GHENT UNIVERSITY

FACULTY OF ENGINEERING AND ARCHITECTURE

E016350 - Artificial Intelligence

Lecture 12

Machine Learning

Unsupervised learning

Aleksandra Pizurica

Ghent University
Fall 2024

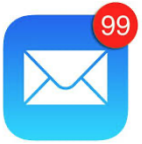
Overview

- Dimensionality reduction
  - PCA
  - A glimpse of nonlinear methods – basic insights
- Clustering
  - K-Means
  - Examples of more advanced methods

Andrew Ng: Lecture notes – Machine Learning (CS229), Ch. 10 & Ch 12

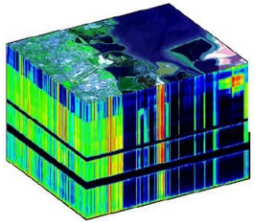
High-dimensional data

High-dimensional data



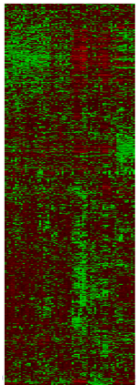
High-dimensional data

100+
bands

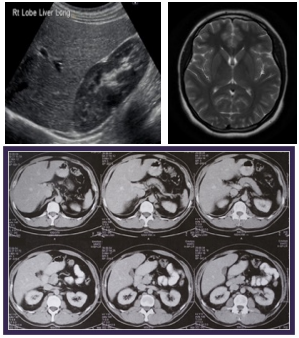


Hyperspectral remote sensing image

1000+
genes

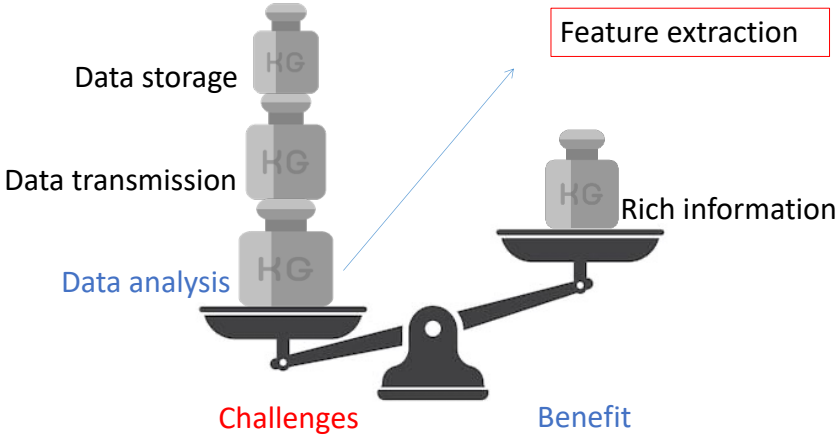


DNA microarray



Medical images
(Ultrasound, MRI, CT scan)

