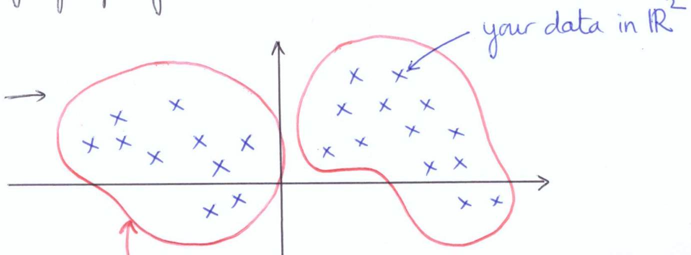
UL = CLUSTERING

In machine learning, unsupervised learning (UL) aims at describing hidden patterns in data. The training sample $\mathcal{D}_n = \{x_1, \dots, x_n\}$ consists of unlabeled data, as opposed to supervised learning (SL) problems, for which pairs (x_i, y_i) are observed, and whose goal is the predition of the response variable y. The tasks of an UL problem can be of a very different nature, and include clustering, dimension reduction and density estimation. Because of the absence of a response variable, the quality of the results is very subjective, and results in a huge amount of methods and algorithms.

x Clustering. The goal is to partition the data into distinct groups. Observations within each group should be similar in some sense.

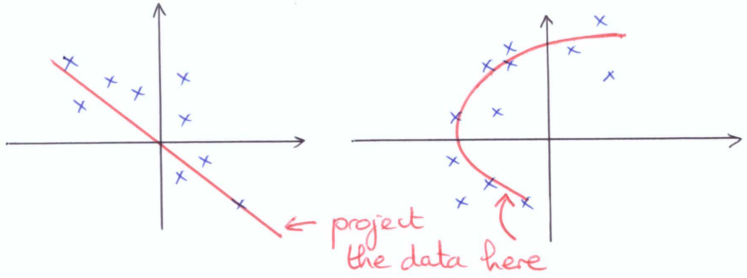
Ex
- Identify groups of people who tend to buy similar products
- Identify areas where there are greater incidences of a particular type of crime.
- In social network analysis, identify communities within large groups of individuals.

Also, provides a reduced representation of the full dataset.



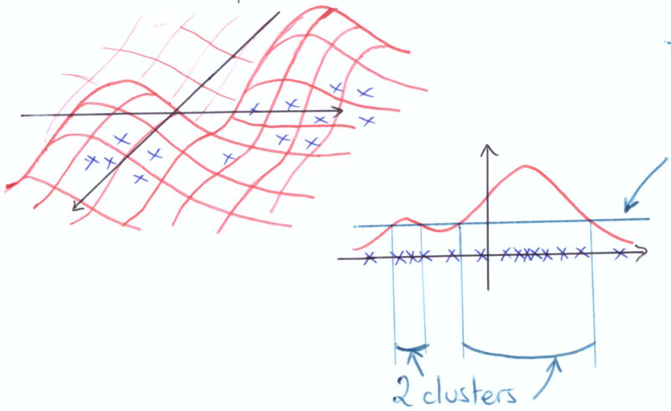
Set of points belonging to one cluster.

x Dimension reduction. The objective here is to project the data into lower dimensional spaces, keeping as much as possible the structure in the data (not loosing too much information). Useful if x lies in a high dimensional space. Dimension reduction techniques can be used as well for data visualization, or as a pre-processing step for a SL task. Usual techniques include Principal Component Analysis (PCA) and its variants.



project the data here

x Density estimation. Observations $x_1, \dots, x_n$ are assumed to be drawn from an unknown probability distribution P_X, with density f. The objective is to come up with an estimate of f. In low-dimensional settings, there exists many non-parametric techniques that perform well.



The density estimate can then be used for clustering: fix some threshold (in green here), and group points together if the density estimate is above the threshold.

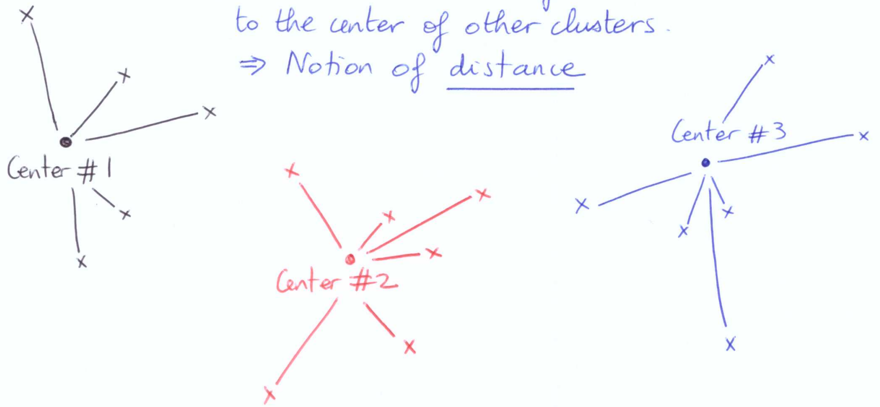
I - K-MEANS CLUSTERING

I.1. The 'basic' K-means algorithm

The goal is to discover similarities in the data. To do so, we partition data into distinct groups, so that observations within a group are similar in some sense.

↖ We need a mathematical description of similarity. For instance, a good cluster is such that the within-cluster variation is small: points are close together: each point of the cluster is closer to the center of the cluster than it is to the center of other clusters.

⇒ Notion of distance



× Several questions arise:

(i) Which distance to use?

(ii) How many clusters do we want?

(iii) Do we want to consider a probabilistic approach?

× Possible answers:

(i) If the data is numerical, i.e. lies in the Euclidean space $\mathbb{R}^d$, one typically uses the ℓ_2 Euclidean distance, defined as $\underset{\|x-y\|_2}{\underset{\shortparallel}{\ell_2(x,y)}} = \left(\sum_{i=1}^{d} (x_i-y_i)^2 \right)^{1/2}$, $x=(x_1,—,x_d)$, $y=(y_1,—,y_d)$ $\in \mathbb{R}^d$.

Alternatively, one may use the ℓ_1 - Manhattan distance, $\ell_1(x,y) = \sum_{i=1}^{d} |x_i - y_i| = \|x-y\|_1$.

(ii) The number K of clusters may or may not be fixed in advance, resulting in different algorithms.

→ K fixed (K means, Gaussian Mixtures)

The choice of the value of K is problematic:

- Use prior knowledge, if available
- Run the algorithm on data with several values of K.

→ K variable (Hierarchical clustering)

(iii) Non-probabilistic approach → K-means

Probabilistic approach → Gaussian Mixtures Model.

More formally, the K-means optimization problem can be stated as follows:

- Input =
  - $\mathcal{L}_n = \{x_1, .., x_n\}$ = training sample $x_i \in \mathbb{R}^d$
  - K = number of clusters
- Output =
  - Set of points $M \subset \mathbb{R}^d$ containing exactly K elements $M = \{m_1, \dots, m_K\}$, $m_i \in \mathbb{R}^d$ aka the centroids / prototypes
- Cost function = Minimize the total cost (over M, π)

$$\mathcal{C}(M,\pi) = \sum_{k=1}^{K} \sum_{i=1}^{n} \pi_{ik} \| x_i - m_k \|_2^2 ,$$

where

$$\pi_{ik} = \begin{cases} 1 & \text{if } x_i \text{ belong to } k\text{-th cluster} \\ 0 & \text{otherwise.} \end{cases}$$

A direct exhaustive search is computationally prohibitive

Remarks = (i) As mentioned earlier, we may consider instead the ℓ_1-distance, and consider

$$\mathcal{C}_1(M) = \sum_{k=1}^{K} \sum_{i=1}^{n} \pi_{ik} \| x_i - m_k \|_1 .$$

(ii) Not a convex optimization problem ⇒ existence of many local minima ⇒ algorithm(s) will be sensitive to initialization.

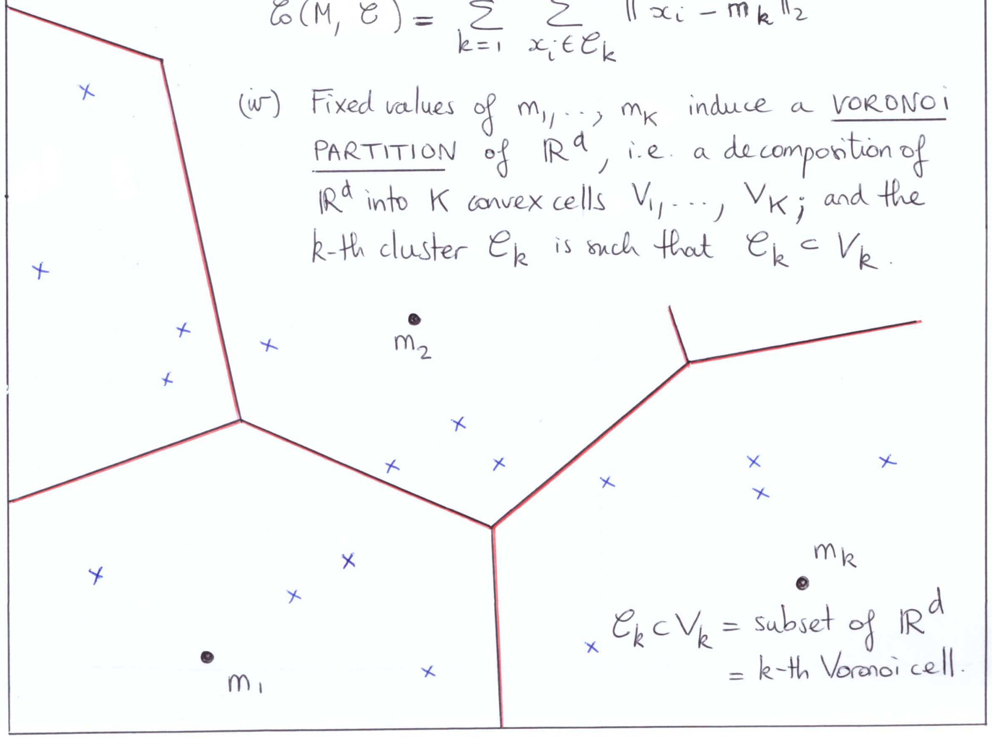
(iii) The objective criterion may be rewritten differently by defining formally clusters as

$$\mathcal{C}_k = \Big\{ x_i \in \mathcal{L}_n \;\Big|\; \| x_i - m_k \|_2^2 \leqslant \| x_i - m_j \|_2^2 \quad \forall j \neq k \Big\}$$

k-th cluster ↑
= Set of points in $\mathcal{L}_n$ closer to m_k than any other m_j, $j \neq k$. ($\mathcal{C} = \{\mathcal{C}_1, \dots, \mathcal{C}_K\}$)

$$\mathcal{C}_0(M, \mathcal{C}) = \sum_{k=1}^{K} \sum_{x_i \in \mathcal{C}_k} \| x_i - m_k \|_2^2$$

(iv) Fixed values of $m_1, \dots, m_K$ induce a <u>VORONOI PARTITION</u> of $\mathbb{R}^d$, i.e. a decomposition of $\mathbb{R}^d$ into K convex cells $V_1, \dots, V_K$; and the k-th cluster $\mathcal{C}_k$ is such that $\mathcal{C}_k \subset V_k$.



$\mathcal{C}_k \subset V_k$ = subset of $\mathbb{R}^d$
= k-th Voronoi cell.

→ If we have a single cluster $(K=1)$, the cost function reduces to $\mathcal{C}_0(m_1) = \sum_{i=1}^{n} \| x_i - m_1 \|_2^2$, which is minimized for $\hat{m}_1 = \frac{1}{n} \sum_{i=1}^{n} x_i = \bar{x}$.

(Under ℓ_1-cost, the minimum occurs at the median value).

