Clustering

Unsupervised data analysis

Mathieu Lagrange & Diana Mateus
mathieu.lagrange@univ-nantes.fr

CENTRALE NANTES

https://mathieulagrange.github.io

Table of contents

Outline

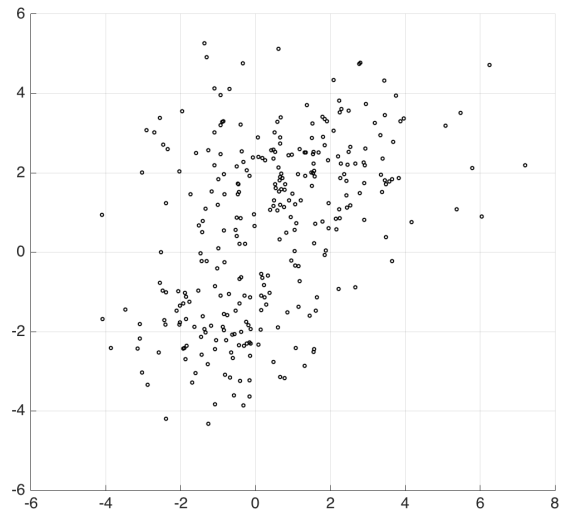
What is clustering?

Definition (Clustering)

Given n unlabelled data points, separate them into K clusters.

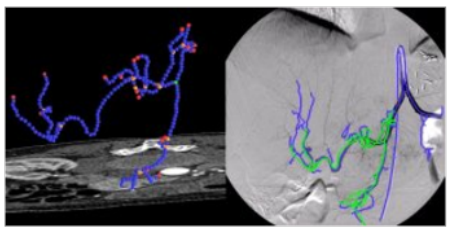
Dilemma!

- What is a Cluster? (Compact vs. Connected)
- How many K clusters? (Parametric vs. Non-parametric)
- Soft vs. Hard clustering. (Model vs. Cost based)
- Data representation. (Vector vs. Similarities)
- Classification vs. Clustering.
- Stability.



Applications

- Retrieval
- Compression
- Segmentation
- Pattern Recognition
- ...



Notation

- $\mathcal{X}^T = \{x_1, x_2, ..., x_N\} \in \mathbb{R}^{d \times N}$ is the data set.
- d is the feature dimension of x_i.
- N is the number of instances.
- K is the number of clusters.
- $\nabla = \{C_1, C_2, ..., C_K\}$, where C_k is a partition of $\mathcal{X}$.
- $c(x_i)$ is the label/cluster of instance x_i.
- r_{nk} where n is the index of instance and k is the index of cluster.

Objective

Find the clusters ∇ minimizing the cost function $\mathcal{L}(\nabla)$.

Parametric, cost-based clustering

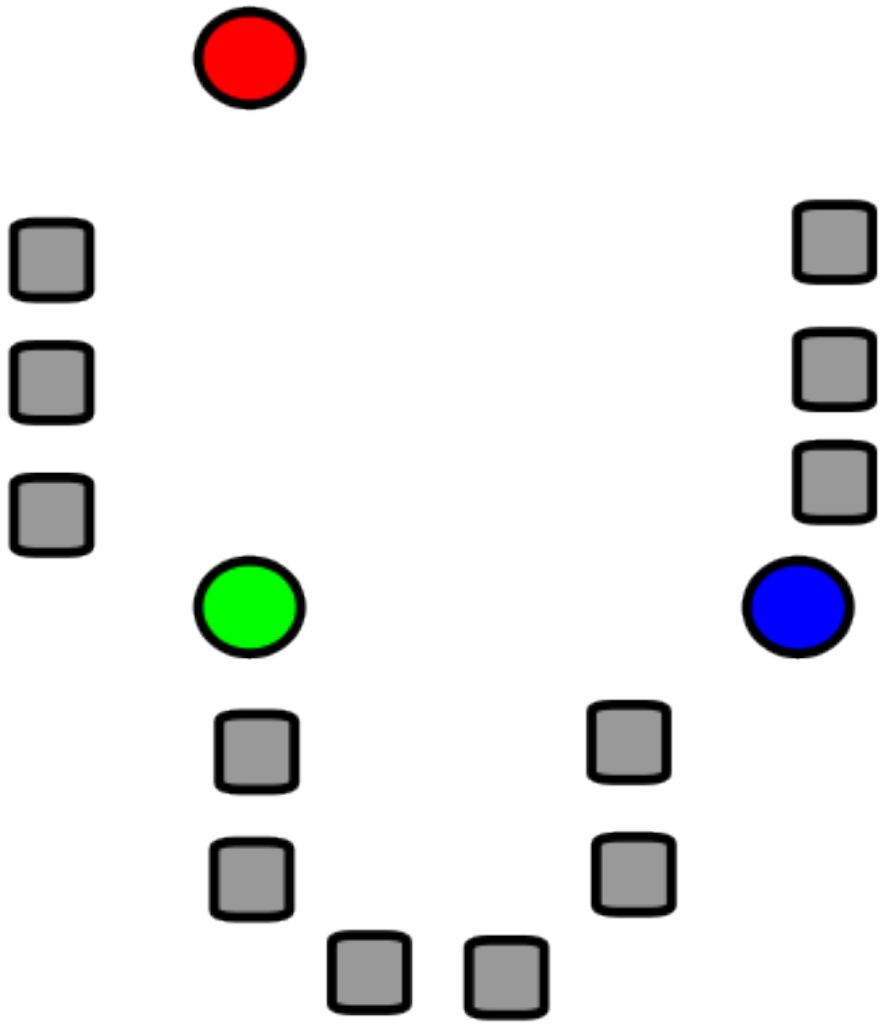
Parametric: K is defined.
Cost-based: It is hard-clustering based on the cost function.
Selected Algorithms:

- K-Means
- K-Medoids
- Kernel K-Means
- Spectral Clustering

K-Means

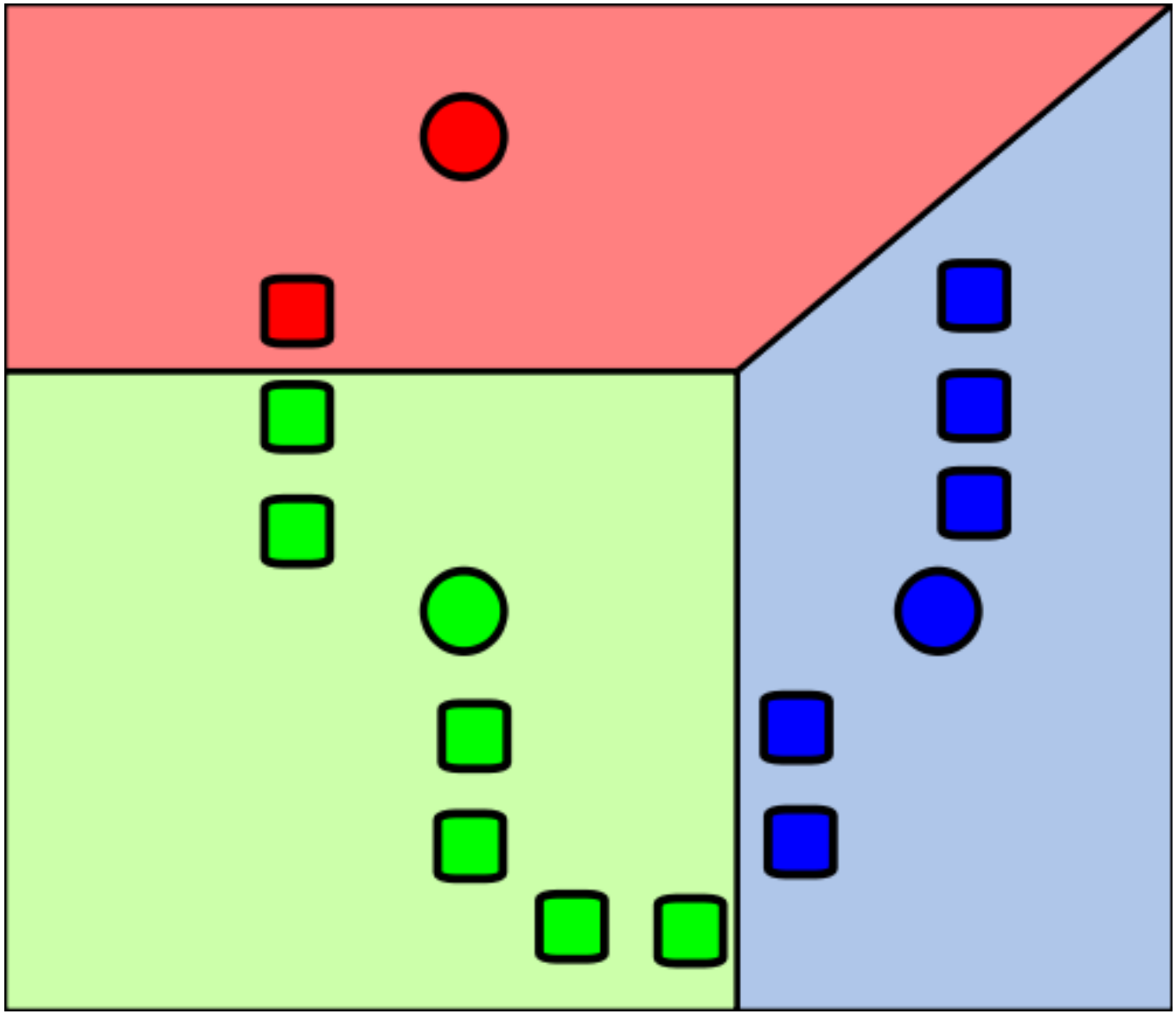
- K-Means algorithm:
  1. **Initialize:** Pick K random samples from the dataset $\mathcal{X}^T$ as the cluster centroids $\mu_k = \{\mu_1, \mu_2, ..., \mu_K\}$.
  2. **Assign Points to the clusters**: Partition data points $\mathcal{X}^T$ into K clusters $\nabla = \{C_1, C_2, ..., C_K\}$ based on the Euclidean distance between the points and centroids (searching for the closest centroid).
  3. **Centroid update**: Based on the points assigned to each cluster, a new centroid is computed μ_k.
  4. **Repeat**: Do step 2 and 3 until convergence.
  5. **Convergence**: if the cluster centroids barley change, or we have compact and/or isolated clusters. Mathematically, when the cost (distortion) function $\mathcal{L}(\nabla) = \sum_{k=1}^{K} \sum_{i \in C_k} \left\| x_i - \mu_k \right\|^2$ is minimum.
- **Practical issues:**
  a) The initialization. b) Pre-processing.

