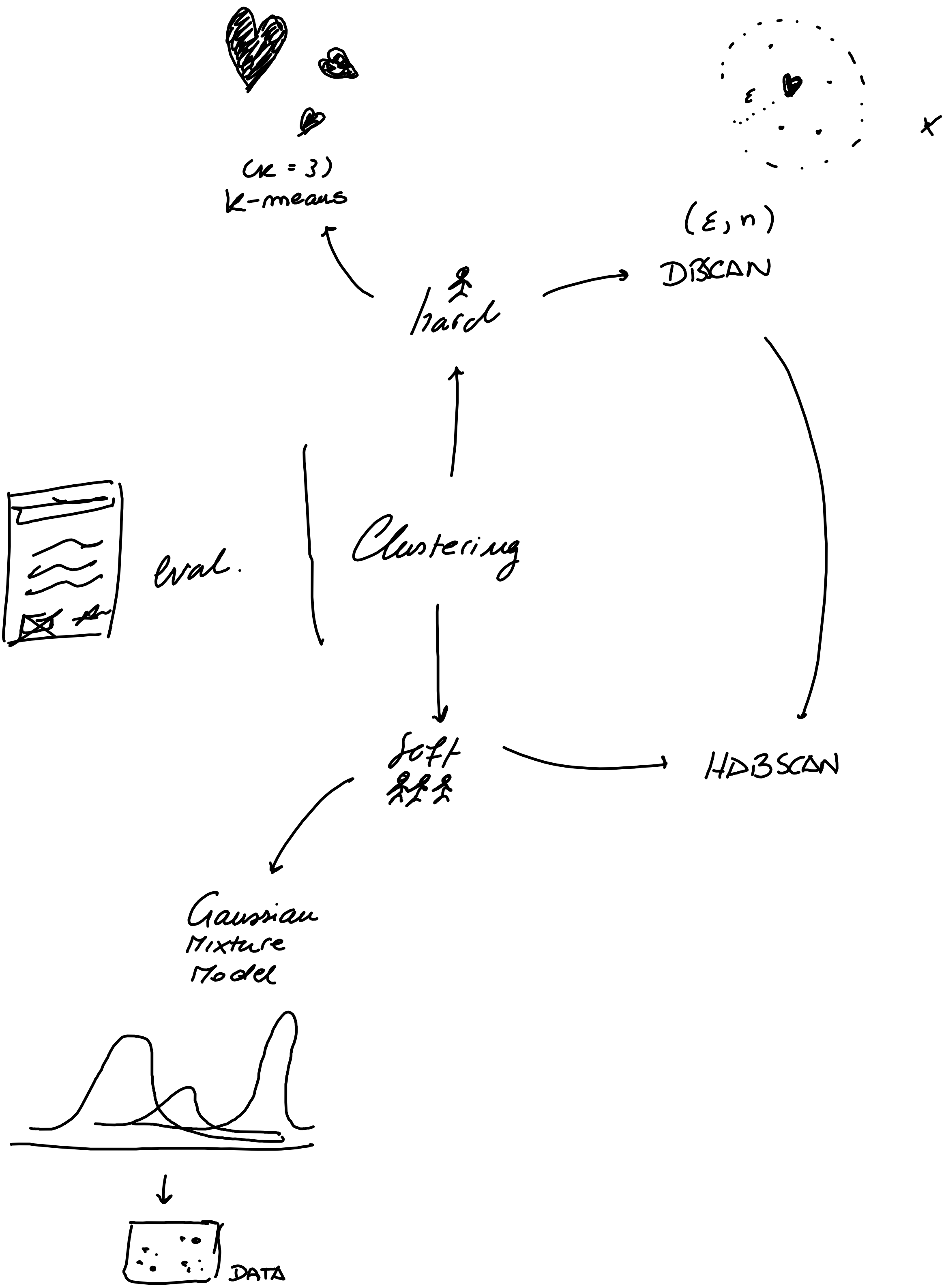


(k = 3)
K-means
ε
(ε, n)
DBSCAN
hard
eval.
Clustering
soft
HDBSCAN
Gaussian
Mixture
Model
DATA

