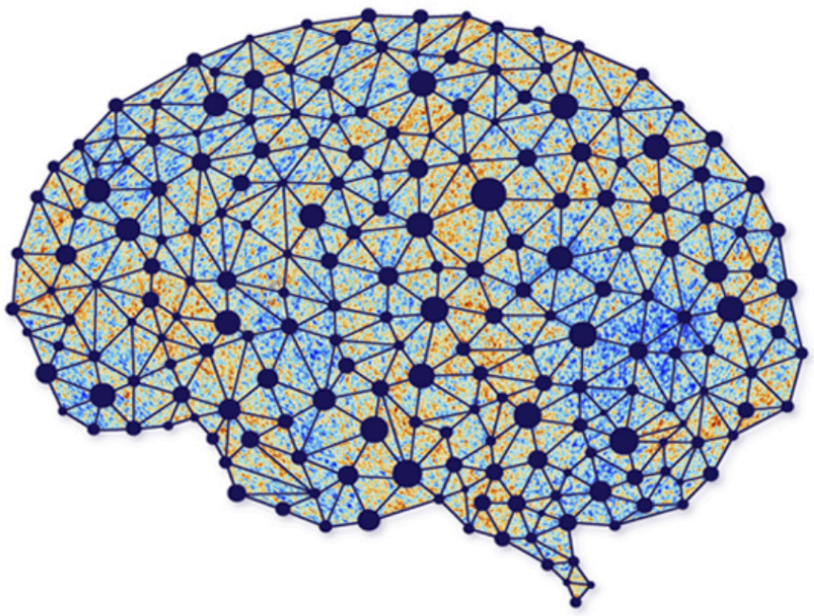
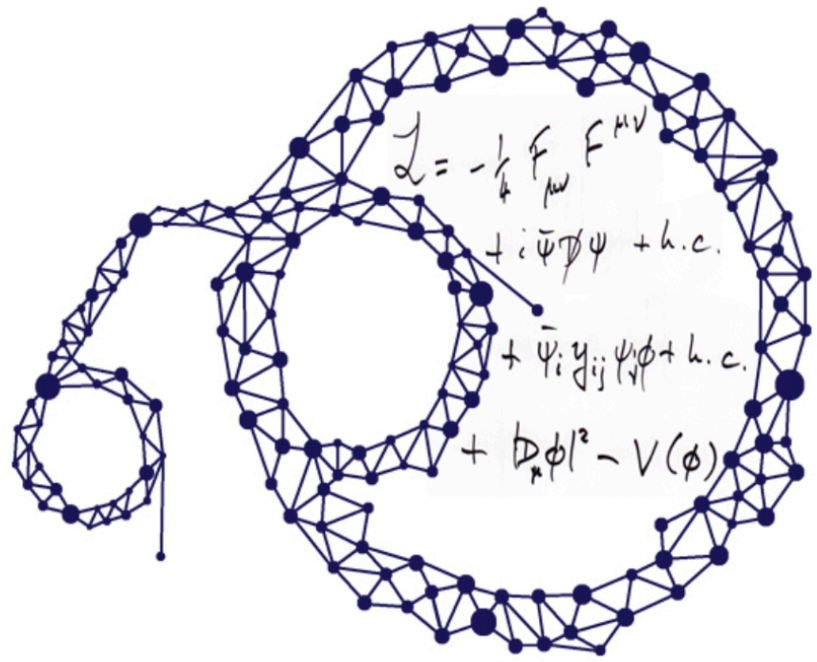
PHY 835: Machine Learning in Physics

Lecture 12: Unsupervised Learning (Clustering)

February 29, 2024



AI ∩ Universe



Gary Shiu

Outline for today

- t-stochastic neighbor embedding (t-SNE)
- K-means clustering
- Agglomerative clustering
- Density-based (DB) clustering
- Gaussian mixture models

References: 1803.08823, Deep Learning Book

https://physics.bu.edu/~pankajm/ML-Notebooks/HTML/NB15-CXII-clustering.html

t-SNE

- t-stochastic neighbor embedding: non-parametric method that constructs non-linear embeddings, optimized to preserve the local data structure.
- Has been used to reduce the dimensionality and classify spin systems such as Ising Model and Fermi-Hubbard models; glass-like problems in quantum control.
- **Idea:** associate a probability distribution to the neighborhood of each data:

$$p_{i|j} = \frac{\exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-||x_i - x_k||^2/2\sigma_i^2)}, \qquad p_{i|i} = 0$$

- σ_i are free bandwidth parameters determined by the local entropy:

$$H(p_i) \equiv -\sum_j p_{j|i}\log_2 p_{j|i}.$$

- Setting $H(p_i)$= constant, $\Sigma = 2^{H(p_i)}$ = perplexity determines σ_i. Points in regions of high density will have small σ_i.

t-SNE

- **Gaussian likelihoods: only nearby points contribute**
  - Similarity of nearby points well represented
  - Problem of outliers (exponentially vanishing contributions to the distribution): embedding coordinates are ambiguous.

