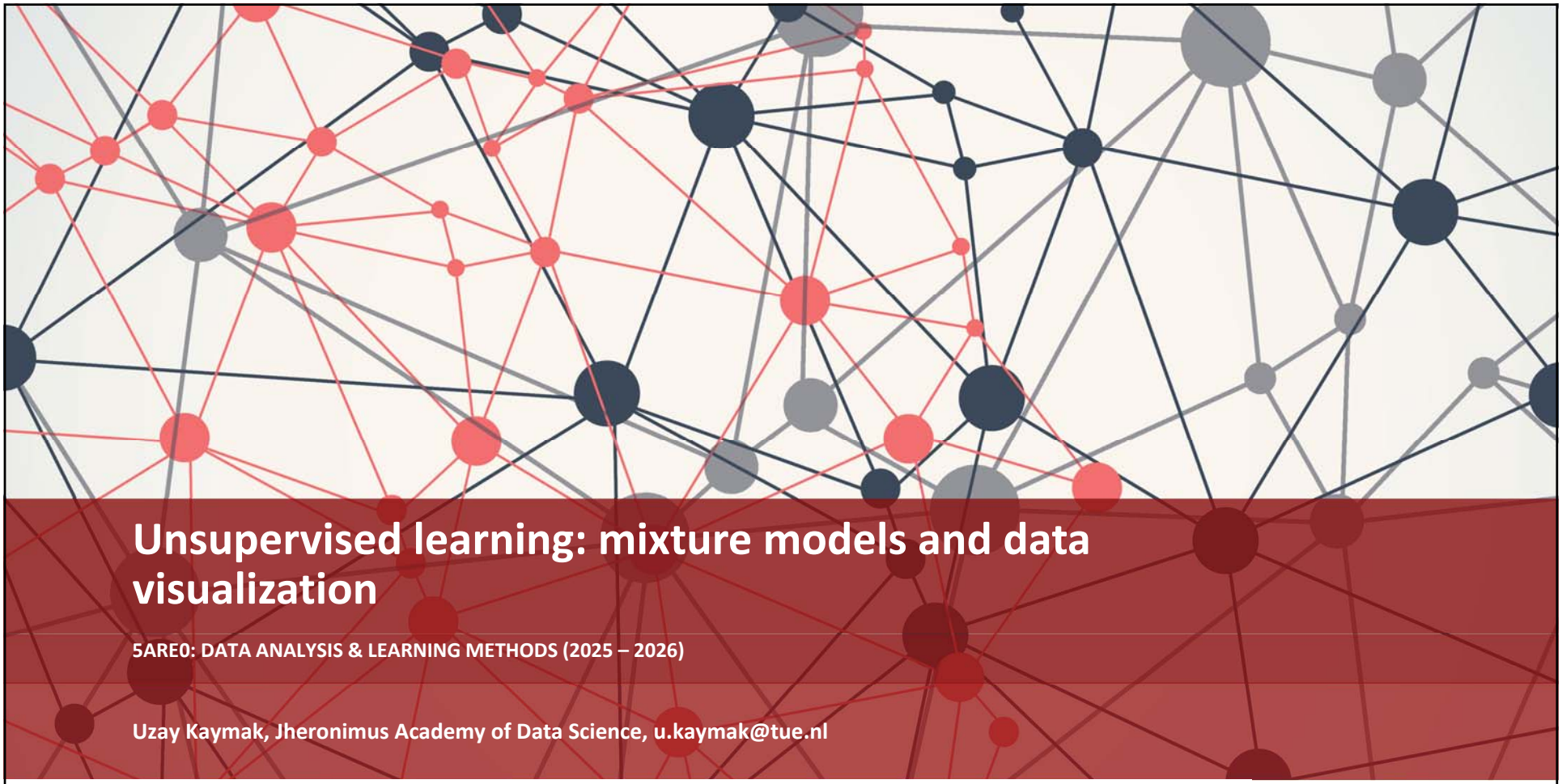


Unsupervised learning: mixture models and data visualization

5ARE0: DATA ANALYSIS & LEARNING METHODS (2025 – 2026)

Uzay Kaymak, Jheronimus Academy of Data Science, u.kaymak@tue.nl

Master: Artificial Intelligence & Engineering Systems

Includes slides courtesy of W. Kouw

Recap

- Clustering: group data such that within group similarity I slarger than between group similarity
- Hierarchical clustering: generates a tree graph
- K-means: minimizes total cluster scatter
- Fuzzy clustering: assigns data to multiple clusters with different membership
- Fuzzy clustering algorithms: FCM, GK, PCM
- Cluster validity measures help estimate correct number of clusters
- Cluster merging also helps determine correct number of clusters

2

TU/e

Outline

- **DBSCAN clustering**
- **Gaussian Mixture Models**
- **Data visualization**
  - Sammon maps
  - t-SNE

4

TU/e

Clustering (definition)

Divide the data into groups based on similarity, such that …

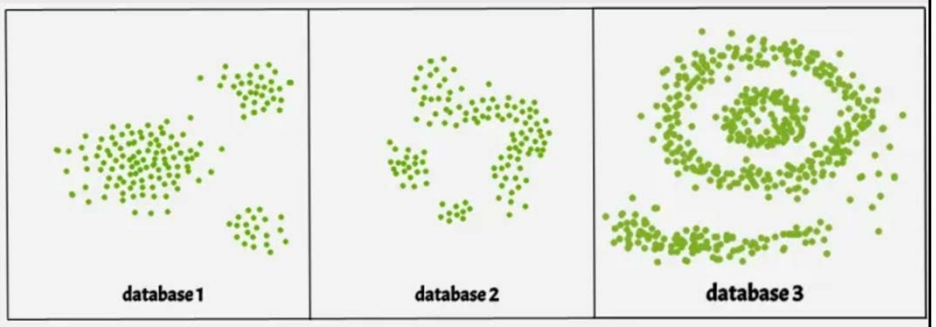
within group similarity is larger than …

between group similarity.