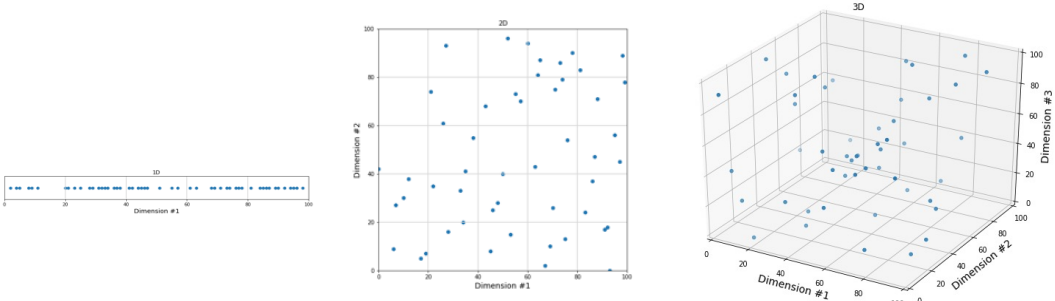
High-dimensional data



Dimensionality reduction

- Goal: convert the high-dimensional data set $\mathcal{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}\}$, $\mathbf{x}^{(i)} \in \mathbb{R}^d$ into a lower-dimensional data $\mathcal{Y} = \{\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots, \mathbf{y}^{(N)}\}$, $\mathbf{y}^{(i)} \in \mathbb{R}^k$, $k << d$
- Often k is 2 or 3 so that $\mathcal{Y}$ can be displayed in a scatterplot
- Note this is different from pure visualization of high-dimensional data
- $\mathbf{y}^{(i)}$ are often refereed to as the map points of $\mathbf{x}^{(i)}$ and $\mathcal{Y}$ is called a map
- Traditional **linear** dimensionality reduction techniques focus on on keeping the low-dimensional representations of dissimilar datapoints far apart.
  - Principal Component Analysis (PCA)
  - Multidimensional scaling (MDS)
- **Nonlinear** dimensionality reduction techniques
  aim to **preserve the local structure**

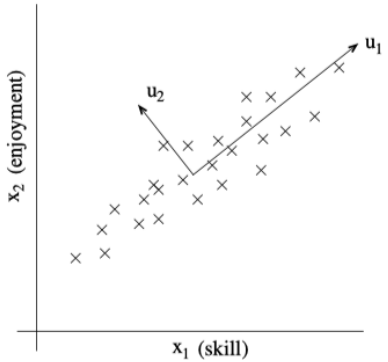
Curse of dimensionality



The same number of data points shown in 1D, 2D and 3D.

- When the data dimensionality increases, the volume of the space increases so fast that the available data become sparse
- The amount of data needed for training grows exponentially with the number of features/dimensions

Principal Component Analysis – PCA



["Piloting example", Andrew Ng]

We may reason that data actually lies along some diagonal axis (the u_1 direction) capturing the intrinsic information, with only a small amount of noise lying off this axis.

$\rightarrow$ How can we automatically compute this u_1 direction?

Preprocessing

Preprocess the data by normalizing each feature to have mean 0 and variance 1:

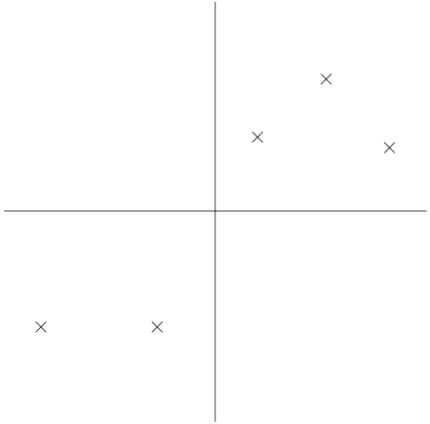
$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{\sigma_j}$$

where

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_j^{(i)} \quad \text{and} \quad \sigma_j^2 = \frac{1}{N}(x_j^{(i)} - \mu_j)^2$$

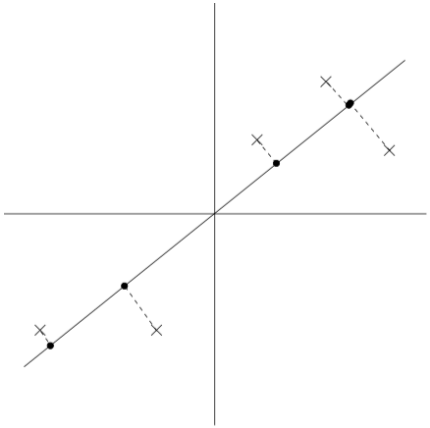
Dividing by σ_j rescales each coordinate to have unit variance, which ensures that different attributes are all treated on the same "scale"

Finding the major axis of variation



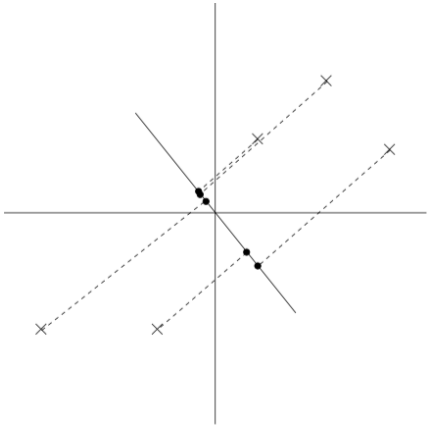
Suppose we have already carried out the normalization and now seeking the projection

Finding the major axis of variation – Intuition



Projected data still has a fairly large variance, and the points tend to be far from zero.