- The outliner problem can be avoided by symmetrization:

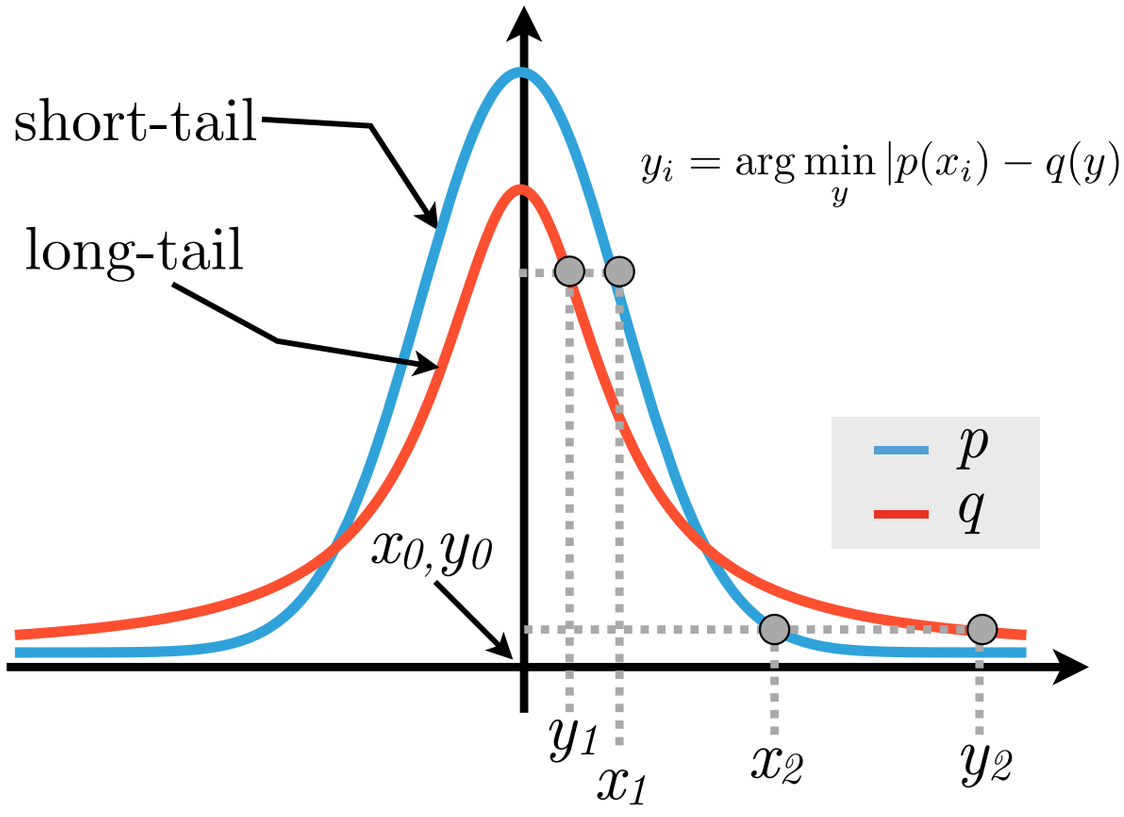
$$p_{ij} \equiv (p_{i|j} + p_{j|i})/(2N). \qquad \Rightarrow \qquad \sum_j p_{ij} > 1/(2N)$$

- t-SNE constructs a similar probability distribution in a lower dimensional latent space:

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq i}(1 + ||y_i - y_k||^2)^{-1}}.$$

t-SNE

- Long tail distribution: preserves short distance information while strongly repelling two points that are far apart in the original space.



- Latent space coordinates are found by minimizing the KL divergence:

$$C(Y) = D_{KL}(p \parallel q) \equiv \sum_{ij} p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right)$$

- Equivalent to **finding equilibrium** configuration of particles:

$$\partial_{y_i} C = \sum_{j \neq i} 4 p_{ij} q_{ij} Z_i (y_i - y_j) - \sum_{j \neq i} 4 q_{ij}^2 Z_i (y_i - y_j),$$
$$= F_{\text{attractive},i} - F_{\text{repulsive},i},$$

where $Z_i = 1/(\sum_{k \neq i}(1 + ||y_k - y_i||^2)^{-1})$.

attractive force comes only between nearby points in the original space

Properties of t-SNE

- Can rotate data: KL divergence is invariant under rotations in latent space.

- Results are stochastic: will depend on initial seed for gradient descent.

- Generally preserves short-distance information (preserves ordination but not actual distance between points).

