DATA MINING 2
Transactional Clustering

Riccardo Guidotti

a.a. 2023/2024

UNIVERSITÀ DI PISA

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

Categorical Data

Hairs	Eyes
brown	black
blond	blue
black	green
red	brown

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.

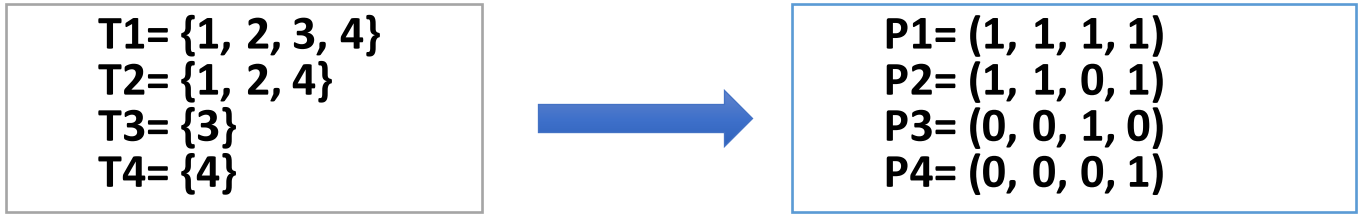
Market Basket Data

- 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: 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



T1= {1, 2, 3, 4}
T2= {1, 2, 4}
T3= {3}
T4= {4}

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

- 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(p12,p3)= √3.25
  - d(p12,p4)= √2.25
  - d(p3,p4)= √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)

Algorithms for Categorical/Transactional Data

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

K-Modes

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

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

- $X = \{ X_1 ,\ldots, X_n \}$ is the dataset of objects.
- $X_i = [x_1 ,\ldots, x_m]$ 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.
- $Q = \{ Q_1 ,\ldots, Q_k \}$ 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

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

K-Modes: Algorithm

1. Randomly 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: 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: The Neighbors Concept

- It captures a notion of **similarity**
  - A and B **are neighbors** if **sim(A, B) ≥ θ**

- ROCK uses the **Jaccard coefficient**
  - sim(A, B)= |A ∩ B| / | A U B |

A = { 1 , 3 , 4 , 7 }

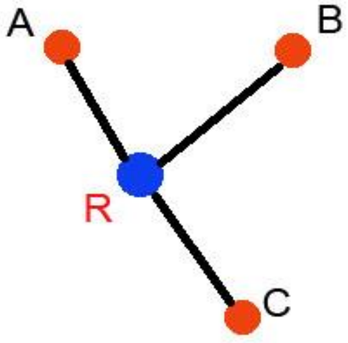
B = { 1 , 2 , 4 , 7 , 8 }

→

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

ROCK: Links

- A **link** defines the number of common neighbors between two objects:
- **link(A, B) = |neighbor(A)** ∩ **neighbor(B) |**
- Higher values of *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
  - 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 $$E_l = \sum_{i=1}^{k} n_i * \sum_{p_q, p_r \in C_i} \frac{link(p_q, p_r)}{n_i^{1+2f(\theta)}}$$

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

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.

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

Number of expected cross-links between two clusters

ROCK: Clustering Algorithm

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.

ROCK Summary

Input: dataset, number of clusters.

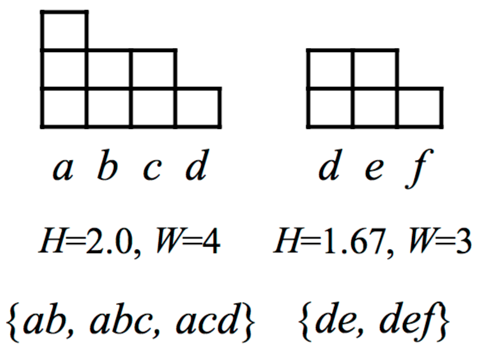
1. Draw a random sample from the data set
2. Places each data point into a separate cluster
3. Compute the similarity measure for all pairs of clusters
4. Merge the two clusters with the highest similarity
5. Verify a stop condition. If it is not met, then go to step 2.
6. Assign not used points to clusters using linkage similarity with respect to selected samples from each cluster

