frac{\partial J(X;U,\theta)}{\partial u_{ij}}=qu_{ij}^{q-1}\wp(x_i,\theta_j)-q\eta_j(1-u_{ij})^{q-1}=0$

$$\Rightarrow\left(\frac{u_{ij}}{1-u_{ij}}\right)^{q-1}=\frac{\eta_j}{\wp(x_i,\theta_j)}\Rightarrow u_{ij}=\frac{1}{1+\left(\frac{\wp(x_i,\theta_j)}{\eta_j}\right)^{\frac{1}{q-1}}}$$

  - u_{ij} is inversely dependent on $\wp(x_i,\theta_j)$, what in effect reduces the influence of outliers
- Since the second term of J does not involve the cluster representatives θ_j, one may conclude that the **updating of θ_j is carried out the same way as for FC**.

- **Generalized Clustering Possibilistic Algorithmic Scheme** (GCPAS):

1. choose η_j and $\theta_j(0)$ as initial estimates for $\theta_j, j = 1, \ldots, M$
2. $t = 0$

repeat

3.1 **for** i = 1 **to** N
for j = 1 **to** M
$u_{ij}(t) = \frac{1}{1+\left(\frac{\wp(x_i,\theta_j(t))}{\eta_j}\right)^{\frac{1}{q-1}}}$
end
end

3.2 **for** j = 1 **to** M
solve for $\theta(j+1)$ in $\sum_{i=1}^{N} u_{ij}^q(t)\frac{\partial\wp(x_i,\theta_j)}{\partial\theta_j} = 0$
end

4. $t = t + 1$

until a termination criterion is met, e.g. $\|\theta(t) - \theta(t-1)\| < \epsilon$

- the algorithm can also be started from $U(0)$ instead of $\theta(0)$
- there are no longer issues with $\wp(x_i, q_j(t)) = 0$
- all FC algorithms can be transformed into their PC counterparts: PCM, PGK, APCS, PCES

- **Possibilistic c-Means** clustering algorithm (PCM):

1. choose η_j and $\theta_j(0)$ as initial estimates for $\theta_j, j = 1, \ldots, M$
2. $t = 0$

repeat

3.1 **for** i $= 1$ **to** N
for j $= 1$ **to** M

$$u_{ij}(t) = \frac{1}{1+\left(\frac{d_2(x_i,\theta_j)^2}{\eta_j}\right)^{\frac{1}{q-1}}}$$

end
end

3.2 **for** j $= 1$ **to** M

$$\theta_j(t+1) = \frac{\sum_{i=1}^{N} u_{ij}^q(t) x_i}{\sum_{i=1}^{N} u_{ij}^q(t)}$$

end

4. $t = t + 1$

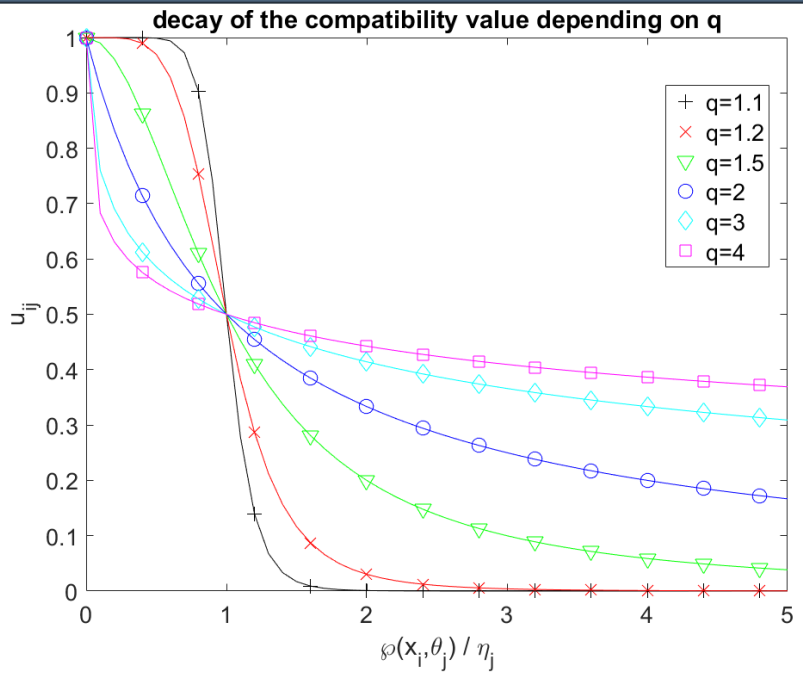
until a termination criterion is met, e.g. $\|\theta(t) - \theta(t-1)\| < \epsilon$