- Deforms scales (not too much emphasis on size of clusters).

- Computationally expensive with a $\mathcal{O}(N^2)$ scaling (can be improved to $\mathcal{O}(N \log N)$ using the Barnes-Hut method.

Performance

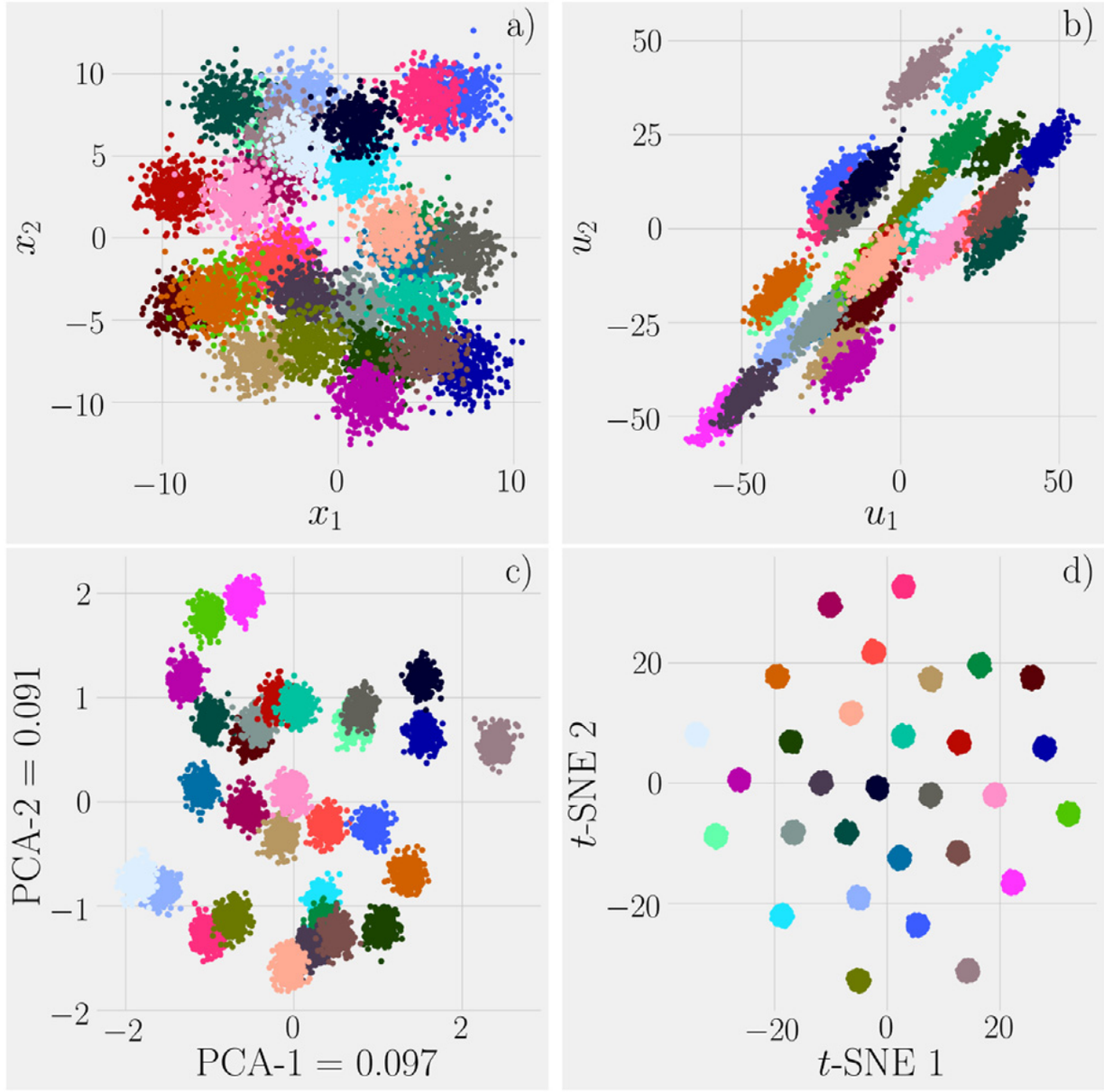


Fig. 53. Different visualizations of a Gaussian mixture formed of $K = 30$ mixtures in a $D = 40$ dimensional space. The Gaussians have the same covariance but have means drawn uniformly at random in the space $[-10, 10]^{40}$. (a) Plot of the first two coordinates. The labels of the different Gaussian are indicated by the different colors. Note that in a realistic setting, label information is of course not available, thus making it very hard to distinguish the different clusters. (b) Random projection of the data onto a 2 dimensional space. (c) Projection onto the first 2 principal components. Only a small fraction of the variance is explained by those components (the ratio is indicated along the axis). (d) t-SNE embedding (perplexity = 60, # iteration = 1000) in a 2 dimensional latent space. t-SNE captures correctly the local structure of the data.

