Clustering

Course of Machine Learning
Master Degree in Computer Science
University of Rome "Tor Vergata"
a.a. 2024-2025

Giorgio Gambosi

Clustering types

- Partitional clustering: Given a set of items (points) $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, we wish to partition $\mathbf{X}$ by assigning each element to one out of k clusters $C_1, \ldots, C_k$ in such a way to maximize (or minimize) a given cost J. The number k of clusters could be given or should have to be computed.

- Hierarchical clustering: Given a set of items (points) $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, we wish to derive a set of nested partitions of $\mathbf{X}$, from the partition composed by all singletons (one cluster for each node) to the one composed by a single item (the whole set).

Partitional clustering

Given a dataset $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, with $\mathbf{x}_i \in \mathbb{R}^d (i = 1, \ldots, n)$.

We wish to derive a **clustering**, that is a partition of $\mathbf{X}$ into subsets of similar elements (that is elements which are "near" according to a predefined distance measure) called **clusters**.

A clustering is represented as follows:

1. Each cluster is represented by a point in feature space denoted as **prototype**. The clustering is then represented by the set of cluster prototypes $(\mathbf{m}_1, \ldots, \mathbf{m}_k)$, with $\mathbf{m}_j \in \mathbb{R}^d (j = 1, \ldots, k)$

2. Each element is assigned to exactly one cluster. In particular, we assume that the assignment of each item $\mathbf{x}_i$, is encoded by means of a 1-to-k encoding. That is, as a vector of k binary flags $r_{ij} \in \{0, 1\}, j = 1, \ldots, k$. If $\mathbf{x}_i$ is assigned the t-th cluster, then $r_{it} = 1$ and $r_{ij} = 0$ for $j \neq t$

A cost function is defined which associates a cost value to each clustering, and has to be minimized by the clustering algorithm. The usual cost function is the sum of squared element-prototype distances:

$$J(R, M) = \sum_{i=1}^{k} \sum_{j=1}^{n} r_{ij} ||\mathbf{x}_j - \mathbf{m}_i||^2 = \sum_{i=1}^{k} \sum_{j=1}^{n} r_{ij} (\mathbf{x}_j - \mathbf{m}_i)^T (\mathbf{x}_j - \mathbf{m}_i)$$

where

- $R_{ij} = r_{ij}$, where $r_{is} = 1$ and $r_{ij} = 0$ for $j \neq s$ if $\mathbf{x}_i$ is assigned to cluster C_s

- $M_i = \mathbf{m}_i$, $i = 1, \ldots, k$ is the prototype of cluster C_i, which is usually the centroid of elements assigned to the cluster

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{j=1}^{n} r_{ij} \mathbf{x}_j$$

k-means clustering

Given a dataset $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$, $\mathbf{x}_i \in \mathbb{R}^d$: we wish to derive k clusters with prototypes $\mathbf{m}_1, \dots, \mathbf{m}_k$ and, for each $\mathbf{x}_i$ a vector of assignments to clusters $r_{i1}, \dots, r_{ik}$ such that the cost

$$J(R, M) = \sum_{i=1}^{n} \sum_{j=1}^{k} r_{ij} \, ||\mathbf{x}_i - \mathbf{m}_j||^2$$

is minimum.

Algorithm

The algorithm iterates on a sequence of steps, where at each step the current clustering is updated in two phases:

1. Given the current set of prototypes $\mathbf{m}_{ij}$, minimize the cost wrt r_{ij} (that is by deriving the best assignment of elements to clusters). This is obtained by minimizing, for each $\mathbf{x}_i$, the value $\sum_{j=1}^{k} r_{ij} \, ||\mathbf{x}_i - \mathbf{m}_j||^2$.

   The minimum is obtained for $r_{ik} = 1$ (and $r_{ij} = 0$ for $j \neq k$), where $||\mathbf{x}_i - \mathbf{m}_k||^2$ is the minimum distance. That is, each point is assigned to the cluster represented by the nearest prototype.

2. Given the current set of assignments r_{ij}, minimize the cost wrt $\mathbf{m}_{ij}$ (that is by defining the best cluster prototypes).