Possibilistic clustering

- Role of **q**:
  - **q** determines the kind of "compatibility" function implemented by the algorithm

$$u_{ij} = \frac{1}{1 + \left(\frac{\wp(x_i, \theta_j)}{\eta_j}\right)^{\frac{1}{q-1}}}$$

  - $q = 1 \Rightarrow u_{ij} = 0, \forall x_i \mid \wp(x_i, \theta_j) > \eta_j$
  - $q \to +\infty \Rightarrow u_{ij}$ tends to constant $0.5, \forall x_i$



- Role of η_j:
  - η_j determines the dissimilarity level between x_i and C_j at which $u_{ij} = 0.5$
    $\Rightarrow$ defines the "size" and "shape" of the cluster
    $\Rightarrow$ defines the influence of a specific point x_i on the estimation of the C_j representative
  - one way to estimate the value of η_j is to run FCM and after its convergence

$$\eta_j = \frac{\sum_{i=1}^{N} u_{ij}^q \wp(x_i, \theta_j)}{\sum_{i=1}^{N} u_{ij}^q} \quad \text{or} \quad \eta_j = \frac{\sum_{u_{ij} > \tau} u_{ij}^q \wp(x_i, \theta_j)}{\sum_{u_{ij} > \tau} 1}, \text{ for an adequate threshold } \tau$$

- PCM as **mode-seeking algorithm**:

$$J(X;U,\theta)=\sum_{j=1}^{M}\left(\sum_{i=1}^{N}u_{ij}^{q}\wp(x_i,\theta_j)+\eta_j\sum_{i=1}^{N}(1-u_{ij})^{q}\right)=\sum_{j=1}^{M}J_j(X;u_j,\theta_j)$$

$$u_{ij}=\frac{1}{1+\left(\frac{\wp(x_i,\theta_j)}{\eta_j}\right)^{\frac{1}{q-1}}}\Rightarrow\wp(x_i,\theta_j)=\eta_j\left(\frac{1-u_{ij}}{u_{ij}}\right)^{q-1}$$

$$\begin{aligned}J_j&=\sum_{i=1}^{N}u_{ij}^{q}\eta_j\left(\frac{1-u_{ij}}{u_{ij}}\right)^{q-1}+\eta_j\sum_{i=1}^{N}(1-u_{ij})^{q}\\&=\eta_j\sum_{i=1}^{N}u_{ij}(1-u_{ij})^{q-1}+\eta_j\sum_{i=1}^{N}(1-u_{ij})(1-u_{ij})^{q-1}\\&=\eta_j\sum_{i=1}^{N}(1-u_{ij})^{q-1}\end{aligned}$$

$\Rightarrow$ minimization of J requires maximizing u_{ij}, which, in turn, requires minimization of $\wp(x_i,q_j)$

$\Rightarrow$ if we run PCM for M clusters but X comprises $K<M$ natural clusters, some of the M clusters will coincide with others, and hence the number of clusters in X need not be known a priori

$\Rightarrow$ C_j are finally placed in **regions dense in samples**, i.e. the modes of the dataset

Possibilistic clustering

- Alternative GCPAS:
  - An alternative possibilistic scheme can be derived from a new cost function:

$$J(X;U,\theta) = \sum_{i=1}^{N}\sum_{j=1}^{M} u_{ij}\wp(x_i,\theta_j) + \sum_{j=1}^{M}\eta_j\sum_{i=1}^{N}(u_{ij}\ln u_{ij} - u_{ij})$$

$$= \sum_{j=1}^{M}\left(\sum_{i=1}^{N} u_{ij}\wp(x_i,\theta_j) + \eta_j\sum_{i=1}^{N}(u_{ij}\ln u_{ij} - u_{ij})\right)$$

