
CLUSTERING

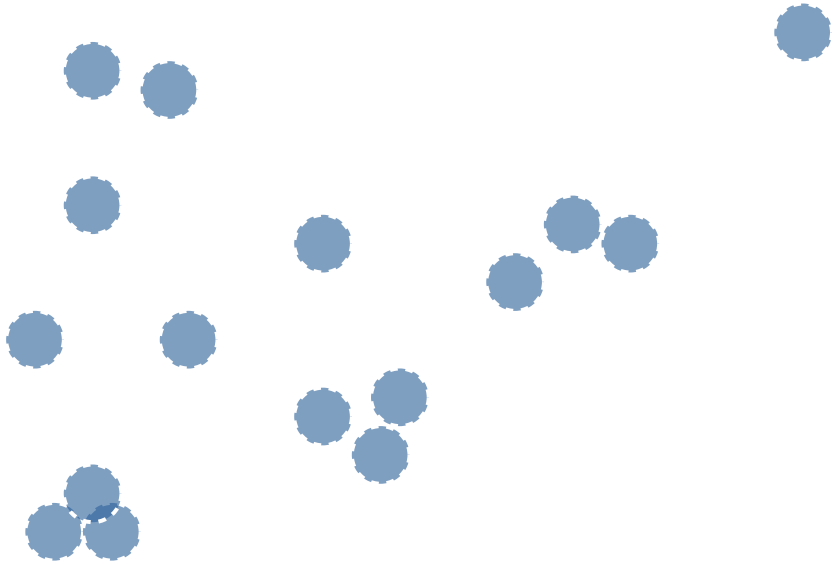
Unsupervised learning

- unlabeled data
- discovering structure
- more difficult and subtle than supervised learning

Arxiv by Mehta et al

Clustering

- simplest example of unsupervised learning
- group unlabeled data into clusters according to similarity or distance
- Applications: data mining, data compression, signal processing, …
- Physics: detecting celestial emission sources in astronomical surveys, inferring groups of genes and proteins with similar functions in biology, ...



Which are the clusters?


7
5
2
4
6
8
10
15
20
25
30
35
40
45
50
55
60


4
6
8
10
15
20
25
30
35
40
45
50
55
60
7
5
2
SOUTH AMERICA
SOUTH AFRICA
INDIA

K-means

- Minimize (locally)

$$J(\{x, \boldsymbol{\mu}\}) = \sum_{k=1}^{K} \sum_{n=1}^{N} r_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k)^2,$$

