# Practical introduction to machine learning

## Part 2 : Unsupervised learning

**Rémi Flamary** - CMAP, École Polytechnique

Master Data Science, Institut Polytechnique de Paris

September 12, 2022

# Overview of MAP654I

## 1. Data and Machine Learning problems

- Data properties and visualization
- Pre-processing
- Finding your Machine Learning problem

## 2. Unsupervised learning

- Clustering
- Density estimation and generative modeling
- Dictionary learning and collaborative filtering
- Dimensionality reduction and manifold learning

## 3. Supervised learning

- Bayesian decision and Nearest neighbors
- Linear models nonlinear methods for regression and classification
- Trees, forest and ensemble methods

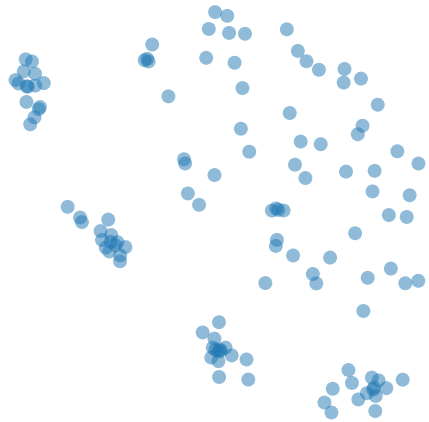
## 4. Validation and interpretation

- Performance measures
- Models and parameter selection (validation)
- Interpretation of the methods

# Overview for the current part

# Unsupervised dataset

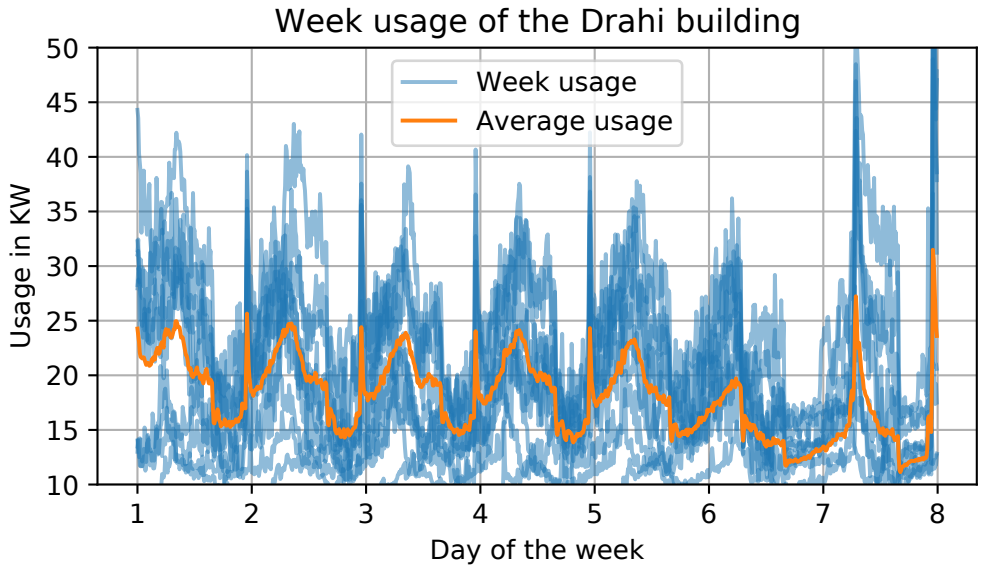
$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_i^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1j} & \dots & x_{1d} \\ x_{21} & x_{22} & \dots & x_{2j} & \dots & x_{2d} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{i1} & x_{i2} & \dots & x_{ij} & \dots & x_{id} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nj} & \dots & x_{nd} \end{bmatrix}$$



## Unsupervised learning

- The dataset contains the samples $\{\mathbf{x}_i\}_{i=1}^n$ with $n$ samples of size $d$.
- $d$ and $n$ define the dimensionality of the learning problem.
- Data stored as a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ with $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^\top$ containing the transposed training samples as lines (features are in columns).
- Note: in the course we use 1-based indexing as standard in math but in python 0-based indexing is used.

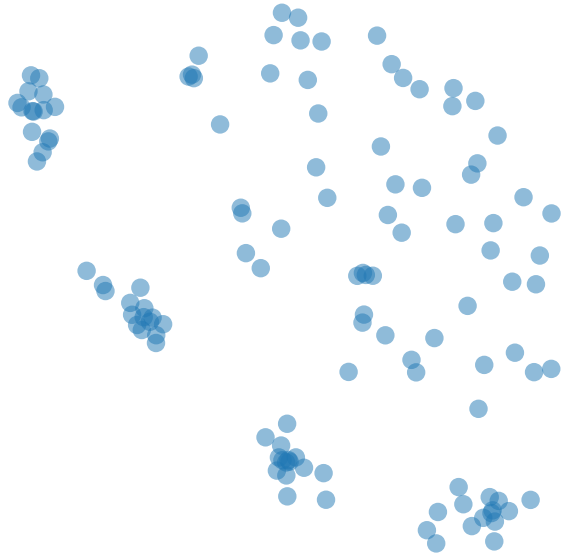
# Example of real life dataset




## Electrical usage of the Drahi X-Novation Center

- Demonstrator of Energy4Climate of IP Paris.
- Recording of the electrical usage of the building during 1.5 years.
- Can be completed by weather measurement (linked to energy usage).
- Data will be used on samples of energy usage during 1 week.
- Note that some pre-processing of the data is necessary before getting the unsupervised of supervised datasets.

# Unsupervised learning, data description/exploration



## Different problems (many methods can solve several of them)

- **Clustering**
  Group in clusters the similar samples.
- **Probability density estimation**
  Estimate from finite samples a probability distribution.
- **Generative modeling**
  Learn model that can generate data similar to the samples.
- **Dimensionality reduction**
  Reduce the dimensionality of the data for visualization or interpretation/modeling.

# Scikit-learn estimator for unsupervised learning

## Scikit-learn object API

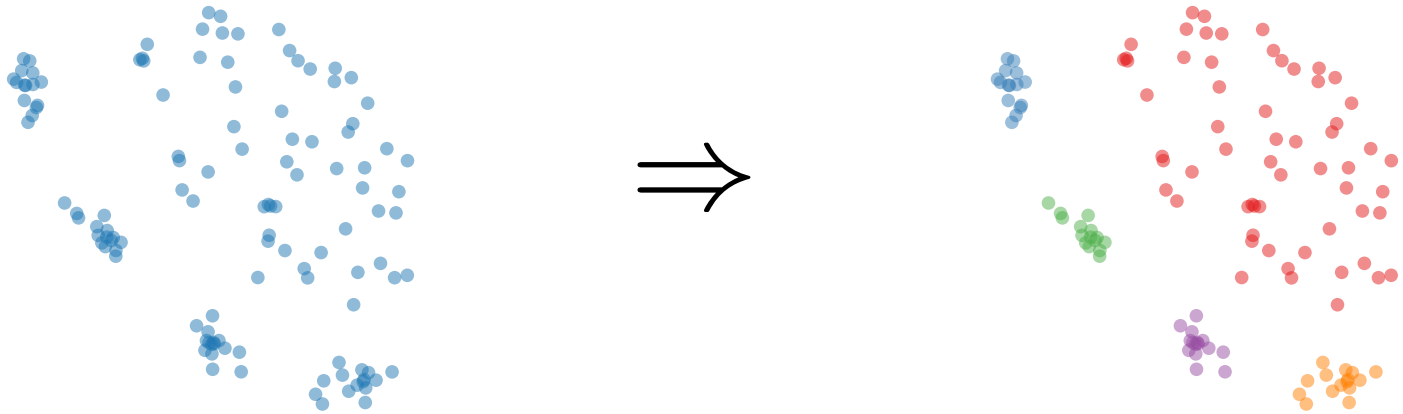
- Scikit-learn and its API became in recent years a standard for ML in Python.
- The estimator is usually used in 2 steps:
  1. Creation of the estimator :
     ```
     est = Estimator(param='parameter value',param2=10)
     ```
  2. Fitting of the estimator to the data:
     ```
     est.fit(X)
     ```
- After the fitting step, new attributes from the algorithms have been added to the object.

## Using the estimator in unsupervised learning

- **Clustering**
  Predict the clusters with `est.predict(X)` or `est.fit_predict(X)`
- **Probability density estimation**
  Compute the log-probability of samples with `est.score_samples(X)`.
- **Generative modeling**
  Generate new samples with `est.sample(n_samples)`.
- **Dimensionality reduction/ Dictionary learning**
  Transform the data (in low dimension) with `est.transform(X)`, sometimes an inverse transform is available with `est.inverse_transform(X)`.

# Section

# Clustering



## Objective

$$\{\mathbf{x}_i\}_{i=1}^n \quad \Rightarrow \quad \{\hat{y}_i\}_{i=1}^n$$

- Organize training examples in groups: Find the labels $\hat{y}_i \in \mathcal{Y} = \{1, \ldots, K\}$.
- Optional : find a clustering function $\hat{f}(\mathbf{x}) \in \mathcal{Y}$ that can cluster new samples.

## Parameters

- $K$ number of classes.
- Similarity measure between samples.
- Minimal distance between clusters.

## Methods

- K-means.
- Gaussian mixtures.
- Spectral clustering.
- Hierarchical clustering.

# Main clustering approaches

## Connectivity-based (Hierachical)

- Use pairwise relation between samples/cluster to agglomerate/divide clusters to create a hierarchical tree.
- The tree contains the whole clustering between $n$ to $1$ cluster and select with parameter (distance threshold or number of cluster $K$)

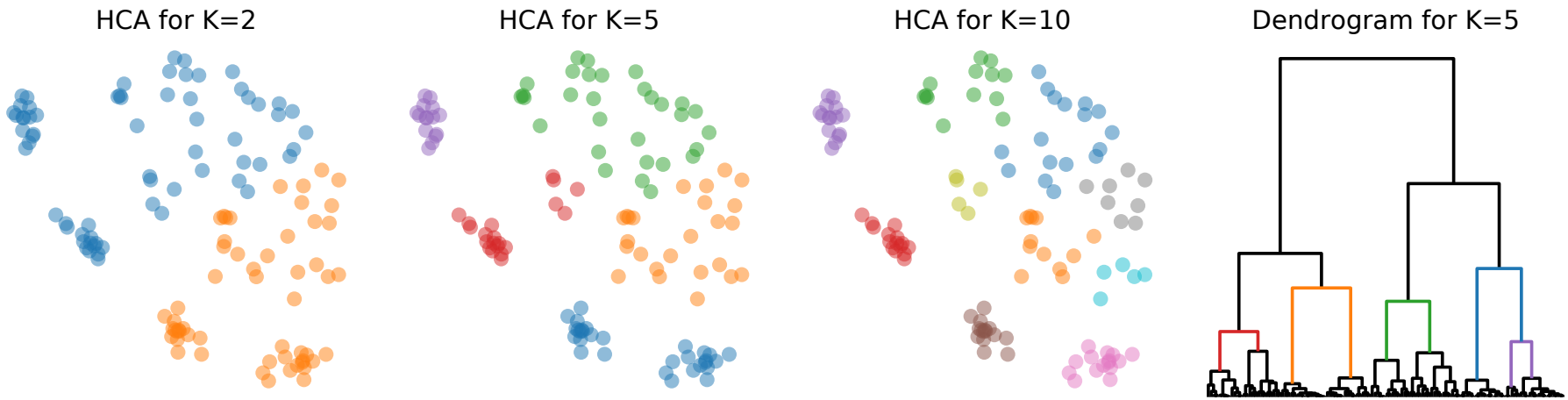
## Centroid Based (K-means)

- Express the dataset as a list of $K$ cluster centroids that represent the diversity of the data (each sample is associated to centroid).
- Minimize the average distance of all samples to their closest centroid (intra cluster variance).

## Density based (DBSCAN)

- Local density estimation for each sample using a neighborhood in a ball around the sample.
- Two samples belong to the same cluster if they are close enough and are in a high density area.

# Hierarchical Clustering Analysis (HCA)




## Principle (Tutorial [Nielsen, 2016])

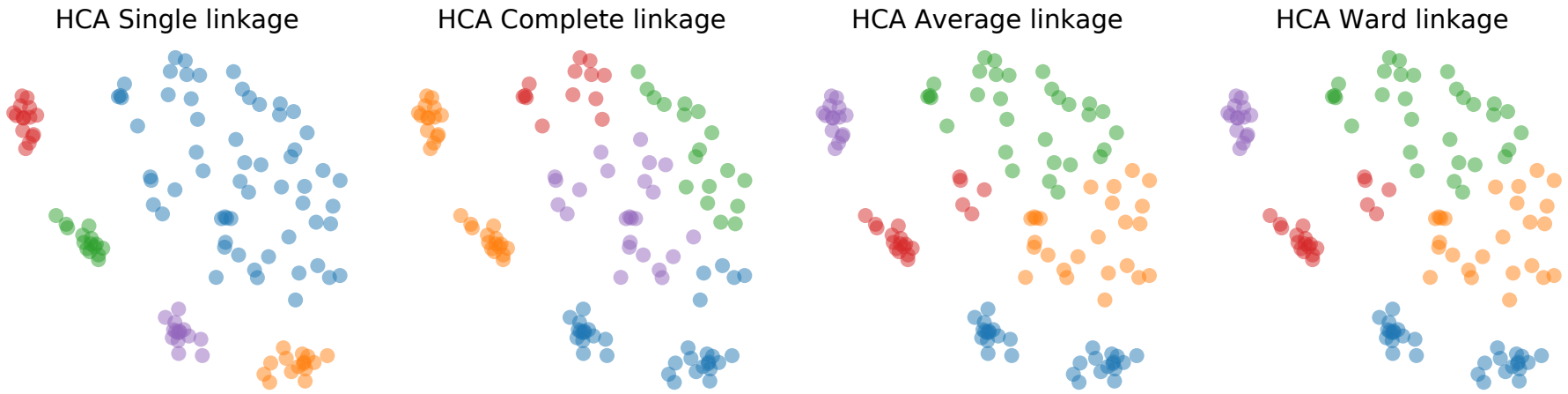
- HCA is an approach that find clusters recursively through **Agglomeration** (or sometime division).
- The **linkage function** $\Delta(\mathcal{C}_i, \mathcal{C}_j)$ is a measure of "distance" between two clusters.
- Final clustering with a fixed $K$ nb. of clusters or a threshold on $\Delta(\mathcal{C}_i, \mathcal{C}_j)$.
- The tree visualization of the agglomeration steps is called the dendrogram.

## Agglomerative HCA algorithm

1: Init. clusters $\{\mathcal{C}_i\}_i$ with $n$ Clusters $\mathcal{C}_i$ (one per sample).
2: **while** $|\{\mathcal{C}_i\}_i| > 1$ **do**
3: Find the pair $\mathcal{C}_i, \mathcal{C}_j$ minimizing $\Delta(\mathcal{C}_i, \mathcal{C}_j)$ among all pairs.
4: Merge $\mathcal{C}_i$ and $\mathcal{C}_j$.
5: **end while**

- Algorithm is $O(n^3)$ in general but $O(n^2)$ possible for single and complete linkage.

# HCA Linkage functions and implementation




## Most common Linkage functions

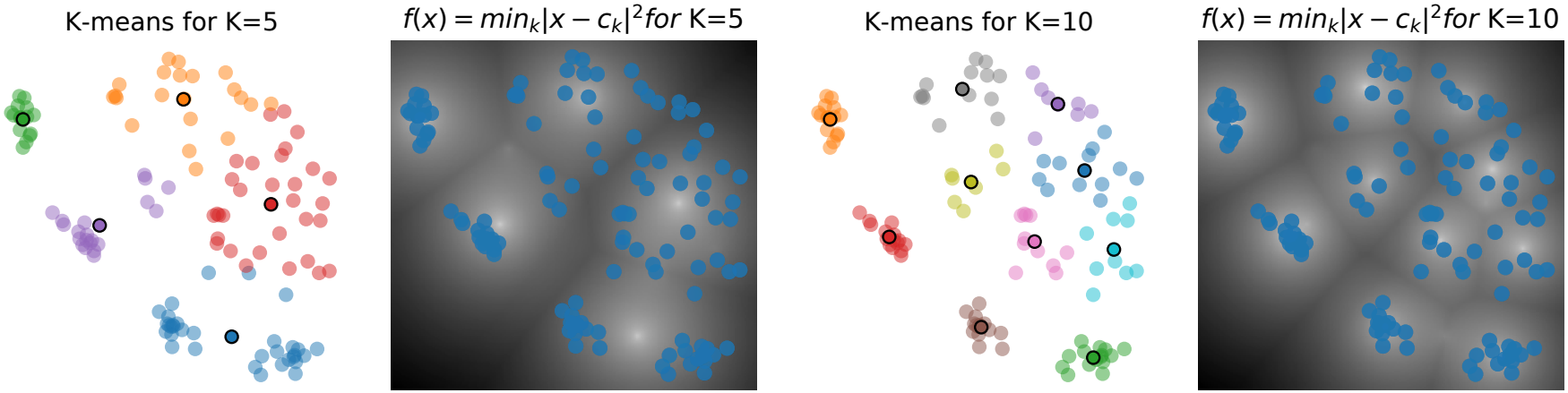
- **Single** [Sibson, 1973]
  $\Delta(\mathcal{C}_i, \mathcal{C}_j) = \min_{\mathbf{x}\in\mathcal{C}_i, \mathbf{x}'\in\mathcal{C}_j} d(\mathbf{x}, \mathbf{x}')$
- **Complete** [Defays, 1977]
  $\Delta(\mathcal{C}_i, \mathcal{C}_j) = \max_{\mathbf{x}\in\mathcal{C}_i, \mathbf{x}'\in\mathcal{C}_j} d(\mathbf{x}, \mathbf{x}')$
- **Average** [Sokal, 1958]
  $\Delta(\mathcal{C}_i, \mathcal{C}_j) = \frac{1}{|\mathcal{C}_i||\mathcal{C}_j|} \sum_{\mathbf{x}\in\mathcal{C}_i, \mathbf{x}'\in\mathcal{C}_j} d(\mathbf{x}, \mathbf{x}')$
- **Ward** [Ward Jr, 1963] ($\bar{\mathcal{C}} = \frac{1}{|\mathcal{C}|} \sum_{\mathbf{x}\in\mathcal{C}} \mathbf{x}$)
  $\Delta(\mathcal{C}_i, \mathcal{C}_j) = \frac{|\mathcal{C}_i||\mathcal{C}_j|}{|\mathcal{C}_i|+|\mathcal{C}_j|} \|\bar{\mathcal{C}}_i - \bar{\mathcal{C}}_j\|^2$

## Python code

```
1  from sklearn.cluster import
       AgglomerativeClustering
2
3  # HCA with K=2
4  clf = AgglomerativeClustering(
       n_clusters=2)
5  # fit the model
6  clf.fit(X)
7  # predict the cluster labels
8  yc = clf.predict(X)
9  # get the structure of the tree
10 children = clf.children_
11 # can be used for visualization with
12 # scipy.cluster.hierarchy.dendrogram
```

Also see `scipy.cluster.hierarchy`.