Example

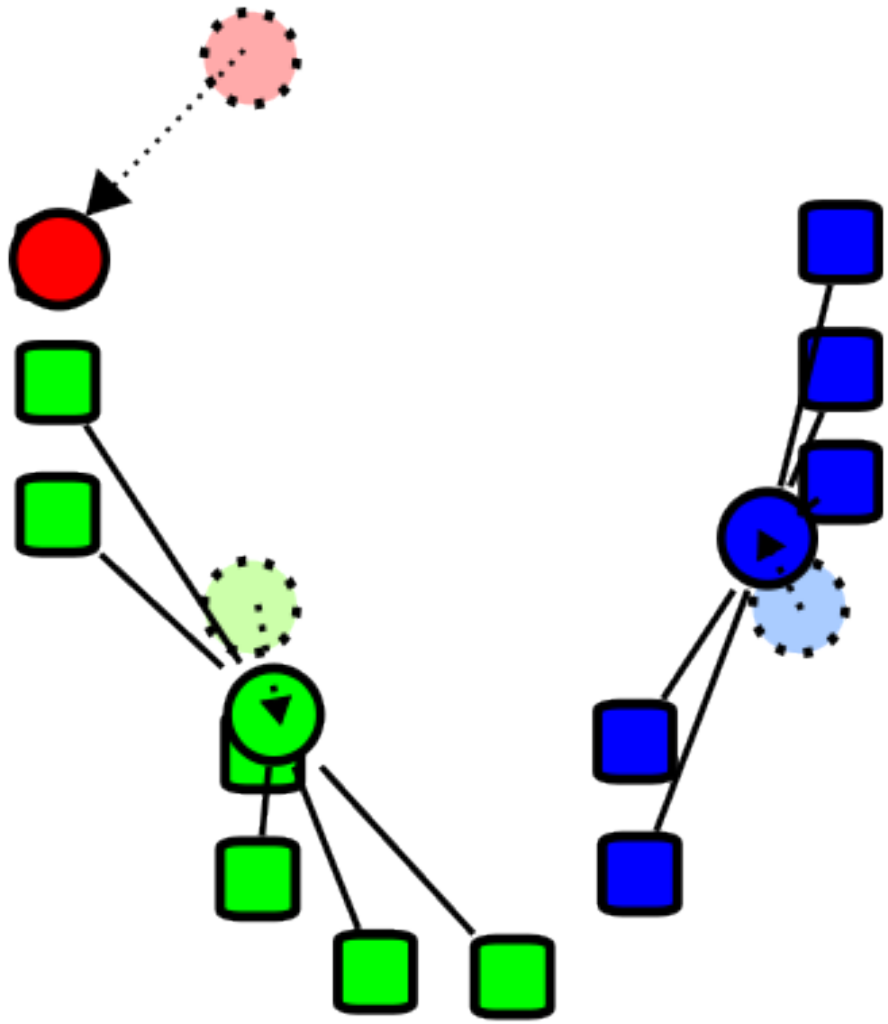


K-Means

Example



Example



K-Means – Algorithm

input : Data points $\mathcal{X}^T = \{x_1, x_2, ..., x_N\}$, number of clusters K
output: Clusters, $\nabla = \{C_1, C_2, ..., C_K\}$

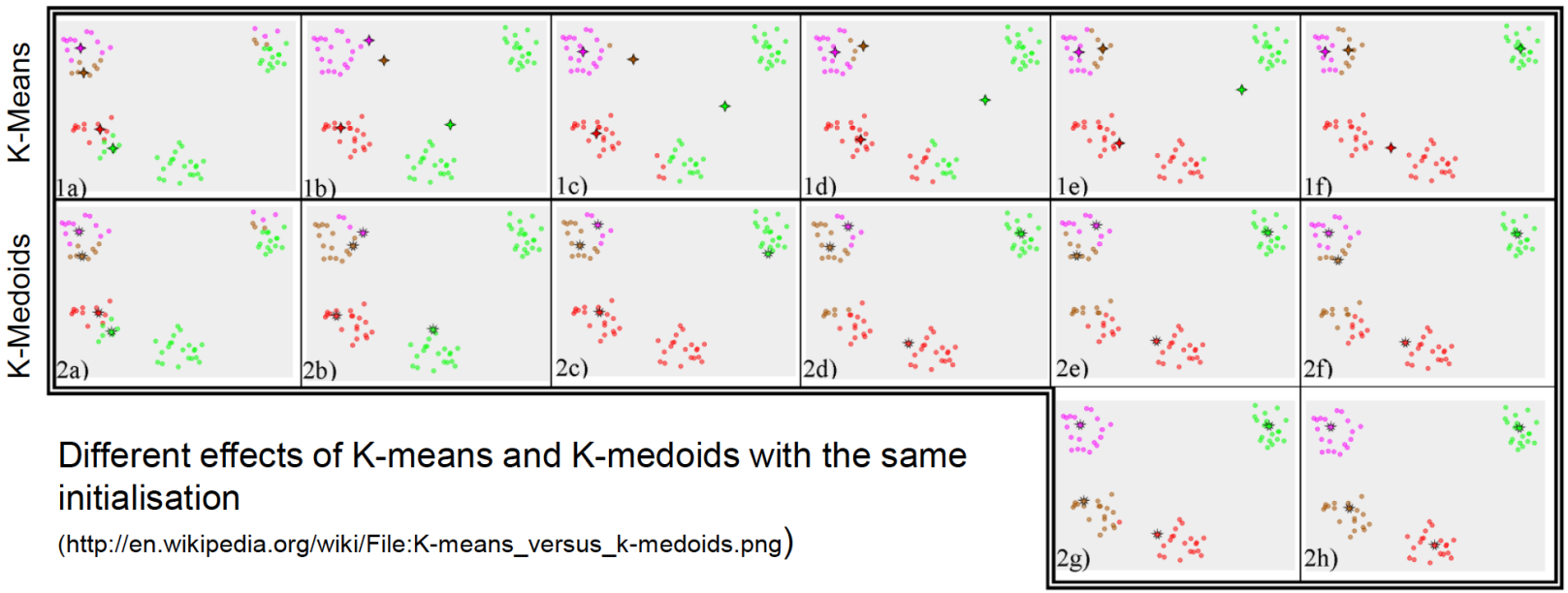
Pick K random samples as the cluster centroids μ_k.
repeat
  for $i = 1$ **to** N **do**
    $c(x_i) = \min_{k \in K} \|x_i - \mu_k\|_2^2$ %Assign points to clusters
  end
  for $k = 1$ **to** K **do**
    $\mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$ %Update the cluster centroid
  end
until *convergence*;