CLOPE: Clustering with sLOPE

- 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}

Clustering 1



H/W=0.5

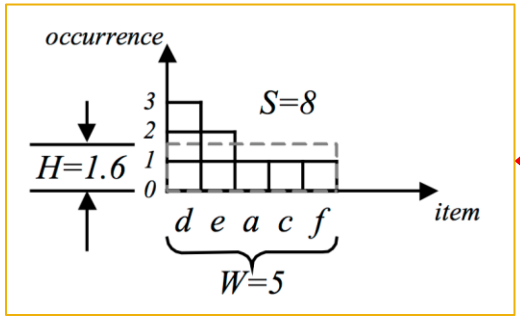
H/W=0.55

$D(C) = set$ of items in C

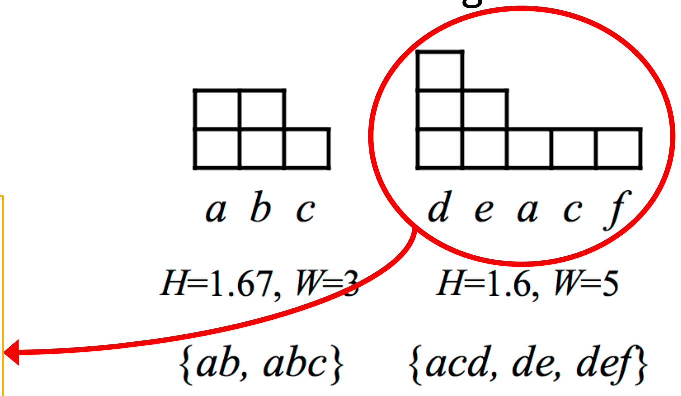
$S(C) = \sum_{t_i \in C} |t_i|$

$W(C) = |D(C)|$

$H(C) = S(C) / W(C)$



Clustering 2



H/W=0.55

H/W=0.32

Higher H/W means higher item overlapping

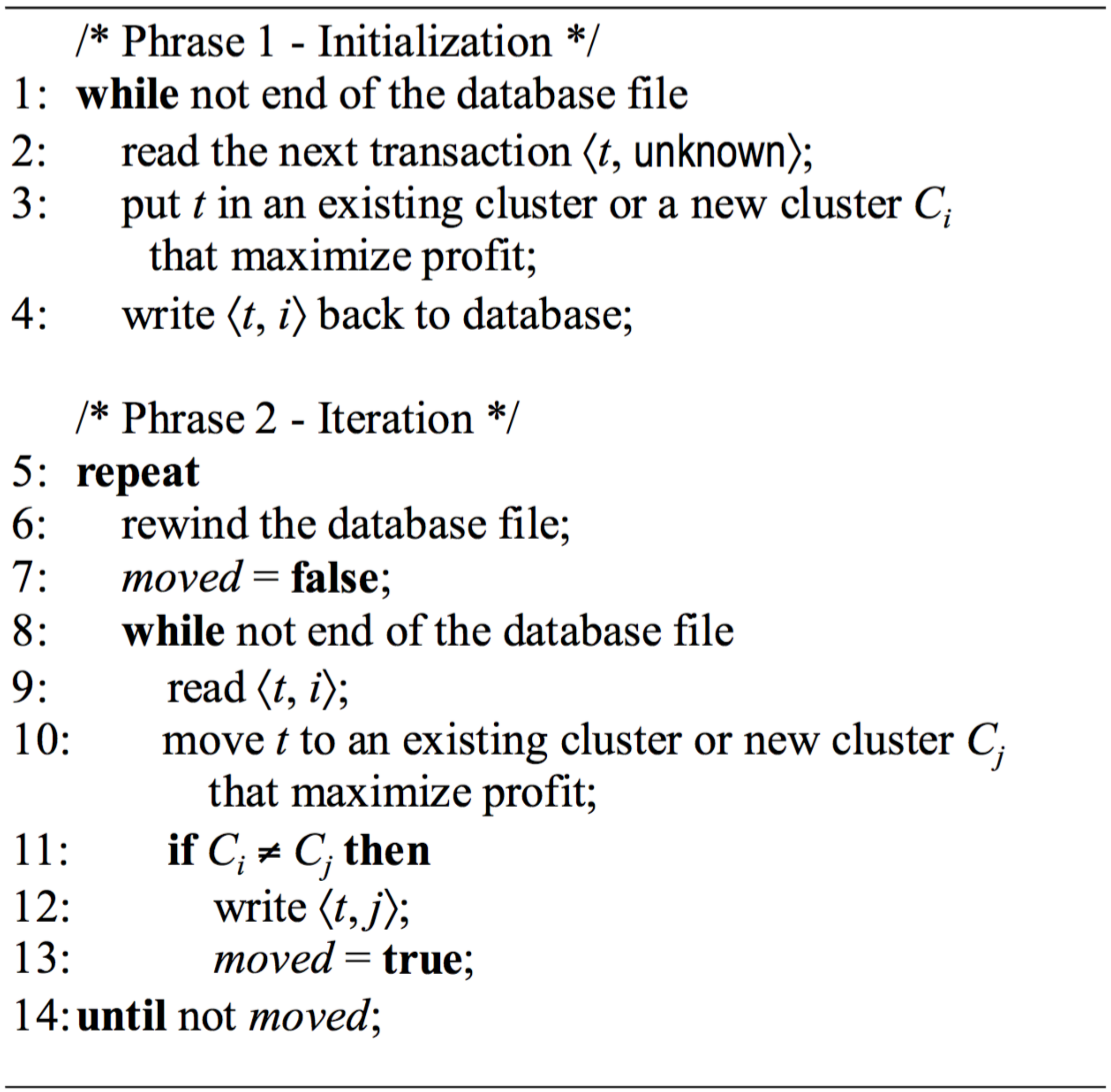
CLOPE: Criterion Function

- 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.