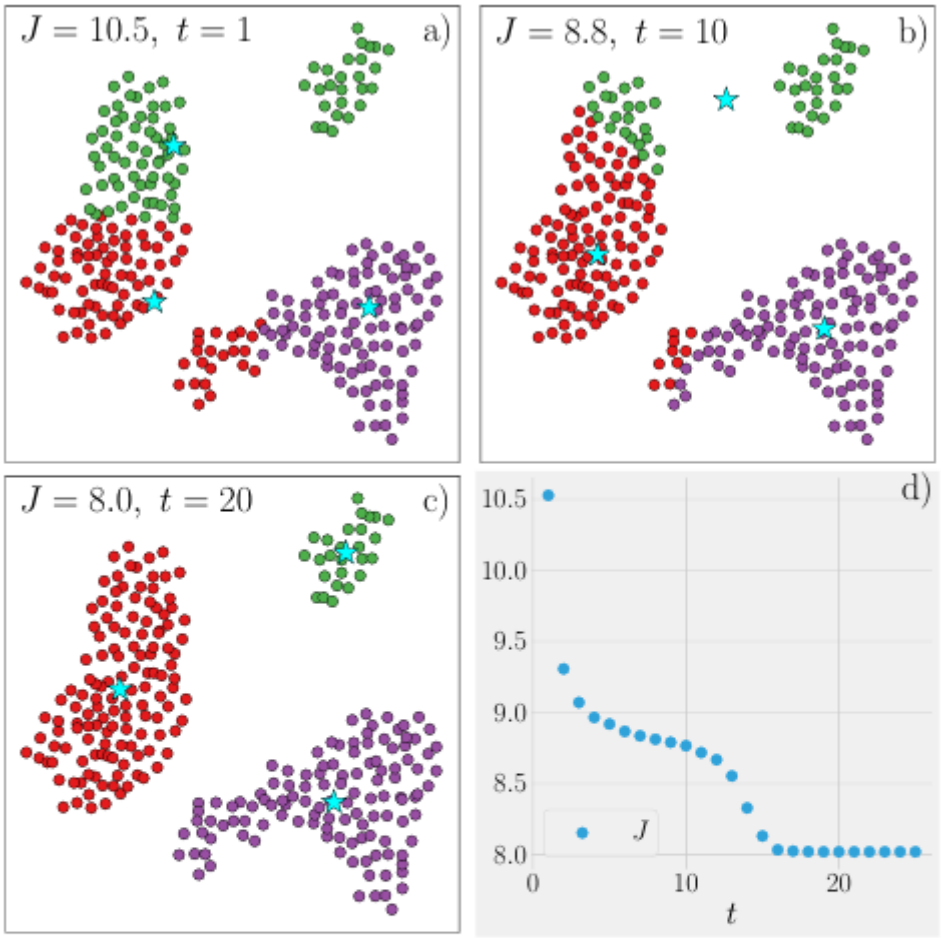
r_{nk} is a binary variable ($r_{nk} \in \{0, 1\}$)

1. *Expectation*: Given a set of assignments $\{r_{nk}\}$, minimize J with respect to $\boldsymbol{\mu}_k$. Taking a simple derivative and setting it to zero yields the following update rule:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_n r_{nk} \mathbf{x}_n. \qquad (136)$$

2. *Maximization*: Given a set of cluster means $\{\boldsymbol{\mu}_k\}$, find the assignments $\{r_{nk}\}$ which minimizes J. Clearly, this is achieved by assigning each data point to their nearest cluster-mean:

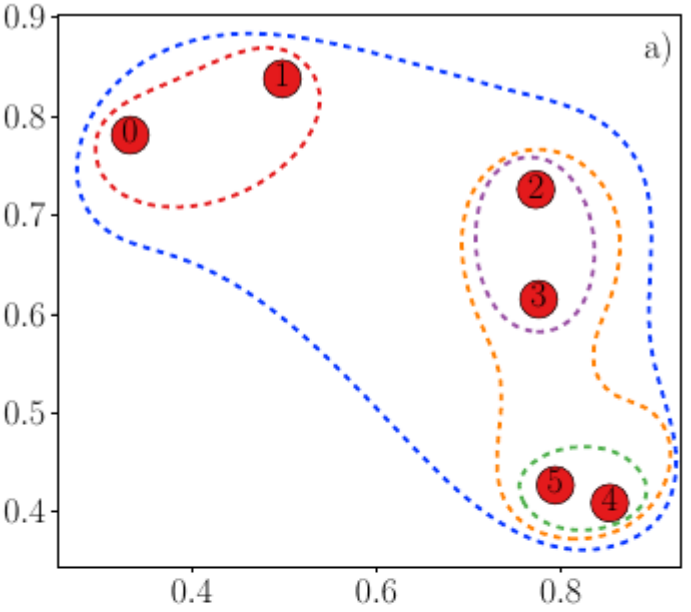
$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_{k'} (\mathbf{x}_n - \boldsymbol{\mu}_{k'})^2 \\ 0 & \text{otherwise} \end{cases} \qquad (137)$$



