

DATA MINING 2

Clustering – Advanced Topics

Riccardo Guidotti

a.a. 2019/2020



Mixture Models and the EM Algorithm

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.

EM 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 $prob(\text{distribution } j | \mathbf{x}_i, \Theta)$.
 - 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.)
-

EM (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 $P(G_j|x_i)$
 - **M-Step:** Given the probabilities $P(G_j|x_i)$, 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

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)

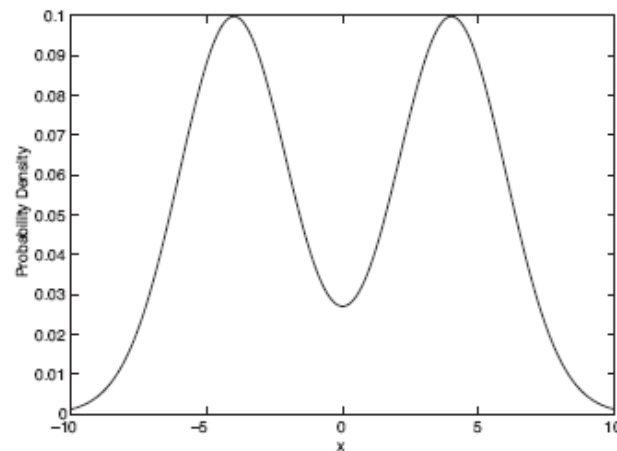
- Note: these are also the most likely parameters given the data.

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

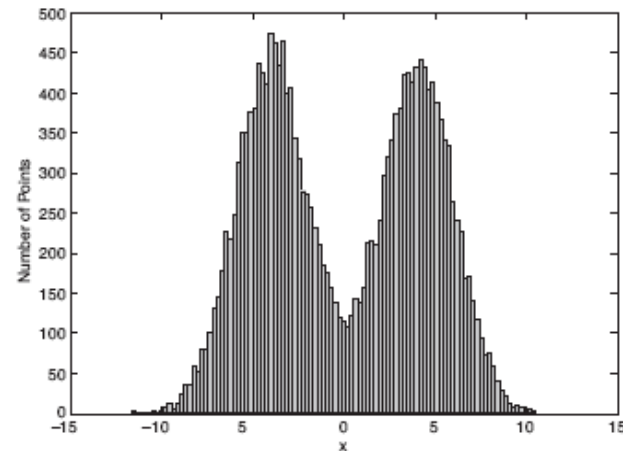
- If we have no prior information about θ , or X , then maximizing $P(\theta|X)$ is the same as maximizing $P(X|\theta)$.

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.

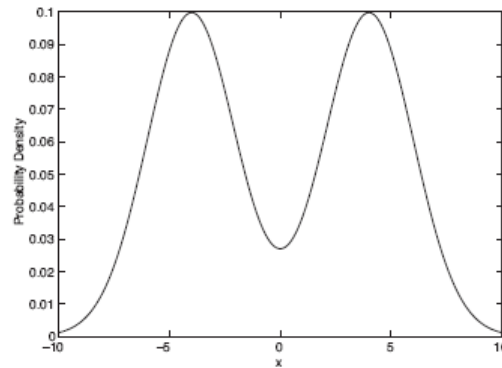


(b) 20,000 points generated from the mixture model.

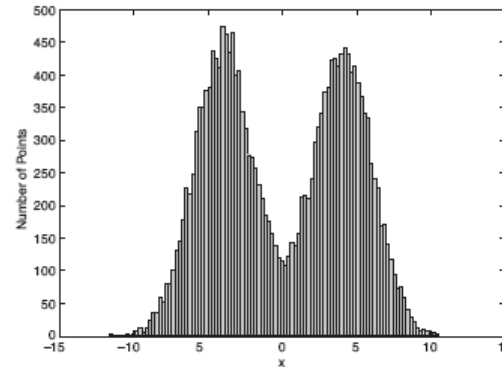
Figure 9.2. Mixture model consisting of two normal distributions with means of -4 and 4, respectively. Both distributions have a standard deviation of 2.

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

- Our model has the following parameters

$$\Theta = (\pi_G, \pi_C, \mu_G, \mu_C, \sigma_G, \sigma_C)$$

Mixture probabilities

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 $X = \{x_1, \dots, x_n\}$

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

- 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)

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

EM (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 $P(G|x_i)$ and $P(C|x_i)$
 - **M-Step:** Calculate the parameter values Θ that (in expectation) **maximize** the data likelihood