   For each cluster C_k, the cost function $J = \sum_{i=1}^{n} \sum_{j=1}^{k} r_{ij} \, ||\mathbf{x}_i - \mathbf{m}_j||^2$ is a quadratic function wrt $\mathbf{m}_k$. By setting its derivative to zero, the values of $\mathbf{m}_k$ providing its minimum are obtained

$$\frac{\partial J}{\partial \mathbf{m}_k} = 2 \sum_{i=1}^{n} r_{ik}(\mathbf{x}_i - \mathbf{m}_k) = 0 \Rightarrow \mathbf{m}_k = \frac{\sum_{i=1}^{n} r_{ik} \mathbf{x}_i}{\sum_{i=1}^{n} r_{ik}}$$

   That is, the new prototype is the mean of the elements assigned to the cluster

It is easy to check that, at each step, J does not increase. Hence, there is a convergence to a local minimum. However, such convergence can be slow, and a stopping rule mu st be defined, for example by checking cost decrease at each step.

Example of application of k-means

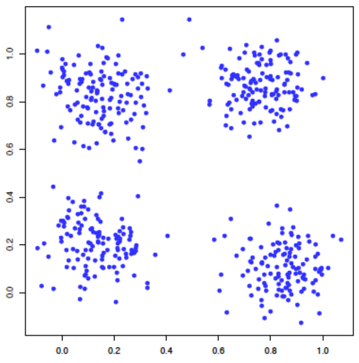


Figure 1: Dataset

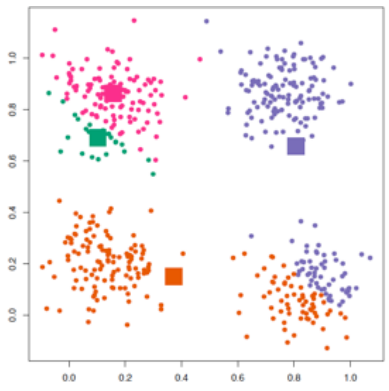


Figure 2: Initial clustering

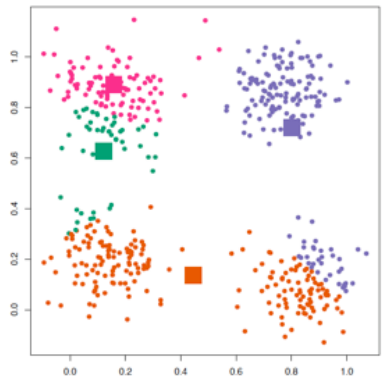
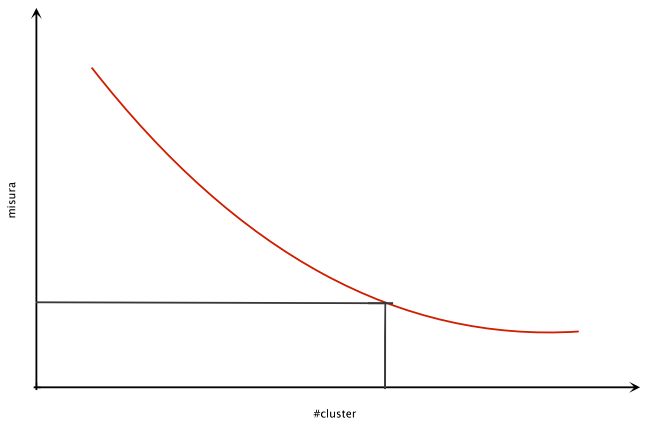


Figure 3: After one step

The algorithm is applied for a given number k of clusters to identify. An important issue is choosing the best value for this parameter. The simplest way to do that is can be done by checking different values, applying the algorithm for different values of k and measuring the associated quality as the cost value for the clustering obtained.

