Machine Learning Notes

by Mattia Orlandi

3. Clustering

Problem of clustering:

- A set of N objects, each described by D values (the *dimensions*), is given.
- The task is to find a *natural* partitioning in K clusters and, possibly, a number of *noise* objects.
- The result is a *clustering scheme*, i.e. a function mapping each data object to the sequence $[1, \dots, K]$ or to noise.

Desired properties of clusters:

- maximize intra-cluster similarity;
- minimize inter-cluster similarity.

Taxonomy of clustering methods:

- Partitioning: K-means, expectation minimization, CLARANS.
- Hierarchic: agglomerative/divisive, BIRCH, CURE.
- Based on linkage.
- Based on density: DBSCAN, DENCLUE.
- Statistics: IBM-IM demographic clustering, COBWEB, Autoclass.

3.1. K-means

1. Ask user the number of clusters K.
2. Random choice of K points as temporary centers.
3. For each data point, find its nearest temporary center and assign the point to the corresponding cluster.
4. For each cluster, calculate the centroid of its points.
5. Move the temporary centers to the centroids of corresponding cluster.

Minimize distortion

Given:

- coding function $\text{Encode} : \mathbb{R}^D \to [1..K]$
- decoding function $\text{Decode} : [1..K] \to \mathbb{R}^D$
- the distortion/Sum of Squared Errors is

$$\sum_{i=1}^{N} (e_i - \text{Decode}(\text{Encode}(e_i)))^2, \quad \text{Decode}(k) = \mathbf{c}_k$$

$$\Rightarrow \text{Distortion} = \sum_{i=1}^{N} (e_i - \mathbf{c}_{\text{Encode}(e_i)})^2 = \sum_{i=1}^{N} \sum_{i \in \text{OwnedBy}(\mathbf{c}_j)} (e_i - \mathbf{c}_j)^2$$

To achieve minimal distortion, $\mathbf{c}_1, \ldots, \mathbf{c}_K$ must have the following properties:

- e_i must be encoded with the nearest center

$$\mathbf{c}_{\text{Encode}(e_i)} = \min_{\mathbf{c}_j \in \{\mathbf{c}_1, \ldots, \mathbf{c}_K\}} (e_i - \mathbf{c}_j)^2.$$

- Partial derivative of distortion w.r.t. the position of each center must be zero, since it means that the function has either a maximum or a minimum.
- Each center must be the centroid of the points it owns.

Issues

There is only a finite number of ways to partition N objects into K groups, and each change of state bring to a state which was never visited before and with a lower distortion $\Rightarrow$ sooner or later the algorithm terminates becose there are no new states reachable.

Obs.: the final state is not necessarily the best possible.

The starting point is important:

- choose randomly the first starting point;
- choose in sequence the $2..K$ starting points as far as possible from the preceding ones;
- re-run the algorithm with different starting points.

To choose the number of clusters:

- try various values;
- quantitative evaluation of the quality of clustering scheme;
- the best value finds optimal compromise between minimization of inter-cluster distances and maximization of intra-cluster distances.

Proximity function:

- Euclidian distance is a good choice for vector spaces.
- Several alternatives for specific data types.

Sum of Squared Errors:

$$SSE = \sum_{i=1}^{N} \sum_{i \in \text{OwnedBy}(\mathbf{c}_j)} (e_i - \mathbf{c}_j)^2 = \sum_{i=1}^{N} SEE_j$$

- a cluster j with high SSE_j has low quality;
- $SSE_j = 0$ iff all the points are coincident with the centroid;
- SEE decreases for increasing K, and is zero when $K = N$;
- $\Rightarrow$ therefore, minimizing SEE is not a viable solution to choose the best K.

Empty clusters:

- it may happen that, at some step, a centroid does not own any points;
- choose a new centroid:
    - among points far away from the empty cluster;
    - among points in the cluster with maximum SSE, in order to split in two the cluster with lowest quality.

Outliers:

- points very distant from their centroids $\Rightarrow$ high contribution to SSE;
- bad influence on clustering results $\Rightarrow$ they can be remove (depending to application domain).

Complexity: $\mathcal{O}(TKND)$, where

- T : number of iterations;
- K : number of clusters;
- N : number of data points;
- D : number of dimensions.

Final remarks:

- Pros:
    - efficient, nearly linear in the number of data points (in general, $T, K, D, << N$);
- Cons:
    - cannot work with nominal data;
    - requires the K parameter;
    - very sensitive to outliers;
    - does not deal with noise;
    - does not deal properly with non-convex custers.