K-Medoids I

- K-Medoids algorithm:
  1. **Initialize:** Pick K random samples from the dataset $\mathcal{X}^T$ as the medoids $\mu_k = \{\mu_1, \mu_2, ..., \mu_K\}$.
  2. **Assign Points to the clusters**: Partition data points $\mathcal{X}^T$ into K clusters $\nabla = \{C_1, C_2, ..., C_K\}$ based on the dissimilarity (Manhattan) distance between the points and medoids (searching for the min. dissimilarity).
  3. **Medoids update**: Based on the points assigned to each cluster, swap the medoid with a new data point and compute the cost. (undo the swap if the cost is getting increased).
  4. **Repeat**: Do step 2 and 3 until convergence.
  5. **Convergence**: if the cluster medoids barley change. Mathematically, when the cost (distortion) function $\mathcal{L}(\nabla) = \sum_{k=1}^{K} \sum_{i \in C_k} \left\| x_i - \mu_k \right\|$ is minimum.

K-Medoids

K-Medoids II



Different effects of K-means and K-medoids with the same initialisation

(http://en.wikipedia.org/wiki/File:K-means_versus_k-medoids.png)

Figure: K-Means vs. K-Medoids

Kernel K-Means I

Definition

It is a generalization of the standard K-Means algorithm.

- What happens if the clusters are not linearly separable?
- Euclidean distance vs. Geodesic distance.

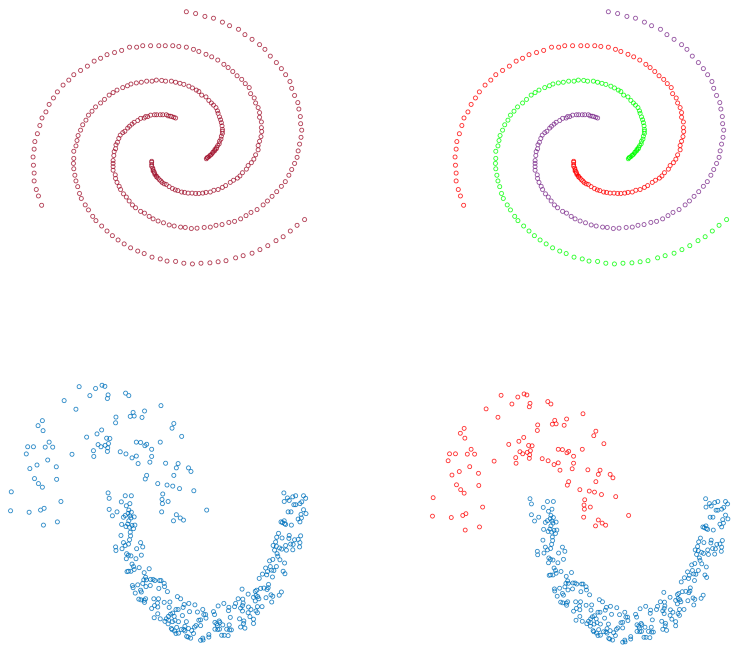


Figure: Spiral and Jain datasets

Kernel K-Means II

- K-Means can not be applied right away.
- Map the data points $x_i \in \mathcal{X}^T$ to a high dimensional feature space $\mathcal{M}$ using a nonlinear function $\phi(x_i)$.
- Assume the clusters in the high dimensional feature space (RKHS) is linearly separable, hence K-Means can be applied.
- The cost function would be

$$\mathcal{L}_{\mathcal{K}}(\nabla) = \sum_{k=1}^{K} \sum_{i \in C_k} \left\| \phi(x_i) - \phi(\mu_k) \right\|^2,$$

where $\left\| \phi(x_i) - \phi(\mu_k) \right\|^2 =$
$\phi(x_i)^T \cdot \phi(x_i) - 2\phi(x_i)^T \cdot \phi(\mu_k) + \phi(\mu_k)^T \cdot \phi(\mu_k).$

Kernel K-Means III

- Using the kernel trick, $K_{ij} = \phi(x_i)^T \cdot \phi(x_j)$, the Euclidean distance in $\mathcal{L}_{\mathcal{K}}(\nabla)$ can be computed easily using any kernel function K_{ij} without explicitly knowing the nonlinear transformation $\phi(x_i)$.
- Examples of kernel functions (positive semidefinite)
  1. Hom. Polynomial kernel: $K_{ij} = (x_i^T x_j)^\delta$
  2. Inho. Polynomial kernel: $K_{ij} = (x_i^T x_j + \gamma)^\delta$
  3. Gaussian kernel: $K_{ij} = e^{\frac{-\|x_i - x_j\|^2}{2\sigma^2}}$
  4. Laplacian kernel $K_{ij} = e^{\frac{-\|x_i - x_j\|}{\sigma}}$
  5. Sigmoid kernel: $K_{ij} = \tanh(\gamma(x_i^T x_j) + \theta)$

Polynomial kernel https://youtu.be/3liCbRZPrZA

Kernel K-Means – Algorithm

input : Data points $\mathcal{X}^T = \{x_1, x_2, ..., x_N\}$, Kernel matrix K_{ij}, number of clusters K
output: Clusters, $\nabla = \{C_1, C_2, ..., C_K\}$

Pick K random samples as the cluster centroids μ_k.
repeat
  for $i = 1$ **to** N **do**
    for $k = 1$ **to** K **do**
      Compute $\left\|\phi(x_i) - \phi(\mu_k)\right\|^2$ using K_{ij}
    end
    $c(x_i) = \min_{k \in K} \left\|\phi(x_i) - \phi(\mu_k)\right\|^2$
  end
  for $k = 1$ **to** K **do**
    $\mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$
  end
until *convergence*;