This measure improves as K increases (overfitting). A value such that further increases provide limited improvement should be found



Hierarchical clustering

Aim

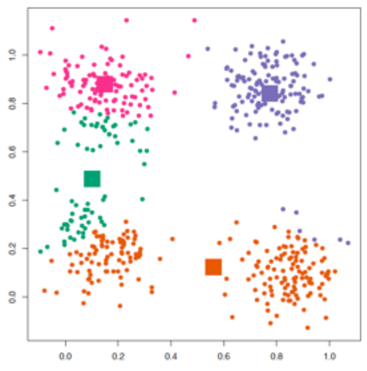


Figure 4: After two steps

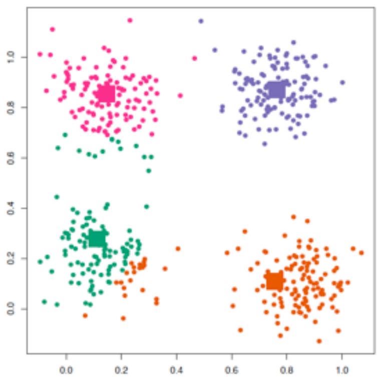


Figure 5: After three steps

Derivation of a binary tree. Node: cluster; arc: inclusion.

The tree specifies a set of pairwise merge of clusters.

- Aggregation, starting from n singleton clusters
- Separation, starting from a single cluster of size n

Requirements: k-means requires:

- a number K of clusters
- an initial assignment
- a distance function between elements

Hierarchical clustering requires:

- a similarity function between clusters

Algorithm

- define n clusters (singleton)
- repeat
  - compute the matrix of distances between clusters

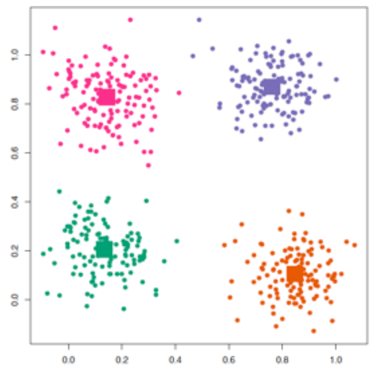


Figure 6: After four steps

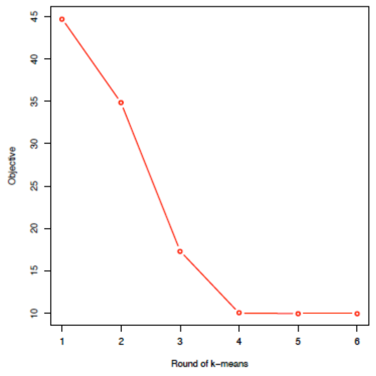


Figure 7: Cost values

  - merge the pair of clusters which are "nearest"

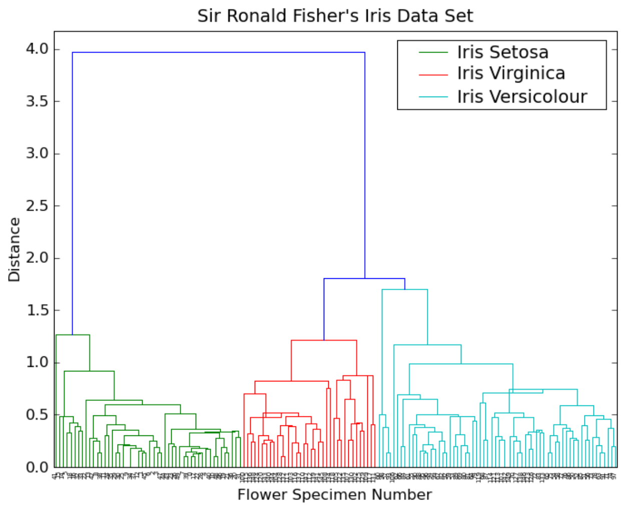
- until a single cluster has remained

Properties

- Each tree prefix is a partition of elements
- The algorithm provides a partial order of clusterings
- The best clustering has to be found
- Monotonicity: similarity between paired clusters decreases

Dendrogram

- Tree of cluster pairings
- The height of the nodes is inversely proportional to the similarity of the paired clusters



Cluster similarity

Many measures. Most frequent ones:

- Similarity between nearest nodes (**Single linkage**)

$$d_{SL}(C_1, C_2) = \min_{\mathbf{x}_1 \in C_1, \mathbf{x}_2 \in C_2} d(\mathbf{x}_i, \mathbf{x}_j)$$

- Similarity between farthest nodes (**Complete linkage**)

$$d_{CL}(C_1, C_2) = \max_{\mathbf{x}_1 \in C_1, \mathbf{x}_2 \in C_2} d(\mathbf{x}_i, \mathbf{x}_j)$$