We can explicitely quantify the extra cost incurred by selecting another value z of the centroid insted of the optimal $\bar{x}$

$$\forall z \in \mathbb{R}^d \qquad \mathcal{C}_0(z) = \mathcal{C}_0(\bar{x}) + n \| z - \bar{x} \|_2^2$$

↳ <u>Probabilistic proof</u>: take $X \in \mathbb{R}^d$ = random variable
$z \in \mathbb{R}^d$ = vector

Then

$$\begin{aligned} & \mathbb{E}\|X - \mathbb{E}X\|^2 + \| z - \mathbb{E}X \|^2 \\ &= \mathbb{E}\|X\|^2 + \|\mathbb{E}X\|^2 - 2 < \mathbb{E}X, \mathbb{E}X > \\ &\qquad + \|z\|^2 - 2 < z, \mathbb{E}X > + \|\mathbb{E}X\|^2 \\ &= \mathbb{E}\|X\|^2 + \|z\|^2 - 2 < z, \mathbb{E}X > \\ &= \mathbb{E}\|X - z\|^2 \end{aligned}$$

Take $\mathbb{P}(X = x_i \mid \mathcal{L}_n) = 1/n$ = empirical distribution of sample $\mathcal{L}_n$. Then

$$\mathbb{E}\|X - z\|^2 = n^{-1} \sum_i \| x_i - z \|_2^2 = n^{-1} \mathcal{C}_0(z)$$

$$\mathbb{E}\|X - \mathbb{E}X\|^2 \underset{\uparrow}{=} n^{-1} \sum_i \| x_i - \bar{x} \|_2^2 = n^{-1} \mathcal{C}_0(\bar{x})$$

$$\mathbb{E}X = n^{-1} \sum_{i=1}^{n} x_i$$

$$\| z - \mathbb{E}X \|^2 = \| z - \bar{x} \| \qquad ■$$

→ If $K \geqslant 2$, we proceed iteratively. Each iteration involves two steps, corresponding to successive optimizations of the centroids and the clusters.

- The second step is straightforward: once the centroids are known, assign x_i to $\mathcal{C}_k$ corresponding to the closest centroid.

- Once the x_is have a label (corresponding to cluster allocation), the centroid update is also straightforward: everything behaves as in the case of a single cluster: simply update the position of the centroid in each cluster to be the mean (or median) value of the observations belonging to that cluster.

K-MEANS ALGORITHM

- Initialization Step: Select K points $m_1^{(0)}, \dots, m_K^{(0)}$ as initial centroids
- Repeat for $\ell = 0, 1, 2, \dots$ (Until centroids do not change)
  - Form K clusters by assigning each point to its closest centroid

    For $k = 1, -, K$,

    $$\mathcal{C}_k^{(\ell+1)} = \{x_i \in \mathcal{L}_n \text{ whose closest centroid is } m_k^{(\ell)}\}$$

  - Recompute the centroid of each cluster

    For $k = 1, -, K$,

    $$m_k^{(\ell+1)} = \frac{1}{|\mathcal{C}_k^{(\ell+1)}|} \sum_{x_i \in \mathcal{C}_k^{(\ell+1)}} x_i$$

    where

    $|\mathcal{C}_k^{(\ell+1)}| = \#$ observations in $\mathcal{C}_k^{(\ell+1)}$.

Claim: Each iteration of the K-means algorithm reduces the value of the cost $\mathcal{C}_0(M)$.

Indeed, let $\mathcal{C}^{(\ell)} = \{\mathcal{C}_1^{(\ell)}, \dots, \mathcal{C}_K^{(\ell)}\}$ = clusters
$M^{(\ell)} = \{m_1^{(\ell)}, \dots, m_K^{(\ell)}\}$ = centers. at ℓ-th iteratio

Cost at iteration ℓ is:

$$\mathcal{C}_0(M^{(\ell)}, \mathcal{C}^{(\ell)}) = \sum_{k=1}^{K} \sum_{x_i \in \mathcal{C}_k^{(\ell)}} \| x_i - m_k^{(\ell)} \|_2^2$$

x Step 1: $M^{(\ell)}$ is fixed, and assign each point to its closest centroid in $M^{(\ell)}$. Thus

$$\mathcal{C}_0(M^{(\ell)}, \mathcal{C}^{(\ell+1)}) \leq \mathcal{C}(M^{(\ell)}, \mathcal{C}^{(\ell)})$$

x Step 2: Re-center each cluster. It follows from the result of page 6 that

$$\mathcal{C}_0(M^{(\ell+1)}, \mathcal{C}^{(\ell+1)}) \leq \mathcal{C}_0(M^{(\ell)}, \mathcal{C}^{(\ell+1)}),$$

since $M^{(\ell)}$ is suboptimal for the current partition $\mathcal{C}^{(\ell+1)}$.

The cost at iteration $\ell+1$ is thus no larger than the cost at iteration ℓ.

Corollary: The K-means algorithm converges to a local minimum.

↳ There is a finite number of ways of labeling the points, each with an optimal way of placing the centers (mean value) ⇒ there is a finite number of possible values for the total cost. Since the K-means algorithm decreases the value of the cost at each iteration, it will converge after a finite number of steps, that is when no re-assignment of observations will result in a decrease of the cost. When no-reassignment take place, the representatives are also unchanged.

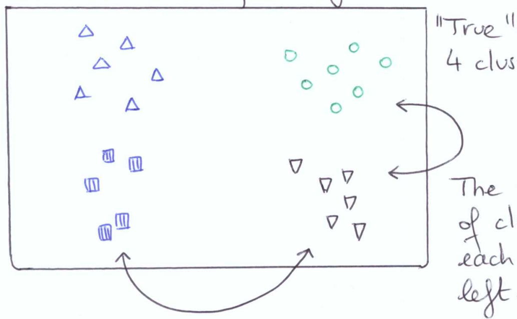
Issues of the K-means algorithm.

(a) Choosing initial centroids.

If random initialization (i.e. select uniformly K points out of n) of centroids is used, different runs of the algorithm will produce different outputs, with different total costs. A common approach is to select the solution with the smallest cost.

⚠ Depending on the dataset, and the number of clusters sought, this may not work very well.

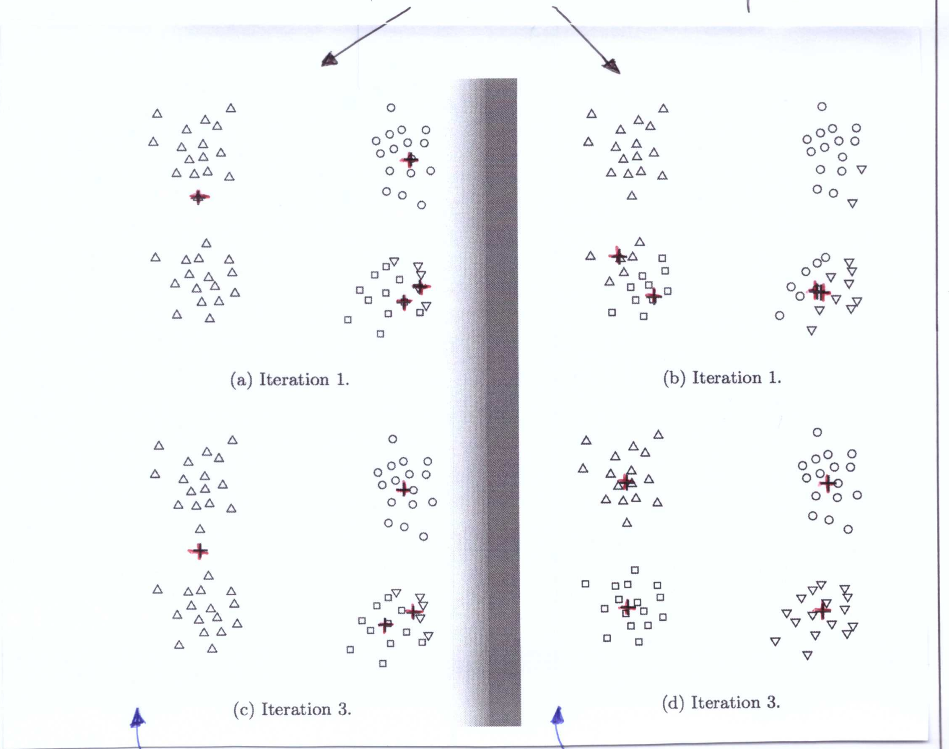
Indeed, consider the following data set:



"True" data, consisting of 4 clusters

The top-down pairs of clusters are closer to each other than the left-right pairs.

Consider two initialization (marked as x) & compare



(a) Iteration 1.

(b) Iteration 1.

(c) Iteration 3.

(d) Iteration 3.

Two clusters are missed

All clusters correctly identified

Observation: starting with two initial centroids per pair of clusters yields the right clusters; while if they are unevenly distributed initially, two clusters are combined in one, and one "true" cluster is split in two.

→ For the K-means algorithm to work well, having one initial centroid in each cluster (or, in our previous example, one in each pair) is a good initialization. However, as K gets larger, this becomes increasingly unlikely.

Consider the case of K equally sized clusters; all having the same number n of observations per cluster. The probability of selecting in a sample of size K one initial centroid from each cluster is (assuming sampling with replacement)

$$\frac{\#\text{ways selecting one centroid from each cluster}}{\#\text{ways selecting } K \text{ centroids}} = \frac{K!\,n^K}{(nK)^K} = \frac{K!}{K^K}$$

↳ For $K = 4$, this probability is ≈ 0.094. Already very small!

⇒ Other approaches are used for initialization. One approach is to first select a point at random from $\mathcal{L}_n$, and then select the farthest away point from any of the already selected initial centroids.

⊕ • Points are well separated

⊖ • Expensive procedure

• May select outliers rather than points in dense regions

↳ It's OK: just apply the procedure to a random sample of points from $\mathcal{L}_n$: points in dense areas are likely to be selected, and computation is greatly reduced.

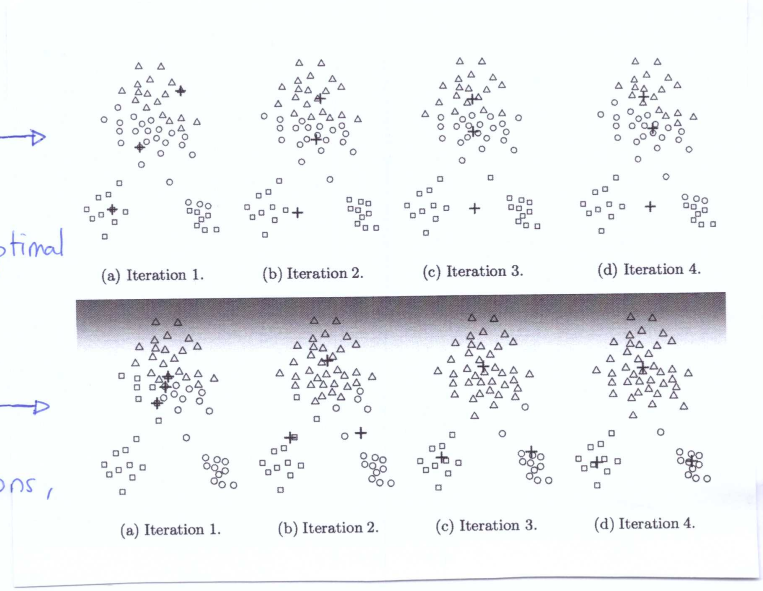
⚠ Points well separated may still produce poorly selected centro

Indeed, consider the following example: 3 clusters; 2 runs of the K-means algorithm are considered.

Init # 1 = initial centroids are from a single cluster

Init # 2 = initial centroids are evenly distributed.

However, the second run of the algorithm returns a suboptimal solution, compared to the first initialization


(a) Iteration 1. (b) Iteration 2. (c) Iteration 3. (d) Iteration 4.