3.2. Evalutation of a cluster

- Related only to result, not to clustering technique.
- Clustering is unsupervised:

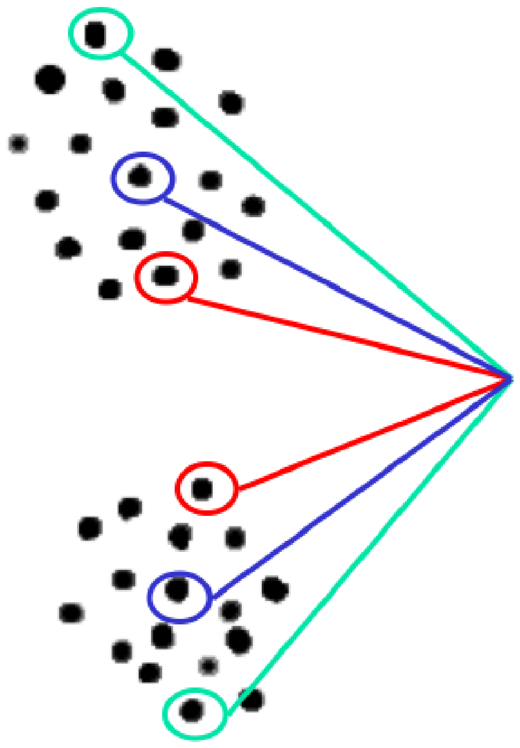
    - very little a priori information;
    - evaluation is critical ⇒ necessity of **indexes** to measure various properties of clusters and clustering scheme;
    - if some supervised data are available, they can be used to evaluate clustering scheme.
- In 2D clusters can be examined visually, but in higher order spaces it's better to use more formal methods.

Issues:

- distinguish patterns from random apparent regularities;
- find best number of clusters;
- unsupervised evaluation;
- supervised evaluation;
- comparison of clustering schemes.

Measurement criteria:

- **Cohesion**: proximity of objects in the same cluster should be high.
- **Separation**: two clusters should be distant
    - distance between the nearest objects of two clusters (red);
    - distance between the most distant objects in the two clusters (green);
    - distance between the centroids of the two clusters (blue).



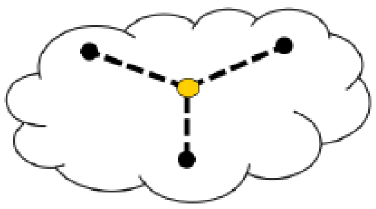
- **Similarity - Proximity**: a two-variable function measuring how two objects are similar according to values of their properties.
- **Dissimilarity**: a two-variable function measuring how two objects are different according to values of their properties.

Cohesion and separation

Cohesion: sum of proximities between elements of the cluster and the geometric center (called **prototype**)

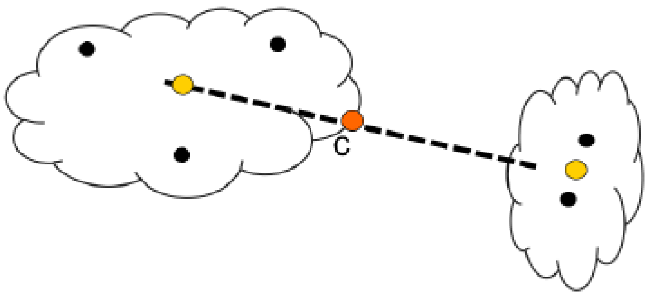
$$\text{Coh}(k_i) = \sum_{x \in k_i} \text{Prox}(x, \mathbf{c}_i)$$

- *centroid*: a point in space whose coordinates are the means of the cluster;
- *medoid*: an element of the dataset whose average dissimilarity with all the elements of the cluster is minimal (not necessarily unique, used in context where mean is not defined).



Separation: proximity of the prototypes of two clusters

$$\text{Sep}(k_i, k_j) = \text{Prox}(\mathbf{c}_i, \mathbf{c}_j)$$



Sum of Squares Between clusters (SSB):

$$\mathbf{c}: \text{ global centroid of dataset}$$
$$SSB = \sum_{i=1}^{K} N_i \cdot \text{Dist}(\mathbf{c}_i, \mathbf{c})^2$$

represents the global separation of clustering scheme.

Total Sum of Squares (TSS):

$$TSS = SEE + SSB$$

is a global property of the dataset, independent for clustering scheme.