5

Density Based Scan (DBSCAN)

- **Groups data points that are closely packed**
  - Clusters are locations with dense distributions
  - Clusters are separated with less dense regions
- **Properties of the algorithm**
  - Deals with arbitrary shaped clusters
  - Identifies noise and outliers



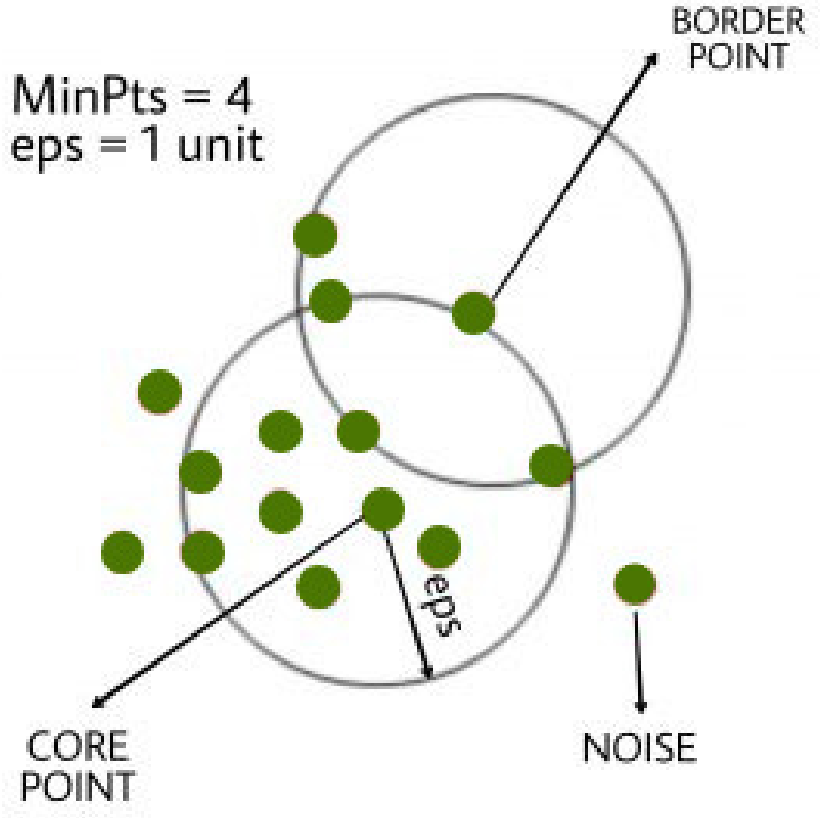
https://www.geeksforgeeks.org/machine-learning/dbscan-clustering-in-ml-density-based-clustering/

6

TU/e

DBSCAN core idea

- **Two parameters:**
  - Eps: defines radius of a neighborhood
  - MinPts: minimum number of points within the eps radius to define it as dense region
- **Categorize data points**
  - Core points: data in a dense region
  - Border points: near core points, but without enough neighbors
  - Noise: anything else



https://www.geeksforgeeks.org/machine-learning/dbscan-clustering-in-ml-density-based-clustering/

TU/e

DBSCAN algorithm

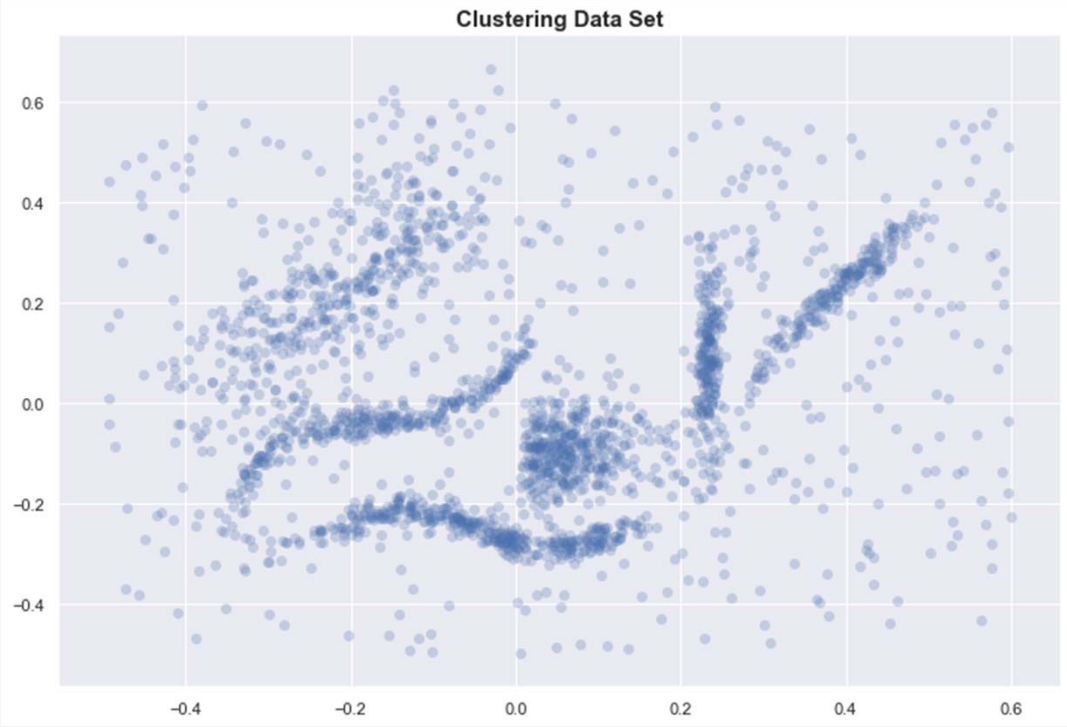
1. Identify a core point and form a cluster
2. For all points in the eps neighborhood of the core, add them to the cluster as core (more than MinPts points) or as border point (less than MinPts points)
3. When no more points can be added, repeat 1. and 2. with the remaining points and form a new cluster, until no new clusters can be formed
4. Label all unassigned clusters as noise