Finding the major axis of variation – Intuition



In contrast, here much smaller variance, and points much closer to the origin.

Finding the major axis of variation – Formally

For a point $\mathbf{x}$ and a unit vector $\mathbf{u}$, the length of the projection of $\mathbf{x}$ onto $\mathbf{u}$ is $\mathbf{x}^\top \mathbf{u}$

Hence, to maximize the variance of the projections, choose $\mathbf{u}$ so as to maximize:

$$\frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}^{(i)^\top}\mathbf{u})^2 = \frac{1}{N}\sum_{i=1}^{N}\mathbf{u}^\top\mathbf{x}^{(i)}\mathbf{x}^{(i)^\top}\mathbf{u} = \mathbf{u}^\top\underbrace{\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}^{(i)}\mathbf{x}^{(i)^\top}\right)}_{\boldsymbol{\Sigma}}\mathbf{u}$$

We have $\min_{\mathbf{u}} \mathbf{u}^\top\boldsymbol{\Sigma}\mathbf{u}$ subject to $\|\mathbf{u}\|_2 = 1$

Using the method of Lagrange multipliers for some λ this becomes:

$$\min_{\mathbf{u}} \mathbf{u}^\top\boldsymbol{\Sigma}\mathbf{u} + \lambda(1 - \mathbf{u}^\top\mathbf{u})$$

By setting the derivative w.r.t. $\mathbf{u}$ equal to zero, this gives $\mathbf{u}^\top\boldsymbol{\Sigma}\mathbf{u} = \lambda$

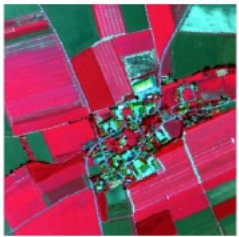
$\Rightarrow$ $\mathbf{u}$ should be the principal **eigen vector** of $\boldsymbol{\Sigma}$

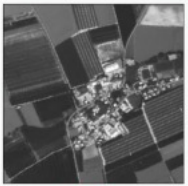
Finding the major axis of variation – Formally

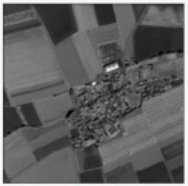
- We showed: to project the data into a 1-D subspace, preserving the maximum variance, $\mathbf{u}$ should be the principal eigenvector of $\boldsymbol{\Sigma}$
- More generally, to project the data into a k-dimensional subspace ($k < d$), we should choose $\mathbf{u}_1, \ldots, \mathbf{u}_k$ to be the top $\mathbf{k}$ eigenvectors of $\boldsymbol{\Sigma}$.
  - The $\mathbf{u}_i$'s now form a new, **orthogonal basis** for the data.
  - the map point of $\mathbf{x}^{(i)}$ in this basis is

$$\mathbf{y}^{(i)} = \begin{bmatrix} \mathbf{u}_1^\top \mathbf{x}^{(i)} \\ \mathbf{u}_2^\top \mathbf{x}^{(i)} \\ \vdots \\ \mathbf{u}_k^\top \mathbf{x}^{(i)} \end{bmatrix} \in \mathbb{R}^k$$