Clustering algs:

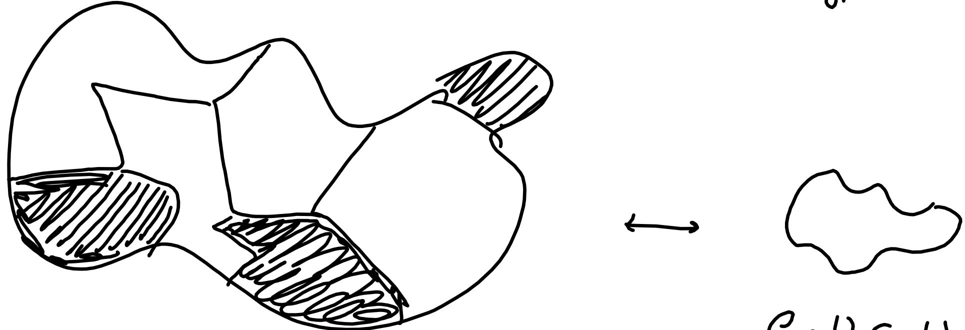
	SOFT	HARD
CENTER BASED	GMM	k-means
DENSITY BASED	HDBSCAN	DBSCAN

K-Means:

find K clusters within the data. K must be set beforehand.

IDea:

hyperparameter: K



1 . . . . n

$C_1 \cup C_2 \cup \dots \cup C_K$

and $C_i \cap C_j = \emptyset$ $(i \neq j)$

Goal function: with cluster variation

$$\min_{C_1,\dots,C_K}\left\{\sum_{k=1}^{K} W(C_k)\right\}$$

The definition of $W(C_k)$ depends on the distance definition we use:

ex Euclidean distance

n° features

$$W(C_k) := \frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij} - x\, x_{i'j})^2$$

K-means: algorithm

1) Randomly choose K clusters among the data

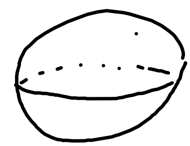
2) Repeat

   2.a) For each cluster, compute the cluster centroid.

$$x_k = \bar{x} \mid x \in C_k = (\bar{x}_1, \dots, \bar{x}_p)$$

   2.b) Reassign points to the cluster whose centroid is closer

Good for: spheric data clusters exampl



Idea:

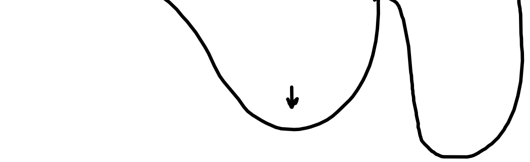
mean of feature j in cluster C_k

$$W(C_k) = 2 \sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj})^2$$

2.a) minimizes the sum of square deviations

2.b) minimize at each iteration $W(C_k)$

Danger:



Find <u>local</u> minima.

Multiple runs are recommended.

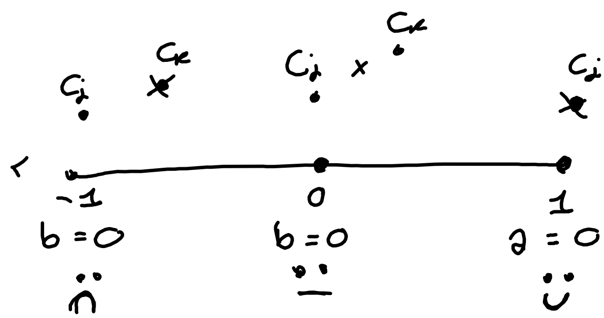
KMEANS:
performance measures

1. Overall d^2 from centroids:

$$\sum_{i=1}^{N} \sum_{\substack{x_i \in C_j \\ j=1}}^{K} d^2(x_i, C_k)$$

2. Silhoutte coefficient: $S = \frac{1}{N} \sum_{i=1}^{N} S(x_i)$

$$S(x_t) = \frac{(b - a)}{\max(b; a)}$$



where:

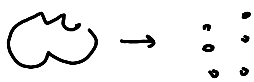
$$a := \frac{1}{|C_j|} \sum_{x \in C_j} d^2(x_t, x)$$ mean distance point-point in the same cluster

$$b := \frac{1}{K} \sum_{j=1}^{K} d^2(x_t; c_j)$$ mean distant point-other centroids

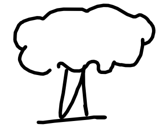
Hierarchical Clustering.

Δ class of Clust. algs. that do not need to choose k a priori.

BOTTOM-UP  (AGGLOMERATIVE)

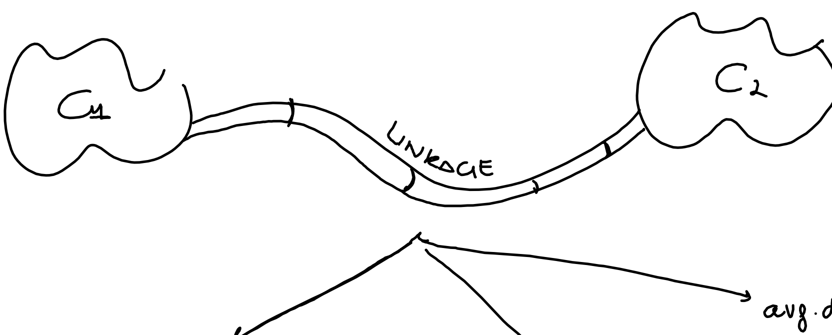
TOP-DOWN:  (DIVISIVE)

A dendrogram can be associated

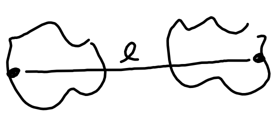


you can choose the degree of clust. precision arbitrarily.

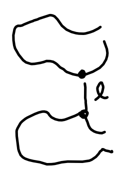
In agglomerative clustering, in the beginning every point is treated like a cluster. Then most similar points get joined in the same cluster. Clusters are joined following a chosen LINKAGE (infra-cluster similarity).



MAXIMUM distance between pair of points in the clusters



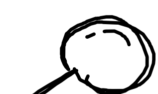
minimum "



avg. distance "

hyperparams) → similarity meas. / distance (ex: euclidean dist d^2)
→ linkage (min, max, avg...)
→ feature scaling (YES; NO)

} GRIDSEARCH

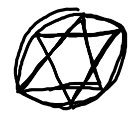


DBSCAN algorithm

1) Define a radius ε and a threshold m:

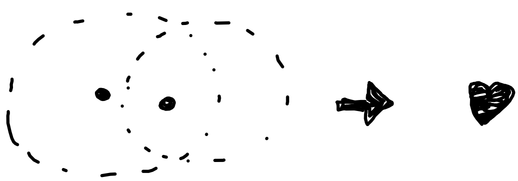
NOTE:
One point is a core if it has at least m point within its radius

2) Label points as core, border, noise



3) Remove noise points

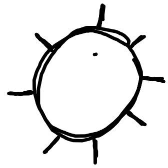
4) Make clusters with nearest core points:



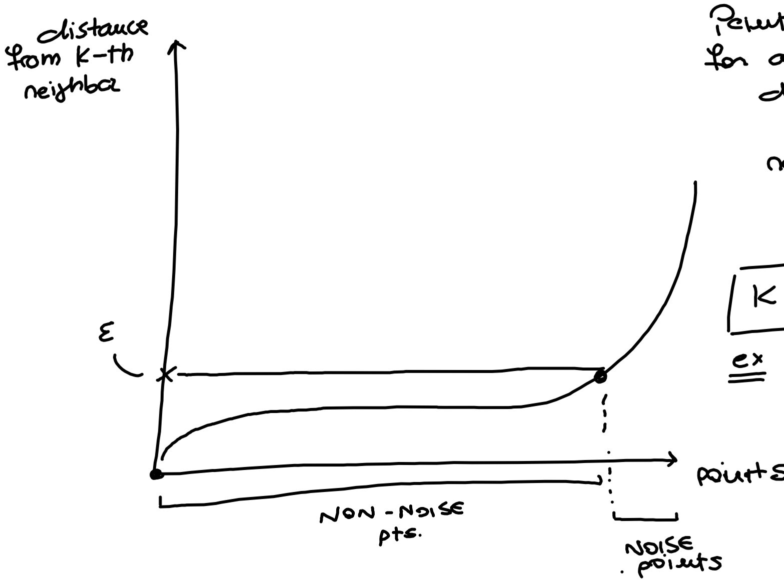
considering also core clusters

5) Assign border points to the nearest core point:

DBSCAN: Setting



NOTE How do we set the hyperparams? (ε and m)



Points are ordered for ascending distance from the k-th nearest point.

$\boxed{K = m}$

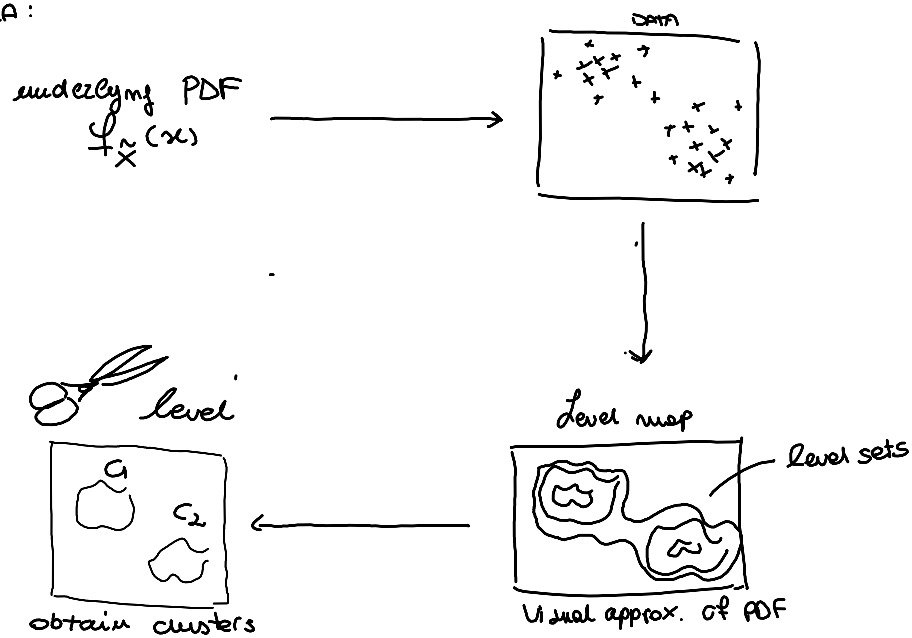
ex $K = 4$ frequently

HDBSCAN: fast density based clustering.

Hierarchical density based clusting. (ε is not needed)

hyperparameter: m (minimum n° of points needed to have a cluster).

IDEA:



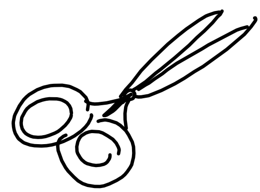
PROBLEM:

- do not know PDF ?

complexity: $O(n^2)$

- not comput. efficient

- hard to know cutting level a priori. The cutting level l corresponds to the threshold density to form a cluster

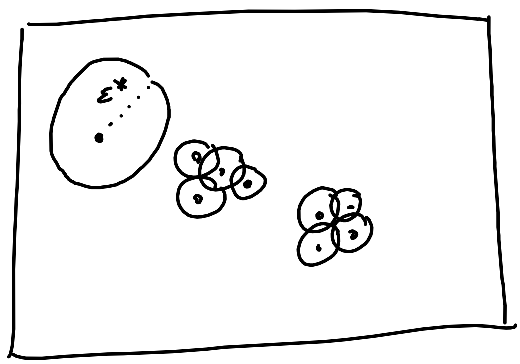


HDBSCAN: how it works

How do we compute $f_{\tilde{X}}(x)$?