Performance

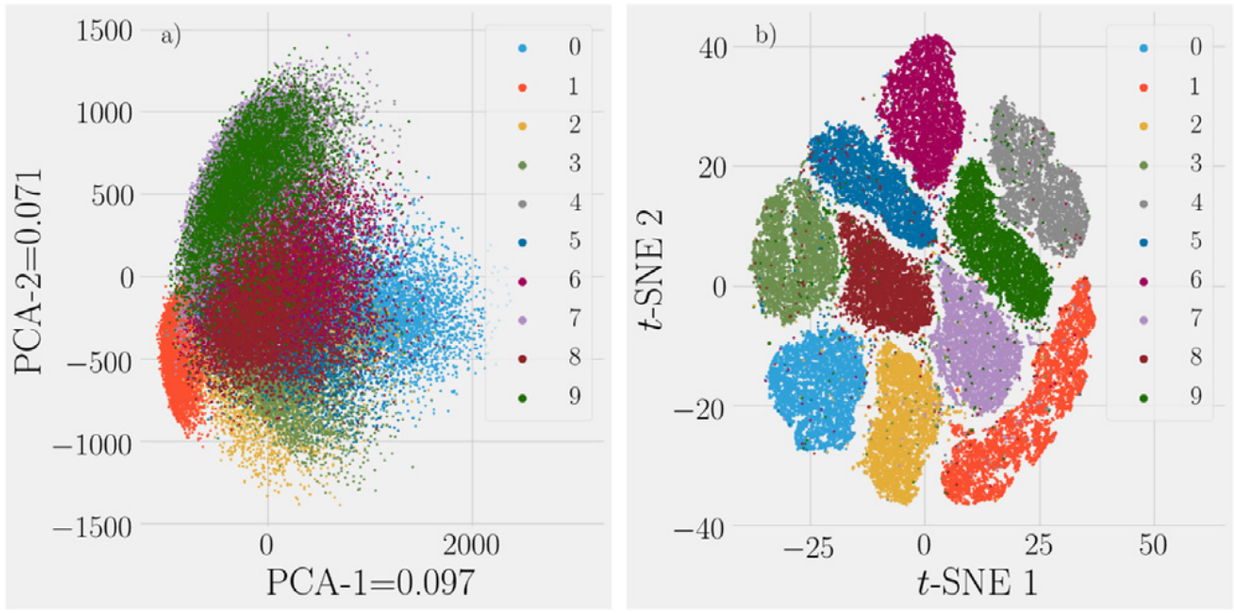


Fig. 54. Visualization of the MNIST handwritten digits training dataset (here $N = 60\ 000$). (a) First two principal components. (b) t-SNE applied with a perplexity of 30, a Barnes–Hut angle of 0.5 and 1000 gradient descent iterations. In order to reduce the noise and speed-up computation, PCA was first applied to the dataset to project it down to 40 dimensions. We used an open-source implementation to produce the results (Linderman et al., 2017), see https://github.com/KlugerLab/FIt-SNE.

t-SNE on GPU

- t-SNE is a great tool but quickly becomes slow to operate with the sklearn implementation.

- Making t-SNE fast by putting it on the GPU: https://medium.com/rapids-ai/tsne-with-gpus-hours-to-seconds-9d9c17c941db

Applications

- How much power is in your dimensions? MNIST: decay of power in components of PCA.

- Interpretability of first component(s): 2D Ising (magnetization)

- Visualize which variables your neural network is using: apply PCA (or other visualization methods) to different layers. Remember, deeper layers use more abstract variables.

- Disclaimer: this is a subset of visualizing techniques. If you face a visualization problem which cannot be dealt with with these methods, take a more detailed look on available algorithms.

Clustering

- Think of it as a simple way to look for hidden structure in high dimensions (coarse features or high-level structures in unlabelled data).

- Points to take into account when choosing a particular method:

    - Distribution of clusters (overlapping/noisy clusters vs. well-separated clusters)
    - Geometry of the data (flat vs. non-flat)
    - Cluster size distribution (multiple vs. uniform sizes)
    - Dimensionality of the data (low-dimensional vs. high-dimensional)
    - Computational efficiency of desired method

K-means Clustering

- Divide training set into K different clusters of data points which are *near* each-other.
- Consider a set of N unlabeled data points $\{\mathbf{x}_n\}_{n=1}^{N}$ where $\mathbf{x}_n \in \mathbb{R}^p$.
- K cluster centers called the cluster means: $\{\mu_k\}_{k=1}^{K}$ with $\mu_k \in \mathbb{R}^p$.
- Minimize the cost: $C(\{x, \boldsymbol{\mu}\}) = \sum_{k=1}^{K} \sum_{n=1}^{N} r_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k)^2,$
- One-hot encoding: $r_{nk} = 1$ if $\mathbf{x}_n \in$ cluster k and 0 otherwise;

$$\sum_k r_{nk} = 1 \ \forall \ n \text{ and } \sum_n r_{nk} \equiv N_k,$$

- Find the best cluster means (center of mass) such that variance (moment of inertia) is minimized.