Spectral Clustering

Graph - Overview

- Fully connected, undirected, and wighted graph with N vertices
- The graph is represented by $G = \{\nu, \varepsilon, \omega\}$, where ν is a set of vertices N, ε is a set of edges, and ω is a set of weights are assigned using a heat kernel as follows to build the Adjacency matrix W

$$W_{ij} = \begin{cases} e^{-\frac{\|x_i - x_j\|_2^2}{\sigma^2}} & e_{ij} \in \varepsilon \\ 0 & else \end{cases}$$

- The degree matrix D, where its diagonal elements $D_{ij} = \sum_j W_{ij}$
- Compute the Normalized graph Laplacian Matrix

$$\tilde{\mathcal{L}} = I - D^{-1/2} W D^{-1/2}$$

Spectral Clustering – Algorithm

input : Normalized Laplacian Matrix $\tilde{\mathcal{L}}$, number of clusters K
output: Clusters, $\nabla = \{C_1, C_2, ..., C_K\}$

Compute the firsts K eigenvectors $U = \{u_1, u_2, ..., u_K\} \in \mathbb{R}^{n \times K}$ of $\tilde{\mathcal{L}}$.
Compute $\tilde{U}$ by normalising the rows to norm 1.
Do K-Means on $\tilde{U} \in \mathbb{R}^{n \times K}$ such that your data points are the rows vectors which have K-dimensions or simply: $\mathcal{D} \leftarrow \tilde{U}^T$.
Pick K random samples as the cluster centroids μ_k.
repeat
  for $i = 1$ **to** n **do**
    $c(x_i) = \min_{k \in K} \|x_i - \mu_k\|_2^2$ %Assign points to clusters
  end
  for $k = 1$ **to** K **do**
    $\mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$ %Update the cluster centroid
  end
until *convergence*;

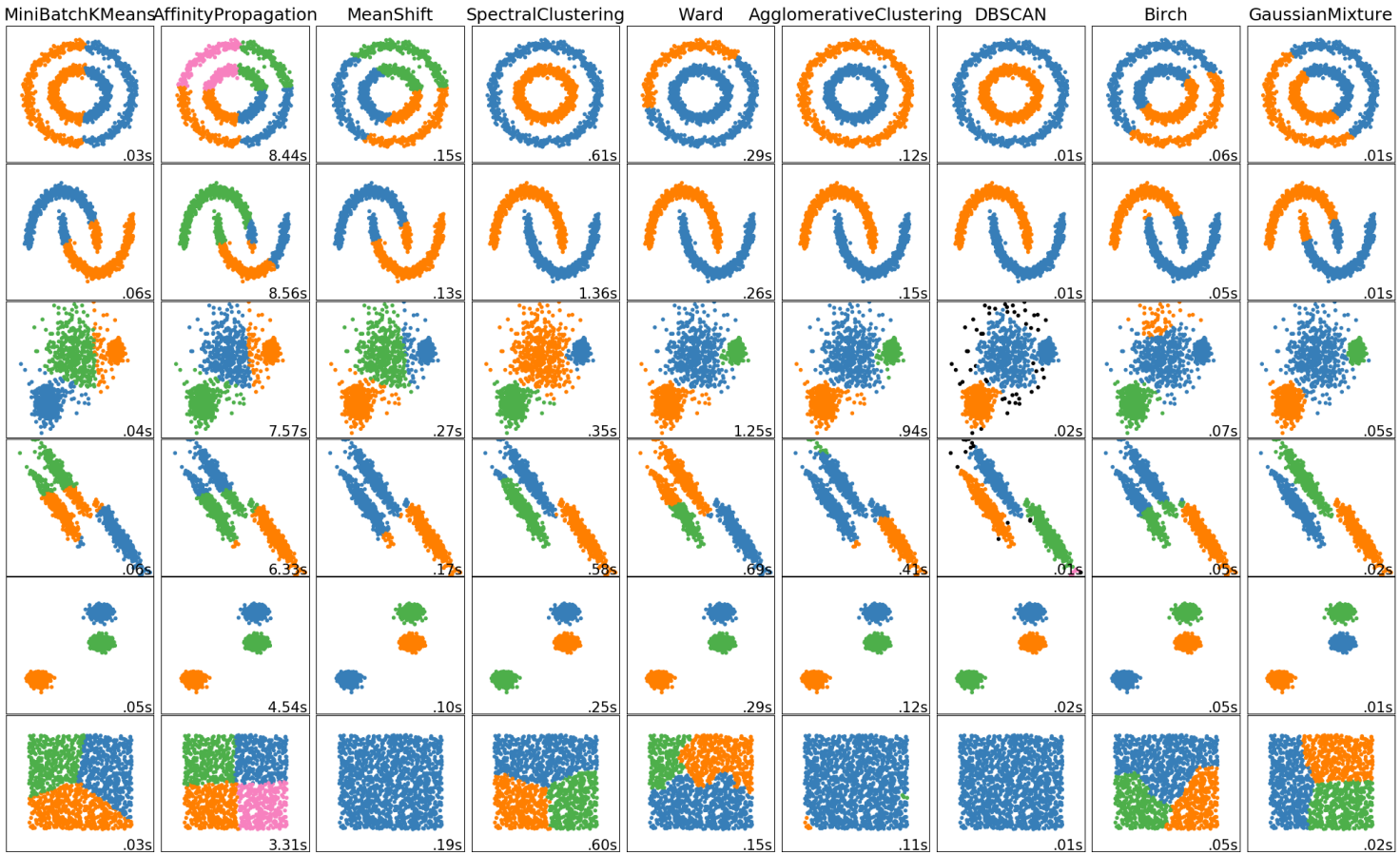
Comparison

Algorithm	Data Rep.	Comp.	Out.	Cent.
K-Means	Vectors	Low	No	$\notin \mathcal{X}^T$
K-Medians	Vectors	High	No	$\notin \mathcal{X}^T$
K-Medoids	Similarity	High	Yes	$\in \mathcal{X}^T$
Kernel K-Means	Kernel	High	N/A	$\notin \mathcal{X}^T$
Spectral Clustering	Similarity	High	N/A	$\notin \mathcal{X}^T$

Data Rep: Data Representation, **Comp.:** Computational cost, **Out.:** Handling outliers, **Cent.:** Centroids.