# K-means clustering




## Principle [Steinhaus et al., 1956, MacQueen et al., 1967]

Find $K$ clusters $\mathbf{c}_k \in \mathbb{R}^d$ that optimize:

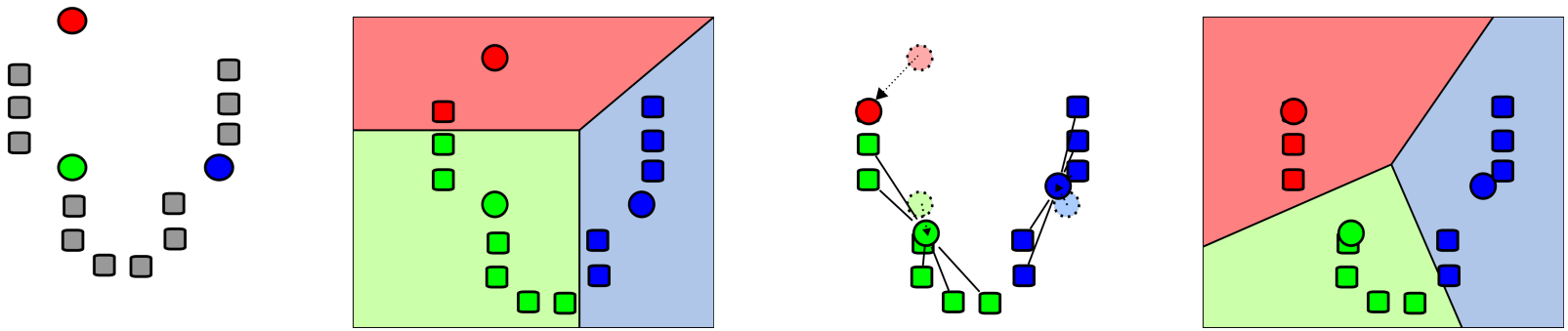
$$\min_{\mathbf{c}_k, \forall k} \quad \sum_{i=1}^{n} \min_k \left\| \mathbf{x}_i - \mathbf{c}_k \right\|^2 \tag{1}$$

- Minimize the sum of squared distance between $\mathbf{x}_i$ and its closest cluster $\mathbf{c}_k$.
- Can be seen as the minimization *w.r.t.* $\mathbf{c}_k$ of the expectation on the data of function $f(\mathbf{x}) = \min_k \left\| \mathbf{x}_i - \mathbf{c}_k \right\|^2$.
- The optimization problem can be reformulated with $\mathbf{A} \in \{0,1\}^{n \times K}$ a cluster assignment binary matrix ($A_{i,k} = 1$ means that $\mathbf{x}_i$ is in cluster $k$) as

$$\min_{\mathbf{c}_k \in \mathbb{R}^d, \forall k, \mathbf{A} \in \{0,1\}^{n \times K}, \mathbf{A}\mathbf{1}_K = \mathbf{1}_n} \quad \sum_{i=1,k=1}^{n,K} A_{i,k} \left\| \mathbf{x}_i - \mathbf{c}_k \right\|^2 \tag{2}$$

# Kmeans algorithm

Init. Step 1 Step 2 Step 3



## K-means Algorithm

1: Init. clusters $\{\mathbf{c}_k\}_k$.
2: **while** Not converged **do**
3: Update $\mathbf{A}$ by assigning each sample to its closest cluster.
4: Update $\mathbf{c}_k$ as the mean of the samples in the cluster.
5: **end while**

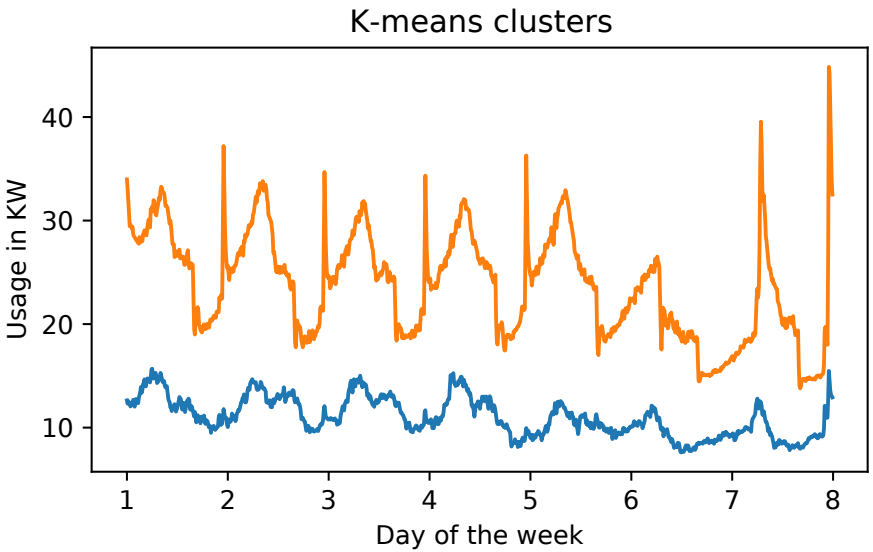
- This is a Block Coordinate Descent (BCD) algorithm on the problem 2.

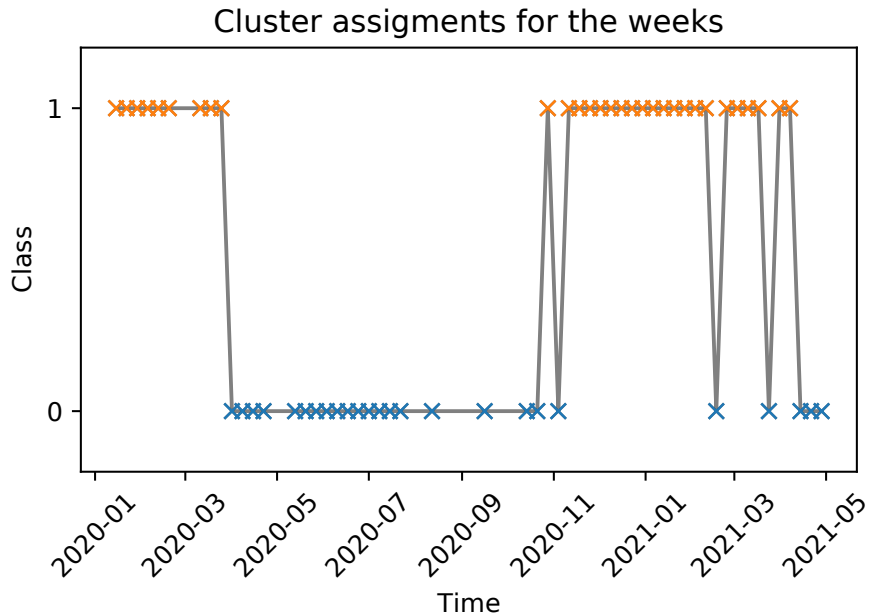
## Python code

```
1  from sklearn.cluster import KMeans
2
3  # K-means with K=2
4  clf = KMeans(2)
5
6  # fit the model et predict classes
7  y = clf.fit_predict(X)
8
9  # distance from samples to clusters
10 dist = clf.transform(X)
11
12 # get the centroids
13 C = clf.cluster_centers_
```

Images from Wikipedia

# K-means on energy usage dataset



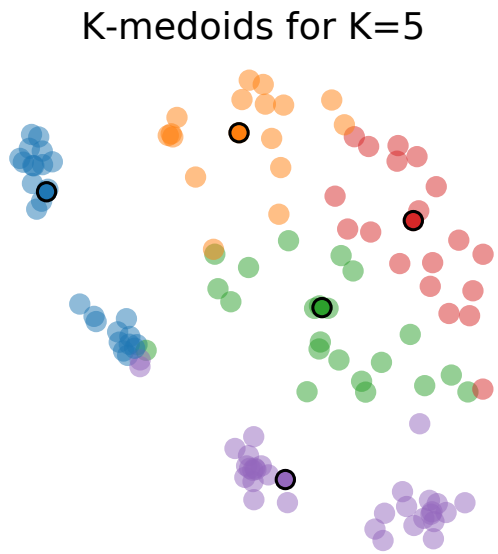

## Application

- Run K-means with $K = 2$ on the $n = 55$ samples of size $d = 1008$.
- Left : plot cluster centroids $\mathbf{c}_k$ as signals of week usage.
- Right : plot clusters assignments as a function of date of the monday of the week.
- Cluster 1 with more energy usage than cluster 0.
- Seasonal clustering along the year (1 for winter, 0 for summer).

# K-means variants

## K-medoids [Maranzana, 1963]

$$\min_{\mathbf{c}_k \in \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}, \forall k} \quad \sum_{i=1}^{n} \min_{k} \|\mathbf{x}_i - \mathbf{c}_k\|^2 \qquad (3)$$

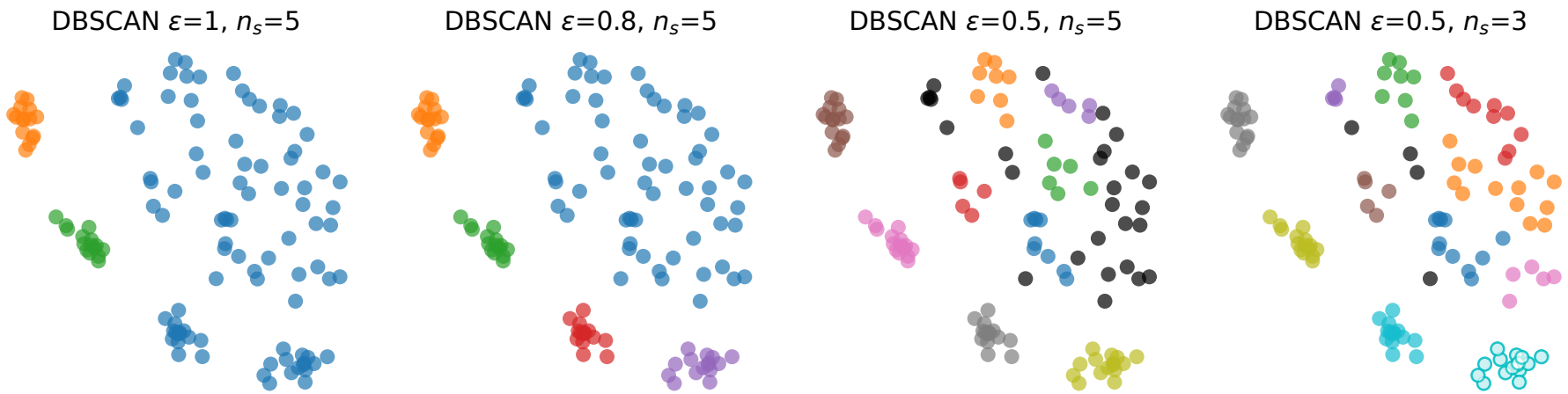



- Similar to K-means but the clusters have to be selected among the data points.
- Can be solved using BCD (as K-means) or the well known Partitioning Around Medoids (PAM) algorithm [Kaufman and Rousseeuw, 1990].

## K-means and extensions

- Initialization of the clusters is important. In Scikit-learn, K-means++ initialization is used by default [Arthur and Vassilvitskii, 2006].
- Large scale dataset K-means solver with Stochastic Gradient Descent [Bottou and Bengio, 1995] or Minibatch-Kmeans [Sculley, 2010] (sklearn.cluster.MiniBatchKMeans).
- K-median [Bradley et al., 1997] allows clustering robust to outlier by changing the norm (L1 instead of L2).

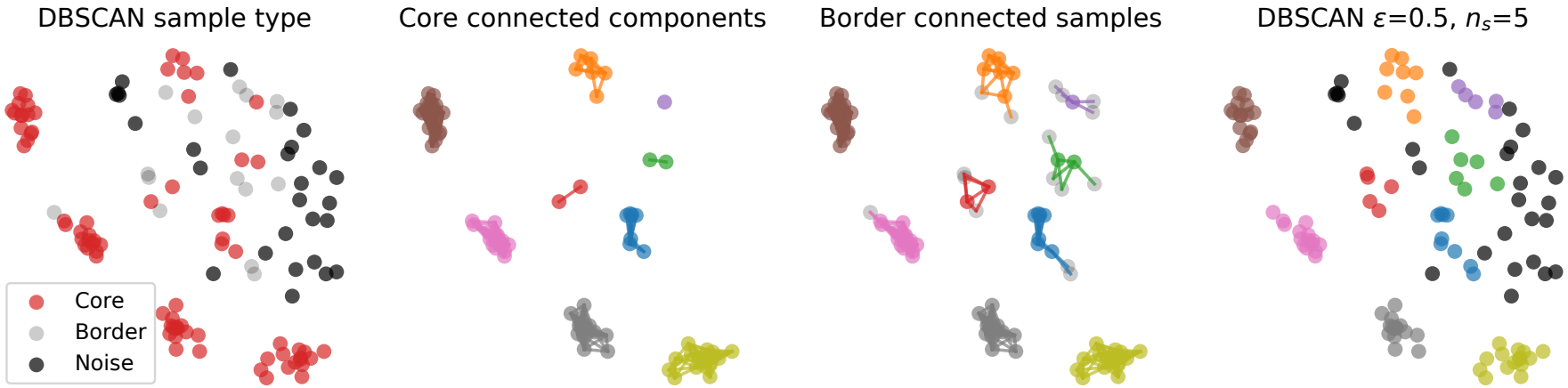
# DBSCAN




## Density-based spatial clustering of applications with noise (DBSCAN) [Ester et al., 1996, Schubert et al., 2017]

- Density estimation method that group into clusters samples that are in high density area and detect noise in low density area (black samples above).
- Local density around a sample is estimated using the number of neighbors in the $\epsilon$ ball $N_\epsilon(\mathbf{x}) = |\{\mathbf{x}_j | D(\mathbf{x}_i, \mathbf{x}) \leq \epsilon\}|$.
- Parameters are $\epsilon$ (size of the ball) around and $n_s$ minimum number of sample in neighborhood for detecting dense areas.
- Clustering uses different type of samples:
  - Core samples have high density : $N_\epsilon(\mathbf{x}) > n_s$.
  - Border (connected) samples : $N(\mathbf{x})_\epsilon \leq n_s$ but $\exists \mathbf{x}_c$ core sample s.t. $D(\mathbf{x}, \mathbf{x}_c) \leq \epsilon$.
  - Noise sample : $N(\mathbf{x})_\epsilon \leq n_s$ and $D(\mathbf{x}, \mathbf{x}_c) > \epsilon$, $\forall \mathbf{x}_c$ core samples.

# DBSCAN Algorithm



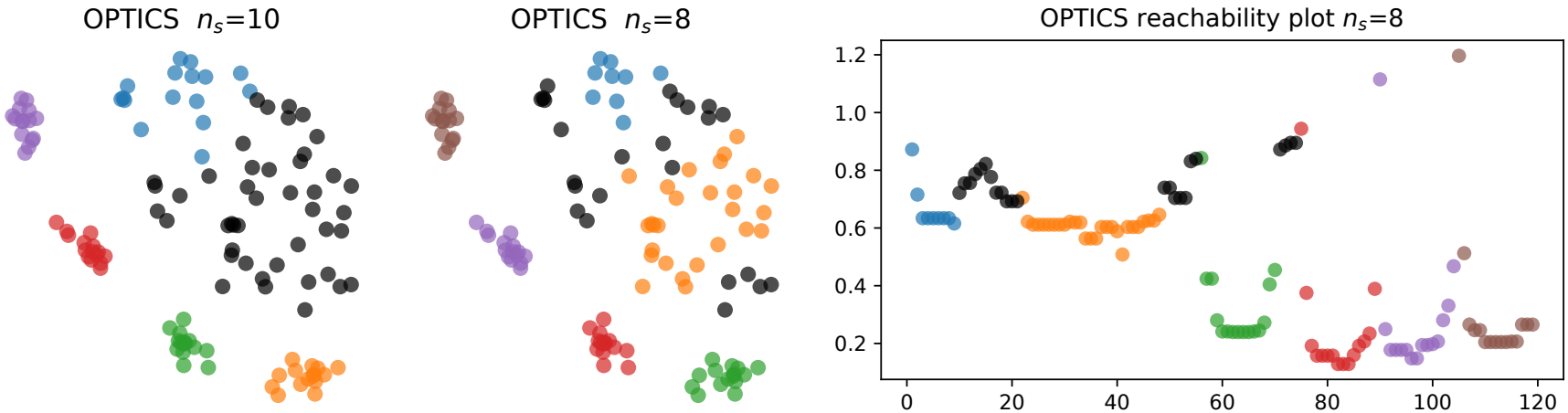

## Algorithm (simplified) [Ester et al., 1996]

1. Compute the neighborhood $N_\epsilon(\mathbf{x}_i)$ of all samples and find core samples.
2. Find the connected components of the core samples (ignore all other samples).
3. Go through the non-core sample and label them to a cluster if in the $\epsilon$ neighborhood of a sore sample or to noise if not.