K-means Algorithm

- **Expectation:** Given $\{r_{nk}\}$, minimize C with respect to μ_k:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_n r_{nk} \mathbf{x}_n$$

- **Maximization:** Given $\{\mu_k\}$, find $\{r_{nk}\}$ which minimizes C:

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

- Alternative between the above two steps until some convergence criterion is met (e.g., change in C is smaller than a threshold).

- Guaranteed to converge to local minimum (different initial random cluster center initializations and post-select). Complexity $\mathcal{O}(kN)$.

- Hard-assignment limit of the Gaussian mixture model (introduce later), where all cluster variances are assumed to be the same.

K-means Algorithm

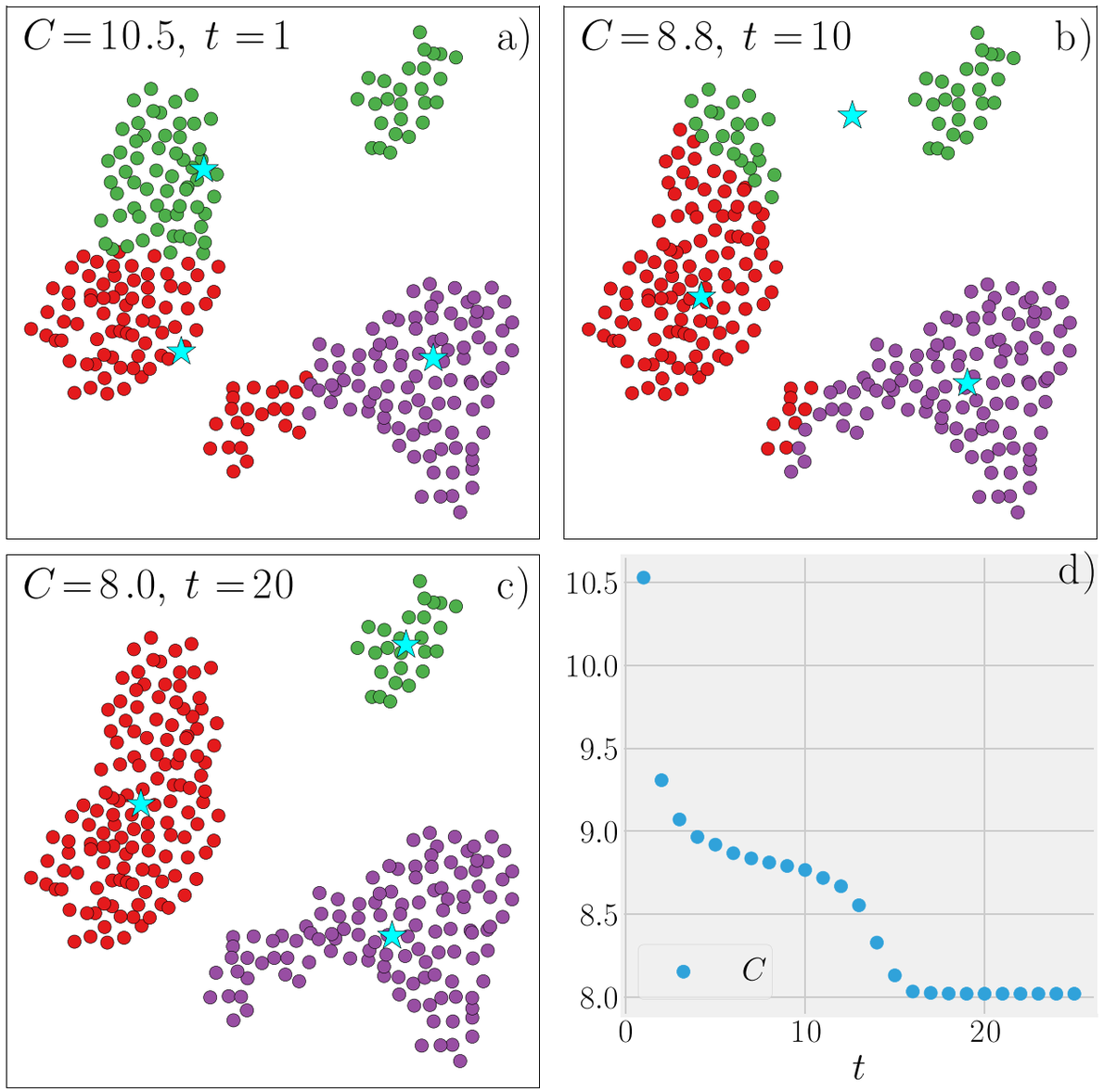