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

CLOPE: Algorithm



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

/* Phrase 2 - Iteration */
repeat
    rewind the database file;
    $moved$ = false;
    while not end of the database file
        read $\langle t, i \rangle$;
        move $t$ to an existing cluster or new cluster $C_j$
          that maximize profit;
        if $C_i \neq C_j$ then
            write $\langle t, j \rangle$;
            $moved$ = true;
until not $moved$;
```

CLOPE Summary

Input: dataset, repulsion, maximum number of clusters

- Phase 1

1. For each transaction, add it to a new cluster or to an existing one such that the profit is maximized

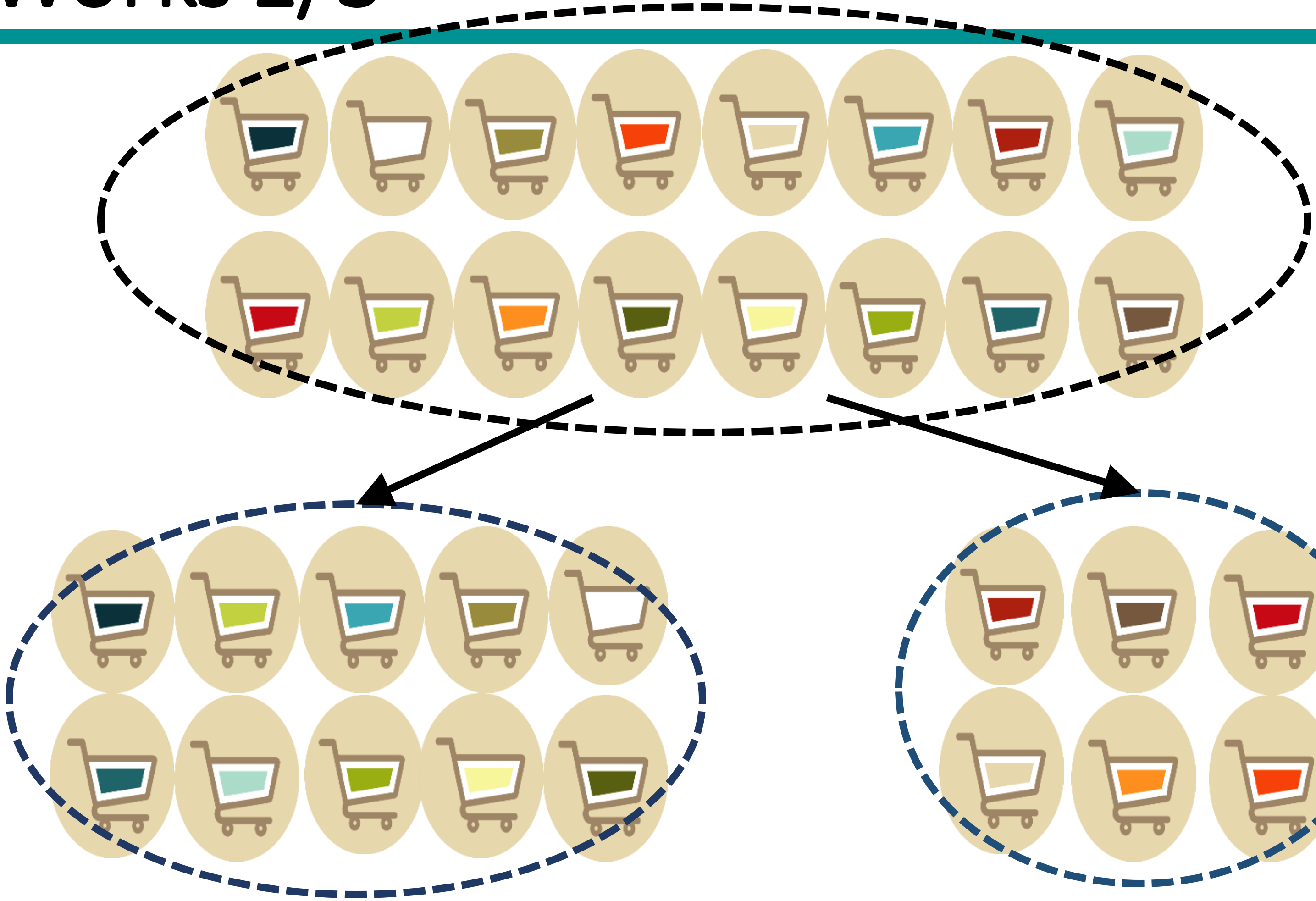
- Phase 2