- In practice DBSCAN goes through the dataset sample by sample. Clustering for border samples connected to more than 1 cluster depends on the order.
- DBSCAN is a celebrated methods[1], and is used a lot in practical applications.
- Scikit-learn estimator : `sklearn.cluster.DBSCAN(eps=0.5,min_samples=5)` .

[1] Test of time award ACM SIGKDD 2014

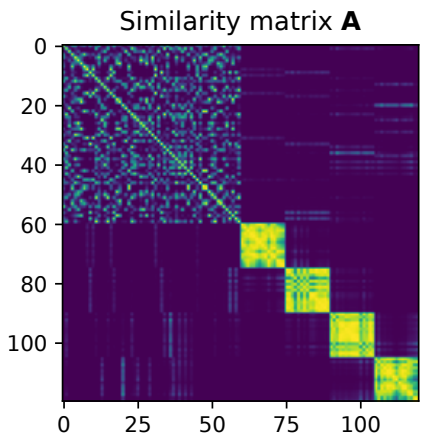
# OPTICS

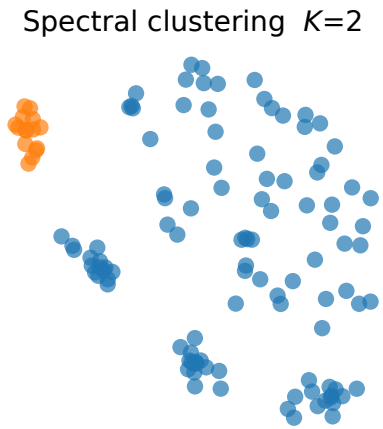


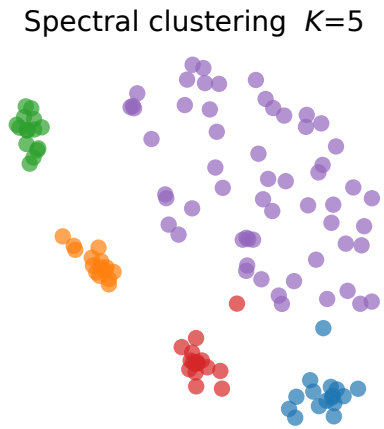

## Ordering points to identify the clustering structure (OPTICS) [Ankerst et al., 1999]

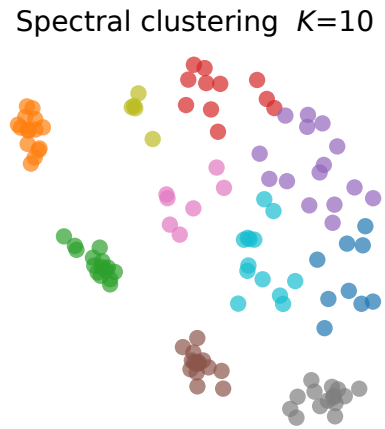
- Local density estimation similar to DBSCAN but done with ordering of the samples.
- Use the reachability of samples (distance to core samples) to order (and go through) samples in a reachability plot.
- Perform clustering from the reachability plot by searching for valleys or thresholding (similar to DBSCAN).
- Scikit-learn estimator : `sklearn.cluster.OPTICS(min_samples=5, max_eps=np.inf)` .

# Spectral clustering




## Principle (Tutorial [Von Luxburg, 2007], [Shi and Malik, 2000])

1. Represent pairwise relationship between samples with a similarity matrix $\mathbf{A}$ (kernel or binary) an compute its Laplacian or normalized Laplacian:

$$\mathbf{L} = \mathbf{D} - \mathbf{A}, \quad \text{or} \quad \mathbf{L}_n = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$$

   with $\mathbf{D} = \text{diag}(\mathbf{A}\mathbf{1}_n)$
2. Perform eigen-decomposition of this matrix and keep the $K$th largest eigenvectors.
3. Perform clustering (usually K-means) on the $n \times K$ matrix of eigenvectors.

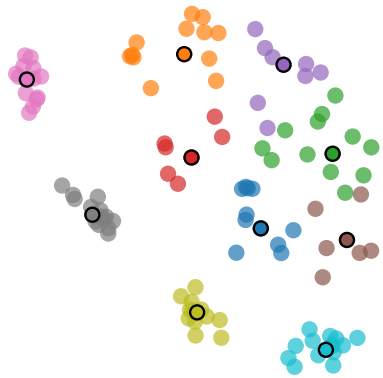
- Strongly related to nonlinear dimensionality reduction (DR + clustering).
- Allows for highly nonlinear separation between clusters.
- Scikit-learn estimator : `sklearn.cluster.SpectralClustering(n_clusters=5)` .

# Other Clustering approaches

## Affinity propagation [Frey and Dueck, 2007]

- Use message passing between samples to estimate a clustering based on selection of "examplars" (similar to K-medoids).
- Scikit-learn : `sklearn.cluster.AffinityPropagation()` .

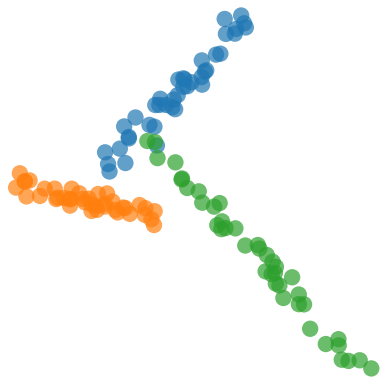
Affinity propagation



## Subspace clustering [Parsons et al., 2004]

- Clusters in high dimension are defined by affine subspaces.
- Estimate optimal subspaces for each clusters and assign labels *w.r.t.* the distance to the subspaces.

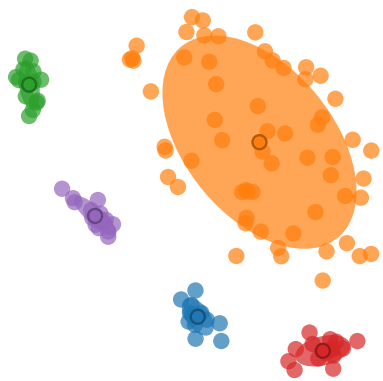
Subspace clustering



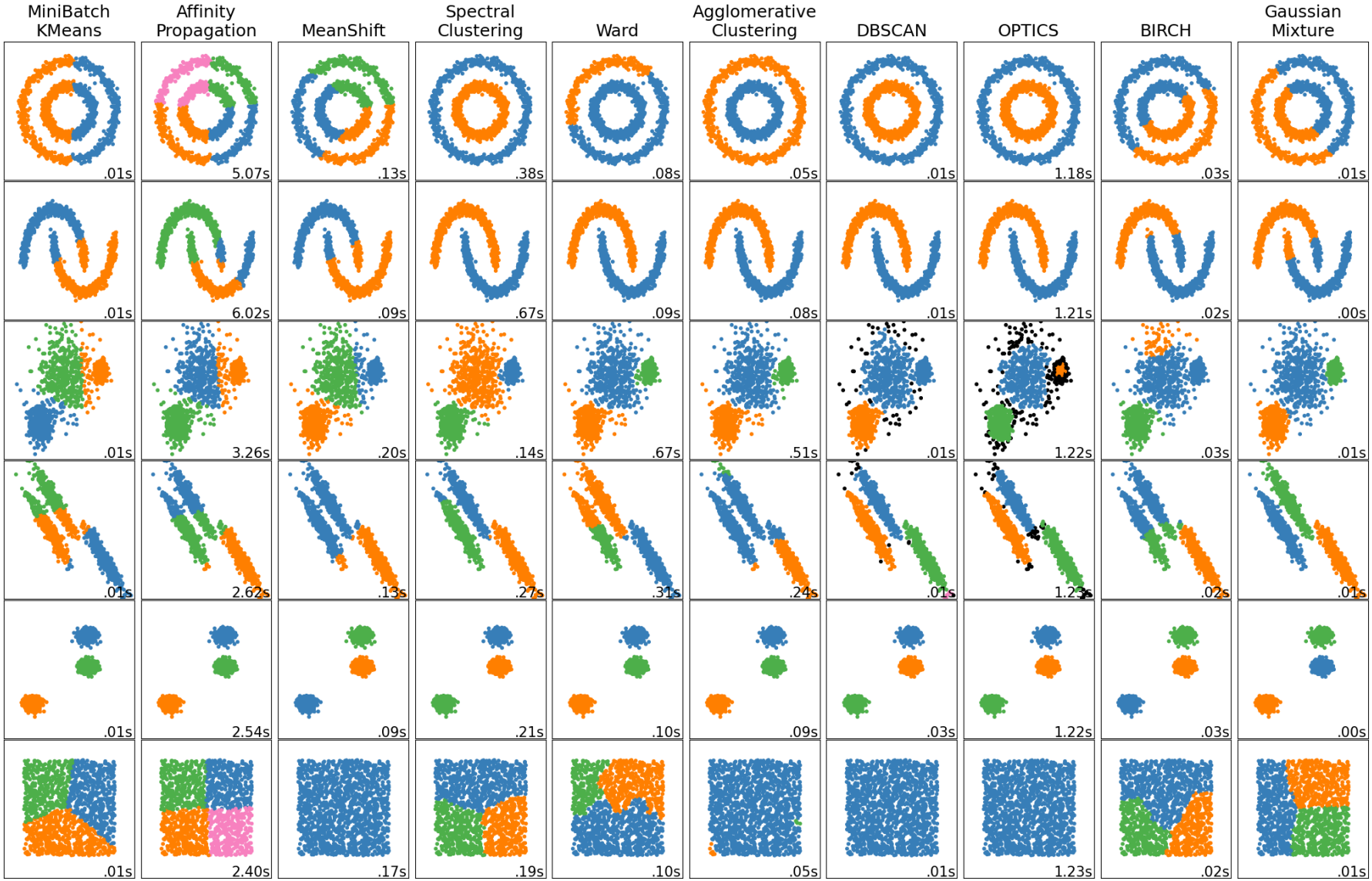
## Mixture models [McLachlan et al., 2019]

- Density estimation based on a mixture of distributions.
- Clustering done by computing the probability of each samples to be generated by one of the distribution in the mixture.
- See Gaussian Mixture Models (GMM) in the next part.

GMM clustering

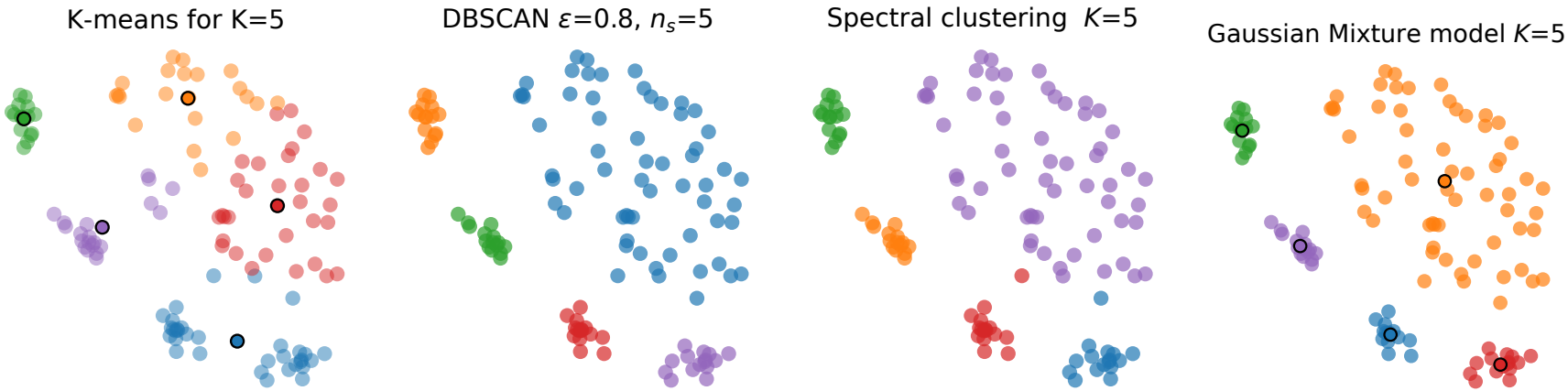


# Comparison of clustering methods




https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html

# Clustering, in practice

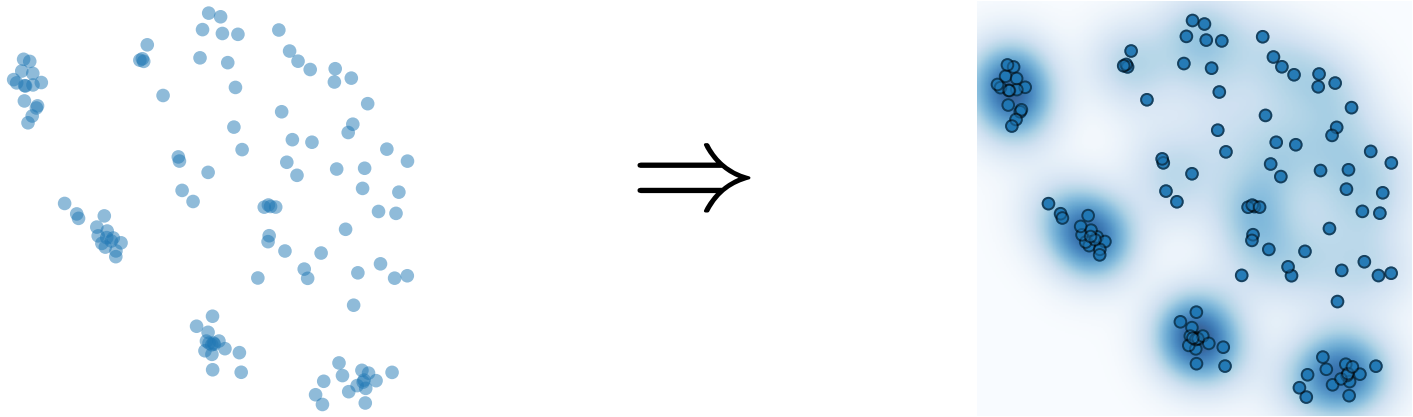



## Which method to use ?

- First step: know you data (expert knowledge or visualization).
- Standard approaches are K-means when the number of cluster is known and DBSCAN when unknown.
- K-means and GMM works well on data with "blobs" and can handle different densities in the clusters (also they have interpretable clusters).
- DBSCAN and OPTICS can handle non-linearly separated clusters and the presence of noise/outliers in the data.
- Subspace clustering can handle data in different subspaces and Spectral clustering in nonlinear manifolds.

# Section

# Probability density estimation



## Objective

$$\{\mathbf{x}_i\}_{i=1}^n \quad \Rightarrow \quad \hat{p}$$

- Estimate a probability density $\hat{p}(\mathbf{x})$ from the IID samples in the data.
- Probability density : $\hat{p}(\mathbf{x}) \geq 0, \ \forall \mathbf{x}$ and $\int \hat{p}(\mathbf{x}) d\mathbf{x} = 1$.
- Optional : generate new data with $\hat{p}(\mathbf{x})$, detect outliers in the data.

## Parameters

- Type of distribution (Histogram, Gaussian, . . . ).
- Parameters of the law $(\mu, \Sigma)$

## Methods

- Histogram (1D/2D).
- Parzen/kernel density estimation.
- Gaussian mixture.

# Maximum Likelihood Estimator (MLE)

## Principle

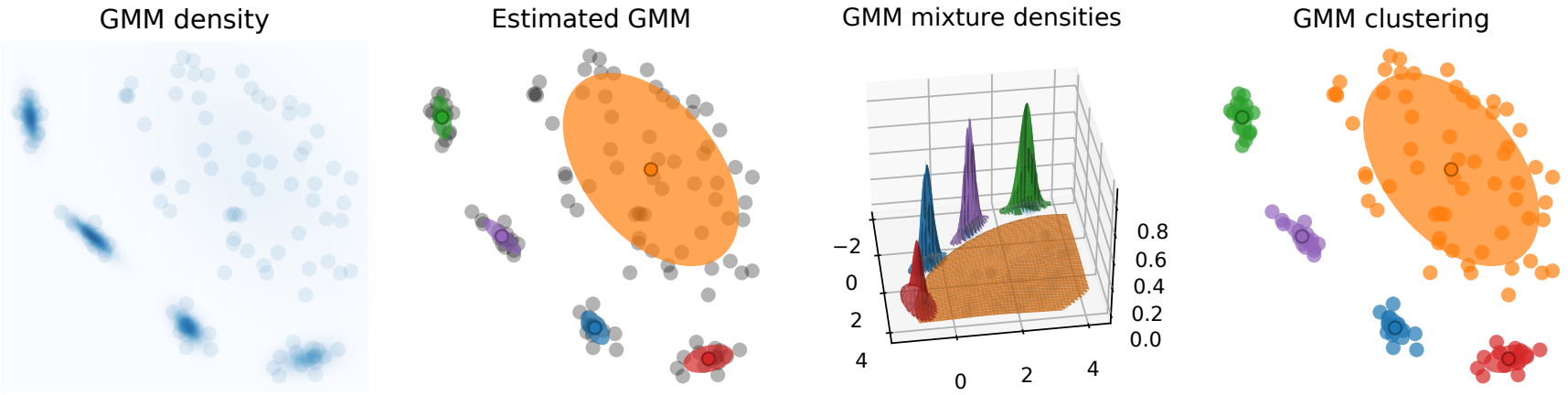
$$\max_{\boldsymbol{\theta}} \quad L(\boldsymbol{\theta}; \{\mathbf{x}_i\}_i)$$

- Let $p(\mathbf{x}|\boldsymbol{\theta})$ be a probability density distribution parametrized by $\boldsymbol{\theta}$.
- MLE consist in finding the optimal parameter $\boldsymbol{\theta}$ that maximizes the likelihood for a given empirical sample $\{\mathbf{x}_i\}_i$.
- For Independent and Identically Distributed (IID) samples the likelihood can be expressed as
$$L(\boldsymbol{\theta}; \{\mathbf{x}_i\}_i) = \prod_{i=1}^{n} p(\mathbf{x}_i|\boldsymbol{\theta})$$
- In practice the log-likelihood $l(\boldsymbol{\theta}; \{\mathbf{x}_i\}_i) = \log(L(\boldsymbol{\theta}; \{\mathbf{x}_i\}_i))$ that transforms the product as a sum is often optimized with the same solution.

