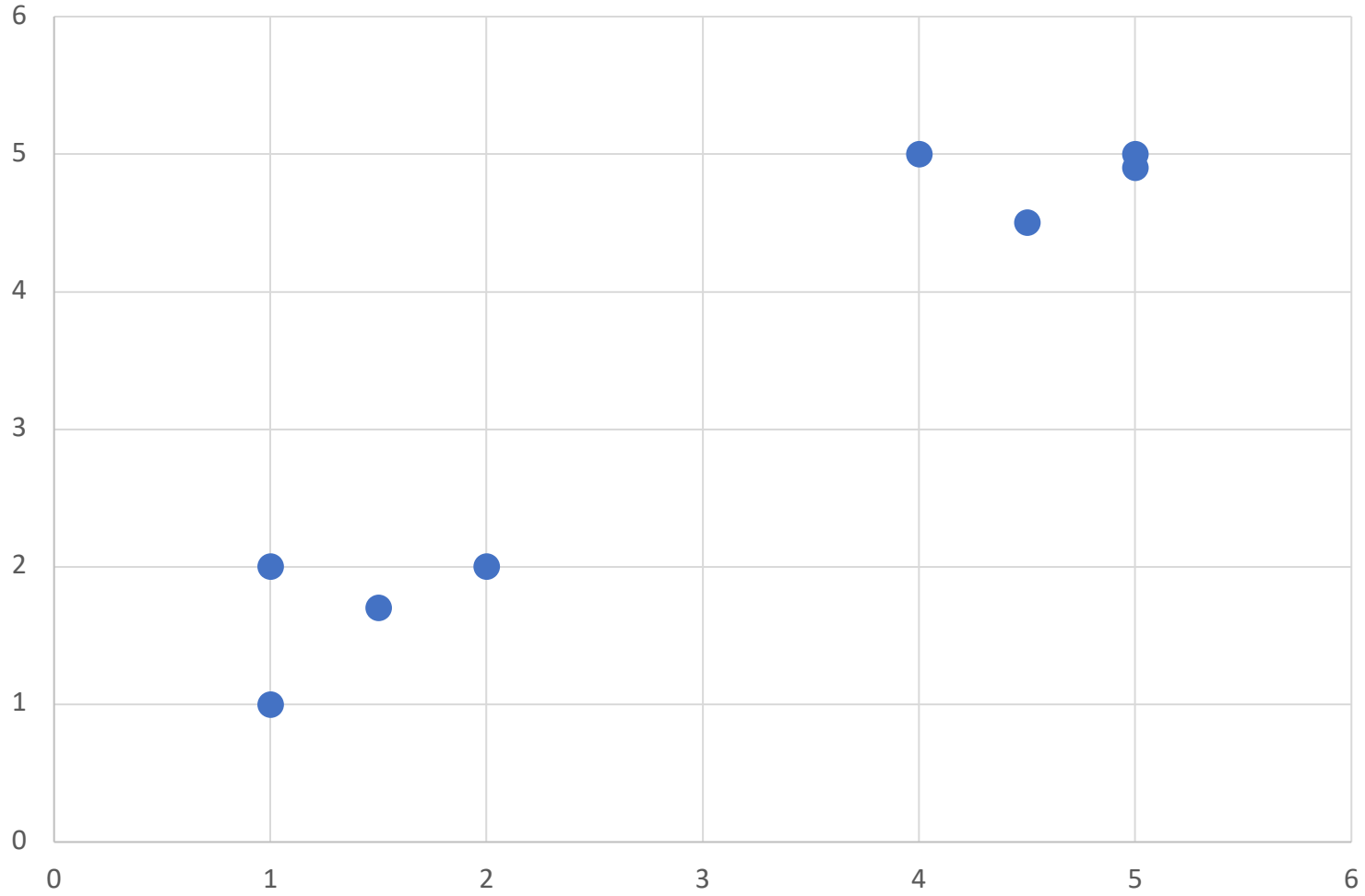
Politecnico di Bari

Unsupervised Learning

Introduction

not interested on label of dataset but on the dataset itself

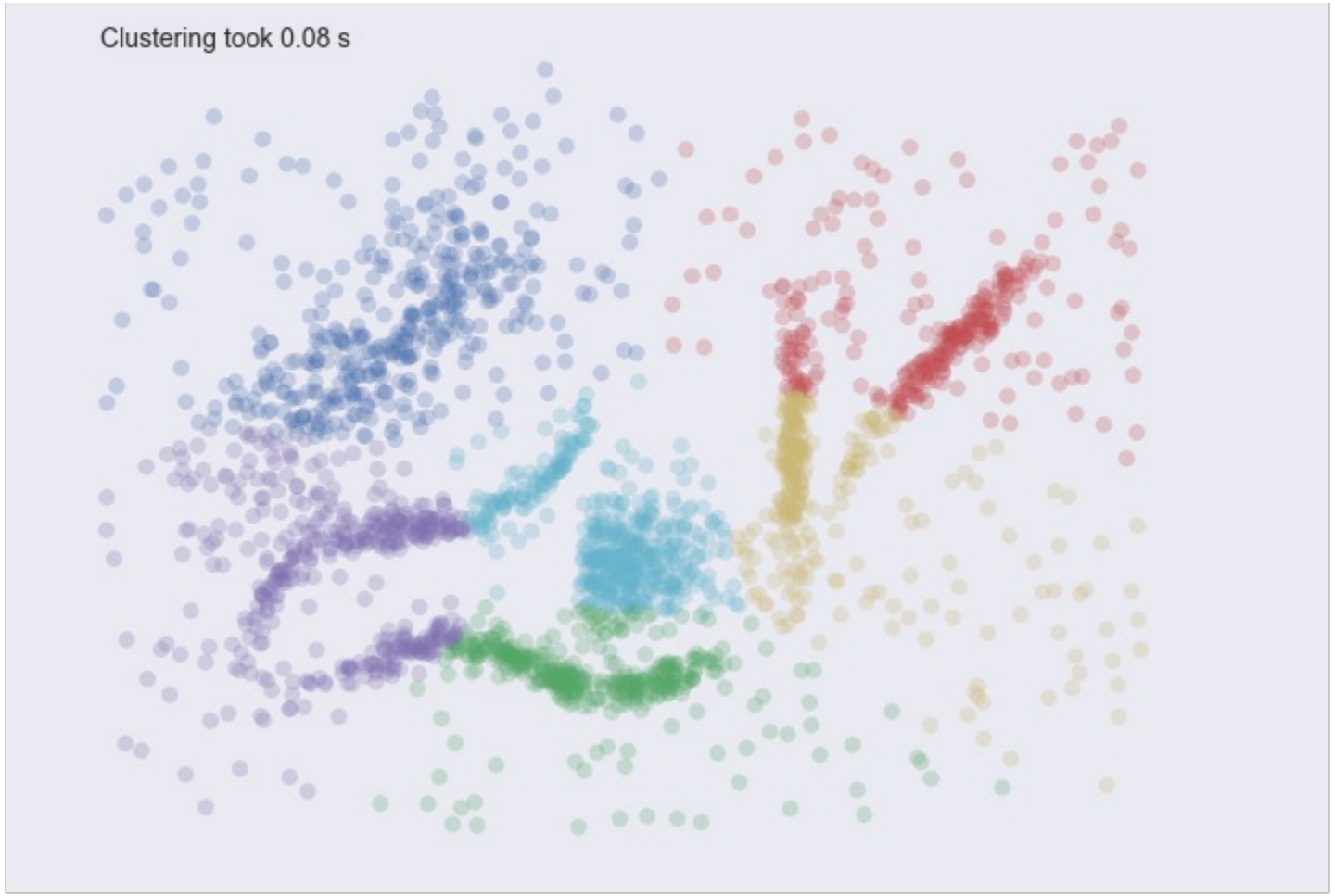
Unsupervised Learning



which datapoints serve the same purpose, we dont care about caratteristic but on datapoints

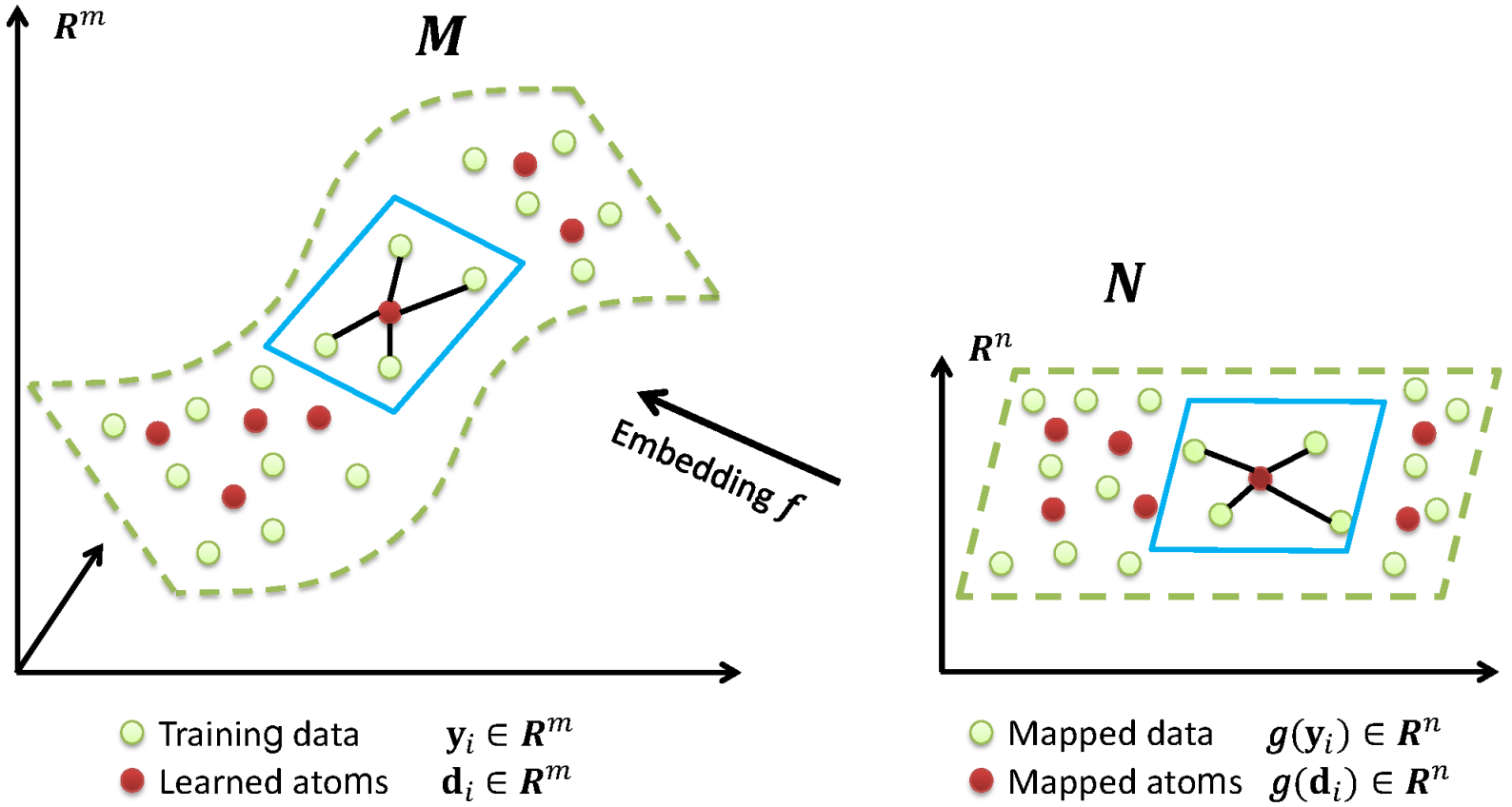
Unsupervised Learning

Unsupervised Learning



we will never know the optimal cluster

Unsupervised Learning



dimensionlaity reduction

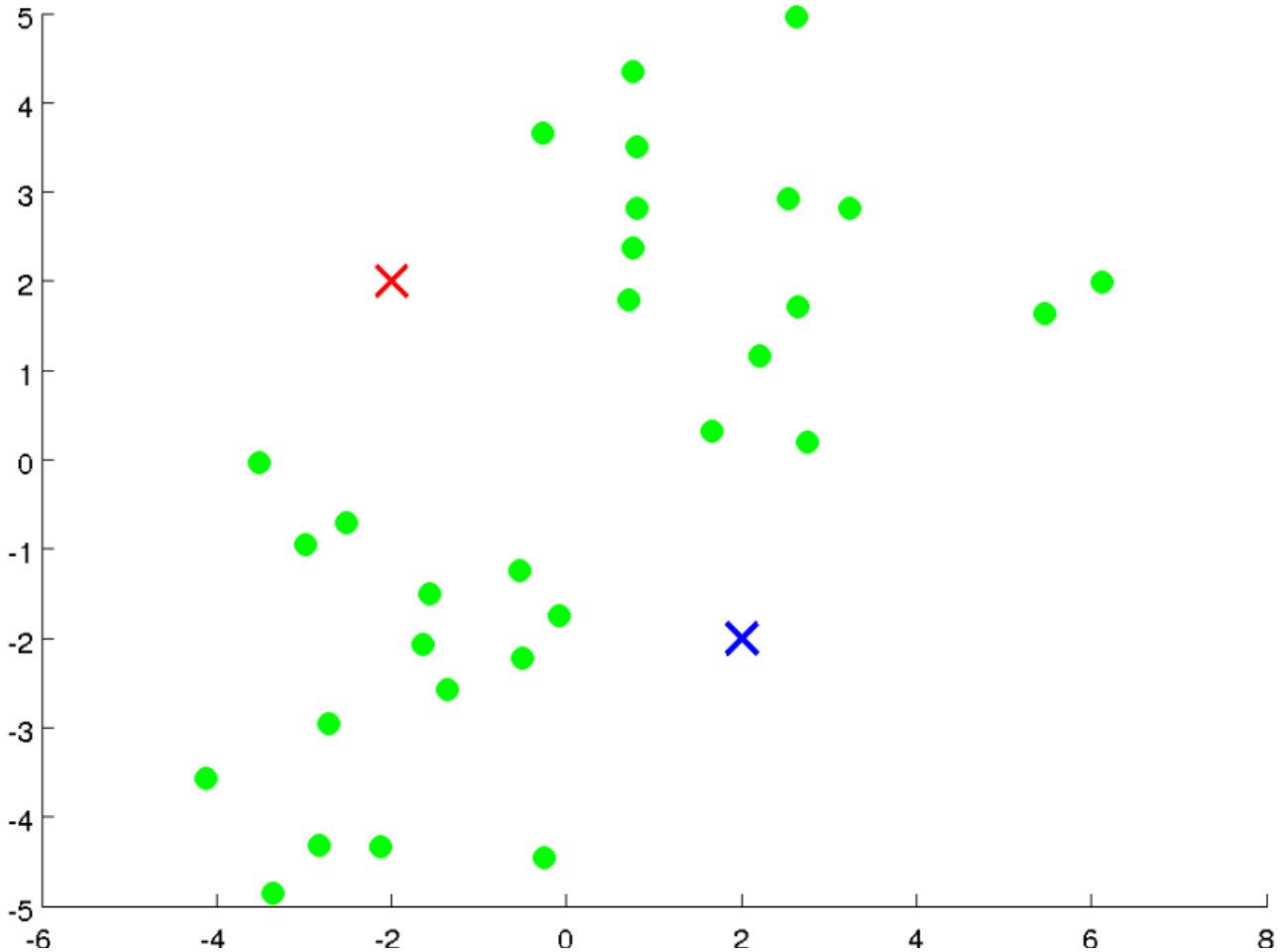
Politecnico di Bari

Clustering

K-means

most simple algo

K-means

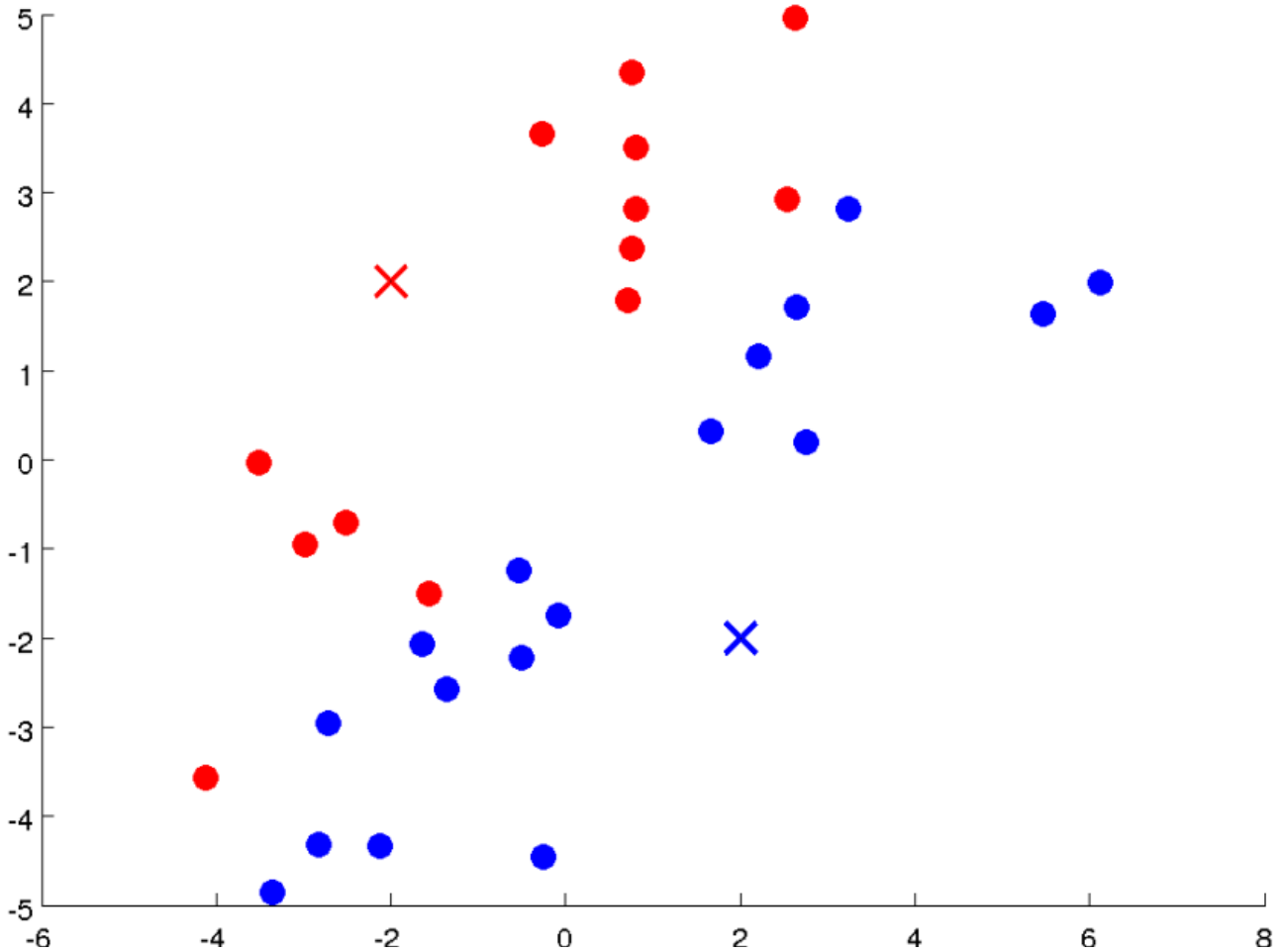


K-means

we choose to have 2 cluster.
practical decision, k=2

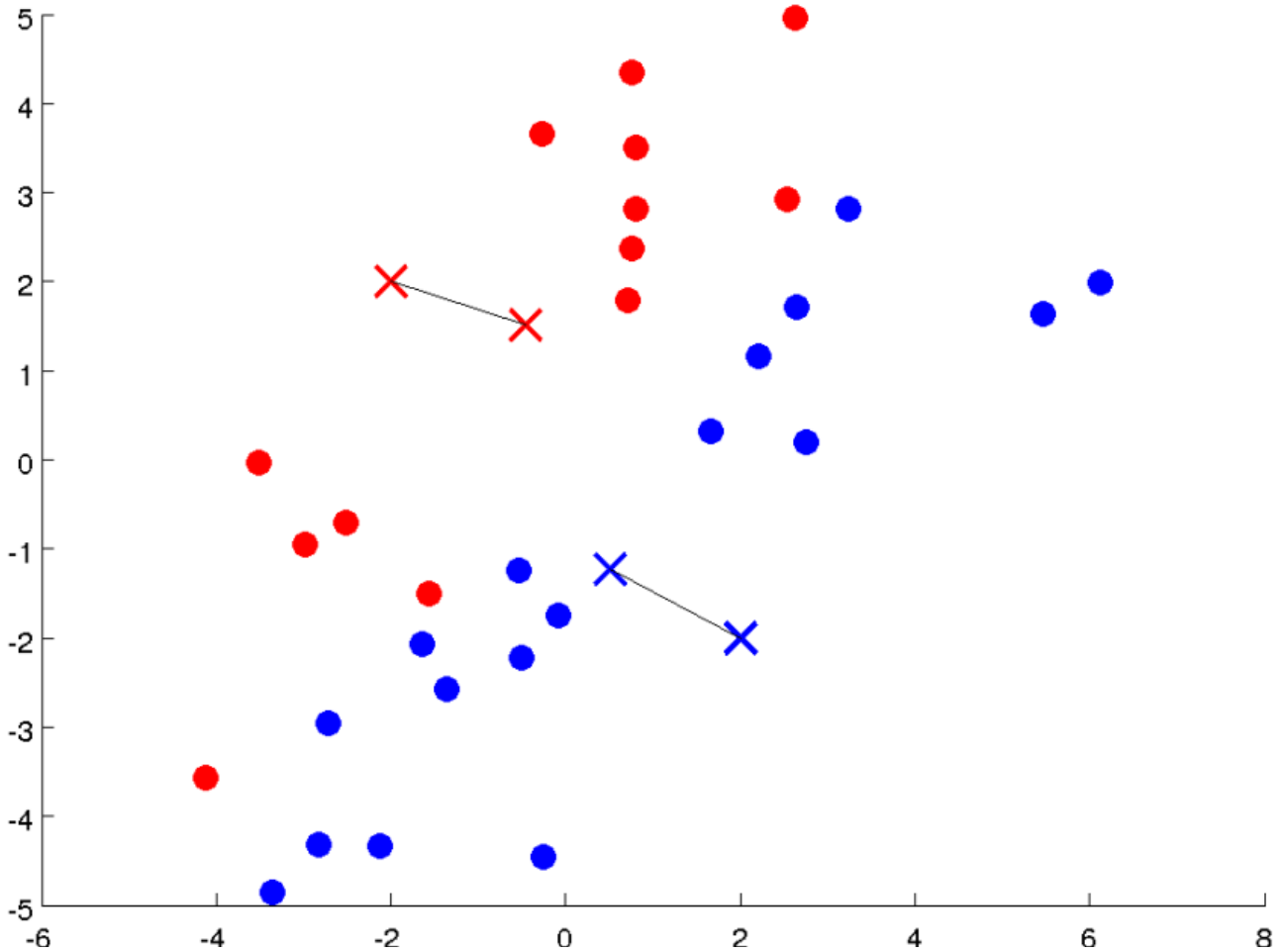
we try to find the centroid right in the middle of the classes.