Comparison

Comparison



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

Parametric, model-based clustering

Parametric: K and the density function are defined (i.e. Gaussian)
Model-based: It is soft-clustering based on the mixture density $f(x)$.

$$f(x) = \sum_{k=1}^{K} \pi_k f_k(x), \quad s.t. \quad \pi_k \geq 0, \sum_{K} \pi_k = 1,$$

where $f_k(x)$ is the component of mixture. $f(x)$ is a Gaussian Mixture Model (GMM) when $f_k(x) \sim \mathcal{N}(x; \mu_k, \sigma_k^2)$.
Degree of Membership:

$$\gamma_{ki} = P[x_i \in C_k] = \frac{\pi_k f_k(x_i)}{f(x_i)}$$

GMM Parameter: $\theta = \{\pi_{1:K}, \mu_{1:K}, \sigma_{1:K}\}$.
Selected Algorithm to estimate the parameter: EM-Algorithm.

Expectation-Maximization (EM) Algorithm

- Given data points $\mathcal{X}^T$ sampled i.i.d from an unknown distribution f
- We need to model the distribution using Maximum Likelihood (ML) principle (log-likelihood):

$$l(\theta) = \ln f_\theta(\mathcal{X}) = \sum_{i=1}^{N} \ln f_\theta(x_i)$$

$$l(\theta) = \sum_{i=1}^{N} \ln \sum_{k=1}^{K} \pi_k f_k(x_i)$$

The objective:
$\theta^{ML} = argmax_\theta l(\theta)$

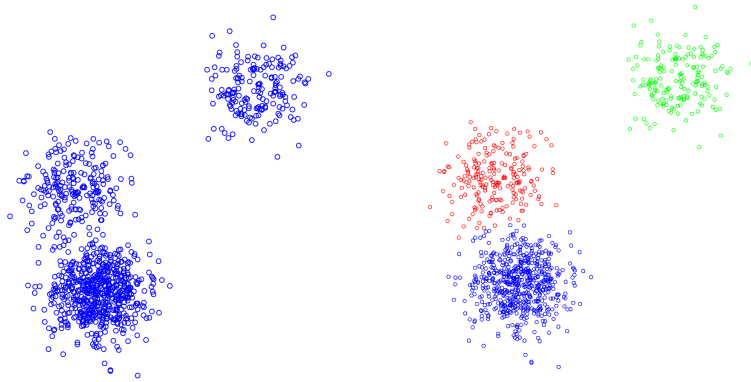


Figure: GMM Clustering

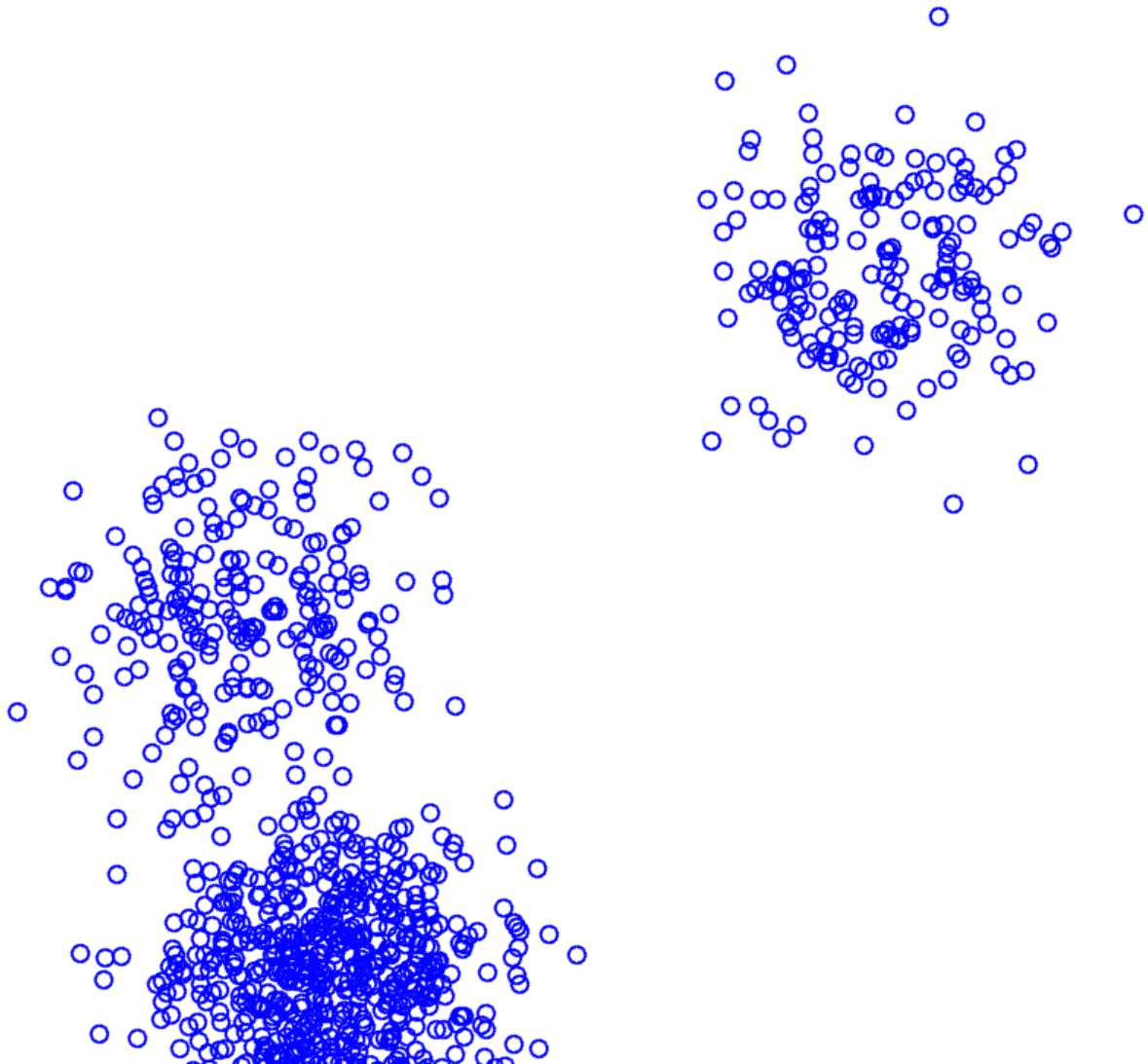
EM – Algorithm

input : data points $\mathcal{X}^T$, number of clusters K