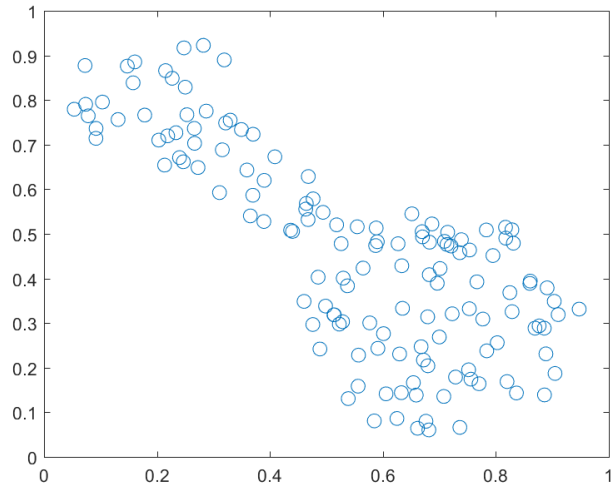
    - q **is no longer involved**
  - It can be shown that, for this case:

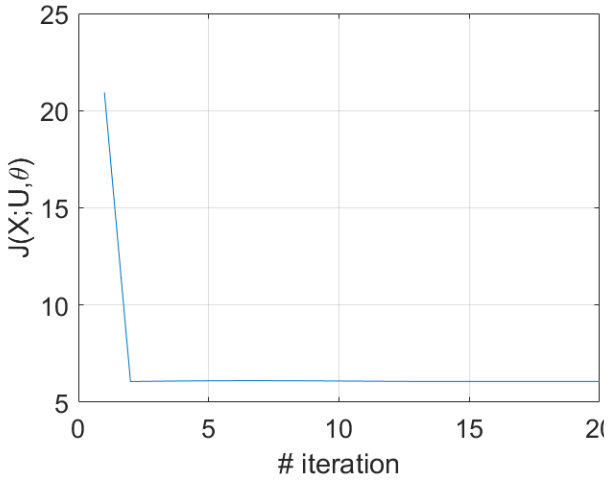
$$u_{ij} = e^{-\frac{\wp(x_i,\theta_j)}{\eta_j}}$$

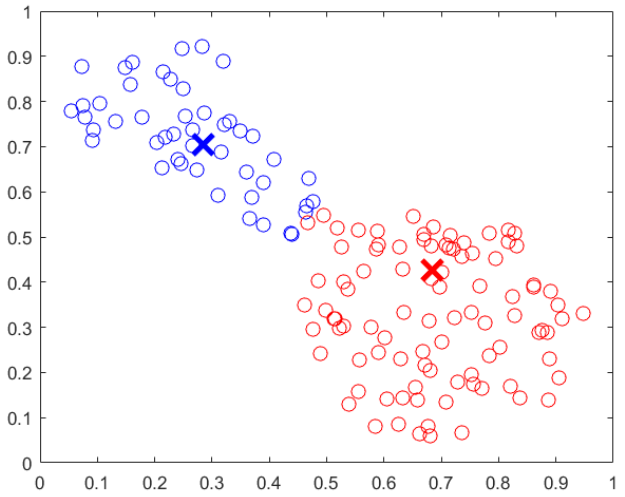
    - u_{ij} decreases more rapidly than in the standard scheme