we start from a random centroid "X".
the closer point to red cross belong to red class



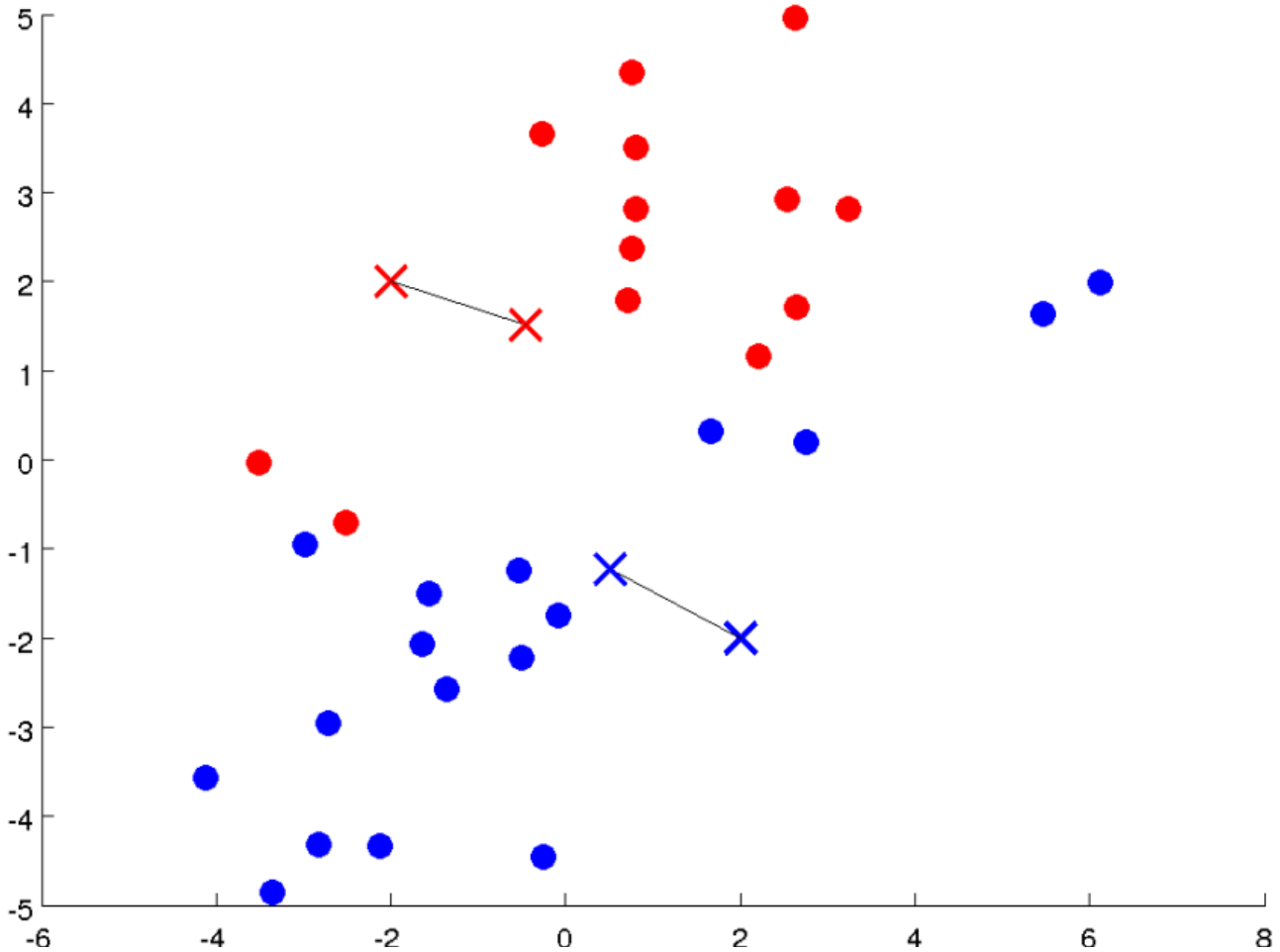
K-means

now i compute the REAL centroid



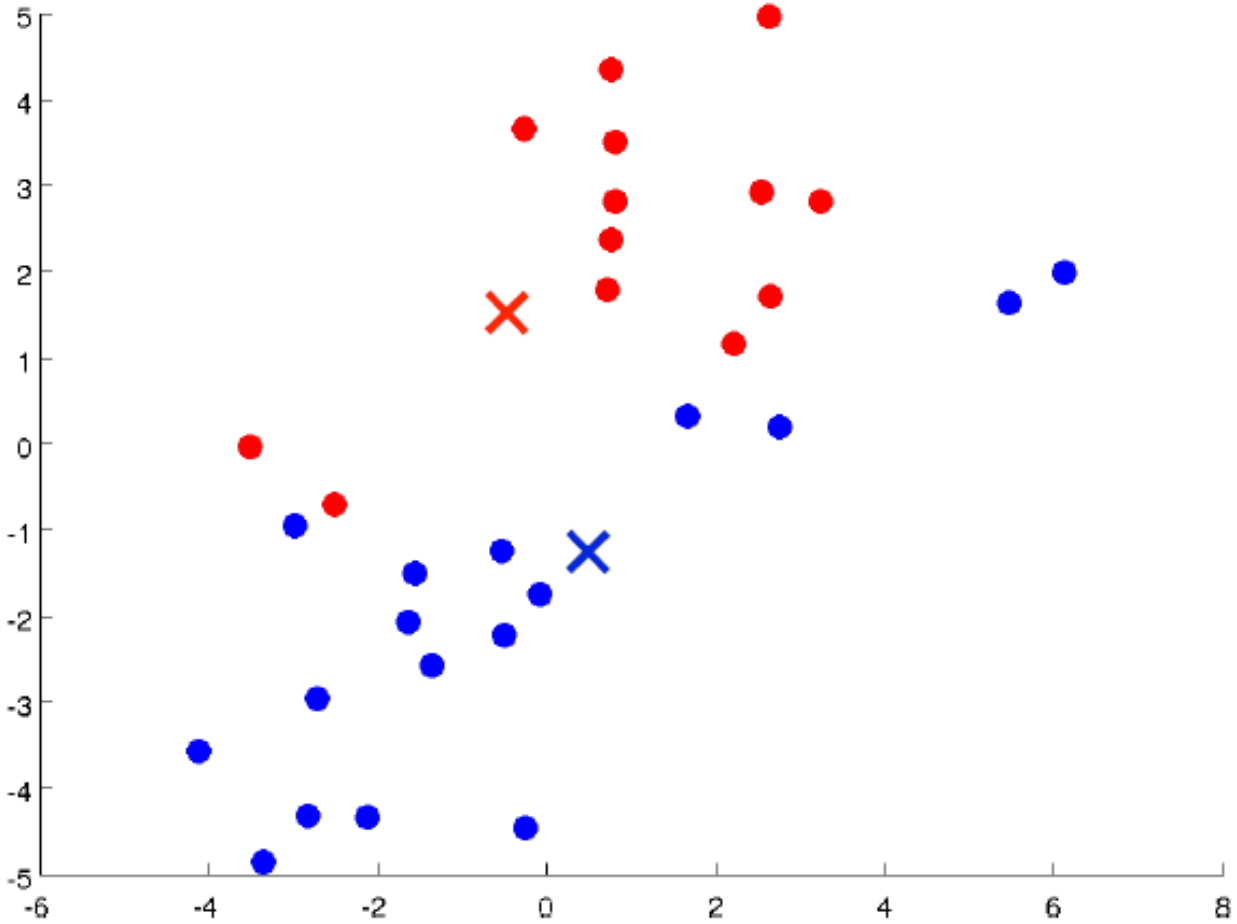
K-means

let me compute again the cluster based on new centroid

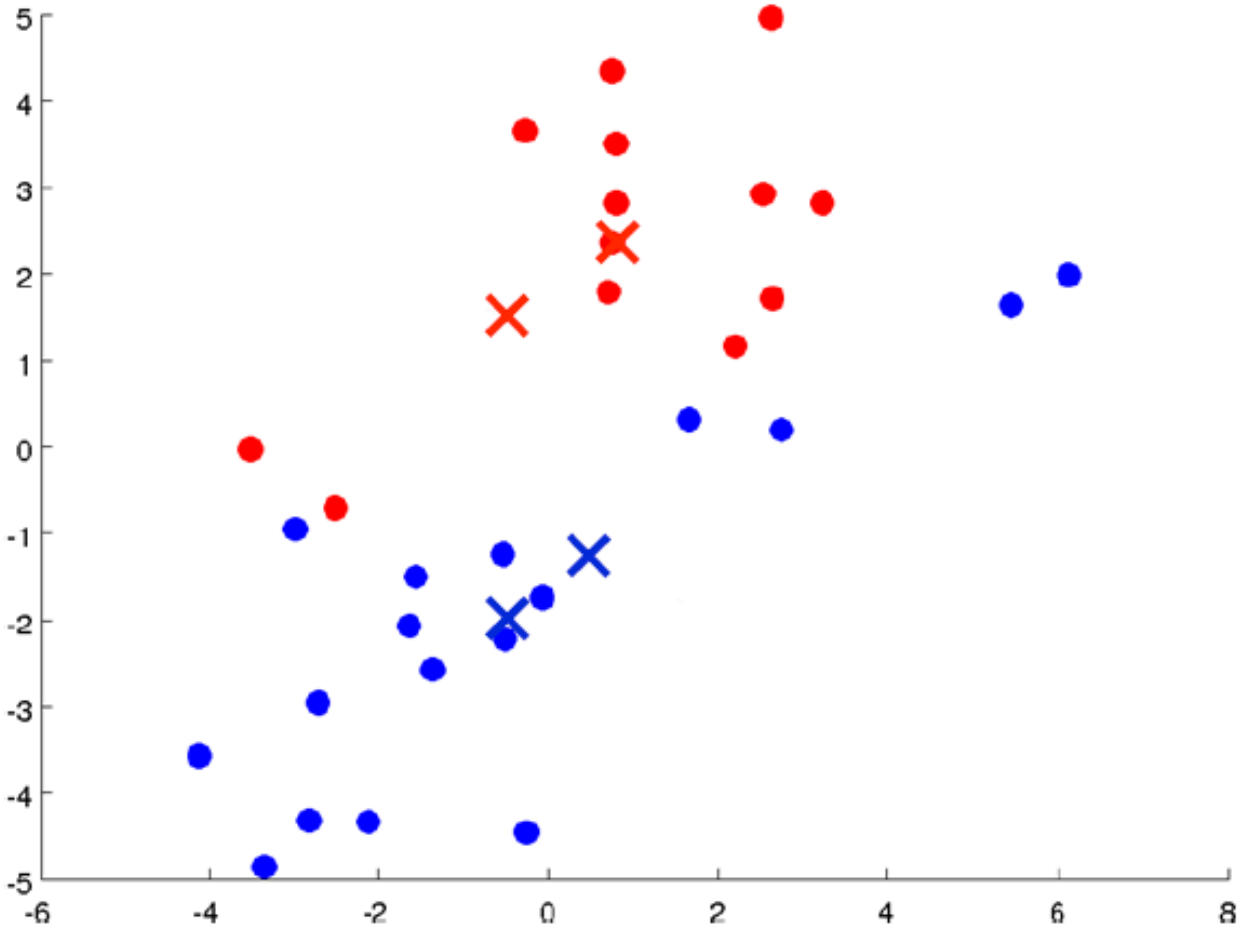


K-means

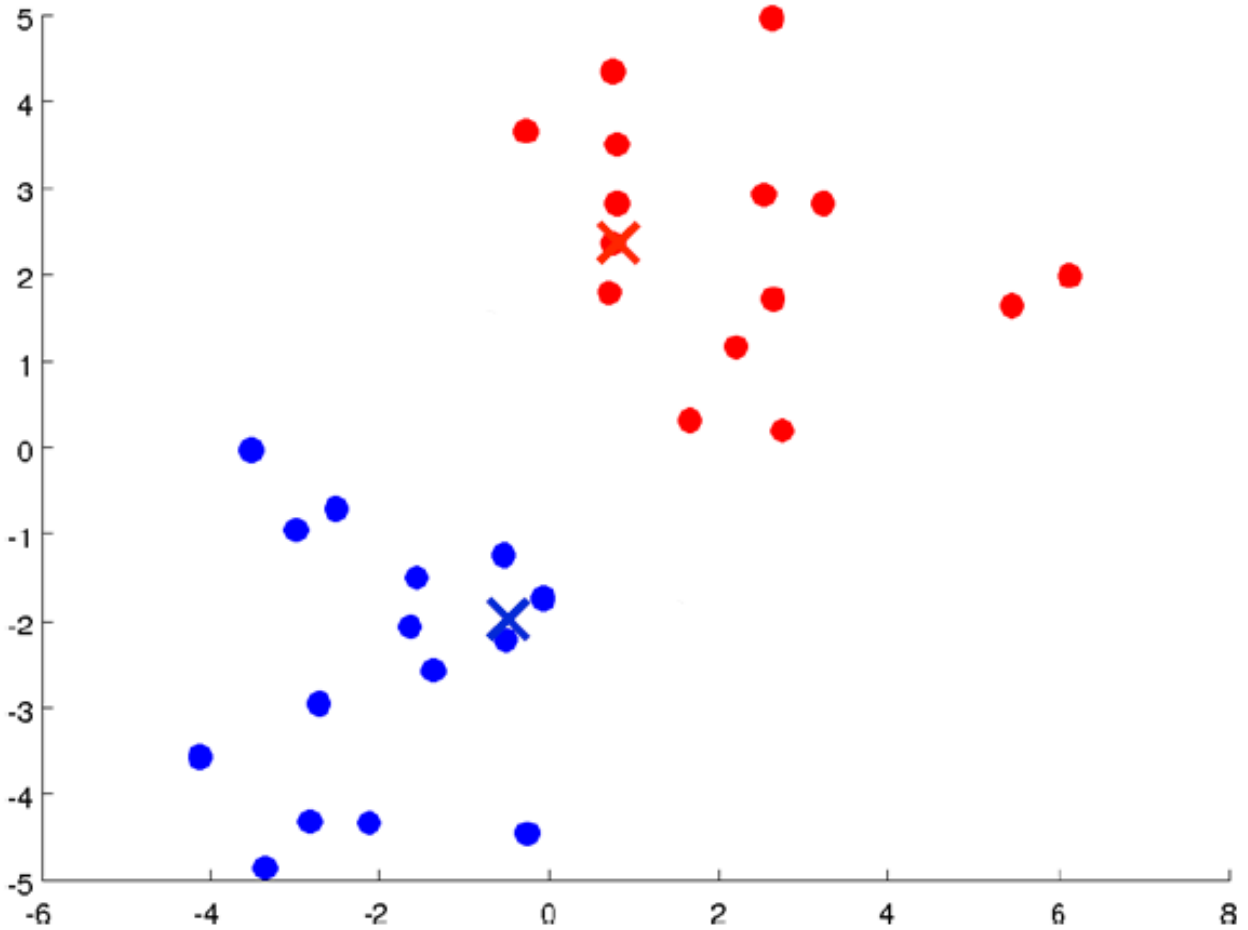
compute again the new centroid and again again until i don't have new changes, converges



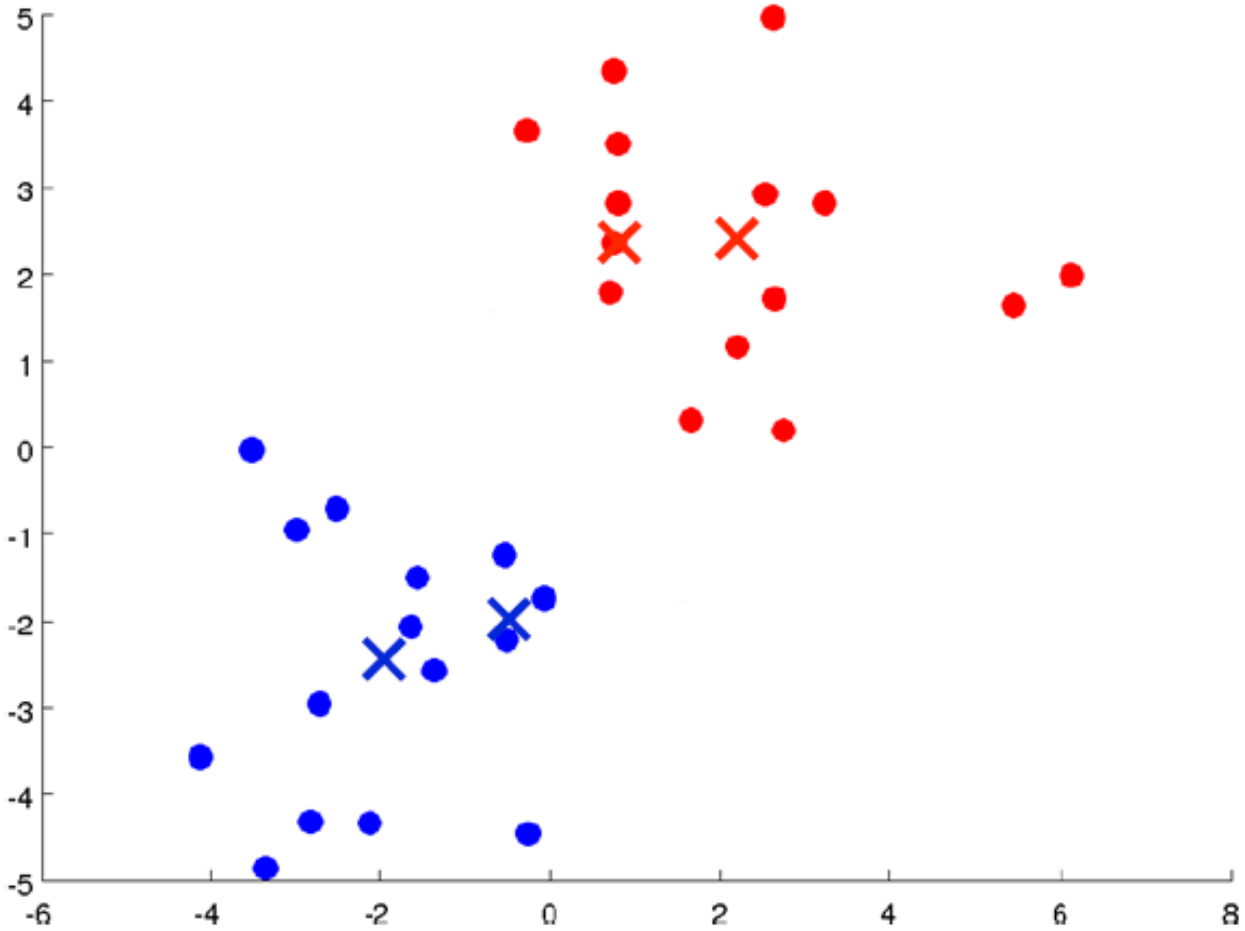
K-means



K-means

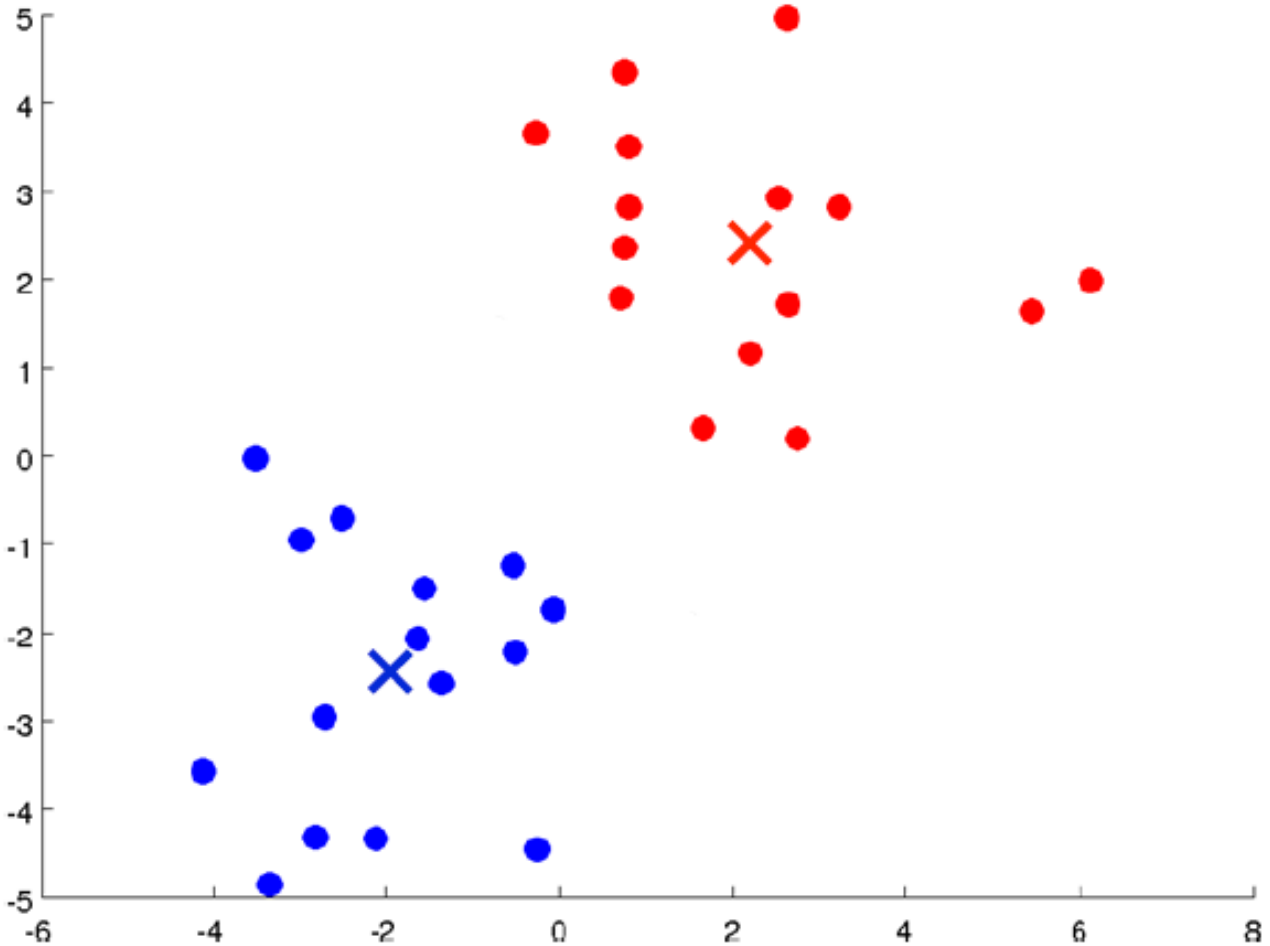


K-means



K-means

final cluster



K-means algorithm

notion of loss

$$L = \sum_{k=1}^{K} \sum_{i=1}^{m} a_{ik} \left\| x^{(i)} - \mu_k \right\|^2$$

likelihood ang gaussian hypothesis

Minimize L with respect to a and μ following these two steps:

[Expectation] Choose optimal a for fixed μ by assigning $x^{(i)}$ to the nearest μ_k

$$a_{ik} = \begin{cases} 1 if\ k = \arg\min_l \left| x^{(i)} - \mu_l \right|^2 \\ 0\ otherwise \end{cases}$$

probability of a datapoint belonging to a cluster, in our case p=0 or 1

[Maximization] Choose optimal μ for fixed a by updating μ_k to be the empirical mean of the points assigned to each cluster

update the centroid

$$\mu_k = \frac{1}{n_k} \sum_{i: x_i \in C_k} x^{(i)} \qquad \text{where } n_k = \sum_{i=1}^{m} a_{ik} \text{ (number of data points in the k-th cluster } C_k)$$

we fix mu and then we compute mu, and then we fix a and recompute mu until convergence

K-means algorithm

Randomly initialize K cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$

Repeat {

for i=1 to m

$c^{(i)}$:= index of cluster centroid (from 1 to K) closest to $x^{(i)}$

for k = 1 to K

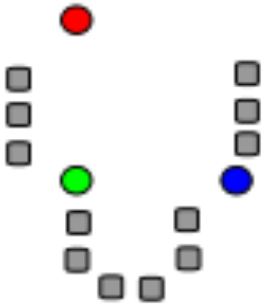
μ_k := mean of points assigned to cluster k

}

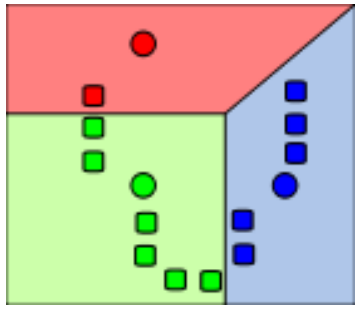
$$\mu_k = \frac{1}{|C_k|} \sum_{x^{(i)} \in C\_k} x^{(i)} \qquad x^{(i)} \in \mathbb{R}^n$$

K-means

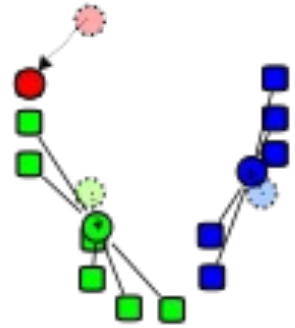
Choice randomly k centroids



Assignment to the clusters

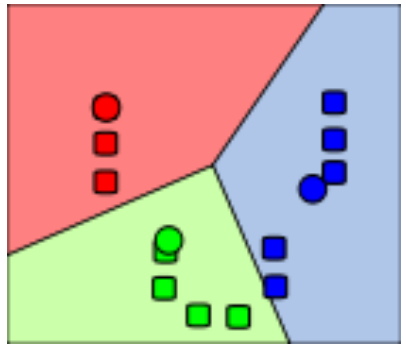


Recompute centroids



iterations

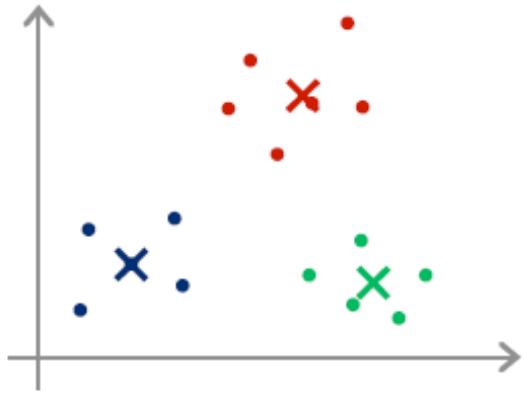
Clusters found after the convergence



Random initialization

- We should have $k < m$
- Randomly pick K training examples
- Set the centroids equal to these examples

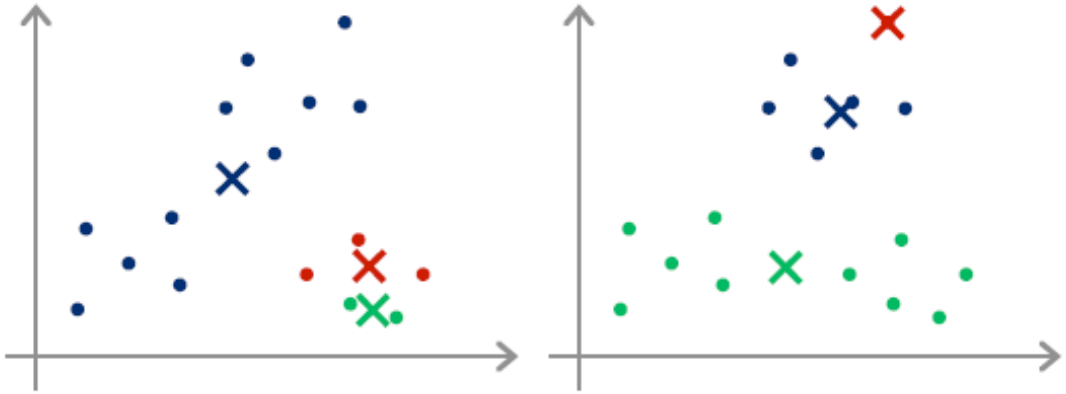
K-means – local minima



we want to minimize this

$$J(c^1, \ldots, c^m, \mu_1, \ldots, \mu_K) = \frac{1}{m} \sum_{i=1}^{m} \left\| x^i - \mu_{c^i} \right\|^2$$

depending on the way i select the initial random centroid i can endup in alocal minimu of function, so have different result



K-means – local minima

For i=1 to 100 {

Randomly initialize K-means

Run K-means. Get $c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_k$

Compute $J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_k)$

}

Choose clustering that gave the lowest cost $J\left(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K\right)$

K-means – PROS/CONS

PROS

- Quite simple and computationally efficient.
- Always terminates

CONS:

- Finding the optimal clustering is not guaranteed
- Not applicable to categorical values (non-numeric)
- Requires to specify K
- Sensible to noise and outliers

Politecnico di Bari

Clustering

K-medoids

as k means, but the centroid is always selected as the closest point from the centroid we have computed

Partitioning Around Medoids (PAM)

1. Randomly initialize K cluster medoids μ^k
2. $M = \bigcup_{k=1}^{K} \mu^k$
3. Associate each $x^{(i)} \in X$ to the closest medoid k
4. $Cost$ = sum of the distances of samples from their medoids
5. $NewCost = 0$
6. **while**$(NewCost \leq Cost)$ {
    for k = 1 **to** K
        for $x^{(i)} \in X - M$
            swap $x^{(i)}$ and μ^k
            repeat steps 2) and 3)
            $NewCost$ = sum of the distances of samples from their medoids
            if $(NewCost > Cost)$ **then** undo swap
            else $Cost = NewCost$
    }

K-Medoids – PROS/CONS

Pros

- The method is less sensitive to outliers than k-means

Cons

- Finding the optimal clustering is not garantueed
- Not applicable to categorical values (non-numeric)
- Requires to specify K
- Yet sensible to noise and outliers
- Fails for non-linear data set
- Computationally more expensive than k-means

Politecnico di Bari

Clustering

Gaussian Mixture Models

Gaussian (Normal) distribution

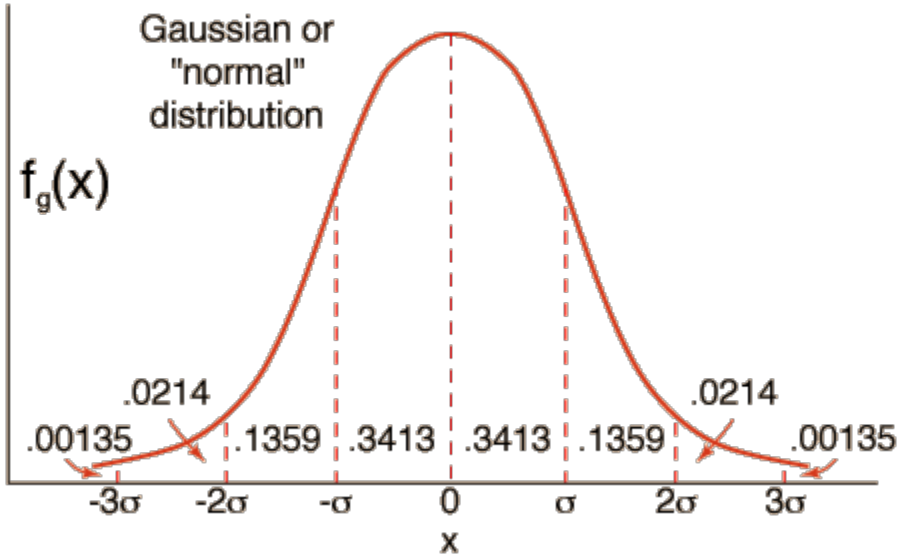
Say $x \in \mathbb{R}$.

x has a Gaussian distribution with mean μ and variance σ^2

$x \sim \mathcal{N}(\mu, \sigma^2)$

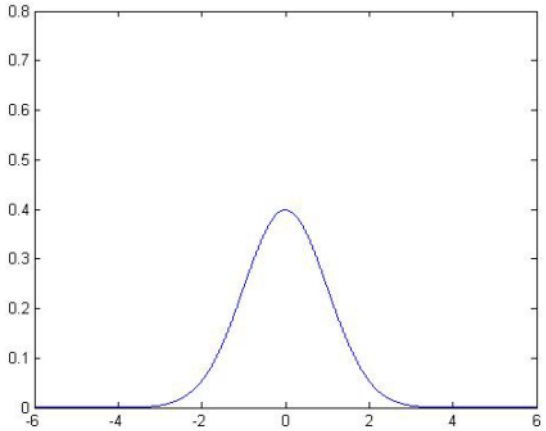
$$p(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

In the picture $\mu = 0$

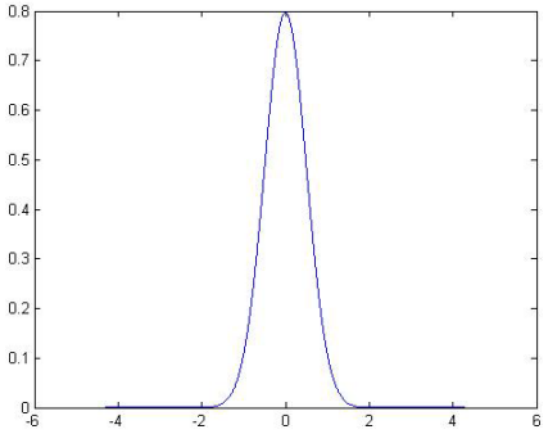


Gaussian distribution example

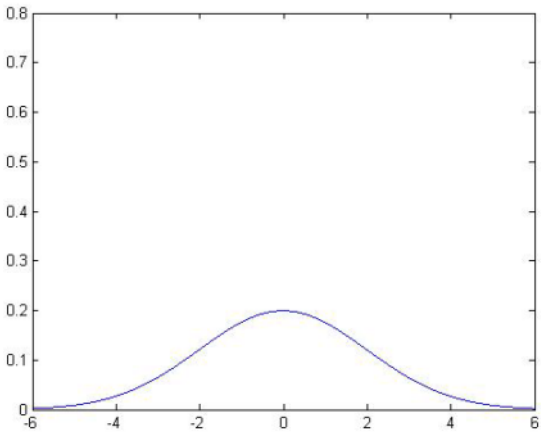
$\mu = 0, \sigma = 1$



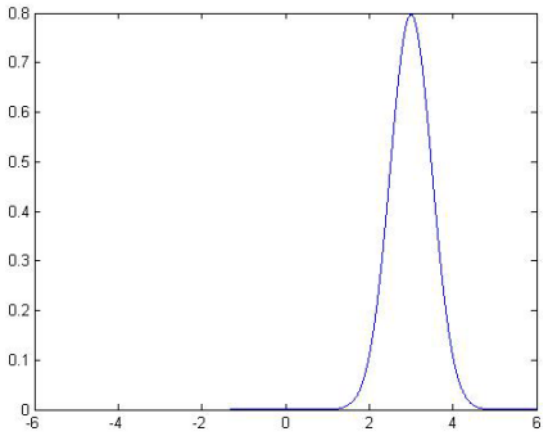
$\mu = 0, \sigma = 0.5$



$\mu = 0, \sigma = 2$

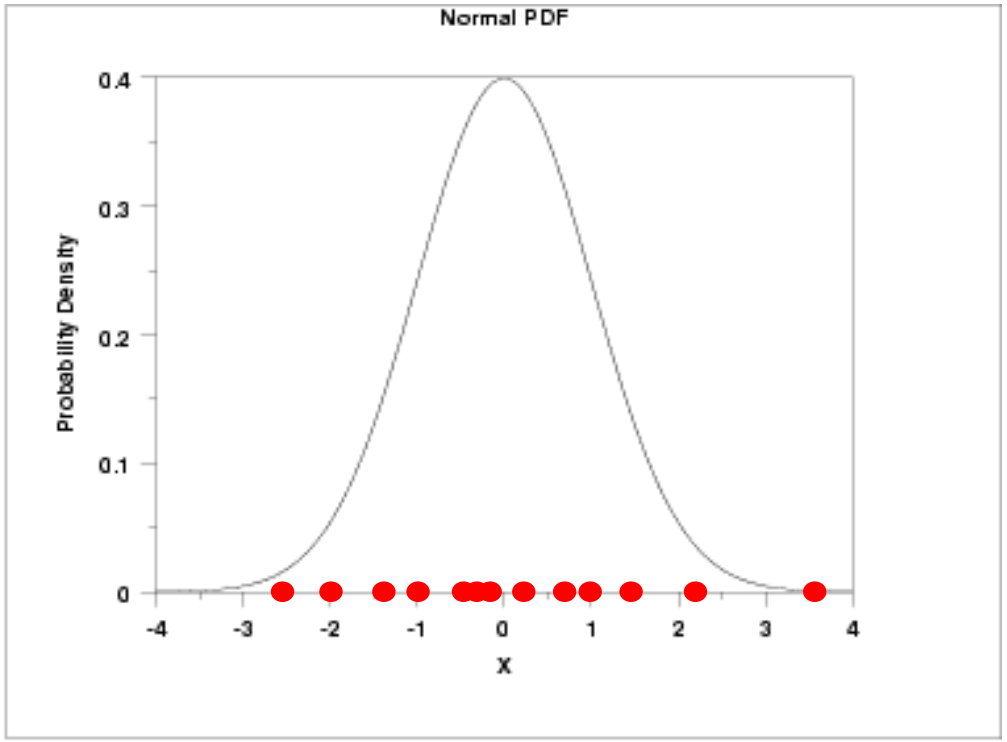


$\mu = 3, \sigma = 0.5$



Parameters estimation

Dataset: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\} x^{(1)} \in \mathbb{R}$



hypotesis of independet and normal distribution

$$x \sim \mathcal{N}(\mu, \sigma^2)$$

$$\mu = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$$

$$\sigma^2 = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)} - \mu)^2$$

Multivariate Gaussian (Normal) distribution

With the previous model we have to model p(x1),p(x2),…,p(xn) separately.

$x \in \mathbb{R}^n$

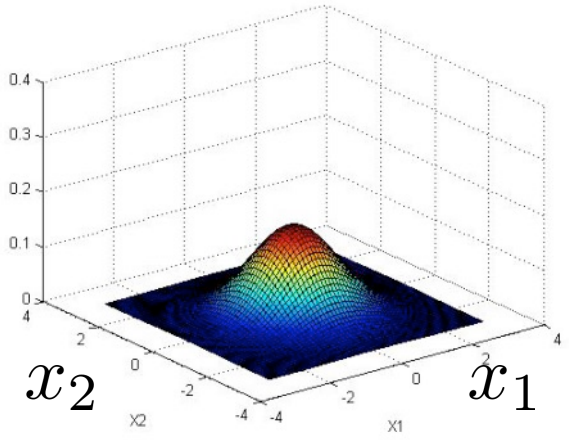
$\mu \in \mathbb{R}^n$

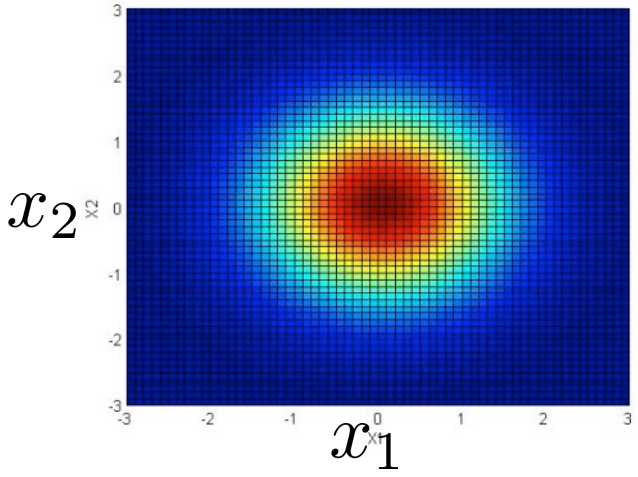
$\Sigma \in \mathbb{R}^{n \times n}$ covariance matrix

$$p(x; \mu, \Sigma) = \frac{e^{\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)}}{\sqrt{(2\pi)^n |\Sigma|}}$$

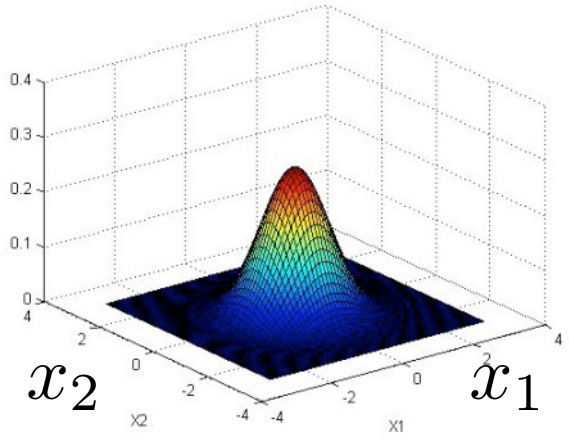
Multivariate Gaussian examples

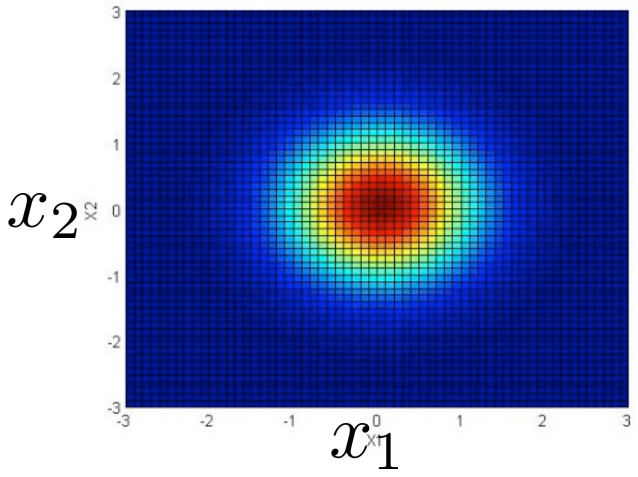
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



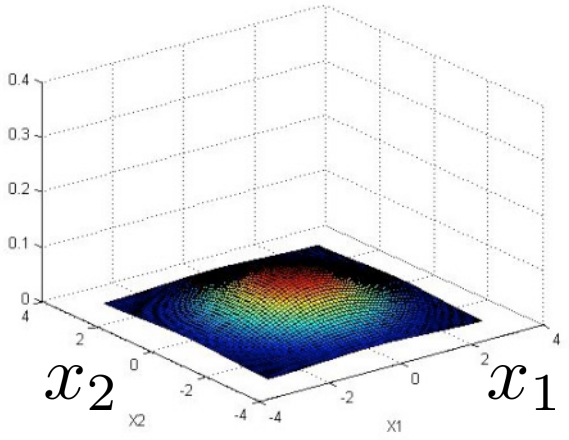


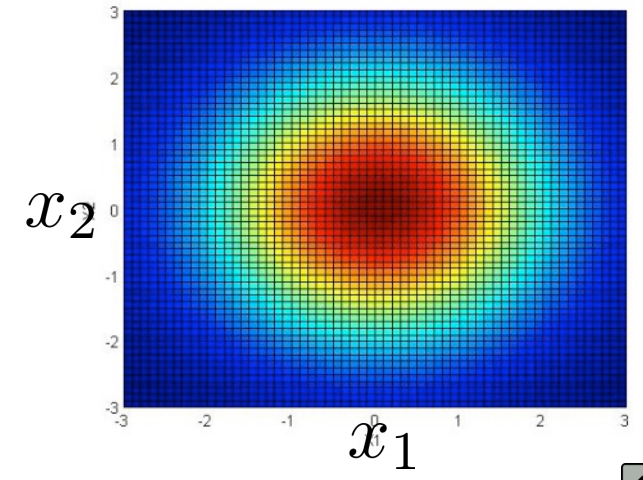
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 0.6 & 0 \\ 0 & 0.6 \end{bmatrix}$$





