Chapter 9
Clustering

(Again : unsupervised learning)

I) K-means clustering

We are given N points : $x_1, x_2, \dots, x_N$ in $\mathbb{R}^D$. We want to group them in clusters according to their distances within one another. We usually use the euclidean distance but one could do otherwise.

Suppose we have K clusters. For each point x_n, $n \in \{1, 2, \dots, N\}$ we have labels $r_{n1}, \dots, r_{nK}$ such that $r_{ni} \begin{cases} = 1 \text{ if } x_n \text{ in cluster } i \\ = 0 \text{ otherwise} \end{cases}$

It is worth noticing that $(\forall n) \sum_{k=1}^{K} r_{nk} = 1$.

We define the mean of the k-th cluster by :

$$\mu_k = \frac{\sum_{n=1}^{N} r_{nk} x_n}{\sum_{n=1}^{N} r_{nk}} \quad \text{(III)}$$

(i.e. the mean of the points in cluster k)

We suppose each point x_n is assigned to the cluster k such that μ_k is the closest to x_n.

i.e : $\forall n$, $r_{nk} = 1 \iff (x_n - \mu_k)^T (x_n - \mu_k) \leq (x_n - \mu_i)^T (x_n - \mu_i)$ $(\forall i)$

We want to end up in such a configuration.

To do this, we use an iterative algorithm

Fix K.
Start with initial random values $\mu_1, \dots, \mu_K$.

1. For each x_n, find k minimizing $(x_n - \mu_k)^T (x_n - \mu_k)$ and set $r_{nk} = 1$ ($r_{nj} = 0$ for $j \neq k$) → assignment step / expectation step
2. If all the assignments (r_{nk}) are unchanged from the previous iteration → stop.
3. Update μ_k with Equation (M1). → update step / maximization step
4. Return to 1

Why does it work?

We are interested in a cost function: $= \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|x_n - \mu_k\|^2$ where $\mu_k = \dots$

$$F\left(\mu_1, \dots, \mu_K, (r_{nk})_{\substack{1 \le k \le K \\ 1 \le n \le N}}\right)$$ (which we want to minimize)

Step 1 assign a cluster number to each x_n. Let us call r_{nk} the labels before the re-assignment and r'_{nk} the labels after the re-assignment. We have:

$$F\left((\mu_k), (r_{nk})\right) \geq \min_{(\bar{r}_{nk})} F\left((\mu_k), (\bar{r}_{nk})\right)$$
$$= F\left((\mu_k), (r'_{nk})\right)$$

Step 3 changes the (μ_k). Let us write (μ_k) for the old μ and (μ'_k) for the new (μ_k). We have

$$F\left((\mu_k), (r'_{nk})\right) \geq \min_{(\bar{\mu}_k)} F\left((\bar{\mu}_k), (r'_{nk})\right)$$
$$= F\left((\mu'_k), (r'_{nk})\right)$$

↙ explanation

the (r'_{nk}) are fixed and we look at the following function of $(\mu_1, \dots, \mu_K)^T$:

$$\Phi(\mu_1, \dots, \mu_k) = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} (x_n - \mu_k)^T (x_n - \mu_k)$$

We have: $\nabla \bar{\Phi}(\mu_1, \dots, \mu_K) = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk} (x_n - \mu_k)$ → WRONG

exercise: prove it

So we have a critical point at (k): $\mu_k = \frac{\sum_{n=1}^{N} \gamma_{nk} x_n}{\sum_{n=1}^{N} \gamma_{nk}}$.

$\bar{\Phi}$ is convex in μ_k (ex: prove it) so the critical point is an absolute min.

exercise: find by yourself the min of $\bar{\Phi}$, write all the details

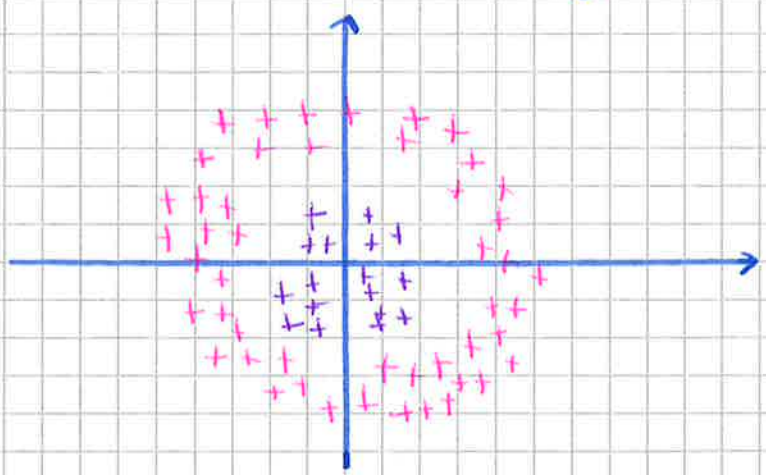
So the algorithm can only decrease F. This is why we end up in a local minimum (which might not be the absolute minimum).