- Mean similarity (**Group average**)

$$d_{GA}(C_1, C_2) = \frac{1}{|C_1| \cdot |C_2|} \sum_{\mathbf{x}_1 \in C_1} \sum_{\mathbf{x}_2 \in C_2} d(\mathbf{x}_i, \mathbf{x}_j)$$

Different measures provide different dendrograms

A probabilistic approach to clustering: mixtures of distributions

1 Mixtures

A probabilistic approach to clustering is based on the fundamental task of modeling the distribution of elements in a parametric way $p(\mathbf{x}; \boldsymbol{\theta})$ and inferring from the dataset $\mathbf{X}$ the best values of the parameter $\boldsymbol{\theta}$. This is equivalent to what is done in the probabilistic approach to supervised learning, where the same is done with respect to the conditional distribution $p(t|\mathbf{x}; \boldsymbol{\theta})$.

The structure of the distribution $p(\mathbf{x}; \boldsymbol{\theta})$ in the case of clustering considered here is such that the values inferred for the parameter $\boldsymbol{\theta}$ can be interpreted in terms of (a smooth) clustering. Such structure is a **mixture of distributions**, defined as follows:

$$p(\mathbf{x}; \boldsymbol{\pi}, \boldsymbol{\Theta}) = \sum_{i=1}^{K} \pi_i q(\mathbf{x}; \boldsymbol{\theta}_i)$$

The set of parameters is composed of

- the set of mixture weights $\boldsymbol{\pi} = \{\pi_1, \dots, \pi_K\}$ and
- the set of parameters of each mixture element $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_K\}$

The distributions can be from different families, for example from beta and normal distributions. However, this makes the problem very complex and sometimes useless; therefore, usually the distributions in a mixture are from one family (e.g., all normal distributions) but with different parameters.

Linear combinations of probability distributions

- Same type of distributions $q(\mathbf{x}|\boldsymbol{\theta})$
- Differ by parameter values

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

where

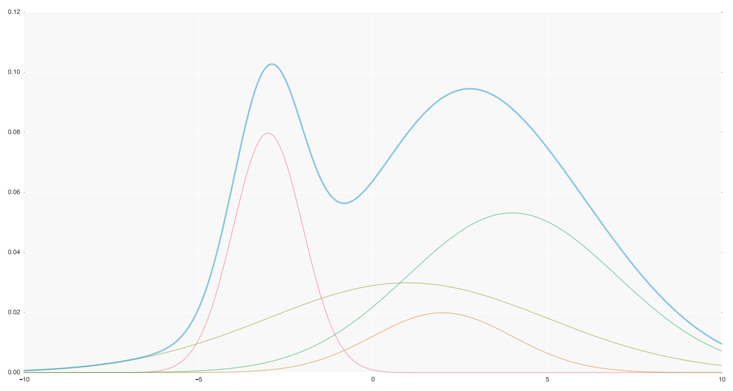
$$\boldsymbol{\pi} = (\pi_1, \dots, \pi_K) \qquad \boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_K)$$

Mixing coefficients

$$0 \le \pi_k \le 1 \quad k = 1, \dots, K \qquad \sum_{k=1}^{K} \pi_k = 1$$

Terms π_k have the properties of probability values

Provide extensive capabilities to model complex distributions. For example, almost all continuous distributions can be modeled by the linear combination of a suitable number of gaussians.



Mixture parameters estimation

Given a dataset $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$, the parameters $\boldsymbol{\pi}, \boldsymbol{\Theta}$ of a mixture can be estimated by maximum likelihood.

$$L(\boldsymbol{\Theta}, \boldsymbol{\pi}|\mathrm{X}) = p(\mathrm{X}|\boldsymbol{\Theta}, \boldsymbol{\pi}) = \prod_{i=1}^{n} p(\mathbf{x}_i|\boldsymbol{\Theta}, \boldsymbol{\pi}) = \prod_{i=1}^{n}\sum_{k=1}^{K} \pi_k q(\mathbf{x}|\boldsymbol{\theta}_k)$$

or maximum log-likelihood

$$l(\boldsymbol{\Theta}, \boldsymbol{\pi}|\mathrm{X}) = \log p(\mathrm{X}|\boldsymbol{\Theta}, \boldsymbol{\pi}) = \sum_{i=1}^{n} \log p(\mathbf{x}_i|\boldsymbol{\Theta}, \boldsymbol{\pi}) = \sum_{i=1}^{n} \log\left(\sum_{k=1}^{K} \pi_k q(\mathbf{x}_i|\boldsymbol{\theta}_k)\right)$$

Maximization is however constrained by the conditions $0 \leq \pi_i \leq 1$ for all i and $\sum_{i=1}^{K} \pi_i = 1$.