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

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

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

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

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

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

Fraction of
population in G,C

MLE Estimates
if π 's were fixed

Bisecting K-Means

Bisecting K-means

- Variant of K-Means that can produce a hierarchical clustering

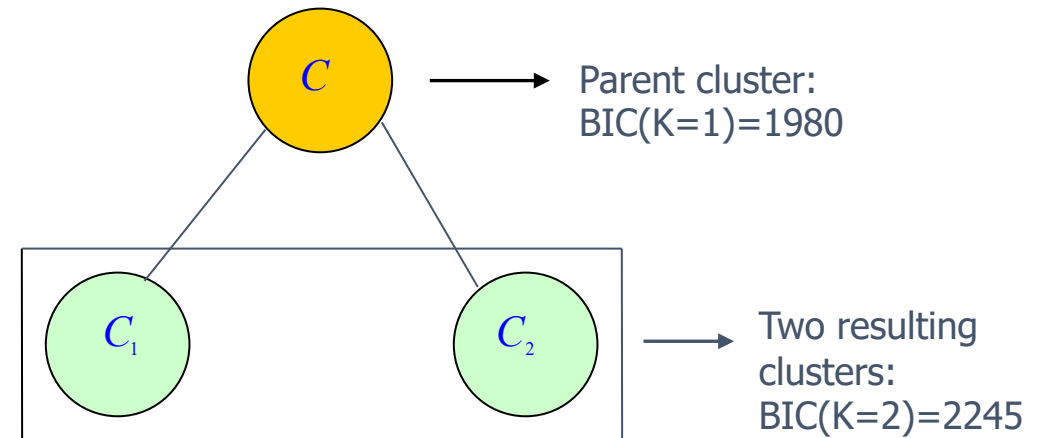
```
1: Initialize the list of clusters to contain the cluster containing all points.
2: repeat
3:   Select a cluster from the list of clusters
4:   for  $i = 1$  to number_of_iterations do
5:     Bisect the selected cluster using basic 2-Means
6:   end for
7:   Add the two clusters from the bisection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

Bisecting K-means Limitations

- The algorithm is exhaustive terminating at singleton clusters (unless K is known or 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 use any criterion for stopping bisections before singleton clusters are reached.

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.



X-Means

X-Means

Search for the appropriate value of k in a given range $[r_1, r_{\max}]$:

1. Improve Params
2. Improve Structure
3. If $k > r_{\max}$ stop and return the best-scoring model

Improve Params

- Run K-Means with with the current k

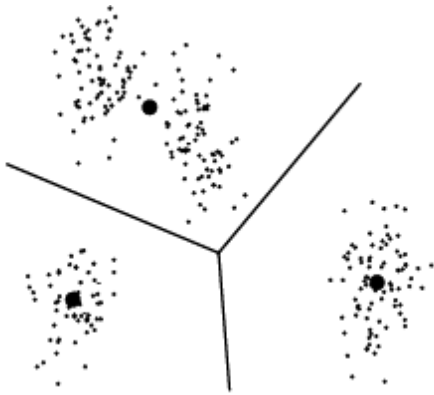
Improve Structure

- Recursively split each cluster in two and use local BIC to decide to keep the split

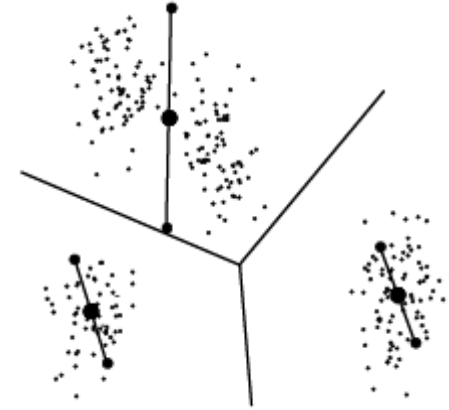
Finally, use global BIC score to decide which K to output at the end

X-Means

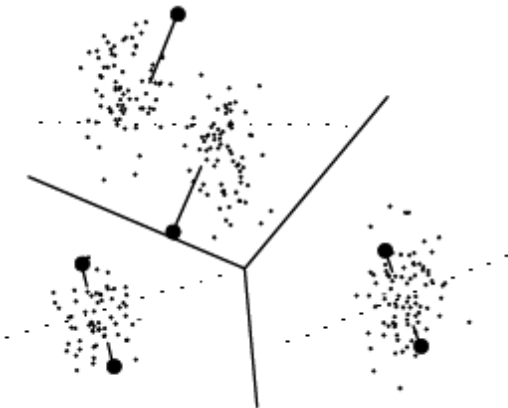
1. K-means with $k=3$



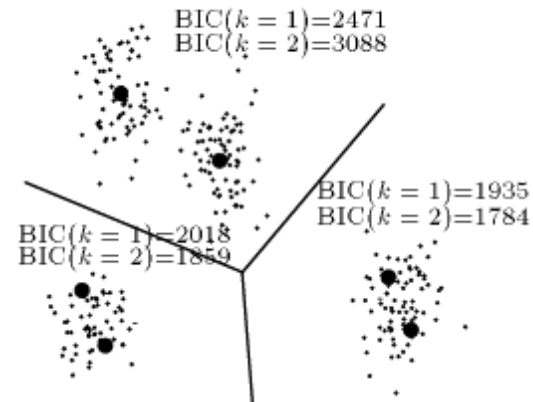
2. Split each centroid in 2 children moved a distance proportional to the region size in opposite direction (random)



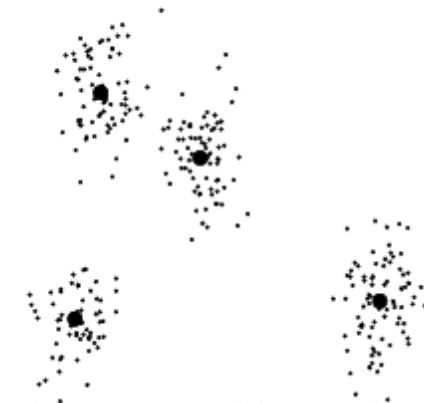
3. Run 2-means in each region locally



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

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

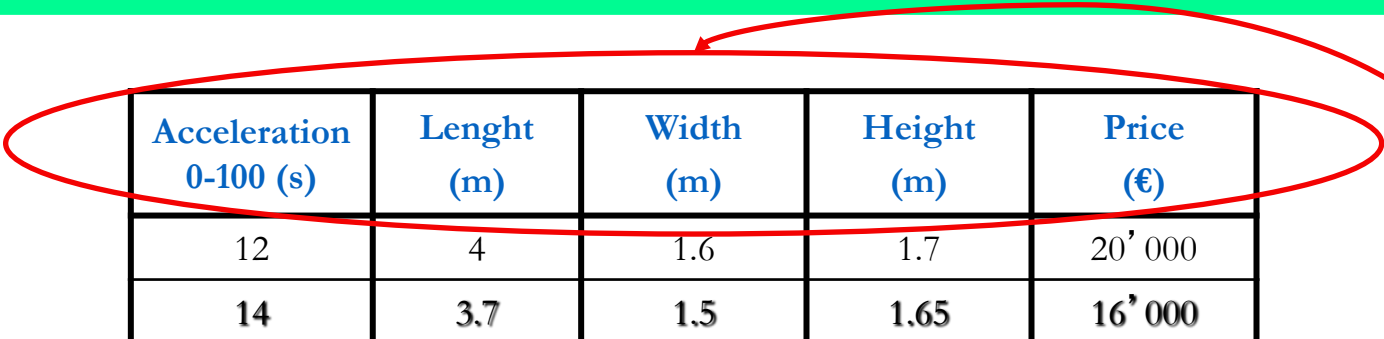
- It estimates how closely to the centroid are the points of the cluster.

Transactional Clustering

Clustering

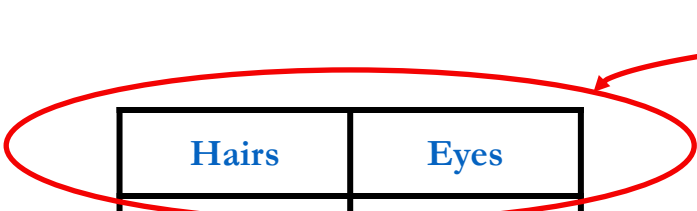
- **Clustering:** Grouping of objects into different sets, or more precisely, the partitioning of a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait - often proximity according to some defined distance measure
- Common distance functions:
 - Euclidean distance, Manhattan distance, ...
- This kind of distance functions are suitable for **numerical data**

Not Only Numerical Data



Acceleration 0-100 (s)	Lenght (m)	Width (m)	Height (m)	Price (€)
12	4	1.6	1.7	20' 000
14	3.7	1.5	1.65	16' 000
15	3.5	1.5	1.6	12' 000
9.4	4.2	1.8	1.7	24' 000

Numerical Data