Evaluation of specific clusters/objects:

- each cluster can have its own evaluation, and the worst ones can be splitted further;
- weakly separated pair of clusters could be considered for merging;
- single objects can give negative contribution to a cluster cohesion/separation (e.g. border objects).

Silhouette

- For i-th object, compute a_i, the average distance w.r.t. other objects of the same cluster.
- For i-th object and for each cluster other than its own, compute the average distance from all the objects of the cluster and find the minimum b_i.
- For the i-th object, the silhouette index is

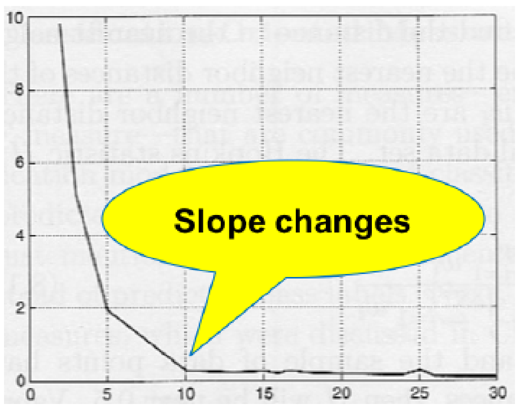
$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)} \in [-1, 1]$$

- Negative silhouette indexes indicates that the object is far from its cluster, whereas positive silhouette indexes indicates that it's near to it.

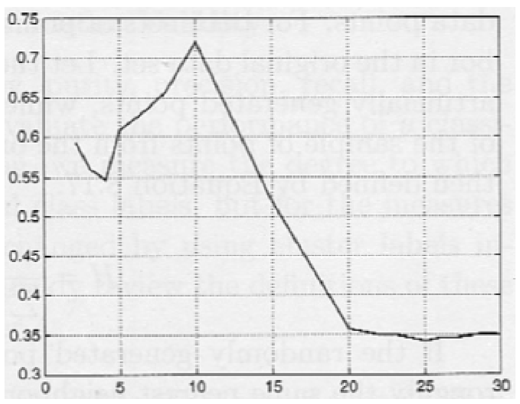
Choice of K

- Some algorithms (e.g. K-means) require the number of clusters as a parameter.
- Measures (e.g. SSE and Silhouette) are influenced by number of clusters $\Rightarrow$ they can be used to optimize K.
- Computation of Silhouette index is expensive.
- Computation of SSE decreases monotonically for increasing K:
  - equal to TSS for $K = 1$;
  - equal to zero when $K = N$.

Elbow method: the value at which SSE changes slope, or Silhouette has a maximum, corresponds to the optimal number of clusters.



SSE



Silhouette

Supervised measures

- Let be available a partition $P = \{P_1, \dots, P_L\}$ called **gold standard** (similar to the labelled data for training a classifier).

- Consider a clustering scheme $K = \{k_1, \dots, k_K\}$.
- By comparing it with the gold standard, it's possible to validate a clustering technique which can then be applied to new, unlabelled data (similar to testing a classifier).
- Classification-oriented methods measure how classes are distributed among clusters:
  - confusion matrix;
  - precision, recall, F-measure.
- Similarity-oriented methods analogous to comparing binary data:
  - Any pair of objects can be labelled as:
    - SS if they belong to the same set in P and K;
    - SD if they belong to the same set in K but not in P;
    - DS if they belong to the same set in P but not in K;
    - DD if they belong to different sets both in P and K.
  - Let a, b, c, d be the numbers of pairs in the above four categories.
  - The following indexes are defined:
    - Rand Index $R = \frac{a+d}{a+b+c+d}$;
    - Jaccard Coefficient $J = \frac{a}{a+b+c}$.

3.3. Hierachical clustering

- It generates a **nested structure** of clusters:
  - Agglomerative (bottom-up, most used):
    1. as a starting state, each data point is a cluster;
    2. in each step the two **less separated** clusters are merged into one;
    3. a measure of **separation between clusters** is needed.
  - Divisive (top-down):
    1. as a starting state, the entire dataset is a single cluster;
    2. in each step the cluster with the **lowest cohesion** is split;
    3. a measure of **cluster cohesion** and a **split procedure** are needed.