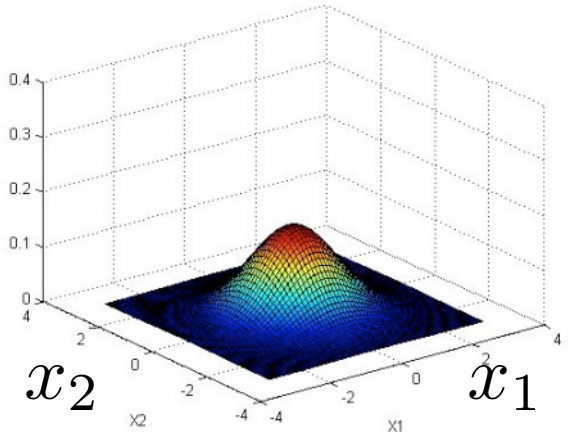
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

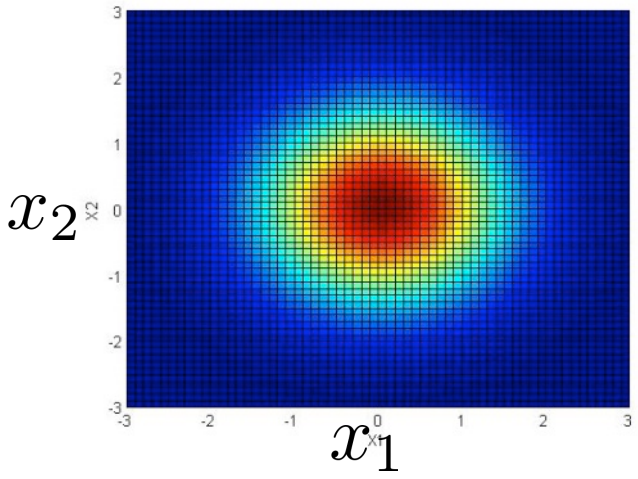




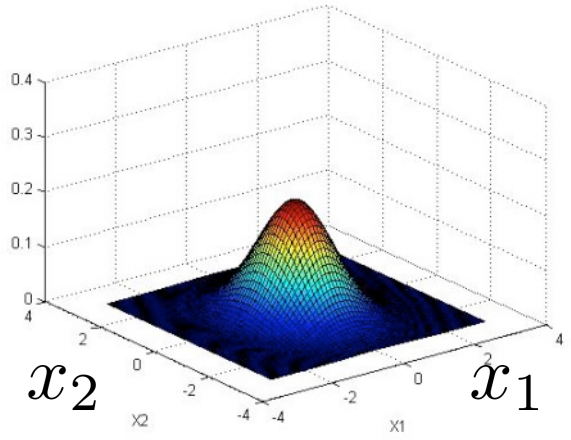
Multivariate Gaussian examples

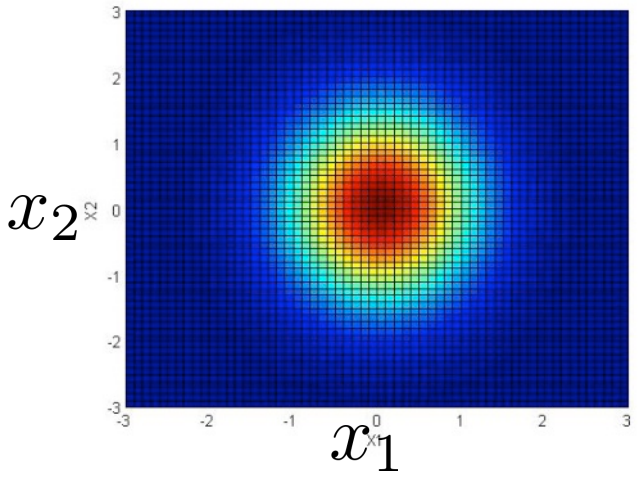
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



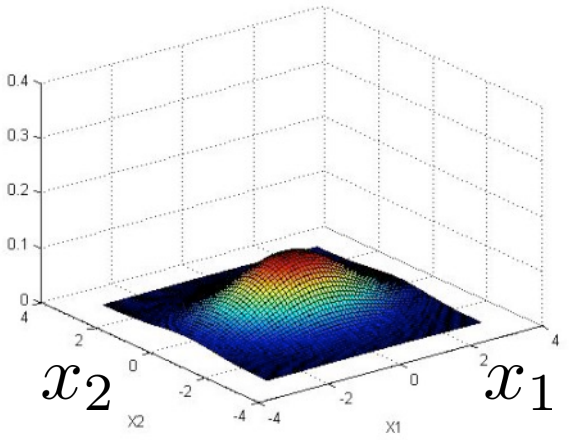


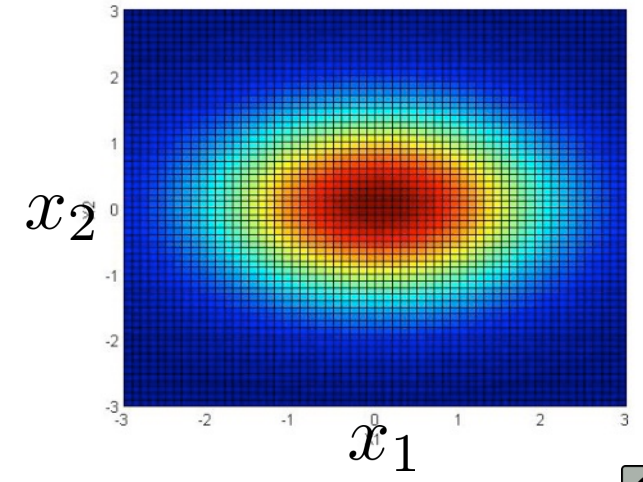
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 0.6 & 0 \\ 0 & 1 \end{bmatrix}$$





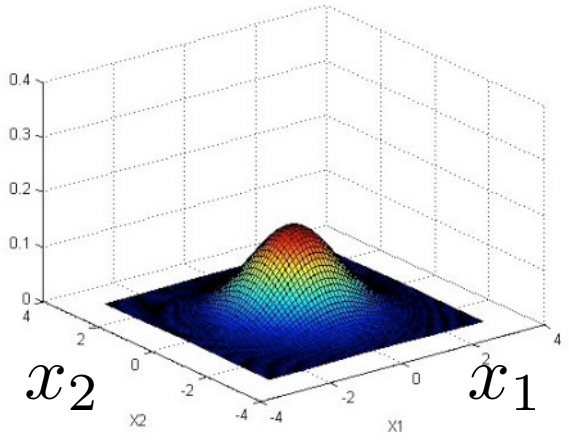
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

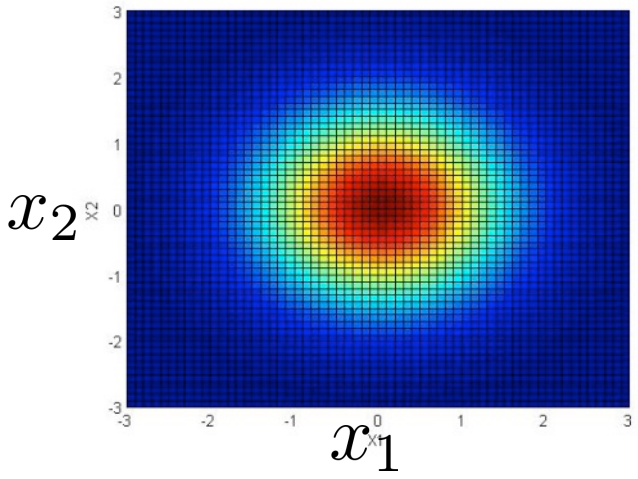




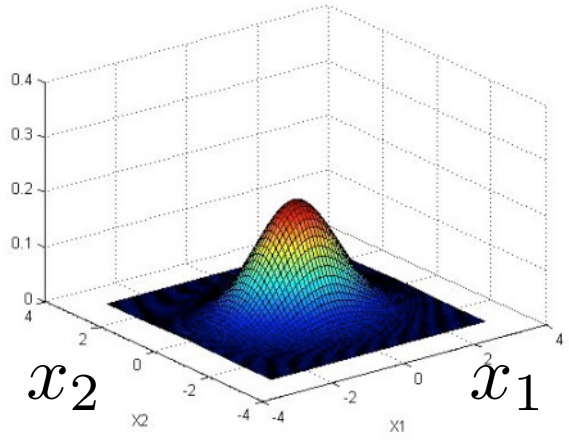
Multivariate Gaussian examples

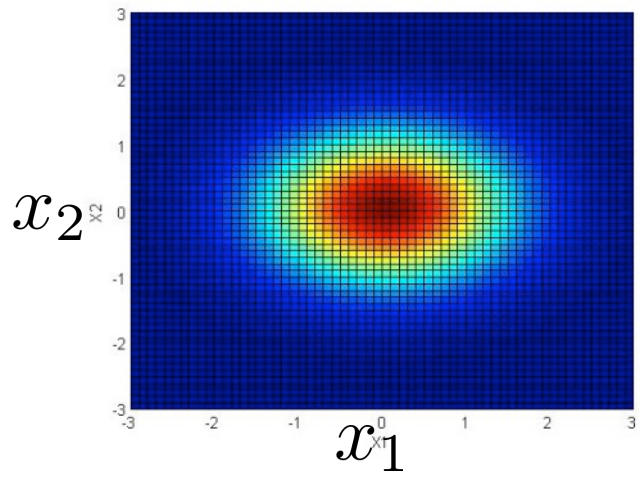
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



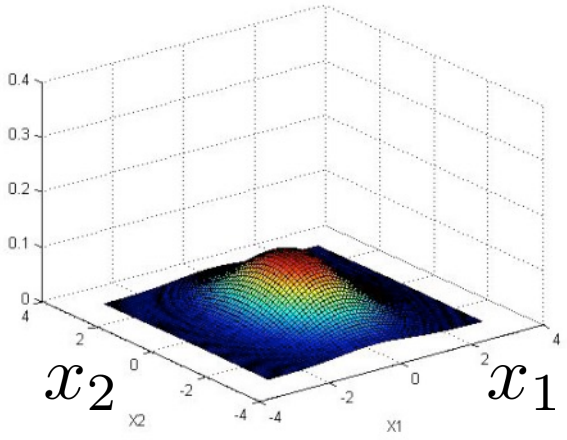


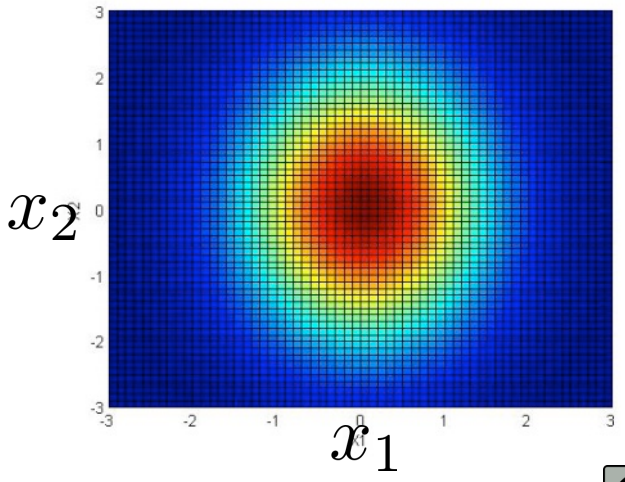
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 0.6 \end{bmatrix}$$





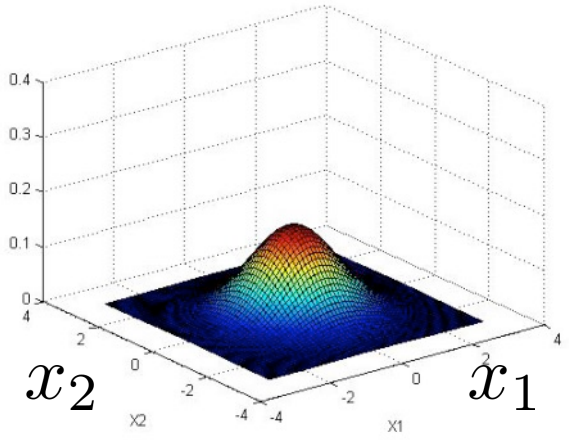
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

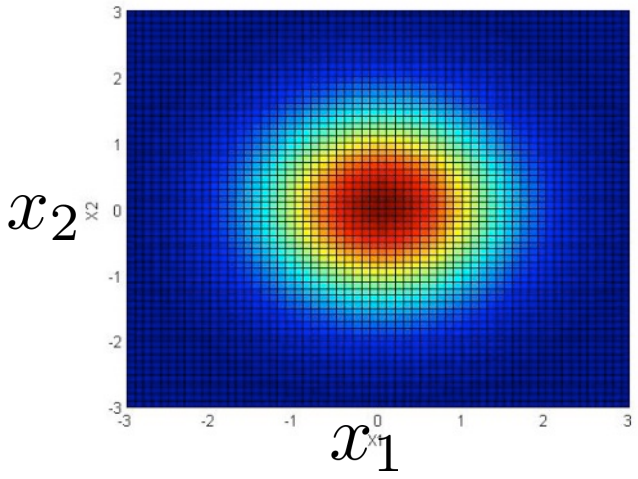




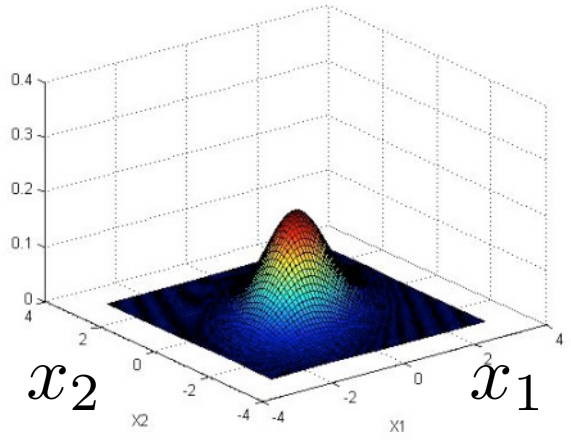
Multivariate Gaussian examples

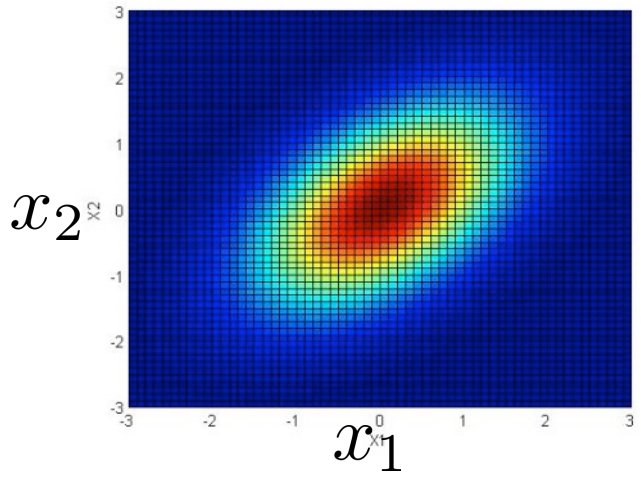
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



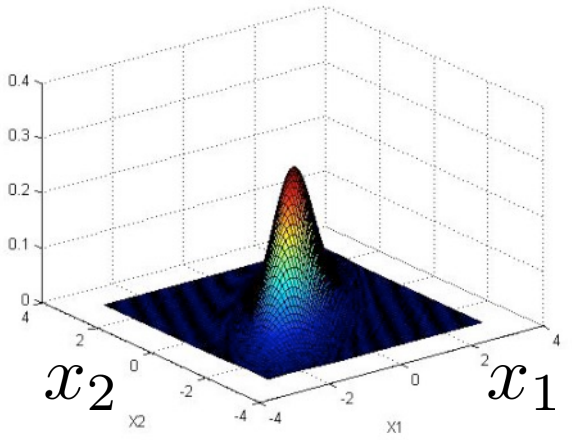


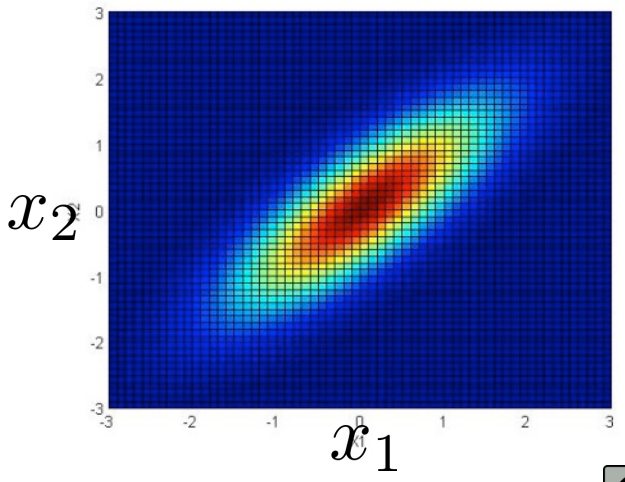
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$





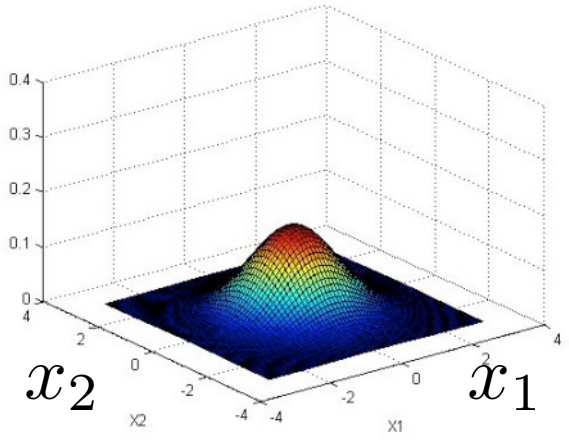
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

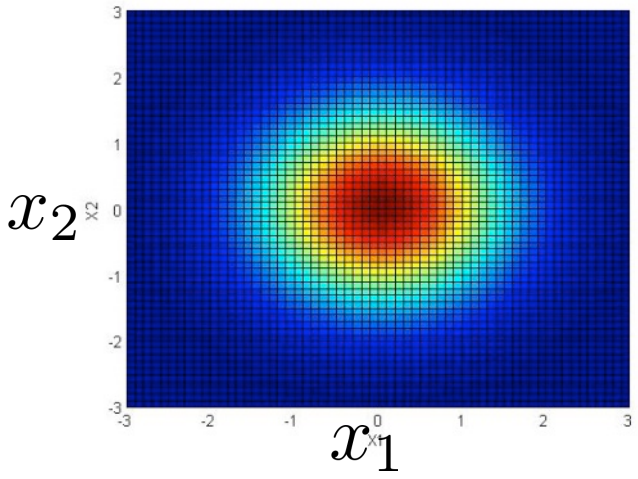




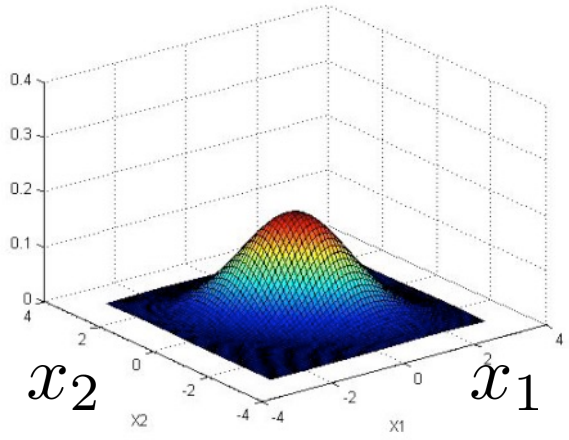
Multivariate Gaussian examples

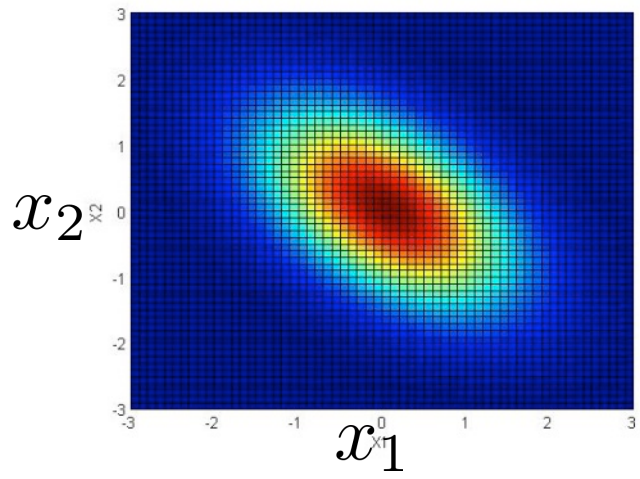
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



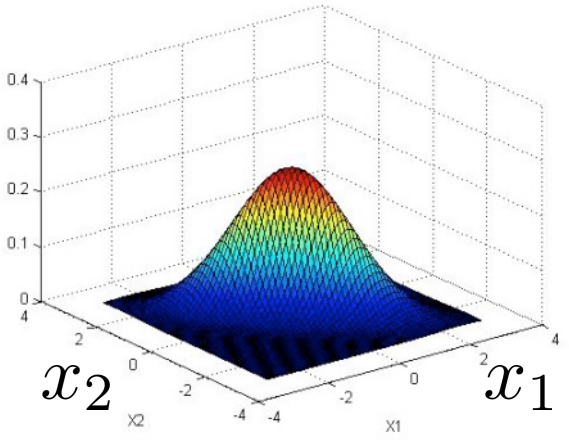


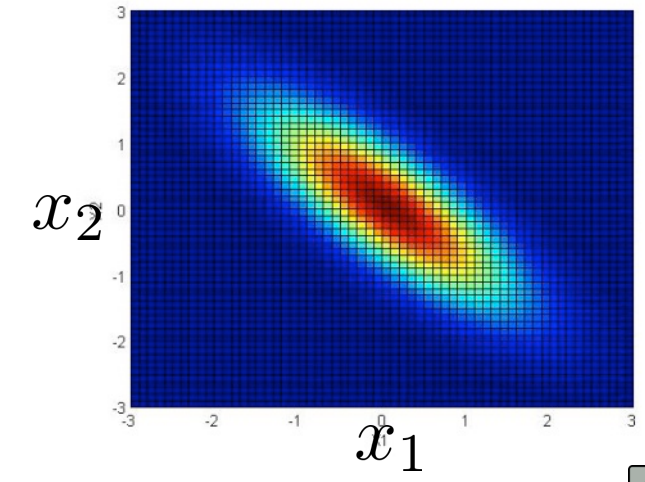
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}$$





