DATA MINING 2
Advanced Clustering

Riccardo Guidotti

a.a. 2020/2021

UNIVERSITÀ DI PISA

K-Means Extensions

Bisecting K-Means

Bisecting K-means

- Variant of K-Means that can produce a hierarchical clustering
- The number of clusters K must be specified.
- Start with a unique cluster containing all the points.

```
1: Initialize the list of clusters to contain the cluster containing all points.
2: repeat
3:     Select the cluster with the highest SSE to the list of clusters
4:     for $i = 1$ to $number\_of\_iterations$ do
5:         Bisect the selected cluster using basic 2-Means
6:     end for
7:     Add the two clusters from the bisection to the list of clusters.
8: until Until the list of clusters contains $K$ clusters
```

Bisecting K-means Limitations

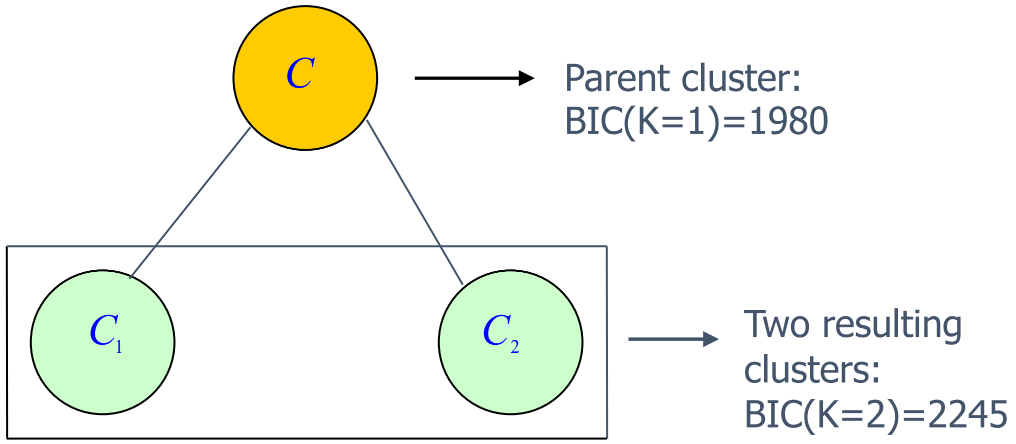
- The algorithm can be also exhaustive and terminating at a singleton clusters if K is not specified.
- Terminating at singleton clusters
  - Is time consuming
  - Singleton clusters are meaningless (i.e., over-splitting)
  - Intermediate clusters are more likely to correspond to real classes
- Bisecting K-Means do not uses any criterion for stopping bisections before singleton clusters are reached.

K-Means Extensions

X-Means

Bayesian Information Criterion (BIC)

- A strategy to stop the Bisecting algorithm when meaningful clusters are reached to avoid over-splitting.
- The **BIC** can be adopted as **splitting criterion** of a cluster in order to decide whether a cluster should split or no.
- BIC **measures the improvement** of the cluster structure between a cluster and its two children clusters.
- If the BIC of the parent is less than BIC of the children than we accept the bisection.


C
Parent cluster: BIC(K=1)=1980
C_1
C_2
Two resulting clusters: BIC(K=2)=2245

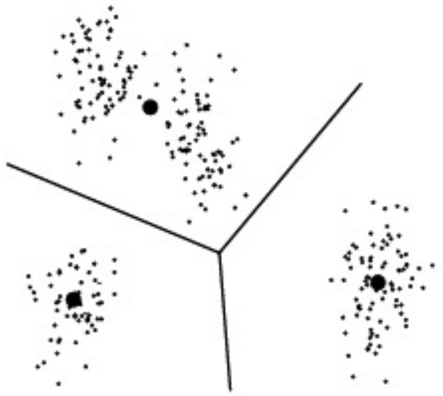
X-Means

For k in a given range $[r_1, r_{max}]$:

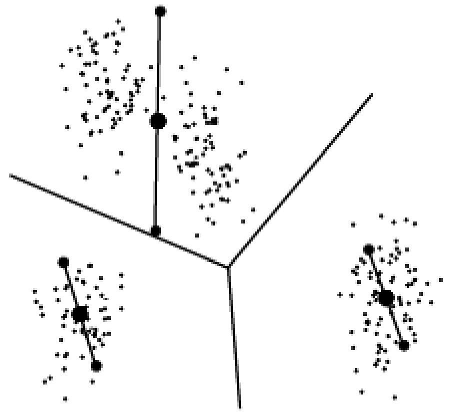
1. Improve Params: run K-Means with with the current k.
2. Improve Structure: recursively split each cluster in two (Bisecting 2-Means) and use *local BIC* to decide to keep the split. Stop if the current structure does not respect *local BIC* or the number of clusters is higher than r_{max}.
3. Store the actual configuration with a global BIC calculated on the whole configuration
4. If $k > r_{max}$ stop and return the best model w.r.t. the *global BIC*.

X-Means

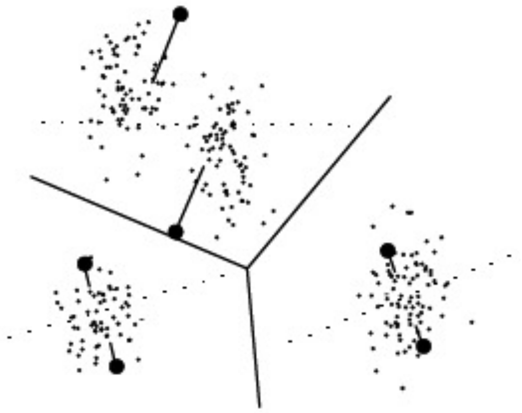
1. K-means with k=3



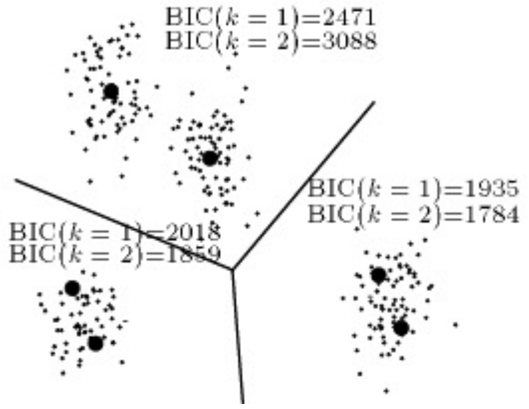
2. Split each centroid in 2 children moved a distance proportional to the region size in opposite direction (random)



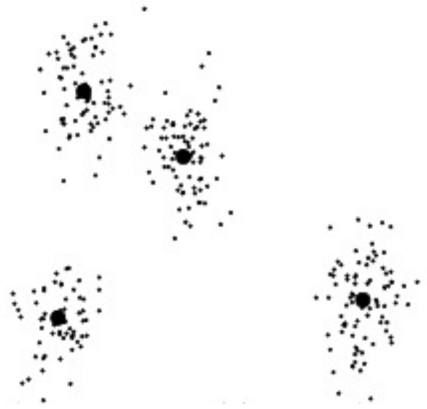
3. Run 2-means in each region locally



4. Compare BIC of parent and children



4. Only centroids with higher BIC survives



BIC Formula in X-Means

- The BIC score of a data collection is defined as (Kass and Wasserman, 1995):

$$BIC(M_j) = \hat{l}_j(D) - \frac{p_j}{2} \log R$$

- $\hat{l}_j(D)$ is the log-likelihood of the dataset D
- p_j is a function of the number of independent parameters: centroids coordinates, variance estimation.
- *R* is the number of points of a cluster, M is the number of dimensions
- Approximate the probability that the clustering in M_j is describing the real clusters in the data