Separation

Graph-based: the distance between sets is based on the distances between objects belonging to the two sets

1. Single link:

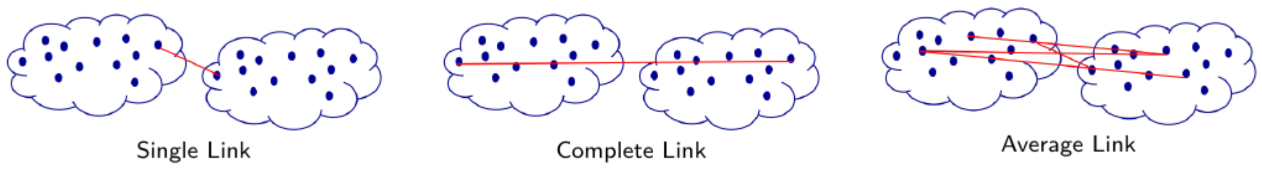
$$Sep(k_i, k_j) = \min_{x \in k_i, y \in k_j} Dist(x, y)$$

2. Complete link:

$$Sep(k_i, k_j) = \max_{x \in k_i, y \in k_j} Dist(x, y)$$

3. Average link:

$$Sep(k_i, k_j) = \frac{1}{|k_i||k_j|} \sum_{x \in k_i, y \in k_j} Dist(x, y)$$

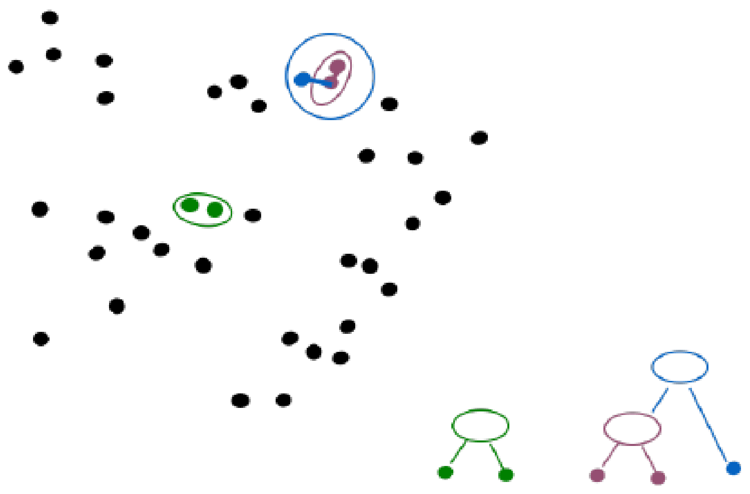


Prototype-based:

1. Distance between centroids.
2. Ward's method:
    - given two sets with the respective SSE, the separation between them is measured as the difference between the total SSE resulting in case of merge and the original SSE;
    - smaller separation implies lower increase in SSE after merging.

Single-linkage hierachical clustering

1. Initialize clusters, one for each object.
2. Compute the **distance matrix** between the clusters:
    - square and symmetric;
    - its size is the number of objects N;
    - main diagonal is null.
3. While there is at least one cluster:
    - find the two clusters with lowest separation k_r and k_s;
    - merge them in a cluster;
    - delete from distance matrix rows and column r and s and insert a new row and column with the distances of the new cluster from the others.
4. Final result is a **dendogram**.



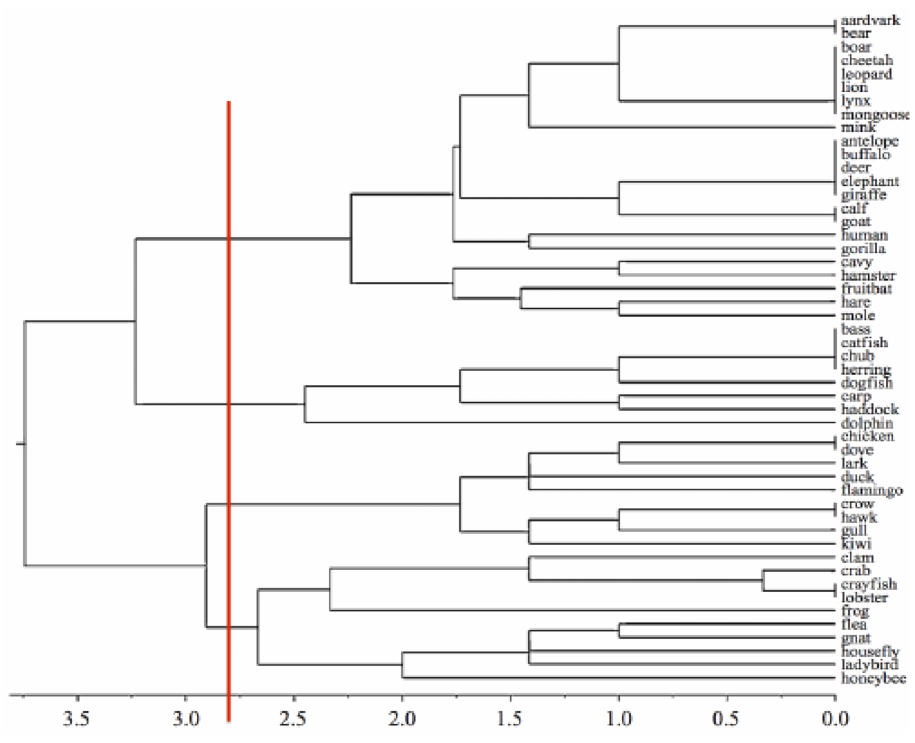
Space and time complexity:

- $\mathcal{O}(N^2)$ for computation and storage of distance matrix;
- worst case: $N-1$ iterations to reach final single cluster $\Rightarrow$ for i-th step:
    - $\mathcal{O}((N-i)^2)$ for search of the pair to merge;
    - $\mathcal{O}(N-i)$ for recomputation of distance matrix;

- $\mathcal{O}(N^3)$ in summary, but it can be reduced to $\mathcal{O}(N^2 \log(N))$ with indexing structures.

Final remarks:

- The desired clustering scheme is obtained by **cutting** the dendogram at some level (application dependent, possibly guided by indexes).
- The horizontal axis in the dendogram is the **total dissimilarity** inside the clusters, which increases for decreasing number of clusters.
- The **diameter** of a cluster is the distance among most separated objects:
  - single linkage ⇒ larger diameters also at low levels;
  - complete linkage ⇒ more compact clusters.



Obs.: - scaling is poor due to high complexity;
- there isn't a global objective function, decision is always local and cannot be undone.

3.4. Density-based clustering

Clusters are high-density regions separated by low-density regions.

Computing density:

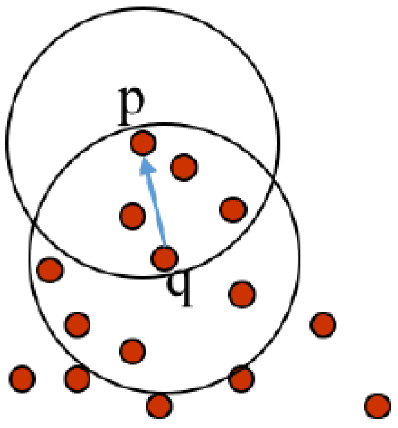
- **Grid-based**:
  - split hyperspace into regularly spaced grid;
  - count objects inside each grid element.
- **Object-centered**:
  - define radius of hypersphere;

    - attach to each object the number of other objects inside the sphere centered on the object itself.

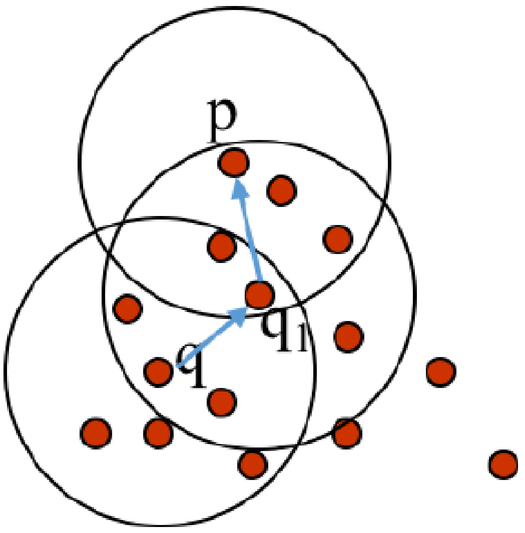
DBSCAN

DBSCAN stands for Density Based Spatial Clustering of Applications with Noise:

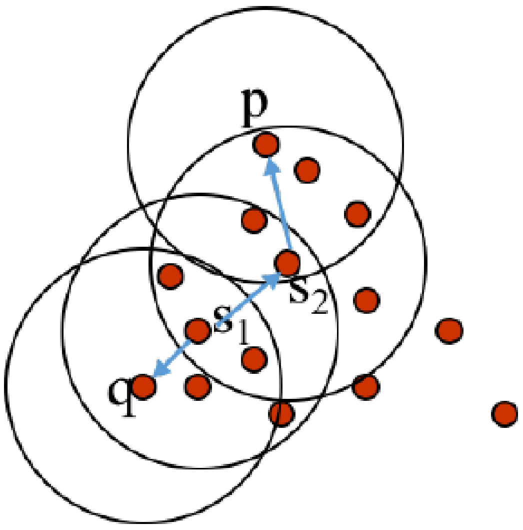
- intuitively, p is a **border** point while q is a **core** point;
- define a radius ϵ and define the **neighbourhood** of a point as the ϵ-hypersphere centered at tha point;
- points p and q are one in the neighbourhood of the other $\Rightarrow$ neighbourhood is *symmetric*;
- define threshold $\mathbf{minPoints}$ and define as **core** a point with at least $\mathbf{minPoints}$ points in its neighbourhood, as **border** otherwise:
    - with $\mathbf{minPoints} = 5$, q is a core and p is a border;
- a point p is **directly density reachable** from point q iff:
    - q is core;
    - q is in the neighbourhood of p;
- direct density reachability is not symmetric:
    - q is not directly density reachable from p, since the latter is border;



- a point p is **density reachable** from point q iff:
    - q is a core;
    - there is a sequence of points q_i s.t. q_{i+1} is directly density reachable from q_i, $i \in [1, nq]$, q_1 is directly density reachable from q and p is directly density reachable from q_{nq}
- density reachability is not symmetric:
    - q is not density reachable from p, since the latter is border;



- a point p is **density connected** to point q iff there is a point s s.t. p and q are density reachable from s;
- density connection is symmetric.



Generation of clusters:

- a cluster is a maximal set of point connected by *density*;
- border points not connected by density to any core point are labelled as **noise**.

Final remarks:

- finds clusters of any shape;
- robust w.r.t. noise;
- problems if clusters have widely varying densities;
- based on distances between points $\Rightarrow$ complexity of $\mathcal{O}(N^2)$, reduced to $\mathcal{O}(N\log(N))$ if spatial indexes are available (e.g. R^*);
- very sensitive to the values of ϵ and $\mathbf{minPoints}$;
- decreasing ϵ and increasing $\mathbf{minPoints}$ reduces clusters size and increases noise points.

Kernel Density Estimation

- Describe the distribution of data by a function.
- Overall density function is the sum of **influence functions** (or **kernel functions**) associated with each point.
- The kernel function:
  - must be symmetric and monotonically decreasing;
  - usually has a parameter to set decreasing rate.

DENCLUE algorithm:

1. Derive density function for the space of data points.
2. Identify local maxima points.
3. Associate each point with a density attractor by moving towards maximum density increase.
4. Define clusters consisting of points associated with a particular density attractor.
5. Discard clusters whose density attractor has a density less than a user-specified threshold ξ.
6. Combine clusters connected by a path of points that all have a density of ξ or higher.

Final remarks:

- precise computation of density (DBSCAN is a special case of DENCLUE where the kernel is a step function);
- complexity of $\mathcal{O}(N^2)$, optimizable with approximated grid-based computation;

3.5. Model based clustering

- Estimate parameters of a statistical model to maximize the ability of the model to **explain data**.
- Main technique is to use **mixture models**: data are seen as a set of observations from a mixture of different probability distributions.
- Usually, base model is a multivariate normal.
- Estimation is usually done using **maximum likelihood**: given a set of data $\mathcal{E}$, its probability w.r.t. to parameters is called **likelihood function**.
- Attribute assumed to be random independent variables.

Expectation Maximization - EM

1. Select an initial set of model parameters.
2. Repeat:
   - Expectation Step: for each object, calculate the probability it belongs to each distribution.
   - Maximization Step: given the probabilities from the expectation step, find the new estimates of the parameters that minimize the expected likelihood.
3. The algorithm terminates when parameters do not change, or the change is below a specified threshold.

3.6. Final remarks

Clustering types:

- **Partitioning**: iteratively find partitions in the dataset, optimizing some quality criterion.
- **Hierarchic**: recursively compute a structured hierarchy of subsets.
- **Density based**: compute densities and aggregates clusters in high density areas.
- **Model based**: assume a model for the distribution of data and find model parameters which guarantee best fitting to data.

Clustering scalability:

- Effectiveness decreases with:
  - dimensionality D;
  - noise level.
- Computational cost increases with:
  - dataset size N, at least linearly;
  - dimensionality D.