(a) Iteration 1. (b) Iteration 2. (c) Iteration 3. (d) Iteration 4.

Initial points are well separated, but K-means returns a suboptimal solution →

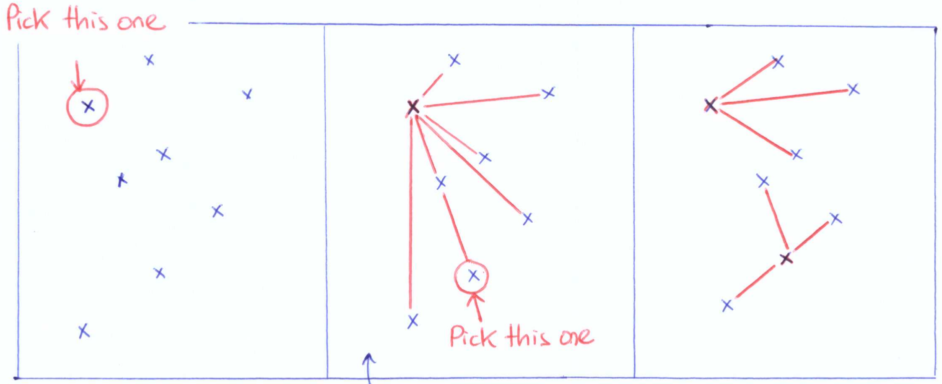
Initial centroids belong to one cluster, but after 4 iterations, the 3 clusters are found. →

Instead of selecting points that are farthest away, we can make the procedure random. This approach was suggested by Arthur & Vassilvitskii (2007), and results in the K-MEANS ++ algorithm.

↘ First, pick a point $x \in \mathcal{L}_n$ at random, and set $M = \{x\}$.

↘ Select another point at random, with probability proportional to $\min_{m \in M} \| x - m \|_2^2$, $x \in \mathcal{L}_n$, and set $M \leftarrow M \cup \{x\}$.

↘ Repeat the procedure until K points are selected.



The further away, the more likely a point $x \in \mathcal{L}_n$ will be selected.

In addition, there is some theoretical guarantee that the procedure returns initial centroids with an initial cost not too far away from the optimal cost.

Theorem =

Let • M = initial centroids returned by the K-means ++ algorithm, with associated cost $\mathcal{C}_0(M)$.

• M^* = optimal centroids, with associated cost $\mathcal{C}_0(M^*)$.

Then

$$\mathbb{E}\{\mathcal{C}_0(M)\} \leqslant O(\log K) \times \mathcal{C}_0(M^*),$$

where $\mathbb{E}(\ldots)$ is taken over the randomness in the initialization procedure.

↖ We are only $O(\log K)$ away from the optimal cost!

(b) Dealing with empty clusters.

It may happens that $\mathcal{C}_k^{(\ell)} = \emptyset$ for some ℓ, k, which is problematic. We can:

- Choose one point farthest away and allocate it to $\mathcal{C}_k^{(\ell)}$. This will reduce the total cost by much.
- Find the cluster with the highest cost, and split it in two. This will typically reduce the total cost.

(c) Presence of outliers.

The ℓ_2-metric gives too much weight to outliers, which may artificially increase the total cost, and the chosen centroids may not be as representative as they should be. You may use the ℓ_1 metric instead, or eliminate outliers beforehand.

(d) The K-means algorithm has trouble detecting the "true" clusters, when these clusters have a non-spherical shape, or are of very different sizes.

the ball induced by the ℓ_2-metric.

I.2. Variations of K-means.

(i) Bisecting K-means: a simple variant to the original K-means algorithm, which is less sensitive to initialization. The idea is to construct the K clusters recursively: start by splitting the data set into two clusters; then select one of these clusters to split in two, and repeat, until K clusters are produced.

The algorithm is presented on the next page.

There are different ways to choose which cluster to split into two:

→ Select the largest one.

→ Select the one with the largest cost

→ A combination of size and cost --- /...

BISECTING K-MEANS

```
• Initialization step: $\mathcal{L}$ = list of clusters containing only one
    cluster containing all data points.
• Repeat until $\mathcal{L}$ contains K clusters
    x Select a cluster from the list (call it C), remove it from $\mathcal{L}$
    x For i = 1 to Number of Trials,
          Bisect C using K-means (With K=2)
      End
      Select the two clusters from the bisection with the lowest
        total cost
      Add these two cluster to the list $\mathcal{L}$
```

(ii) K-medoids algorithm: in the original K-means algorithm, the centroids are the average of all data points in each cluster. In some applications however, we may want each center to be one of the points itself (for interpretation purposes: if each point represents an image, average of points might not be a meaningful image) → These representative points are called MEDOIDS. The resulting algorithm is similar to K-means:

K-MEDOIDS ALGORITHM

```
• Initialization step: Select K points $m_k^{(0)}$ from $\mathcal{L}_n$
• Repeat for $\ell = 0, 1, 2, \ldots$
      • For $k = 1, -, K$
        $\mathcal{C}_k^{(\ell+1)} = \{ x_i \in \mathcal{L}_n$ whose closest medoid is $m_k^{(\ell)} \}$
      • For $k = 1, -, K$
        $m_k^{(\ell+1)} = x^* \in \mathcal{C}_k^{(\ell+1)}$ s.t. $\sum_{x_i \in \mathcal{C}_k^{(\ell+1)}} \|x_i - x^*\|_2^2 < \sum_{x_i \in \mathcal{C}_k^{(\ell+1)}} \|x_i - x\|_2^2$
                                                            $\forall x \in \mathcal{C}_k^{(\ell+1)}$
```

The K-medoids algorithm shares the properties of the K-means algorithm = each iteration decreases the total cost, the algorithm always converges, different starts will produce different results, it does not achieves the global minimum. Moreover, the total cost of K-medoids is usually higher by the final total cost produced by K-means (why ?). Also, K-medoids is computationally more demanding than K-means, since computing the medoid is harder than computing the mean value.

↳ One prefers a sub optimal less costly greedy approach. One such algorithm is called Partitioning Around Medoids (PAM).

(Absolute error used instead of ℓ_2-error)

Introduced by Kaufmann & Rousseeuw (1987)

PAM

- Initialization step: Select K points $m_k^{(0)}$ from $\mathcal{L}_n$
- Repeat for $\ell = 0, 1, \dots$
    - For $k = 1, \dots, K$
      $\mathcal{C}_k^{(\ell+1)} = \{ x_i \in \mathcal{L}_n$ whose closest medoid is $m_k^{(\ell)} \}$
    - For each representative object $m_k^{(\ell)}$ $(k = 1, \dots, K)$, randomly select a non representative point x_{rdm}, and compute the total cost of swapping $m_k^{(\ell)}$ with x_{rdm}. If the total cost is reduced, replace $m_k^{(\ell)}$ with x_{rdm}, otherwise keep $m_k^{(\ell)}$.

↑ Does not scale well with large datasets. Sampling based method CLARA (Clustering LARge Applications) was proposed by Kaufmann & Rousseeuw (1990).

(iii) K-modes clustering. Another issue with the K-means algorithm is that it cannot handle non-numerical (aka categorical) data (and mapping categorical value to 1/0 usually generates poor quality clusters in high dimensional data).

→ use a DISSIMILARITY MEASURE instead of the ℓ_1 or ℓ_2 metric.

Suppose observations have d attributes $A_1, \dots, A_d$, where each attribute A_j describes a set of (unordered) values $\{a_{j,1}, \dots, a_{j,n_j}\}$, $j = 1, \dots, d$.

If x and y are two categorical observations represented by $x = (x_1, \dots, x_d)$, $x_j \in A_j$

$y = (y_1, \dots, y_d)$, $y_j \in A_j$,

the simple MATCHING DISSIMILARITY measure between x and y is $d(x, y) = \sum_{j=1}^{d} \delta(x_j, y_j)$, where

$$\delta(x_j, y_j) = \begin{cases} 1 & \text{if } x_j \neq y_j \\ 0 & \text{otherwise} \end{cases}$$

The K-modes algorithm uses the same paradigm as K-means to cluster categorical data: the goal is to return K objects $M = \{m_1, \dots, m_K\}$ and their respective clusters $\mathcal{C} = \{\mathcal{C}_1, \dots, \mathcal{C}_K\}$ such that the cost

$$\mathcal{C}o(M, \mathcal{C}) = \sum_{k=1}^{K} \sum_{x_i \in \mathcal{C}_k} d(x_i, m_k)$$

is minimized.

K-MODES ALGORITHM

- Initialization Step = Randomly select **K** points in $\mathcal{L}_n$, denoted
  $m_1^{(0)}, \dots, m_K^{(0)}$.

- Repeat for $l = 0, 1, 2, \dots$
    - For $k = 1, \dots, K$

      $$\mathcal{C}_k^{(l+1)} = \{ x_i \in \mathcal{L}_n \text{ whose } \underset{\substack{\uparrow \\ \text{defined in terms of the metric } d}}{\text{closest}} \text{ centroid is } m_k^{(l)} \}$$
    - For each cluster $\mathcal{C}_k^{(l+1)}$, $k = 1, \dots, K$, compute the new mode $m_k^{(l+1)}$; where the mode of an attribute is equal to the most frequent value.

Aternative dissimilarity measure was proposed by Ng, Li, Huang & He (2007) between a data point x_i and a cluster center m_k:

$$x_i = (x_{i,1}, \dots, x_{i,d}) \qquad m_k = (m_{k,1}, \dots, m_{k,d})$$

$$d'(x_i, m_k) = \sum_{j=1}^{d} \phi(x_{i,j}, m_{k,j}),$$

where

$$\phi(x_{ij}, m_{k,j}) = \begin{cases} 1 & \text{if } x_{i,j} \neq m_{k,j} \\ 1 - \dfrac{n_{i,j,k}}{|\mathcal{C}_k|} & \text{otherwise} \end{cases},$$

where $n_{i,j,k}$ = number of observations in $\mathcal{C}_k$ that have the same attribute as $x_{i,j}$ and $m_{k,j}$

(if all observations in $\mathcal{C}_k$ have the same j-th attribute, then $n_{i,j,k} = |\mathcal{C}_k|$ and $\phi(x_{i,j}, m_{k,j}) = 0$; and we recover the matching dissimilarity measure)

$\Rightarrow \phi$ takes into account the dominant level of the mode category in the calculation of the dissimilarity measure.

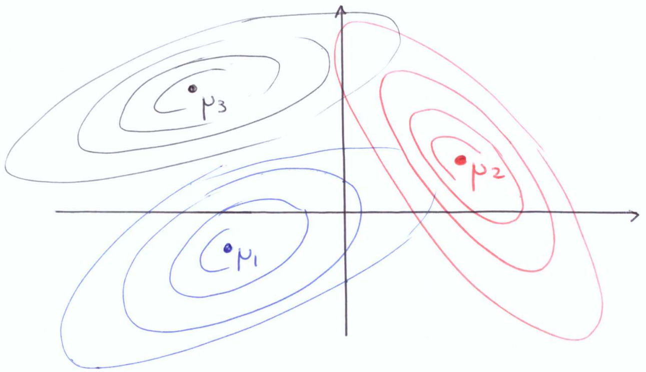
II_ GAUSSIAN MIXTURE MODEL

Consider a probabilistic approach to clustering. We assume that we have K (fixed) clusters, and data from the k-th cluster arises from a multivariate normal distribution $\mathcal{N}(\mu_k, \Sigma_k)$, $k = 1, \dots, K$, where $\mu_k \in \mathbb{R}^d$, and $\Sigma_k \in \mathbb{R}^{d \times d}$ is positive definite. The prior probability that an observation belongs to cluster k is denoted p_k, so that the resulting distribution is a mixture of K normal distributions:

$$f(x \mid \alpha, \mu, \Sigma) = \sum_{k=1}^{K} \underset{\substack{\uparrow \\ \text{aka MIXING COEFFICIENTS}}}{p_k} \mathcal{N}(x \mid \mu_k, \Sigma_k),$$

where

- $\alpha = \{\alpha_1, \dots, \alpha_K\}$, $\mu = \{\mu_1, \dots, \mu_K\}$, $\Sigma = \{\Sigma_1, \dots, \Sigma_k\}$
- $\mathcal{N}(x \mid \mu_k, \Sigma_k) = \dfrac{1}{(2\pi)^{d/2} |\Sigma_k|^{1/2}} \exp\left\{ -\frac{1}{2} (x - \mu_k)^t \Sigma_k^{-1} (x - \mu_k) \right\}$
- $\sum_{k=1}^{K} p_k = 1$



For a particular observation x, we do not know from which density it was generated from.