BIC Formula in X-Means

- Adjusted Log-likelihood of the model.
- **The likelihood that the data is “explained by” the clusters** according to the spherical-Gaussian assumption of K-Means

$$BIC(M_j) = \hat{l}_j(D) - \frac{p_j}{2} \log R$$

- Focusing on the set *D_n* of points which belong to centroid *n*

$$\begin{aligned}\hat{l}(D_n) \quad = \quad & -\frac{R_n}{2}\log(2\pi) - \frac{R_n \cdot M}{2}\log(\hat{\sigma}^2) - \frac{R_n - K}{2} \\ & + R_n \log R_n - R_n \log R\end{aligned}$$

- It estimates how closely to the centroid are the points of the cluster.

K-Means Origins

Expectation Maximization

Model-based Clustering (probabilistic)

- In order to understand our data, we will assume that there is a generative process (a model) that creates/describes the data, and we will try to find the model that **best fits** the data.
  - Models of different complexity can be defined, but we will assume that our model is a distribution from which data points are sampled
  - **Example**: the data is the height of all people in **Greece**
- In most cases, a single distribution is not good enough to describe all data points**: different parts of the data follow a different distribution**
  - **Example**: the data is the height of all people in Greece and China
  - We need a mixture model
  - Different distributions correspond to different clusters in the data.

Expectation Maximization Algorithm

- Initialize the values of the parameters in Θ to some random values
- Repeat until convergence
  - **E-Step**: Given the parameters Θ **estimate** the membership probabilities $\mathrm{P}(G_j|x_i)$
  - **M-Step:** Given the probabilities $\mathrm{P}(G_j|x_i)$, calculate the parameter values Θ that (in expectation) **maximize** the data likelihood

- **Examples**
  - **E-Step**: Assignment of points to clusters
    - K-Means: **hard** assignment, EM: soft assignment
  - **M-Step**: Parameters estimation
    - K-Means: Computation of centroids, EM: Computation of the new model parameters

EM in K-Means

centroids

- Initialize the values of the parameters in Θ to some random values (randomly select the centroids)
- Repeat until convergence
  - **E-Step**: Given the parameters Θ (given the centroids) **estimate** the membership probabilities $\mathrm{P}\left(G_j \middle| x_i\right)$ (assign points to clusters based on distances with the centroids)
  - **M-Step:** Given the probabilities $\mathrm{P}\left(G_j \middle| x_i\right)$ (given the membership of points to clusters, i.e., 100% probability of belonging to a cluster) calculate the parameter values Θ that (in expectation) **maximize** the data likelihood (calculate the new centroids as mean values, i.e., those that minimize the distances with the other points in the cluster)

Expectation Maximization Algorithm

Algorithm 9.2 EM algorithm.

```
1: Select an initial set of model parameters.
   (As with K-means, this can be done randomly or in a variety of ways.)
2: repeat
3:     Expectation Step For each object, calculate the probability
       that each object belongs to each distribution, i.e., calculate
       $prob(distribution\ j|\mathbf{x}_i, \Theta)$.
4:     Maximization Step Given the probabilities from the expectation step,
       find the new estimates of the parameters that maximize the expected
       likelihood.
5: until The parameters do not change.
   (Alternatively, stop if the change in the parameters is below a specified
   threshold.)
```

K-Means Brother

Mixture Gaussian Model

Gaussian Distribution

- Example: the data is the height of all people in Greece
- Experience has shown that this data follows a Gaussian (Normal) distribution

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

- μ = mean, σ = standard deviation

Mixture Gaussian Model

- What is a model?
  - A Gaussian distribution is defined by the mean μ and the standard deviation σ
  - We define our model as the pair of parameters $\theta = (\mu, \sigma)$
- More generally, a model is defined as a vector of parameters θ

- We want to find the normal distribution $N(\mu, \sigma)$ that best fits our data
  - Find the best values for μ and σ
  - But what does "best fit" mean?

Maximum Likelihood Estimation (MLE)

- Suppose that we have a vector $X = \{x_1, \dots, x_n\}$ of values
- We want to fit a Gaussian model $N(\mu, \sigma)$ to the data
- Probability of observing a point x_i

$$P(x_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

- Probability of observing all points (we assume independence)

$$P(X) = \prod_{i=1}^{n} P(x_i) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

- We want to find the parameters $\theta = (\mu, \sigma)$ that maximizes the probability $P(X|\theta)$

Maximum Likelihood Estimation (MLE)

- The probability $P(X|\theta)$ as a function of θ is the **Likelihood** function

$$L(\theta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

- It is usually easier to work with the **Log-Likelihood** function

$$LL(\theta) = -\sum_{i=1}^{n} \frac{(x_i-\mu)^2}{2\sigma^2} - \frac{1}{2} n \log 2\pi - n \log \sigma$$

- Thus, the Maximum Likelihood Estimation for the Gaussian Model consists in finding the parameters μ, σ that maximize $LL(\theta)$

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i = \mu_X$$

Sample Mean

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} (x_i-\mu)^2 = \sigma_X^2$$

Sample Variance

Maximum Likelihood Estimation (MLE)

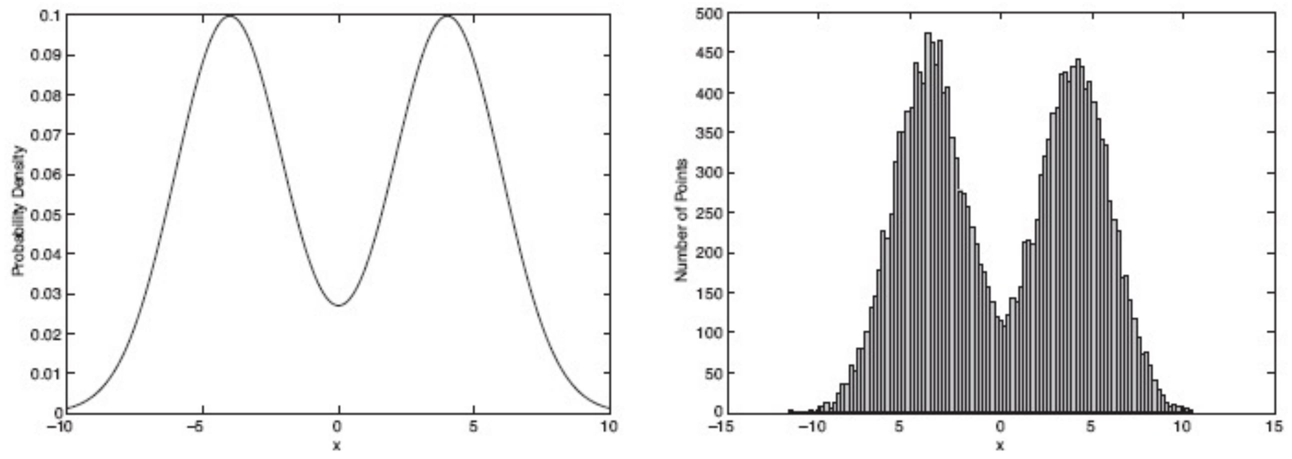
- Note: these are also the most likely parameters given the data.

$$P(\theta|X) = \frac{P(X|\theta)P(\theta)}{P(X)}$$

- If we have no prior information about θ, or X, then maximizing $P(\theta|X)$ is the same as maximizing $P(X|\theta)$.