Note that DBSCAN can be computationally expensive

8

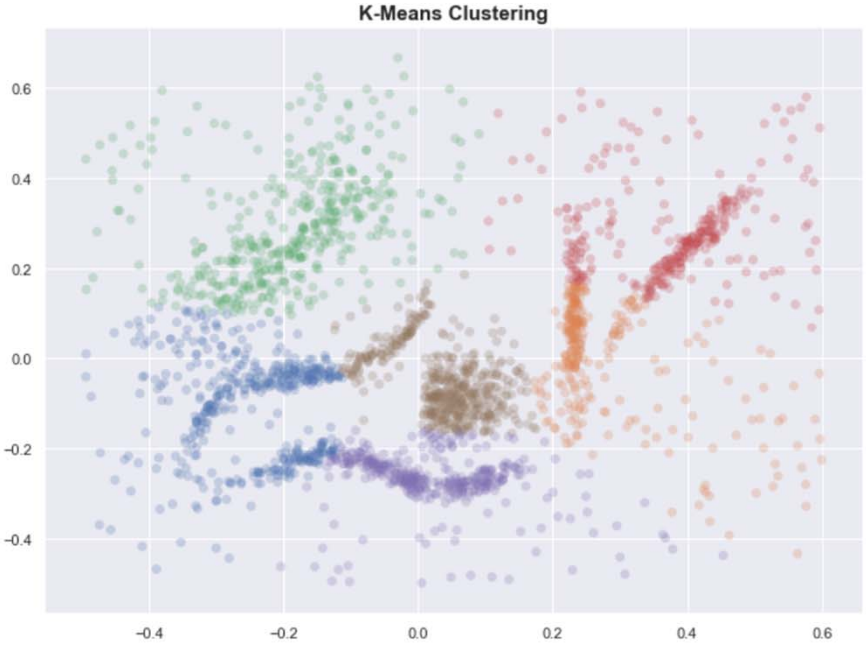
TU/e

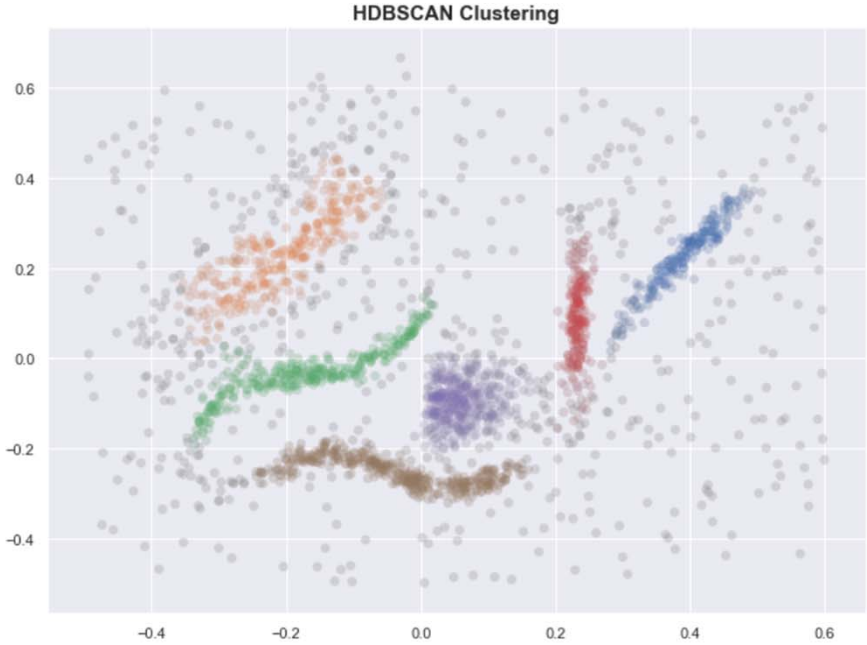
Hierarchical Density-Based Spatial Clustering