By applying the lagrangian multipliers method, we will maximize

$$\mathcal{L}(\boldsymbol{\Theta}, \boldsymbol{\pi}, \lambda) = l(\boldsymbol{\Theta}, \boldsymbol{\pi}|\mathrm{X}) + \lambda(1 - \sum_{i=1}^{K} \pi_i)$$

Let us first consider the derivatives with respect to the weights $\boldsymbol{\pi}$, which we set to 0

$$\frac{\partial \mathcal{L}(\boldsymbol{\Theta}, \boldsymbol{\pi}|\mathrm{X})}{\partial \pi_j} = \frac{\partial l(\boldsymbol{\Theta}, \boldsymbol{\pi}|\mathrm{X})}{\partial \pi_j} - \lambda = 0$$

This is equivalent to

$$\lambda = \frac{\partial l(\boldsymbol{\Theta}, \boldsymbol{\pi}|\mathrm{X})}{\partial \pi_j} = \frac{\partial}{\partial \pi_j}\left[\sum_{i=1}^{n} \log\left(\sum_{k=1}^{K} \pi_k q(\mathbf{x}_i|\boldsymbol{\theta}_k)\right)\right] = \sum_{i=1}^{n} \frac{\partial}{\partial \pi_j}\left[\log\left(\sum_{k=1}^{K} \pi_k q(\mathbf{x}_i|\boldsymbol{\theta}_k)\right)\right]$$
$$= \sum_{i=1}^{n} \frac{q(\mathbf{x}_i|\boldsymbol{\theta}_j)}{\sum_{k=1}^{K} \pi_k q(\mathbf{x}_i|\boldsymbol{\theta}_k)} = \sum_{i=1}^{n} \frac{\gamma_j(\mathbf{x}_i)}{\pi_j} = \frac{1}{\pi_j}\sum_{i=1}^{n} \gamma_j(\mathbf{x}_i)$$

where,

$$\gamma_k(\mathbf{x}) = \frac{\pi_k q(\mathbf{x}|\boldsymbol{\theta}_k)}{\sum_{j=1}^{K} \pi_j q(\mathbf{x}|\boldsymbol{\theta}_j)}$$

By setting the derivative wrt λ to 0

$$\frac{\partial \mathcal{L}(\boldsymbol{\Theta}, \boldsymbol{\pi}|\mathrm{X})}{\partial \lambda} = \frac{\partial}{\partial \lambda}\left(l(\boldsymbol{\Theta}, \boldsymbol{\pi}|\mathrm{X}) + \lambda(1 - \sum_{i=1}^{K} \pi_i)\right) = 0$$

we obtain

$$\sum_{i=1}^{K} \pi_i = 1$$

As a consequence, since, as shown above,

$$\pi_j = \frac{1}{\lambda} \sum_{i=1}^{n} \gamma_j(\mathbf{x}_i)$$

it results

$$\sum_{j=1}^{K} \pi_j = \frac{1}{\lambda} \sum_{j=1}^{K} \sum_{i=1}^{n} \gamma_j(\mathbf{x}_i) = 1$$

which implies

$$\lambda = \sum_{j=1}^{K} \sum_{i=1}^{n} \gamma_j(\mathbf{x}_i) = \sum_{i=1}^{n} \sum_{j=1}^{K} \gamma_j(\mathbf{x}_i) = \sum_{i=1}^{n} \sum_{j=1}^{K} \frac{\pi_j q(\mathbf{x}_i|\boldsymbol{\theta}_j)}{\sum_{k=1}^{K} \pi_k q(\mathbf{x}_i|\boldsymbol{\theta}_k)} = \sum_{i=1}^{n} 1 = n$$