Mixture of Gaussians

- Suppose that you have the heights of people from Greece and China and the distribution looks like the figure below (dramatization)



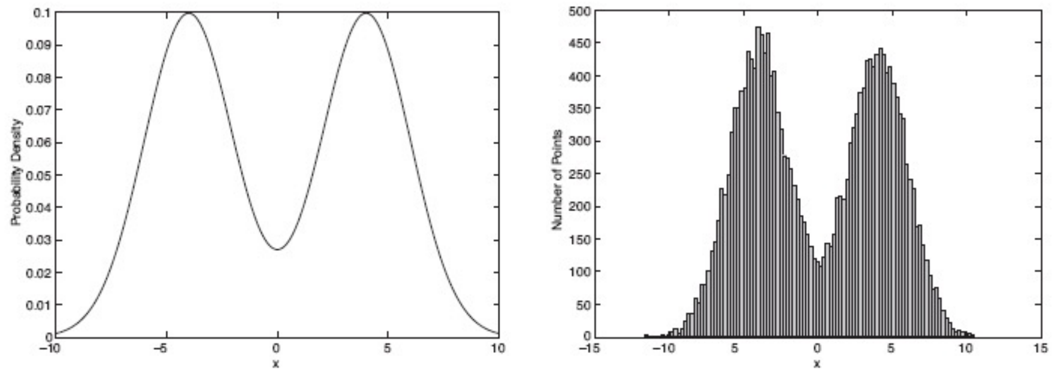
(a) Probability density function for the mixture model.

(b) 20,000 points generated from the mixture model.

Figure 9.2. Mixture model consisting of two normal distributions with means of -4 and 4, respectively. Both distributions have a standard deviation of 2.

Mixture of Gaussians

- In this case the data is the result of the **mixture** of two Gaussians
  - One for Greek people, and one for Chinese people
  - Identifying for each value which Gaussian is most likely to have generated it will give us a clustering.



(a) Probability density function for the mixture model.

(b) 20,000 points generated from the mixture model.

Figure 9.2. Mixture model consisting of two normal distributions with means of -4 and 4, respectively. Both distributions have a standard deviation of 2.

Mixture Model

- A value x_i is generated according to the following process:
  - First select the nationality
    - With probability π_G select Greek, with probability π_C select China ($\pi_G + \pi_C = 1$)
  - Given the nationality, generate the point from the corresponding Gaussian
    - $P(x_i|\theta_G) \sim N(\mu_G, \sigma_G)$ if Greece
    - $P(x_i|\theta_C) \sim N(\mu_C, \sigma_C)$ if China

Mixture Model

Assign a point to a cluster. In K-Means they are the membership: hard assignment.

Describe a cluster. In K-Means they are the centroids.

- Our model has the following parameters

$$\Theta = (\underbrace{\pi_G, \pi_C}_{\text{Mixture probabilities}}, \underbrace{\mu_G, \mu_C, \sigma_G, \sigma_C}_{\text{Distribution Parameters}})$$

- For value x_i, we have:

$$P(x_i|\Theta) = \pi_G P(x_i|\theta_G) + \pi_C P(x_i|\theta_C)$$

- For all values $X = \{x_1, \ldots, x_n\}$

$$P(X|\Theta) = \prod_{i=1}^{n} P(x_i|\Theta)$$

- We want to estimate the parameters that **maximize** the Likelihood

Mixture Model

- Once we have the parameters $\theta = (\pi_G, \pi_C, \mu_G, \sigma_G, \mu_C, \sigma_C)$, we can **estimate** the **membership probabilities** $P(G|x_i)$ and $P(C|x_i)$ for each point x_i:

- This is the probability that point x_i belongs to the Greek or the Chinese population (cluster)

$$P(G|x_i) = \frac{P(x_i|G)P(G)}{P(x_i|G)P(G) + P(x_i|C)P(C)}$$
$$= \frac{P(x_i|G)\pi_G}{P(x_i|G)\pi_G + P(x_i|C)\pi_C}$$

Mixture of Gaussians as EM

- Initialize the values of the parameters in θ to some random values
- Repeat until convergence
  - **E-Step**: Given the parameters Θ **estimate** the membership probabilities $\mathrm{P}(G|x_i)$ and $\mathrm{P}(C|x_i)$.
  - **M-Step:** Calculate the parameter values Θ that (in expectation) **maximize** the data likelihood.

$$\pi_G = \frac{1}{n}\sum_{i=1}^{n} P(G|x_i) \qquad \pi_C = \frac{1}{n}\sum_{i=1}^{n} P(C|x_i)$$

Fraction of population in G,C

$$\mu_C = \sum_{i=1}^{n} \frac{P(C|x_i)}{n * \pi_C} x_i \qquad \mu_G = \sum_{i=1}^{n} \frac{P(G|x_i)}{n * \pi_G} x_i$$

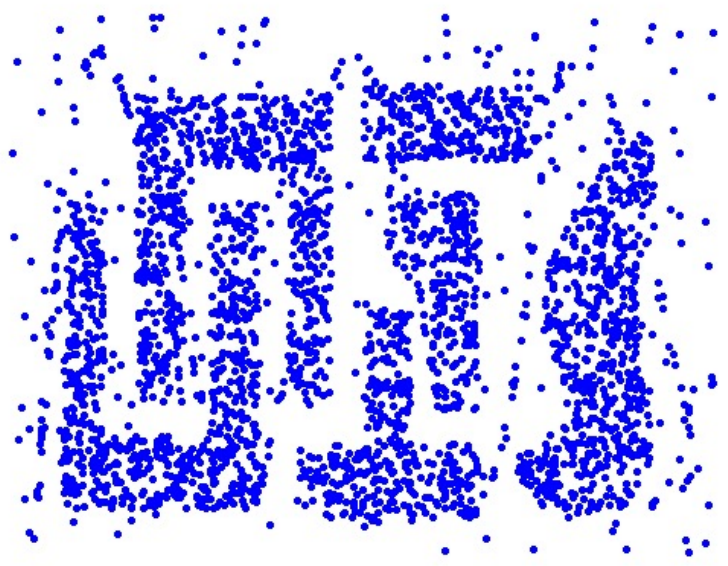
$$\sigma_C^2 = \sum_{i=1}^{n} \frac{P(C|x_i)}{n * \pi_C} (x_i - \mu_C)^2 \qquad \sigma_G^2 = \sum_{i=1}^{n} \frac{P(G|x_i)}{n * \pi_G} (x_i - \mu_G)^2$$