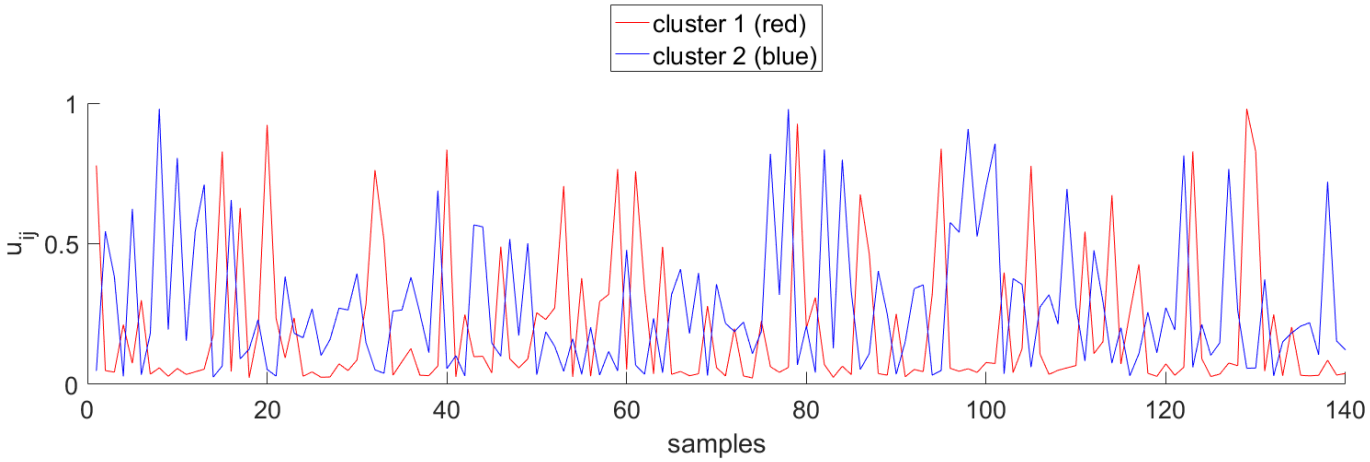
Possibilistic clustering

- Example (standard PCM): N = 140 samples, M = 2 clusters, q = 2, η_j set by FCM









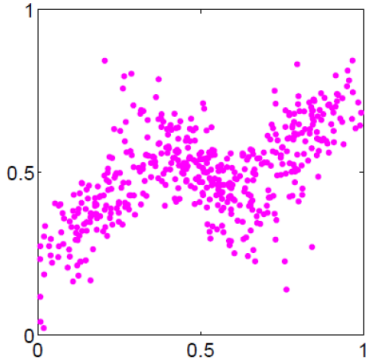
Contents

- Introduction
- Membership-based clustering
- Possibilistic clustering
- Supplementary material: Mixture models

Mixture models

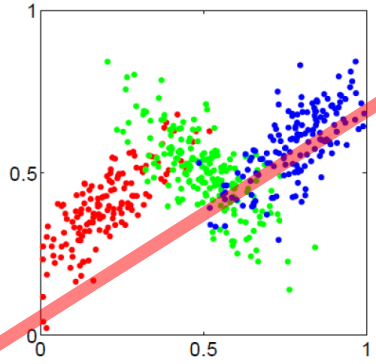
- These algorithms assume that there are M clusters underlying the dataset, each coming from a population obeying a certain probability distribution, so that the goal is to discover the parameters of the mixture. They are also termed as **soft clustering**.

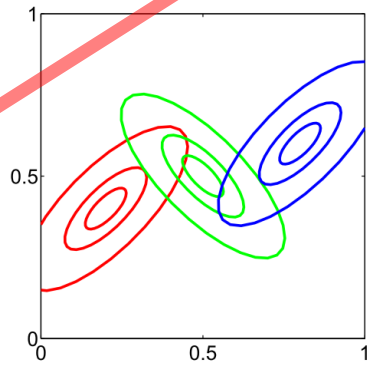
Example: We have a dataset such as:

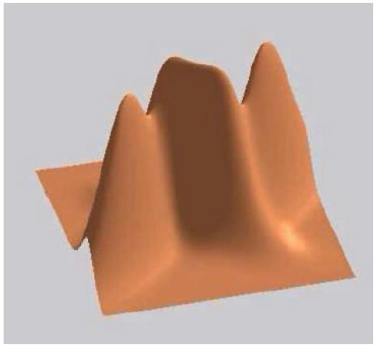


We guess there are M = 3 clusters normally distributed …

… and, in effect, there are 3 clusters, everyone coming from a different Gaussian:







- We intend to discover the parameters of the three Gaussians (μ_i, Σ_i) and the assignments of samples to clusters.