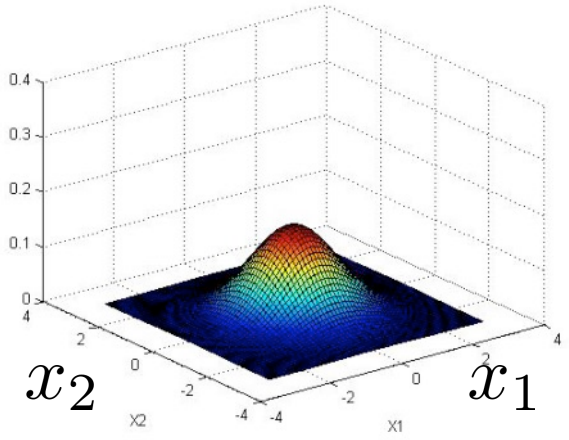
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix}$$

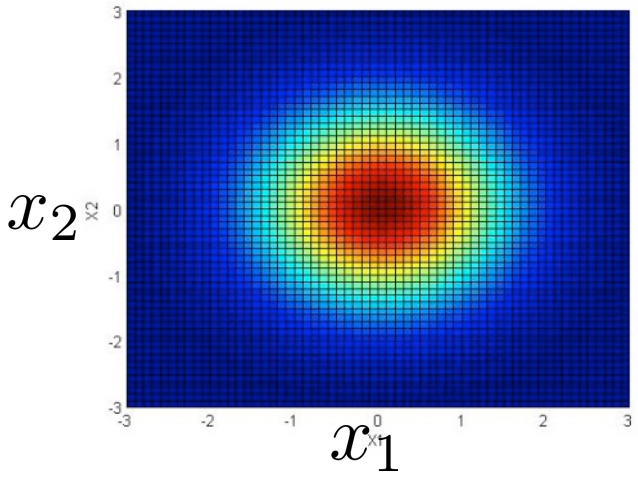




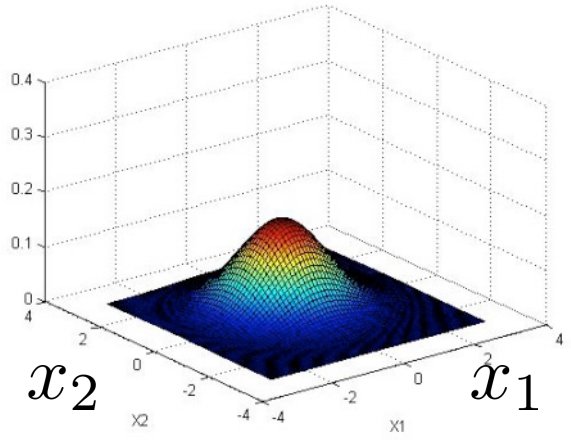
Multivariate Gaussian examples

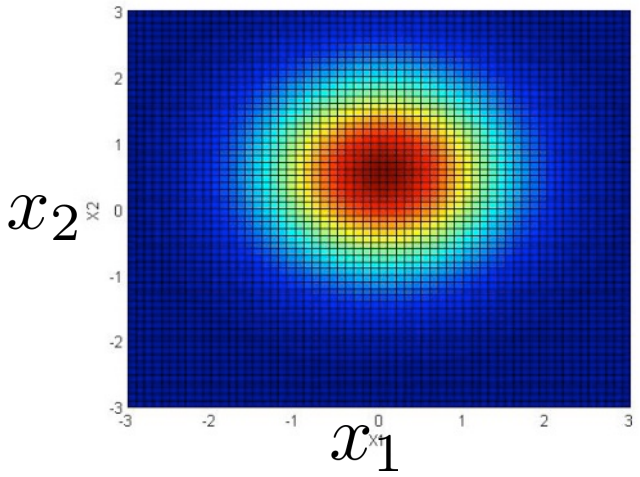
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



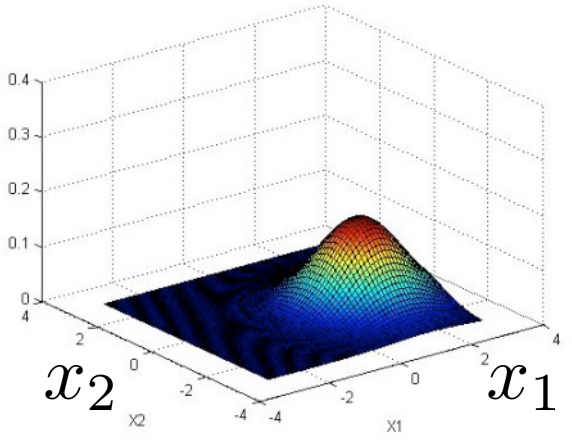


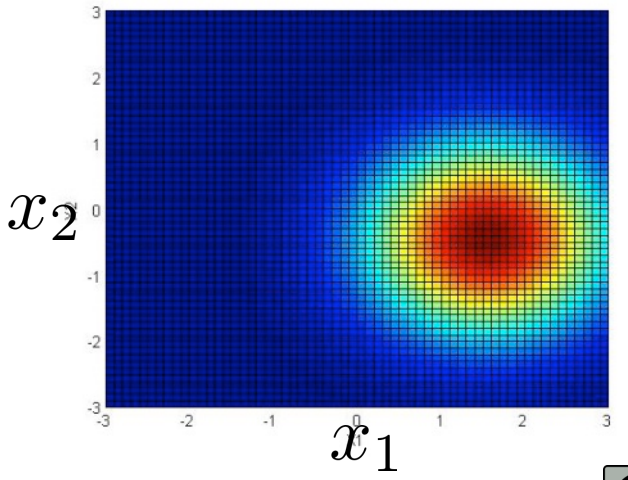
$$\mu = \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$





$$\mu = \begin{bmatrix} 1.5 \\ -0.5 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$





Univariate/Multivariate Gaussian

Univariate Gaussian

$$p(x_1; \mu_1, \sigma_1^2) \times \ldots \times p(x_n; \mu_n, \sigma_n^2)$$

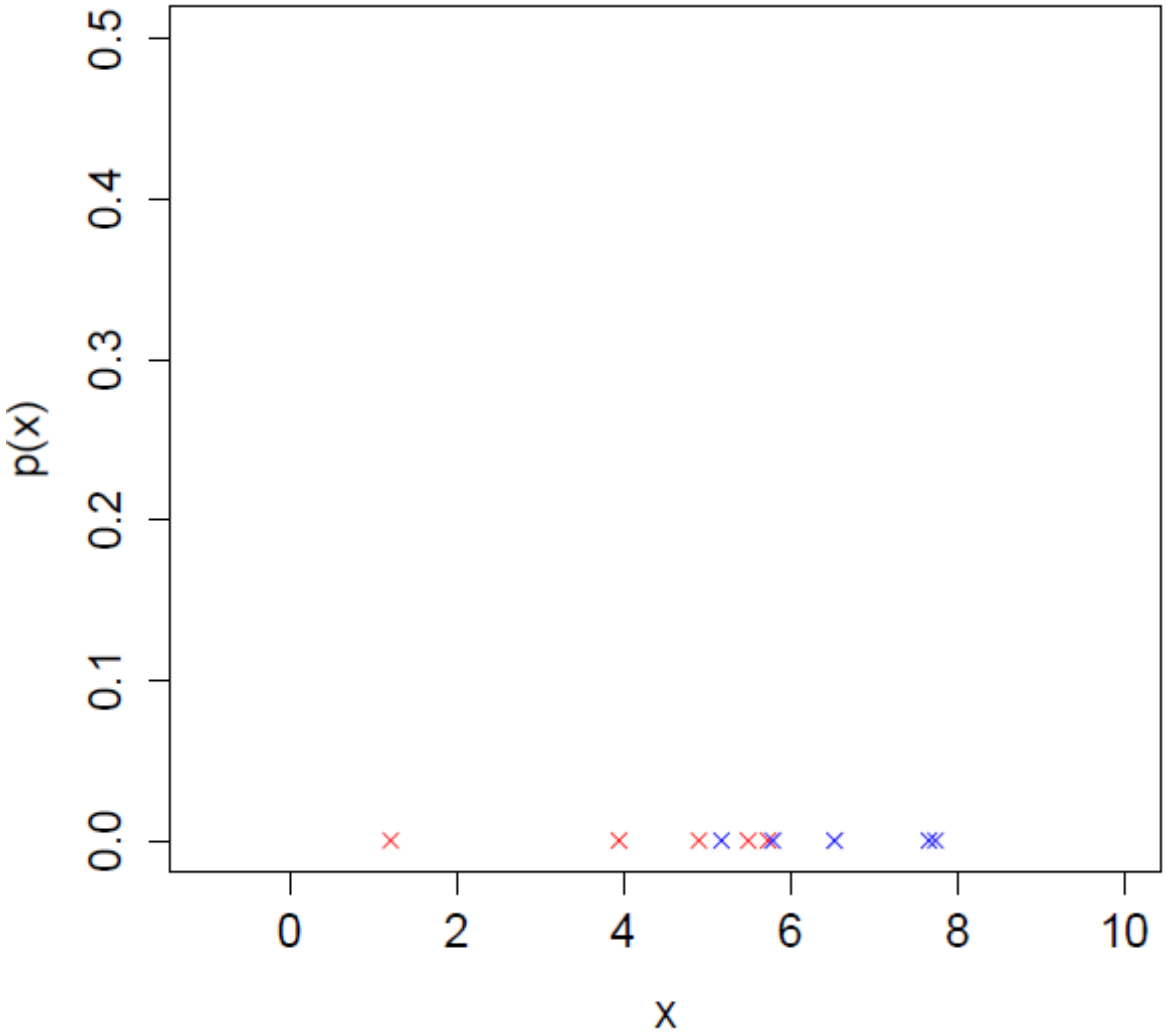
- You can manually create features with unusual combinations of values.
- Computationally cheaper
- Scales better for large n
- Works even if m is small

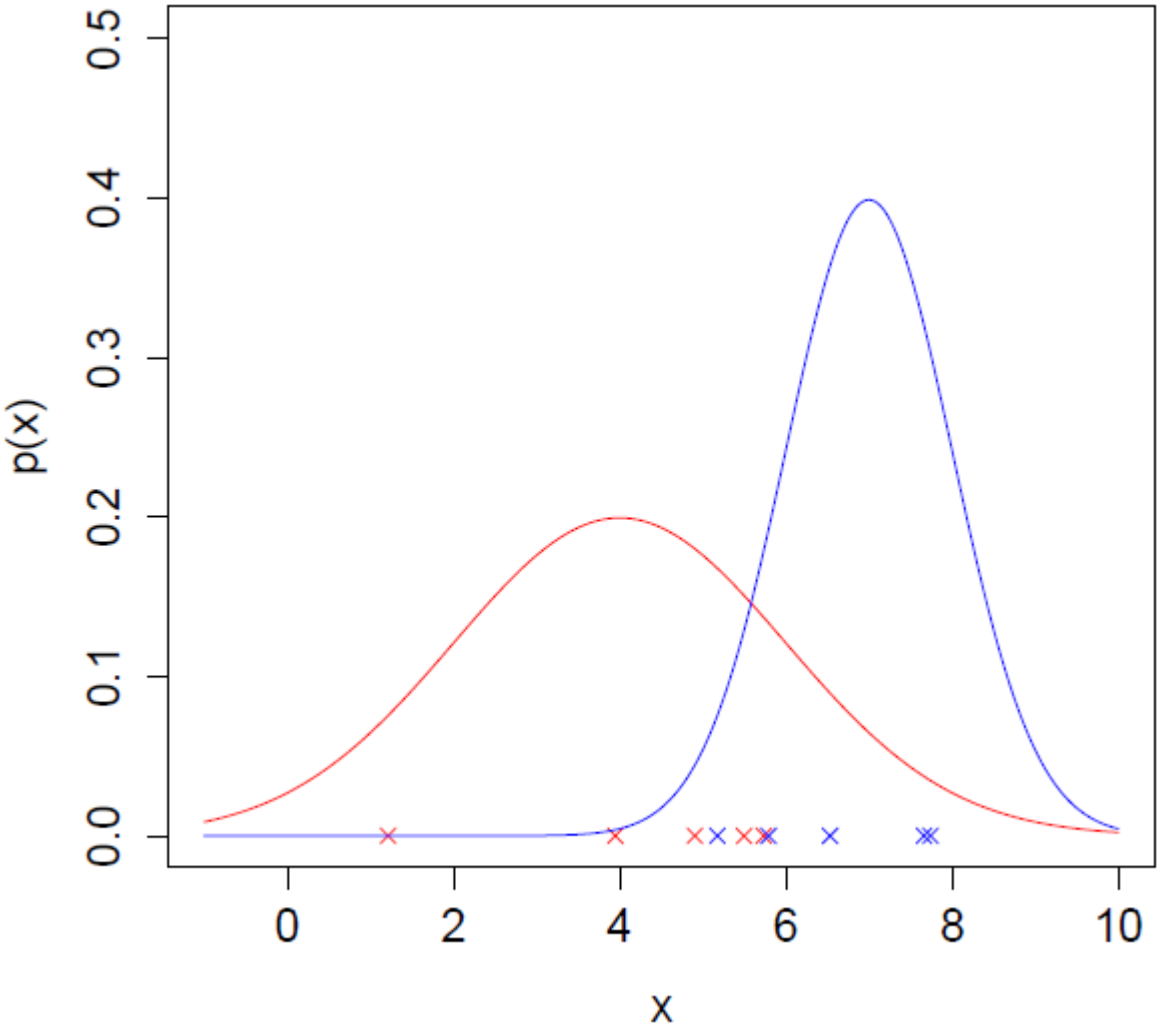
Multivariate Gaussian

$$p(x; \mu, \Sigma) = \frac{e^{\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)}}{\sqrt{(2\pi)^n |\Sigma|}}$$

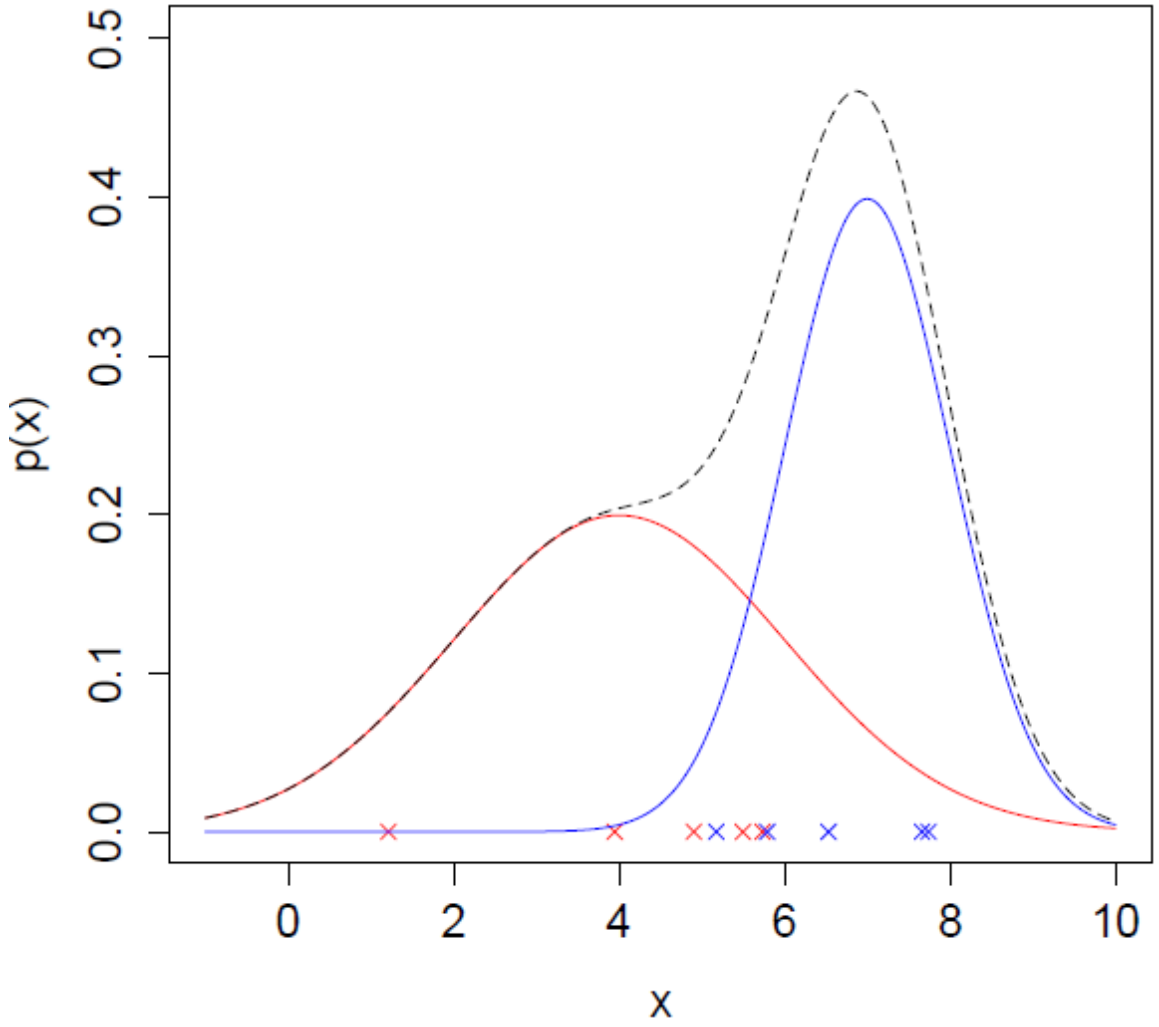
- Automatically captures correlations between dimensions
- Computationally more expensive
- If $m \leq n$ then Σ is not invertible

Mixture Model





Mixture Model



Mixture Model

The probability of a point (sample) in the mixture is the sum of the probability (the marginal probability) of the same point in each k Gaussians.

$$p(\mathbf{x}) = \sum_{j=1}^{k} p(\mathbf{x} \wedge z = j).$$

The probability can be explicitly rewritten as

$$p(\mathbf{x}) = \sum_{j=1}^{k} p(\mathbf{x}|z = j)p(z = j).$$

Mixture Model

We can set:

- $p_j(x) = p(\mathbf{x}|z = j)$ the probability density of the jth gaussian in the point x
- $\pi_j = p(z = j)$ the probability of the jth gaussian among all gaussian, called mixing coefficient

We can rewrite:

$$p(\mathbf{x}) = \sum_{j=1}^{k} \pi_j p_j(\mathbf{x}).$$

Mixture Model

We can further make explicit the parameters of the gaussians (mean and variance), through a theta parameter.

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{j=1}^{k} \pi_j p_j(\mathbf{x}|\boldsymbol{\theta_j})$$

the prob we want to maximize

θ_j represents the parameters of the jth gaussian

θ represents the parameters of all gaussians

Mixture Model

A multivariate gaussian under the mixture perspective can be defined as:

$$p(x;\mu_j,\Sigma_j) = \frac{e^{\left(-\frac{1}{2}(x-\mu_j)^T\Sigma_j^{-1}(x-\mu_j)\right)}}{\sqrt{(2\pi)^n|\Sigma_j|}}$$

Where:

$\boldsymbol{\mu}_j$ is the vector of the means of the jth gaussian

$\boldsymbol{\Sigma}_j$ is the covariance matrix of the jth gaussian

n is the number of the features

Mixture Model

What we want to know are the parameters of the Gaussians:

$$\boldsymbol{\mu}_j \text{ e } \boldsymbol{\Sigma}_j \quad j = 1, \ldots, k.$$

Let us define C_j as the jth cluster, and $|C_j|$ the number of the points that belong to it.

If we knew which point belongs to which cluster, we would compute the parameters as:

Univariate

$$\hat{\mu}_j = \frac{1}{|C_j|} \sum_{x^{(i)} \in C_j} x^{(i)}$$

$$\hat{\sigma}_j^2 = \frac{1}{|C_j| - 1} \sum_{x^{(i)} \in C_j} (x^{(i)} - \hat{\mu}_j)^2$$

Multivariate

$$\hat{\boldsymbol{\mu}}_{\boldsymbol{j}} = \frac{1}{|C_j|} \sum_{\mathbf{x^{(i)}} \in C_j} \mathbf{x^{(i)}}$$

$$\hat{\Sigma}_j = \frac{1}{|C_j| - 1} \sum_{\mathbf{x^{(i)}} \in C_j} (\mathbf{x^{(i)}} - \hat{\boldsymbol{\mu}}_{\boldsymbol{j}})(\mathbf{x^{(i)}} - \hat{\boldsymbol{\mu}}_{\boldsymbol{j}})^T$$

Expectation-Maximization Algorithm (EM)

1. Randomly set the parameters (μ_j, Σ_j and π_j) for each of the K gaussians.
2. Repeat {
   1. **(E) Expectation step**: for each cluster-point pair the «responsibility» coefficient is computed. This coefficient gives an idea of how much the point is generated from the gaussian that is paired with it at the moment, w.r.t. the entire mixture.
   2. **(M) Maximization step**: the parameters (means, variances and mixing coefficient) are recomputed as a function of the responsibility coefficient to maximize the Likelihood.

   } until convergence

Expectation step

For each cluster-point pair, the «responsibility» coefficient is computed. This coefficient shows how much the point is generated from the Gaussian that is paired with it at the moment, w.r.t. the entire mixture.

$$r_{ij} := p(z = j|\mathbf{x}^{(\mathbf{i})}, \boldsymbol{\theta}^{t-1})$$

$\boldsymbol{\theta_j}^{(t-1)}$ are the parameters set in the previous iteration.