## Example of MLE : Multivariate Gaussian (Normal) distribution

- The density is parametrized by $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ and can expressed as
$$p(\mathbf{x}|\boldsymbol{\theta}) = p_{\mathcal{N}}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \left((2\pi)^d |\boldsymbol{\Sigma}|\right)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})\right)$$
- The MLE estimated on the samples $\{\mathbf{x}_i\}_i$ is
$$\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i, \qquad \hat{\boldsymbol{\Sigma}} = \frac{1}{n}\sum_{i=1}^{n} (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^{\top}$$

# Gaussian Mixture Models (GMM)



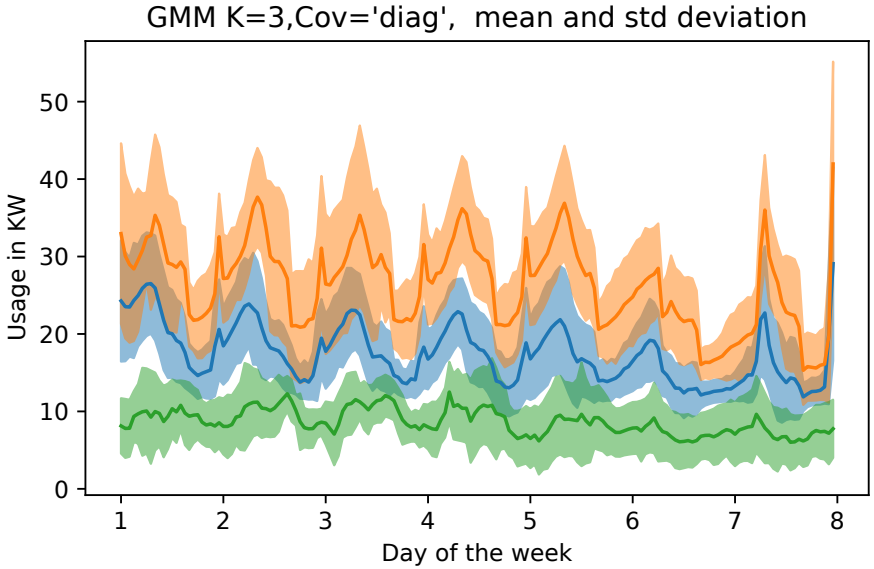

## Principle [Dempster et al., 1977, Yu et al., 2011]

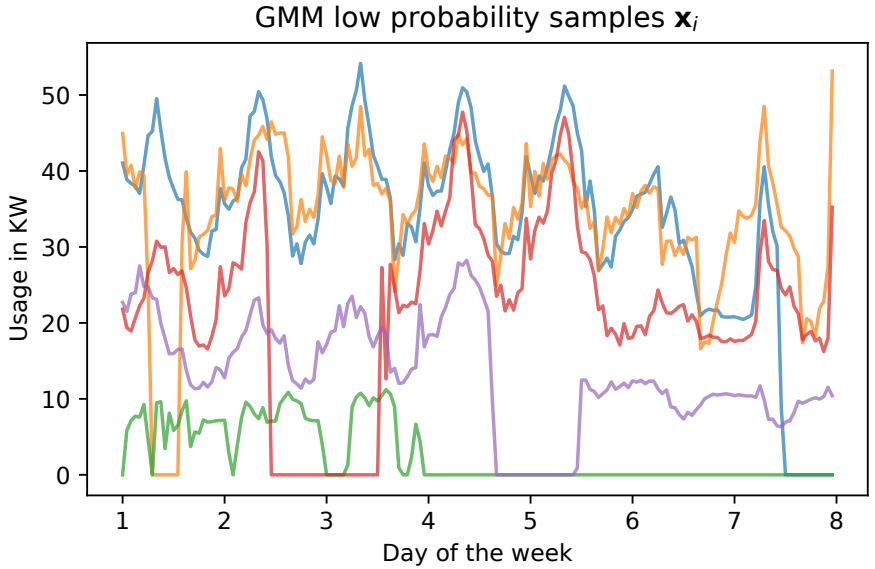
- Model the prob. distribution of the data as a sum of $K$ Gaussian distributions :

$$p_{GMM}(\mathbf{x}|\boldsymbol{\theta}) = \sum_{k=1}^{K} \phi_k p_{\mathcal{N}}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{4}$$

- Estimate $\boldsymbol{\theta} = \{\boldsymbol{\phi}, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \forall k\}$ by maximizing the likelihood on the data.
- Optimization performed using the Expectation Maximization that consists in maximizing at each iteration a lower bound of the likelihood.
- The algorithm updates iteratively the probability that each component $k$ generated each sample $\mathbf{x}_i$ and the parameters $\boldsymbol{\theta}$.
- Covariances can be full, diagonal, or low rank [Houdard et al., 2018].
- Scikit-learn implementation : sklearn.mixture.GaussianMixture

# GMM on energy usage data




## Application

- GMM with $K = 3$ on week energy usage data with diagonal covariances (because data in high dimension).
- Plot left shows the mean and standard deviation of each component in the mixture.
- Each component correspond to a low/medium/high energy consumption.
- Plot right the 5 samples with lowest probability score to detect outliers in the dataset (week usage with missing data in this case).

```
from sklearn.mixture import
      GaussianMixture
# create and fit the model
clf = GaussianMixture(3)
clf.fit(X)

# predict cluster class
yc = clf.predict(X)
# compute proba of samples
p = np.exp(clf.score_samples(x))
# generate new samples
Xg = clf.sample(100)
# Get estimated parameters
phi = clf.weights_
mus = clf.means_
Sigmas = clf.covariances_
```

# Maximum A Posteriori estimator (MAP)

## Principle

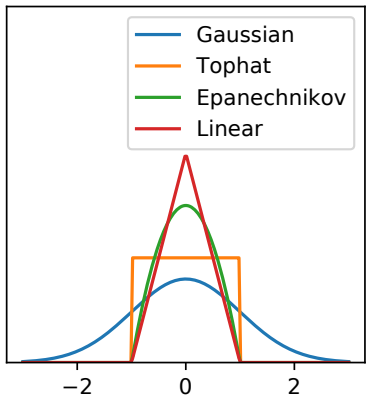
$$\max_{\boldsymbol{\theta}} \quad q(\boldsymbol{\theta}) L(\boldsymbol{\theta}; \{\mathbf{x}_i\}_i)$$

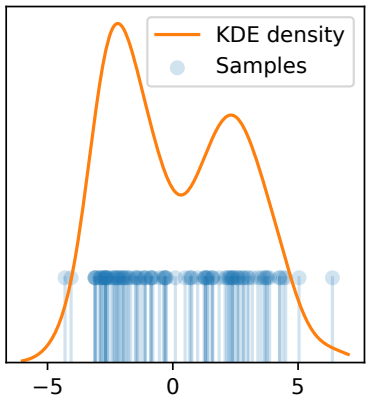
- $q(\boldsymbol{\theta})$ defines a prior distribution about the distribution parameter in addition to the likelihood $L(\boldsymbol{\theta}; \{\mathbf{x}_i\}_i)$ on the data.
- When the prior $q$ is non informative (uniform), we recover the MLE, for simple priors (gaussian) we recover regularized estimators.
- The optimization problem can be solved with numerical optimization : EM algorithm, variational inference or Monte Carlo method.

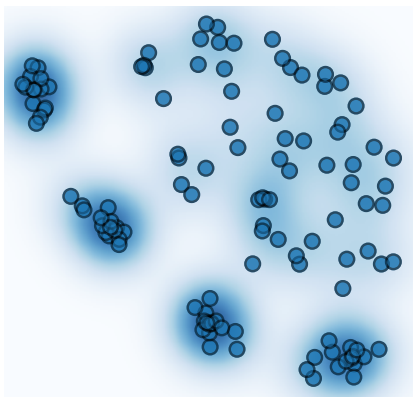
## Example: Variational Gaussian Mixture Models [Blei and Jordan, 2006]

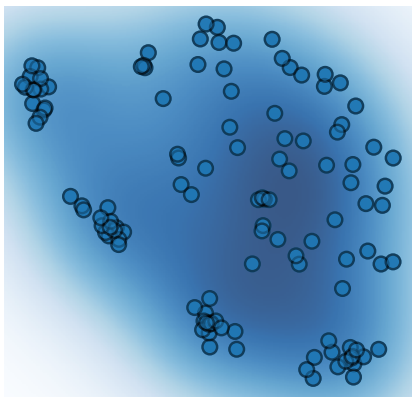
- Principle : Use a sparsity promoting prior on the weight $\phi$ of the components that can be of infinite size (Dirichlet process).
- The final number of components is controlled by the `weight_concentration_prior` parameter (less components for small values).
- In practice it allows to find automatically the number of components $K$.
- Scikit-learn implementation : BayesianGaussianMixture()

# Kernel Density Estimation (KDE)



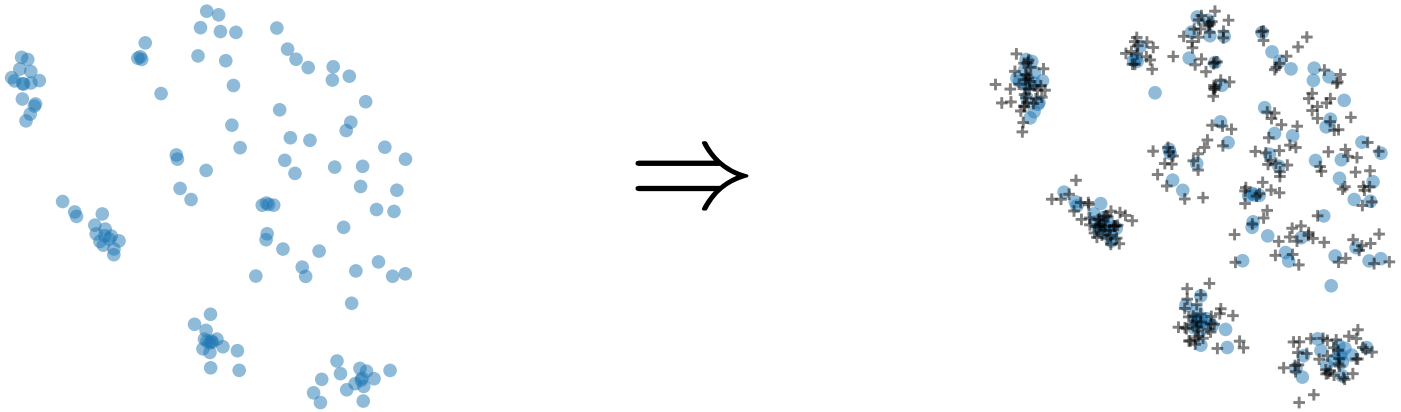




## Principle [Rosenblatt, 1956, Parzen, 1962]

$$\hat{p}_h(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} k_h(\mathbf{x}, \mathbf{x}_i)$$

- $\{\mathbf{x}_i\}_{i=1,\dots,n}$ are supposed to be IID and the kernel function $k_h(\mathbf{x}, \mathbf{x}_i)$ is positive and of the form $k_h(\mathbf{x}, \mathbf{x}_i) = \tilde{k}(\frac{\mathbf{x}-\mathbf{x}_i}{h})$ where $h > 0$ is a bandwidth parameter.
- Can be seen as a convolution between the empirical distribution $\frac{1}{n}\sum_i \delta_{\mathbf{x}_i}$ and the centered kernel $k(\mathbf{x}, \mathbf{0})$, *i.e.* a low pass smoothing of the distribution.
- Common kernels (positive, symmetric and normalized $\int k(\mathbf{x}, \mathbf{0})d\mathbf{x} = 1$) are :
  - Gaussian kernel : $\tilde{k}(\mathbf{x}, \mathbf{x}') = \left((2\pi)^d d\right)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\|\mathbf{x} - \mathbf{x}'\|^2\right)$.
  - [Epanechnikov, 1969] : $\tilde{k}(x, x') = \frac{3}{4}\max(1 - |x - x'|^2, 0)$ in 1D.
  - Tothat (Rectangular) and Linear (Triangular) kernels.
- Scikit-learn implementation : sklearn.neighbors.KernelDensity

# Generative modeling



## Objective

$$\{\mathbf{x}_i\}_{i=1}^n \quad \Rightarrow \quad \hat{g} \text{ such that } p(\hat{g}(\mathbf{z})) \approx p(\mathbf{x}) \text{ with } \mathbf{z} \sim \mathcal{N}$$

- Estimate a mapping function $\hat{g}(\mathbf{z}) \in \mathbb{R}^d$ that generates similar samples to $\{\mathbf{x}_i\}_{i=1}^n$.
- Latent variable $\mathbf{z}$ follows a known Normal or Uniform distribution.
- Optional : recover an estimation of $\hat{p}(\mathbf{x})$ using the change of variable formula.

## Parameters

- Type of distribution for $\mathbf{z}$ (Gaussian, uniform, ...).
- Type of function for $g$.

## Methods

- PCA (Gaussian data), KDE, GMM.
- Gen. Adversarial Networks (GAN)
- Variational Auto-Encoders (VAE)

# Generative modeling by divergence minimization

## Generator function

- $g : \mathbb{R}^p \to \mathbb{R}^d$ is a continuous function and $\mu_z$ a distribution on $\mathbb{R}^p$.
- $g$ can be used to generate samples in $\mathbb{R}^d$ from samples $\mathbf{z} \sim \mu_z$ in $\mathbb{R}^p$.
- Notation : $g\#\mu_z$ is the distribution of the random variable $g(\mathbf{z})$ with $\mathbf{z} \sim \mu_z$.

## Minimizing the divergence betwen distributions

$$\min_g \quad D(\mu_d, g\#\mu_z)$$

- Learn a generator $g$ that minimize the divergence $D$ between the generated data with samples $\mathbf{z} \sim \mu_z$ and the empirical data distribution $\mu_d = \frac{1}{n}\sum_i \delta_{\mathbf{x}_i}$.
- Different divergences that can be used:
  - Jensen-Shannon (JS) : Classical GAN [Goodfellow et al., 2014].
  - Wasserstein (Optimal Transport) [Arjovsky et al., 2017]
  - Maximum mean Discrepancy (MMD) [Li et al., 2015, Dziugaite et al., 2015].
  - $f$-divergences [Nowozin et al., 2016].
- Problem above can often be reformulated as a minimax between two functions hence the name adversarial.
- Not provided by Scikit-learn, see implementations in Pytorch or tensorflow.

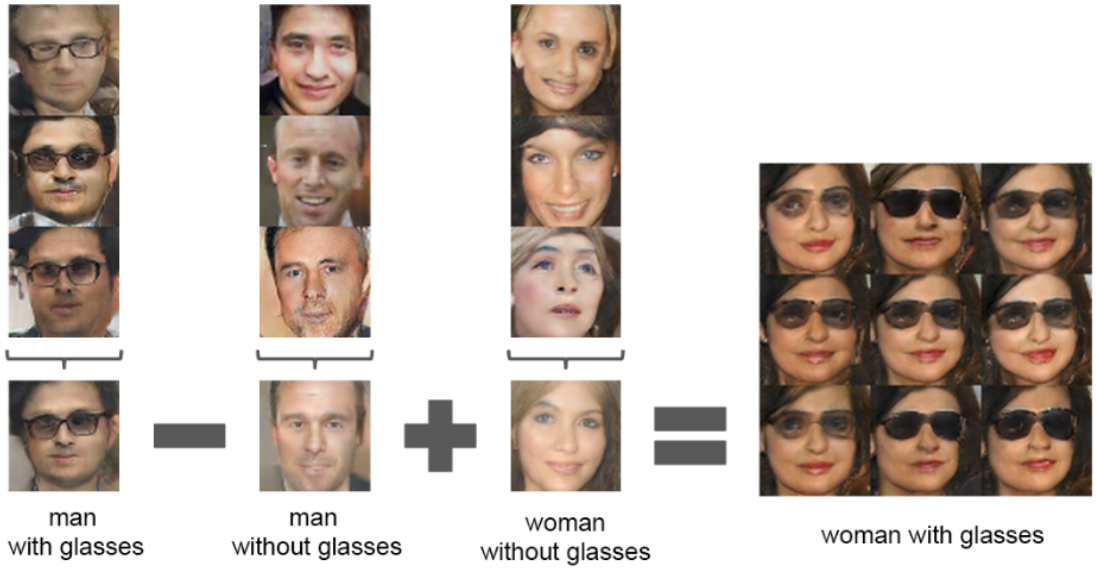
# Generative adversarial learning (GAN)

## Generative Adversarial Networks (GAN) [Goodfellow et al., 2014]

$$\min_g \max_h \quad E_{\mathbf{x}\sim\mu_d}[\log h(\mathbf{x})] + E_{\mathbf{z}\sim\mathcal{N}(0,\mathbf{I})}[\log(1 - h(g(\mathbf{z})))]$$

- $h$ is a classifier trying to discriminate real data and data simulated by $g$.
- Data generated with $g$ from IID random samples ($g(\mathbf{z})$ with $\mathbf{z} \sim N(0, \sigma^2)$).
- Both the generator $g$ and classifier $h$ compete (are adversaries).
- Generator space has semantic meaning [Radford et al., 2015].