Example of dimensionality reduction in HSI



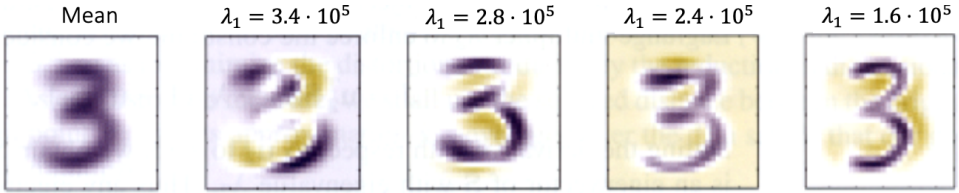




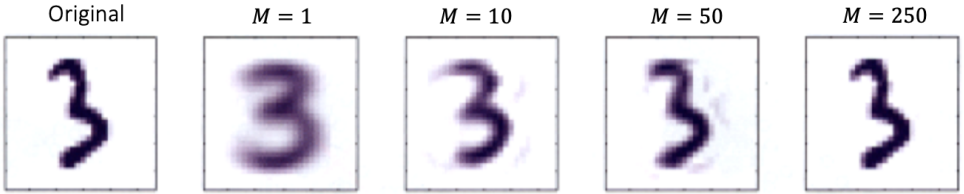
Demonstration of the informative content of different components. The dataset has 126 spectral bands (HyMAP), but only NIR, Red and Green are shown. (Neusling, Bavaria, Germany)

In courtesy of Martin Danner

Other applications: Data compression



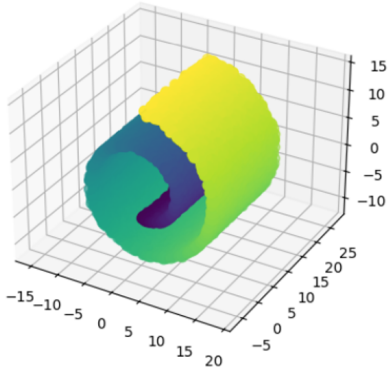
The mean vector and the first four PCA eigen vectors u_1, ..., u_4 for a digit dataset, with the corresponding eigen values.



An original example from the dataset together with its reconstructions obtained retaining M PCs for various M.
The reconstruction would be perfect for M=28 x 28 =784

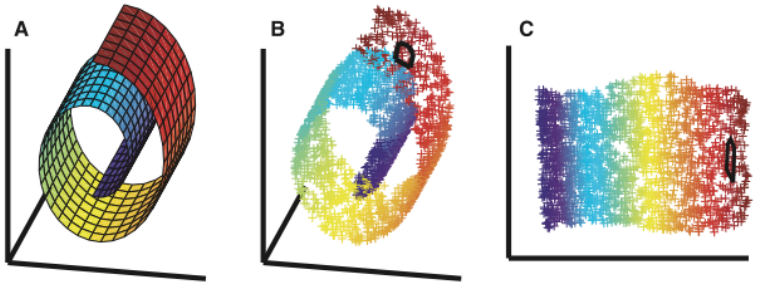
Example from C. Bishop: *Pattern Recognition and Machine Learning*

Limitations



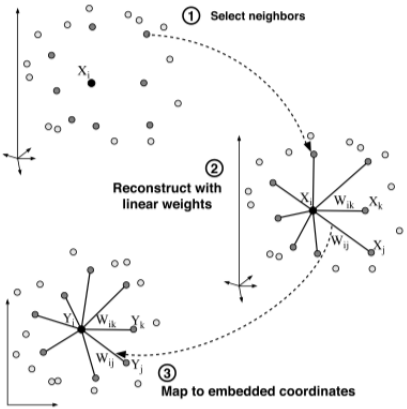
PCA doesn't unroll the structure. We need nonlinear methods for that!

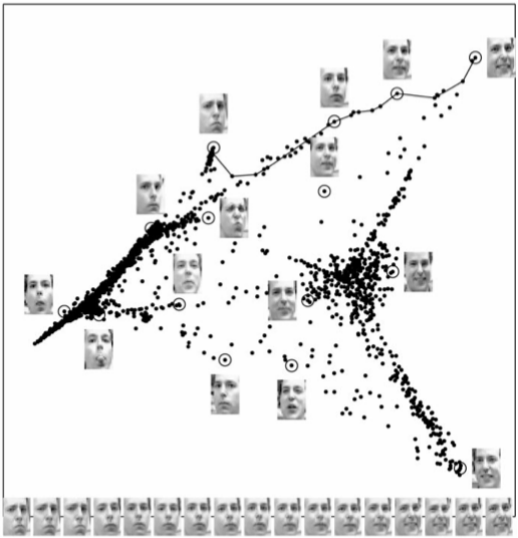
Nonlinear dimensionality reduction methods