1. For each transaction, try to move it to another cluster and do it if this maximizes the profit
2. Repeat 1. until all the transactions remain in the same cluster

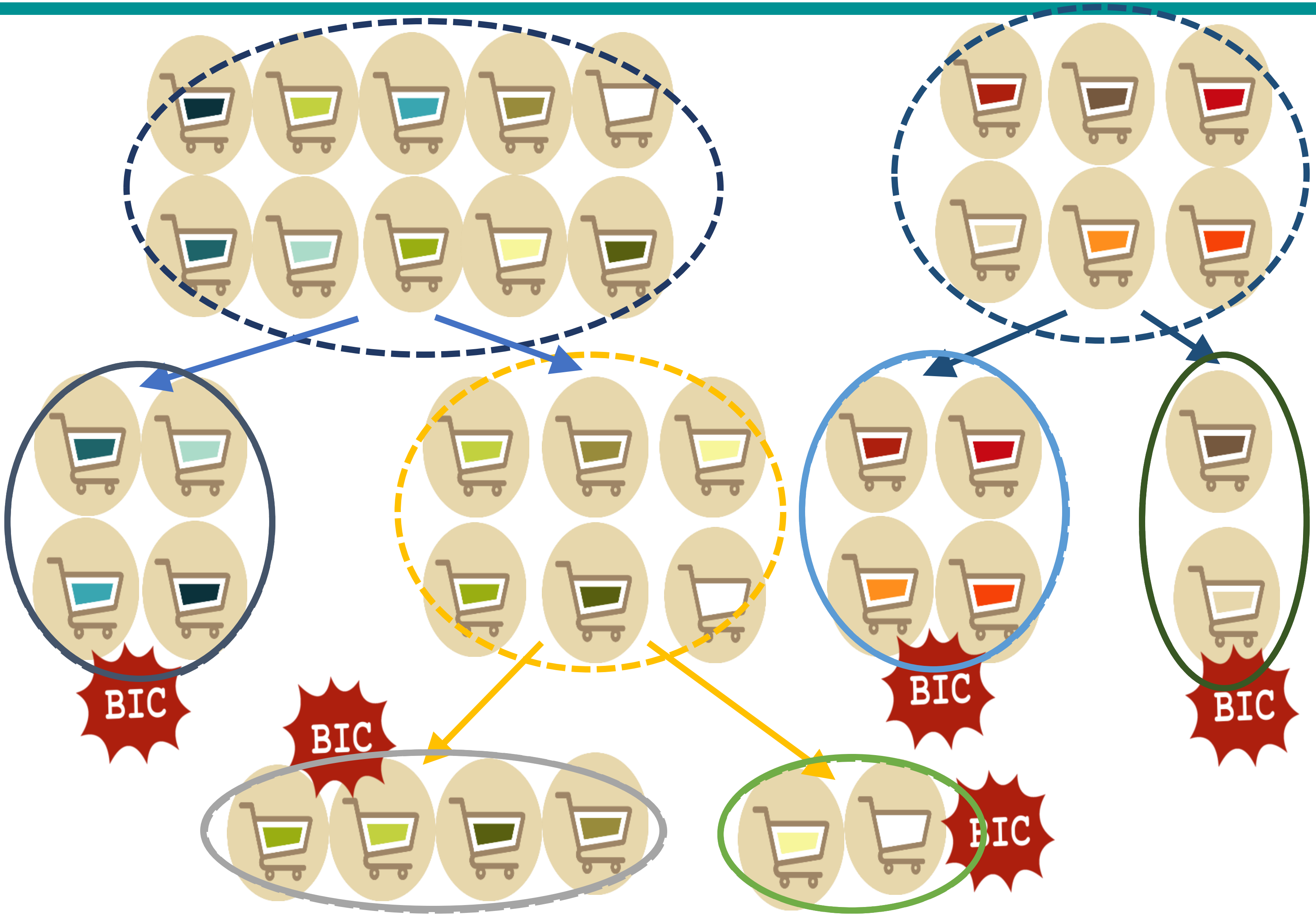
TX-MEANS

- 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



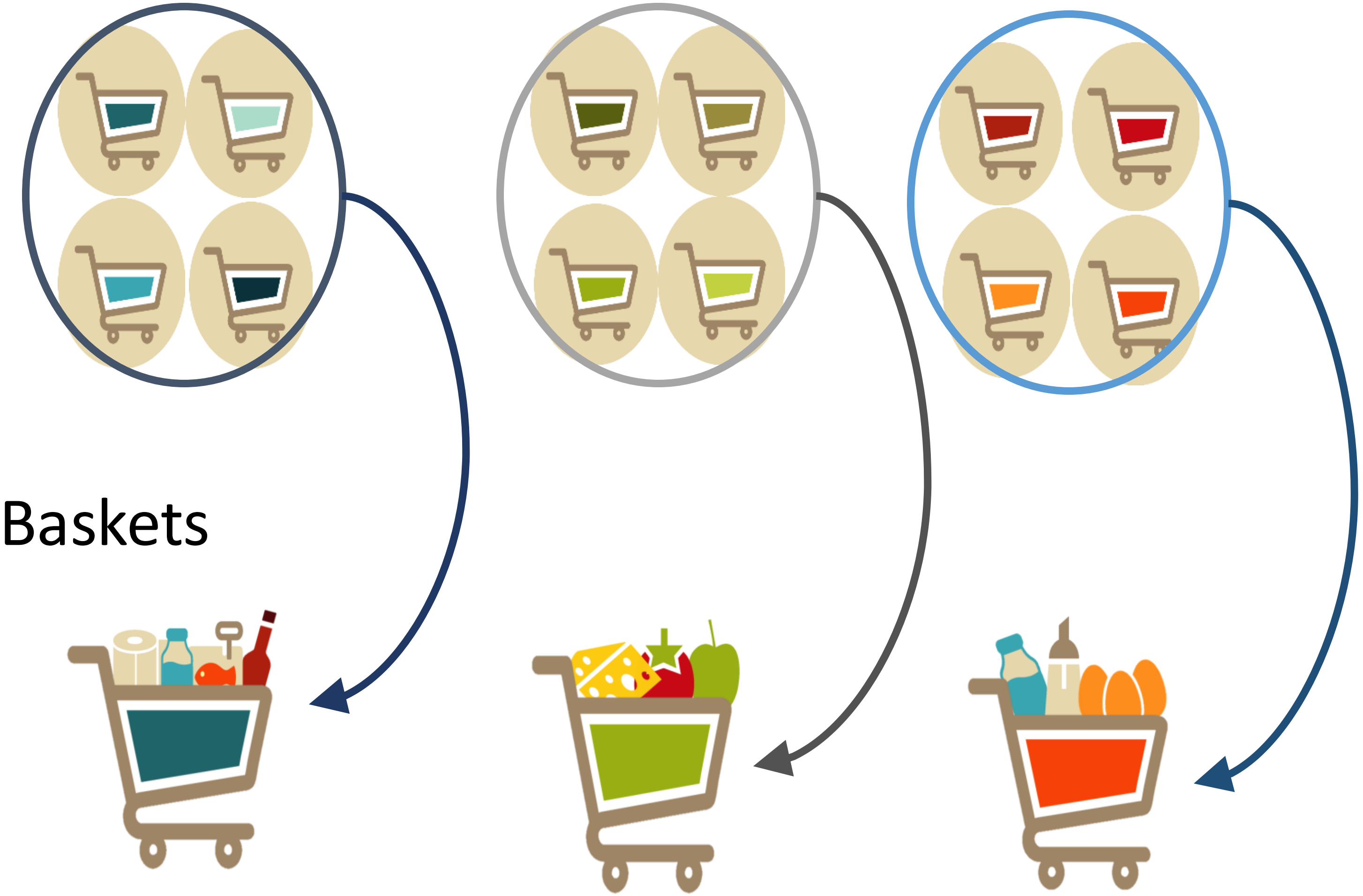
How It Works 2/3



How It Works 3/3

- Clusters

- Representative Baskets



TX-Means Algorithm