Fig. 55. K-means with $K = 3$ applied to an artificial two-dimensional dataset. The cluster means at each iteration are indicated by cyan star markers. t indicates the iteration number and $\mathcal{C}$ the value of the objective function. (a) The algorithm is initialized by randomly partitioning the space into 3 sectors to generate an initial assignment. (b)–(c) For well separated clusters, the algorithm converges rapidly to the true clusters. (d) The objective function as a function of the iteration. $\mathcal{C}$ converges after $t = 18$ iterations for this choice of random seed (for center initialization).

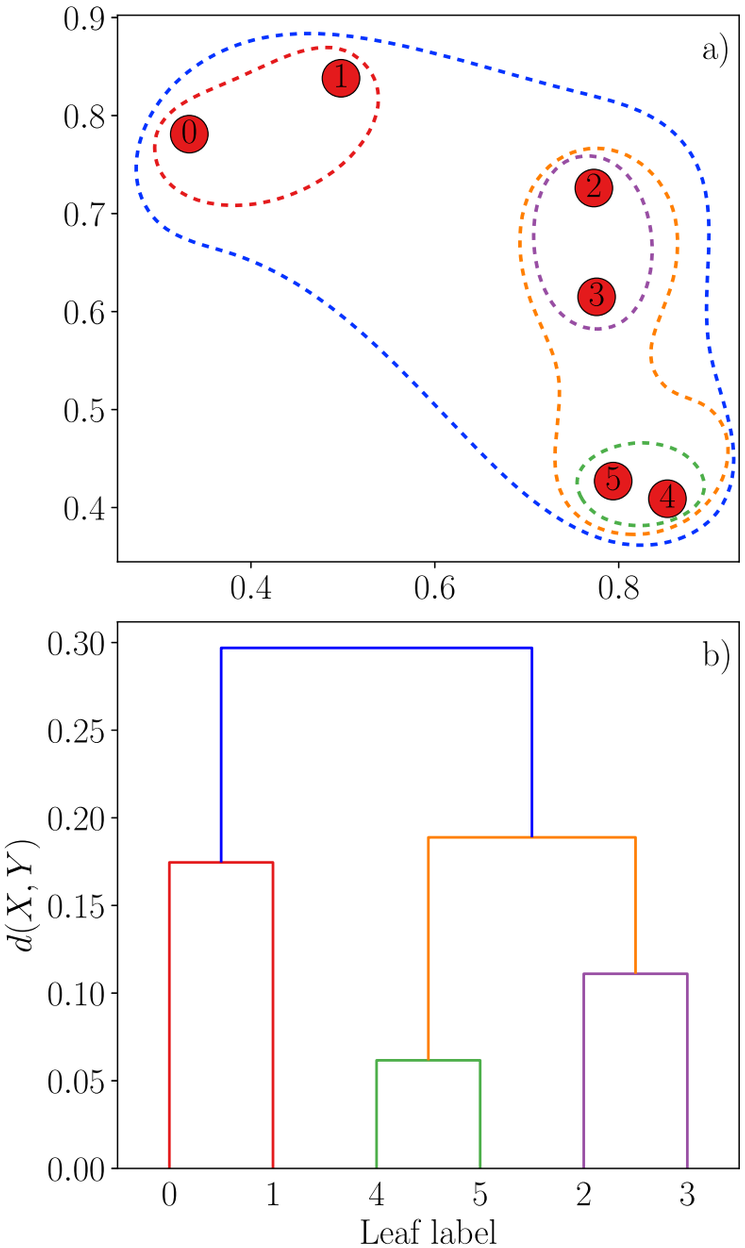
Agglomerative Method

- Start from small initial clusters, then progressively merged to form larger clusters.

- Hierarchy of cluster can be visualized in the form of a **dendrogram**.