Source: https://pberba.github.io/stats/2020/01/17/hdbscan/

HDensityBasedSCAN

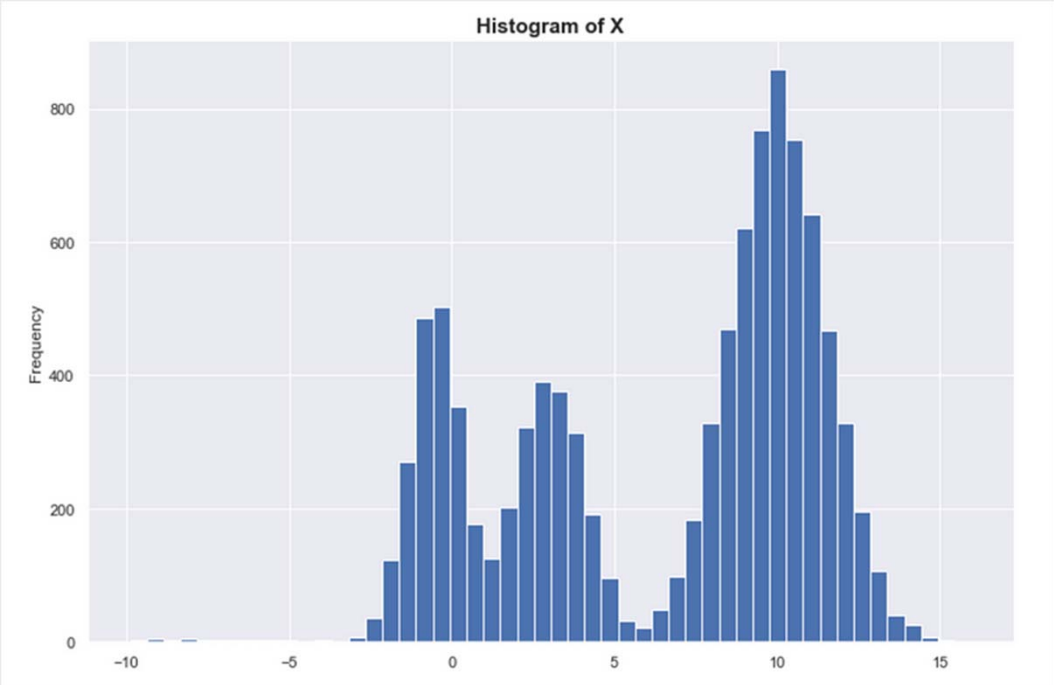




Source: https://pberba.github.io/stats/2020/01/17/hdbscan/

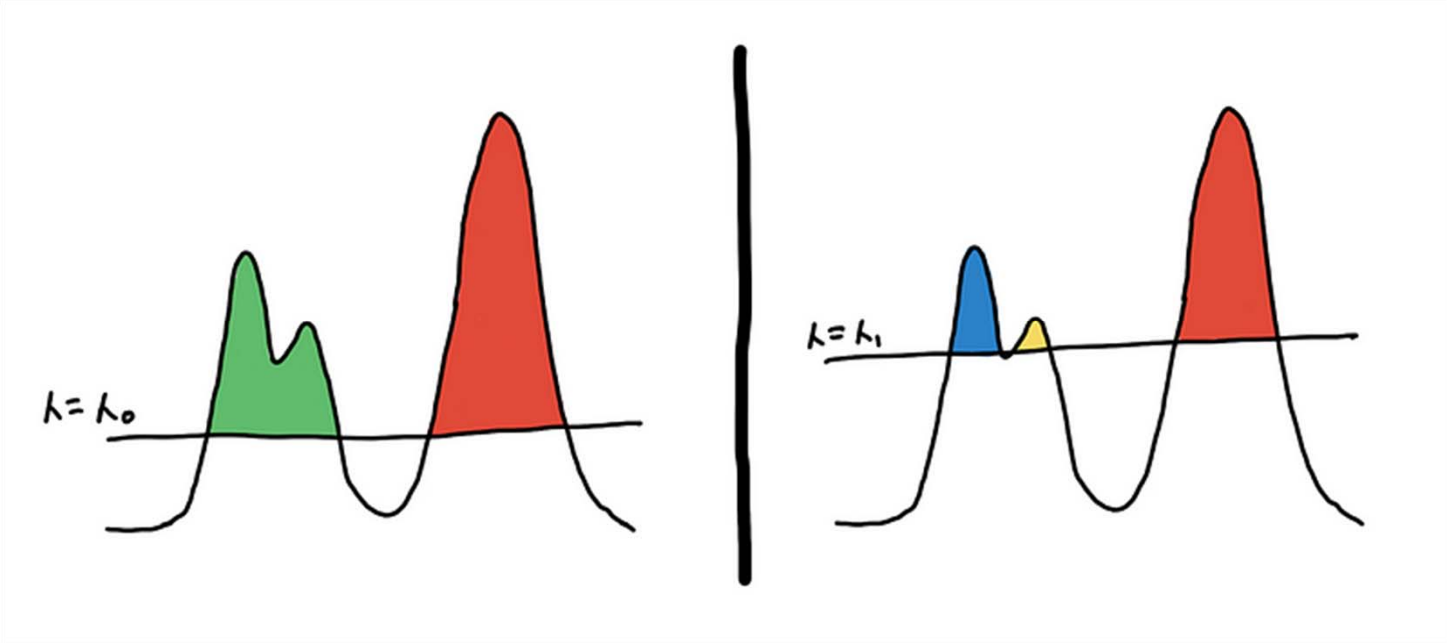
HierarchicalDBSCAN

How many clusters are in the histogram below?



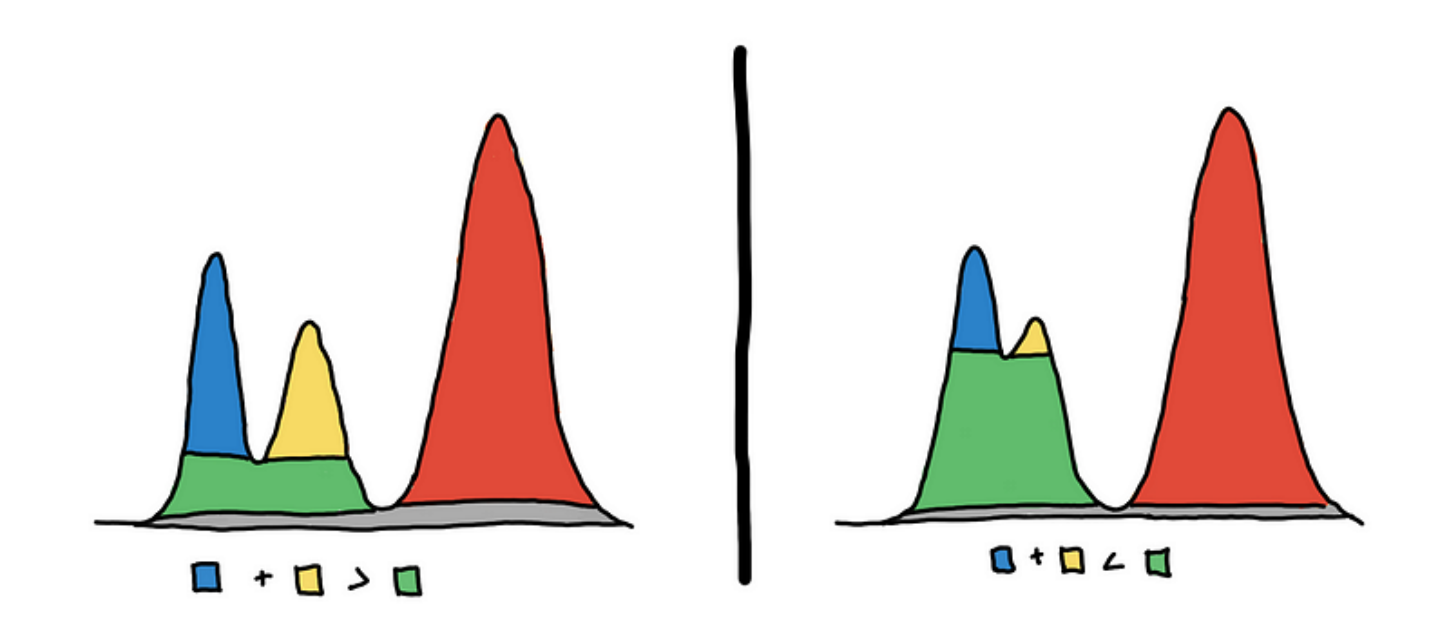
Source: https://pberba.github.io/stats/2020/01/17/hdbscan/

Based on the threshold, there are 2 or 3 clusters



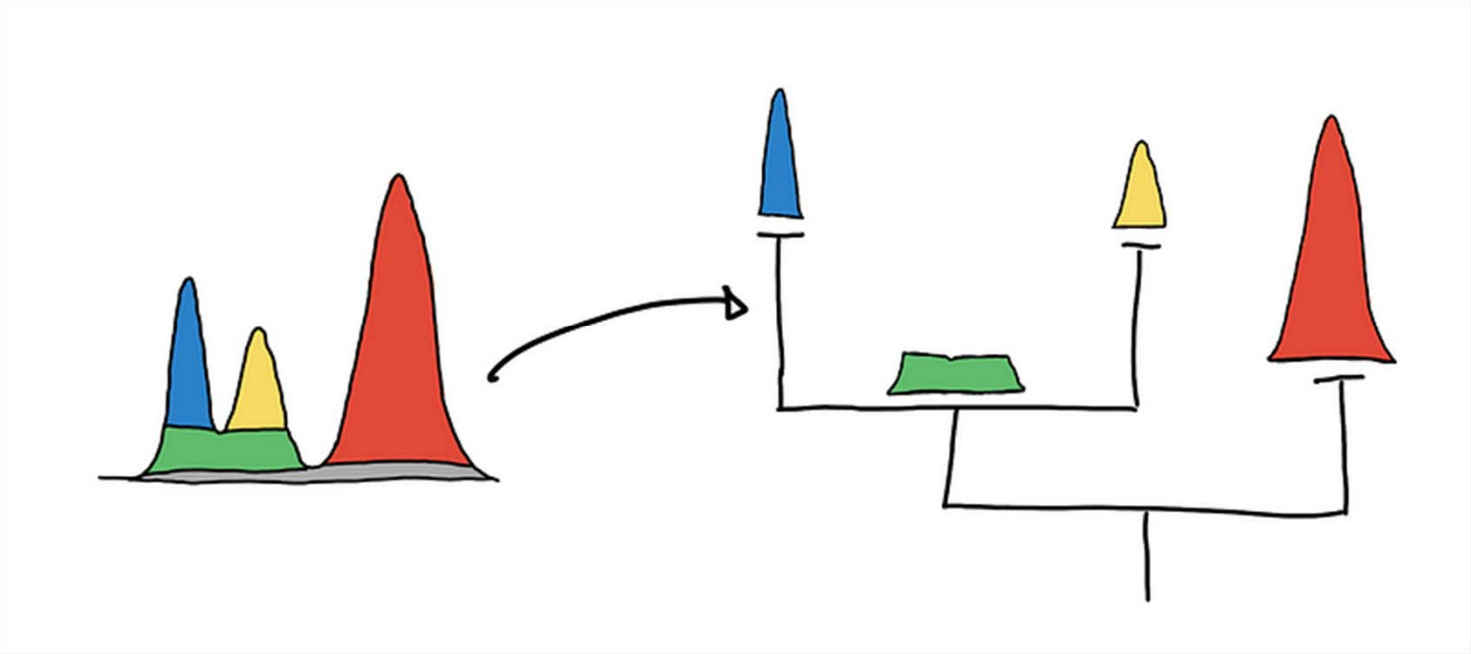
Source: https://pberba.github.io/stats/2020/01/17/hdbscan/