# Generative adversarial learning (GAN)




## Generative Adversarial Networks (GAN) [Goodfellow et al., 2014]

$$\min_g \max_h \quad E_{\mathbf{x}\sim\mu_d}[\log h(\mathbf{x})] + E_{\mathbf{z}\sim\mathcal{N}(0,\mathbf{I})}[\log(1 - h(g(\mathbf{z})))]$$

- $h$ is a classifier trying to discriminate real data and data simulated by $g$.
- Data generated with $g$ from IID random samples ($g(\mathbf{z})$ with $\mathbf{z} \sim N(0, \sigma^2)$).
- Both the generator $g$ and classifier $h$ compete (are adversaries).
- Generator space has semantic meaning [Radford et al., 2015].

# Normalizing Flows

## Change of variable

- Let $g : \mathbb{R}^d \to \mathbb{R}^d$ be an invertible (bijective) function and $f = g^{-1}$ with $f(g(\mathbf{z})) = \mathbf{z}$ and $\mu_z = \mathcal{N}$ is the normal distribution.
- The **change of variable** formula gives us the density of $g\#\mu_z$, *i.e.* of $g(\mathbf{z})$ when $\mathbf{z} \sim \mu_z$ as a function of the density $p_z(\mathbf{z})$ of $\mu_z$:

$$p_x(\mathbf{x}) = p_z(f(\mathbf{x}))|\text{det}(Df(\mathbf{x}))|$$

where $Df(\mathbf{x}) = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$ is the Jacobian of the function $f$.

## Principle of normalizing flows (Tutorial [Kobyzev et al., 2020])

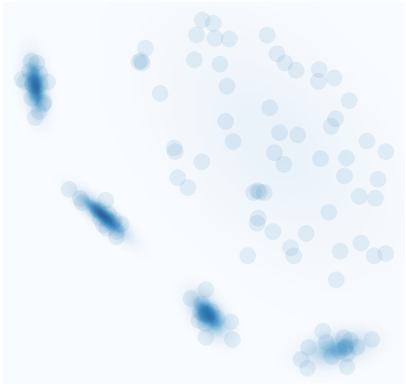
- $g$ is called the generator function and $f = g^{-1}$ is the normalizing function.
- Density estimation and generator estimation can be done by maximum the log-likelihood on the IID dataset $\{\mathbf{x}_i\}_i$:

$$\max_f \quad \sum_{i=1}^{n} \log(p_z(f(\mathbf{x}_i))) + \log|\text{det}(Df(\mathbf{x}_i)|)$$

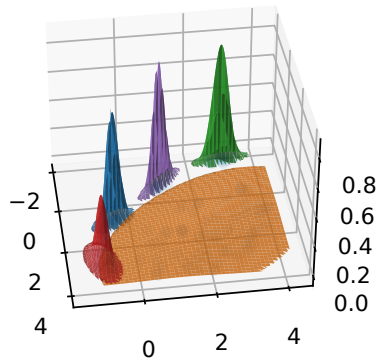
- The functions $g$ have to be easy to apply, invert, and compute the determinant of its jacobian, they are formulated as neural networks:
  - Linear flows [Tomczak and Welling, 2017].
  - Planar or radial flows [Berg et al., 2018], [Rezende and Mohamed, 2015]
  - Coupling or autoregressive flows [Dinh et al., 2016, Kingma et al., 2016].

# Probabiliy Density Estimation and generative modeling

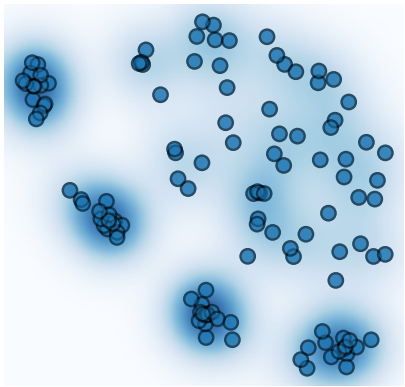
GMM density



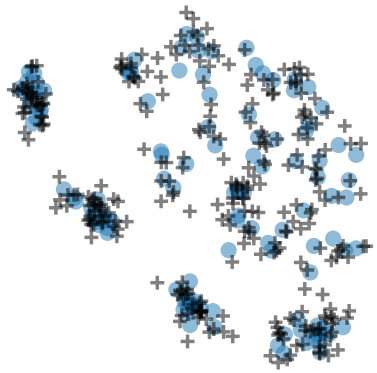
GMM mixture densities




KDE density $h$=0.4



KDE generated

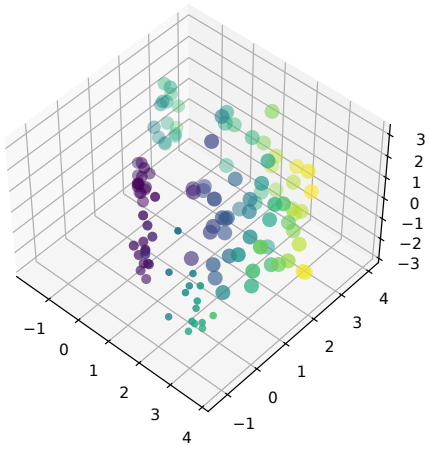


## Why and when?

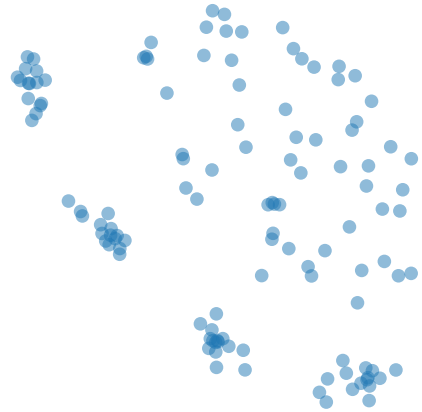
- Density estimation is hard (non-convex, large number of parameters).
- But it's the most informative modeling of unsupervised data.
- Density can be used for data generation, interpretation, outlier detection.
- When density is not necessary, generative modeling can be easier to estimate.
- Generative modeling can be used for other ML tasks (regularization for instance) but usually requires deep learning, harder to interpret.

# Section

# Dimensionality reduction




$\Rightarrow$



## Objective

$$\{\mathbf{x}_i\}_{i=1}^n \quad \Rightarrow \quad \{\tilde{\mathbf{x}}_i \in \mathbb{R}^p\}_{i=1}^n \text{ with } p \ll d$$

- Project the data into a low dimensional space of size $p \ll d$.
- Preserve the information in the data (class, subspace, manifold).
- Optional : Learning a projection function $\hat{m} : \mathbb{R}^d \to \mathbb{R}^p$ for new data.

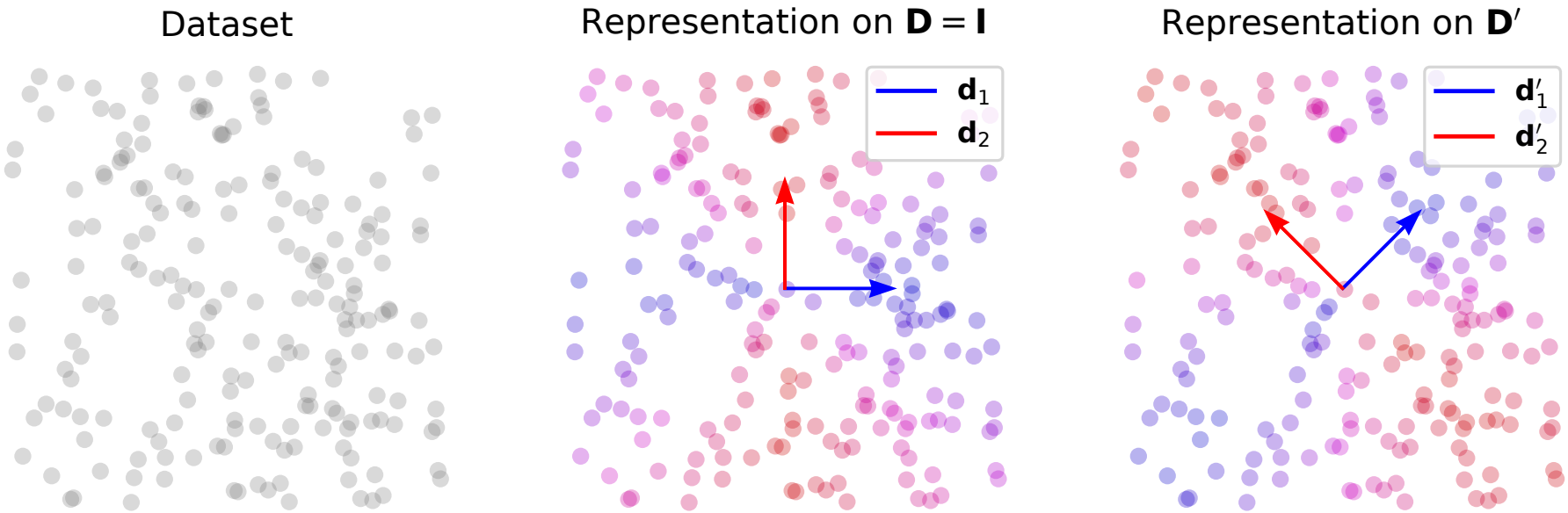
## Parameters

- Type of projection (linear, nonlinear).
- Assumptions about the data (subspace, manifold).
- Similarity between samples.

## Methods

- Feature selection.
- Principal Component Analysis (PCA).
- Dictionary learning, ICA.
- Non-linear dimensionality reduction (MDS, tSNE, Auto-Encoder)

# Linear model for the data




## Linear model

We suppose that $\mathbf{x} \in \mathbb{R}^d$ can be represented as a weighted sum of basis vectors:

$$\mathbf{x} \approx \mathbf{D}\mathbf{a} = \sum_{j=1}^{p} a_j \mathbf{d}_j \tag{5}$$

- $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_p] \in \mathbb{R}^{d \times p}$ is the dictionary and the $\mathbf{d}_k$ are the basis vectors.
- $\mathbf{a} \in \mathbb{R}^p$ is the representation of the sample $\mathbf{x}$ on the dictionary $\mathbf{D}$.
- When $p < d$ the equality might not stand depending on $\mathbf{D}$ and the samples can be approximated in a smaller dimensionality.

# Linear unmixing and dictionary learning

## Linear unmixing

$$\hat{\mathbf{a}} = \arg\min_{\mathbf{a}} \quad L(\mathbf{x}, \mathbf{D}\mathbf{a}) \tag{6}$$

- $L$ is a measure of divergence (usually quadratic $L(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|^2$).
- Linear unmixing is a projection onto the linear subspace defined by $\mathbf{D}$.
- $\hat{\mathbf{a}}$ is the representation of sample $\mathbf{x}$ on dictionary $\mathbf{D}$ and the samples can be reconstructed by $\hat{\mathbf{x}} = \mathbf{D}\hat{\mathbf{a}}$.
- When $\mathbf{D}$ is orthonormal ($\mathbf{D}^\top \mathbf{D} = \mathbf{I}_p$), the solution of the problem with quadratic divergence is $\hat{a} = \mathbf{D}^\top \mathbf{x}$.

## Dictionary Learning (DL)

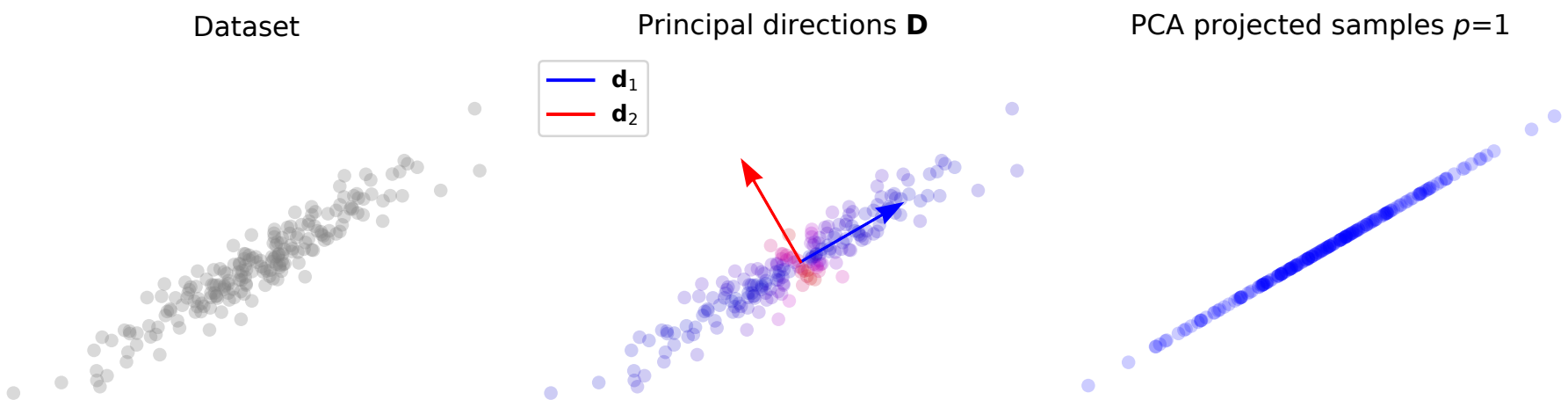
$$\hat{\mathbf{D}}, \hat{\mathbf{A}} = \arg\min_{\mathbf{D}, \mathbf{A}} \quad \sum_{i=1}^{n} L(\mathbf{x}_i, \mathbf{D}\mathbf{a}_i) \tag{7}$$

- Estimate simultaneously a dictionary $\hat{\mathbf{D}}$ and the representations $\hat{\mathbf{A}} = [\mathbf{a}_1, \ldots, \mathbf{a}_n]^T \in \mathbb{R}^{n \times p}$ on the dataset.
- DL is often called matrix factorization because the objective is to model the dataset $\mathbf{X}$ as a factorization:

$$\mathbf{X} \approx \hat{\mathbf{A}}\hat{\mathbf{D}}^\top$$

- Most linear dimensionality reduction methods also add constraints on $\mathbf{D}$ or $\mathbf{A}$

# Principal Component Analysis (PCA)




## Principle [Pearson, 1901]

$$\min_{\mathbf{D},\mathbf{D}^\top\mathbf{D}=\mathbf{I}_p} \sum_{i=1}^{n} \|\mathbf{x}_i^c - \mathbf{D}\mathbf{D}^\top\mathbf{x}_i^c\|^2 \quad \equiv \quad \max_{\mathbf{D},\mathbf{D}^\top\mathbf{D}=\mathbf{I}_p} \sum_{i=1}^{n} \|\mathbf{D}^\top\mathbf{x}_i^c\|^2 \tag{8}$$

- Find a linear subspace of dimensionality $p$ defined by $\mathbf{D}$ that minimize the reconstruction error of the centered data $\mathbf{X}^c$ ($0$ means in the columns of $\mathbf{X}^c$).
- Equivalent to maximizing the variance of the projected samples $\hat{\mathbf{a}}_i = \mathbf{D}^\top\mathbf{x}_i^c$.
- DL problem (7) with orthonormality constraints on $\mathbf{D}$ and $\hat{\mathbf{a}}_i = \mathbf{D}^\top\mathbf{x}_i^c$.
- Principal directions are the columns $\mathbf{d}_k$ or $\mathbf{D}$.
- Scikit-learn implementation : sklearn.decomposition.PCA.

# Principal Component Analysis in practice

## PCA Algorithm

1. Center the data and compute the covariance $\hat{\mathbf{\Sigma}} = \frac{1}{n}\mathbf{X}^{c\top}\mathbf{X}^{c} = \frac{1}{n}\sum_i \mathbf{x}_i^c(\mathbf{x}_i^c)^\top$.
2. Perform eigendecomposition $\{\mathbf{v}_j, \lambda_j\}$ of the covariance matrix $\hat{\mathbf{\Sigma}}$ and sort the eigenvalues by decreasing order.
3. The optimal dictionary (projection matrix) is:

$$\hat{\mathbf{D}} = [\mathbf{v}_1, \ldots, \mathbf{v}_p]$$

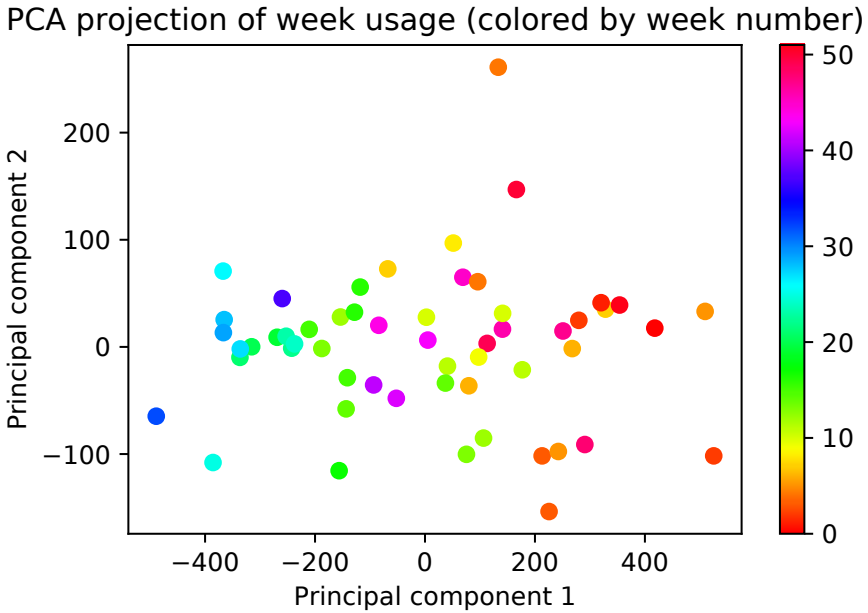
where $\{\mathbf{v}_1, \ldots, \mathbf{v}_p\}$ are the eigenvectors associated to the $p$ largest eigenvalues.