S. T. Roweis and L. K. Saul: Nonlinear Dimensionality Reduction by Locally Linear Embedding, Science, 2000.

Locally Linear Embedding (LLE)

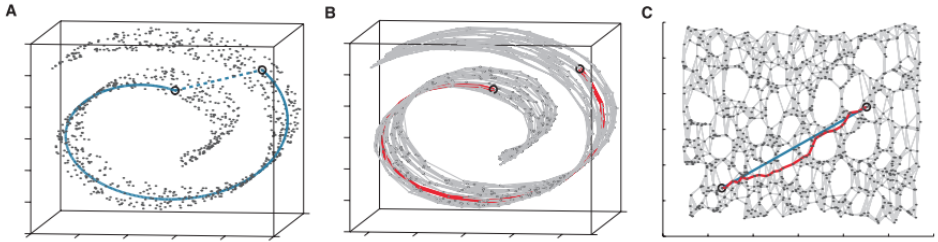




S. T. Roweis and L. K. Saul: Nonlinear Dimensionality Reduction by Locally Linear Embedding, Science, 2000.

Isomap

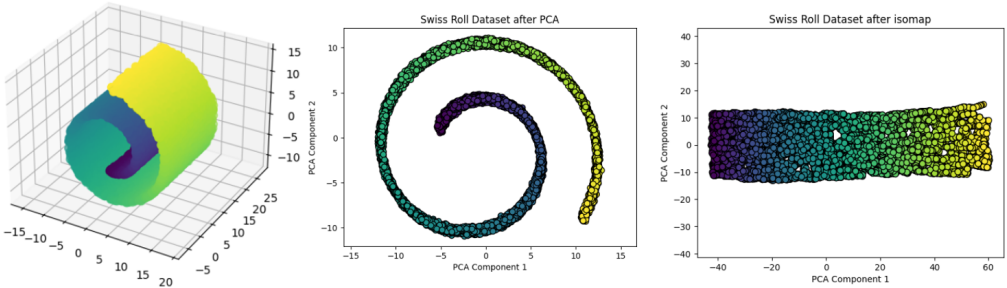
Idea: preserve geodesic distances in the lower-dimension



(A) In the highdimensional input space the Euclidean distance may not accurately reflect the intrinsic similarity between data points, as measured by geodesic distance. (B) The neighborhood graph allows an approximation (red segments) to the true geodesic path. (C) The 2-D embedding, which best preserves the shortest path distances in the neighborhood graph.

J. B. Tenenbaum, V. de Silva, J. C. Langford: A Global Geometric Framework for Nonlinear Dimensionality Reduction, Science, 2000.

Example: Swiss roll after PCA and Isomap



Experiment with this in the notebook 'Principal Component Analysis' (Ufora).

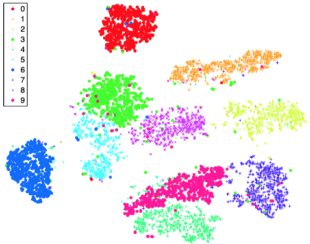
t-SNE

- A variant of SNE (Stochastic Neighbor Embedding) [Hinton & Roweis, 2002]
- SNE starts by converting the high-dimensional Euclidean distances between datapoints into conditional probabilities that represent similarities
  - The similarity of $\mathbf{x}^{(j)}$ to $\mathbf{x}^{(i)}$ is the conditional probability, $p_{j|i}$, that $\mathbf{x}^{(i)}$ would pick $\mathbf{x}^{(j)}$ as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian centered at $\mathbf{x}^{(i)}$, i.e.

$$p_{j|i} = \frac{\exp(\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(\|\mathbf{x}^{(i)} - \mathbf{x}^{(k)}\|^2/2\sigma_i^2)}$$