and, finally,

$$\pi_k = \frac{1}{n} \sum_{i=1}^{n} \gamma_k(\mathbf{x}_i)$$

For what concerns derivatives (or gradients) wrt distribution parameters $\boldsymbol{\theta}$, it results

$$\begin{aligned}\nabla_{\boldsymbol{\theta}_j} \left[\mathcal{L}(\boldsymbol{\Theta}, \boldsymbol{\pi}|\mathrm{X})\right] &= \nabla_{\boldsymbol{\theta}_j} \left[\sum_{i=1}^{n} \log\left(\sum_{k=1}^{K} \pi_k q(\mathbf{x}_i|\boldsymbol{\theta}_k)\right)\right] = \sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}_j} \left[\log\left(\sum_{k=1}^{K} \pi_k q(\mathbf{x}_i|\boldsymbol{\theta}_k)\right)\right] \\ &= \sum_{i=1}^{n} \gamma_j(\mathbf{x}_i) \nabla_{\boldsymbol{\theta}_j} \left[\log q(\mathbf{x}_i|\boldsymbol{\theta}_j)\right] = \mathbf{0}\end{aligned}$$

The solution of these equations is dependent from the definition of distribution $q(\mathbf{x}|\boldsymbol{\theta})$. However, it will depend also from the values $\gamma_j(\mathbf{x}_i)$ which, as seen above, is dependent from $\boldsymbol{\pi}$ and $\boldsymbol{\Theta}$.

Reassuming, log likelihood maximization is in general intractable analytically: its solution cannot be given in closed form. Also,

- $\boldsymbol{\pi}$ and $\boldsymbol{\Theta}$ can be derived from $\gamma_k(\mathbf{x}_i)$
- Also, $\gamma_k(\mathbf{x}_i)$ can be derived from $\boldsymbol{\pi}$ e $\boldsymbol{\Theta}$

This makes it possible to apply an iterative approach:

- Given an estimation for $\boldsymbol{\pi}$ e $\boldsymbol{\Theta}$...
- derive an estimation for $\gamma_k(\mathbf{x}_i)$, from which ...
- derive a new estimation for $\boldsymbol{\pi}$ e $\boldsymbol{\Theta}$, from which ...
- derive a new estimation for $\gamma_k(\mathbf{x}_i)$ ...

Gaussian mixtures

In this case, the distributions $q(\mathbf{x}|\boldsymbol{\theta})$ are gaussians, hence we have $\boldsymbol{\theta}_k = \{\boldsymbol{\mu}_k, \Sigma_k\}$, the mean and covariance matrix of the k-th gaussian.

$$q(\mathbf{x}; \boldsymbol{\mu}_r, \Sigma_r) \triangleq \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_r, \Sigma_r) = \frac{1}{(2\pi)^{d/2}} \frac{1}{|\Sigma_r|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_r)^T \Sigma_r^{-1} (\mathbf{x}-\boldsymbol{\mu}_r)\right)$$

As shown before, given the current values $\boldsymbol{\pi}^{(k)}$, $\boldsymbol{\Theta}^{(k)}$, the coefficients

$$\begin{aligned}\gamma_j^{(k-1)}(\mathbf{x}_i) &= \frac{\pi_j^{(k-1)} q(\mathbf{x}_i; \boldsymbol{\theta}_j^{(k-1)})}{\sum_{r=1}^K \pi_r^{(k-1)} q(\mathbf{x}_i; \boldsymbol{\theta}_r^{(k-1)})} \\ &= \frac{\pi_j^{(k-1)} \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_j^{(k-1)}, \Sigma_j^{(k-1)})}{\sum_{r=1}^K \pi_r^{(k-1)} \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_r^{(k-1)}, \Sigma_r^{(k-1)})}\end{aligned}$$

are computed.