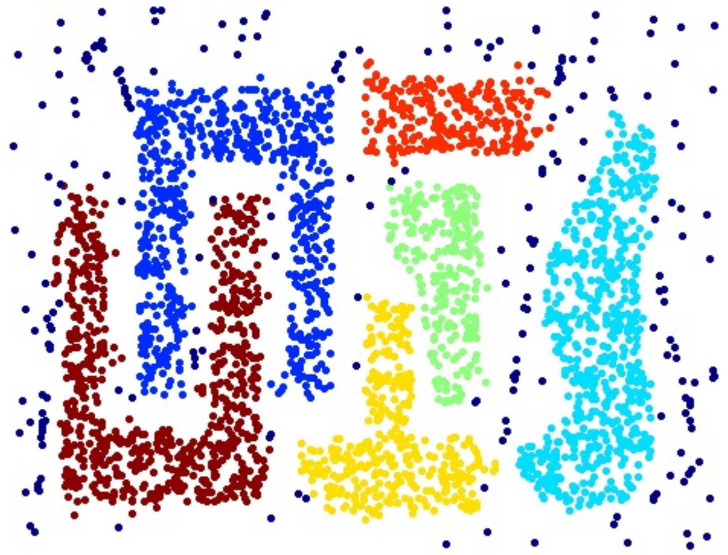
MLE Estimates if π's were fixed

DBSCAN Evolution

OPTICS

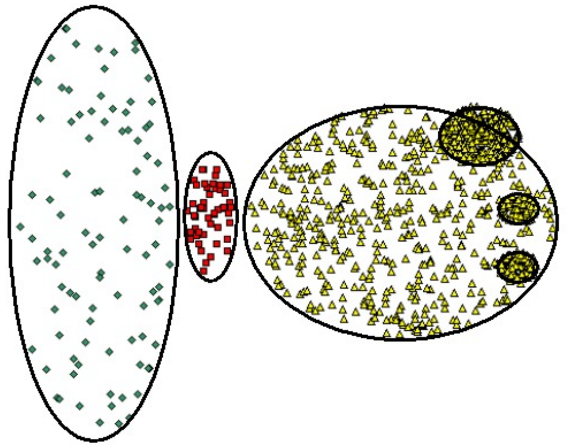
When DBSCAN Works Well


Original Points


Clusters

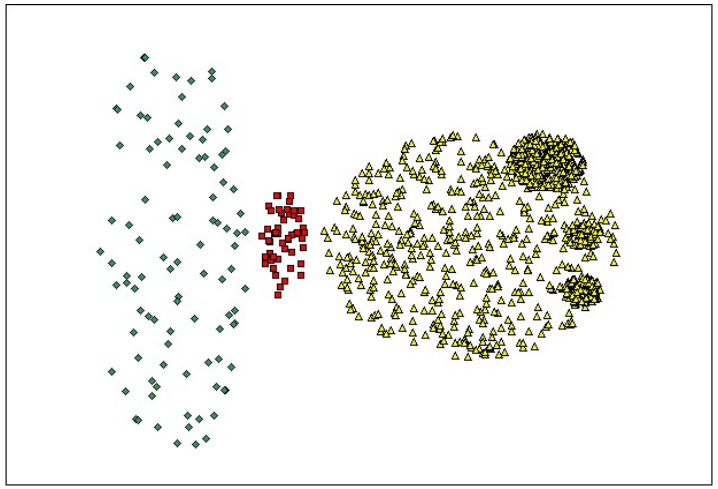
- **Resistant to Noise**
- **Can handle clusters of different shapes and sizes**

When DBSCAN Does NOT Work Well

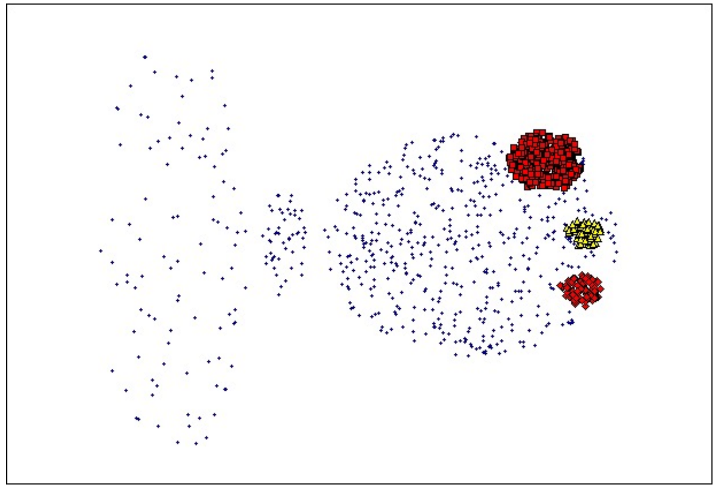


Original Points

- **Varying densities**
- **High-dimensional data**



(MinPts=4, Eps=9.75).



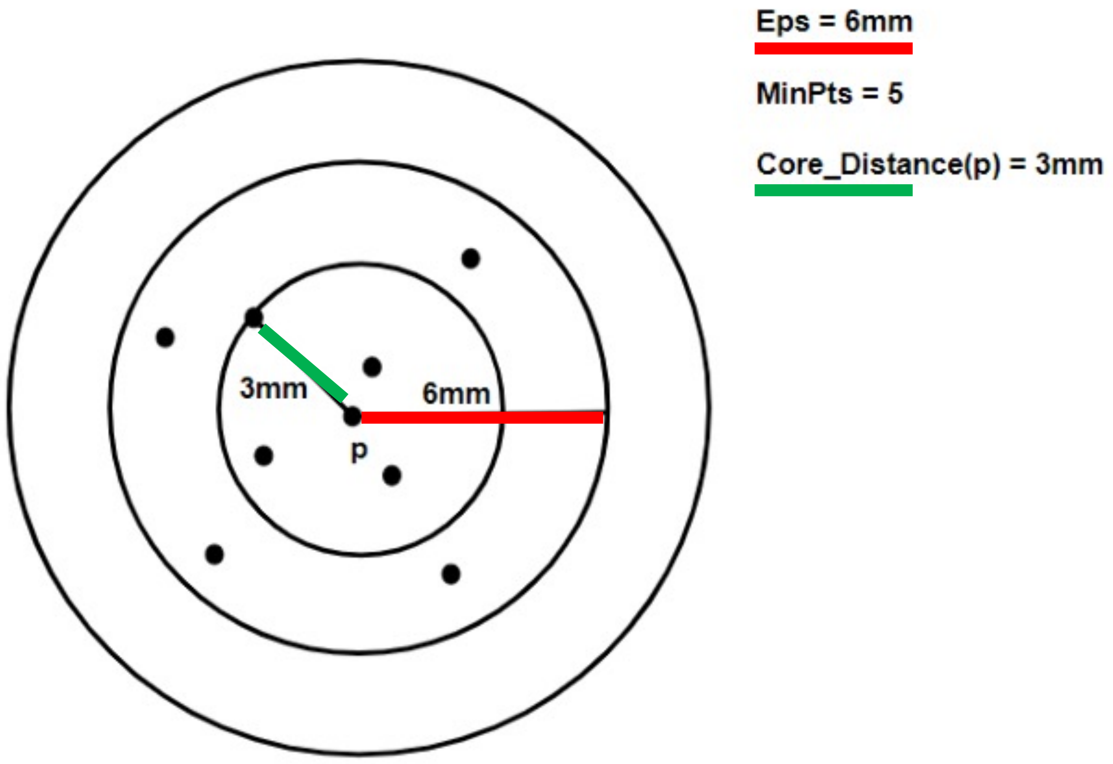
(MinPts=4, Eps=9.92)