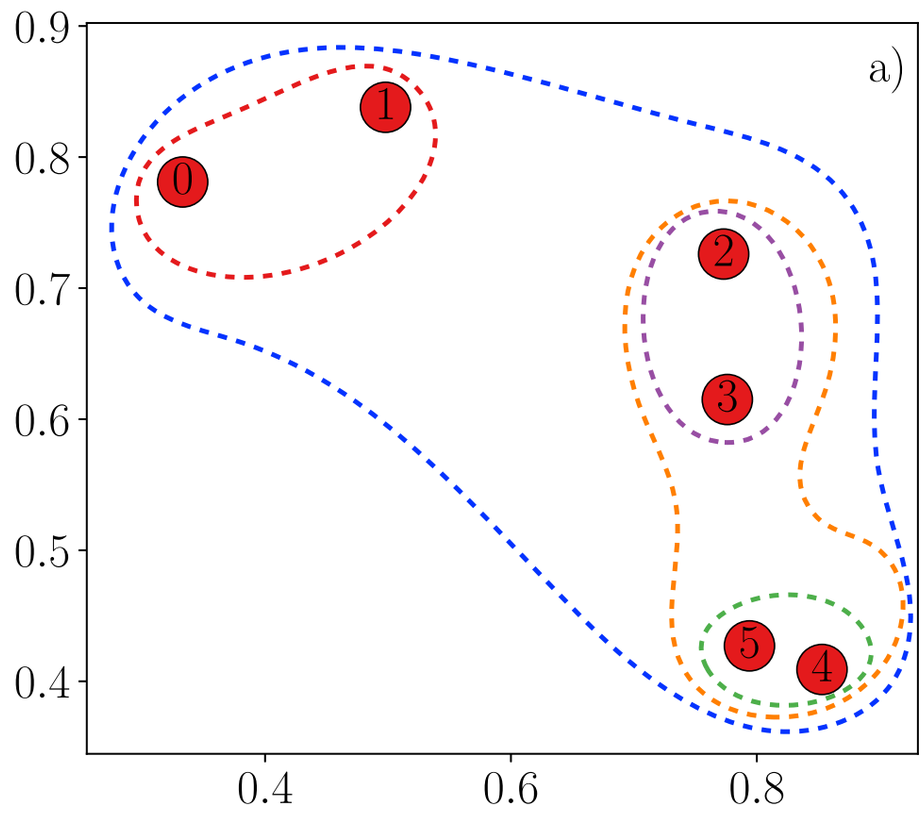
- Define a distance measure $d(X, Y)$ between clusters X and Y.

- Two distances that are closest with respect to $d(X, Y)$ are merged until a single cluster is left.



Agglomerative Clustering Algorithm

- Initialize each point to its own cluster.
- Given a set of K clusters $X_1, X_2, \ldots, X_K$, merge clusters until one cluster is left ($K = 1$):
  - Find the closest pair of clusters $(X_i, X_j) : (i, j) = \text{argmin}_{(i',j')} d(X_{i'}, X_{j'})$
  - Merge the pair. Update $K \leftarrow K - 1$.
- Different linkage methods (distances) result in different algorithms (next page).



Agglomerative Clustering Algorithm

- Different linkage methods (distances):

Single linkage $$d(X_i, X_j) = \min_{\mathbf{x}_i \in X_i, \mathbf{x}_j \in X_j} ||\mathbf{x}_i - \mathbf{x}_j||_2$$

Complete linkage $$d(X_i, X_j) = \max_{\mathbf{x}_i \in X_i, \mathbf{x}_j \in X_j} ||\mathbf{x}_i - \mathbf{x}_j||_2$$

Average linkage $$d(X_i, X_j) = \frac{1}{|X_i| \cdot |X_j|} \sum_{\mathbf{x}_i \in X_i, \mathbf{x}_j \in X_j} ||\mathbf{x}_i - \mathbf{x}_j||_2$$

Ward linkage $$d(X_i, X_j) = \frac{|X_i||X_j|}{|X_i \cup X_j|} (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^2,$$

- The Wald linkage is analogous to k-means in that it minimizes the moment of inertia.

- Problem: Calculation complexity $\mathcal{O}(N^2)$ (suitable for small datasets)

- Practical solution: start with k-means and then proceed with hierarchical (agglomerative) clustering.

Density-based (DB) Clustering

- Clusters are defined by regions with high density of data points.
- Noise or outliers are expected to form regions of low density.
- Unlike a distance-based approach, DB clustering considers clusters of multiple shapes and sizes while identifying outliers.
- Assumption: relative local density estimation is possible (normally inaccessible for high-dimensional data due to large sampling noise).
- Widely used algorithms: DBSCAN, DB Clustering, etc. See: https://pypi.org/project/fdc/

DBScan Algorithm

- Density-based spatial clustering of application with noise (Ester et al, 1996).

- Crude estimate of local density is the ϵ-neighborhood of point $\mathbf{x}_n$:

$$N_\varepsilon(\mathbf{x}_n) = \{\mathbf{x} \in X | d(\mathbf{x}, \mathbf{x}_n) < \varepsilon\}$$

- $\mathbf{x}_n$ is a **core-point** if at least **minPts** are in its ϵ-neighborhood. A point $\mathbf{x}_i$ is **density-reachable** if it's in a core-point's ϵ-neighborhood.

- $\rightarrow$ 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.