Objective: estimate model parameters p, μ, Σ.

A general approach in parametric statistics is to consider Maximum Likelihood Estimation (MLE) based on a random sample $\mathcal{L}_n = \{ x_1, \dots, x_n \}$.
Put $\Theta := \{ p, \mu, \Sigma \}$ = set of all model parameters.

$$\ell(\Theta \mid \mathcal{L}_n) = \log \left\{ \prod_{i=1}^{n} f(x_i \mid \Theta) \right\}$$

$$= \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} p_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k) \right)$$

Try to find analytically the expression of the MLE by differentiating ℓ with respect to p, μ, and Σ. You will run into troubles because of the presence of the sum inside the log. This is a common issue in the presence of lattent (aka hidden) variables: if we knew from which density $\mathcal{N}(x \mid \mu_k, \Sigma_k)$ observation x_i was coming from, the sum inside the log would disappear: suppose instead that we observe pairs (x_i, z_i), where $z_i \in \{1, \dots, K\}$ denotes the cluster identity of x_i: $z_i = k$ means that observation x_i was generated by the k-th gaussian distribution.

Then $f(x_i, z_i) = \prod_{k=1}^{K} \left[\mathcal{N}(x_i \mid \mu_k, \Sigma_k) \right]^{\mathbb{1}(z_i = k)}$

The log likelihood becomes

$$\ell(\Theta \mid \mathcal{L}'_n) = \log \prod_{i=1}^{n} \prod_{k=1}^{K} \left[\mathcal{N}(x_i \mid \mu_k, \Sigma_k) \right]^{\mathbb{1}(z_i = k)}$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}(z_i = k) \log \mathcal{N}(x_i \mid \mu_k, \Sigma_k),$$

$\mathcal{L}'_n = \{ (x_1, z_1), \dots, (x_n, z_n) \}$

and maximizing this function with respect to μ and Σ presents no difficulty.

In such situations, a general approach is to augment the log-likelihood by artificially adding the missing information, and to maximize the so-called COMPLETE LOG-LIKELIHOOD. Since the missing information is – by definition – unobserved, we marginalize out the effect of the hidden variable and make the procedure iterative, resulting in the EM ALGORITHM. The EM algorithm is a general algorithm that can be adapted to the problem at hand. Some applications include parameter estimation in

- Gaussian Mixture Models (See section II.2)
- Hidden Markov Models (Time Series)
- Kalman Filtering (Time Series)
- Mixture Discriminant Analysis (Supervised Learning)

II.1 EM algorithm.

We adopt the following notation: observed variables are denoted X, and hidden / latent / unobserved variables Z. The joint distribution $p(x, z \mid \Theta)$ is parametrized by Θ.

The goal is to maximize the log-likelihood

$$\ell(\Theta) = \log p(x \mid \Theta) = \log \sum_{z} p(x, z \mid \Theta).$$

calculations are presented for one observations, but can easily be generalized for n obs:

$$\ell(\Theta) = \log \prod_{i=1}^{n} p(x_i \mid \Theta) = \sum_{i=1}^{n} \log p(x_i \mid \Theta) = \sum_{i=1}^{n} \log \sum_{z_i} p(x_i, z_i \mid \Theta).$$

We assume that Z is discrete. But the forecoming calculations work for an absolutely continuous Z as well, just replace sums with integrals.

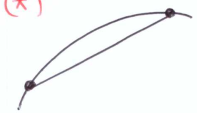
Idea = direct maximization of $p(x \mid \Theta)$ is analytically challenging, while maximization of the COMPLETE-LIKELIHOOD $p(x, z \mid \Theta)$ is easier.

Instead of maximizing $\ell(\theta)$ directly, we maximize a lower bound for it.

Let $q(z)$ = arbitrary distribution over z (to be specified shortly)

$$
\begin{aligned}
\ell(\theta) &= \log p(x|\theta) \\
&= \log\Big\{\sum_z p(x,z|\theta)\Big\} \\
&= \log\Big\{\sum_z q(z)\,\frac{p(x,z|\theta)}{q(z)}\Big\} \\
&\geq \sum_z q(z)\log\Big\{\frac{p(x,z|\theta)}{q(z)}\Big\} \quad (*) \\
&=: F(q,\theta)
\end{aligned}
$$

log is concave and $q(z)$ sum to one.



Find the best possible lower bound: maximize (iteratively) F over q and θ:

E-step $\quad q^{(m+1)} = \arg\max_q F(q,\theta^{(m)})$

M-step $\quad \theta^{(m+1)} = \arg\max_\theta F(q^{(m+1)},\theta)$

We now have a closer look at the E-step. Maximization is over a set of probability distributions, which is huge, and maximization is thus non-trivial. However, we notice that a sufficient condition for the inequality in $(*)$ to become an equality is to enforce the ratio $\frac{p(x,z|\theta)}{q(z)}$ to be independent of the variable z. Indeed, in this case,

$$
\underbrace{\log\Big\{\sum_z q(z)\frac{p(x,z|\theta)}{q(z)}\Big\}}_{\ell(\theta)} = \log\Big\{\frac{p(x,z|\theta)}{q(z)}\overset{1}{\overbrace{\sum_z q(z)}}\Big\}
$$

independent of z

$$
= \log\Big\{\frac{p(x,z|\theta)}{q(z)}\Big\}
$$

$$
= \Big\{\sum_z q(z)\Big\}\log\Big\{\frac{p(x,z|\theta)}{q(z)}\Big\} = F(q,\theta)
$$

To make $\frac{p(x,z|\theta)}{q(z)}$ independent of z, choose $q(z)$ proportional to $p(x,z|\theta)$: $\quad q(z) \propto p(x,z|\theta)$

read "proportional to"

$$
q(z) = \frac{p(x,z|\theta)}{\sum_z p(x,z|\theta)} = \frac{p(x,z|\theta)}{p(x|\theta)},
$$

normalizing constant

and we conclude that $q(z) = p(z|x,\theta)$ = conditional distribution of z given x.

The E-step becomes $\quad q^{(m+1)}(z) = p(z|x,\theta^{(m)})$

We plug in here the current parameter estimate $\theta^{(m)}$.

We turn our attention to the M-step $\theta^{(m+1)} = \arg\max_\theta F(q^{(m+1)},\theta)$, where

$$
\begin{aligned}
F(q^{(m+1)},\theta) &= \sum_z q^{(m+1)}(z)\log\Big\{\frac{p(x,z|\theta)}{q^{(m+1)}(z)}\Big\} \\
&= \sum_z p(z|x,\theta^{(m)})\log\Big\{\frac{p(x,z|\theta)}{p(z|x,\theta^{(m)})}\Big\} \\
&= \sum_z p(z|x,\theta^{(m)})\log\{p(x,z|\theta)\} \\
&\quad - \boxed{\sum_z p(z|x,\theta^{(m)})\log\{p(z|x,\theta^{(m)})\}}
\end{aligned}
$$

looking for a maximizer of this term with respect to θ.

term independent of θ

Put
$$
\begin{aligned}
Q(\theta,\theta^{(m)}) &:= \sum_z p(z|x,\theta^{(m)})\log\{p(x,z|\theta)\} \\
&= \mathbb{E}_{p(z|x,\theta^{(m)})}\{\log p(x,z|\theta)\}
\end{aligned}
$$

fixed

The hidden variable z is averaged out (since unobserved): we get rid of it

Summarizing,

E-step Compute $Q(\theta, \theta^{(m)}) = \mathbb{E}_{p(z|x,\theta^{(m)})}\{\log p(x, Z|\theta)\}$

M-step $\theta^{(m+1)} = \underset{\theta}{\operatorname{argmax}}\ Q(\theta, \theta^{(m)})$.

EM ALGORITHM + initialization

E-step ≡ computing an Expectation
M-step ≡ computing a Maximization task.

Theorem: The log-likelihood increases at each iteration of the algorithm: $\ell(\theta^{(m)}) \leqslant \ell(\theta^{(m+1)})$. In other words, the EM algorithm converges to a local maximum.

proof = We derive expressions for $\ell(\theta^{(m)})$ and $\ell(\theta^{(m+1)})$, and compare them.

$$\searrow \ \ell(\theta^{(m)}) \underset{\substack{\uparrow \\ \text{equality since } q = q^{(m+1)}}}{=} F(q^{(m+1)}, \theta^{(m)}) = \sum_z p(z|x, \theta^{(m)}) \log\left\{\frac{p(x,z|\theta^{(m)})}{p(z|x,\theta^{(m)})}\right\}$$

$$\searrow \ \ell(\theta^{(m+1)}) \geqslant \sum_z \underset{\uparrow}{q(z)} \log\left\{\frac{p(x,z|\theta^{(m+1)})}{q(z)}\right\}$$

This inequality hold true ∀ distribution q; in particular for $q(z) = p(z|x, \theta^{(m)})$. Thus:

$$\geqslant \sum_z p(z|x,\theta^{(m)}) \log\left\{\frac{p(x,z|\theta^{(m+1)})}{p(z|x,\theta^{(m)})}\right\}$$

$$= \sum_z p(z|x,\theta^{(m)}) \log\{p(x,z|\theta^{(m+1)})\} - \sum_z p(z|x,\theta^{(m)}) \log\{p(z|x,\theta^{(m)})\}.$$

$$= Q(\theta^{(m+1)}, \theta^{(m)}) - \text{something independent of } \theta^{(m+1)}.$$

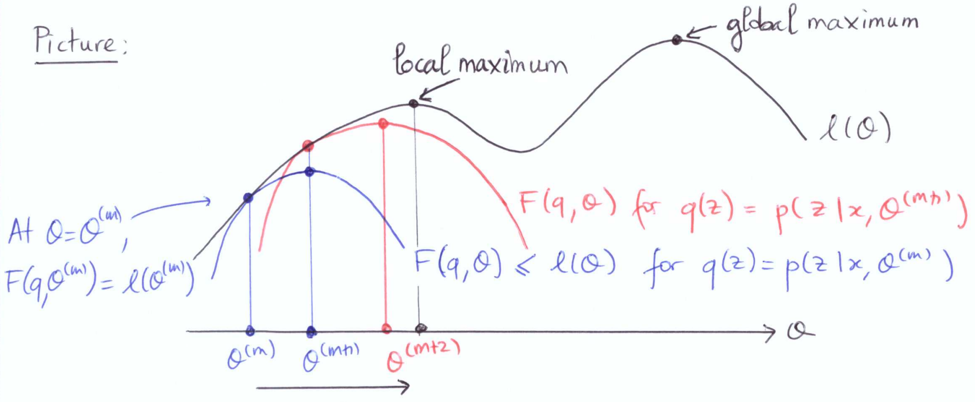
Since $\theta^{(m+1)} = \underset{\theta}{\operatorname{argmax}}\ Q(\theta, \theta^{(m)})$, we have that $Q(\theta^{(m)}, \theta^{(m)}) \leqslant Q(\theta^{(m+1)}, \theta^{(m)})$. Thus,

$$\ell(\theta^{(m+1)}) \geqslant Q(\theta^{(m)}, \theta^{(m)}) - \sum_z p(z|x,\theta^{(m)}) \log\{p(z|x,\theta^{(m)})\}$$

$$= \sum_z p(z|x,\theta^{(m)}) \log\left\{\frac{p(x,z|\theta^{(m)})}{p(z|x,\theta^{(m)})}\right\}$$

$$= \ell(\theta^{(m)}) \qquad \blacksquare$$

Picture:


global maximum
local maximum
$\ell(\theta)$
At $\theta = \theta^{(m)}$,
$F(q, \theta^{(m)}) = \ell(\theta^{(m)})$
$F(q, \theta)$ for $q(z) = p(z|x, \theta^{(m+1)})$
$F(q, \theta) \leqslant \ell(\theta)$ for $q(z) = p(z|x, \theta^{(m)})$
θ
$\theta^{(m)}$
$\theta^{(m+1)}$
$\theta^{(m+2)}$