OPTICS

- OPTICS: Ordering Points To Identify the Clustering Structure
  - Produces a special order of the dataset wrt its density-based clustering structure.
  - This cluster-ordering contains info equivalent to the density-based clusterings corresponding to a broad range of parameter settings.
  - Good for both automatic and interactive cluster analysis, including finding intrinsic clustering structure.
  - Can be represented graphically or using visualization techniques.

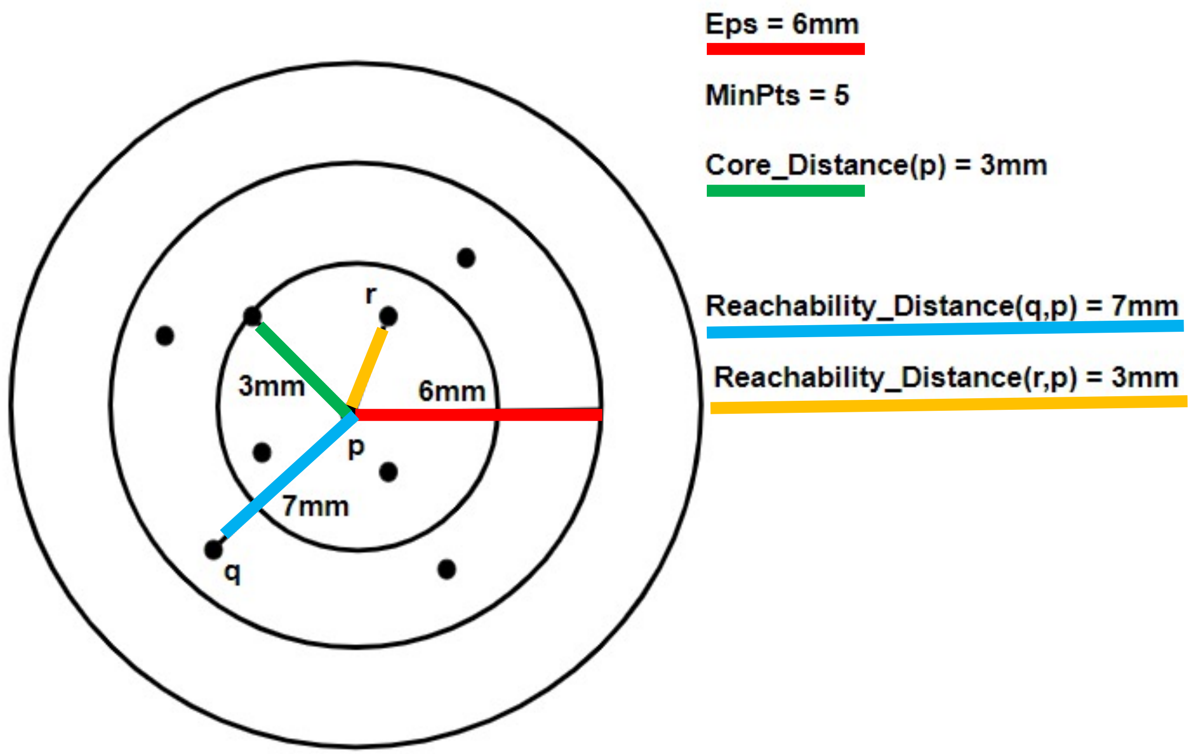
OPTICS: Extension from DBSCAN

- OPTICS requires two **parameters**:
  - ε, which describes the maximum distance (radius) to consider,
  - MinPts, describing the number of points required to form a cluster
- **Core point**. A point *p* is a core point if at least MinPts points are found within its ε-neighborhood.
- **Core Distance**. It is the **minimum** value of radius required to classify a given point as a core point. If the given point is not a Core point, then it's Core Distance is undefined.



OPTICS: Extension from DBSCAN

- **Reachability Distance**. The reachability distance between a point *p* and *q* is the **maximum** of the Core Distance of *p* and the Distance between p and q.
- The Reachability Distance is not defined if *q* is not a Core point. Below is the example of the Reachability Distance.
- In other words, if *q* is within the core distance of *p* then use the core distance, otherwise the real distance.


Eps = 6mm
MinPts = 5
Core_Distance(p) = 3mm
Reachability_Distance(q,p) = 7mm
Reachability_Distance(r,p) = 3mm
r
3mm
6mm
p
7mm
q

OPTICS Pseudo-Code

- For each point *p* in the dataset
  - Initialize the reachability distance of *p* as undefined
- For each unprocessed point *p* in the dataset
  - Get the neighbors *N* of p
  - Mark *p* as processed and output to the *ordered list*
  - If *p* is a core point
    - Initialize a priority queue *Q* to get the closest point to *p* in terms of reachability
    - Call the function *update(N, p, Q)*
    - For each point *q* in *Q*
      - Get the neighbors *N’* of *q*
      - Mark q as processed and output to the *ordered list*
        - If *q* is a core point Call the function update(N’, q, Q)