1. locally approximate the density with ε^* :

$$\varepsilon^* = \min \varepsilon \;\Big|\; \text{Point is a core point}$$

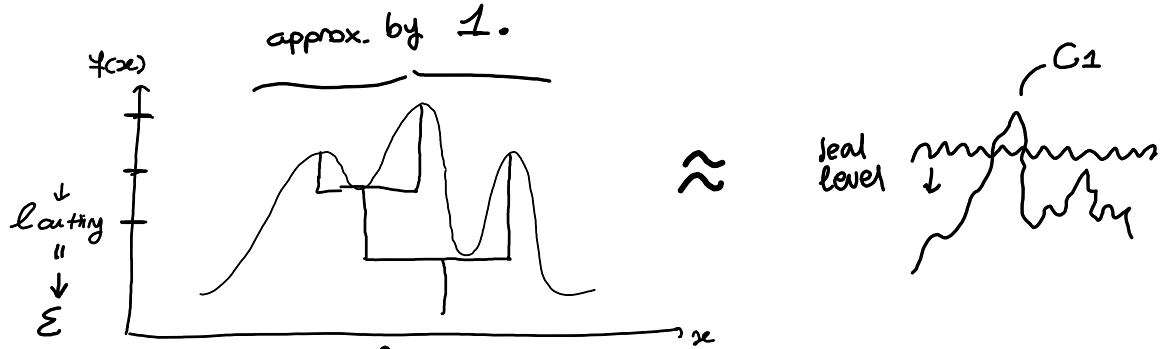


Denser regions automatically detected.

This draws a sketch (approx.) of $f_{\tilde{X}}(x)$.

How do choose the $\ell_{cutting}$?

2. decreasing $\ell_{cutting}$ form a tree (smart. dendrogram)



Dendrogram forms

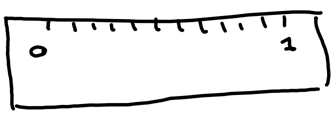
NOTE: in choosing ε in DBSAN we implicitly choose the cutting level ℓ.

HDBSCAN: how it works

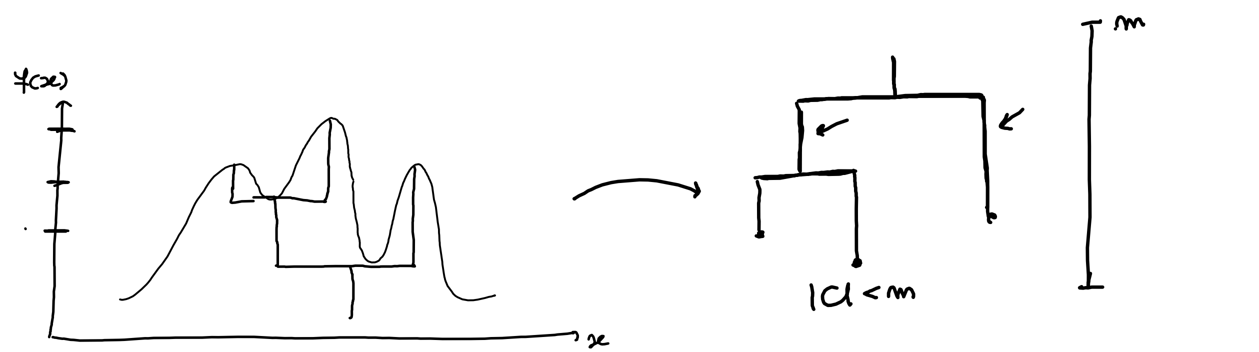
3. Cutting the tree according to mutual Reachibility distance:

$$d_{mreac}(x_i, x_j) = \begin{cases} \max\{K(x_i); K(x_j); d(x_i; x_j)\} & \\ 0 & x_i = x_j \end{cases}$$

where $K(x)$ = distance (x, k-th near. neighbour x)



4. Actual cut:



- vetical cut with the constraint that if you choose one vertical cluster you cannot choose its descendants.