```
TXMEANS(B: baskets):
• r <-- GETREPR(B);
• Q.push(B,r);
• While there is a cluster B,r to split in Q:
    • Remove common items from B;
    • B1, B2, r1, r2 <-- BISECTBASKET(B);
    • If BIC(B1,B2,r1,r2) > BIC(B,r) Then:
        • add B1,B2,r1,r2 to the clusters to split Q;
    • Else
        • add B,r to the clustering result C;
• Return C;
```

representative basket

bisecting schema

stopping criterion

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 <-- not common items in B;
• r <-- 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

- 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 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.

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

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

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)

bution patterns in the underlying
.g., euclidean) similarity measure
partition are more similar than
rithms for data with boolean and

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

from other users. This requires tha
included in any data mining metho
the necessity to automatically capt
vidual behaviors. Due to the potent
(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 pr
tional clustering for a large numbe
collection of transactions, transacti
covering groups of homogeneous t
common items [30]. In the state c
transactional clustering require eith
that is not automatic, or an extrem
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

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

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.

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

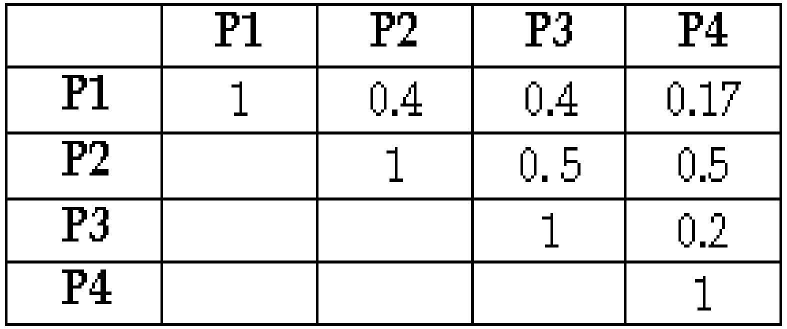
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or 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.

Exercises Transactional Clustering

Rock – Exercise 1

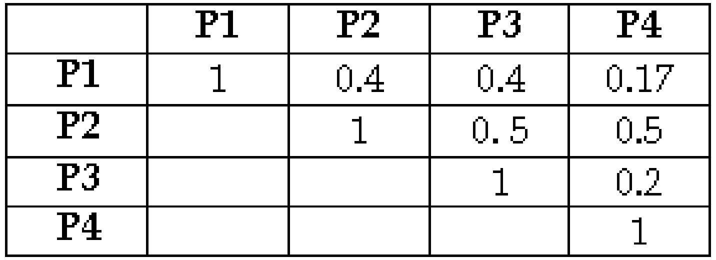
- Suppose we have four verses contains some subjects, as follows:
- P1={ judgment, faith, prayer, fair}
- P2={ fasting, faith, prayer}
- P3={ fair, fasting, faith}
- P4={ fasting, prayer, pilgrimage}
- **the similarity threshold = 0.3, and number of required cluster is 2.**

Using Jaccard coefficient as a similarity measure, we obtain the following similarity table



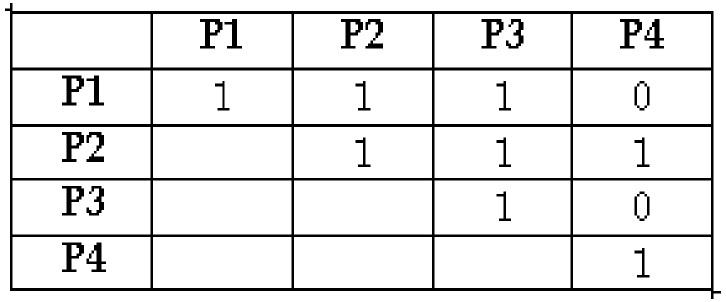
	P1	P2	P3	P4
P1	1	0.4	0.4	0.17
P2		1	0. 5	0.5
P3			1	0.2
P4				1

Rock – Exercise 1