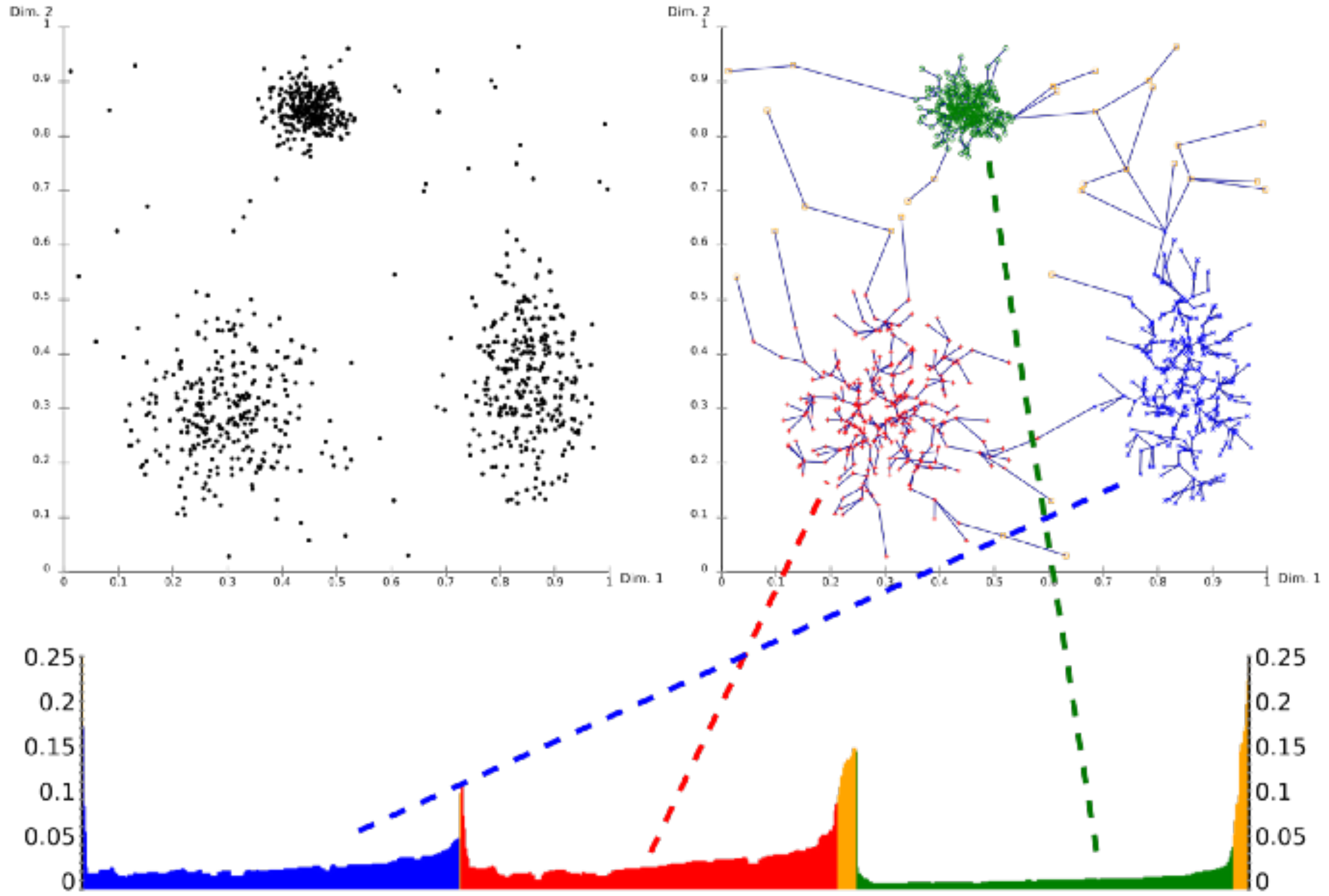
OPTICS Pseudo-Code

- Function *update(N, p Q)*
  - Calculate the core distance for *p*
  - For each neighbor *q in N* (update the reachability)
    - If *q* is not processed
      - *new_rd* = reachability distance between *p* and *q*
    - If *q is not in Q*
      - *Q.insert(q, new_rd)*
    - Else
      - *If new_rd < q.rd*
        - *Q.move_up(q, new_rd)*

OPTICS Output

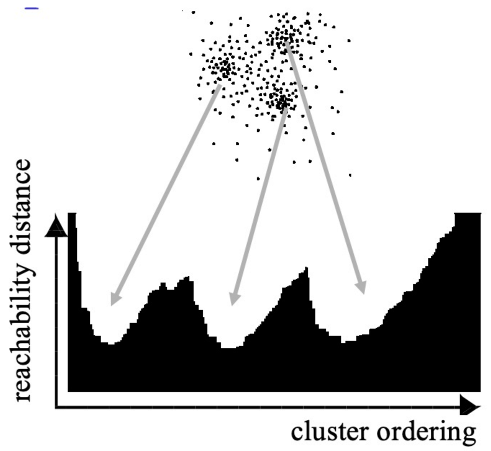
- OPTICS outputs the points in a particular ordering, annotated with their smallest reachability distance.
- A reachability-plot (a special kind of dendrogram), the hierarchical structure of the clusters can be obtained easily.
- x-axis: the ordering of the points as processed by OPTICS
- y-axis: the reachability distance
- Points belonging to a cluster have a low reachability distance to their nearest neighbor, the clusters show up as valleys in the reachability plot. The deeper the valley, the denser the cluster.