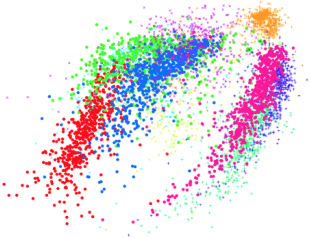
  - Do equivalently for the low-dimensional counterparts $\mathbf{y}^{(i)}$ and $\mathbf{y}^{(j)}$ of the high-dim. datapoints $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ and denote this conditional probability by $q_{j|i}$. (Some technicalities: normalize the variance of the Gaussian and set $p_{i|i} = 0$ and $q_{i|i} = 0$)
  - Idea: If the map points $\mathbf{y}^{(i)}$ and $\mathbf{y}^{(j)}$ correctly model the similarity between $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$, the conditional probabilities $p_{j|i}$ and $q_{j|i}$ will be equal
    - ⋆ Minimize the Kullback-Leibler (KL) divergence between $p_{j|i}$ and $q_{j|i}$
    - ⋆ practically, minimize KL divergences over all datapoints using gradient descent

Example: handwritten digits from the MNIST data set

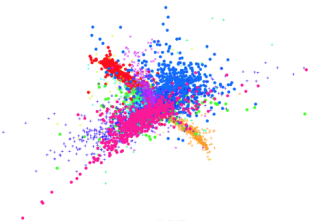


t-SNE

Sammon mapping



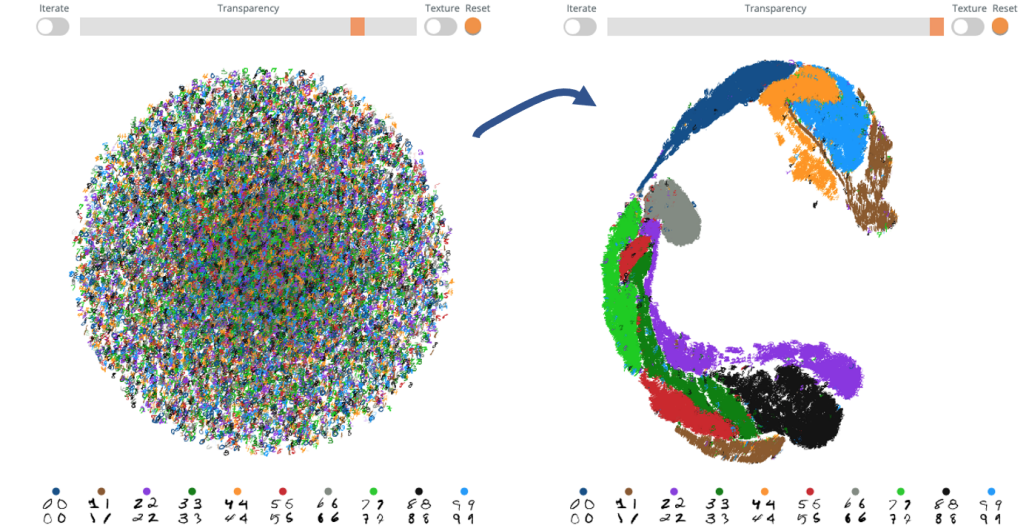
Isomap



LLE

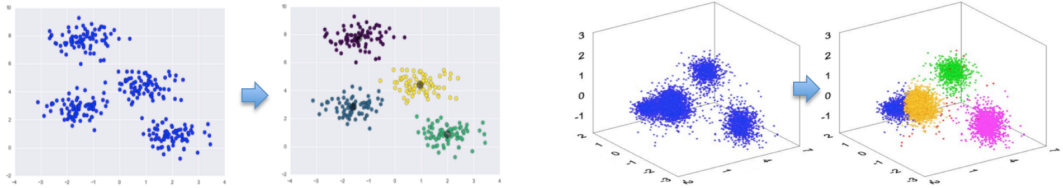
L. van der Maaten and G. Hinton: Visualizing Data using t-SNE, Journal of Machine Learning Research, 2008.