Mixture models

- We adopt an optimization-based approach which assigns each sample x_i to the cluster with **highest likelihood**:

$$\max \prod_{i=1}^{N} p(x_i|\theta), \quad \theta = \{\theta_j\}$$

and we model the probability of each sample introducing weights π_j:

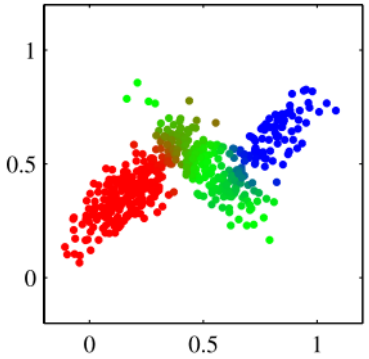
$$p(x_i|\theta) = \sum_{j=1}^{M} \pi_j p(x_i|\theta_j)$$

since, a priori, **we do not know which sample belongs to which cluster**, where:

$$\pi_j = \sum_{i=1}^{N} r_{ij}$$

so that: $j = \arg\max_k \{r_{ik}\} \Rightarrow$ assign sample x_i to cluster C_j

- Therefore, the output of the optimization process is the parameters θ_j of the clusters and the weights r_{ij}
  - e.g. for the previous example, we can plot samples x_i using a colour code proportional to the corresponding r_{ij} →



- Adopting a **log-likelihood approach** and assuming **i.i.d samples**, J is defined as:

$$J(X;\pi,\theta) = \ln p(X) = \ln \prod_{i=1}^{N} p(x_i|\theta) = \ln \prod_{i=1}^{N} \left(\sum_{j=1}^{M} \pi_j p(x_i|\theta_j) \right) = \sum_{i=1}^{N} \ln \left(\sum_{j=1}^{M} \pi_j p(x_i|\theta_j) \right)$$

$$\pi_j = \sum_{i=1}^{N} r_{ij} \text{ and } r_{ij} = p(z_{ij} = 1|x_i)$$

$$z_{ij} = 1 \text{ if } x_i \text{ comes from cluster } C_j, \text{ with } z_{ij} = 0 \text{ for } k \neq j$$

- The **maximum likelihood** optimization problem hence becomes:

$$\max J(X;\pi,\theta)$$
$$\text{subject to } 0 \leq \pi_j \leq 1, \quad j = 1,\ldots,M$$
$$\sum_{j=1}^{M} \pi_j = 1$$

  - The elements of the N × M matrix **Z** = {z_{ij}} are known as **latent variables** in this problem.
  - r_{ij} is named as the **responsibility** that cluster C_j takes for explaining x_i.
  - π_j are the **mixing coefficients**, which somehow can be seen as the amount of explanation of the data performed by cluster C_j.

- The mixture problem can be solved by means of the **Expectation-Maximization** (EM) algorithm.
  - EM is an iterative algorithm that comprises two steps:
    - The **E step**, where the latent variables z_{ij} are re-estimated (keeping constant the parameters of the clusters)
    - The **M step**, where the distribution parameters are re-estimated (keeping constant the latent variables z_{ij})
  - EM is useful to solve other problems apart from clustering on the basis of a mixture model
- The most widely used mixture model is the **mixture of Gaussians** (MOG), also called a **Gaussian mixture model** (GMM):