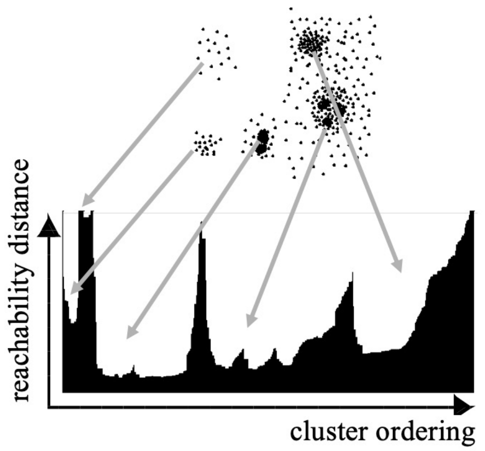
OPTICS Output



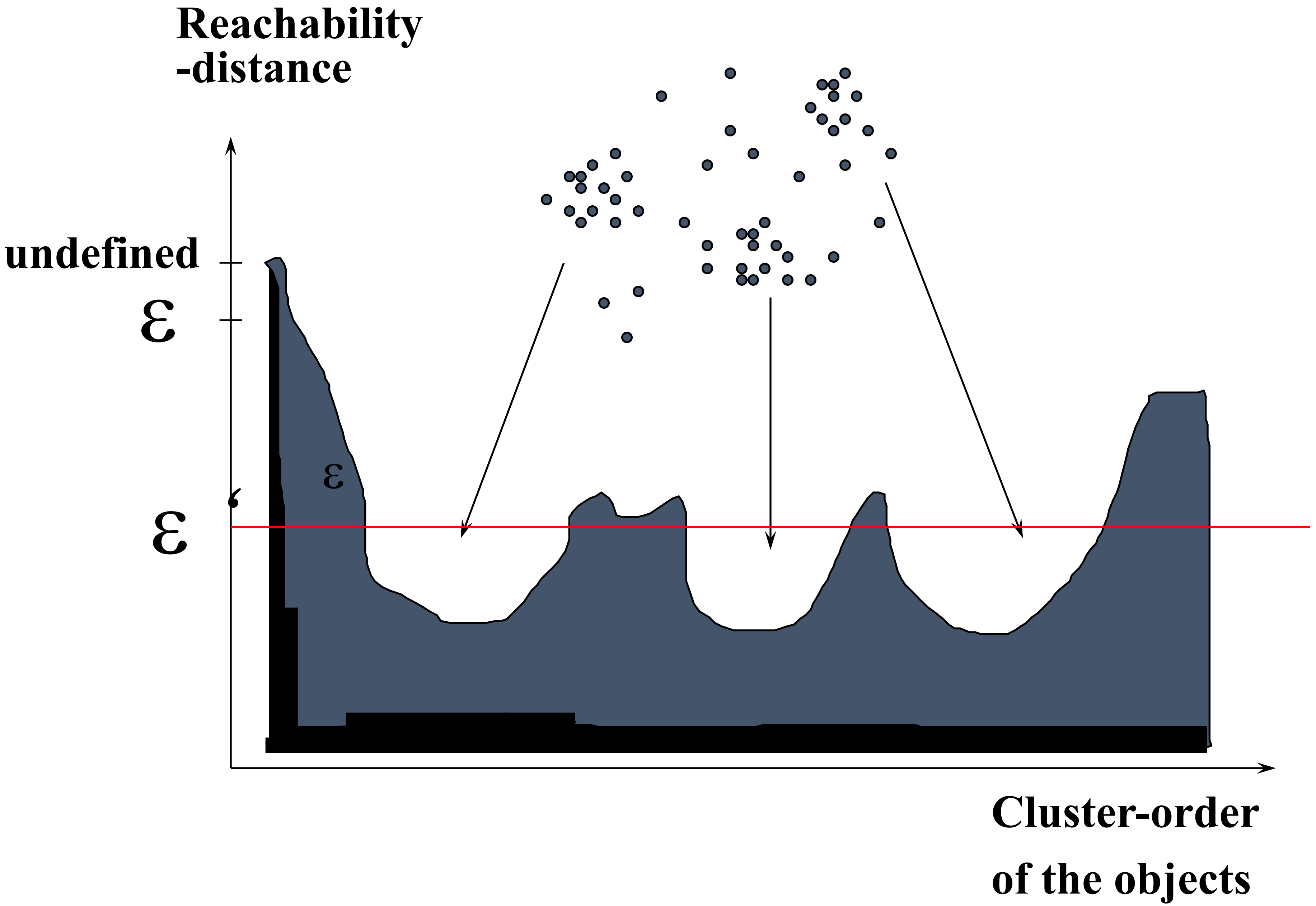
OPTICS Output

- Clusters are extracted
  1. by selecting a range on the x-axis after visual inspection,
  2. by selecting a threshold on the y-axis
  3. by different algorithms that try to detect the valleys by steepness, knee detection, or local maxima. Clustering obtained this way usually are hierarchical, and cannot be achieved by a single DBSCAN run.





https://scikit-learn.org/stable/auto_examples/cluster/plot_optics.html#sphx-glr-auto-examples-cluster-plot-optics-py



Reachability
-distance
undefined
ε
ε
ε'
ε
Cluster-order
of the objects

OPTICS: The Radius Parameter

- Both core-distance and reachability-distance are undefined if no sufficiently dense cluster (w.r.t. ε) is available.
- Given a sufficiently large ε, this never happens, but then every ε-neighborhood query returns the entire database.
- Hence, the ε parameter is required to cut off the density of clusters that are no longer interesting, and to speed up the algorithm.
- The parameter ε is, strictly speaking, not necessary.
- It can simply be set to the maximum possible value.
- When a spatial index is available, however, it does play a practical role with regards to complexity.
- OPTICS abstracts from DBSCAN by removing this parameter, at least to the extent of only having to give the maximum value.

References

- Advanced Clustering. Chapter 9.2.2. Introduction to Data Mining.
- Pelleg, Dan, and Andrew W. Moore. "X-means: Extending k-means with efficient estimation of the number of clusters." 2000.
- Mihael Ankerst; Markus M. Breunig; Hans-Peter Kriegel; Jörg Sander (1999). OPTICS: Ordering Points To Identify the Clustering Structure.

X-means: Extending K-means with Efficient Estimation of the Number of Clusters

Dan Pelleg DPELLEG@CS.CMU.EDU
Andrew Moore AWM@CS.CMU.EDU
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 USA

Abstract

Despite its popularity for general clustering, K-means suffers three major shortcomings; it scales poorly computationally, the number of clusters K has to be supplied by the user, and the search is prone to local minima. We propose solutions for the first two problems, and a partial remedy for the third. Building on prior work for algorithmic acceleration that is not based on approximation, we introduce a new algorithm that efficiently, searches the space of cluster locations and number of clusters to optimize the Bayesian Information Criterion (BIC) or the Akaike Information Criterion (AIC) measure. The innovations include two new ways of exploiting cached sufficient statistics and a new very efficient test that in one K-means sweep selects the most promising subset of classes for refinement. This gives rise to a fast, statistically founded algorithm that outputs both the number of classes and their parameters. Experiments show this technique reveals the true number of classes in the underlying distribution, and that it is much faster than repeatedly using accelerated K-means for different values of K.

1. Introduction

K-means (Duda & Hart, 1973; Bishop, 1995) has long been the workhorse for metric data. Its attractiveness lies in its simplicity, and in its local-minimum convergence properties. It has, however, three main shortcomings. One, it is slow and scales poorly with

lutions for these problems. Speed is greatly improved by embedding the dataset in a multiresolution *kd*-tree and storing sufficient statistics at its nodes. A careful analysis of the centroid locations allows for geometric "proofs" about the Voronoi boundaries, and (unlike all of (Deng & Moore, 1995; Zhang et al., 1995; Moore, 1999)) there is absolutely no approximation anywhere in the computation. An additional geometric computation, *blacklisting*, maintains a list of just those centroids that need to be considered for a given region (Pelleg & Moore, 2000). Blacklisting is not only extremely fast but also scales very well with the number of centroids, allowing tractable 10, 000-means algorithms. This fast algorithm is used as a building-block in X-means: a new algorithm that quickly estimates K. It goes into action after each run of K-means, making local decisions about which subset of the current centroids should split themselves in order to better fit the data. The splitting decision is done by computing

introduction to
DATA MINING
SECOND EDITION
Pang-Ning Tan Michael Steinbach Anuj Karpatne Vipin Kumar

Pergamon

Information Systems Vol. 25, No. 5, pp. 345-366, 2000
© 2000 Elsevier Science Ltd. All rights reserved
Printed in Great Britain
0306-4379/00 $20.00 + 0.00

ROCK: A ROBUST CLUSTERING ALGORITHM FOR CATEGORICAL ATTRIBUTES†

SUDIPTO GUHA[1], RAJEEV RASTOGI[2], and KYUSEOK SHIM[3]

[1]Stanford University, Stanford, CA 94305, USA
[2]Bell Laboratories, Murray Hill, NJ 07974, USA
[3]Korea Advanced Institute of Science and Technology and Advanced Information Technology Research Center, Taejon 305-701, Korea

31 May 2000)

Clustering Individual Transactional Data for Masses of Users

Riccardo Guidotti, ISTI-CNR & University of Pisa, Italy, riccardo.guidotti@isti.cnr.it
Anna Monreale, University of Pisa, Italy, anna.monreale@di.unipi.it
Mirco Nanni, ISTI-CNR, Pisa, Italy, mirco.nanni@isti.cnr.it
Fosca Giannotti, ISTI-CNR, Pisa, Italy, fosca.giannotti@isti.cnr.it
Dino Pedreschi, University of Pisa, Italy, dino.pedreschi@di.unipi.it

ABSTRACT

Mining a large number of datasets recording human activities for making sense of individual data is the key enabler of a new wave of personalized knowledge-based services. In this paper we focus on the problem of clustering individual transactional data for a large mass of users. Transactional data is a very pervasive kind of information that is collected by several services, often involving huge pools of users. We propose *txmeans*, a parameter-free clustering algorithm able to efficiently partitioning transactional data in a completely automatic way. *Txmeans* is designed for the case where clustering must be applied on a massive number of different datasets, for instance when a large set of users need to be analyzed individually and each of them has generated a long history of transactions. A deep experimentation on both real and synthetic datasets shows the practical effectiveness of *txmeans* for the mass clustering of different personal datasets, and suggests that *txmeans* outperforms existing methods in terms of quality and efficiency. Finally, we present a *personal cart assistant* application based on *txmeans*.