Demo: web-based fast t-SNE



https://nicola17.github.io/tfjs-tsne-demo/

Clustering



Example: motion segmentation:

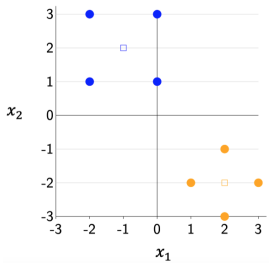
E. Elhamifar and R. Vidal: Sparse Subspace Clustering: Algorithm, Theory, and Applications, IEEE Trans. Pattern Anal. Mach. Intell. 2013

Clustering task

Given the input training data $\mathcal{D}_{train} = [\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}]$, output the **assignment** of each point to a cluster. The assignment vector is:

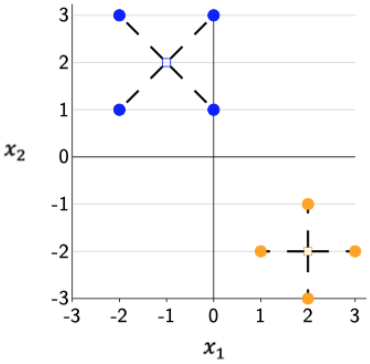
$$\mathbf{c} = [c^{(1)}, \ldots, c^{(N)}], \quad \text{where} \quad c^{(i)} \in \{1, \ldots K\}$$

Each **cluster** is represented by a **centroid** $\boldsymbol{\mu} = \mu_1, \ldots \mu_k$



Intuition: we want each point $\mathbf{x}^{(i)}$ to be close to its assigned centroid $\mu_{c^{(i)}}$

K-Means

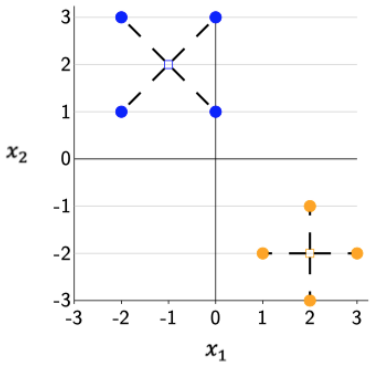


$$Loss_{kmeans}(\mathbf{c}, \boldsymbol{\mu}) = \sum_{j=1}^{N} \|\mathbf{x}^{(j)} - \mu_{c^{(j)}}\|$$

K-Means objective:

$$\min_{\mathbf{c}, \boldsymbol{\mu}} Loss_{kmeans}(\mathbf{c}, \boldsymbol{\mu})$$

K-Means algorithm



K-Means

Initialize $\boldsymbol{\mu} = \mu_1, \dots \mu_k$ randomly
Iterate:

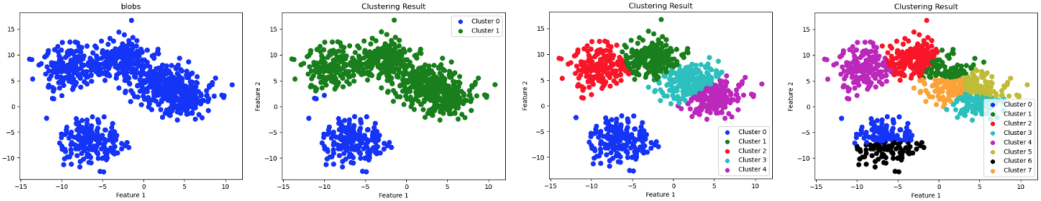
- **Step 1**: Set assignments $\mathbf{c}$ given $\boldsymbol{\mu}$ ($\mathbf{x}^{(j)} \rightarrow$ nearest centroid):

$$\forall j, \quad c^{(j)} = \arg \min_{i=1,\dots,k} \|\mathbf{x}^{(j)} - \mu_i\|^2$$

- **Step 2**: Set centroids $\boldsymbol{\mu}$ given $\mathbf{c}$ $i = 1, \dots, k$:

$$\mu_i \leftarrow \frac{1}{|\{j : c^{(j)} = i\}|} \sum_{j:c^{(j)}=i} \mathbf{x}^{(j)}$$