Choosing the number of clusters (K).

The cost F decreases when K increases. So there is no point in choosing K in order to minimize F.
The best is to look beyond the clustering to the overall aim of the analysis.

Where K-means fails: these are essentially cases where the distance used is not a good criterion for clustering

ex:



II) Kernelized k-means

We observe that we only need to compute distances of the kind: $d_{nk} = (x_n - \mu_k)^T (x_n - \mu_k)$

$$= \left(x_n - \frac{1}{N_k}\sum_{m=1}^{N} r_{mk} x_m\right)^T \left(x_n - \frac{1}{N_k}\sum_{m=1}^{N} r_{mk} x_m\right)$$

where $N_k = \sum_{n=1}^{N} r_{nk}$

$$d_{nk} = x_n^T x_n - \frac{2}{N_k}\sum_{m=1}^{N} r_{mk} x_n^T x_m + \frac{1}{N_k^2}\sum_{m=1}^{N}\sum_{n=1}^{N} r_{mk} r_{nk} x_m^T x_n$$

So we replace the inner products $x_m^T x_n$ with a kernel function to get a "kernelised distance":

$$d_{nk} = K(x_n, x_n) - \frac{2}{N_k}\sum_{m=1}^{N} r_{mk} K(x_n, x_m) + \frac{1}{N_k^2}\sum_{m=1}^{N}\sum_{n=1}^{N} r_{mk} r_{nk} K(x_m, x_n) \quad (D1)$$

the idea is that $K(x_n, x_m) = \phi(x_n)^T \phi(x_m)$ where ϕ is a transformation which makes the cluster easy to separate. the mean of the cluster k becomes:

$$\mu_k = \frac{\sum_{n=1}^{N} r_{nk}\,\phi(x_n)}{\sum_{n=1}^{N} r_{nk}}$$

But now, ϕ is a mysterious (hidden) application. So we modify the algorithm into:

1. Randomly initialize r_{nk} for each n
2. Compute $d_{n1}, \ldots, d_{nK}$ for each x_n (using Equation (D1))
3. Assign each object to the cluster with the lowest d_{nk} (i.e. $r_{nk} = 1$ if $d_{nk} = \min_i d_{ni}$; $= 0$ otherwise)

4. If assignments have changed, return to step 2, otherwise stop.

Caveat: K-means is sensitive to initialization so we better choose the μ_k non-random (look at the data / perform a standard k-means / ...)

Remarks: some kernels are designed to work well on texts, graphs, networks...

III) Hierarchical clustering

1) Agglomerative clustering

The algorithm begins with every point (→ N points at the beginning) representing a singleton cluster. At each of the N-1 steps, the closest two (least dissimilar) clusters are merged into a single cluster.

Let us define a measure of dissimilarity.

We fix a distance $d(.,.)$ in our space of data. For the points $x_i, x_{i'}$, we set $d_{ii'} = d(x_i, x_{i'})$ and we call this the pairwise observation dissimilarity.

When we have two clusters G and H, we can compute the dissimilarity $d(G,H)$ between G and H in two ways:

Single linkage (SL): $d_{SL}(G,H) = \min_{\substack{i \in G \\ i' \in H}} d_{ii'}$

(also called nearest neighbor technique)

Group average (GA): $d_{GA}(G,H) = \frac{1}{N_G N_H} \sum_{i \in G} \sum_{i' \in H} d_{ii'}$