1 INTRODUCTION

The most disruptive effect of our always-connected society is *data*, the digital breadcrumbs left behind us as a side effect of our everyday usage of digital technologies. Thanks to these data, human activities are becoming observable, measurable, quantifiable and, predictable. At individual level, each person generates more than 5Gb of data per year. An avalanche of information that, for the most part, consists of *transactions* (or baskets), i.e., a special kind of categorical data in the form of sets of event data, such as the items purchased in a shopping cart, the web pages visited in a browsing session, the songs listened in a time period, the clinical events in a patient's history. Such kind of data may be key enablers of a new wave of knowledge-based services, and of new scientific discoveries.

Several application contexts involve the analysis of a large number of datasets, each one characterized by different properties. For instance, this is the case of individual transactional data – retail sales, web sessions, credit card transactions, etc. – where each user produces historical data that need to be analyzed separately

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD '17, August 13-17, 2017, Halifax, NS, Canada
© 2017 ACM. 978-1-4503-4887-4/17/08…$15.00
DOI: 10.1145/3097983.3098034

from other users. This requires tha
included in any data mining metho
the necessity to automatically capt
vidual behaviors. Due to the poten
(e.g. users in nowadays massive sy
generally unfeasible to determine in
parameter configuration for each of
focus data mining methods that adju
characteristics of the dataset under a
personalized patterns from transact
In this paper we focus on the p
tional clustering for a large numbe
collection of transactions, transacti
covering groups of homogeneous t
common items [30]. In the state
transactional clustering require eit
that is not automatic, or an extren
that does not scale to large user b
repeatedly applying the existing pr
lions of different datasets – which i
large population of users – is simp
problem, i.e., the separate individua
transactional datasets, as *mass clust*
The problem to design parameter
been addressed in the context of non
like *xmeans* [22], which are perfe
of the clustering problems. Unfort
applicable to transactional data. To
only existing parameter-free transac
[5, 7]. Nevertheless, they are based
generally not efficient and overestim
In addition, they do not provide repr
items that characterize the transact
In this paper we propose *txmean*
ing method providing a viable soluti
a massive number of different dat
strategy similar to *xmeans* [22], bu
finding clusters in the specific cont
Txmeans overcomes the deficienc
it automatically estimates the num
tracting the clusters, it provides th
each cluster, which summarizes the
Txmeans employs a top-down divisi
a unique cluster, and then iterativ
sub-clusters. *Txmeans* calculates th
centroids of the sub-clusters by ado

CLOPE: A Fast and Effective Clustering Algorithm for Transactional Data

Yiling Yang Xudong Guan Jinyuan You
Dept. of Computer Science & Engineering., Shanghai Jiao Tong University
Shanghai, 200030, P.R.China
+86-21-52581638
{yang-yl, guan-xd, you-jy}@cs.sjtu.edu.cn

ABSTRACT

This paper studies the problem of categorical data clustering, especially for transactional data characterized by high dimensionality and large volume. Starting from a heuristic method of increasing the height-to-width ratio of the cluster histogram, we develop a novel algorithm – CLOPE, which is very fast and scalable, while being quite effective. We demonstrate the performance of our algorithm on two real world datasets, and compare CLOPE with the state-of-art algorithms.

Keywords

data mining, clustering, categorical data, scalability

1. INTRODUCTION

Clustering is an important data mining technique that groups together similar data records [12, 14, 4, 1]. Recently, more attention has been put on clustering categorical data [10, 8, 6, 5, 7, 13], where records are made up of non-numerical attributes. Transactional data, like market basket data and web usage data, can be thought of a special type of categorical data having boolean value, with all the possible items as attributes. Fast and accurate clustering of transactional data has many potential applications in retail industry, e-commerce intelligence, etc.

However, fast and effective clustering of transactional databases is extremely difficult because of the high dimensionality, sparsity, and huge volumes often characterizing these databases. Distance-based approaches like k-means [11] and CLARANS [12] are effective for low dimensional numerical data. Their performances on high dimensional categorical data, however, are often unsatisfactory [7]. Hierarchical clustering methods like ROCK [7] have been demonstrated to be quite effective in categorical data clustering, but they are naturally inefficient in processing large databases.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGKDD '02, July 23-26, 2002, Edmonton, Alberta, Canada.
Copyright 2002 ACM 1-58113-567-X/02/0007…$5.00.

The LargeItem [13] algorithm groups large categorical databases by iterative optimization of a *global* criterion function. The criterion function is based on the notion of *large item* that is the item in a cluster having occurrence rates larger than a user-defined parameter *minimum support*. Computing the global criterion function is much faster than those *local* criterion functions defined on top of pair-wise similarities. This global approach makes LargeItem very suitable for clustering large categorical databases.

In this paper, we propose a novel global criterion function that tries to increase the intra-cluster overlapping of transaction items by increasing the height-to-width ratio of the cluster *histogram*. Moreover, we generalize the idea by introducing a parameter to control the tightness of the cluster. Different number of clusters can be obtained by varying this parameter. Experiments show that our algorithm runs much faster than LargeItem, with clustering quality quite close to that of the ROCK algorithm [7].

To gain some basic idea behind our algorithm, let's take a small market basket database with 5 transactions {(*apple*, *banana*), (*apple*, *banana*, *cake*), (*apple*, *cake*, *dish*), (*dish*, *egg*), (*dish*, *egg*, *fish*)}. For simplicity, transaction (*apple*, *banana*) is abbreviated to *ab*, etc. For this small database, we want to compare the following two clustering (1) {{*ab*, *abc*, *acd*}, {*de*, *def*}} and (2) {{*ab*, *abc*}, {*acd*, *de*, *def*}}. For each cluster, we count the occurrence of every distinct item, and then obtain the height (H) and width (W) of the cluster. For example, cluster {*ab*, *abc*, *acd*} has the occurrences of a:3, b:2, c:2, and d:1, with H=2.0 and W=4. Figure 1 shows these results geometrically as histograms, with items sorted in reverse order of their occurrences, only for the sake of easier visual interpretation.

Figure 1. Histograms of the two clusterings.

We judge the qualities of these two clusterings geometrically, by analyzing the heights and widths of the clusters. Leaving out the two identical histograms for cluster {*de*, *def*} and cluster {*ab*, *abc*}, the other two histograms are of different quality. The histogram for cluster {*ab*, *abc*, *acd*} has only 4 distinct items for 8 blocks (H=2.0, H/W=0.5), but the one for cluster {*acd*, *de*, *def*} has 5, for