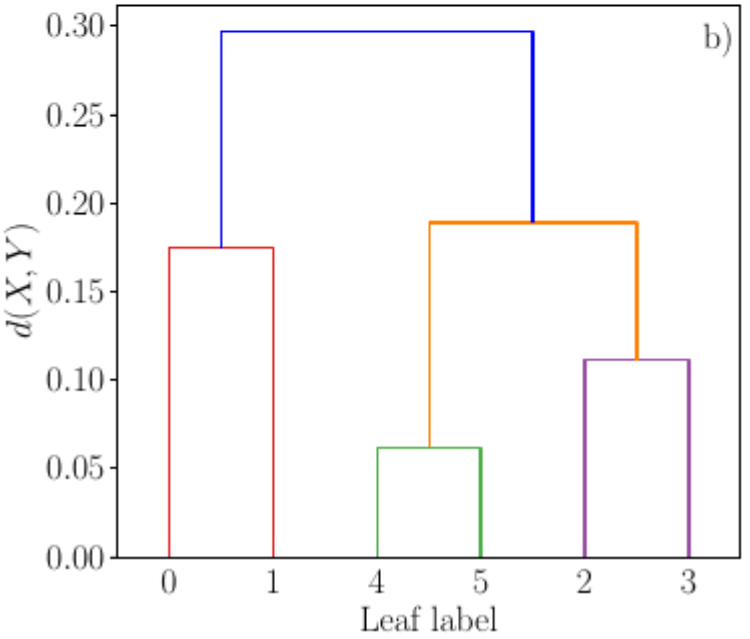
Needs the number (K) of clusters to be decided by the user

Hierarchical clustering

- Bottom-up approach, from small clusters
- Dendrogram



1. Initialize each point to its own cluster.
2. Given a set of K clusters $X_1, X_2, \cdots, X_K$, merge clusters until one cluster is left ($K = 1$):
   (a) Find the closest pair of clusters (X_i, X_j): $(i, j) = \arg\min_{(i',j')} d(X_{i'}, X_{j'})$
   (b) Merge the pair. Update: $K \leftarrow K - 1$



– Various choices for metric $d(X_i, X_j)$
– $O(N^2)$

Density-based (DB) clustering

- intuitive assumption:

  clusters ~ high density of data points
- Noise & outliers: expected low density
- Advantages:
  - handles clusters of multiple shapes and sizes while identifying outliers
  - suitable for large-scale applications

DBSCAN

"DB spatial clustering of applications with noise"
(the most prominent DB clustering algorithm)

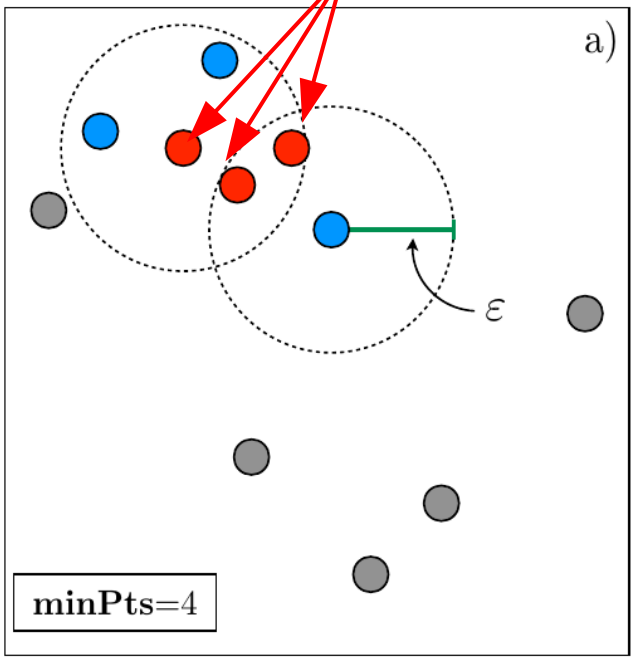
- Two parameters:
  - Radius **ε**
  - Min cluster size **minPts**

→ Until all points in X have been visited; **do**

- Pick a point $\mathbf{x}_i$ that has not been visited
- Mark $\mathbf{x}_i$ as a visited point
- If $\mathbf{x}_i$ is a core point; **then**
  - Find the set $\mathcal{C}$ of all points that are *density reachable* from $\mathbf{x}_i$.
  - $\mathcal{C}$ now forms a cluster. Mark all points within that cluster as being visited.