Hairs	Eyes
brown	black
blond	blue
black	green
red	brown

Categorical Data

Boolean and Categorical Attributes

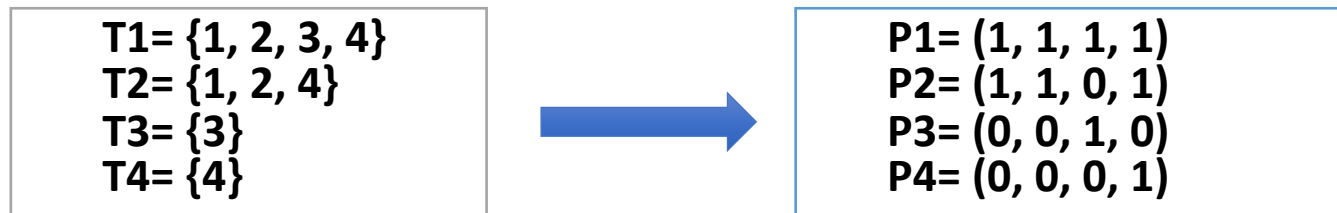
- A **boolean** attribute corresponding to a single item in a transaction, if that item appears, the boolean attribute is set to '1' or '0' otherwise.
- A **categorical** attribute may have several values, each value can be treated as an item and represented by a boolean attribute.

Market Basket Data

- A transaction represents one customer, and each transaction contains set of items purchased by the customer.
- Clustering customers reveals customers with similar buying patterns putting them into the same cluster.
- It is useful for
 - Characterizing different customer groups
 - Targeted Marketing
 - Predict buying patterns of new customers based on profile
- A market basket database: A scenario where attributes of data points are non-numeric, transaction viewed as records with boolean attributes corresponding to a single item (TRUE if transaction contain item, FALSE otherwise).
- **Boolean** attributes are special case of **Categorical** attributes.

Shortcomings of Traditional Clustering

- For categorical data we:
 - Define new criterion for *neighbors* and/or *similarity*
 - Define the ordering criterion
- Consider the following 4 market basket transactions



- using Euclidean distance to measure the closeness between all pairs of points, we find that $d(P1, P2)$ is the smallest distance: **it is equal to 1**

Shortcomings of Traditional Clustering

- If we use a hierarchical algorithm then we merge P1 and P2 and get a new cluster (P12) with (1, 1, 0.5, 1) as a centroid
- Then, using Euclidean distance again, we find:
 - $d(p_{12}, p_3) = \sqrt{3.25}$
 - $d(p_{12}, p_4) = \sqrt{2.25}$
 - $d(p_3, p_4) = \sqrt{2}$
- So, **we should merge P3 and P4** since the distance between them is the shortest.
- **However, T3 and T4 don't have even a single common item.**
- So, using distance metrics as similarity measure for **categorical** data is not appropriate.

P1= (1, 1, 1, 1)
P2= (1, 1, 0, 1)
P3= (0, 0, 1, 0)
P4= (0, 0, 0, 1)

Clustering Algorithms for Categorical/Transactional Data

- K-Modes
- ROCK
- CLOPE
- TX-Means

K-Modes

$$\text{Minimise } P(W, \mathcal{Q}) = \sum_{l=1}^k \sum_{i=1}^n w_{i,l} d(X_i, Q_l)$$

$$\text{subject to } \sum_{l=1}^k w_{i,l} = 1, \quad 1 \leq i \leq n$$
$$w_{i,l} \in \{0, 1\}, \quad 1 \leq i \leq n, 1 \leq l \leq k$$

- $X = \{X_1, \dots, X_n\}$ is the dataset of objects.
- $X_i = [x_1, \dots, x_m]$ is an object i.e., a vector of m categorical attributes
- W is a matrix $n \times k$, with $w_{i,l}$ equal to 1 if X_i belongs to Cluster l , 0 otherwise.
- $Q = \{Q_1, \dots, Q_k\}$ is the set of representative objects (mode) for the k clusters.
- $d(X_i, Q_l)$ is a distance function for objects in the data

K-Modes: Distance

- K-Means as distance uses Euclidean distance