HDBSCAN decides the number of clusters by assessing which parts 'persists' more



Source: https://pberba.github.io/stats/2020/01/17/hdbscan/

By creating hierarchical trees, the model should be able to detect clusters of varying densities



Source: https://pberba.github.io/stats/2020/01/17/hdbscan/

Gaussian mixture models

Unsupervised learning

Unsupervised learning **refers to improving the performance of mathematical models with respect to some task using data, but without labels or without responses.**

- Unsupervised learners can also do clustering.

Unsupervised learning is a general framework where we often want to:

- Find hierarchical factors, such as clusters of clusters.
- Estimate uncertainties / point spreads in latent spaces.
- Detect anomalies / outliers / surprising events.
- Generate new samples from a learned representation.

17

TU/e

Unsupervised learning

Typically, the kinds of models where unsupervised learning is applied to assume that the data was generated according to an unknown or "latent" set of variables.

- There is often a complicated functional relationship, characterized by parameters.

Examples include:

- Mixture models: the observed data is a mixture of a latent set of components.
- Autoencoders: the observed data was transformed non-linearly from latent factors.

We "fit" models to data by optimizing the parameters with respect to the probability of the observed data or the prediction error of the model on unseen data.

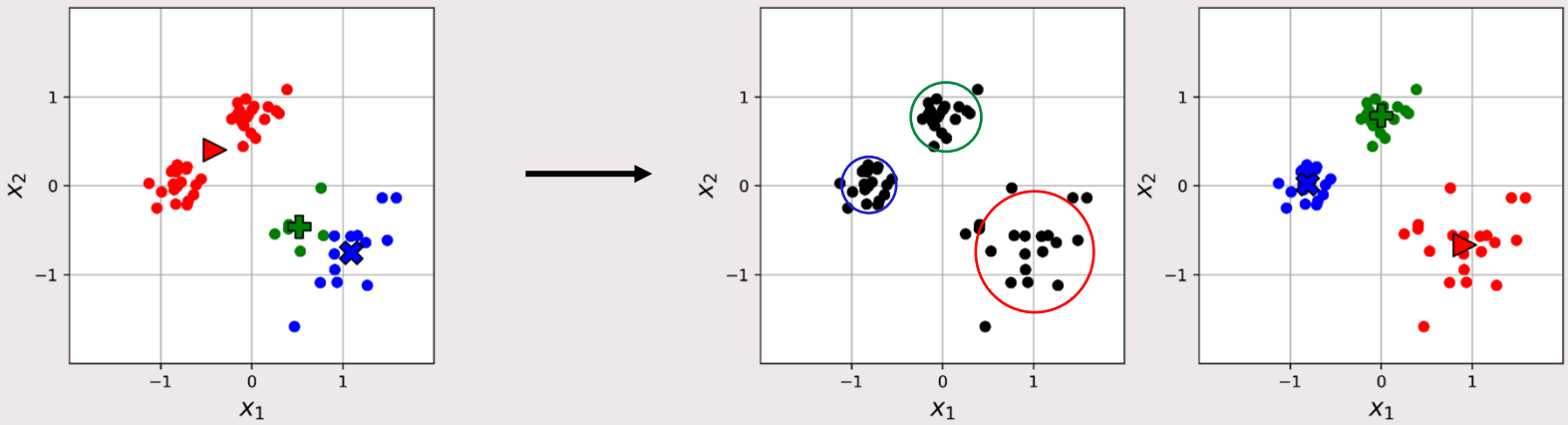
18

TU/e

Unsupervised learning

We saw that conventional K-Means fails to account for cluster size.

With mixture models, we can take the cluster density into account seamlessly.



19

Mixture models

A mixture model is a probability distribution that consists of a weighted combination of K "component" probability distribution functions:

$$p(x) = \sum_{k=1}^{K} \pi_k \, p_k(x)$$

where $p_k(x)$ is the k-th component and the π_k are the weights.

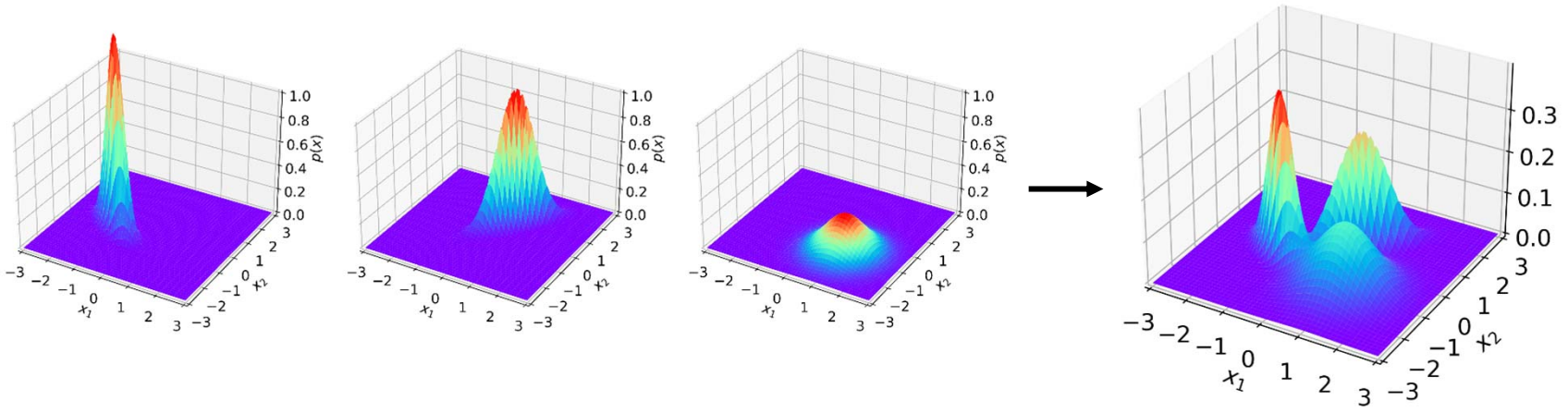
The weights π_k are fixed to be numbers between 0 and 1, that sum to 1.