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

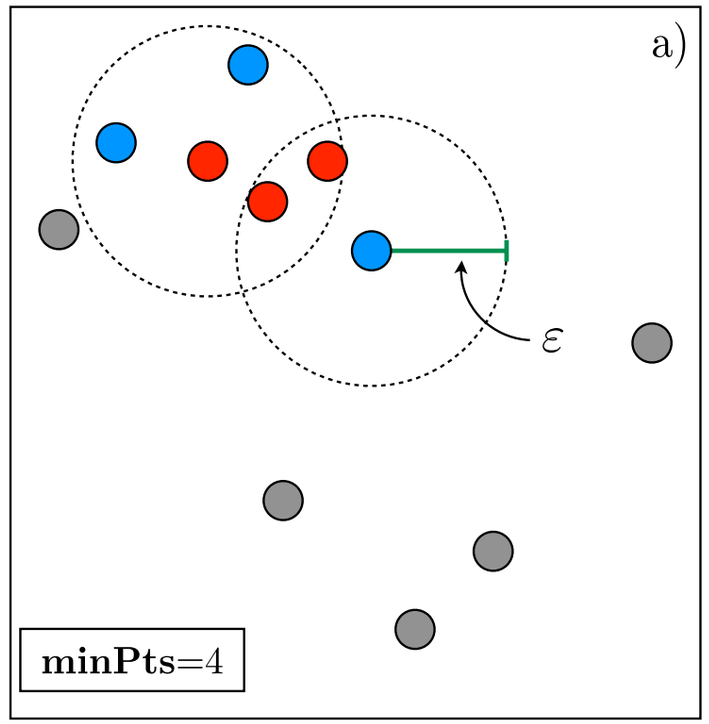
DBScan Algorithm

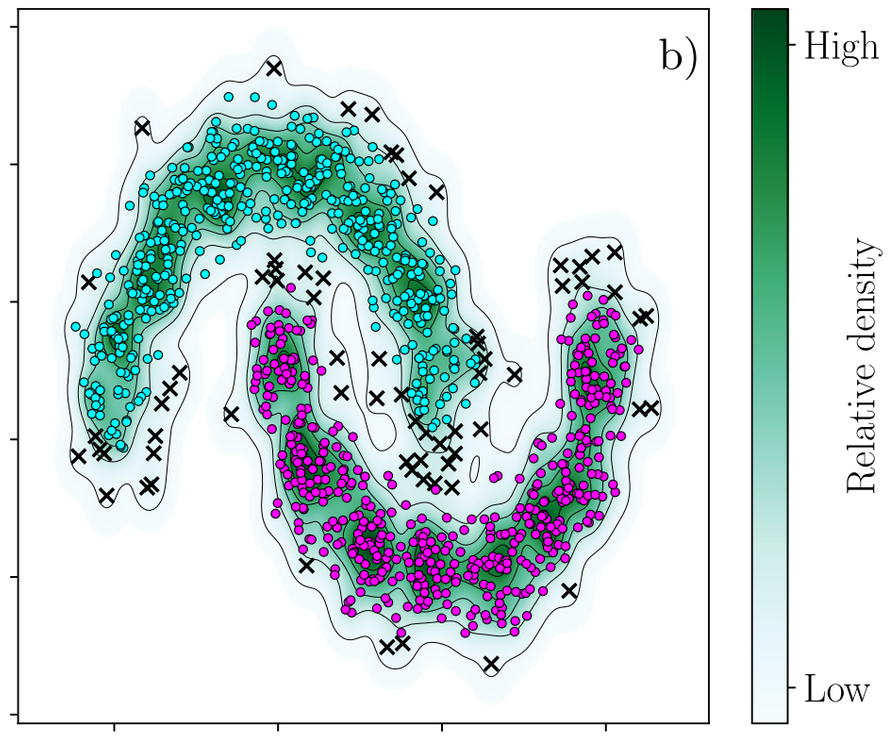
- Do not need to specify # clusters but only the hyperparameters ϵ and **minPts**.

- Scalable to large datasets as computational cost $\sim \mathcal{O}(N \log N)$.

- Note cluster with different shapes and sizes.

- Crosses are outliers.





Latent Variables

- Central to unsupervised learning is the idea of a latent or hidden variable (not directly observable; yet influence visible structure).

- The cluster identify of each datapoint is a latent variable. We cannot observe the label directly, but points in the same cluster are “close”.

- In this abstract language, clustering is an algorithm to learn the most probably value of a latent variable associated with each datapoint.

- Need to make assumption about the structure of data (common to unsupervised learning), e.g., underlying probability distribution from which the data was generated — generative model.

- E.g., in clustering, each cluster is characterized by some probability distribution (e.g. Gaussian distribution with some mean & variance). The latent variable is chosen to minimize some cost function.