output: Parameters, $\theta^{ML} = \{\pi_{1:K}, \mu_{1:K}, \sigma_{1:K}\}$

Initialize the parameters θ at random.
repeat
  for $i = 1$ **to** N **do**
    for $k = 1$ **to** K **do**
      $\gamma_{ik} = \frac{\pi_k f_k(x_i)}{f(x_i)}$ %E-Step
    end
  end
  for $k = 1$ **to** K **do**
    $\pi_k = \frac{1}{N} \sum_{i=1}^{N} \gamma_{ik}$ %M-Step
    $\mu_k = \frac{1}{N\pi_k} \sum_{i=1}^{N} \gamma_{ik} x_i$
    $\sigma_k = \frac{1}{N\pi_k} \sum_{i=1}^{N} \gamma_{ik} (x_i - \mu_k)(x_i - \mu_k)^T$
  end
until *convergence*;

Mixture Models

Gmms in action



Selecting K

Determining the number of clusters K is a model selection problem that can be answered depending on the application

- quantization: rate/distortion tradeoff
- density estimation: bias/variance tradeoff
- cluster analysis: data fitness

Data fitness

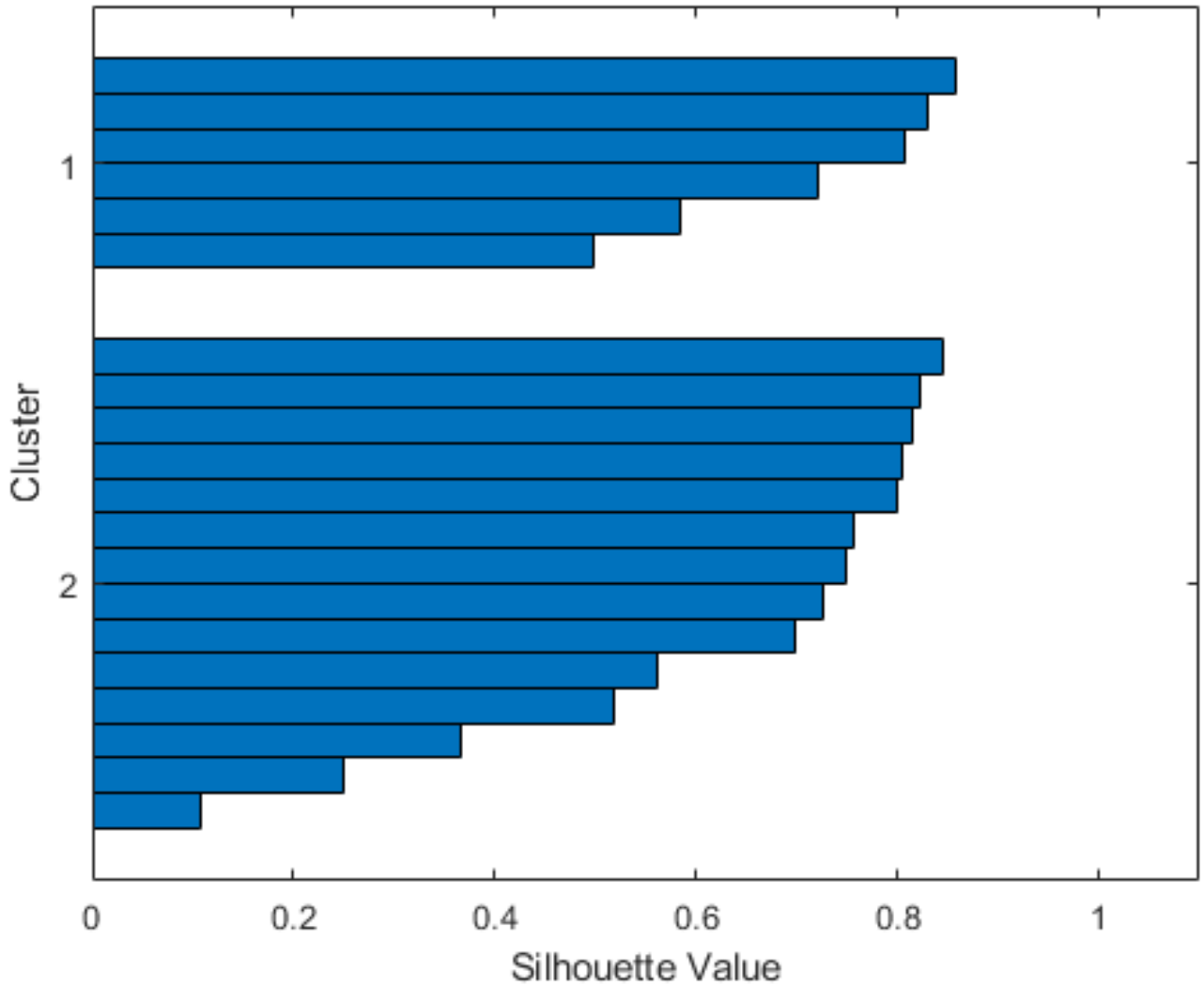
- Many indicators can be considered to estimate the goodness of fit.
- But it is usually make sense to consider the one considered during the clustering.
- Usually some sort of average intra cluster distance
- The problem with this kind of criterion is that the average intra class distance $D_K(X)$ over the dataset X decreases monotonously with K.
- need for some sort of normalization.