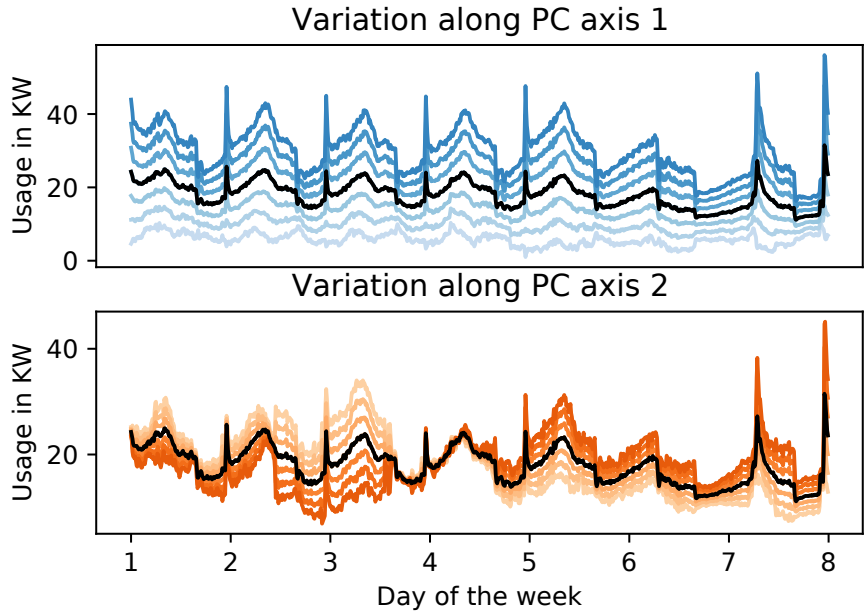
One can also use the equivalent Singular Value Decomposition of $\mathbf{X}^c$ (Scikit-learn).

## PCA in practice

- PCA can be used for denoising data : additive random IID noise located in low variance subspaces.
- Selection of $p$ can be used by plotting the sorted eigenvalues (searching for an elbow or ratio of explained variance) of with probabilistic modeling [Tipping and Bishop, 1999, Minka, 2000]
- Sparse PCA promotes sparsity on $\mathbf{d}_k$ for feature selection [Zou et al., 2006] .
- Warning : PCA focuses on correlation, *i.e.* linear relationship between features and can miss more complex relationships.

# PCA on energy usage data




## Application

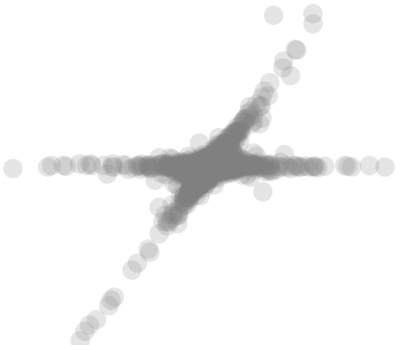
- Run PCA with $p = 2$.
- Plot plot projection in 2D colored by week number for interpretability.
- Summer/Winter dynamic along axis 1.
- Part of the week usage variation along axis 2.

## Python Code

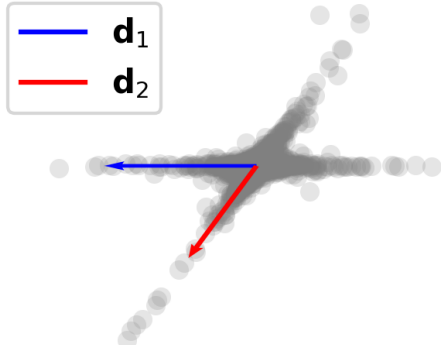
```
from sklearn.decomposition import PCA

# PCA with p=2
clf = PCA(2)
# fit the model and project in 2D
Xp = clf.fit_transform(X)
# Get the projections/axis P
D = clf.components_.T
```

# Independent Component Analysis

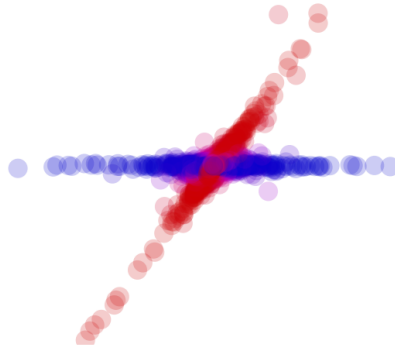
Dataset in 2D



ICA Dictionary



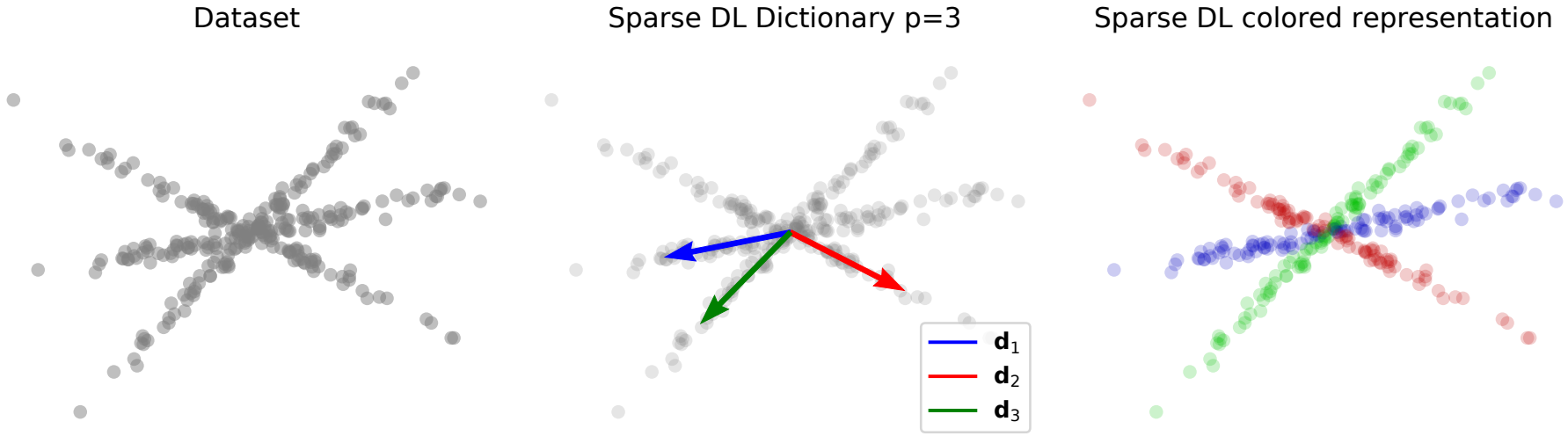

ICA Colored representation



## Principle [Herault and Jutten, 1986]

- Find a decomposition of the samples $\hat{\mathbf{a}} = \mathbf{D}^\top \mathbf{x}$ that is independent (columns of $\mathbf{A}$ are independent, not necessarily orthogonal as in PCA).
- Linear model but not expressed as the general optimization problem (7).
- Works particularly well on non Gaussian data (or else PCA is optimal).
- Efficient algorithm : FastICA [Hyvärinen and Oja, 2000].
- Applied with success to several source separation problems (biomedical data).
- Scikit-learn implementation : sklearn.decomposition.FastICA.

# Sparse Dictionary Learning



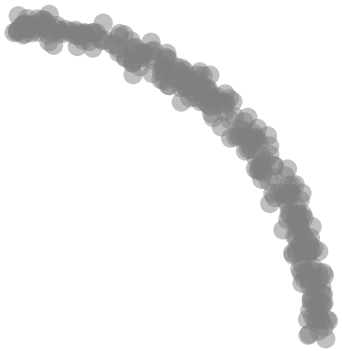

## Principle

$$\min_{\mathbf{A}\in\mathbb{R}^{n\times p},\mathbf{D}\in\mathbb{R}^{d\times p},\|\mathbf{d}_k\|=1,\forall k} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{D}\mathbf{a}_i\|^2 + \lambda\|\mathbf{a}_i\|_1 \tag{9}$$

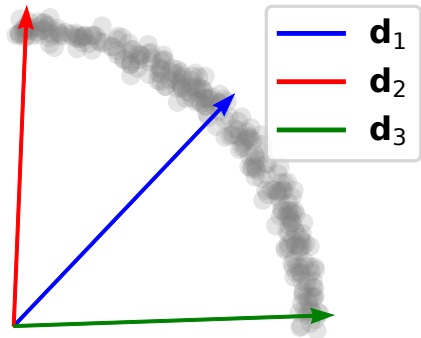
- Constraints on the norm of $\mathbf{d}_i$ ensure normalized basis (not orthogonal).
- Sparsity promoting $L_1$ regularization (see Lasso in next course) on the representations $\mathbf{a}_i$ promotes samples in linear subspaces of the span of $\mathbf{D}$.
- Similar to Sparse PCA but sparsity on $\mathbf{a}_i$ instead of the dictionary $\mathbf{d}_k$.
- Can be solved efficiently with stochastic optimization [Mairal et al., 2009].
- Scikit-learn implementation : sklearn.decomposition.DictionaryLearning.

# Non-negative Matrix Factorization (NMF)

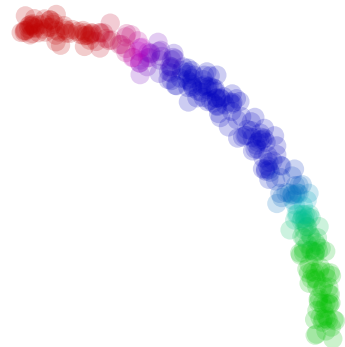
Dataset



NMF Dictionary p=3




NMF colored representation

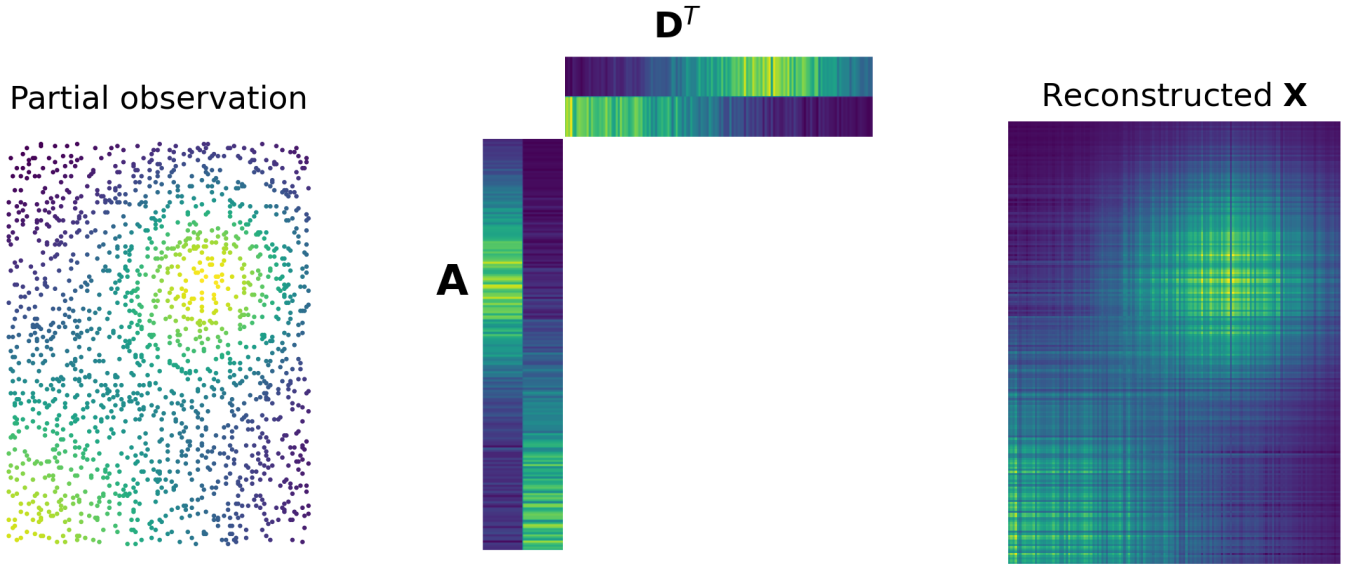


## Principle [Lee and Seung, 2000]

$$\min_{\mathbf{A}\in\mathbb{R}_+^{n\times p},\mathbf{D}\in\mathbb{R}_+^{d\times p},\|\mathbf{d}_k\|=1,\forall k} \sum_{i=1}^{n} L(\mathbf{x}_i, \mathbf{D}\mathbf{a}_i) \tag{10}$$

- For positive data (for instance power densities) it makes sens to have both dictionary elements $\mathbf{d}_j$ and representations $\mathbf{a}_j$ positive.
- Different losses $L$ have been proposed:
  - Quadratic, Guassian noise [Lee and Seung, 2000].
  - Kullback–Leibler divergence, Poisson noise [Dhillon and Sra, 2005].
  - Itakura-Saito, audio spectrum [Févotte et al., 2009]).
- Optimization problem can be solved with gradient descent, block coordinate descent and multiplicative updates.
- Sparsity regularization can also be used similarly to SparseDL.
- Scikit-learn implementation : sklearn.decomposition.NMF.

# Matrix factorization (collaborative filtering)

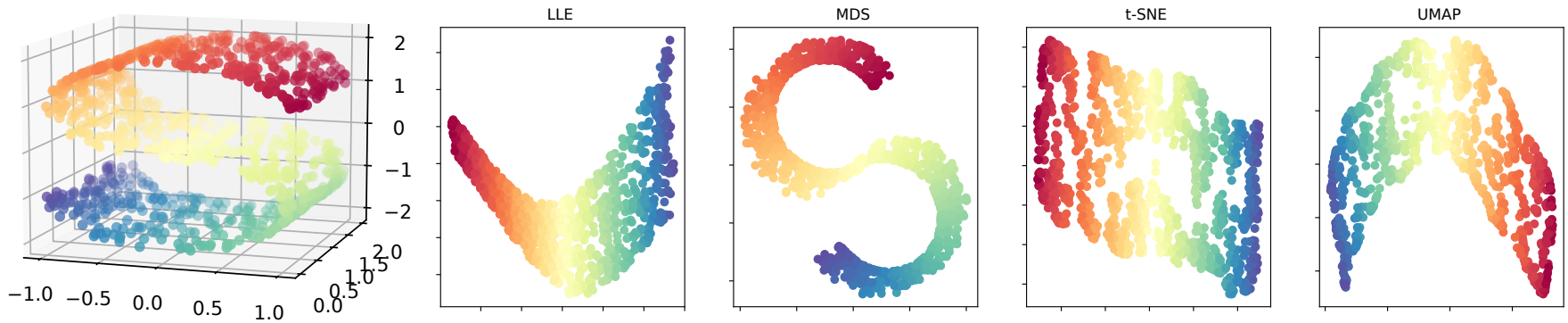



## Principle (Survey [Bokde et al., 2015])

$$\min_{\mathbf{D},\mathbf{A}} \quad \sum_{i=1}^{n} \|\mathbf{m}_i \odot (\mathbf{x}_i - \mathbf{D}\mathbf{a}_i)\|^2 \tag{11}$$

- $\odot$ is the pointwise multiplication and $\mathbf{m}_i \in \{0,1\}^d$ is a binary mask denoting which features in $\mathbf{x}_i$ that are observed for sample $\mathbf{x}_i$.
- Data is only partially observed but one wants to predict the values for all components of the matrix $\mathbf{X}$ (observed values are stored in a sparse matrix).
- Solved using truncated Singular Vector Decomposition that return a low rank $p < \min(d, n)$ factorization $\mathbf{X} \approx \mathbf{A}\mathbf{D}^T$.
- Used in recommender systems for user/product recommendation.

# Nonlinear dimension reduction methods (manifold learning)




## Nonlinear subspaces

- The dataset often lies in a nonlinear subspace (a manifold) of $\mathbb{R}^d$.
- Manifold learning method aim at recovering this low dimensional manifold.
- Example above of 2D manifold in a 3D ambient space and the projection of the samples in 2D for different methods (colors only to check that the relation between samples are preserved).

## Manifold learning problems

- **Projection** $\{\mathbf{x}_i\}_{i=1}^n \Rightarrow \{\tilde{\mathbf{x}}_i \in \mathbb{R}^p\}_{i=1}^n$ with $p \ll d$ : Project dataset in low dimension (visualization).
- **Inductive** $\{\mathbf{x}_i\}_{i=1}^n \Rightarrow g : \mathbb{R}^d \to \mathbb{R}^p$ : learn a nonlinear projection function.
- **Inductive+Invertible** $\{\mathbf{x}_i\}_{i=1}^n \Rightarrow g : \mathbb{R}^d \to \mathbb{R}^p, f : \mathbb{R}^p \to \mathbb{R}^d, f(g(\mathbf{x})) \approx \mathbf{x}$ : learn both projection and reconstruction nonlinear functions.

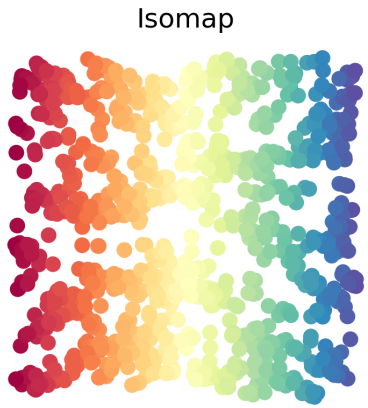
# Common manifold learning methods

## Multi-Dimensional Scaling (MDS) [Kruskal, 1964]

- Search for positions $\{\tilde{\mathbf{x}}_i\}$ that have a similar pairwise distance matrix as the original data $\{\mathbf{x}_i\}$.
- Solved with eigendecomposition (PCA on distances).
- Scikit-learn : sklearn.manifold.MDS.