$$d(X, Y) = \sum_{i=1}^m (x_i - y_i)^2$$

- K-Modes as distance uses the number of mismatches between the attributes of two objects.

$$d_1(X, Y) = \sum_{j=1}^m \delta(x_j, y_j)$$

$$\delta(x_j, y_j) = \begin{cases} 0 & (x_j = y_j) \\ 1 & (x_j \neq y_j) \end{cases}$$

K-Modes: Mode

- K-Modes uses the mode as representative object of a cluster
- Given the set of objects in the cluster C_l the mode is get computing the max frequency for each attribute

$$f_r(A_j = c_{l,j} | X_l) = \frac{n_{c_{l,k}}}{n}$$

K-Modes: Algorithm

1. Select the initial objects as modes
2. Scan of the data to assign each object to the closer cluster identified by the mode
3. Re-compute the mode of each cluster
4. Repeat the steps 2 and 3 until no object changes the assigned cluster

ROCK: RObust Clustering using link

- ROCK is a **hierarchical** algorithm for clustering transactional data (market basket databases)
- ROCK uses **links to cluster** instead of the classical distance notion
- ROCK uses the notion of **neighborhood** between pair of objects to identify **the number of links** between two objects

ROCK: The Neighbors Concept

- It captures a notion of **similarity**
 - A and B are neighbors if $\text{sim}(A, B) \geq \theta$
- ROCK uses the **Jaccard coefficient**
 - $\text{sim}(A, B) = |A \cap B| / |A \cup B|$

$$A = \{1, 3, 4, 7\}$$

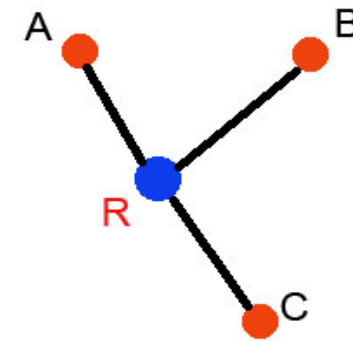
$$B = \{1, 2, 4, 7, 8\}$$



$$\text{sim}(A, B) = \frac{3}{6} = \frac{1}{2} = 0.5$$

ROCK: Links

- A **link** defines the number of common neighbors between two objects:
- $\text{link}(A, B) = |\text{neighbor}(A) \cap \text{neighbor}(B)|$
- Higher values of $\text{link}(A, B)$ means higher probability that A and B belong to the same cluster
- **Similarity** is **local** while **link** is capturing **global** information
- A point is considered a neighbor of itself
- There is a link from each neighbor of the “root” point back to itself through the root
- Therefore, if a point has n neighbors, then n^2 links are due to it.



A->R->B
A->R->C
B->R->A
B->R->C
C->R->B
C->R->A
A->R->A
B->R->B
C->R->C

ROCK: Example

- Data consisting of 6 Attributes: {Book, Water, Sun, Sand, Swimming, Reading}
 - {Book}
 - {Water, Sun, Sand, Swimming}
 - {Water, Sun, Sand, Reading}
 - {Reading, Sand}

- Resulting Jaccard Coefficient Matrix

	A	B	C	D
A	1	0	0	0
B	0	1	0.6	0.2
C	0	0.6	1	0.5
D	0	0.2	0.5	1

- Set Threshold = 0.2. Neighbors:

- $N(A) = \{A\}$; $N(B) = \{B, C, D\}$
- $N(C) = \{B, C, D\}$, $N(D) = \{B, C, D\}$

- Number of Links Table

- $\text{Link}(B, C) = |\{B, C, D\}| = 3$

	A	B	C	D
A	1	0	0	0
B	0	3	3	3
C	0	3	3	3
D	0	3	3	3

- Resulting Clusters after applying ROCK: {A}, {B,C,D}

ROCK – Criterion Function

Maximize

$$E_l = \sum_{i=1}^k n_i * \sum_{p_q, p_r \in C_i} \frac{\text{link}(p_q, p_r)}{n_i^{1+2f(\theta)}}$$

$$f(\theta) = \frac{1-\theta}{1+\theta}$$