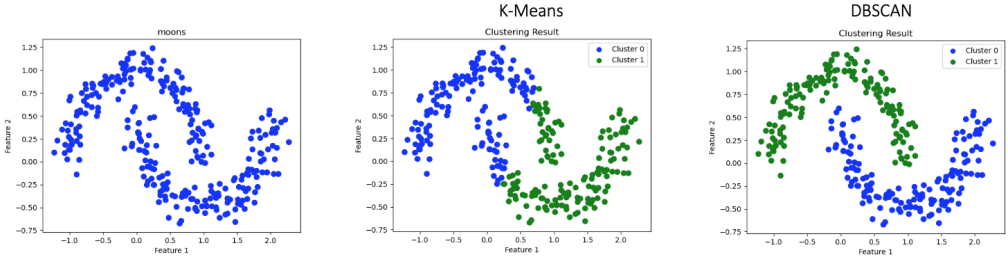
Example



The number of clusters is a parameter of K-Means

Experiment with this in the notebook 'Clustering Tutorial' (Ufora).

Other clustering algorithms



DBSCAN – Density-based clustering (determines automatically the number of clusters)

Experiment with this in the notebook 'Clustering Tutorial' (Ufora).

K-Means vs GMM

K-Means

Initialize $\boldsymbol{\mu} = \mu_1, \dots \mu_k$ randomly
Iterate:

- **Step 1**: Set assignments $\mathbf{c}$ given $\boldsymbol{\mu}$ ($\mathbf{x}^{(j)} \rightarrow$ nearest centroid):

$$\forall j, \quad c^{(j)} = \arg \min_{i=1,\dots,k} \|\mathbf{x}^{(j)} - \mu_i\|^2$$

- **Step 2**: Set centroids $\boldsymbol{\mu}$ given $\mathbf{c}$ $i = 1, \dots, k$:

$$\mu_i \leftarrow \frac{1}{|\{j : c^{(j)} = i\}|} \sum_{j:c^{(j)}=i} \mathbf{x}^{(j)}$$

GMM

Initialize $\mu_i, \Sigma_i, i = 1, \dots k$
Iterate:

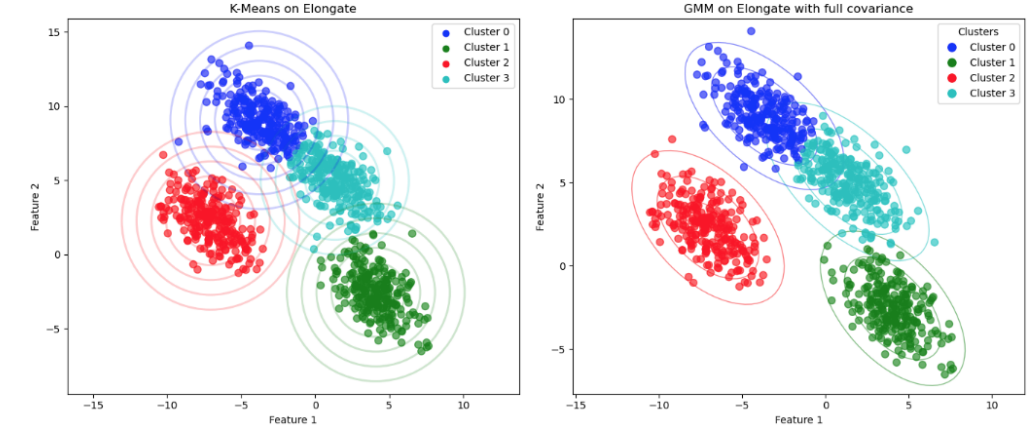
- **E-step**: Find probabilities that data points were generated by different components

$$\forall i, j \quad p_{ij} = P(C = i|\mathbf{x}^{(j)})$$

- **M-step**: Find new parameters that maximize the log-likelihood:

Using p_{ij}, update μ_i, Σ_i, π_i

K-Means vs GMM



Experiment with this in the notebook 'Kmeans&Gmm' (Ufora).