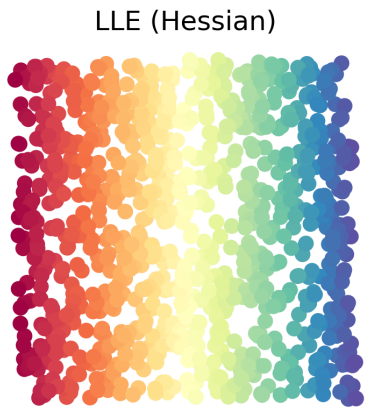

## ISOMAP [Tenenbaum et al., 2000]

- Estimate a graph of neighbors in the ambient space.
- Use the geodesic distance on the graph between samples and perform MDS (preserve distance on the manifold).
- Scikit-learn : sklearn.manifold.Isomap.

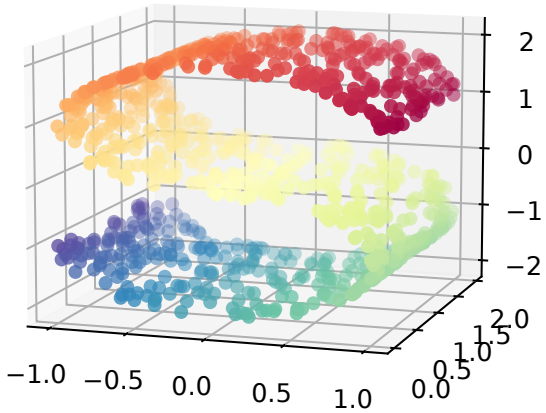


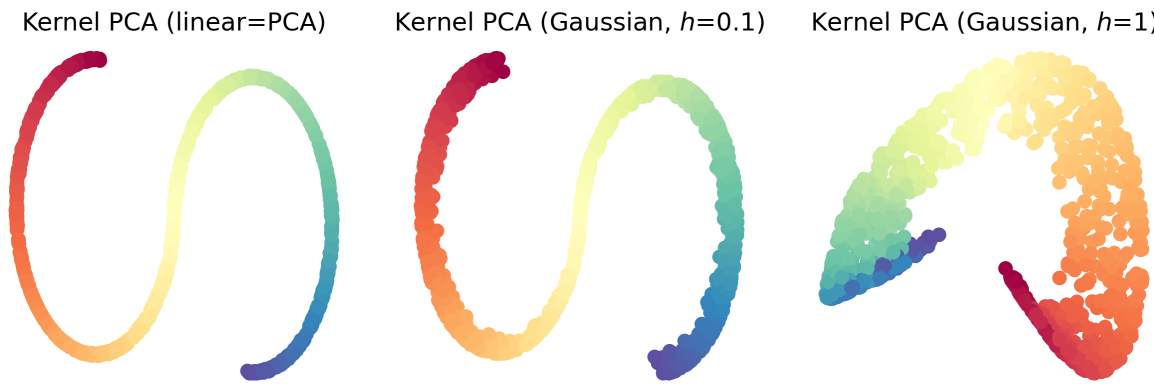

## Locally Linear Embedding (LLE) [Roweis and Saul, 2000]

- Find an embedding that preserve distance in local neighborhood (many PCA).
- Regularized LLE : modified [Zhang and Wang, 2007], Hessian [Donoho and Grimes, 2003] .
- Scikit-learn : sklearn.manifold.LocallyLinearEmbedding.



# Kernel PCA (KPCA)

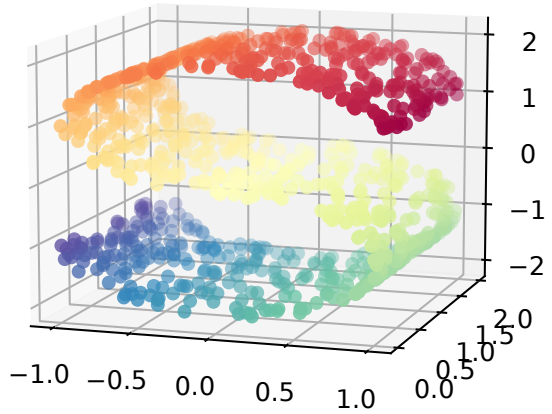


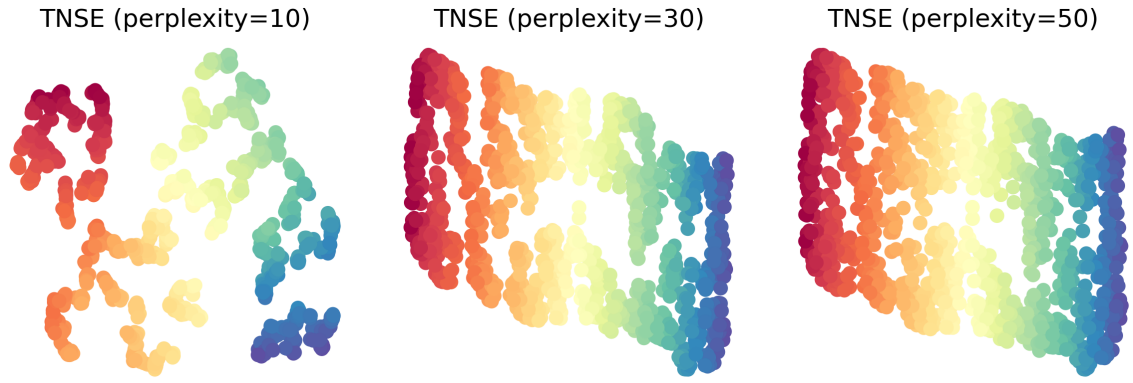




## Principle [Schölkopf et al., 1997]

- Perform PCA in a high-dimensional non-linear embedding $\phi(\mathbf{x})$ of the data.
- Embedding is implicit, thanks to the use of a kernel $k(\mathbf{x}, \mathbf{x}') =< \phi(\mathbf{x}), \phi(\mathbf{x}') >$, only the kernel matrix between samples is necessary. This is called the "kernel trick"(used also for SVM classification).
- Inductive method, reconstruction is possible but requires solving an inverse problem (kernel pre-image problem [Kwok and Tsang, 2004]).
- Classical kernels are linear kernel (equivalent to PCA) and Gaussian kernel (Radial Basis Function RBF in Sckikit-learn).
- LLE and ISOMAP are actually special cases of KPCA with specifically designed kernels [Ham et al., 2004]
- Scikit-learn implementation : sklearn.decomposition.KernelPCA.

# t-Stochastic Neighbor Embedding (TSNE)




## Principle [Van der Maaten and Hinton, 2008]

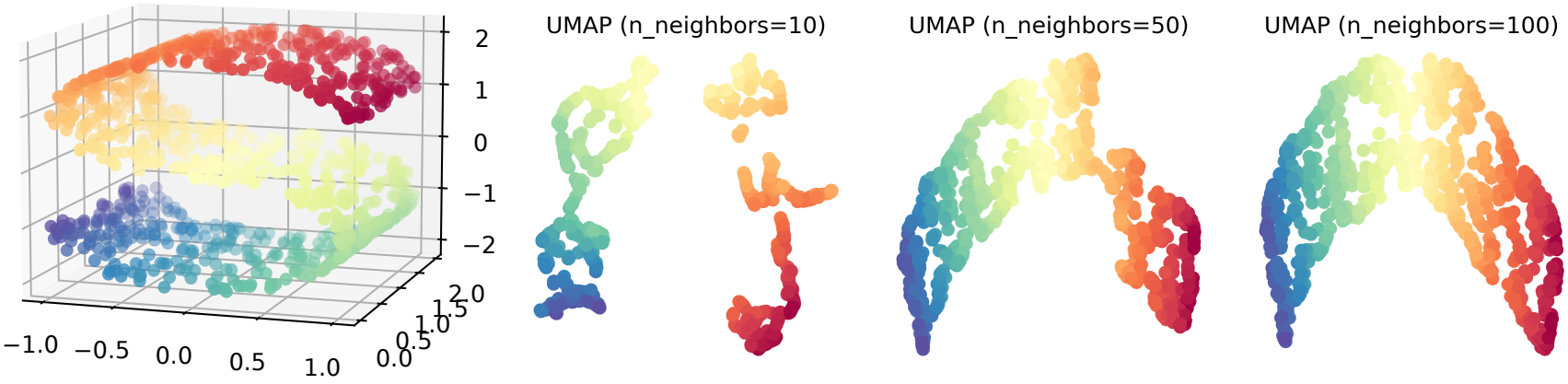
$$\min_{\tilde{\mathbf{x}}_i, \forall i} \sum_{i,j} KL(p_x(\mathbf{x}_i|\mathbf{x}_j)||q_{\tilde{x}}(\tilde{\mathbf{x}}_i|\tilde{\mathbf{x}}_j))$$

- KL is the Kullback–Leibler divergence and the distributions $p$ and $q$ are the probability that two samples are neighbors expressed as:

$$p_x(\mathbf{x}_i|\mathbf{x}_j) = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2)}, \quad q_{\tilde{x}}(\tilde{\mathbf{x}}_i|\tilde{\mathbf{x}}_j) = \frac{(1+\|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\|^2)^{-1}}{\sum_k \sum_{l\neq k}(1+\|\tilde{\mathbf{x}}_k - \tilde{\mathbf{x}}_l\|^2)^{-1}}$$

- The bandwith $\sigma_i$ are set to provide a given perplexity parameter (accuracy of density estimation on training data).
- t-SNE uses a t-Student distribution on the projected samples instead of the Gaussian kernel in classical SNE [Hinton and Roweis, 2002].
- Warning: TNSE has a tendency to show non-existent clusters for small perplexity.
- Scikit-learn implementation : `sklearn.manifold.TSNE(n_components=2,perplexity=50)`.

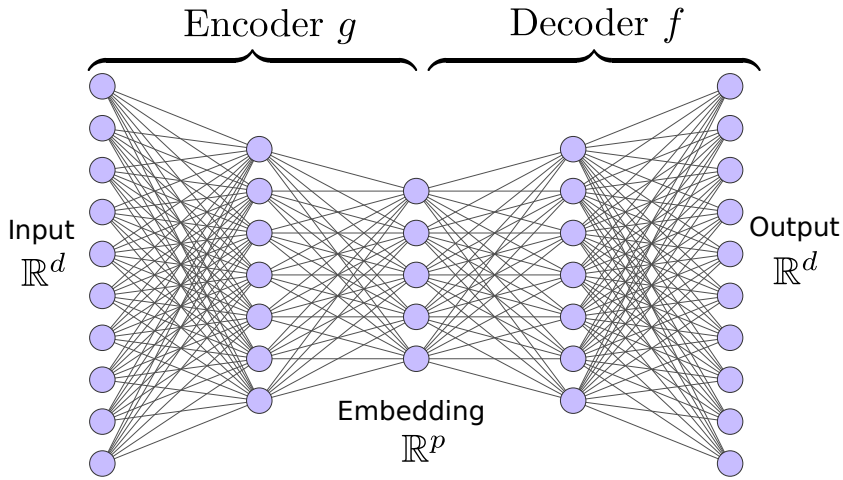
# Uniform Manifold Approximation and Projection (UMAP)




## Principle [McInnes et al., 2018]

- Suppose that the data is uniformly distributed on Riemannian manifold and this manifold is locally connected.
- Construct a graph of neighbors (`n_neighbors` parameter) and approximate the manifold with a Fuzzy topological structure.
- Can be adapted to non-uniform densities [Narayan et al., 2020].
- While not designed to be inductive and invertible, UMAP implementation provide numerical estimation for both.
- More efficient that TSNE on large dataset because does not a requires normalization step to compute pairwise relations.
- Implementation in `umap-learn` : `umap.UMAP(n_components = 2, n_neighbors=n_neighbors)`

# Auto-Encoder (AE)




## Principle (Tutorial [Goodfellow et al., 2016, Chapter 14])

$$\min_{f,g} \quad \sum_{i=1}^{n} L(\mathbf{x}_i, f(g(\mathbf{x}_i)))$$

- Train two neural networks : $g : \mathbb{R}^d \to \mathbb{R}^p$ the encoder and $f : \mathbb{R}^p \to \mathbb{R}^d$ the decoder such that $f(g(\mathbf{x})) \approx \mathbf{x}$.
- Models are often deep neural networks such as $g(\mathbf{x}) = g_K(g_{K-1}(\dots g_1(\mathbf{X})))$ where $g_k(\mathbf{x}) = \sigma(\mathbf{W}_k\mathbf{x} + \mathbf{b}_k)$ and $\sigma$ is a nonlinear activation function.
- When $p < d$ the AE searches for a nonlinear subspace (manifold) that optimize data reconstruction *w.r.t.* the loss L.
- Sparse AE use a regularization (KL [Makhzani and Frey, 2013] or L1/L2 [Arpit et al., 2016]) to promote sparse activations $g(\mathbf{x})$.

# Variational Auto-Encoders (VAE)

## Principle [Kingma and Welling, 2013]

- Estimate probabilistic encoder $q(\mathbf{z}|\mathbf{x})$ and decoder $p(\mathbf{x}|\mathbf{z})$ that model the dataset.
- Optimize the MAP on the data with a Bayesian prior on the embedding is $p(\mathbf{z})$.
- The embedding $q(\mathbf{z}|\mathbf{x})$ of sample $\mathbf{x}$ is a distribution (usually a Normal distribution).
- The reconstruction is also probabilistic : one can generate several reconstructions from and embedding for instance to model uncertainty.

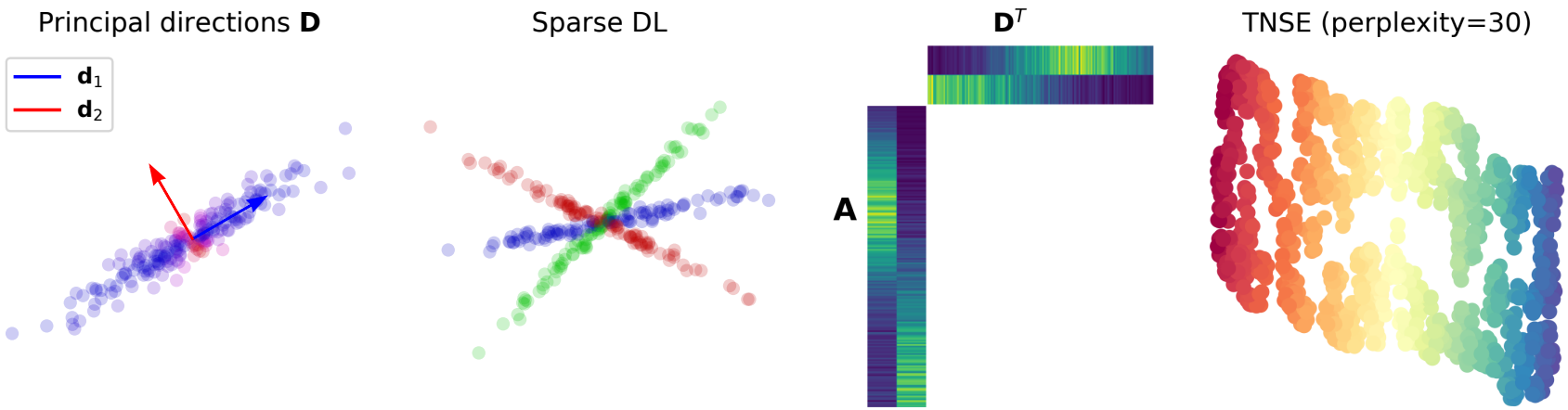
## VAE in practice

- In practice optimization of the Evidence Lower BOund (ELBO) that is a variational lower bound of the MLE or MAP.
- The embedding if often modeled with $z \sim q(\mathbf{z}|\mathbf{x}) = \mathcal{N}(m(\mathbf{x}), \text{diag}(s(\mathbf{x})))$ where $m$ and $s$ are deep neural network predicting the mean and variances respectively.
- The reparametrization trick allows to propagate the gradients with Stochastic Gradient Descent (SGD) by generating samples :

$$\mathbf{z} = m(\mathbf{x}) + s(\mathbf{x}) \odot \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}_p, \mathbf{I}_p)$$

- VAE can be used for data imputation when data is partially observed [Mattei and Frellsen, 2019].