TU/e

Mixture models

The most common variant is the *Gaussian mixture model* (GMM), which uses Gaussian probability distribution functions as components:

$$p_k(x) = \mathcal{N}(x \mid \mu_k, \Sigma_k)$$



Mixture models

Q: How do we "fit" this mixture model to a data set?

Expectation-maximization.

EM is an elegant estimation procedure with roots in statistical inference.

For mixture models, the procedure boils down to the same steps as in K-means:

1. "Assign" data points to component distributions.
2. Update the parameters of each component distribution.

22

TU/e

Mixture models

1. **"Assign" data points to component distributions.**

We define a latent variable called the *responsibility* of component k for data point i:

$$r_{ik} = \frac{\pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}$$

The responsibilities represent the probabilities under each component normalized by the total probability, i.e., the sum of probabilities under components.

For example, for probabilities $p_1(x_i)$ = 0.2 and $p_2(x_i)$ = 0.4 with weights $\pi_1 = \pi_2 = \frac{1}{2}$,

$$r_{i1} = \frac{\frac{1}{2}0.2}{\frac{1}{2}0.2+\frac{1}{2}0.4} = \frac{1}{3} \text{ and } r_{i2} = \frac{\frac{1}{2}0.4}{\frac{1}{2}0.2+\frac{1}{2}0.4} = \frac{2}{3}.$$

TU/e

Mixture models

In k-means, we had an assignment variable indicating which cluster the data point x_i belonged to, $z_i \in \{1 \dots K\}$.

- That can be re-written as a one-hot vector, e.g., $\tilde{z}_i = [0\ 0\ 1]$.
- That resembles the responsibilities, e.g., $r_i = [0.2\ \ 0.3\ \ 0.5]$.

The responsibilities are often considered "soft assignments"; the data points do not belong to only one component, but rather to all components simultaneously.

- This ensures robustness to outlying data points.
- This is quite similar to fuzzy clustering.

24

TU/e

Mixture models

2. Update parameters of each component distribution.

The likelihood of a GMM with K components on N samples is:

$$p(X|\theta) = \prod_{i=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)$$

where $\theta = (\pi_1, \dots \pi_K, \mu_1, \dots \mu_K, \Sigma_1, \dots \Sigma_K)$.

We will use maximum-likelihood to estimate the parameters:

$$\theta^* = \arg\max_{\theta} \ln p(X|\theta)$$

where $\ln$ represents the natural logarithm.

25

TU/e

Mixture models

Q: Why do we maximize the log-likelihood and not just the likelihood?

Because products become sums and exponents become multiplications:

$$\ln p(X|\theta_k) = \sum_{i=1}^{N} \ln \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)$$

Note that the maximizer is invariant to the logarithm,

$$\arg\max_{\theta} \ln p(X|\theta) = \arg\max_{\theta} p(X|\theta),$$

because the logarithm is a monotonically increasing function.

26

TU/e

Mixture models

First, we take the derivative of the likelihood with respect to the means:

$$\frac{\partial}{\partial \boldsymbol{\mu}_k} \ln p(\boldsymbol{X}|\boldsymbol{\theta}) = \sum_{i=1}^{N} \frac{1}{p(\boldsymbol{x}_i|\boldsymbol{\theta})} \frac{\partial p(\boldsymbol{x}_i|\boldsymbol{\theta})}{\partial \boldsymbol{\mu}_k}$$

$$= \sum_{i=1}^{N} \frac{1}{\sum_{k=1}^{K} \boldsymbol{\pi}_k \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)} \boldsymbol{\pi}_k (\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Using the definition of the responsibilities, we get:

$$\frac{\partial}{\partial \boldsymbol{\mu}_k} \ln p(\boldsymbol{X}|\boldsymbol{\theta}) = \sum_{i=1}^{N} \boldsymbol{r}_{ik} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}$$

Mixture models

If we set that derivative to 0, we get:

$$\sum_{i=1}^{N} r_{ik}(x_i - \mu_k)^T \Sigma_k^{-1} = 0$$

$$\sum_{i=1}^{N} r_{ik} x_i = \sum_{i=1}^{N} r_{ik} \mu_k$$

$$\mu_k = \sum_{i=1}^{N} \frac{r_{ik} x_i}{\sum_{i=1}^{N} r_{ik}}$$

which resembles the arithmetic mean operation in k-means.

28

Mixture models

Next, we have take the derivative of the likelihood with respect to the covariances:

$$\frac{\partial}{\partial \boldsymbol{\Sigma}_k} \ln p(\boldsymbol{X}|\boldsymbol{\theta}) = \sum_{i=1}^{N} \frac{1}{p(\boldsymbol{x}_i|\boldsymbol{\theta})} \frac{\partial p(\boldsymbol{x}_i|\boldsymbol{\theta})}{\partial \boldsymbol{\Sigma}_k}$$

This follows the same line of reasoning as with the update for the means but with much more terms and steps.