Remarks = (i) For n iid observations, we need to compute

$$Q(\theta, \theta^{(m)}) = \sum_{i=1}^{n} \left\{ \sum_z p(z|x_i, \theta^{(m)}) \log p(x_i, z|\theta) \right\}$$

$$= \sum_{i=1}^{n} \mathbb{E}_{p(z|x_i,\theta^{(m)})}\{\log p(x_i, Z|\theta)\}$$

If Z is continuous, then

$$Q(\theta, \theta^{(m)}) = \sum_{i=1}^{n} \int f(z|x_i, \theta^{(m)}) \log p(x_i, z|\theta)\, dz$$

(ii) **EM and KL divergence**. We showed page 21 that

$$\ell(\theta) \geqslant \sum_z q(z) \log\left\{\frac{p(x,z|\theta)}{q(z)}\right\} = F(q, \theta)$$

We find the missing term on the right-hand-side to obtain $\ell(\theta) = F(q, \theta) + \text{something}$.

Since $p(x,z|\theta) = p(z|x,\theta)\, p(x|\theta)$,

$$F(q,\theta) = \sum_z q(z) \log\left\{\frac{p(z|x,\theta)\, p(x|\theta)}{q(z)}\right\}$$

$$= \sum_z q(z) \log\left\{\frac{p(z|x,\theta)}{q(z)}\right\} + \underbrace{\boxed{\sum_z q(z)}}_{=1} \underbrace{\log p(x|\theta)}_{=\ell(\theta)}$$

$$\Rightarrow \ell(\theta) = F(q,\theta) - \underbrace{\sum_z q(z) \log\left\{\frac{p(z|x,\theta)}{q(z)}\right\}}$$

= Kullback-Leibler divergence between $q(z)$ and $p(z|x,\theta) =: p_x(z)$ denoted $KL(q \,\|\, p_x)$

$$\boxed{\ell(\theta) = F(q,\theta) + KL(q \,\|\, p_x)} \text{ ——— } (**)$$

Recall that KL divergence is non-negative $KL(q \,\|\, p_x) \geq 0$, with equality if and only if $q = p_x$.

Therefore, we see that ← $\ell(\theta) \geq F(q,\theta)$ indeed.
When keeping θ fixed, the largest value of the lower bound is obtained when the KL divergence vanishes, that is for $q = p_x$.

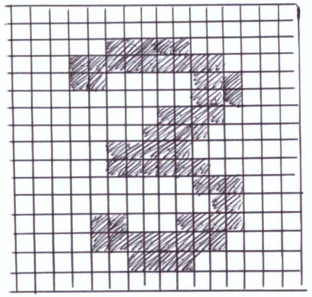
(iii) As indicated on the picture page 24, the curves of $\ell(\theta)$ and $F(q,\theta)$ [for $q(z) = p(z|x,\theta^{(m)})$] coincide at $\theta = \theta^{(m)}$. Moreover, the two curves have the same gradient, so that $\left.\frac{\partial \ell(\theta)}{\partial \theta}\right|_{\theta=\theta^{(m)}} = \left.\frac{\partial F(q,\theta)}{\partial \theta}\right|_{\theta=\theta^{(m)}}$.

This follows directly from the decomposition (**), and the fact that the KL diverge reaches its minimum 0 at $\theta = \theta^{(m)}$, so that its derivative vanishes at that point.

II.2. Application to clustering of binary images of handwritten digits.

We illustrate the use of the EM algorithm on a simple clustering example. Observations x correspond to binary images, representing handwritten digits '2', '3' and '4'. The dimension d of $x \in \mathbb{R}^d$ represents the number of pixels in the image, and each pixel can have a binary (0 or 1) value only.

We consider the clustering of a dataset $\mathcal{L}_n = \{x_1, \dots, x_n\}$ using $K = 3$ clusters.



Observation x

For an image $x = (x_1, \dots, x_d) \in \mathbb{R}^d$ belonging to cluster k, each variable x_i is assumed to follow a Bernoulli distribution with parameter p_{ik}, so that

$$p(x \mid p_k) = \prod_{j=1}^{d} p_{kj}^{x_j} (1 - p_{kj})^{1-x_j}, \quad x_j \in \{0,1\}$$

$p_k := (p_{k1}, \dots, p_{kd})$ $(k = 1, \dots, K)$

This model assumes independence between consecutive pixels.

$\Rightarrow$ Observations belonging to cluster k have mean p_k, and covariance matrix $\text{diag}\{p_k(1-p_k)\}$

We consider a mixture of these distributions, so that an image x is assumed to arise from the distribution

$$p(x \mid p, \pi) = \sum_{k=1}^{K} \pi_k\, p(x \mid p_k), \quad \sum_{k=1}^{K} \pi_k = 1, \quad 0 \leq \pi_k \leq 1$$

$\pi := (\pi_1, \dots, \pi_K)$

$p := (p_1, \dots, p_K)$

The log-likelihood of a dataset $\mathcal{L}_n = \{x_1, \dots, x_n\}$ is thus given by

$$\ell(\theta) = \log\left\{\prod_{i=1}^{n} p(x_i \mid p, \pi)\right\}$$

$$= \sum_{i=1}^{n} \log\left\{\sum_{k=1}^{K} \pi_k \, p(x_i \mid p_k)\right\}$$

Use the EM algorithm to maximize $\ell(\theta)$ numerically.

$\theta = (p, \pi)$

= set of all model parameters

- Augment the data set $\mathcal{L}_n$ by adding hidden variables $z_1, \dots, z_n$, corresponding to the cluster identity of observations $x_1, \dots, x_n$. We denote the COMPLETE DATASTET by $\mathcal{L}'_n = \{(x_1, z_1), \dots, (x_n, z_n)\}$. Each (x_i, z_i) is a realization of a generic (X, Z). To derive the E-step of the EM algorithm, we need to calculate the posterior distribution $p(z \mid x, \theta)$ [see the expression of $Q(\theta, \theta^{(m)})$ on the bottom of page 22]

$$p(z \mid x, \theta) := \mathbb{P}(Z = z \mid X = x, \theta) \quad , \quad z = 1, \dots, K$$

$$= \frac{\mathbb{P}(X = x \mid Z = z, \theta)\, \mathbb{P}(Z = z \mid \theta)}{\sum_{k=1}^{K} \mathbb{P}(X = x \mid Z = k, \theta)\, \mathbb{P}(Z = k \mid \theta)}$$

$$= \frac{\pi_z \, p(x \mid p_z)}{\sum_{k=1}^{K} \pi_k \, p(x \mid p_k)} .$$

Let $\theta^{(m)} = (p^{(m)}, \pi^{(m)})$; $p^{(m)} = (p_1^{(m)}, \dots, p_K^{(m)})$
$\pi^{(m)} = (\pi_1^{(m)}, \dots, \pi_K^{(m)})$
$p_k^{(m)} = (p_{k1}^{(m)}, \dots, p_{kd}^{(m)})$, $k = 1, \dots, K$

denote the current parameter estimates.

Then

$$Q(\theta, \theta^{(m)}) = \sum_{i=1}^{n} \left\{\sum_{z=1}^{K} p(z \mid x_i, \theta^{(m)}) \log p(x_i, z \mid \theta)\right\}$$

(page 24)

with
- $$p(z \mid x_i, \theta^{(m)}) = \frac{\pi_z^{(m)} \, p(x_i \mid p_z^{(m)})}{\sum_{k=1}^{K} \pi_k^{(m)} \, p(x_i \mid p_k^{(m)})} =: \hat{p}_{z,i}^{(m)}$$

  $$\left\{ \text{and } p(x_i \mid p_z^{(m)}) = \prod_{j=1}^{d} [p_{zj}^{(m)}]^{x_{ij}} (1 - p_{zj}^{(m)})^{1 - x_{ij}} \quad x_i = (x_{i1}, \dots, x_{id}) \in \mathbb{R}^d \right\}$$

- $$p(x_i, z \mid \theta) = p(x_i \mid z, \theta)\, p(z \mid \theta) = \pi_z \prod_{j=1}^{d} p_{zj}^{x_{ij}} (1 - p_{zj})^{1 - x_{ij}}$$

It follows that

$$Q(\theta, \theta^{(m)}) = \sum_{i=1}^{n} \left\{\sum_{z=1}^{K} \hat{p}_{z,i}^{(m)} \log\left(\pi_z \prod_{j=1}^{d} p_{zj}^{x_{ij}} (1 - p_{zj})^{1 - x_{ij}}\right)\right\}$$

$$Q(\theta, \theta^{(m)}) = \sum_{i=1}^{n} \sum_{z=1}^{K} \hat{p}_{z,i}^{(m)} \log \pi_z + \sum_{i=1}^{n} \sum_{z=1}^{K} \hat{p}_{z,i}^{(m)} \sum_{j=1}^{d} \log\left\{p_{zj}^{x_{ij}} (1 - p_{zj})^{1 - x_{ij}}\right\}$$

- M-step: update the current estimate $\theta^{(m)}$ = we need to find the gradient of Q with respect to θ [variables $p_{z,i}^{(m)}$ are thus treated as constants].

→ Maximization with respect to p_{jz}

$$\frac{\partial Q(\theta, \theta^{(m)})}{\partial p_{zj}} = \sum_{i=1}^{n} \hat{p}_{z,i}^{(m)} \left\{\frac{x_{ij}}{p_{zj}} - \frac{1 - x_{ij}}{1 - p_{zj}}\right\}$$

$j = 1, \dots, d$
$z = 1, \dots, K$

Set this derivative to zero and solve for p_{zj} to obtain $p_{zj}^{(m+1)}$.

We find that $\left(1 - p_{zj}^{(m+1)}\right) \sum_{i=1}^{n} \hat{p}_{z,i}^{(m)} x_{ij} = p_{zj}^{(m+1)} \sum_{i=1}^{n} \hat{p}_{z,i}^{(m)} (1 - x_{ij})$

After simplifications,
$$p_{zj}^{(m+1)} = \frac{\sum_{i=1}^{n} \hat{p}_{z,i}^{(m)} x_{ij}}{\sum_{i=1}^{n} \hat{p}_{z,i}^{(m)}}, \quad j = 1, -, d \quad z = 1, -, K$$

In the notation $p_z^{(m+1)} = \left(p_{zj}^{(m+1)}, \dots, p_{zj}^{(m+1)}\right)$, $z = 1, -, K$

$x_i = (x_{i1}, \dots, x_{id})$, we obtain

$$\boxed{p_z^{(m+1)} = \frac{\sum_{i=1}^{n} \hat{p}_{z,i}^{(m)} x_i}{\sum_{i=1}^{n} \hat{p}_{z,i}^{(m)}}, \quad z = 1, -, K}$$

In words =

$\hat{p}_{z,i}^{(m)} \approx$ weight / belief that observation x_i belongs to cluster z, at the m-th iteration

$\Rightarrow p_z^{(m+1)} \approx$ soft average of observations $x_1, \dots, x_n$, with heavier weights on observations thought to belong to cluster z.