Dividing by the number of expected links between pairs of objects in the cluster C_i we avoid that objects with a low number of links are assigned all to the same cluster

Where C_i denotes cluster i
 n_i is the number of points in C_i
 k is the number of clusters
 θ is the similarity threshold

This goodness measure helps to identify the best pair of clusters to be merged during each step of ROCK.

$$g(C_i, C_j) = \frac{\text{link}[C_i, C_j]}{(n_i + n_j)^{1+2f(\theta)} - n_i^{1+2f(\theta)} - n_j^{1+2f(\theta)}}$$

Number of expected cross-links between two clusters

ROCK: Clustering Algorithm

Input:

- A set S of data points
- Number of k clusters to be found
- The similarity threshold

Output:

- Groups of clustered data

The ROCK algorithm is divided into three major parts:

1. Draw a random sample from the data set
2. Perform a hierarchical agglomerative clustering algorithm
3. Label data

ROCK: Clustering Algorithm

Draw a random sample from the data set:

- Sampling is used to ensure scalability to very large data sets
- The initial sample is used to form clusters, then the remaining data on dataset is assigned to these clusters

ROCK: Clustering Algorithm

Perform a hierarchical agglomerative clustering algorithm:

- ROCK performs the following steps which are common to all hierarchical agglomerative clustering algorithms, but with different definition to the similarity measures:
 1. Places each single data point into a separate cluster
 2. Compute the similarity measure for all pairs of clusters
 3. Merge the two clusters with the highest similarity (goodness measure)
 4. Verify a stop condition. If it is not met then go to step 2.

ROCK: Clustering Algorithm

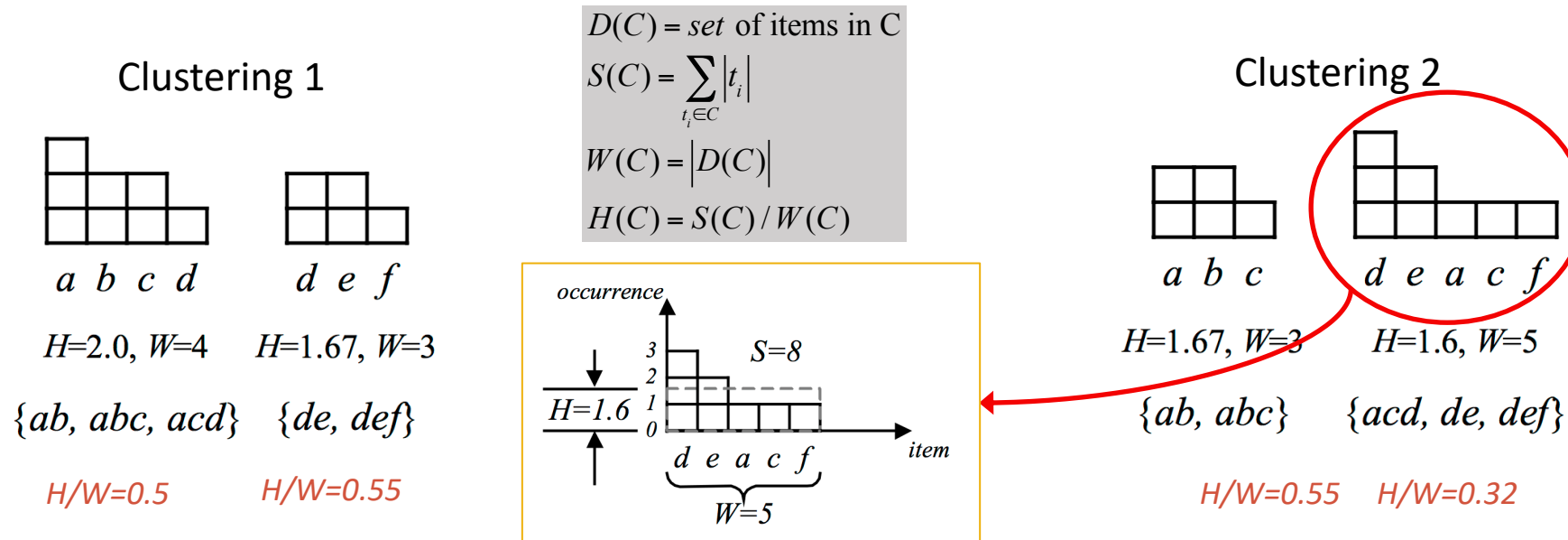
Label data

- Finally, the remaining data points are assigned to the clusters.
- This is done by selecting a random sample L_i from each cluster C_i , then we assign each point p to the cluster for which it has the strongest linkage with L_i .

CLOPE (Clustering with LOPE)

- Transactional clustering efficient for high dimensional data
- Uses a **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**.

Example: 5 transactions {a,b} {a,b,c} {a,c,d} {d,e} {d,e,f}




Higher H/W means higher item overlapping

CLOPE: Criterion Function

- For evaluating the goodness of a clustering the **gradient of a cluster** is
- $G(C)=H(C)/W(C)=S(C)/W(C)^2$

Repulsion.

When r is large,
transactions within the
same cluster must
share a large portion of
common items.


$$Profit_r(C) = \frac{\sum_{i=1}^k \frac{S(C_i)}{W(C_i)^r} \times |C_i|}{\sum_{i=1}^k |C_i|}$$

CLOPE: Algorithm

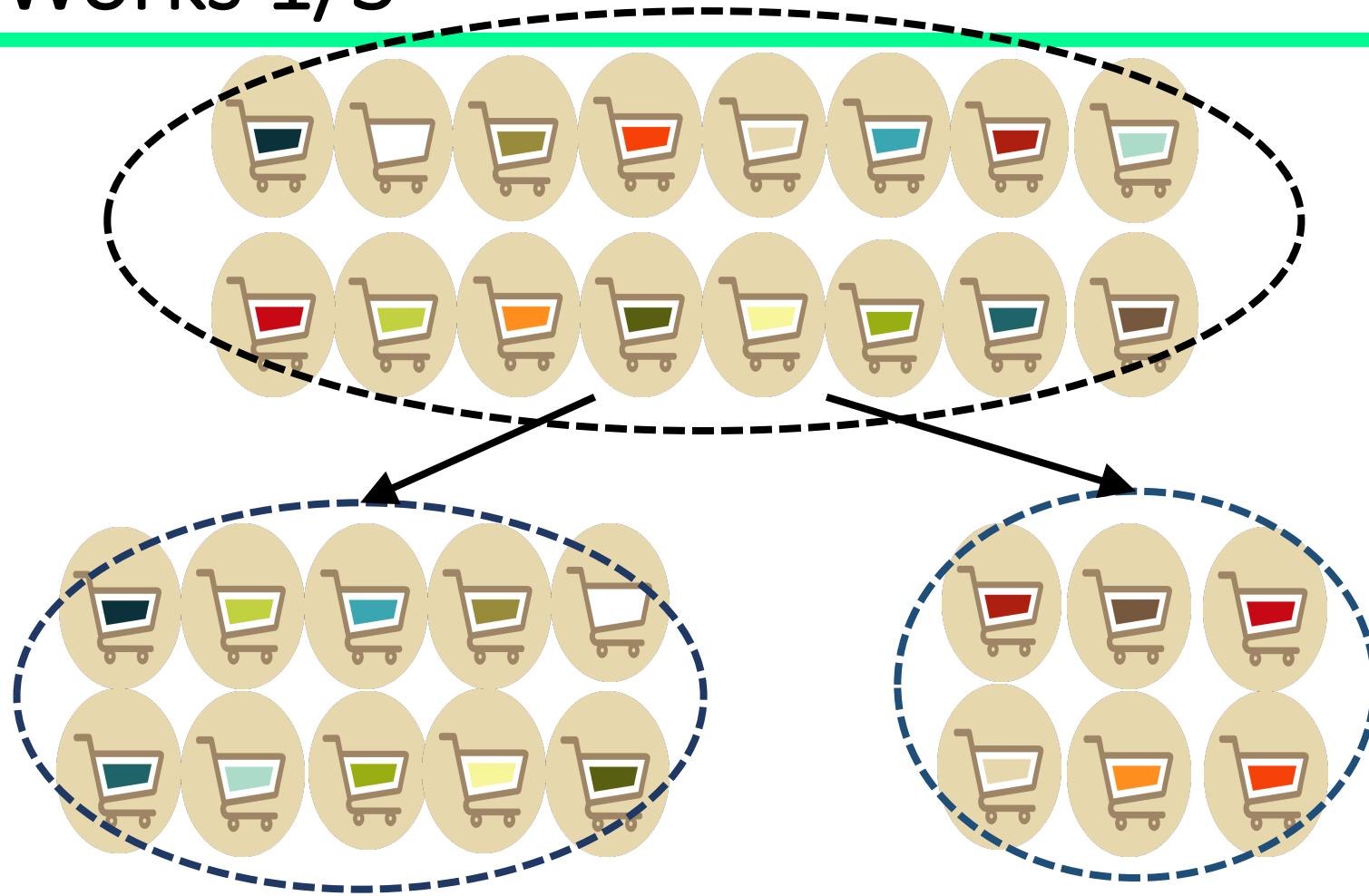
```
/* Phrase 1 - Initialization */
1: while not end of the database file
2:   read the next transaction  $\langle t, \text{unknown} \rangle$ ;
3:   put  $t$  in an existing cluster or a new cluster  $C_i$ 
   that maximize profit;
4:   write  $\langle t, i \rangle$  back to database;