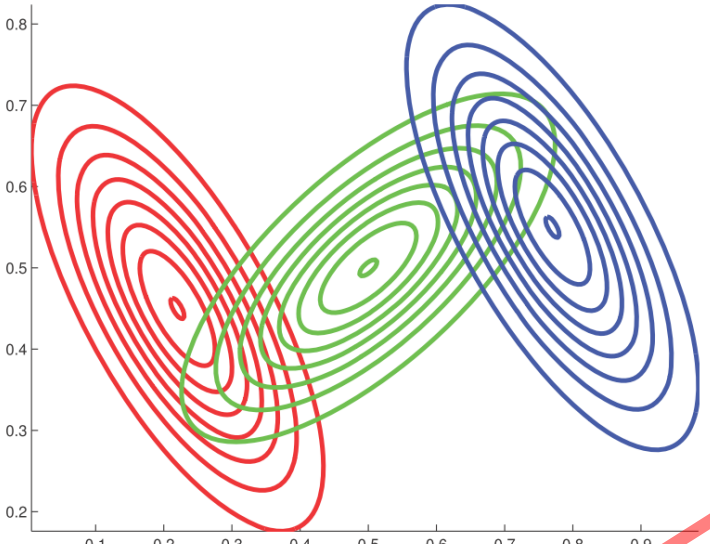
$$J(X;\pi,\theta) = \sum_{i=1}^{N} \ln \left(\sum_{j=1}^{M} \pi_j p(x_i|\theta_j) \right) = \sum_{i=1}^{N} \ln \left(\sum_{j=1}^{M} \pi_j \mathcal{N}(x_i|\mu_j,\Sigma_j) \right)$$

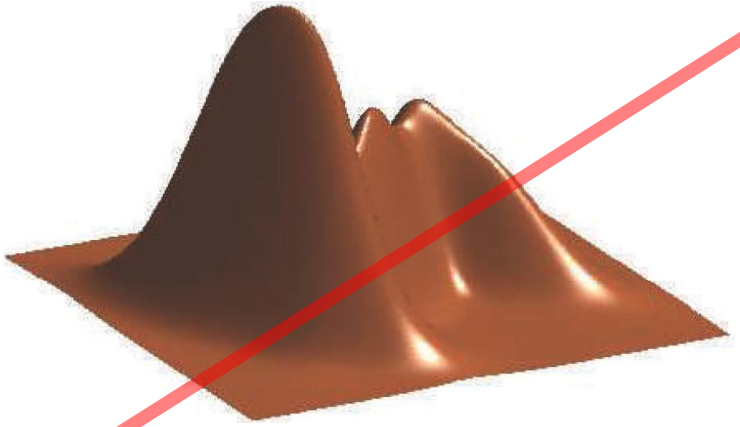
$$\mathcal{N}(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \mathcal{N}(x|\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^L|\Sigma_j|}} \exp\left(-\frac{1}{2}(x-\mu_j)^T\Sigma_j^{-1}(x-\mu_j)\right)$$

- Because of the shape of a Gaussian pdf, this algorithm is useful for detecting **compact and hyperellipsoidal clusters**