I will skip these and give the final result directly:

$$\boldsymbol{\Sigma}_k = \sum_{i=1}^{N} \frac{\boldsymbol{r}_{ik}}{\sum_{i=1}^{N} \boldsymbol{r}_{ik}} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T (\boldsymbol{x}_i - \boldsymbol{\mu}_k)$$

29

TU/e

Mixture models

Lastly, for the mixture weights, we also need to consider the constraint that they should sum to 1, which we can do by forming a Lagrangian:

$$\mathcal{L} = \ln p(X|\theta) + \lambda\left(\sum_{k=1}^{K} \pi_k - 1\right)$$

The optimal solution can be obtained by setting the partial derivatives with respect to π_k and λ to 0 and solving simultaneously:

$$\frac{\partial \mathcal{L}}{\partial \pi_k} = 0 \text{ and } \frac{\partial \mathcal{L}}{\partial \lambda} = 0$$

30

TU/e

Mixture models

The first partial derivative is:

$$\frac{\partial \mathcal{L}}{\partial \pi_k} = \sum_{i=1}^{N} \frac{1}{p(x_i|\theta)} \frac{\partial p(x_i|\theta)}{\partial \pi_k} + \frac{\partial \mathcal{L}}{\partial \pi_k} \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right)$$

$$= \sum_{i=1}^{N} \frac{\mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)} + \lambda$$

We can again use our definition of the responsibilities to get:

$$\frac{\partial \mathcal{L}}{\partial \pi_k} = \sum_{i=1}^{N} \frac{r_{ik}}{\pi_k} + \lambda$$

31

TU/e

Mixture models

The second partial derivative is:

$$\frac{\partial \mathcal{L}}{\partial \lambda} = \sum_{k=1}^{K} \pi_k - 1$$

Setting both to 0 yields the system of equations:

$$\sum_{i=1}^{N} \frac{r_{ik}}{\pi_k} = -\lambda$$

$$\sum_{k=1}^{K} \pi_k = 1$$

TU/e

Mixture models

Re-arranging the first equation gives:

$$-\sum_{i=1}^{N}\frac{r_{ik}}{\lambda}=\pi_k$$

We may plug this result in the second equation and solve for λ:

$$-\sum_{k=1}^{K}\sum_{i=1}^{N}\frac{r_{ik}}{\lambda}=1$$

$$-\frac{N}{\lambda}=1$$

$$\lambda=-N$$

TU/e

Mixture models

Plugging the optimal Lagrange multiplier into the partial derivative gives:

$$\sum_{i=1}^{N} \frac{r_{ik}}{\pi_k} - N = 0$$

Which leads to a final result of:

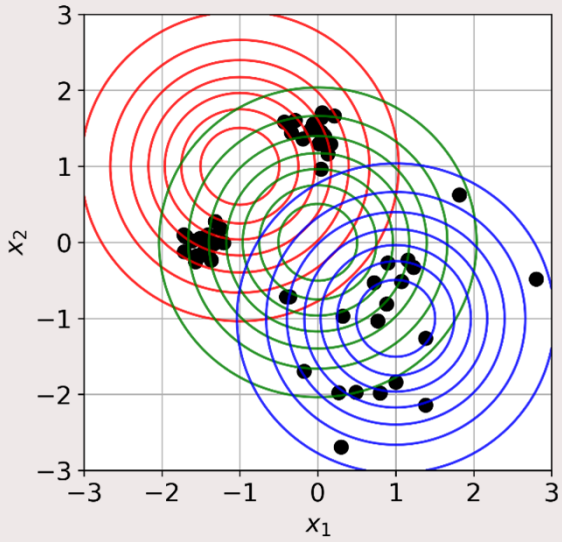
$$\pi_k = \frac{1}{N} \sum_{i=1}^{N} r_{ik}$$

So, the mixture weights can be interpreted as the "average responsibility" of each component.

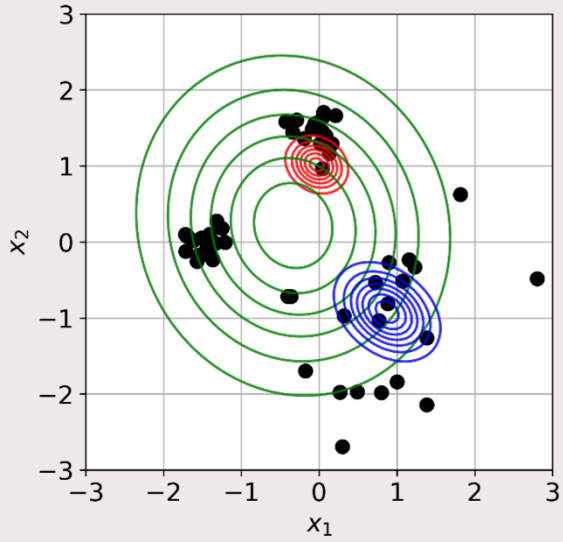
TU/e

Mixture models

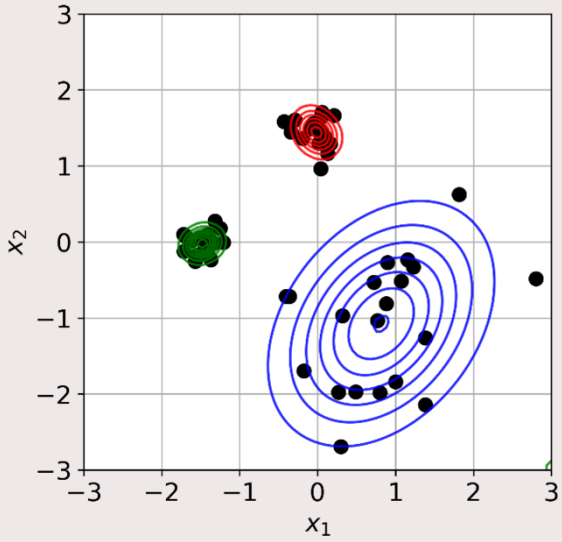
EM for a GMM may evolve as follows:



Initial configuration



After 1 iteration



At convergence

35

Limitations of Gaussian mixture models

1. **The total optimization procedure is not convex.**
   - Different starting positions may lead to different local optima.
   - Hence, the initial parameters affect the final estimates.
2. **Numerical stability can be a problem.**
   - During optimization, a GMM component might focus on a single data point, which will cause the covariance matrix to shrink to having eigenvalues of 0.
   - This covariance matrix with eigenvalues 0 has to be inverted which will typically result in numerical under- and overflow problems.
   - Regularization methods could be used to resolve this problem.