→ Maximization with respect to π_z. Maximization must be performed under the constraint that $\sum_{z=1}^{K} \pi_z = 1$. Introduce Lagrange multiplier λ & the Lagrangian function

$$\mathcal{L}(\pi) = \sum_{i=1}^{n} \sum_{z=1}^{K} \hat{p}_{z,i}^{(m)} \log \pi_z + \lambda \left(\sum_{z=1}^{K} \pi_z - 1 \right).$$

$$\frac{\partial \mathcal{L}(\pi)}{\partial \pi_z} = \sum_{i=1}^{n} \frac{\hat{p}_{z,i}^{(m)}}{\pi_z^{(m+1)}} + \lambda = 0$$

We conclude that $\pi_z^{(m+1)}$ must be proportional to $\sum_{i=1}^{n} \hat{p}_{z,i}^{(m)}$,

for all z : $\pi_z^{(m+1)} = \propto \sum_{i=1}^{n} \hat{p}_{z,i}^{(m)}$

They sum to one $\Rightarrow$ $\pi_z^{(m+1)} = \dfrac{\sum_{i=1}^{n} \hat{p}_{z,i}^{(m)}}{\underbrace{\sum_{z=1}^{K} \sum_{i=1}^{n} \hat{p}_{z,i}^{(m)}}_{= n}} = \dfrac{1}{n} \sum_{i=1}^{n} \hat{p}_{z,i}^{(m)}.$

(look at the expression of $\hat{p}_{z,i}^{(m)}$ top of page 28)

× EM algorithm for Bernoulli data:

(i) Initialize parameters $\pi_k^{(1)}$, $p_k^{(1)} = (p_{k1}^{(1)}, \dots, p_{kd}^{(1)})$, $k = 1, -, K$

(ii) Repeat until convergence $(m = 1, 2, \dots)$

E-Step

- $$\hat{p}_{k,i}^{(m)} = \frac{\pi_k^{(m)} p(x_i \mid p_k^{(m)})}{\sum_{z=1}^{K} \pi_z^{(m)} p(x_i \mid p_z^{(m)})}, \quad k = 1, \dots, K \quad i = 1, \dots, n$$

  with $p(x_i \mid p_k^{(m)}) = \prod_{j=1}^{d} \left[p_{kj}^{(m)}\right]^{x_{ij}} \left[1 - p_{kj}^{(m)}\right]^{1 - x_{ij}}$

M-Step

- $$\pi_k^{(m+1)} = \frac{1}{n} \sum_{i=1}^{n} \hat{p}_{k,i}^{(m)}$$
- $$p_k^{(m+1)} = \frac{\sum_{i=1}^{n} \hat{p}_{k,i}^{(m)} x_i}{\sum_{i=1}^{n} \hat{p}_{k,i}^{(m)}}$$

EM ALGORITHM

II.3. EM algorithm for Gaussian Mixtures.

Back to our original maximization problem:

$$\hat{\theta} \in \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} p_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k) \right)$$

Introduce the lattent variables $z_1, \dots, z_n$, and the complete dataset $\mathcal{L}_n' = \{(x_1, z_1), \dots, (x_n, z_n)\}$. The variable z_i denotes the cluster identity of observation x_i: formally, $z_i \in \{1, \dots, K\}$ and $\{z_i = k\}$ indicates that x_i was generated from $\mathcal{N}(x \mid \mu_k, \Sigma_k)$.

- E-step. We need to compute
$$Q(\theta, \theta^{(m)}) = \sum_{i=1}^{n} \mathbb{E}_{p(z \mid x_i, \theta^{(m)})} \left\{ \log p(x_i, z \mid \theta) \right\}.$$

⇒ We need to derive the posterior distribution $p(z|x,\Theta)$.

Bayes formula + law of Total Probability (LTP) give:

$$p(z|x,\Theta) = \mathbb{P}(Z=z \mid X=x, \Theta)$$

$$= \frac{\mathbb{P}(X\in dx \mid Z=z,\Theta)\,\mathbb{P}(Z=z\ \Theta)}{\sum_{k=1}^{K} \mathbb{P}(X\in dx \mid Z=k,\Theta)\,\mathbb{P}(Z=k \mid \Theta)}$$

$$p(z|x,\Theta) = \frac{p_z \mathcal{N}(x \mid \mu_z, \Sigma_z)}{\sum_{k=1}^{K} p_k \mathcal{N}(x \mid \mu_k, \Sigma_k)}$$

↓

We denote

$$\hat{p}_{z,i}^{(m)} := \mathbb{P}(Z=z \mid X=x_i, \Theta^{(m)})$$

$$= \frac{p_z^{(m)} \mathcal{N}(x_i \mid \mu_z^{(m)}, \Sigma_z^{(m)})}{\sum_{k=1}^{K} p_k^{(m)} \mathcal{N}(x_i \mid \mu_k^{(m)}, \Sigma_k^{(m)})}$$

↖ Where $\Theta^{(m)}$ = current parameter estimates

$\overset{\shortparallel}{\{p^{(m)}, \mu^{(m)}, \Sigma^{(m)}\}}$,

with $p^{(m)} = \{p_1^{(m)}, -, p_K^{(m)}\}$

$\mu^{(m)} = \{\mu_1^{(m)}, -, \mu_K^{(m)}\}$

$\Sigma^{(m)} = \{\Sigma_1^{(m)}, -, \Sigma_K^{(m)}\}$

We plug $\hat{p}_{z,i}^{(m)}$ back into the expression of $Q(\Theta, \Theta^{(m)})$,

$$Q(\Theta,\Theta^{(m)}) = \sum_{i=1}^{n}\sum_{k=1}^{k} \hat{p}_{k,i}^{(m)} \log\{p(x_i, k \mid \Theta)\}$$

↖ complete log likelihood.

$$Q(\Theta,\Theta^{(m)}) = \sum_{i=1}^{n}\sum_{k=1}^{K} \hat{p}_{k,i}^{(m)} \log\{p_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k)\}$$

$$= \underbrace{\sum_{i=1}^{n}\sum_{k=1}^{K} \hat{p}_{k,i}^{(m)} \log p_k}_{\text{I}} + \underbrace{\sum_{i=1}^{n}\sum_{k=1}^{K} \hat{p}_{k,i}^{(m)} \log \mathcal{N}(x_i \mid \mu_k, \Sigma_k)}_{\text{II}}$$

- M-step. We maximize $Q(\Theta, \Theta^{(m)})$ with respect to p, μ, and Σ separately.

I term. Find
$$\boxed{(p_1^{(m+1)}, -, p_K^{(m+1)}) = \underset{p_1,-,p_K}{\operatorname{argmax}} \sum_{i=1}^{n}\sum_{k=1}^{K} \hat{p}_{k,i}^{(m)} \log p_k \quad \text{subject to } \sum_{k=1}^{K} p_k = 1.}$$

Proceed as on page 29 for the maximization of π_z: introduce the Lagrange multiplier λ, and consider the Lagrangian function

$$\mathcal{L}(p) = \sum_{i=1}^{n}\sum_{k=1}^{K} \hat{p}_{k,i}^{(m)} \log p_k + \lambda\left(\sum_{k=1}^{K} p_k - 1\right)$$

Differentiating $\mathcal{L}$ with respect to p_k gives

$$\frac{\partial \mathcal{L}(p)}{\partial p_k} = \sum_{i=1}^{n} \frac{\hat{p}_{k,i}^{(m)}}{p_k} + \lambda.$$

Setting this derivative to zero shows that the solution $p_k^{(m+1)}$ is proportional to $\sum_{i=1}^{n} \hat{p}_{k,i}^{(m)}$ for all k. Since the $p_k^{(m+1)}$ must sum to one, we conclude that

$$p_k^{(m+1)} = \frac{\sum_{i=1}^{n} \hat{p}_{k,i}^{(m)}}{\underbrace{\sum_{k=1}^{K}\sum_{i=1}^{n} \hat{p}_{k,i}^{(m)}}_{= n \text{ since } \sum_{k=1}^{K} \hat{p}_{k,i}^{(m)} = 1}} = \frac{1}{n}\sum_{i=1}^{n} \hat{p}_{k,i}^{(m)}.$$

Summarizing, $\boxed{p_k^{(m+1)} = \frac{1}{n}\sum_{i=1}^{n} \hat{p}_{k,i}^{(m)}, \quad \text{for } k=1,-,K.}$

Ⅱ term. First, consider maximization with respect to μ_k:

Maximizing Ⅱ with respect to μ_k ⟺

Maximizing $\displaystyle\sum_{i=1}^{n}\sum_{k=1}^{K}\hat{p}_{k,i}^{(m)}\left\{-\frac{1}{2}(x_i-\mu_k)^t\Sigma_k^{-1}(x_i-\mu_k)\right\}$

$\dfrac{\partial(\ldots)}{\partial\mu_k}$ is proportional to $\displaystyle\sum_{i=1}^{n}\hat{p}_{k,i}^{(m)}(x_i-\mu_k^{(m+1)})^t\Sigma_k^{-1}=0$

Multiply this expression by Σ_k : $\displaystyle\sum_{i=1}^{n}\hat{p}_{k,i}^{(m)}(x_i-\mu_k^{(m+1)})=0$,

and we find

$$\boxed{\mu_k^{(m+1)}=\frac{\sum_{i=1}^{n}\hat{p}_{k,i}^{(m)}x_i}{\sum_{i=1}^{n}\hat{p}_{k,i}}}\quad\text{, for } k=1,-,K.$$

We turn our attention to the maximization of Ⅱ with respect to Σ_k:

$$\Leftrightarrow \text{Maximize } \sum_{i,k}\hat{p}_{k,i}^{(m)}\left\{-\frac{1}{2}\underbrace{(x_i-\mu_k^{(m+1)})^t\Sigma_k^{-1}(x_i-\mu_k^{(m+1)})}_{\in\mathbb{R}\text{, so must be equal to its TRACE.}}\right\}-\frac{1}{2}\sum_{i,k}\hat{p}_{k,i}^{(m)}\log\det\Sigma_k.$$

$$=-\frac{1}{2}\left\{\sum_{i,k}\hat{p}_{k,i}^{(m)}\,\mathrm{Tr}\left(\Sigma_k^{-1}(x_i-\mu_k^{(m+1)})(x_i-\mu_k^{(m+1)})^t\right)+\sum_{i,k}\hat{p}_{k,i}^{(m)}\log\det\Sigma_k\right\}$$

since Tr(AB) = Tr(BA) for two matrices that permute.

$$=-\frac{1}{2}\left\{\sum_{k=1}^{K}\mathrm{Tr}\left(\Sigma_k^{-1}\sum_{i=1}^{n}\hat{p}_{k,i}^{(m)}(x_i-\mu_k^{(m+1)})(x_i-\mu_k^{(m+1)})^t\right)+\sum_{k=1}^{K}\log\det\Sigma_k\underbrace{\left(\sum_{i=1}^{n}\hat{p}_{k,i}^{(m)}\right)}_{\text{denote this } \hat{p}_k^{(m)}}\right\}$$

Put $\displaystyle S_k^{(m)}:=\sum_{i=1}^{n}\hat{p}_{k,i}^{(m)}(x_i-\mu_k^{(m+1)})(x_i-\mu_k^{(m+1)})^t\equiv$ sample covariance matrix.