# Dimensionality Reduction (DR) in practice

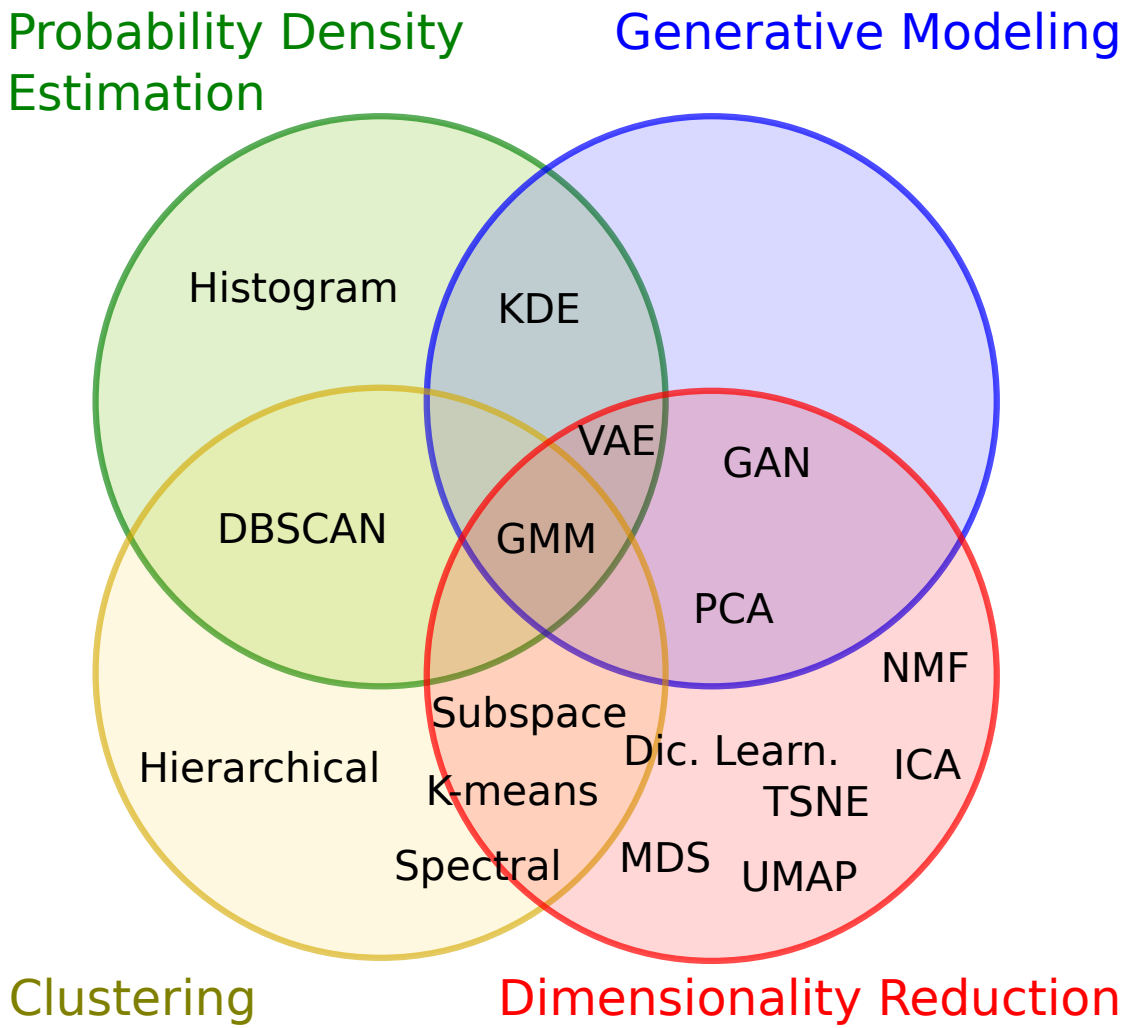



## Why and when?

- Dimensionality reduction is a classical tool for visualizing high dimensional data in 2D but always comes with loss of information ($d = 2$ is very small).
- Inductive and invertible DR methods can be used for denoising because noise is high dimensional and data is usually low dimensional.
- Standard 2D visualization for data/feature manifolds are TSNE and more recently UMAP but beware of false clusters.
- PCA is a classical pre-processing step but quantization with K-means or dictionary learning (bag of visual words) also used in practice.

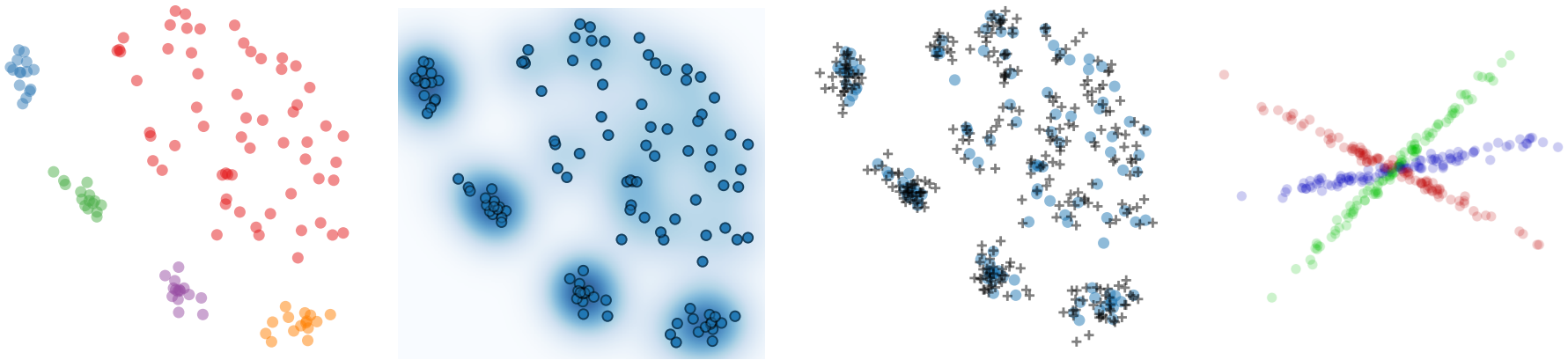
# Section

# Unsupervised learning problems VS methods




Most methods can be used to solve several ML problems.

# Conclusion



## Unsupervised learning

- Most datasets are unlabeled because labeling is expensive (requires humans).
- Unsupervised learning aim at modeling and interpreting the data without human annotations.
- It's difficult to measure and evaluate the quality of the models in unsupervised learning (the criterion is often optimized).
- Numerous research in self-supervised learning (learning representations that are good for predictions without labels, invariant to some variability).
- Most unsupervised learning methods can be used for pre-processing and feature extraction before supervised learning (when inductive).

# References I


[Ankerst et al., 1999] Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J. (1999).
Optics: Ordering points to identify the clustering structure.
*ACM Sigmod record*, 28(2):49–60.

[Arjovsky et al., 2017] Arjovsky, M., Chintala, S., and Bottou, L. (2017).
Wasserstein generative adversarial networks.
In *International conference on machine learning*, pages 214–223. PMLR.

[Arpit et al., 2016] Arpit, D., Zhou, Y., Ngo, H., and Govindaraju, V. (2016).
Why regularized auto-encoders learn sparse representation?
In *International Conference on Machine Learning*, pages 136–144. PMLR.

[Arthur and Vassilvitskii, 2006] Arthur, D. and Vassilvitskii, S. (2006).
k-means++: The advantages of careful seeding.
Technical report, Stanford.

[Berg et al., 2018] Berg, R. v. d., Hasenclever, L., Tomczak, J. M., and Welling, M. (2018).
Sylvester normalizing flows for variational inference.
*arXiv preprint arXiv:1803.05649*.

[Blei and Jordan, 2006] Blei, D. M. and Jordan, M. I. (2006).
Variational inference for dirichlet process mixtures.
*Bayesian analysis*, 1(1):121–143.

# References II


[Bokde et al., 2015] Bokde, D., Girase, S., and Mukhopadhyay, D. (2015).
Matrix factorization model in collaborative filtering algorithms: A survey.
*Procedia Computer Science*, 49:136–146.

[Bottou and Bengio, 1995] Bottou, L. and Bengio, Y. (1995).
Convergence properties of the k-means algorithms.
In *Advances in neural information processing systems*, pages 585–592.

[Bradley et al., 1997] Bradley, P. S., Mangasarian, O. L., and Street, W. N. (1997).
Clustering via concave minimization.
*Advances in neural information processing systems*, pages 368–374.

[Defays, 1977] Defays, D. (1977).
An efficient algorithm for a complete link method.
*The Computer Journal*, 20(4):364–366.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977).
Maximum likelihood from incomplete data via the em algorithm.
*Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

[Dhillon and Sra, 2005] Dhillon, I. S. and Sra, S. (2005).
Generalized nonnegative matrix approximations with bregman divergences.
In *NIPS*, volume 18. Citeseer.

# References III


[Dinh et al., 2016] Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2016).
Density estimation using real nvp.
*arXiv preprint arXiv:1605.08803*.

[Donoho and Grimes, 2003] Donoho, D. L. and Grimes, C. (2003).
Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data.
*Proceedings of the National Academy of Sciences*, 100(10):5591–5596.

[Dziugaite et al., 2015] Dziugaite, G. K., Roy, D. M., and Ghahramani, Z. (2015).
Training generative neural networks via maximum mean discrepancy optimization.
*arXiv preprint arXiv:1505.03906*.

[Epanechnikov, 1969] Epanechnikov, V. A. (1969).
Non-parametric estimation of a multivariate probability density.
*Theory of Probability & Its Applications*, 14(1):153–158.

[Ester et al., 1996] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996).
A density-based algorithm for discovering clusters in large spatial databases with noise.
In *kdd*, volume 96, pages 226–231.

[Févotte et al., 2009] Févotte, C., Bertin, N., and Durrieu, J.-L. (2009).
Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis.
*Neural computation*, 21(3):793–830.

# References IV


[Frey and Dueck, 2007] Frey, B. J. and Dueck, D. (2007).
Clustering by passing messages between data points.
*science*, 315(5814):972–976.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016).
*Deep learning*.
MIT press.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014).
Generative adversarial nets.
In *Advances in neural information processing systems*, pages 2672–2680.

[Ham et al., 2004] Ham, J., Lee, D. D., Mika, S., and Schölkopf, B. (2004).
A kernel view of the dimensionality reduction of manifolds.
In *Proceedings of the twenty-first international conference on Machine learning*, page 47.

[Herault and Jutten, 1986] Herault, J. and Jutten, C. (1986).
Space or time adaptive signal processing by neural network models.
In *AIP conference proceedings*, volume 151, pages 206–211. American Institute of Physics.

[Hinton and Roweis, 2002] Hinton, G. and Roweis, S. T. (2002).
Stochastic neighbor embedding.
In *NIPS*, volume 15, pages 833–840. Citeseer.

# References V


[Houdard et al., 2018] Houdard, A., Bouveyron, C., and Delon, J. (2018).
High-dimensional mixture models for unsupervised image denoising (hdmi).
*SIAM Journal on Imaging Sciences*, 11(4):2815–2846.

[Hyvärinen and Oja, 2000] Hyvärinen, A. and Oja, E. (2000).
Independent component analysis: algorithms and applications.
*Neural networks*, 13(4-5):411–430.

[Kaufman and Rousseeuw, 1990] Kaufman, L. and Rousseeuw, P. J. (1990).
Partitioning around medoids (program pam).
*Finding groups in data: an introduction to cluster analysis*, 344:68–125.

[Kingma et al., 2016] Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016).
Improved variational inference with inverse autoregressive flow.
*Advances in neural information processing systems*, 29:4743–4751.

[Kingma and Welling, 2013] Kingma, D. P. and Welling, M. (2013).
Auto-encoding variational bayes.
*arXiv preprint arXiv:1312.6114*.

[Kobyzev et al., 2020] Kobyzev, I., Prince, S., and Brubaker, M. (2020).
Normalizing flows: An introduction and review of current methods.
*IEEE Transactions on Pattern Analysis and Machine Intelligence*.

# References VI


[Kruskal, 1964] Kruskal, J. B. (1964).
Nonmetric multidimensional scaling: a numerical method.
*Psychometrika*, 29(2):115–129.

[Kwok and Tsang, 2004] Kwok, J.-Y. and Tsang, I.-H. (2004).
The pre-image problem in kernel methods.
*IEEE transactions on neural networks*, 15(6):1517–1525.

[Lee and Seung, 2000] Lee, D. and Seung, H. S. (2000).
Algorithms for non-negative matrix factorization.
*Advances in neural information processing systems*, 13:556–562.

[Li et al., 2015] Li, Y., Swersky, K., and Zemel, R. (2015).
Generative moment matching networks.
In *International Conference on Machine Learning*, pages 1718–1727. PMLR.

[MacQueen et al., 1967] MacQueen, J. et al. (1967).
Some methods for classification and analysis of multivariate observations.
In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

[Mairal et al., 2009] Mairal, J., Bach, F., Ponce, J., and Sapiro, G. (2009).
Online dictionary learning for sparse coding.
In *Proceedings of the 26th annual international conference on machine learning*, pages 689–696.

# References VII


[Makhzani and Frey, 2013] Makhzani, A. and Frey, B. (2013).
K-sparse autoencoders.
*arXiv preprint arXiv:1312.5663*.

[Maranzana, 1963] Maranzana, F. E. (1963).
On the location of supply points to minimize transportation costs.
*IBM Systems Journal*, 2(2):129–135.

[Mattei and Frellsen, 2019] Mattei, P.-A. and Frellsen, J. (2019).
Miwae: Deep generative modelling and imputation of incomplete data sets.
In *International Conference on Machine Learning*, pages 4413–4423. PMLR.

[McInnes et al., 2018] McInnes, L., Healy, J., and Melville, J. (2018).
Umap: Uniform manifold approximation and projection for dimension reduction.
*arXiv preprint arXiv:1802.03426*.

[McLachlan et al., 2019] McLachlan, G. J., Lee, S. X., and Rathnayake, S. I. (2019).
Finite mixture models.
*Annual review of statistics and its application*, 6:355–378.

[Minka, 2000] Minka, T. (2000).
Automatic choice of dimensionality for pca.
*Advances in neural information processing systems*, 13:598–604.

# References VIII


[Narayan et al., 2020] Narayan, A., Berger, B., and Cho, H. (2020).

Density-preserving data visualization unveils dynamic patterns of single-cell transcriptomic variability.

*bioRxiv*.

[Nielsen, 2016] Nielsen, F. (2016).

Hierarchical clustering.

In *Introduction to HPC with MPI for Data Science*, pages 195–211. Springer.

[Nowozin et al., 2016] Nowozin, S., Cseke, B., and Tomioka, R. (2016).

f-gan: Training generative neural samplers using variational divergence minimization.

In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 271–279.

[Parsons et al., 2004] Parsons, L., Haque, E., and Liu, H. (2004).

Subspace clustering for high dimensional data: a review.

*Acm sigkdd explorations newsletter*, 6(1):90–105.

[Parzen, 1962] Parzen, E. (1962).

On estimation of a probability density function and mode.

*The annals of mathematical statistics*, 33(3):1065–1076.

[Pearson, 1901] Pearson, K. (1901).

Liii. on lines and planes of closest fit to systems of points in space.

*The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572.

# References IX


[Radford et al., 2015] Radford, A., Metz, L., and Chintala, S. (2015).
Unsupervised representation learning with deep convolutional generative adversarial networks.
*arXiv preprint arXiv:1511.06434*.

[Rezende and Mohamed, 2015] Rezende, D. and Mohamed, S. (2015).
Variational inference with normalizing flows.
In *International conference on machine learning*, pages 1530–1538. PMLR.

[Rosenblatt, 1956] Rosenblatt, M. (1956).
Remarks on Some Nonparametric Estimates of a Density Function.
*The Annals of Mathematical Statistics*, 27(3):832 – 837.

[Roweis and Saul, 2000] Roweis, S. T. and Saul, L. K. (2000).
Nonlinear dimensionality reduction by locally linear embedding.
*science*, 290(5500):2323–2326.

[Schölkopf et al., 1997] Schölkopf, B., Smola, A., and Müller, K.-R. (1997).
Kernel principal component analysis.
In *International conference on artificial neural networks*, pages 583–588. Springer.

[Schubert et al., 2017] Schubert, E., Sander, J., Ester, M., Kriegel, H. P., and Xu, X. (2017).
Dbscan revisited, revisited: why and how you should (still) use dbscan.
*ACM Transactions on Database Systems (TODS)*, 42(3):1–21.

# References X


[Sculley, 2010] Sculley, D. (2010).
Web-scale k-means clustering.
In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178.

[Shi and Malik, 2000] Shi, J. and Malik, J. (2000).
Normalized cuts and image segmentation.
*IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905.

[Sibson, 1973] Sibson, R. (1973).
Slink: an optimally efficient algorithm for the single-link cluster method.
*The computer journal*, 16(1):30–34.

[Sokal, 1958] Sokal, R. R. (1958).
A statistical method for evaluating systematic relationships.
*Univ. Kansas, Sci. Bull.*, 38:1409–1438.

[Steinhaus et al., 1956] Steinhaus, H. et al. (1956).
Sur la division des corps matériels en parties.
*Bull. Acad. Polon. Sci*, 1(804):801.

[Tenenbaum et al., 2000] Tenenbaum, J. B., De Silva, V., and Langford, J. C. (2000).
A global geometric framework for nonlinear dimensionality reduction.
*science*, 290(5500):2319–2323.

# References XI


[Tipping and Bishop, 1999] Tipping, M. E. and Bishop, C. M. (1999).
Probabilistic principal component analysis.
*Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622.

[Tomczak and Welling, 2017] Tomczak, J. M. and Welling, M. (2017).
Improving variational auto-encoders using convex combination linear inverse autoregressive flow.
*arXiv preprint arXiv:1706.02326*.

[Van der Maaten and Hinton, 2008] Van der Maaten, L. and Hinton, G. (2008).
Visualizing data using t-sne.
*Journal of machine learning research*, 9(11).

[Von Luxburg, 2007] Von Luxburg, U. (2007).
A tutorial on spectral clustering.
*Statistics and computing*, 17(4):395–416.

[Ward Jr, 1963] Ward Jr, J. H. (1963).
Hierarchical grouping to optimize an objective function.
*Journal of the American statistical association*, 58(301):236–244.

[Yu et al., 2011] Yu, G., Sapiro, G., and Mallat, S. (2011).
Solving inverse problems with piecewise linear estimators: From gaussian mixture models to structured sparsity.
*IEEE Transactions on Image Processing*, 21(5):2481–2499.

# References XII

[Zhang and Wang, 2007] Zhang, Z. and Wang, J. (2007).

MIle: Modified locally linear embedding using multiple weights.

In *Advances in neural information processing systems*, pages 1593–1600. Citeseer.

[Zou et al., 2006] Zou, H., Hastie, T., and Tibshirani, R. (2006).

Sparse principal component analysis.

*Journal of computational and graphical statistics*, 15(2):265–286.