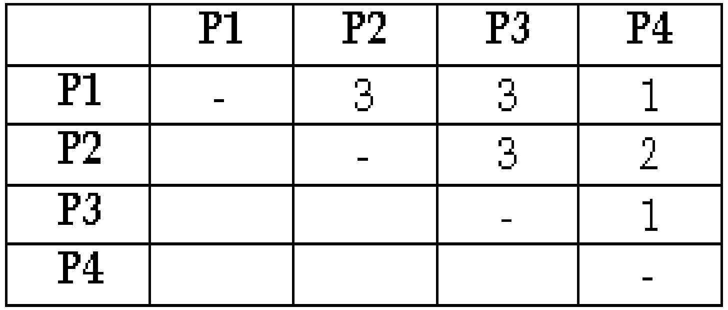
	P1	P2	P3	P4
P1	1	0.4	0.4	0.17
P2		1	0. 5	0.5
P3			1	0.2
P4				1

- Since we have a similarity threshold equal to 0.3, then we derive the adjacency table: →



	P1	P2	P3	P4
P1	1	1	1	0
P2		1	1	1
P3			1	0
P4				1

- By multiplying the adjacency table with itself, we derive the following table which shows the number of links (or common neighbors): →



	P1	P2	P3	P4
P1	-	3	3	1
P2		-	3	2
P3			-	1
P4				-

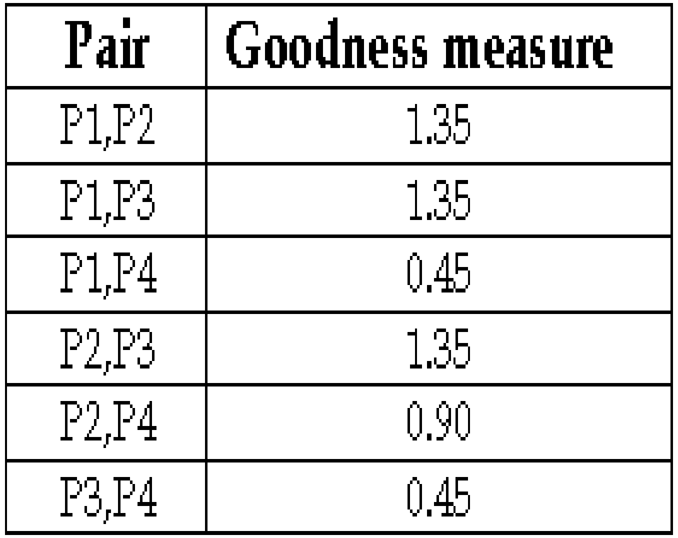
Rock – Exercise 1

- we compute the goodness measure for all adjacent points ,assuming that

$$g(P_i, P_j) = \frac{link[P_i, P_j]}{(n+m)^{1+2f(\theta)} - n^{1+2f(\theta)} - m^{1+2f(\theta)}}$$

- $f(\theta) = 1-\theta\ /\ 1+\theta$ = 1-0.3 /1+0.3 = 0.54

- we obtain the following table➔



Pair	Goodness measure
P1,P2	1.35
P1,P3	1.35
P1,P4	0.45
P2,P3	1.35
P2,P4	0.90
P3,P4	0.45

- we have an equal goodness measure for merging ((P1,P2), (P2,P3), (P3,P1))

Rock – Exercise 1

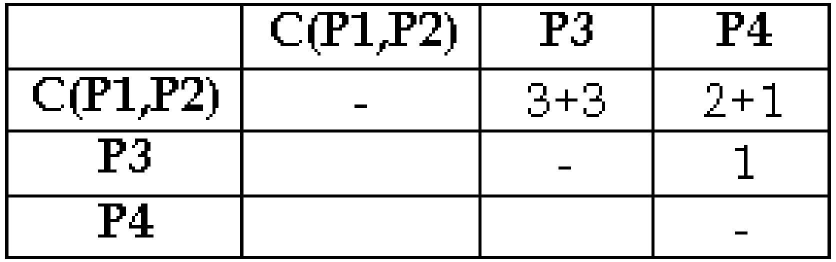
- Now, we start the hierarchical algorithm by merging, say P1 and P2.

- A new cluster (let’s call it C(P1,P2)) is formed.

- It should be noted that for some other hierarchical clustering techniques, we will not start the clustering process by merging P1 and P2, since Sim(P1,P2) = 0.4,which is not the highest. But, ROCK uses the number of links as the similarity measure rather than distance.

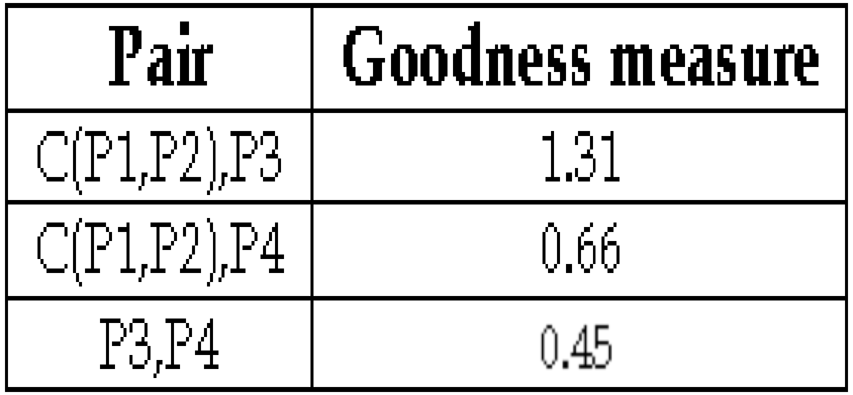
Rock – Exercise 1

- Now, after merging P1 and P2, we have only three clusters. The following table shows the number of common neighbors for these clusters:→



	C(P1,P2)	P3	P4
C(P1,P2)	-	3+3	2+1
P3		-	1
P4			-

- Then we can obtain the following goodness measures for all adjacent clusters:→



Pair	Goodness measure
C(P1,P2),P3	1.31
C(P1,P2),P4	0.66
P3,P4	0.45

Rock – Exercise 1

- Since the number of required clusters is 2, then we finish the clustering algorithm by merging C(P1,P2) and P3, obtaining a new cluster C(P1,P2,P3) which contains {P1,P2,P3} leaving P4 alone in a separate cluster.

Rock – Exercise 2

- Given the following similarity matrix find the clustering result knowing that the similarity threshold = 0.4, and number of required cluster is 2.

	p1	p2	p3	p4	p5
p1	1	0.7	0.2	0.5	0.5
p2		1	0.6	0.8	0.1
p3			1	0.5	0.4
p4				1	0.3
p5					1

Rock – Exercise 2 – Solution

	p1	p2	p3	p4	p5
p1	1	0.7	0.2	0.5	0.5
p2		1	0.6	0.8	0.1