→ Return the cluster assignments $\mathcal{C}_1, \cdots, \mathcal{C}_k$, with k the number of clusters. Points that have not been assigned to a cluster are considered noise or outliers.

Core point: at least **minPts** neighbors



Efficient: O(N ln N)

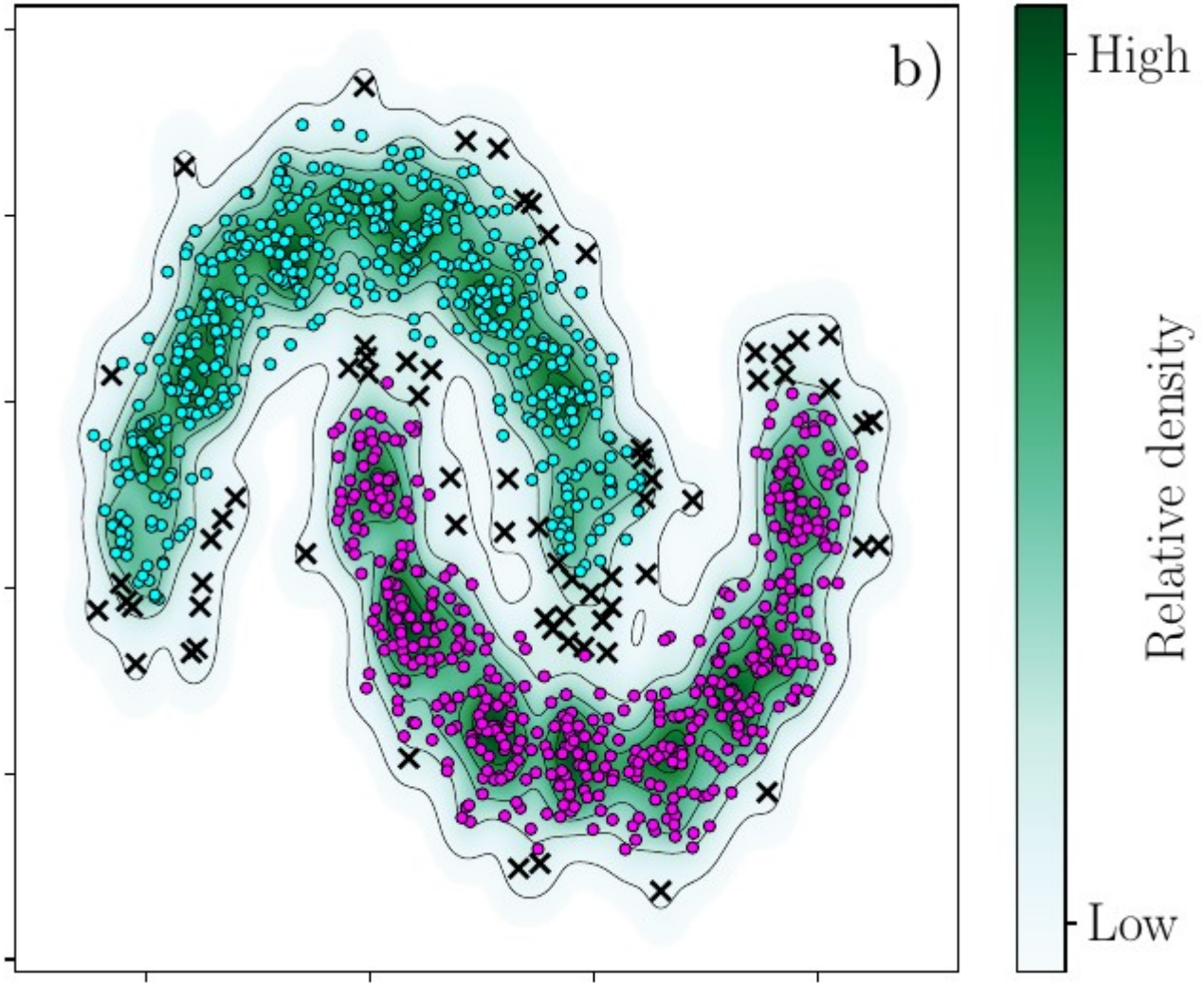
DBSCAN: wikipedia

```
DBSCAN(DB, distFunc, eps, minPts) {
   C = 0                                                  /* Cluster counter */
   for each point P in database DB {
      if label(P) ≠ undefined then continue               /* Previously processed in inner loop */
      Neighbors N = RangeQuery(DB, distFunc, P, eps)      /* Find neighbors */
      if |N| < minPts then {                              /* Density check */
         label(P) = Noise                                 /* Label as Noise */
         continue
      }
      C = C + 1                                           /* next cluster label */
      label(P) = C                                        /* Label initial point */
      Seed set S = N \ {P}                                /* Neighbors to expand */
      for each point Q in S {                             /* Process every seed point */
         if label(Q) = Noise then label(Q) = C            /* Change Noise to border point */
         if label(Q) ≠ undefined then continue            /* Previously processed */
         label(Q) = C                                     /* Label neighbor */
         Neighbors N = RangeQuery(DB, distFunc, Q, eps)   /* Find neighbors */
         if |N| ≥ minPts then {                           /* Density check */
            S = S ∪ N                                     /* Add new neighbors to seed set */
         }
      }
   }
}
```