⟺ Maximize

$$-\frac{1}{2}\left\{\sum_{k=1}^{K}\mathrm{Tr}\left(\Sigma_k^{-1}S_k^{(m)}\right)+\sum_{k=1}^{K}\hat{p}_k^{(m)}\log\det\Sigma_k\right\}$$

Toolbox:

$$\frac{\partial}{\partial A}\mathrm{Tr}(AB)=\frac{\partial}{\partial A}\mathrm{Tr}(BA)=B^t$$

$$\frac{\partial}{\partial A}\log\det A=(A^{-1})^t$$

$$\det A^{-1}=\frac{1}{\det A}.$$

We take derivatives with respect to Σ_k^{-1}:

$$\frac{\partial(\ldots)}{\partial\Sigma_k^{-1}}=S_k^{(m)}-\hat{p}_k^{(m)}\Sigma_k^{(m+1)}=0\;\Rightarrow\;\Sigma_k^{(m+1)}=\frac{S_k^{(m)}}{p_k^{(m)}}.$$

Conclusion:

EM ALGORITHM FOR GAUSSIAN MIXT.

(i) Initialize parameters $p_k^{(1)}$, $\mu_k^{(1)}$, $\Sigma_k^{(1)}$.

(ii) Repeat, for $m=1,2,\ldots$

E-step

- $\displaystyle\hat{p}_{k,i}^{(m)}=\frac{p_k^{(m)}\mathcal{N}(x_i\mid\mu_k^{(m)},\Sigma_k^{(m)})}{\sum_{j=1}^{K}p_j^{(m)}\mathcal{N}(x_i\mid\mu_j^{(m)},\Sigma_j^{(m)})}$, $k=1,-,K$, $i=1,-,n$

M-step

- $\displaystyle p_k^{(m+1)}=\frac{1}{n}\sum_{i=1}^{n}\hat{p}_{k,i}^{(m)}$
- $\displaystyle\mu_k^{(m+1)}=\frac{\sum_{i=1}^{n}\hat{p}_{k,i}^{(m)}x_i}{\sum_{i=1}^{n}\hat{p}_{k,i}^{(m)}}$
- $\displaystyle\Sigma_k^{(m+1)}=\frac{\sum_{i=1}^{n}\hat{p}_{k,i}^{(m)}(x_i-\mu_k^{(m+1)})(x_i-\mu_k^{(m+1)})^t}{\sum_{i=1}^{n}\hat{p}_{k,i}^{(m)}}$

Interpretation:

Recall the definition of $\hat{p}_{k,i}^{(m)} = \mathbb{P}(Z=k \mid X=x_i, \theta = \theta^{(m)})$

$$\Rightarrow \sum_{i=1}^{n} \hat{p}_{k,i}^{(m)} = \mathbb{E}\left\{ \sum_{i=1}^{n} \mathbb{1}(Z=k \mid X=x_i, \theta=\theta^{(m)}) \right\}$$

↑ This term, appearing in the updates of $p_k^{(m+1)}, \mu_k^{(m+1)}, \Sigma_k^{(m+1)}$, represents the expected number of observations believed to originate from the k-th cluster, at iteration m.

We have a soft assignment of observation x_i to cluster k, given by $\hat{p}_{k,i}^{(m)} \in [0,1]$, compared to the K-mean algorithm, which returns a hard assignment of observation x_i, given by $\pi_{ik} \in \{0,1\}$.

Moreover, $\mu_k^{(m+1)}$ = weighted average of observations x_i, the higher the weight $\hat{p}_{k,i}^{(m)}$, the more likely x_i belongs to the k-th cluster.

likewise $\Sigma_k^{(m+1)}$ = weighted sample covariance matrix.

Now, compare the E and M steps of the algorithm, with the K-means algorithm:

E step ≡ Assigning x_i to its closest centroid

M step ≡ Recomputing the centroid of each cluster.

We can push the analogy between the K-means algorithm and the EM algorithm for Gaussian mixtures a bit further...

II. 4. Relation to K-means.

Consider a Gaussian Mixture Model with known covariance matrices $\Sigma_k = \varepsilon I_d$, where
- $\varepsilon > 0$ some small positive number
- $I_d = d \times d$ identity matrix.

$(\forall k)$

Assume ε fixed.

$\Rightarrow$ Parameters to be estimated $= \{\mu_1, \dots, \mu_K, p_1, \dots, p_K\}$

The EM algorithm is (see page 34)

$$\hat{p}_{k,i}^{(m)} = \frac{p_k^{(m)} \mathcal{N}(x_i \mid \mu_k^{(m)}, \varepsilon I_d)}{\sum_{j=1}^{K} p_j^{(m)} \mathcal{N}(x_i \mid \mu_j^{(m)}, \varepsilon I_d)}$$

$$p_k^{(m+1)} = \frac{1}{n} \sum_{i=1}^{n} \hat{p}_{k,i}^{(m)}$$

$$\mu_k^{(m+1)} = \sum_{i=1}^{n} \hat{p}_{k,i}^{(m)} x_i \Big/ \sum_{i=1}^{n} \hat{p}_{k,i}^{(m)}$$

$$\hat{p}_{k,i}^{(m)} = \frac{p_k^{(m)} \exp\left(-\frac{1}{2\varepsilon} \| x_i - \mu_k^{(m)} \|^2\right)}{\sum_{j=1}^{K} p_j^{(m)} \exp\left(-\frac{1}{2\varepsilon} \| x_i - \mu_j^{(m)} \|^2\right)}$$

$$= \frac{1}{1 + \sum_{j \neq k} \frac{p_j^{(m)}}{p_k^{(m)}} \exp\left(-\frac{1}{2\varepsilon}\left[\| x_i - \mu_j^{(m)} \|^2 - \| x_i - \mu_k^{(m)} \|^2\right]\right)}$$

- If x_i is closer to $\mu_k^{(m)}$ than any other $\mu_j^{(m)}$, $j \neq k$, the term $[\dots]$ is positive, and $\exp\left(-\frac{1}{2\varepsilon}[\dots]\right) \longrightarrow 0$ as $\varepsilon \to 0$, so that $\hat{p}_{k,i}^{(m)} \xrightarrow[\varepsilon \to 0]{} 1$ as $\varepsilon \to 0$.
- Alternatively, if x_i is closer to some $\mu_j^{(m)}$, $j \neq k$, than it is to $\mu_k^{(m)}$, then the term $\left[\| x_i - \mu_j^{(m)} \|^2 - \| x_i - \mu_k^{(m)} \|^2\right] < 0$, and $\exp\left(-\frac{1}{2\varepsilon}[\dots]\right) \to \infty$ as $\varepsilon \to 0$, so that $\hat{p}_{k,i}^{(m)} \longrightarrow 0$ as $\varepsilon \to 0$.

Summarising,

- $\hat{p}_{k,i}^{(m)} \xrightarrow{\varepsilon \to 0} \underset{\text{Variable introduced page 4 for the K-means optimization problem.}}{\pi_{ik}} = \begin{cases} 1 & \text{if } \underset{\text{In other words, observation } x_i \text{ belongs to the k-th cluster}}{\|x_i - \mu_k^{(m)}\| < \|x_i - \mu_j^{(m)}\|} \quad \forall j \neq k \\ 0 & \text{otherwise} \end{cases}$

In addition,

- $\sum_{i=1}^{n} \hat{p}_{k,i}^{(m)} \longrightarrow n_k^{(m)} = \sum_{i=1}^{n} \pi_{ik}$ = number of observations in the k-th cluster.

- $p_k^{(m+1)} \longrightarrow \frac{n_k^{(m)}}{n}$ = proportion of points belonging to the k-th cluster.

- $\mu_k^{(m+1)} \longrightarrow \underset{\substack{\| \\ m_k^{(m)} \text{ defined on page 7}}}{\frac{1}{n_k^{(m)}} \sum_{i=1}^{n} \pi_{ik} x_i}$ = mean value of the points belonging to the k-th cluster.

⇒ We recover the K-means algorithm.

<u>Remark</u>: Recall the expression of $Q(\theta, \theta^{(m)})$ on the top of page 32:

$$Q(\theta, \theta^{(m)}) = \sum_{i,k} \hat{p}_{k,i}^{(m)} \log \left\{ \mathcal{N}(x_i \mid \mu_k, \varepsilon I_d) \right\} + \textcircled{I}$$

$$= \sum_{i,k} \hat{p}_{k,i}^{(m)} \left(-\frac{1}{2\varepsilon}\right) \|x_i - \mu_k\|^2 + \text{extra terms independent of } \mu.$$

$$\text{minimizing} \Leftrightarrow \sum_{i,k} \hat{p}_{k,i}^{(m)} \|x_i - \mu_k\|^2$$

$$\approx \sum_{i,k} \pi_{ik} \|x_i - \mu_k\|^2 = \text{cost function of the K-means algorithm.}$$

⇒ In the limit $\varepsilon \to 0$, maximizing the expected complete log-likelihood is equivalent to minimizing the cost function of the K-means algorithm.

K-means is a non-probabilistic clustering method; and does not allow variance estimation. It can be used as an initialization of the EM algorithm of the Gaussian Mixture Model.

III _ FUZZY C-MEANS

The Fuzzy C-means algorithm represents a compromize between a <u>soft</u> (<u>probabilistic</u>) assignment of observations in a GMM, and a <u>hard</u> (<u>non-probabilistic</u>) assignment of observations of the K-means algorithm by allowing a <u>soft</u> <u>non-probabilistic</u> assignment.

- The cost function of the K-means algorithm is (page 4)

$$\mathcal{C}_0(M, \pi) = \sum_{k=1}^{K} \sum_{i=1}^{n} \pi_{ik} \|x_i - m_k\|_2^2,$$

where $\pi_{ik} = \begin{cases} 1 & \text{if } x_i \text{ belongs to cluster } k \\ 0 & \text{otherwise.} \end{cases}$, $\sum_{k=1}^{K} \pi_{ik} = 1$

- We relax the constraint $\pi_{ik} \in \{0, 1\}$, and we suppose instead that $\pi_{ik} \in [0, 1]$.

The objective of fuzzy clustering is then

$$\mathcal{C}_m(M, u) = \sum_{k=1}^{K} \sum_{i=1}^{n} (u_{ik})^m \|x_i - m_k\|_2^2,$$

where $\begin{cases} \cdot\ m \geq 1 \\ \cdot\ u_{ik} \in [0, 1] \ ; \quad u = \{u_{ik}\} \\ \cdot\ \sum_{k=1}^{K} u_{ik} = 1 \\ \cdot\ 0 < \sum_{k=1}^{K} u_{ik} < n \end{cases}$

The u_{ik} can be interpreted as a probability.

- Each cluster has at least one observation ($\sum_k u_{ik} > 0$): <u>NO SUPER WEAK CLUSTER</u>.
- There is <u>no</u> cluster which gets allocated all observations with full weight equal to 1: <u>NO SUPER STRONG CLUSTER</u>.

m = parameter set by the practitioner (role of m discussed page 41)

⇒ There exists an infinite number of fuzzy clustering algorithms, one for each m.

Theorem = A necessary condition for (M, u) to be a global minimum of $\mathcal{C}_m$ is that

① $$m_k = \frac{\sum_{i=1}^{n} (u_{ik})^m x_i}{\sum_{i=1}^{n} (u_{ik})^m} \quad \text{for } k = 1, -, K.$$

② $$u_{ik} = \left[\sum_{j=1}^{K} \left(\frac{\| x_i - m_k \|_2}{\| x_i - m_j \|_2} \right)^{\frac{2}{m-1}} \right]^{-1}, \quad \begin{array}{l} i = 1, -, n \\ k = 1, -, K. \end{array}$$

Assuming that none of the x_i is equal to one of the centers m_j. If $x_i = m_j$ for some j, there is a way to adapt to this situation. However, this rarely occurs in practice due to machine roundoff.

proof = • First, fix u_{ik}. Then

$$\frac{\partial \mathcal{C}_m}{\partial m_j} = \frac{\partial}{\partial m_j} \sum_{i,k} (u_{ik})^m \| x_i - m_k \|_2^2$$
$$= 2 \sum_{i=1}^{n} (u_{ij})^m (m_j - x_i) = 0$$

And we see that m_j must satisfy $m_j = \sum (u_{ij})^m x_i \,/\, \sum (u_{ij})^m$.