/* Phrase 2 - Iteration */
5: repeat
6:   rewind the database file;
7:    $moved = \text{false}$ ;
8:   while not end of the database file
9:     read  $\langle t, i \rangle$ ;
10:    move  $t$  to an existing cluster or new cluster  $C_j$ 
    that maximize profit;
11:    if  $C_i \neq C_j$  then
12:      write  $\langle t, j \rangle$ ;
13:       $moved = \text{true}$ ;
14: until not  $moved$ ;
```

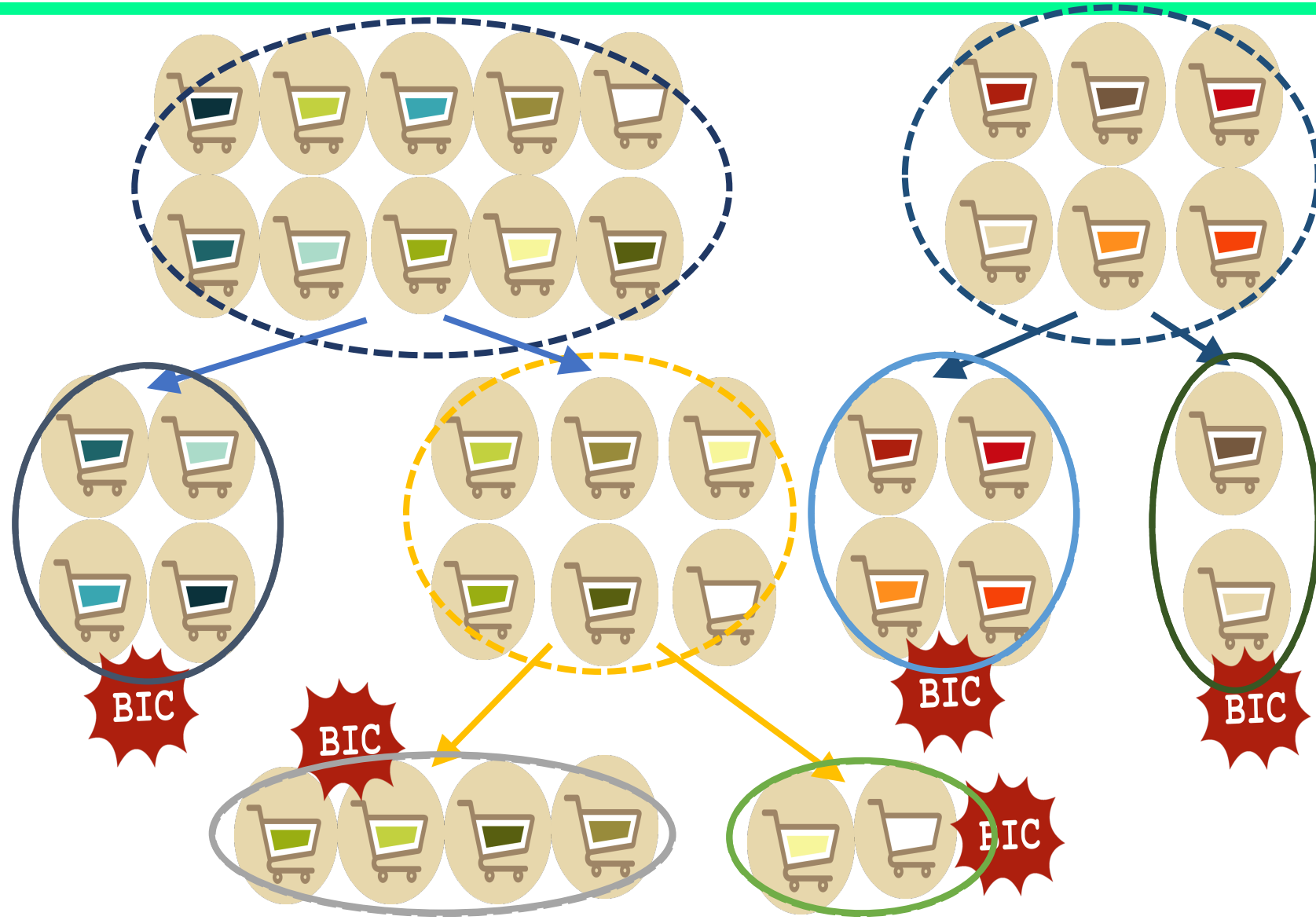
TX-MEANS

- A parameter-free clustering algorithm able to efficiently partitioning transactional data automatically
- Suitable for the case where clustering must be applied on a massive number of different datasets
 - E.g.: when a large set of users need to be analyzed individually and each of them has generated a long history of transactions
- TX-Means automatically estimates **the number of clusters**
- TX-Means provides the **representative transaction** of each cluster, which summarizes the pattern captured by that cluster.

How It Works 1/3

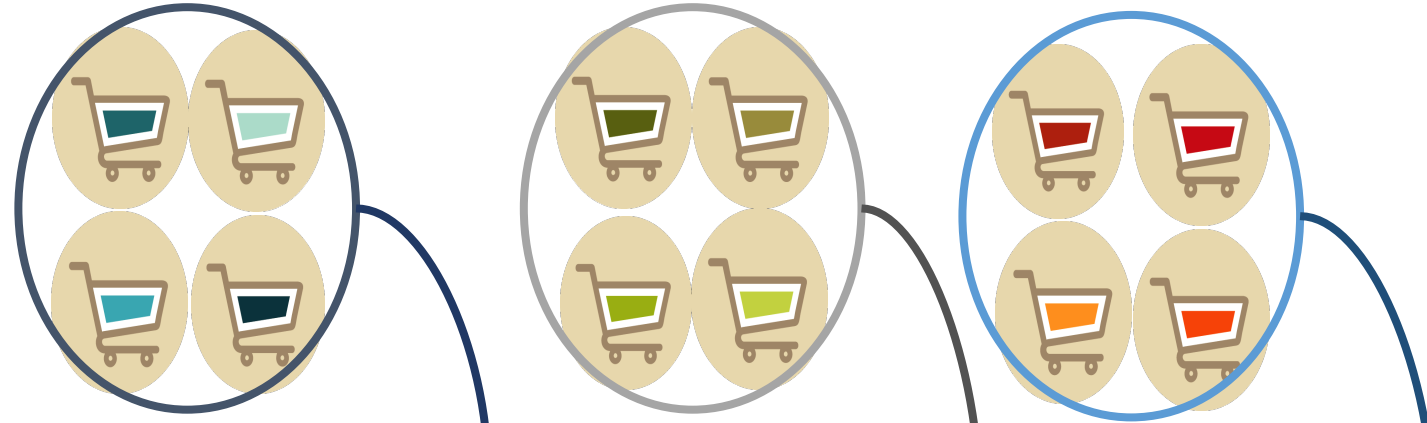


How It Works 2/3



How It Works 3/3

- Clusters



- Representative Baskets



TX-Means Algorithm

TXMEANS(**B**: baskets):

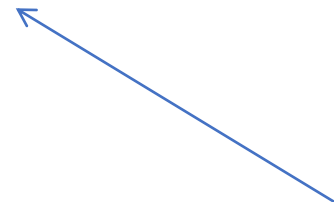
- $r \leftarrow \text{GETREPR}(B);$ ← representative basket
- $Q.\text{push}(B, r);$
- **While** there is a cluster B, r to split in Q :
 - Remove common items from B ;
 - $B_1, B_2, r_1, r_2 \leftarrow \text{BISECTBASKET}(B);$ ← bisecting schema
 - **If** $\text{BIC}(B_1, B_2, r_1, r_2) > \text{BIC}(B, r)$ **Then**: ← stopping criterion
 - add B_1, B_2, r_1, r_2 to the clusters to split Q ;
 - **Else**
 - add B, r to the clustering result C ;
- **Return** C ;

Bisecting Schema

BISECTBASKET(B: baskets):

- SSE `<-- inf;`
- `r1,r2 <--` select random initial baskets in B as representative;
- **While True:**
 - `C1,C2 <--` assign baskets in B with respect to `r1,r2`;
 - `r1_new <-- GETREPR(C1); r2_new <-- GETREPR(C2);`
 - `SSE_new <-- SSE(C1,C2,r1_new,r2_new);`
 - **If** `SSE_new >= SSE` **Then:**
 - **Return** `C1,C2,r1,r2`;
 - `r1,r2 <-- r1_new,r2_new;`

overlap-based
distance function:
Jaccard coefficient

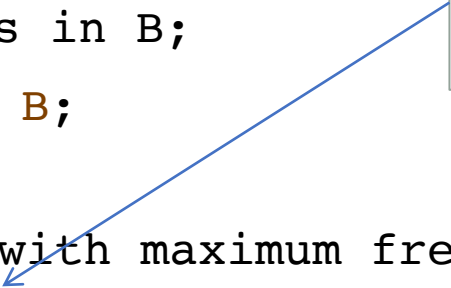


Get Representative Baskets

GETREPR(**B**: baskets):

- `I` \leftarrow not common items in `B`;
- `r` \leftarrow common items in `B`;
- **While** `I` is not empty:
 - Add to `r` the items with maximum frequency in `I`;
 - Calculate the **distance** between `r` and the baskets in `B`;
 - **If** the **distance** no longer decreases **Then**:
 - **Return** `r`;
 - **Else**
 - remove from `I` the items with maximum frequency;
- **Return** `r`;

overlap-based distance
function (Jaccard
coefficient)



Dealing with Big Datasets