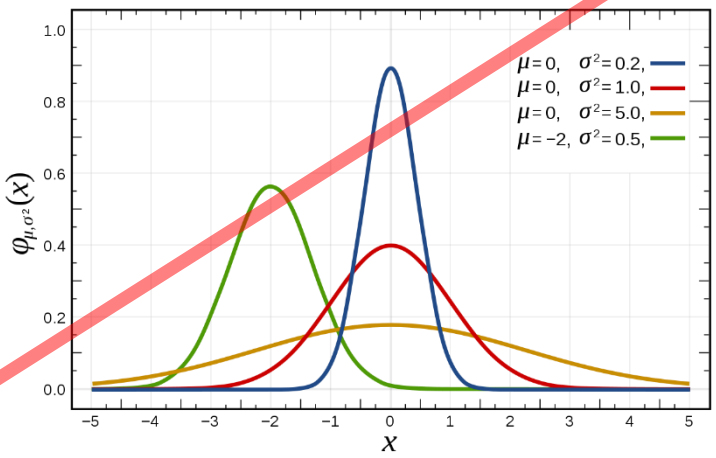
Mixture models

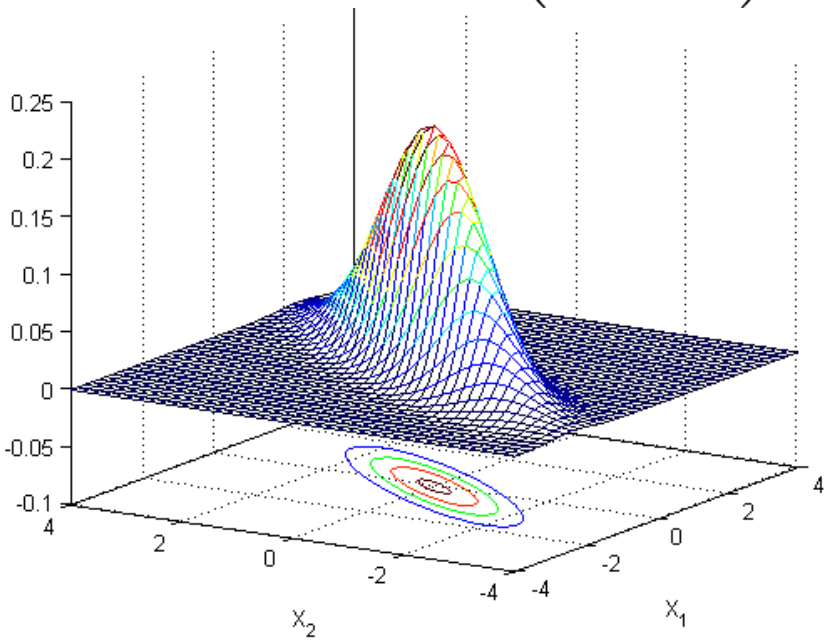
- Another example of Mixture of Gaussians:





$$\mu = (0,0) \quad \Sigma = \begin{pmatrix} 1 & 1.2 \\ 1.2 & 2 \end{pmatrix}$$





- **Algorithm EM for Gaussian mixtures**:

1. choose $\mu_j(0)$ and $\Sigma_j(0)$ as initial estimates for $\theta_j, j = 1, \ldots, M$
2. choose the mixing coefficients π_j
3. $t = 0$

repeat

E-step: estimate r_{ij} keeping θ_j and π_j at their previous values

4.1 **for** i = 1 **to** N
for j = 1 **to** M
$r_{ij}(t) = \frac{\pi_j(t)\mathcal{N}(x_i;\mu_j(t),\Sigma_j(t))}{\sum_{k=1}^{M}\pi_k(t)\mathcal{N}(x_i;\mu_k(t),\Sigma_k(t))}$
end
end

M-step: estimate θ_j and π_j keeping r_{ij} at their previous values

4.2 **for** j = 1 **to** M
$N_j(t+1) = \sum_{i=1}^{N} r_{ij}(t)$
$\mu_j(t+1) = \frac{1}{N_j(t+1)}\sum_{i=1}^{N} r_{ij}(t)x_i$
$\Sigma_j(t+1) = \frac{1}{N_j(t+1)}\sum_{i=1}^{N} r_{ij}(x_i - \mu_j(t+1))(x_i - \mu_j(t+1))^T$
$\pi_j(t+1) = \frac{N_j(t+1)}{N}$
end