We can make the formula more explicit:

$$r_{ij} = \frac{\pi_j p_j(\mathbf{x}^{(i)}|\boldsymbol{\theta_j}^{(t-1)})}{\sum_{j=1}^{k} \pi_j p_j(\mathbf{x}^{(i)}|\boldsymbol{\theta_j}^{(t-1)})}$$

Maximization step

The parameters (means, variances and mixing coefficient) are recomputed as a function of the responsibility coefficient to maximize the likelihood.

$$\pi_j = \frac{1}{m}\sum_{i=1}^{m} r_{ij}$$

$$\boldsymbol{\mu_j} = \frac{\sum_i r_{ij}\mathbf{x^{(i)}}}{\sum_i r_{ij}}$$

$$\mathbf{\Sigma_j} = \frac{\sum_i r_{ij}(\mathbf{x}^{(i)} - \boldsymbol{\mu_j})(\mathbf{x}^{(i)} - \boldsymbol{\mu_j})^T}{\sum_i r_{ij}}$$

Gaussian Mixture Models – PROS/CONS

Pros

- It is a flexible algorithm that is able to approximate multivariate gaussian to fit data maximing the likelihood
- It is not affected by bias approaching zero means
- Less sensible to outliers

Cons

- The algorithm can diverge if there are only few points w.r.t. the overall mixture
- All the flexibility of the model is always used, even if it is not needed

Politecnico di Bari

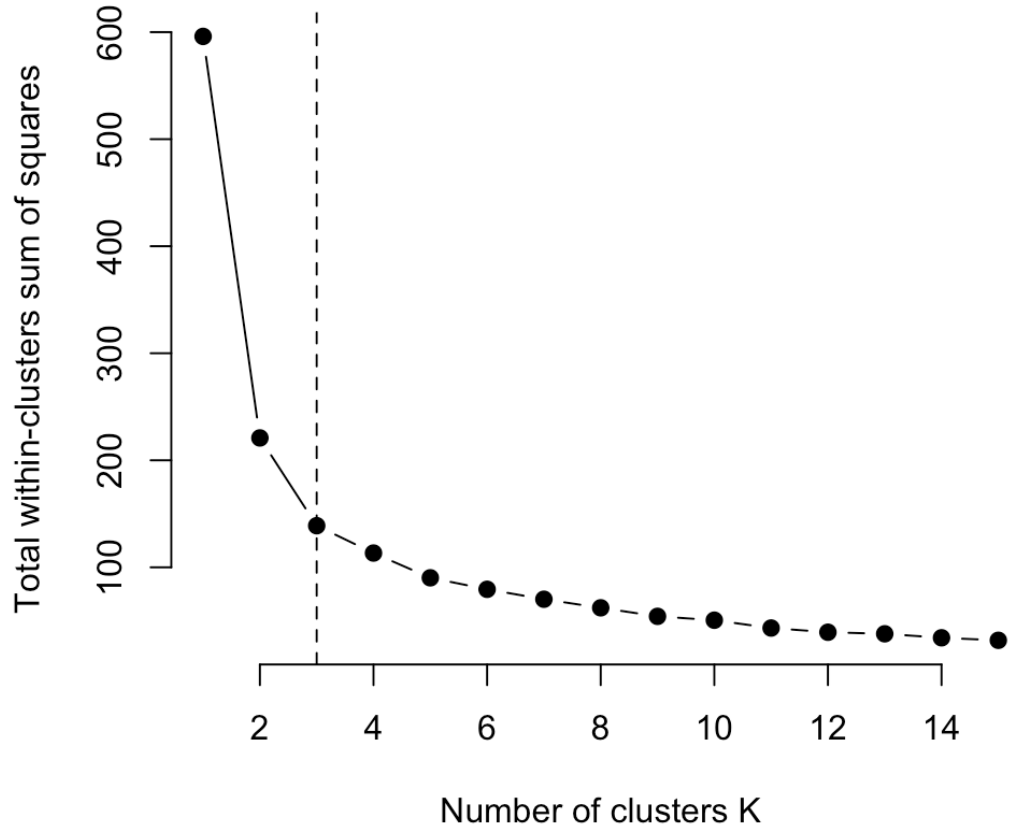
Clustering

Choosing the value of K

Choosing the value of K

Elbow method:

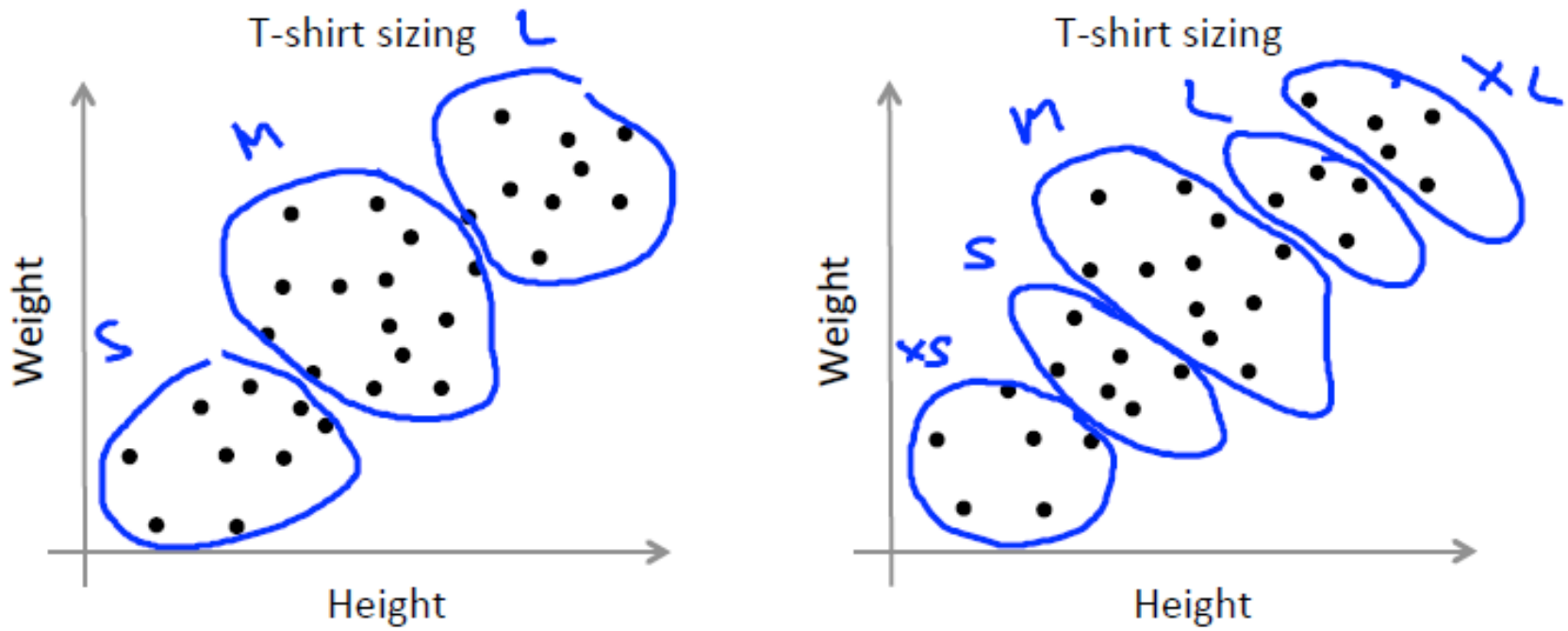
easy visual way to choose it.
aproach of 99.999% of company



Choosing the value of K

Sometimes our ML system must fit some external constraints, in those cases the number of clusters depends on how well the clusters performs on an external task.

E.g.



Kullback–Leibler divergence

It can be used to measures how the «true» probability distribution $P(i)$ diverges from the expected probability distribution, i.e.,the distribution of the model, $Q(i)$.

It can assume values from 0 (when the two distributions behave in the same way) to 1 (when their behaviour is completely different)

$$D_{KL}(P\|Q) = -\sum_i P(i)\log\frac{Q(i)}{P(i)} = \sum_i P(i)\log\frac{P(i)}{Q(i)}$$

Information criterion approach

AIC: Akaike information criterion

BIC: Bayesian information criterion

DIC: Deviance information criterion

BF: Bayes factors

MDL: Minimum Description Length

FIA: Fisher Information Aproximation

Recap Likelihood $\hat{L} = L(\hat{\theta}) = \max_{\theta} L(\theta; X; y; M) = p(y|X; \hat{\theta}, M)$

this is not specific only to clustering but also for other predictor like regression exc.

Akaike information criterion (AIC)

AIC derived as asymptotic approximation of Kullback-Liebler information distance between the model of interest and the truth:

$$AIC = 2k - 2\ln(\hat{L})$$

optimal likelihood

number of parameters

It can also be used AICc, in which a correction related to the number of samples is injected:

$$AICc = AIC + \frac{2k \cdot (k+1)}{m - k - 1}$$

The model that minimizes AIC is selected

Bayesian information criterion

BIC is the asymptotic approximation under the assumption that the model can result in an exponential distribution.

$$BIC = \ln(m) \cdot k \ - 2 \ln(\hat{L})$$

number of parameters

number of samples

The model that minimizes BIC is selected

Deviance information criterion

It represents an asymptotic approximation of the deviance under the assumption that the posterior distribution is a multivariate normal

$$D(\theta) = -2\log\left(p(y|X;\theta,M)\right)$$ ← deviance

$$\bar{D(\theta)} = \mathbb{E}[D(\theta)]$$

$$\widetilde{\theta} = \mathbb{E}[\theta|y]$$ ← posterior mean deviance

$$p_D = \mathbb{E}_{\theta|y}[-2\log\left(p(y|X;\theta,M)\right) + 2\log\left(p(y|X;\widetilde{\theta},M)\right)]$$

$$DIC = \bar{D(\theta)} + p_D$$

The model that minimizes DIC is selected

Information criterions comparison

- AIC lacks the finite samples correction in its base form
- AIC is asymptotically equivalent to k-folds cross-validation on linear regression models with a least squares optimization
- BIC is able to detect the true model (if present in the comparison)
- BIC is only valid if n is much greater than k
- BIC penalizes complex models more than AIC
- DIC tends to suggest overfitted models

Silhouette Coefficient

is used only for clustering

Silhouette Coefficient combines the ideas of cohesion and separation for single samples, clusters, and clusterings

For an individual sample, $x^{(i)}$

cohesian
- ***coh*** = average distance of $x^{(i)}$ to the samples in the same cluster

separation
- ***sep*** = min (average distance of $x^{(i)}$ to samples in another cluster)
- silhouette coefficient of $x^{(i)}$:

$$s_i = 1 - \frac{coh}{sep} \text{ if } coh < sep$$

The closer to 1 the better.

Can calculate the Average Silhouette width for a cluster or a clustering

Politecnico di Bari

Clustering

Hierarchical Clustering

Hierarchical Clustering

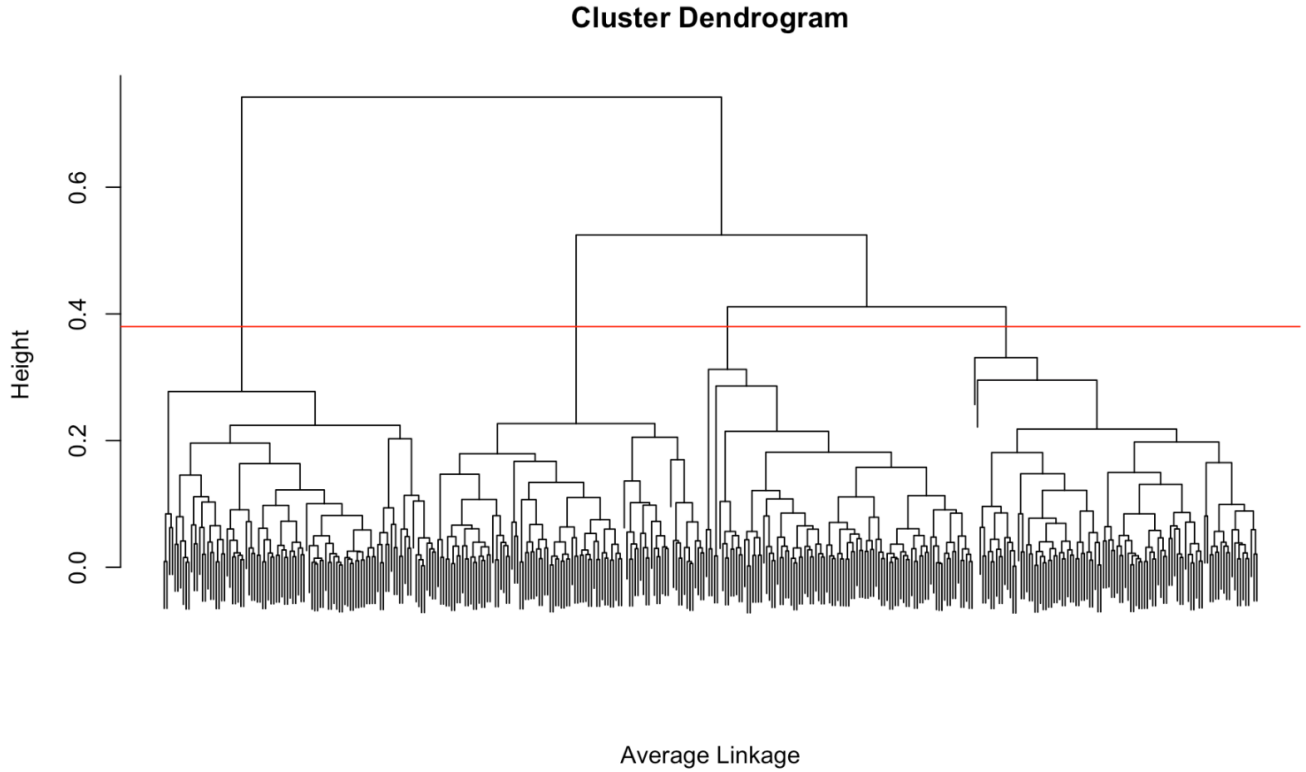
Hierarchical cluster involves building a hierarchy of (potentially overlapping) clusters.

- Agglomerative (“bottom-up”)
- Divisive (“top-down”)

Once the hierarchy is built,

cluster extraction is done

by selecting some height to ‘cut’



Hierarchical Clustering

Deciding how to *merge* or *split* clusters depends on 1) the **similarity measure/metric** and 2) the **linkage criterion**

1. Euclidean Distance, Mahalanobis distance, etc.
2. Could be:
   - **Distance based**:Max/min/mean distance (complete/single/ average linkage)
   - **Probability based**: the probability that the candidate clusters come from the same distribution
   - The product of the **in-and-out degree** of a node (graph degree linkage)
   - **Instance-based constraints** (see Wagstaff 2002)

Hierarchical Clustering: Single Linkage

Perhaps the most well-known hierarchical clustering algorithmis called the **Single-Linkage algorithm**

The idea:

1. Start with every single point in its own cluster (*n* clusters)
2. Merge the *two closest points* to form a new cluster
3. Repeat: each time *merging the two closest pair of points*

The configuration:

- The **similarity metric** is **Euclidean distance**
- The **linkage criterion** is the [pairwise] minimum distance

Hierarchical Clustering: Wards Criterion

Rather than use distance metrics as measures of similarity, why not merge based on a more statistically intuitive notion

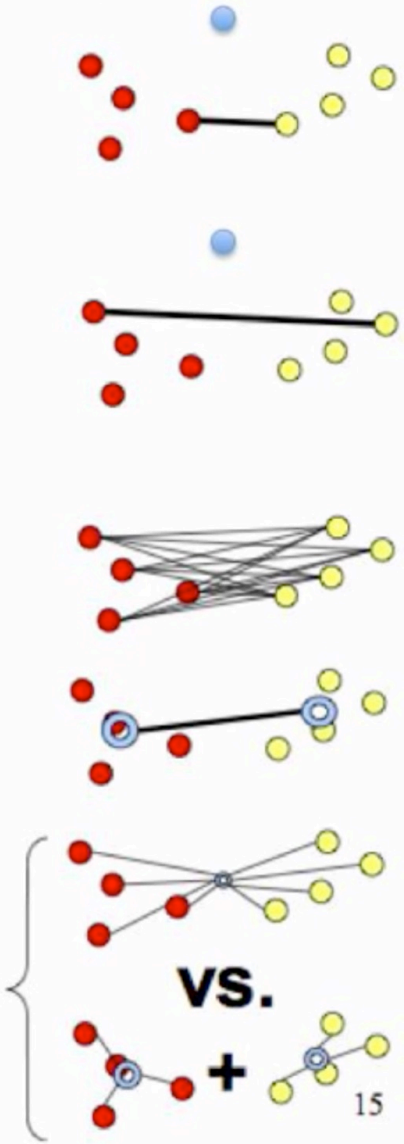
Perhaps its fair to assume observations come from an approximately elliptically-shaped distribution

Ward proposed agglomeration where observations are merged in such as way that minimizes the total squared error (maximizes r^2 at each merge)

"…Interpreted as the **proportion of variation explained by a particular clustering of the observations.**"

Cluster distance measures

- Single link: $D(c_1, c_2) = \min_{x_1 \in c_1, x_2 \in c_2} D(x_1, x_2)$
  - distance between closest elements in clusters
  - produces long chains a→b→c→…→z
- Complete link: $D(c_1, c_2) = \max_{x_1 \in c_1, x_2 \in c_2} D(x_1, x_2)$
  - distance between farthest elements in clusters
  - forces "spherical" clusters with consistent "diameter"
- Average link: $D(c_1, c_2) = \frac{1}{|c_1|}\frac{1}{|c_2|}\sum_{x_1 \in c_1}\sum_{x_2 \in c_2} D(x_1, x_2)$
  - average of all pairwise distances
  - less affected by outliers
- Centroids: $D(c_1, c_2) = D\left(\left(\frac{1}{|c_1|}\sum_{x \in c_1} \vec{x}\right), \left(\frac{1}{|c_2|}\sum_{x \in c_2} \vec{x}\right)\right)$
  - distance between centroids (means) of two clusters
- Ward's method: $TD_{c_1 \cup c_2} = \sum_{x \in c_1 \cup c_2} D(x, \mu_{c_1 \cup c_2})^2$
  - consider joining two clusters, how does it change the total distance (TD) from centroids?



Politecnico di Bari

Clustering

DBSCAN

DBSCAN

DBSCAN stands for **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise

DBSCAN is a **density-based agglomerative** algorithm

Density-based Clustering locates regions of high density that are separated from one another by regions of low density

In this case **Density** is defined as the number of points within a specified radius (**Eps**)

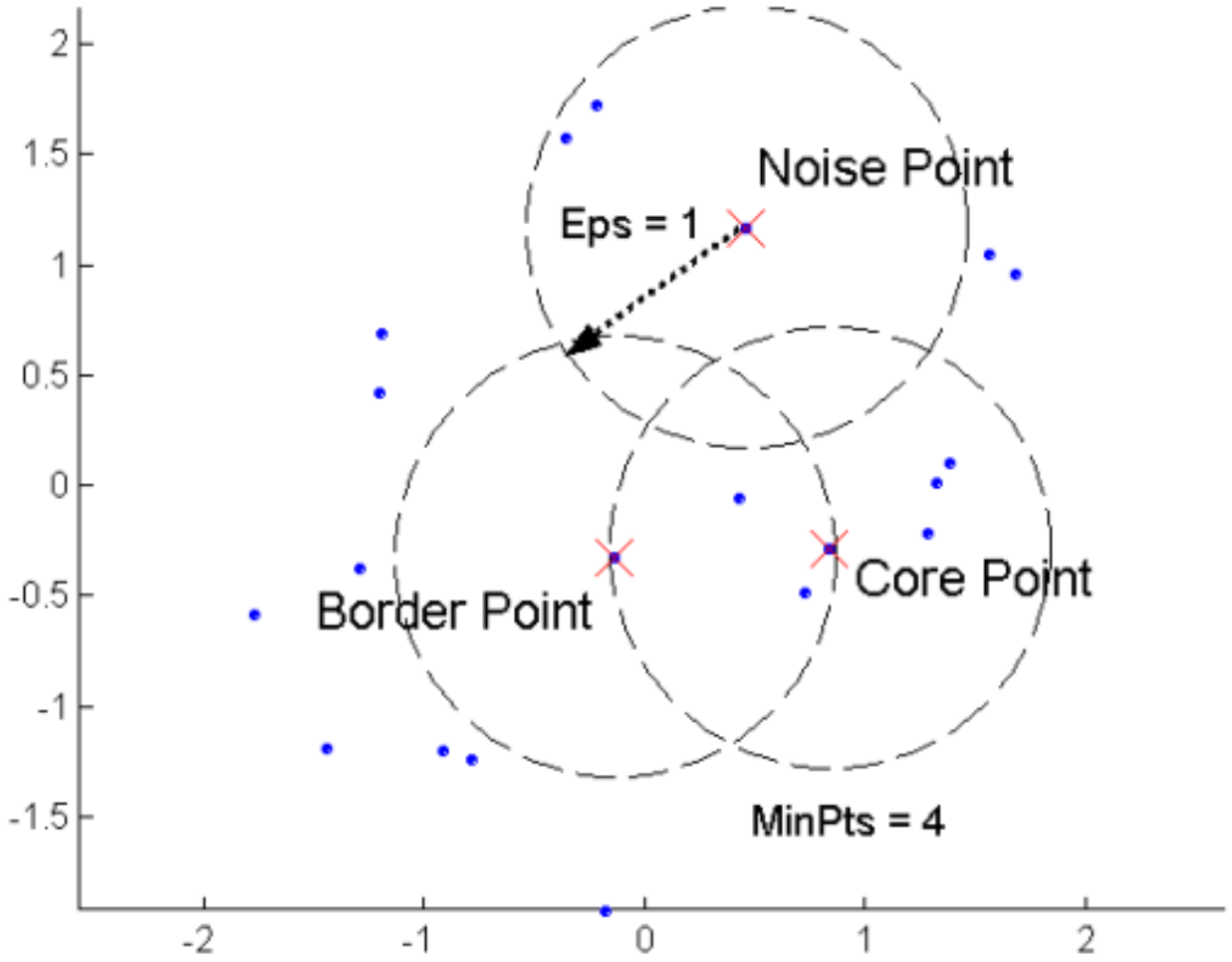
DBSCAN concepts

- A point is a core point if it has more than a specified number of points (MinPts) within Eps
  - These are points that are at the interior of a cluster
- A border point has fewer than MinPts within Eps, but is in the neighborhood of a core point
- A noise point is any point that is not a core point or a border point
- Eps-Neighborhood are the points within a radius of Eps from a point

DBSCAN concepts

- Any two core points are close enough, within a distance *Eps* of one another, are put in the same cluster
- Any border point that is close enough to a core point is put in the same cluster as the core point
- Noise points are discarded

DBSCAN


Noise Point
Eps = 1
Border Point
Core Point
MinPts = 4
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-1
0
1
2

DBSCAN

We have to set two different parameters:

Epsilon := physical distance or radius

minPts := desired minimum cluster size

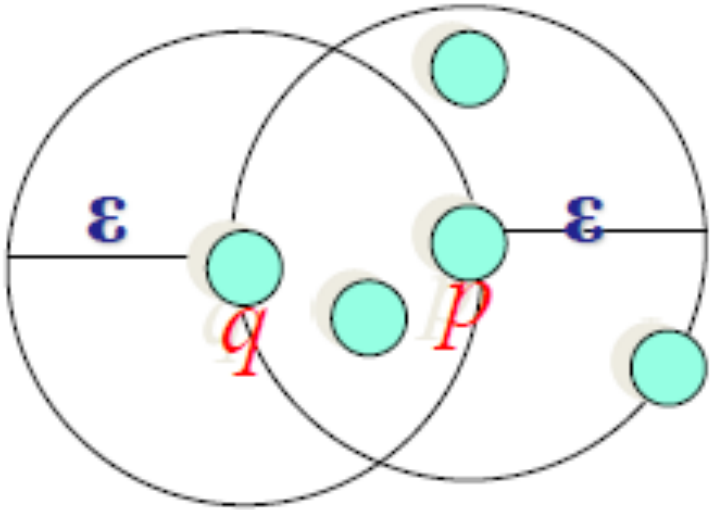
- Epsilon can be chosen using a k-distance graph
- minPts can be chosen with the empirical rule:

  minPts>= (number of Dimensions) +1

DBSCAN Reachability

Directly density-reachable

An object q is directly density-reachable from object p if q is within the ε-Neighborhood of p and p is a core object.