Moreover, given the values $\gamma_j^{(k-1)}(\mathbf{x}_i)$, new values $\boldsymbol{\pi}^{(k)}$, $\boldsymbol{\mu}^{(k)}$, $\Sigma^{(k)}$ are computed, with

$$\pi_j^{(k)} = \frac{1}{n} \sum_{i=1}^n \gamma_j^{(k-1)}(\mathbf{x}_i)$$

The next value for $\boldsymbol{\mu}_j$ derives in general from the solution of

$$\sum_{i=1}^n \frac{\gamma_j(\mathbf{x}_i)}{\mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_j, \Sigma_j)} \nabla_{\boldsymbol{\mu}_j} \left[\mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_j, \Sigma_j)\right] = 0$$

which is

$$\boldsymbol{\mu}_j = \frac{\sum_{i=1}^n \gamma_j(\mathbf{x}_i)\mathbf{x}_i}{\sum_{i=1}^n \gamma_j(\mathbf{x}_i)}$$

Similarly, the next value for Σ_j derives in general from the solution of

$$\sum_{i=1}^n \frac{\gamma_j(\mathbf{x}_i)}{\mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_j, \Sigma_j)} \nabla_{\Sigma_j} \left[\mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_j, \Sigma_j)\right] = 0$$

which can be proved to be

$$\begin{aligned}\Sigma_j &= \frac{1}{\sum_{i=1}^n \gamma_j(\mathbf{x}_i)} \sum_{i=1}^n \gamma_j(\mathbf{x}_i)(\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j)^T \\ &= \frac{1}{\sum_{i=1}^n \gamma_j(\mathbf{x}_i)} \sum_{i=1}^n \gamma_j(\mathbf{x}_i)\mathbf{x}_i\mathbf{x}_i^T - \boldsymbol{\mu}_j\boldsymbol{\mu}_j^T\end{aligned}$$

Notice that as a result, the iterative algorithm, at each step

1. knowing $\pi_j^{(k)}, \boldsymbol{\mu}_j^{(k)}, \Sigma_j^{(k)}$ for $j = 1, \dots, K$ computes $\gamma_j^{(k)}(\mathbf{x}_i)$ for $j = 1, \dots, K$ and $i = 1, \dots, n$
2. also, knowing $\gamma_j^{(k)}(\mathbf{x}_i)$ for $j = 1, \dots, K$ and $i = 1, \dots, n$ computes $\pi_j^{(k+1)}$, $\boldsymbol{\mu}_j^{(k+1)}$, $\Sigma_j^{(k+1)}$

Mixtures of Poissons

In the case of a mixture of K Poisson distributions

$$q(x;\boldsymbol{\theta}_k) = q(x;\lambda_k) = \frac{e^{-\lambda_k}\lambda_k^x}{x!},$$

As a consequence, it results

$$\gamma_j^{(k-1)}(x_i) = \frac{\pi_j^{(k)}\frac{e^{-\lambda_j^{(k)}}\lambda_j^{(k)x_i}}{x_i!}}{\sum_{r=1}^K \pi_r^{(k)}\frac{e^{-\lambda_r^{(k)}}\lambda_r^{(k)x_i}}{x_i!}}.$$

For what regards the maximization step, the new values $\boldsymbol{\pi}^{(k)}$ are still given by

$$\pi_j^{(k)} = \frac{1}{n}\sum_{i=1}^n \gamma_j^{(k-1)}(\mathbf{x}_i)$$

while the new values $\lambda_j^{(k)}$ derive by setting

$$\begin{aligned}
0 &= \sum_{i=1}^n \gamma_j^{(k-1)}(x_i)\frac{\partial}{\partial\lambda_j}\log q(x_i;\lambda_j)\\
&= \sum_{i=1}^n \gamma_j^{(k-1)}(x_i)\frac{\partial}{\partial\lambda_j}(-\lambda_j + x_i\log\lambda_j - \log x_i!)\\
&= \sum_{i=1}^n \gamma_j^{(k-1)}(x_i)\left(-1+\frac{x_i}{\lambda_j}\right)\\
&= -\sum_{i=1}^n \gamma_j^{(k-1)}(x_i) + \frac{1}{\lambda_j}\sum_{i=1}^n \gamma_j^{(k-1)}(x_i)x_i
\end{aligned}$$

which results into

$$\lambda_j^{(k)} = \frac{\sum_{i=1}^n \gamma_j^{(k-1)}(x_i)x_i}{\sum_{i=1}^n \gamma_j^{(k-1)}(x_i)}$$

These algorithms are indeed applications of a general schema named **Expectation-Maximization**