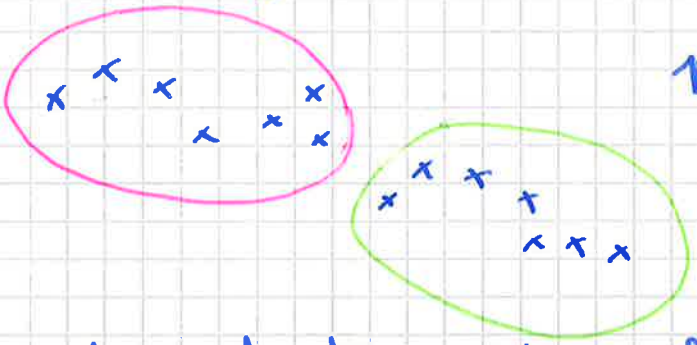
(N_G, N_H: number of points in G, H)

Complete linkage: $d_{CL}(G,H) = \max_{i \in G,\, i' \in H} d_{ii'}$

If clusters are compact, the three methods will produce similar results.

* SL can produce clusters where observations are linked by a series of close intermediate observations (this is referred to as chaining)

ex:

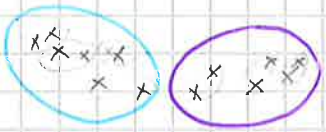


two clusters produced by SL

SL can produce clusters with very large diameters

* CL produces compact clusters with small diameters. It can produce clusters such that observations in a cluster can be much closer to members of other clusters than they are to some members of their own cluster.

ex:



* GA represents a compromise between the two extremes. Applying a monotone strictly increasing transformation $h(.)$ to the $d_{ii'}$ ($h_{ii'} = h(d_{ii'})$) can change the result (this is not the case for CL and SL)

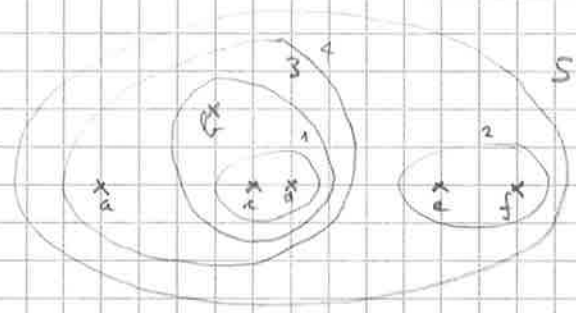
Statistical consistency property of GA: if population in G is sampled $\sim p_G(.)$, population in H is sampled $\sim p_H(.)$, then $d_{GA}(G,H)$ is an estimate of

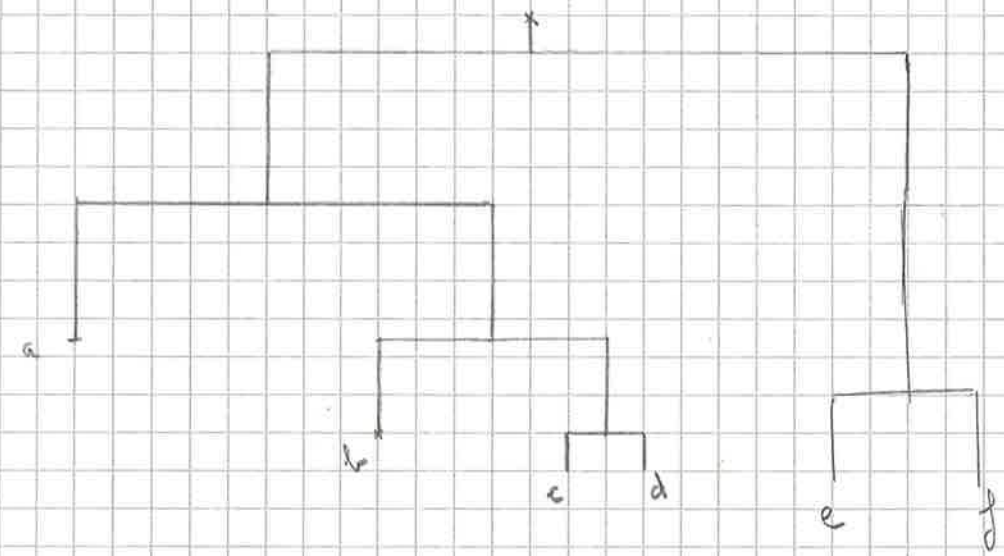
$$\iint d(x,x')\, p_G(x)\, p_H(x')\, dx\, dx' \qquad (D2)$$

When N_G and $N_H \to \infty$, $d_{GA}(G,H)$ approaches (D1) whereas $\begin{cases} d_{SL}(G,H) \to 0, \\ d_{CL}(G,H) \to +\infty. \end{cases}$

Binary trees: In all cases, the dissimilarity between merged clusters is monotone increasing with the level of the merges. Thus the binary tree representing the clustering process can be plotted so that the height of each node is proportional to the value of the intergroup dissimilarity between its two daughters. This type of graphical display is called a dendogram.