where RangeQuery can be implemented using a database index for better performance, or using a slow linear scan:

```
RangeQuery(DB, distFunc, Q, eps) {
   Neighbors = empty list
   for each point P in database DB {                      /* Scan all points in the database */
      if distFunc(Q, P) ≤ eps then {                      /* Compute distance and check epsilon */
         Neighbors = Neighbors ∪ {P}                      /* Add to result */
      }
   }
   return Neighbors
}
```

Example of DBSCAN & underlying density



Clustering: vast field

- flurry of clustering methods suited for different purposes
- choosing method, consider:
  - distribution of the clusters (overlapping or noisy clusters / well-separated clusters)
  - the geometry of the data (flat / non-flat)
  - the cluster size distribution (multiple / uniform size)
  - the dimensionality of the data (low dim. / high dim.)
  - computational efficiency of the desired method (small vs. large dataset)

Clustering data in high-dimension can be very challenging. One major problem that is aggravated in high-dimensions is the accumulation of noise coming from spurious features that tends to "blur" distances (Domingos, 2012; Kriegel *et al.*, 2009; Zimek *et al.*, 2012). Many clustering algorithms rely on the explicit use of a similarity measure or distance metrics that weigh all features equally. For this reason, one must be careful when using an off-the-shelf method in high dimensions. In order to perform clustering on high-dimensional data, it is often useful to denoise the data before proceeding with using a standard clustering method such as K-means

PCA

Often clustering validation, i.e. verifying whether the obtained labels seem "valid" is done by direct visual inspection

To read

B. Clustering and Latent Variables via the Gaussian Mixture Models

In the previous section, we introduced several practical methods for clustering. In this section, we will approach clustering from a more abstract vantage point, and in the process, introduce many of the core ideas underlying unsupervised learning. A central concept in many unsupervised learning techniques is the idea of a latent or hidden variable. Even though latent variables are not directly observable, they still influence the visible structure of the data. For example, in the context of clustering we can think of the cluster identity of each datapoint (i.e. which cluster does a datapoint belongs to) as a latent variable. And even though we cannot see the cluster label explicitly, we know that points in the same cluster tend to be closer together. The latent variables in our data (cluster identity) are a way of representing and abstracting the correlations between datapoint.

In this language, we can think of clustering as an algorithm to learn the most probable value of a latent variable

Material

- Notebook

  NB15-CXII-clustering.ipynb

Aim

- Modify it to read and cluster data
- Some data available in the drive
- Color files → clusters [0,1/3), [1/3,2/3), [2/3,1]