p3			1	0.5	0.4
p4				1	0.3
p5					1

	p1	p2	p3	p4	p5
p1	1	1	0	1	1
p2	1	1	1	1	0
p3	0	1	1	1	1
p4	1	1	1	1	0
p5	1	0	1	0	1

Rock – Exercise 2 – Solution

	p1	p2	p3	p4	p5
p1	1	1	0	1	1
p2	1	1	1	1	0
p3	0	1	1	1	1
p4	1	1	1	1	0
p5	1	0	1	0	1

	p1	p2	p3	p4	p5
p1	-	3	3	3	2
p2		-	3	4	2
p3			-	3	2
p4				-	2
p5					-

Rock – Exercise 2 – Solution

- $f(\theta) = 1-\theta \,/\, 1+\theta$ = 1-0.4 /1+0.4 = 0.43
- 1 + 2 $f(\theta) = 1.86$

$$g(P_i, P_j) = \frac{link[P_i, P_j]}{(n+m)^{1+2f(\theta)} - n^{1+2f(\theta)} - m^{1+2f(\theta)}}$$

	p1	p2	p3	p4	p5
p1	-	3	3	3	2
p2		-	3	4	2
p3			-	3	2
p4				-	2
p5					-

	p1	p2	p3	p4	p5
p1	-	1.84	1.84	1.84	1.22
p2		-	1.84	2.45	1.22
p3			-	1.84	1.22
p4				-	1.84
p5					-

Rock – Exercise 2 – Solution

- $f(\theta) = 1-\theta\ /\ 1+\theta$ = 1-0.4 /1+0.4 = 0.43
- 1 + 2 $f(\theta) = 1.86$

$$g(P_i, P_j) = \frac{link[P_i, P_j]}{(n+m)^{1+2f(\theta)} - n^{1+2f(\theta)} - m^{1+2f(\theta)}}$$

	p1	p2	p3	p4	p5
p1	-	3	3	3	2
p2		-	3	4	2
p3			-	3	2
p4				-	2
p5					-

	p1	p2p4	p3	p5
p1	-	6	3	2
p2p4		-	6	4
p3			-	2
p5				-

	p1	p2p4	p3	p5
p1	-	1.94	1.84	1.22
p2p4		-	1.94	1.29
p3			-	1.22
p5				-

- *Final Clusters: p1234 p5*

Clope Exercise 1

Transactions: abc, abc, ab, ad, def, ade, ade

Split1:

- 4 transactions: abc, abc, ab, ad
  - a:4, b:3, c:2, d:1 -> S=10; W=4; H=10/4=2,5; H/W=2,5/4=0,625
- 3 transactions: def, ade, ade
  - a:2, d:3, e:3, f:1 -> S=9; W=4; H=9/4=2,25; H/W=2,25/4=0,56

Split2:

- 2 transactions: abc, abc, ab
  - a:3, b:3, c:2 -> S=8; W=3; H=8/3=3,6; H/W=0,88
- 2 transactions: ad, def, ade, ade
  - a:3, d:4, e:3, f:1 -> S=11; W=4; H=11/4=2,75; H/W=2,75/4=0,68

Split1 is the best clustering considering r=2

Profit(Split1) = (10/4^2 * 4 + 9/4^2 * 3) /7 = 0.59

Profit(Split2) = (8/3^2 * 3 + 11/4^2 * 4) /7 = 0.77

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

Clope Exercise 2

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

Split1:

- 4 transactions: abc, abc, ab, a
  - a: 4, b:3, c: 2 -> sol: S=9; W=3; H=9/3=3; H/W=1
- 3 transactions: def, de, de
  - d: 3, e:3, f: 1 -> sol: S=7; W=3; H=7/3=2.33; H/W=0.77

Split2:

- 2 transactions: abcd, ab
  - a: 2, b:2, c: 1, d:1 -> sol: S=6; W=4; H=6/4=1.5; H/W=0.37
- 2 transactions: ec, ec
  - e:2, c: 2 -> sol: S=4; W=2; H=4/2=2; H/W=1

Split1 is the best clustering considering r=2

Profit(Split1) = $(9/3^2 * 4 + 7/3^2 * 3) /7 = 0.90$

Profit(Split2) = $(6/4^2 * 2 + 4/2^2 * 2) /4 = 0.16$