If you choose vertical clusters that exist for long enough maybe you are choosing the NATURAL clusters you data shows

GMM : Gaussian Mixture Models

Assumption: data were drawn from a population having:

$$f_{\tilde{x}}(x) = \sum_{j=1}^{m} \phi_j \, f^{(x)}_{\mu_j, \Sigma_j}$$

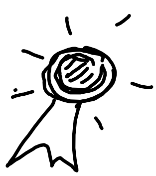
weight

$P(x_i \in C_j)$

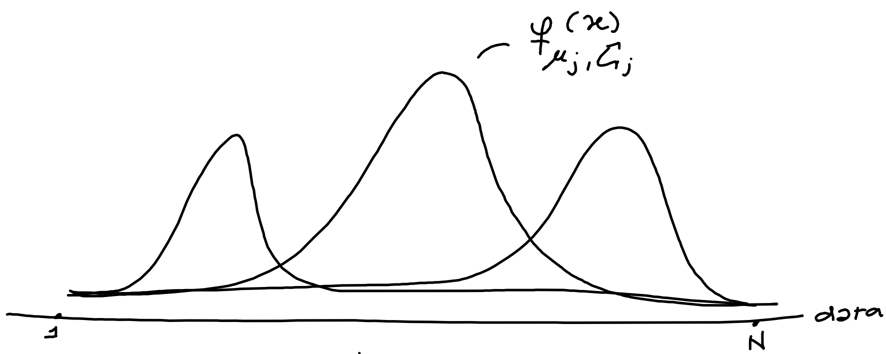
≈ importance of the ith (more data were likely drawn from it.)

multivariate normal distribution.

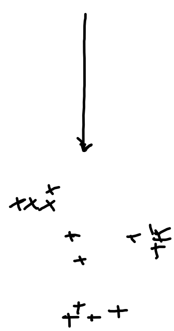
$P(\tilde{X} \in C_j)$



$\varphi_j, \mu_j, \Sigma_j$ are obtained through expectance maximization



data :



output: estimates for $f_{\mu_j,\Sigma_j}(x)$ withdrawn from data ($j = 1, \dots, m$).

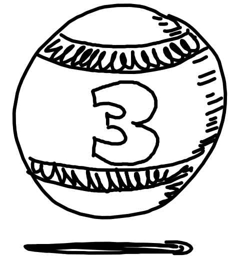
GMM: expectation maximization algorithm

FOR $j = 1, \dots, m$

0) Guess initial values for all $\varphi_j, \mu_j, \Sigma_j$ $\quad (\varphi_j = \frac{1}{m})$

1) FOR all $i = 1, \dots, N$ compute:

$$f_{\mu_j, \Sigma_j}(x_i)$$



2) Use Bayes Rule

$$P(x_i \in C_j \mid x_i) = \frac{P(x_i \mid x_i \in C_j)\, P(x_i \in C_j)}{P(x_i)}$$

$$= \frac{f_{\mu_j, \Sigma_j}(x_i)\, \varphi_j}{\sum_{k=1}^{m} f_{\mu_k, \Sigma_k}(x_i)\, \varphi_k}$$