• Next, we fix m_k.

We want to minimize $\sum_{i,k} (u_{ik})^m \| x_i - m_k \|_2^2$ with respect to the u_{ik}, subject to $\sum_k u_{ik} = 1$, for all i.

Introduce Lagrange multipliers $\lambda := (\lambda_1, .., \lambda_n)$, and the Lagrange function

$$\mathcal{L}(\lambda, u) = \sum_{i=1}^{n} \Big\{ \sum_{k=1}^{K} (u_{ik})^m \| x_i - m_k \|_2^2 - \underset{\uparrow}{\lambda_i} \Big(\sum_k u_{ik} - 1 \Big) \Big\}$$

No restriction on the sign of λ_i, since we have an equality constraint.

$$=: \sum_{i=1}^{n} \mathcal{L}_i(\lambda_i, \underbrace{u_{i1}, -, u_{iK}}_{\text{Put } u_i = (u_{i1}, -, u_{iK})})$$

We are cheating a little bit since the condition $\sum_i u_{ik} < n$ tells us that we cannot separate $u_1, -, u_n$ and optimize each term individually.
⇒ Relax this condition and allow $\sum_i u_{ik} \leq n$.

(λ_i, u_i) is a stationary point for $\mathcal{L}_i$ if and only if $\nabla_{\lambda_i, u_i} \mathcal{L}_i(\lambda_i, u_i) = 0$

$$\begin{cases} \dfrac{\partial \mathcal{L}_i}{\partial \lambda_i}(\lambda_i, u_i) = \displaystyle\sum_{k=1}^{K} u_{ik} - 1 = 0 & \text{———} \ (1) \\ \dfrac{\partial \mathcal{L}_i}{\partial u_{ik}}(\lambda_i, u_i) = m\, u_{ik}^{m-1} \| x_i - m_k \|_2^2 - \lambda_i = 0 & \text{———} \ (2) \end{cases}$$

From (2) we get $$u_{ik} = \left[\frac{\lambda_i}{m \| x_i - m_k \|_2^2} \right]^{\frac{1}{m-1}} \quad \text{———} \ (3)$$

Plugging this expression back into (1), we get

$$1 = \sum_{k=1}^{K} u_{ik} = \sum_{k=1}^{K} \left[\frac{\lambda_i}{m \| x_i - m_k \|_2^2} \right]^{\frac{1}{m-1}}$$
$$= \left(\frac{\lambda_i}{m} \right)^{\frac{1}{m-1}} \sum_{k=1}^{K} \left(\frac{1}{\| x_i - m_k \|_2^2} \right)^{\frac{1}{m-1}}$$

So that $$\left(\frac{\lambda_i}{m} \right)^{\frac{1}{m-1}} = \left[\sum_{k=1}^{K} \left(\frac{1}{\| x_i - m_k \|_2^2} \right)^{\frac{1}{m-1}} \right]^{-1}$$

We eliminate the term $\left(\frac{\lambda_i}{m} \right)^{\frac{1}{m-1}}$ in the expression of u_{ik} in (3), and obtain

$$u_{ik} = \left(\frac{1}{\| x_i - m_k \|_2^2} \right)^{\frac{1}{m-1}} \left[\sum_{j=1}^{K} \left(\frac{1}{\| x_i - m_j \|_2^2} \right)^{\frac{1}{m-1}} \right]^{-1}$$
$$= \left[\sum_{j=1}^{K} \left(\frac{\| x_i - m_k \|_2}{\| x_i - m_j \|_2} \right)^{\frac{2}{m-1}} \right]^{-1}, \text{ as required} \quad ■$$

The fuzzy C-means algorithms simply circle through the necessary conditions of this theorem.

FUZZY C-MEANS

$(m>1)$

Initialization $u_{11}^{(0)}, \dots, u_{nK}^{(0)}$

Repeat for $\ell = 0, 1, 2, \dots$

- Calculate the fuzzy centers

$$m_k^{(\ell)} = \frac{\sum_{i=1}^{n} (u_{ik}^{(\ell)})^m x_i}{\sum_{i=1}^{n} (u_{ik}^{(\ell)})^m}, \quad \text{for } k=1,-,K$$

- Update the weights

$$u_{ik}^{(\ell+1)} = \left[\sum_{j=1}^{K} \left(\frac{\|x_i - m_k^{(\ell)}\|_2}{\|x_i - m_j^{(\ell)}\|_2}\right)^{\frac{2}{m-1}}\right]^{-1}$$

- Compare $u^{(\ell)}$ and $u^{(\ell+1)}$. If $\|u^{(\ell)} - u^{(\ell+1)}\| < \varepsilon$, stop.

See Bezdek (1981) - Pattern Recognition with fuzzy objective function algorithms.

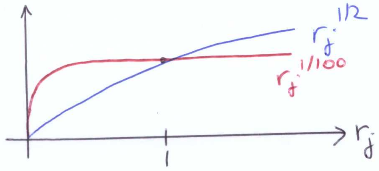
Remark: As • $m \to \infty$, the "fuzzier" the membership assignment
• $m \to 1$, solutions become "hard".

There is however no theoretical basis for an optimal choice of m.

To see this, put $r_j := \dfrac{\|x_i - m_k\|_2}{\|x_i - m_j\|_2}$

Then $u_{ik} = \dfrac{1}{1 + \sum_{j \neq k} (r_j)^{\frac{2}{m-1}}}$

→ For m large, the term $(r_j)^{\frac{2}{m-1}}$ is close to one, irrespectively of the value of $r_j > 0$ ⇒ The denominator in the expression of u_{ik} is approximately equal to K, so that $u_{ik} \approx \frac{1}{K}$



⇒ Assignment of observations is more fuzzy for m large

→ For m small, if $r_j \ll 1 \quad \forall j \neq k$, so that x_i is closer to m_k than any other m_j, $j \neq k$, then $r_j^{2/m-1}$ remains close to zero, so that u_{ik} is close to one.

⇒ Assignment of observation is harder for m close to 1.

IV - ONLINE VERSIONS.

We consider an on-line version of the EM algorithm: in the E-step, instead of computing/updating $\hat{p}_{k,i}$ [for the Gaussian Mixture Model] for all observations ("batch mode"), update $\hat{p}_{k,j}$ for one observation x_j only, and update p_k, μ_k and Σ_k in the M-step.

Notation: Let • $\hat{p}_{k,i}^{old}$ $\quad k=1,-,K$, $i=1,-,n$

• $p_k^{old}, \mu_k^{old}, \Sigma_k^{old}$ be the current estimates.

The value of the posterior distribution is then updated for observation x_j only, giving $p_{k,j}^{new}$. The M-step becomes:

→ Writing $n_k^{old} = \sum_{i=1}^{n} \hat{p}_{k,i}^{old}$

→ $n_k^{new} = n_k^{old} + \hat{p}_{k,j}^{new} - \hat{p}_{k,j}^{old}$

→ $p_k^{new} = p_k^{old} + \dfrac{\hat{p}_{k,j}^{new} - \hat{p}_{k,j}^{old}}{n}$

→ $$\mu_k^{new} = \frac{\sum_{i \neq j}^{n} \hat{p}_{k,i}^{old} x_i + \hat{p}_{k,j}^{new} x_j}{n_k^{new}}$$

$$= \frac{\sum_{i=1}^{n} \hat{p}_{k,i}^{old} x_i + \hat{p}_{k,j}^{new} x_j - \hat{p}_{k,j}^{old} x_j}{n_k^{new}}$$

$$\mu_k^{new} = \frac{n_k^{old}}{n_k^{new}}\mu_k^{old} + \frac{\hat{p}_{k,j}^{new}\, x_j}{n_k^{new}} - \frac{\hat{p}_{k,j}^{old}\, x_j}{n_k^{new}}$$

$$= \mu_k^{old} + \left(\frac{\hat{p}_{k,j}^{new} - \hat{p}_{k,j}^{old}}{n_k^{new}}\right)(x_j - \mu_k^{old})$$

obtained by writing $\frac{n_k^{old}}{n_k^{new}} = 1 + \left(\frac{n_k^{old}}{n_k^{new}} - 1\right) = 1 + \frac{n_k^{old} - n_k^{new}}{n_k^{new}}$, and noticing that $n_k^{old} - n_k^{new} = \hat{p}_{k,j}^{old}\, x_j - \hat{p}_{k,j}^{new}\, x_j$.

$$\rightarrow \Sigma_k^{new} = \frac{\sum_{i=1}^{n} \hat{p}_{k,i}^{old}(x_i - \mu_k^{\boxed{old}})(x_i - \mu_k^{old})^t + \hat{p}_{k,j}^{new}(x_j - \mu_k^{new})(x_j - \mu_k^{new})^t}{n_k^{new}} \cdots$$

$$\cdots - \frac{\hat{p}_{k,j}^{old}(x_j - \mu_k^{old})(x_j - \mu_k^{old})^t}{n_k^{new}}$$

note that we need to take the old value of μ_k here, otherwise we cannot obtain a nice recursion for Σ_k.

$$= \frac{n_k^{old}}{n_k^{new}}\Sigma_k^{old} + \frac{1}{n_k^{new}}\left\{ \begin{array}{l} \hat{p}_{k,j}^{new}(x_j - \mu_k^{new})(x_j - \mu_k^{new})^t \\ -\hat{p}_{k,j}^{old}(x_j - \mu_k^{old})(x_j - \mu_k^{old})^t \end{array} \right\}$$

$$= \Sigma_k^{old} + \frac{\hat{p}_{k,j}^{new}}{n_k^{new}}\left\{(x_j - \mu_k^{new})(x_j - \mu_k^{new})^t - \Sigma_k^{old}\right\}$$

$$- \frac{\hat{p}_{k,j}^{old}}{n_k^{new}}\left\{(x_j - \mu_k^{old})(x_j - \mu_k^{old})^t - \Sigma_k^{old}\right\}.$$

INCREMENTAL EM ALGORITHM FOR GMM

(i) Initialise parameters p_k^{old}, μ_k^{old}, Σ_k^{old}, $\hat{p}_{k,i}^{old} = \frac{1}{K}$

$$n_k^{old} = \sum_{i=1}^{n} \hat{p}_{k,i}^{old}$$

(ii) For $j = 1, -, n$

- $\hat{p}_{k,j}^{new} = \dfrac{p_k^{old}\,\mathcal{N}(x_j \mid \mu_k^{old}, \Sigma_k^{old})}{\sum_{\ell=1}^{K} p_\ell^{old}\,\mathcal{N}(x_j \mid \mu_\ell^{old}, \Sigma_\ell^{old})}$, $k = 1, -, K$
- $n_k^{new} = n_k^{old} + \hat{p}_{k,j}^{new} - \hat{p}_{k,j}^{old}$, $k = 1, -, K$
- $p_k^{new} = p_k^{old} + \dfrac{\hat{p}_{k,j}^{new}}{n} - \dfrac{\hat{p}_{k,j}^{old}}{n}$, $k = 1, -, K$
- $\mu_k^{new} = \mu_k^{old} + \left(\dfrac{\hat{p}_{k,j}^{new} - \hat{p}_{k,j}^{old}}{n_k^{new}}\right)(x_j - \mu_k^{old})$
- $\Sigma_k^{new} = \Sigma_k^{old} + \dfrac{\hat{p}_{k,j}^{new}}{n_k^{new}}\{\dots\} - \dfrac{\hat{p}_{k,j}^{old}}{n_k^{new}}\{\dots\}$.

(iii) Repeat step (ii) until convergence.

The parameters are revised after each data point, instead of waiting until the full dataset is processed ⇒ this generally speeds up the algorithm.

Remark on the initialization step. Instead of setting $\hat{p}_{k,i}^{old} = \frac{1}{K}$, we may run one iteration of the batch EM algorithm, providing initial estimates for $\hat{p}_{k,i}$. Then switch to the incremental version.