The Silhouette

The silhouette value for each point is a measure of how similar that point is to points in its own cluster, when compared to points in other clusters.

- The silhouette value for the ith point, S_i, is defined as $S_i = \frac{(b_i - a_i)}{max(a_i, b_i)}$
- where a_i is the average distance from the ith point to the other points in the same cluster as i,
- and b_i is the minimum average distance from the ith point to points in a different cluster, minimized over clusters.

Silhouette



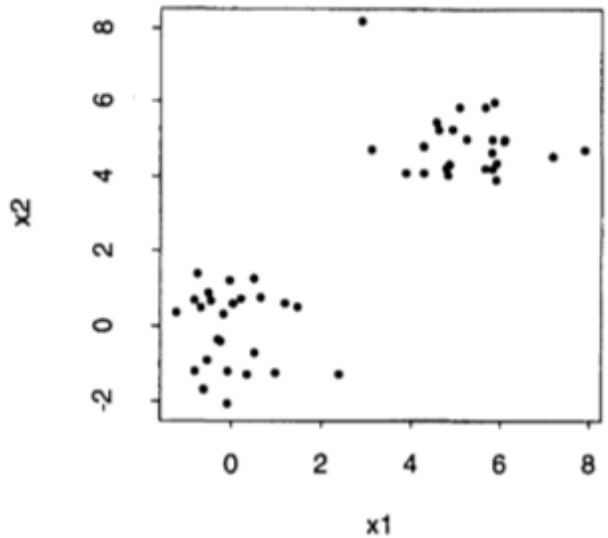
The gap statistic

The idea of the gap statistic

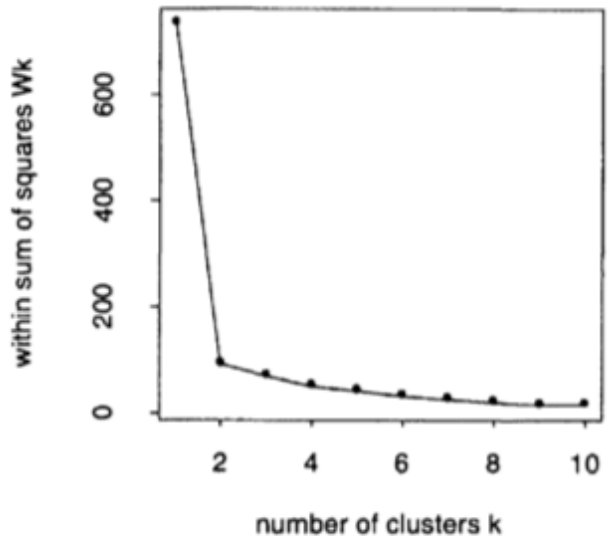
- is to standardize $D_K(X)$
- by comparing it with is expectation under an appropriate null reference of the distribution of the data.
- $$Gap_K(X) = \frac{D_K(X)}{D_K(null)}$$

Tibshirani & al Estimating the number of clusters in a data set via the gap statistics 2001

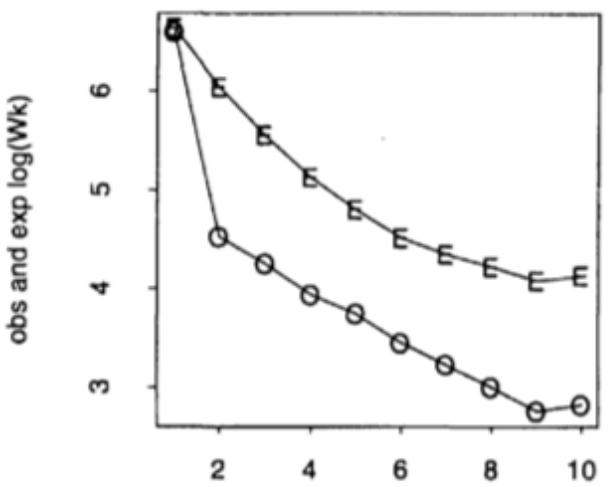
The gap statistic

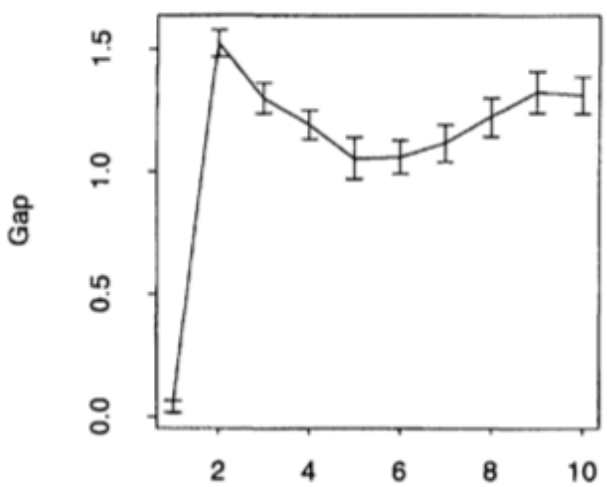


(a)



(b)





The gap statistic

Selecting the right null distribution can be of importance.

- plain: sample uniformly over the range of observed data
- pca-projected: project the uniform samples over the dataset distribution using pca projection.

Acknowledgment

This course is based on

- slides of Shadi Albarqouni