- Clustering of a big individual transactional dataset **B**.
- TX-Means is scalable thanks to the following sampling strategy.
- Sampling strategy:
 - Random selection of a subset **S** of the baskets in **B**;
 - Run of TX-Means on the subset **S** and obtain clusters **C** and representative baskets **R**;
 - Assign the remaining baskets **B/S** to the clusters **C** using a nearest neighbor approach with respect to the representative baskets **R**.

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.
- Guha, S., et al. ROCK: A robust clustering algorithm for categorical attributes. 2000.
- Yang, Y., et al. CLOPE: a fast and effective clustering algorithm for transactional data. 2002.
- Guidotti, R., et al. Clustering individual transactional data for masses of users. 2017.

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

Dan Pelleg
Andrew Moore
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 exact 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 major shortcomings. One, it is slow and scales poorly with

Clustering Individual Transactional Data for Masses of Users
Riccardo Guidotti
ISTI-CNR & University of Pisa, Italy
riccardo.guidotti@isti.cnr.it

Fosca Giannotti
ISTI-CNR, Pisa, Italy
fosca.giannotti@isti.cnr.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 *trmeans*, a parameter-free clustering algorithm able to efficiently partitioning transactional data in a completely automatic way. *Trmeans* 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 show the practical effectiveness of *trmeans* for the mass clustering of different personal datasets, and suggests that *trmeans* outperforms existing methods in terms of quality and efficiency. Finally, we present a *personal cart assistant* application based on *trmeans*.

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 500 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, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permission from permissions@acm.org.
EUD: 10.1145/3099333.3099344
© 2017 ACM. 978-1-4503-467-7/17/06...\$15.00
DOI: 10.1145/3099333.3099344

lutions for these problems. Speed is greatly improved by embedding the dataset in a multiresolution k -tree and storing sufficient statistics at its nodes. A careful analysis of the centroid locations allows for geometric "pruning" 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 optimization, *MerKling*, 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 th

Pergamon
ROCK: A ROBUST CLUSTERING ALGORITHM FOR CATEGORICAL ATTRIBUTES¹
SUDHITO GUHA¹, RAJEEV RASTOPI², and KYUSOOK SHIM²
¹Stanford University, Stanford, CA 94305, USA
²Shell Laboratories, Murray Hill, NJ 07974, USA