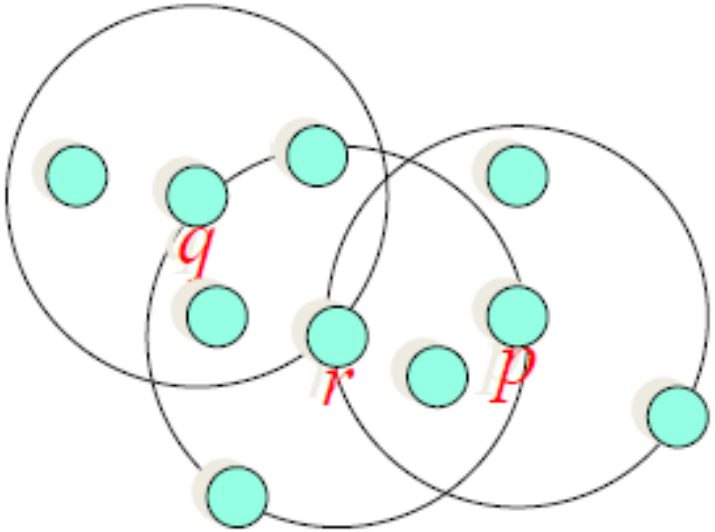
q is directly density-reachable from p

p is not directly density-reachable from q

DBSCAN Connectivity

Density-connectivity

Object *p* is density-connected to object *q* w.r.t ε and *MinPts* if there is an object *o* such that both *p* and *q* are density-reachable from *o* w.r.t ε and *MinPts*



P and *q* are density-connected to each other by *r*

Density-connectivity is symmetric

DBSCAN Algorithm

CorePoints =[]

NoisePoints = []

For $x^{(i)}$ in X:

    Retrieve all points density-reachable from $x^{(i)}$

    if $x^{(i)}$ is a core point, a cluster is formed.

    CorePoints.push($x^{(i)}$)

For $x^{(i)}$ in CorePoints:

    Retrieve all the connected components.

    Merge the clusters

For $x^{(i)}$ in X/CorePoints:

    if distance from the nearest cluster < Eps:

        Assign $x^{(i)}$ to that cluster

    else

        NoisePoints.push($x^{(i)}$)

DBSCAN Pros

- DBSCAN does not require one to specify the number of clusters in the data a priori, as opposed to k-means.
- DBSCAN can find arbitrarily shaped clusters. It can even find a cluster completely surrounded by (but not connected to) a different cluster. Due to the MinPts parameter, the so-called single-link effect (different clusters being connected by a thin line of points) is reduced.
- DBSCAN has a notion of noise, and is robust to outliers.
- DBSCAN requires just two parameters and is mostly insensitive to the ordering of the points in the database.
- DBSCAN is designed for use with databases that can accelerate region queries, e.g. using an R* tree.
- The parameters minPts and ε can be set by a domain expert, if the data is well understood.

DBSCAN Cons

- DBSCAN is not entirely deterministic: border points that are reachable from more than one cluster can be part of either cluster, depending on the order the data is processed.
- The quality of DBSCAN depends on the distance measure. The most common distance metric used is Euclidean distance. Especially for high-dimensional data, this metric can be rendered almost useless due to the so-called “Curse of dimensionality”
- DBSCAN cannot cluster data sets well with large differences in densities, since the minPts-ε combination cannot then be chosen appropriately for all clusters.
- If the data and scale are not well understood, choosing a meaningful distance threshold ε can be difficult.

DBSCAN Complexity

- **Time Complexity:** O(n^2)
  - for each point it has to be determined if it is a core point.
  - can be reduced to O(n*log(n)) in lower dimensional spaces by using efficient data structures (n is the number of objects to be clustered);
- **Space Complexity:** O(n).

Politecnico di Bari

Clustering

HDBSCAN

Hierarchical DBSCAN

- Shown that DBSCAN corresponds *exactly* with cutting the Robust Single Linkage tree at height Eps

- ‘Flat’ extraction method proposed based on stability-based cluster validation methods

Hierarchical DBSCAN

1. Transform the space according to the density/sparsity.
2. Build the minimum spanning tree of the distance weighted graph.
3. Construct a cluster hierarchy of connected components.
4. Condense the cluster hierarchy based on minimum cluster size.
5. Extract the stable clusters from the condensed tree.

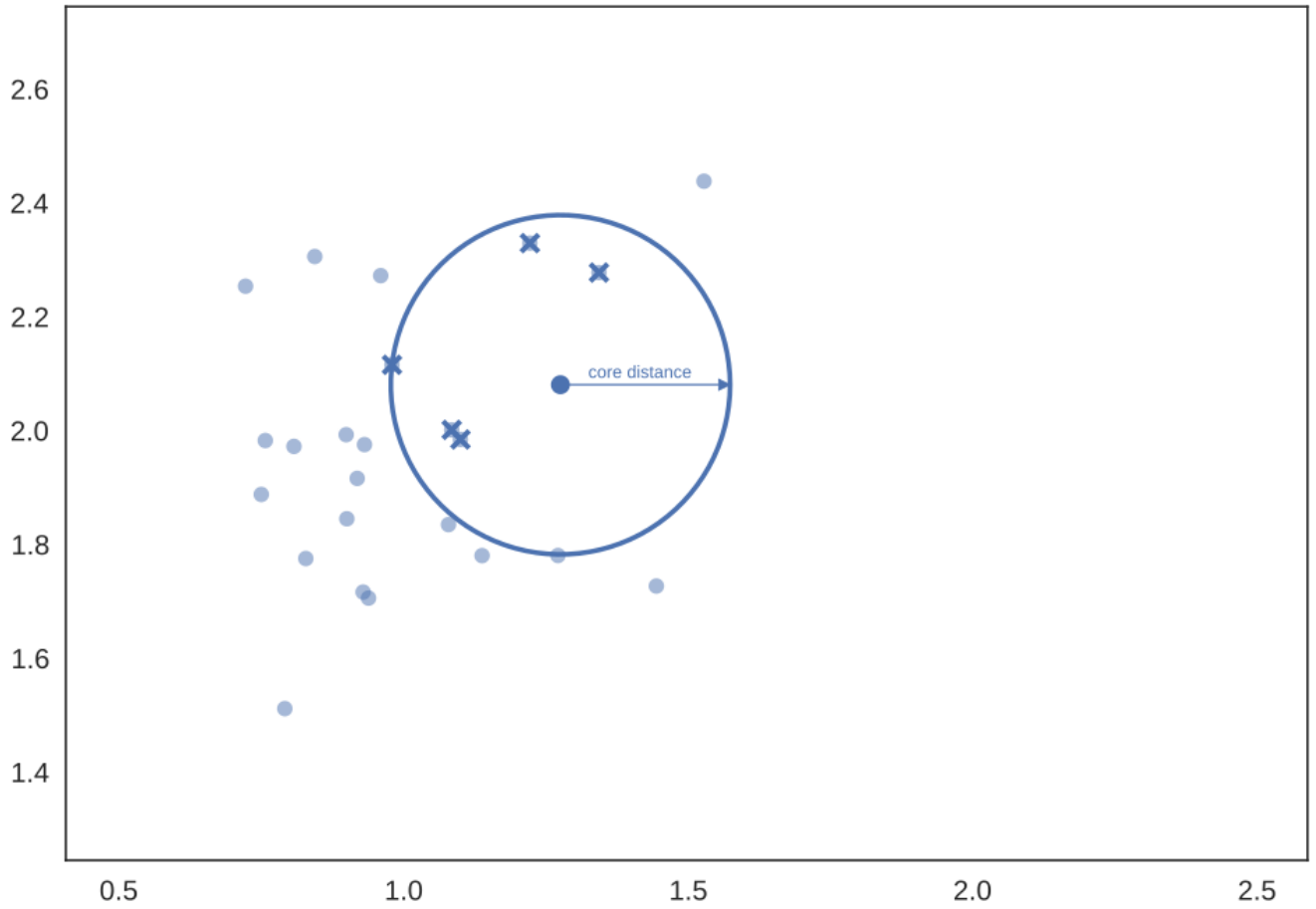
Mutual reachability distance

- First we define the core distance, $core_k(x)$ as the distance from x to the k^{th} nearest neighbor
- d(a,b) is the distance between two different samples

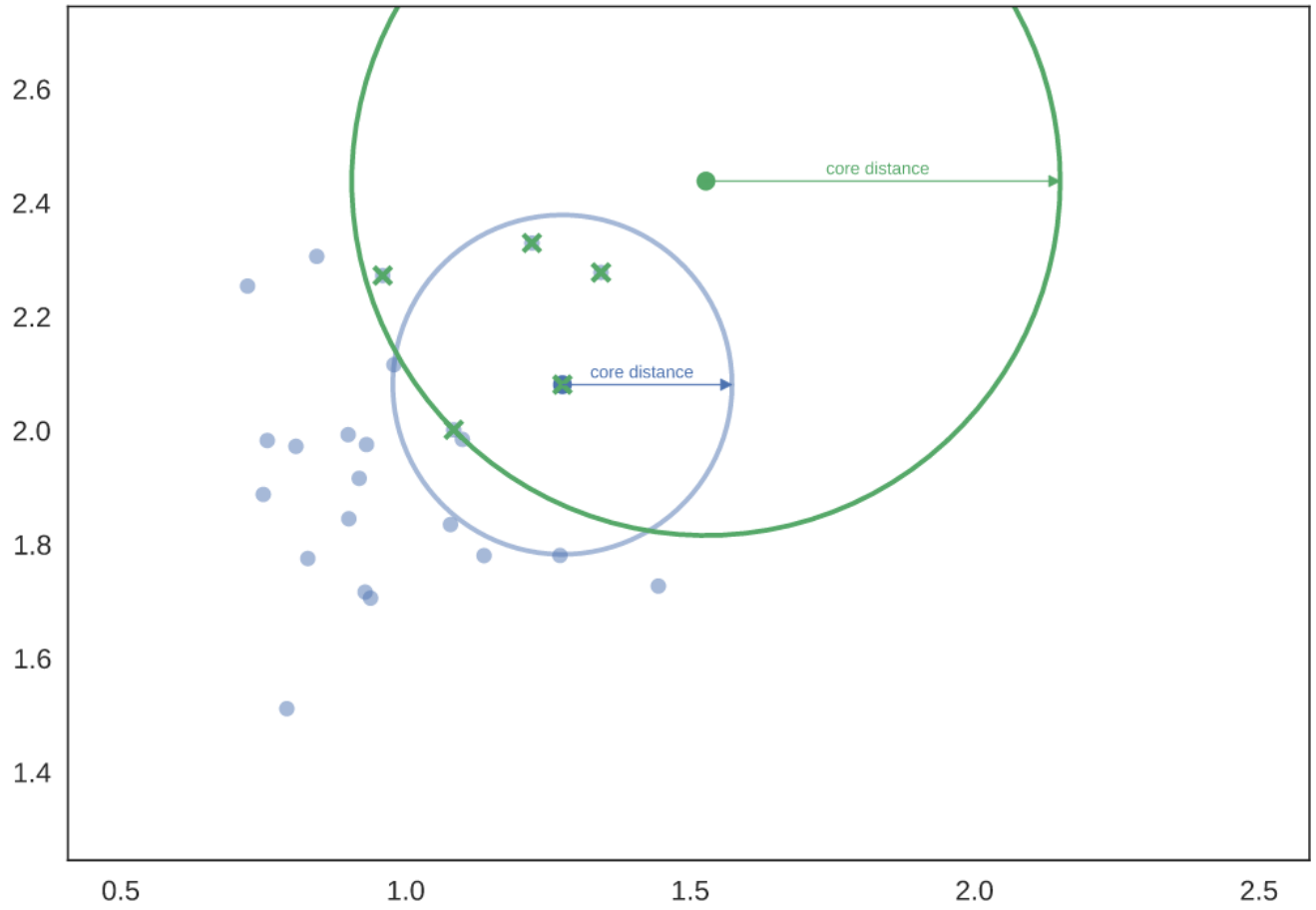
The mutual reachability distance is finally defined as:

$$d_{mreach-k}(a,b) = \max\{core_k(a), core_k(b), d(a,b)\}$$

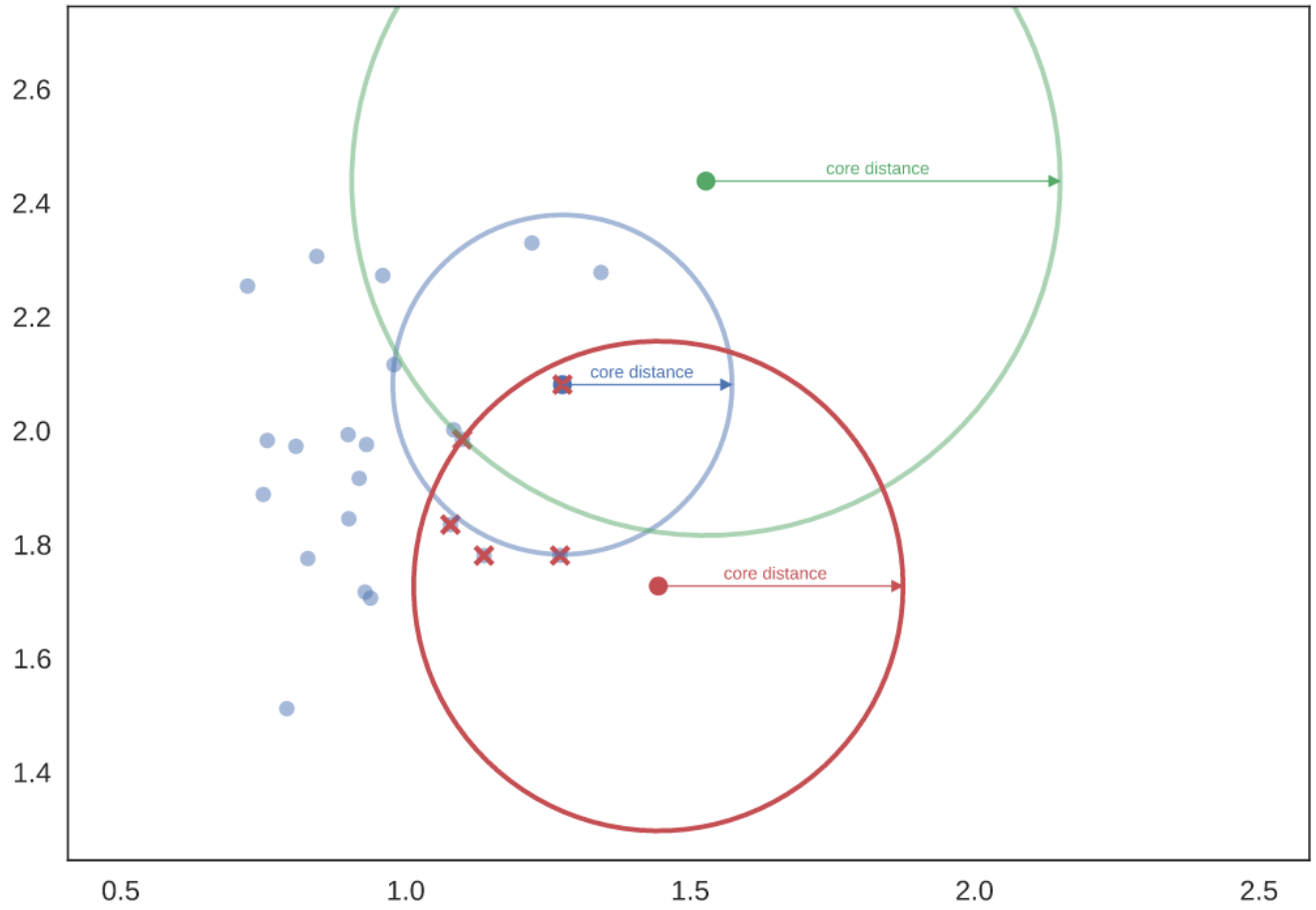
Mutual reachability distance



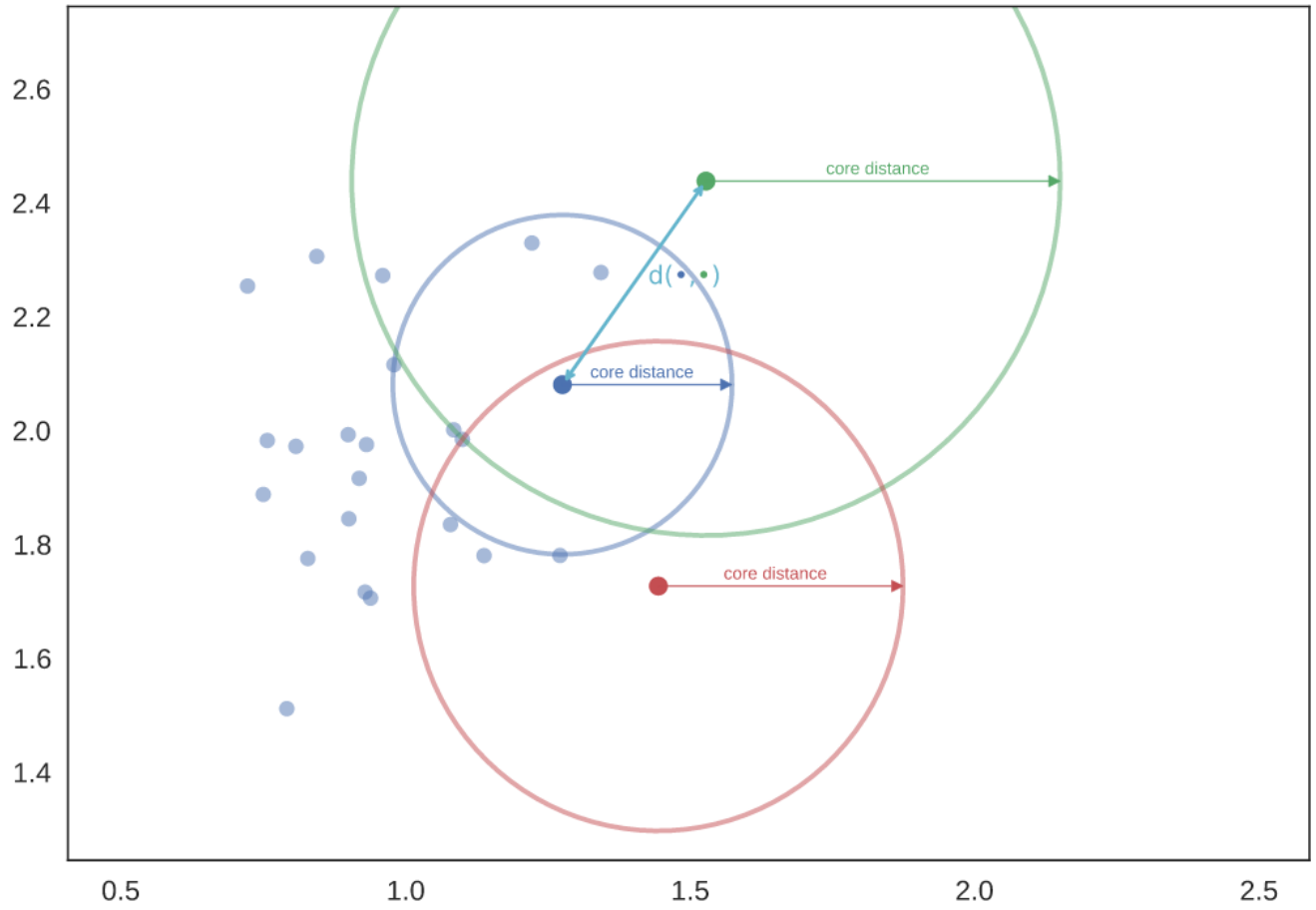
Mutual reachability distance



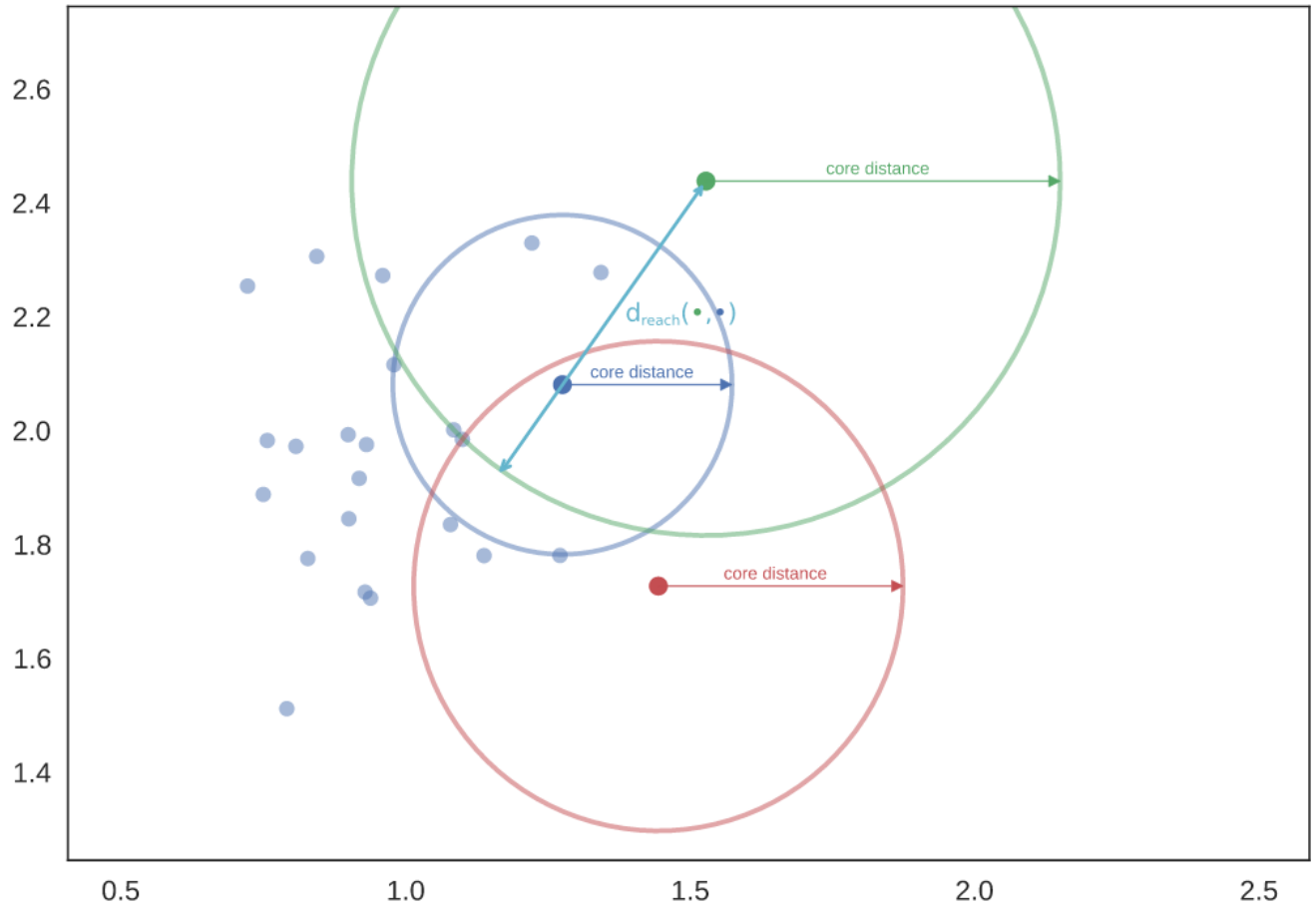
Mutual reachability distance



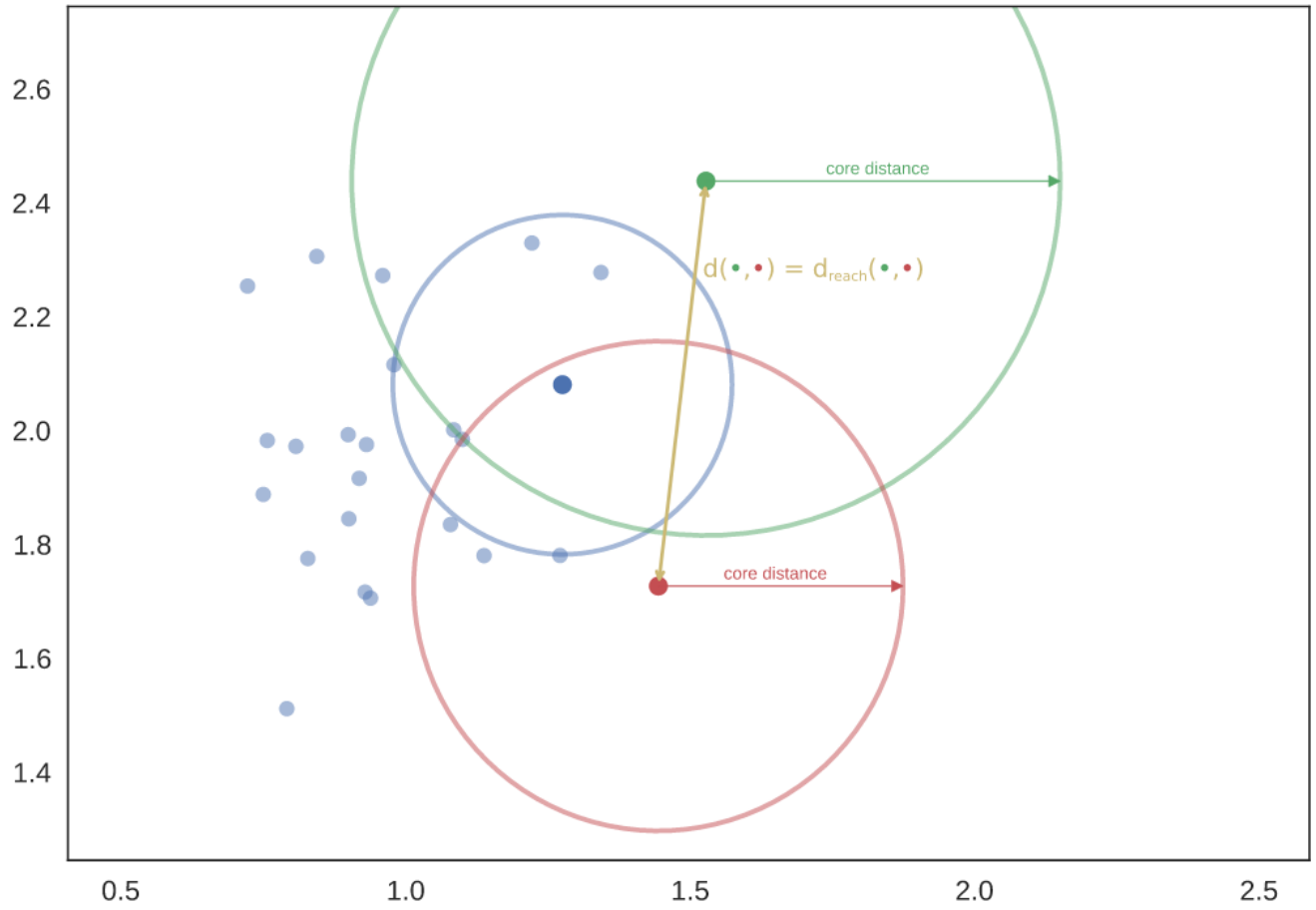
Mutual reachability distance



Mutual reachability distance



Mutual reachability distance



Build the minimum spanning tree

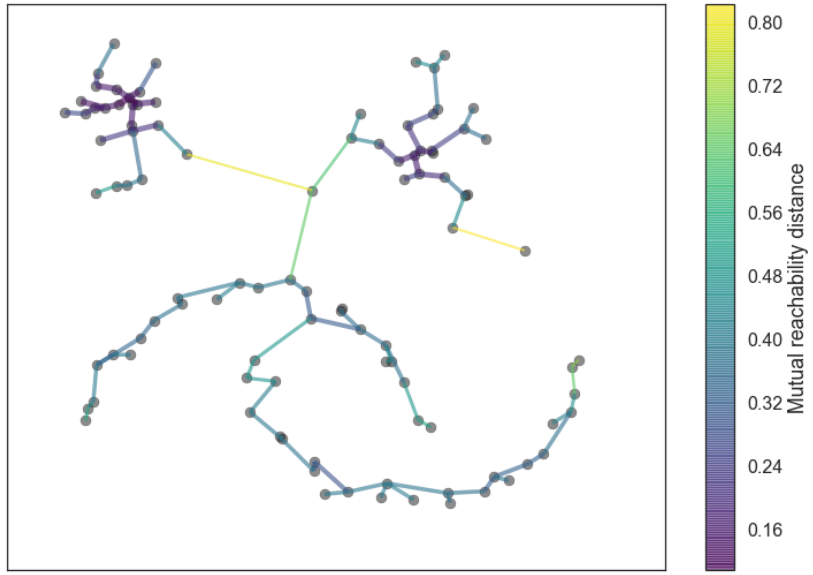
Very common algorithms to find the minimum spanning tree are:

Prim's algorithm

Kruskal's algorithm

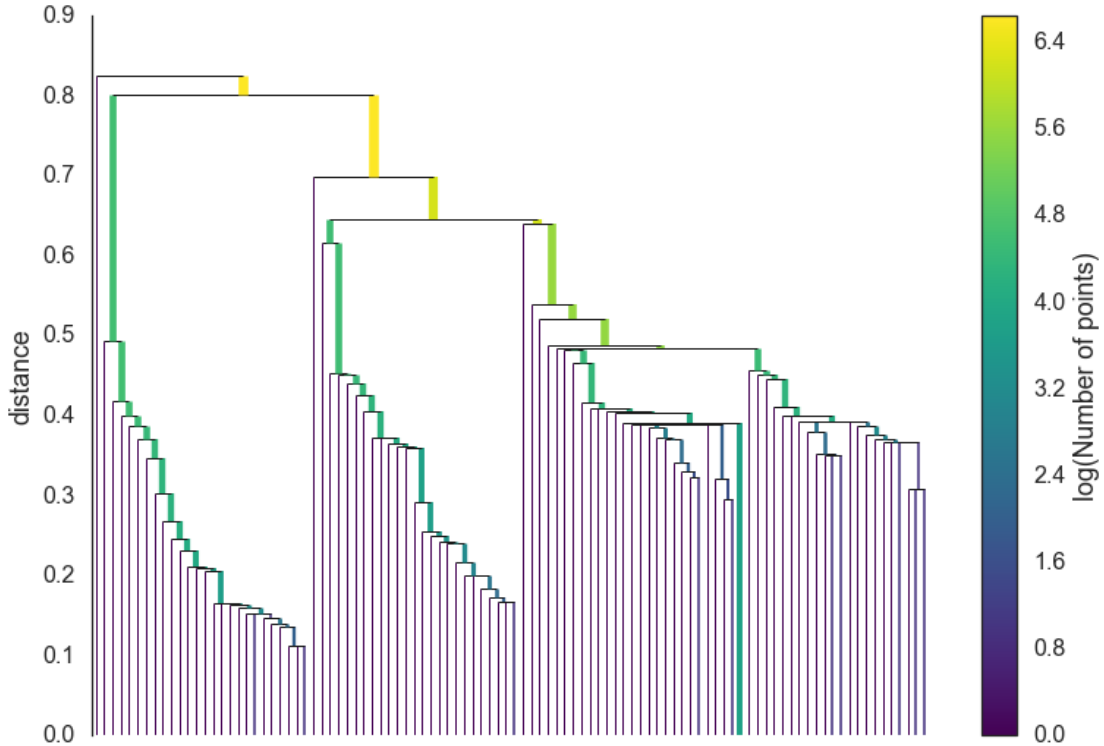
and Borůvka's algorithm

The procedure builds the tree one edge at a time, adding the lowest weighted edge that connects the tree to a vertex not yet in the tree.



Cluster hierarchy

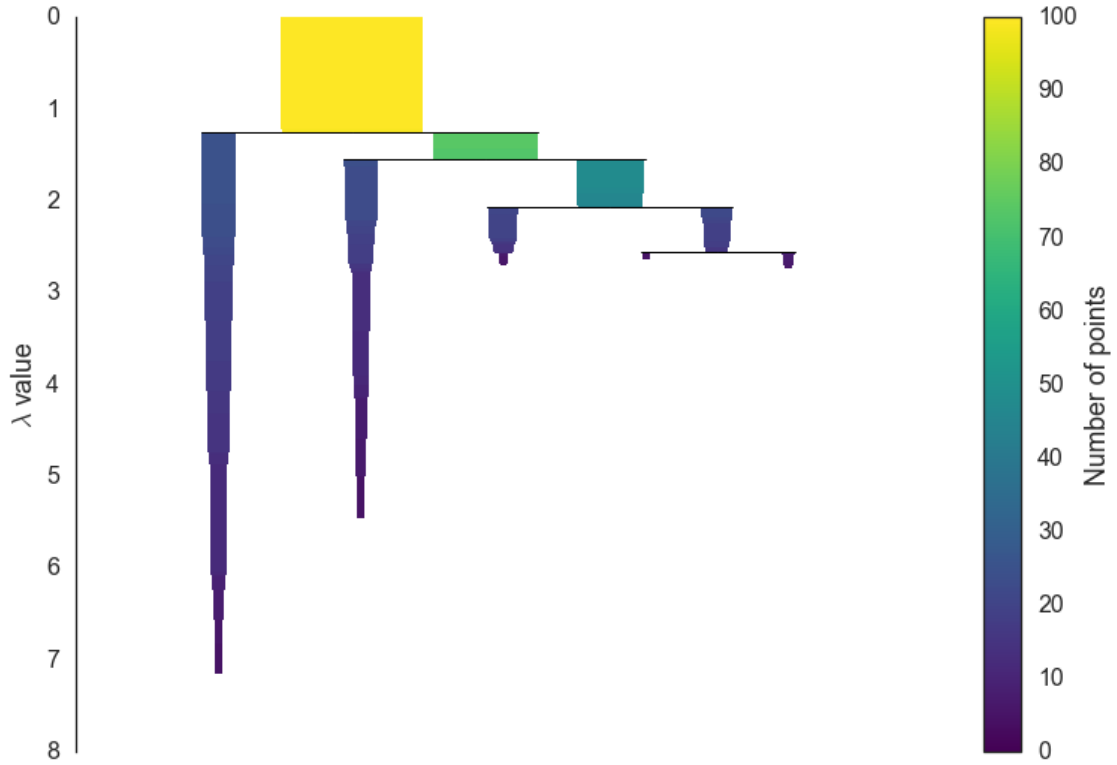
In an agglomerative perspective we can build a hierarchy of clusters:



But we still need a linkage stop condition

Condense the cluster tree

In order to condense the tree we set a minClusterSize, a threshold above which a set of point is considered as cluster.



Flat the clusters

Now we will briefly focus on the stability-based cluster validation method

We define λ as a new quantity called persistence, equal to the inverse of the distance

We can explore the tree Top-Down: at a certain point the tree is splitted and new clusters are created, that point of the tree is the λ_{birth} of the clusters. When the cluster is splitted again, or we reach the end, that point is called λ_{death}.

When a sample falls out of the cluster (because of the distance or because the cluster is splitted in smaller clusters) that point is called λ_p

At the end we can compute the stability as

$$\lambda = \frac{1}{distance}$$

$$\lambda_{birth}, \lambda_{death}, \lambda_p$$

$$stability = \sum_{p \in cluster} (\lambda_p - \lambda_{birth})$$

Flat the clusters

Suppose we have three clusters, one parent (C1) and two childs (C2,C3).

Suppose all leaf nodes are clusters. Now we explore Bottom-Up.

If stability(C2)+ stability(C2) > stability(C1)

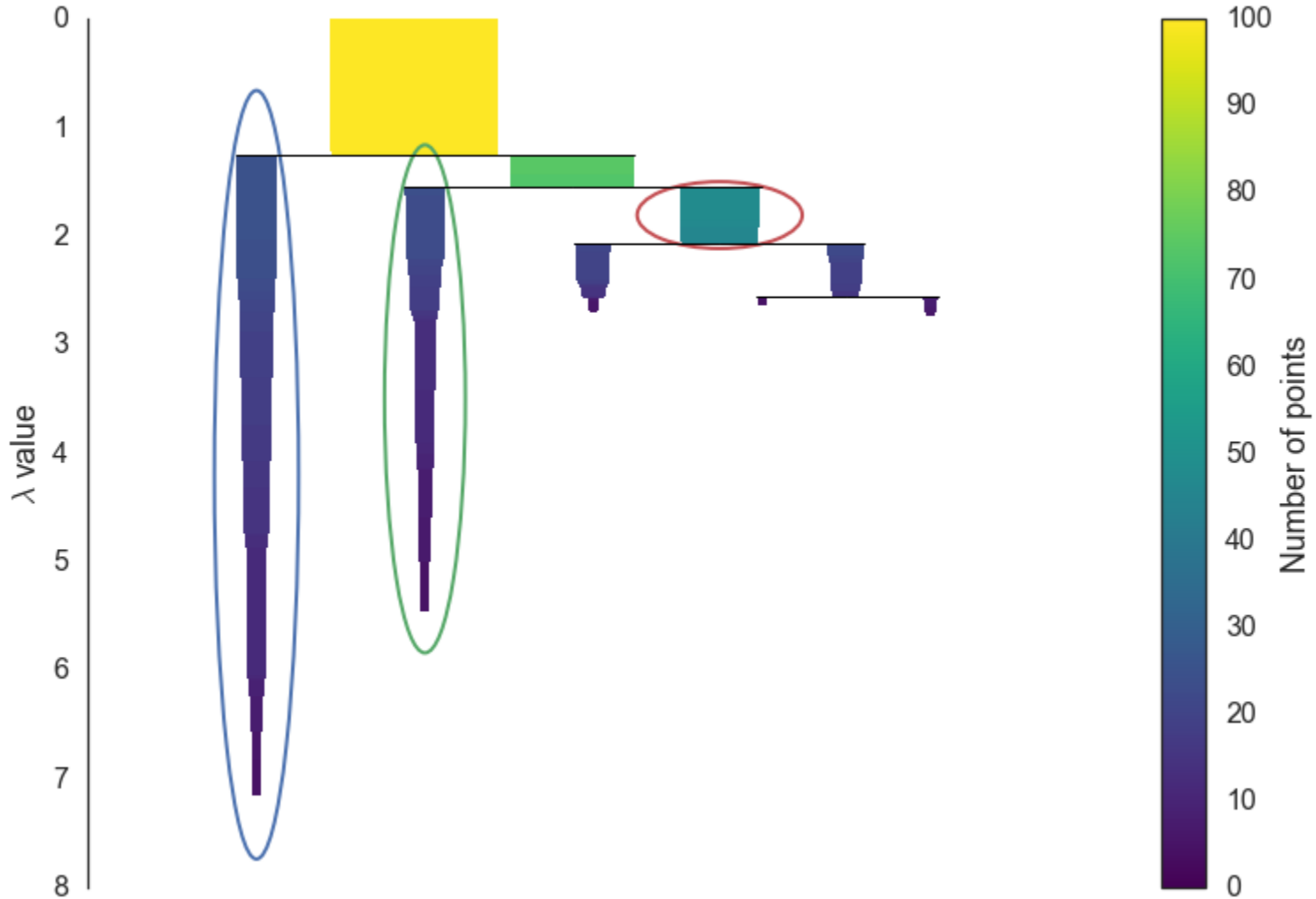
the cluster C1 is not stable

we set stability(C1) = stability(C2)+ stability(C2)

Else if stability(C1) >= stability(C2)+ stability(C2)

C1 is a stable cluster, unset all descendants as clusters

Flat the clusters



HDBSCAN – PROS/CONS

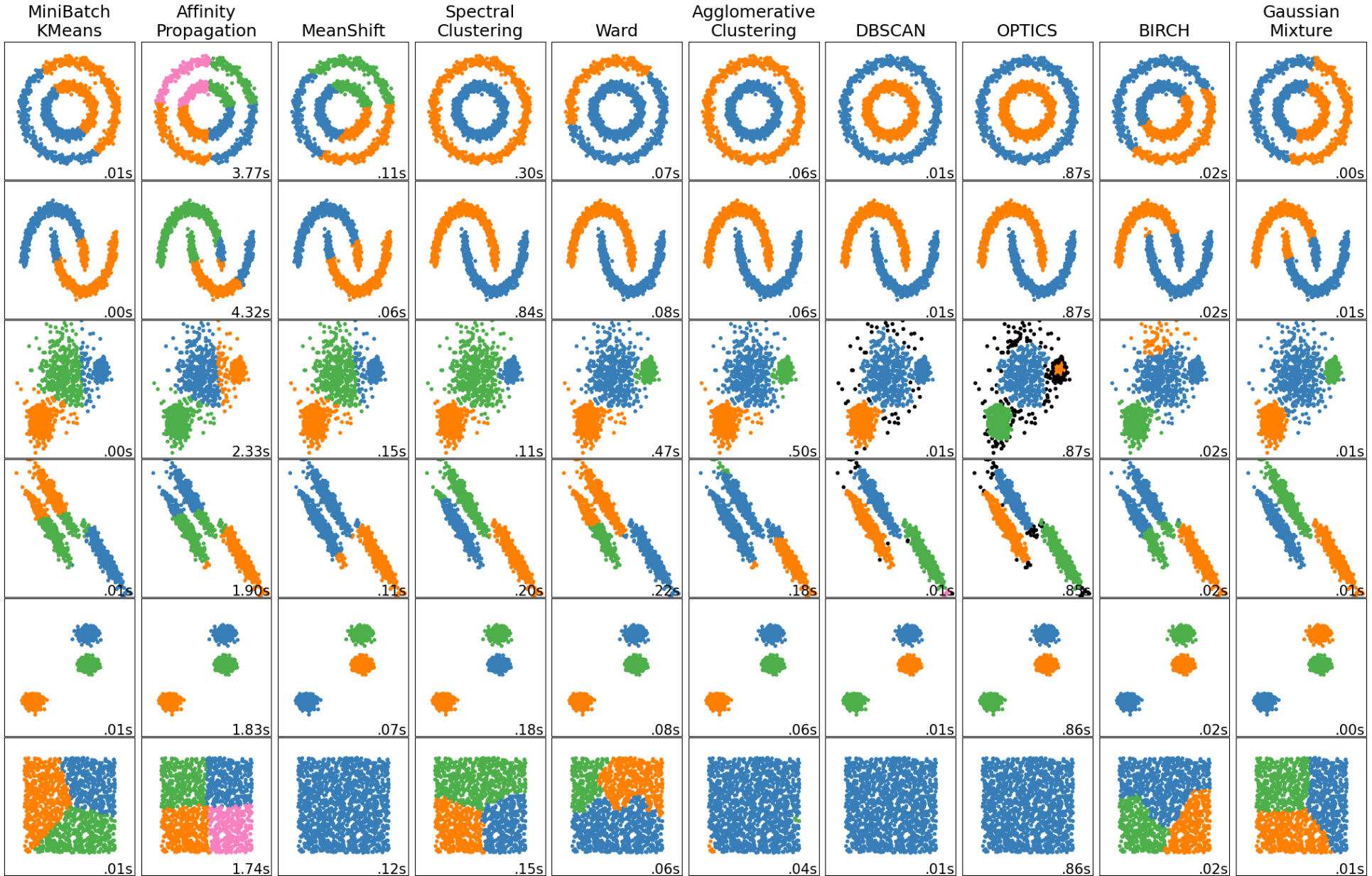
Pros

- Can handle varying densities
- The algorithm is stable over runs and parameter choices
- Robust to outliers

Cons

- Some important parameters must be set by hand

A comparison (implementation in scikit-learn)



you will never know the best algo.
because you dont know the probability distribution of the data.
so its important to use different algoritm and compare the result instead of using only kmeans