Ex: SL





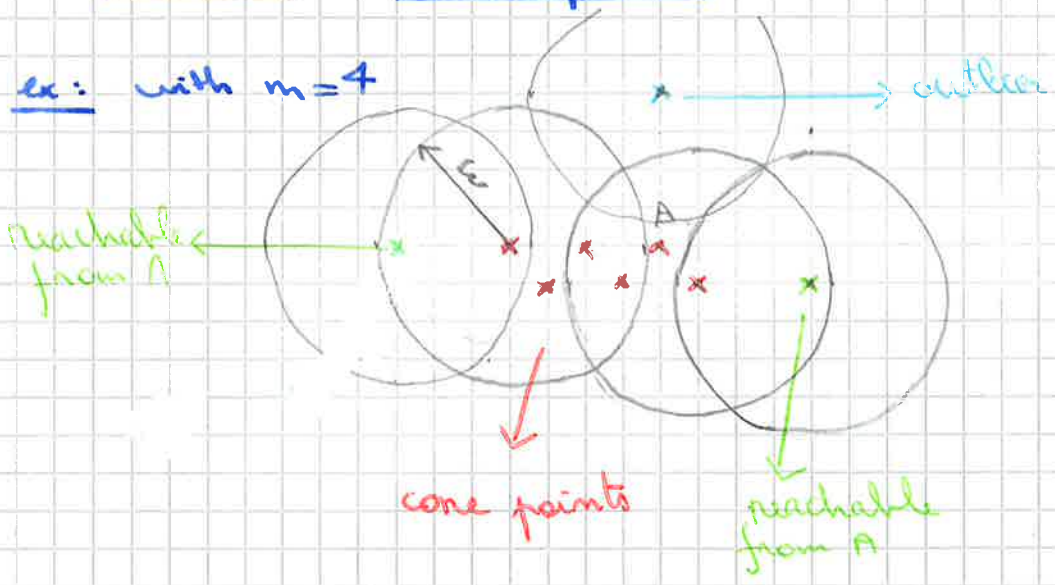
2) Density-based spatial clustering of applications with noise (DBSCAN)

a) Definitions

We have a set of points we want to cluster. We have to specify two parameters: $\varepsilon > 0$, $m \in \mathbb{N}^*$ (a minimum number of points).

* a point p is a <u>core point</u> if at least m points are within distance ε of it (including p)
* a point q is <u>directly reachable</u> from p if q is within distance ε from core point p
* a point q is <u>reachable</u> from p if there is a path $p_1, \dots, p_n$ with $p_1 = p$ and $p_n = q$ and each p_{i+1} is directly reachable from p_i (implicitely, all points in the path must be core points, with the possible exception of q)
* all points not reachable from any other points are <u>outliers</u> or <u>noise points</u>

ex: with $m = 4$



If p is a core point, it forms a cluster with all points that are reachable from it.
In a cluster, points which are non-core points form the edge of the cluster.

Two points p and q are density-connected if there is a point o such that p and q are reachable from o.
A cluster then satisfies two properties:

- all points within the cluster are mutually density-connected,
- if a point is reachable from the cluster, it is part of the cluster as well.

2) Algorithm

i) Find the points in the ε-neighborhood of every point, and identify the core points (those with at least m points in their ε-neighborhood).
ii) Find the connected components of core points in the neighbor graph (ignoring all non-core points)
iii) Assign each non-core point to a nearby cluster or assign it to noise.

3) Discussion

Pro
- DBSCAN does not require to specify the number of clusters (≠ k-means)
- can find arbitrarily shaped clusters
- has a notion of noise and so is robust to outliers
- DBSCAN requires only two parameters and is

mostly insensitive the ordering of the points in the database (except for points on the edge)
- parameters ε and m can be set by an expert if the data is well understood

Cons
- algorithm is not entirely deterministic (for edge points)
- DBSCAN depends on the distance chosen, euclidean distance will perform poorly in high-dimension
- parameters ε and m can be difficult to choose

Parameters: rule of thumb $m = 2 \times \dim + 3$

Exercises: p. 168-190 of the python book