3) compute μ_j, σ_j^2

$$\left.\begin{aligned} \mu_j &= \frac{\sum_{i=1}^{N} P(x_i \in C_j \mid x_i)\, x_i}{\sum_{i=1}^{N} P(x_i \in C_j \mid x_i)} \\ \sigma_j^2 &= \frac{\sum_{i=1}^{N} P(x_i \in C_j \mid x_i)\, (x_i - \mu_j)^2}{\sum_{i=1}^{N} P(x_i \in C_j \mid x_i)} \end{aligned}\right\}$$

Since we do NOT know if an element belongs certainly to a cluster, the best thing we can do is updating the MVN params (μ_j; σ_j^2) as a prob. weigh. term over the data. (*)

4) compute φ_j:

$$\varphi_j \leftarrow \frac{1}{N} \sum_{i=1}^{N} P(x_i \in C_j \mid x_i) \Big\} \text{ Average: } \overline{P}(x_i \in C_j)$$

STOP: μ_j, σ_j^2 don't change much.

(*) As a result μ_j, σ_j^2 are modified heavily just by the instances that are very likely to $\in C_j$

GMM:
scores for hyperparameter tuning ($n°$ clusters)

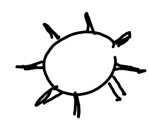
$$\left.\begin{aligned} AIC &= 2p - 2\log L \\ BIC &= p\log m - 2\log L \end{aligned}\right\}$$ penalized likelihood scores

$\max L \iff p(\vartheta = \vartheta_* \mid \underline{x})$ is high $\iff$ the estimations made by the GMM are good

$\Updownarrow$

the right number of cluster has been chosen

low AIC and BIC $\longleftrightarrow$ good $n°$ clust



Bayesian GMM: auto-evaluating $n°$ clusters

Another approach would be using Bayesian GMM with the advantages of:

→ $n°$ clusters > real $n°$ clust $\Rightarrow$ the alg. automatically detects the relevant $n°$ of clust. looking at the data

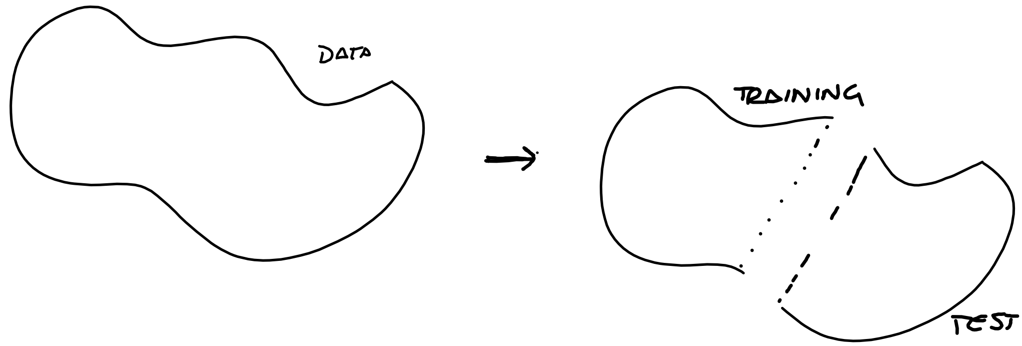
→ prior knowledge could be used about the weights (φ) of each RVN.

$\ll 1$ ———————— $\gg 1$ $\quad \alpha$

$(\varphi) \approx 1$ few clusters found $\qquad (\varphi \approx 0)$ more clusters found

NOTE
the more data we have the less the prior matters.

Evaluation: PREDICTION SCORE



1) Run a Clustering algorithm on both:

$$C(S_{tr}, k) = A \quad , \quad C(S_{te}, k)$$

considering k clusters

2) Build a co-membership matrix:

$$M \in N^{|S_{te}| \times |S_{te}|} \quad ; \quad m_{ij} := \begin{cases} 1 & x_i, x_j \in \text{same cluster according to } A \\ 0 & \text{otherwise} \end{cases}$$

3) Compute the prediction strength

$$PS(k) := \min_{j=1,\dots,k} \frac{1}{|A_j|(|A_j| - 1)} \sum_{i,j \in A_j} m_{ij}$$