³Korea Advanced Institute of Science and Technology and Advanced Information Technology Research Center, Tjeon 305-701, Korea
21 May 2000

Clustering Individual Transactional Data for Masses of Users
Riccardo Guidotti
ISTI-CNR & University of Pisa, Italy
riccardo.guidotti@isti.cnr.it

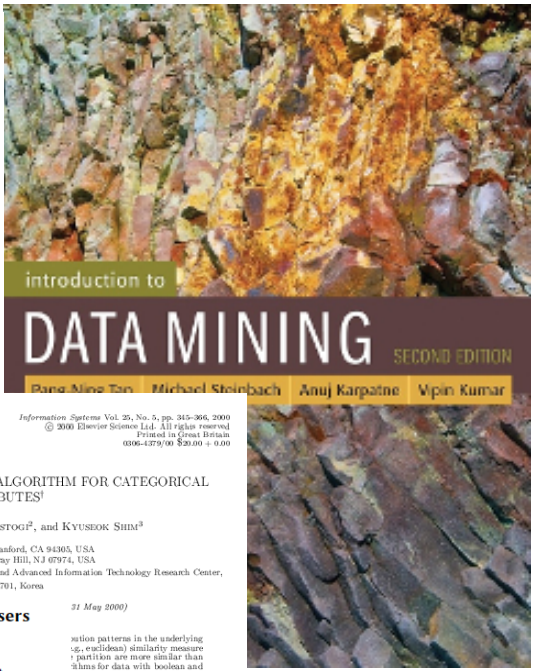
Dino Pedreschi
University of Pisa, Italy
dino.pedreschi@di.unipi.it

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 *trmeans*, a parameter-free clustering algorithm able to efficiently partitioning transactional data in a completely automatic way. *Trmeans* 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 show the practical effectiveness of *trmeans* for the mass clustering of different personal datasets, and suggests that *trmeans* outperforms existing methods in terms of quality and efficiency. Finally, we present a *personal cart assistant* application based on *trmeans*.

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 500 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, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permission from permissions@acm.org.
EUD: 10.1145/3099333.3099344
© 2017 ACM. 978-1-4503-467-7/17/06...\$15.00
DOI: 10.1145/3099333.3099344



CLOPE: A Fast and Effective Clustering Algorithm for Transactional Data

Yiling Yang Xudong Guan Jinyuan Yu
Dept. of Computer Science & Engineering, Shanghai Jiao Tong University
Shanghai, 200030, P.R.China
+86-21-52981636
{yang-yl, guan-xd, you-yy}@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, 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 datasets 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 [13] are effective for low dimensional numerical 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 datasets.

The Largeltem [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 Largeltem 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 Largeltem, with clustering quality quite close to that of the ROCK algorithm [7].

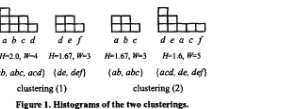


Figure 1. Histograms of the two clusters. We judge the qualities of these two clusterings geometrically, by analyzing the heights and widths of the clusters. Leaving out the two identical histograms for cluster {acd, def} and cluster {ab, abc}, the other two histograms are of different quality. The histogram for cluster {ab, abc, acd} has only 4 distinct items for 8 blocks ($H=2.0$, $W=0.5$), but the one for cluster {acd, def} has 5, for

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, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permission from permissions@acm.org.
SIGMOD '02, July 23-26, 2002, Edmonton, Alberta, Canada.
Copyright 2002 ACM 1-58113-567-X/02/0007...\$5.00.