TU/e

Data visualization

Data visualization

Projection of high dimensional data into small dimensional space allows for visualization

Most visualization methods project to 2-D or 3-D space

Example methods

- Taking 2-D slices of the data
- PCA
- Multidimensional scaling
- t-SNE

39

TU/e

Multi-dimensional scaling

Project higher dimensional data into lower dimensions in such a way that distances in the original space are preserved as much as possible in the projection

- **E.g. Sammon maps**
- **Minimize Sammon stress:**

$$E = \frac{1}{\sum_{i<j} d^*_{ij}} \sum_{i<j} \frac{(d^*_{ij} - d_{ij})^2}{d^*_{ij}}$$

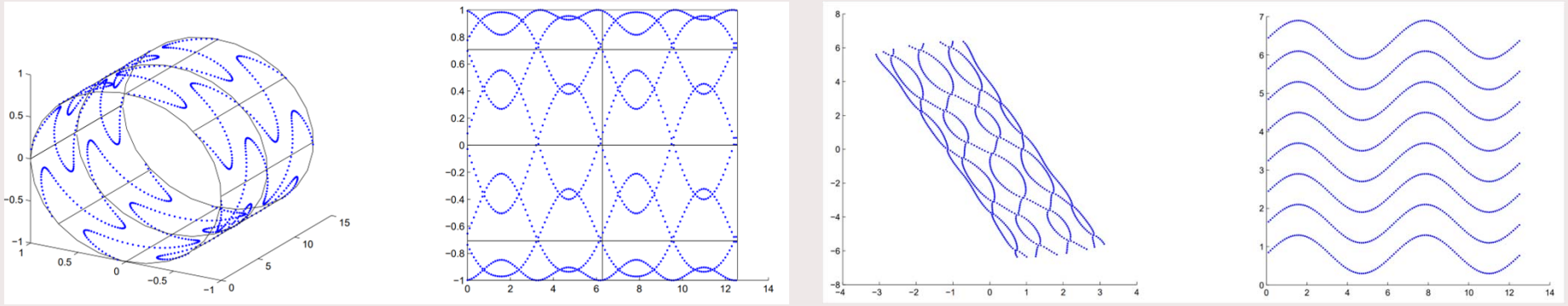
Distance in the original space

Distance in the projected space (between data i and j)

- **Minimization can be done by gradient descent**

TU/e

Sammon mapping – example



Original data PCA Sammon mapping Cylinder unrolled

Sammon maps try to preserve the topology as much as possible

41

t-SNE – t-distributed Stochastic Neighbor Embedding

- **Unsupervised, non-linear dimensionality reduction technique**
- **Conceptually, somewhat similar to Sammon map, but optimizes a probability-distribution based metric**

$$C = D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{P(x)}{Q(x)} \right)$$

Probability distribution in the original space

Probability distribution in the projection space

Kullback-Leibler divergence

42

TU/e

t-SNE algorithm

Check the recommended literature on Canvas for algorithm details

Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

Data: data set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, \ldots, y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, \ldots, y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
  for $t=1$ **to** T **do**
    compute low-dimensional affinities q_{ij} (using Equation 4)
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
  end
end

43

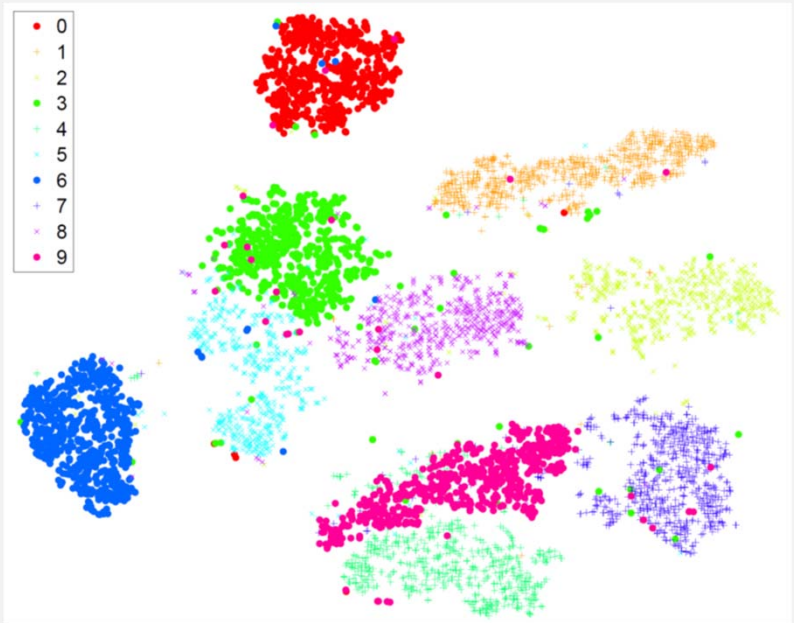
t-SNE vs PCA

Characteristic	t-SNE	PCA
Type	Non-linear dimensionality reduction	Linear dimensionality reduction
Goal	Preserve local pairwise similarities	Preserve global variance
Best used for	Visualizing complex, high-dimensional data	Data with linear structure
Output	Low-dimensional representation	Principal components
Use cases	Clustering, anomaly detection, NLP	Noise reduction, feature extraction
Computational intensity	High	Low
Interpretation	Harder to interpret	Easier to interpret

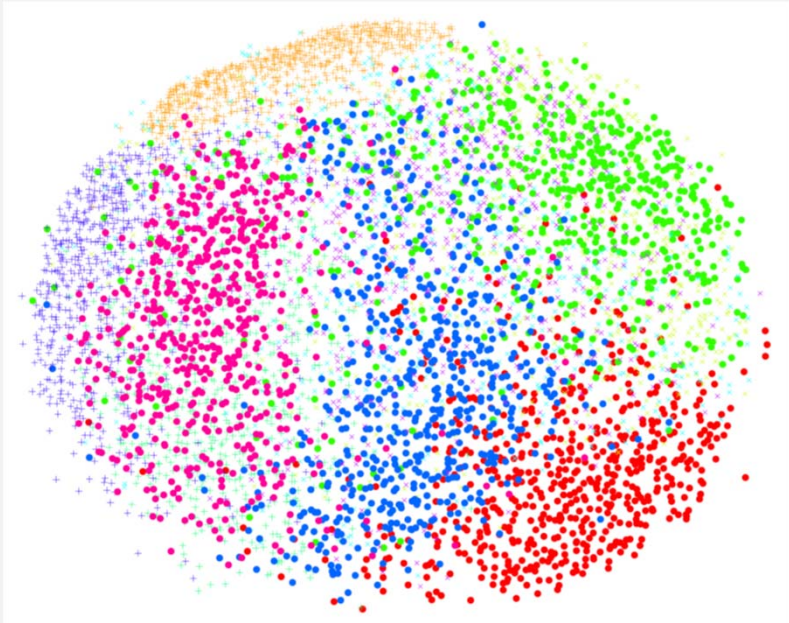
44

https://www.datacamp.com/tutorial/introduction-t-sne

Visualization of MNIST data (hand-written digits)



t-SNE



Sammon map

45

Recap

- DBSCAN
  - clusters data based on data density
  - Can deal with arbitrary cluster shapes and noise
- Gaussian Mixture Models
  - Approximate data distribution as a mixture of finite number of Gaussian distributions
- Visualization of high-dimensional data
  - Sammon maps
  - t-SNE

46

TU/e

Questions?

47