4.3 $t = t + 1$

until a termination criterion is met

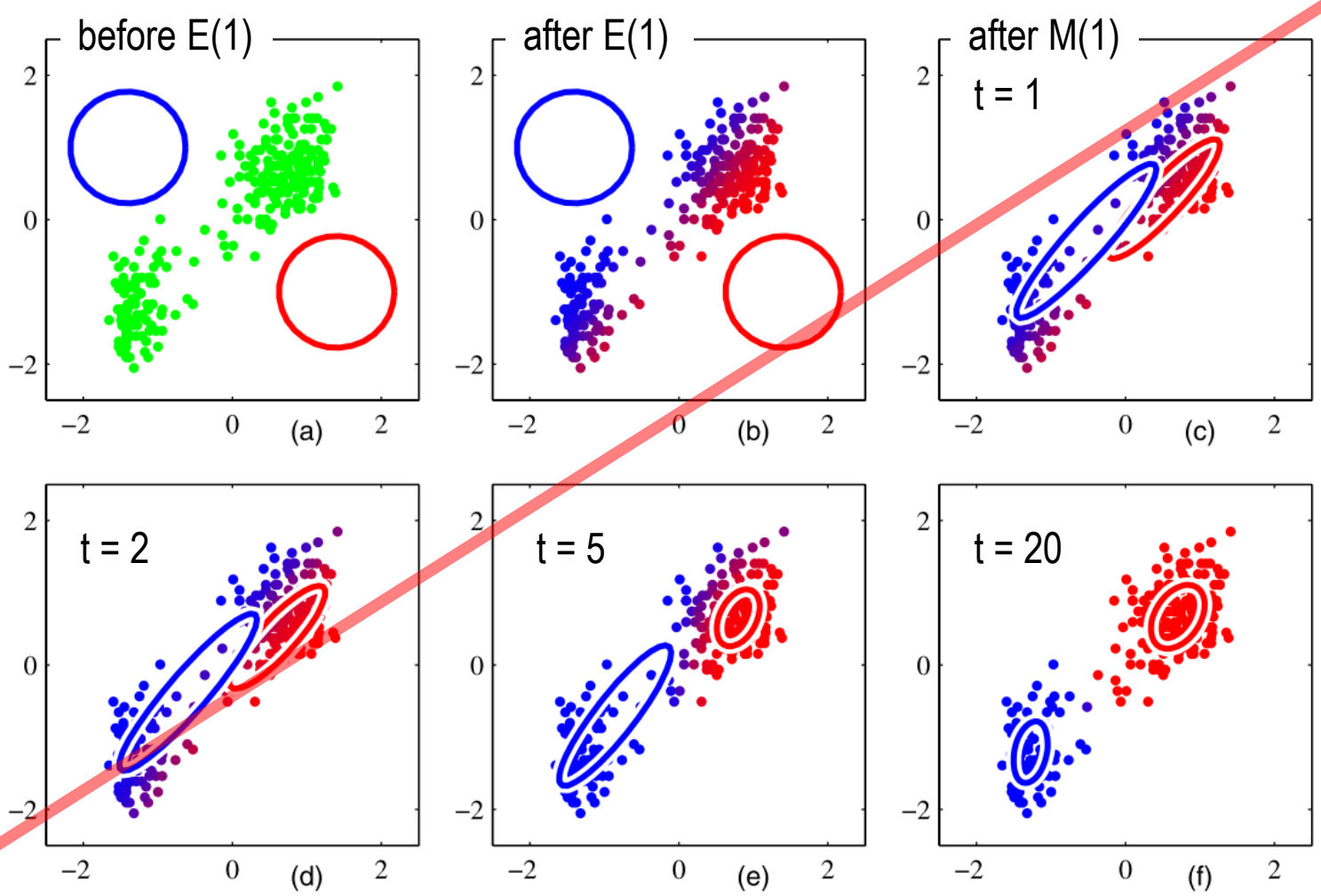
4.4 Evaluate the log-likelihood:

$$J(t) = \sum_{i=1}^{N} \ln\left(\sum_{j=1}^{M}\pi_j(t)\mathcal{N}(x_i|\mu_j(t),\Sigma_j(t))\right)$$

and check for convergence of either the parameters or the log-likelihood

termination criterion: $J(t) - J(t-1) < \epsilon$

- **Example**: rescaled Old Faithful data set, M = 2



- **Final remarks**:
  - Remember that, once the mixture model has been fitted, we can make use of

$$r_{ij} = p(C_j|x_i) > p(C_k|x_i) = r_{ik}, k \neq j \Rightarrow \text{ assign sample } x_i \text{ into } C_j$$

    for "hard" sample allocation to clusters
  - The E-step can be computationally demanding because of the calculations of Σ_j^{-1}.
    - One way to relax this is to assume that all covariance matrices are diagonal or that they are all equal to each other (only one inversion is required at each iteration step).
  - The EM-GMM algorithm reduces to K-means when $\Sigma_j = \sigma^{\wedge 2} I$ and $\pi_j = 1/M$.

Unsupervised Learning: Optimization-based clustering

Universitat de les Illes Balears
Departament de Ciències Matemàtiques i Informàtica

11752 Aprendizaje Automático
11752 Machine Learning
Máster Universitario
en Sistemas Inteligentes

Alberto ORTIZ RODRÍGUEZ