TRANSACTIONAL CLUSTERING

Anna Monreale
University of 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

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

Categorical data

Hairs	Eyes
brown	black
blond	blue
black	green
red	brown

Types of Attributes

- Boolean attribute and Categorical attribute
  - 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
- Used to cluster the customers so that customers with similar buying pattern are in a cluster. 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

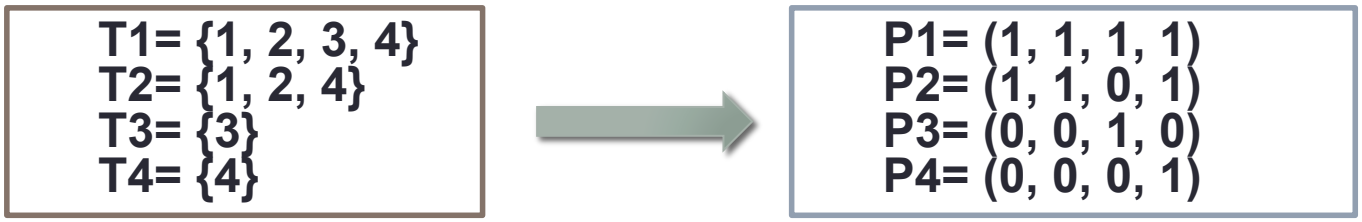
Criterion Function

- Given **n** data points in a **d**-dimensional space, a clustering algorithm partitions the data points into **k** clusters
- Partitional Clustering divides the point space into k clusters that optimize a certain criterion function
- Criterion function **F** for metric spaces commonly used is Euclidean Distance
- Criterion function **F** attempts to **minimize distance of every point from the mean of the cluster** to which the point belongs
- Another approach is iterative hill climbing technique

Shortcomings of Traditional Clustering Algorithms (1)

- 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 Algorithms (2)

- If we use the centroid-based 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

Algorithms for categorical data

- Extensions of *k*-means
  - *k*-modes
- ROCK
- CLOPE
- TX-Means

k-modes

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

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

$$w_{i,l} \in \{0, 1\}, \quad 1 \le i \le n,\ 1 \le l \le k$$

$X = \{ X_1 , \ldots, X_n \}$ 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 uses Euclideian 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. 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 ech cluster
4. Repeat the steps 2 and 3 until no object changes the assigned cluster

Time Comlexity like K-means

ROCK: RObust Clustering using linK

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

The Neighbours Concept

- It captures a notion of **similarity**

  A and B are neighbours if $sim(A, B) \geq \theta$

- ROCK uses the **Jaccard coefficient**

  $sim(A, B) = |A \cap B| / |A \cup B|$

A = { 1 , 3 , 4 , 7 }

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

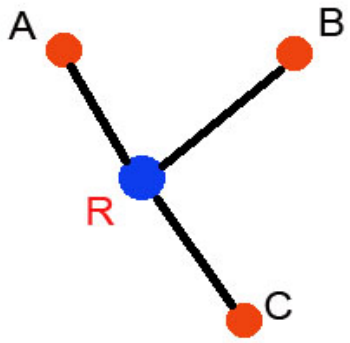
$$sim(A,B) = \frac{3}{6} = \frac{1}{2} = 0.5$$

ROCK - links

- A **link** defines the number of common neighbors between two objects

$$Link(A, B) = |neighbor(A) \cap neighbor(B)|$$

- **Higher values of *link(A, B)* means higher probability that p_i and p_j belong to the same cluster**
- **Similarity** is **local** while **link** is capturing **global** information
- Note that a point is considered as a neighbour of itself as well
- There is a link from each neighbour of the “root” point back to itself through the root
- Therefore, if a point has x neighbours, then x^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

An Example

- Data consisting of 6 Attributes {Book, Water, Sun, Sand, Swimming, Reading}
- Records
  A. {Book}
  B. {Water, Sun, Sand, Swimming}
  C. {Water, Sun, Sand, Reading}
  D. {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. Neighbours:

  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

- **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 on disk*

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 on disk*

Complexity:

$$O(n^2 + n\,L_a\,L_{max} + n^2 \log n)$$

L_a= average n. of links for object
L_{max}= max n. of links for object

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 disk 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:

a. places each single data point into a separate cluster
b. compute the similarity measure for all pairs of clusters
c. merge the two clusters with the highest similarity (goodness measure)
d. Verify a stop condition. If it is not met then go to step b

ROCK Clustering algorithm

Label data on disk

- Finally, the remaining data points in the disk are assigned to the generated clusters.

- This is done by selecting a random sample Li from each cluster Ci, then we assign each point p to the cluster for which it has the strongest linkage with Li.

Categorical Attributes Handling

- Reduction of Records to Transactions
- **For every attribute A and value u, an item A.u is introduced**
- A Transaction includes A.u if and only if the attribute value of A is u
- If the value of an attribute is missing in the record, then the corresponding transaction does not contain items for the attribute
- So, missing values are ruled out “magically”!
- That is, we measure the similarity of two records based only on the common items

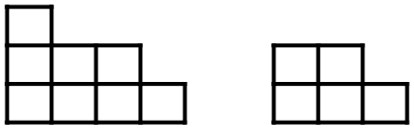
CLOPE (Clustering with LOPE)

- 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}

$$D(C) = set \text{ of items in C}$$
$$S(C) = \sum_{t_i \in C} |t_i|$$
$$W(C) = |D(C)|$$
$$H(C) = S(C) / W(C)$$

Clustering 1

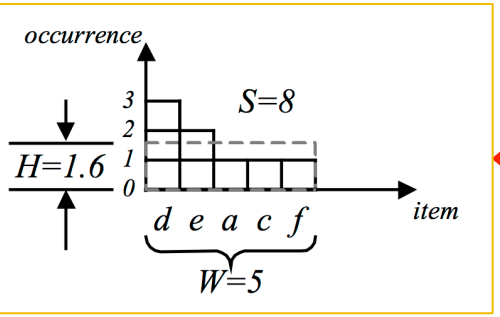


a b c d d e f

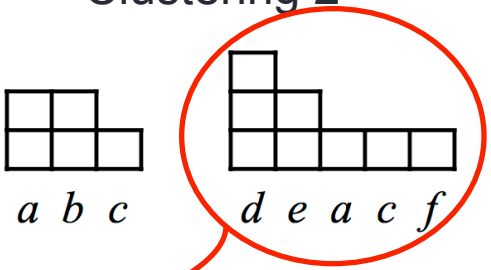
H=2.0, W=4 H=1.67, W=3

{ab, abc, acd} {de, def}

H/W=0.5 H/W=0.55



Clustering 2



a b c d e a c f

H=1.67, W=3 H=1.6, W=5

{ab, abc} {acd, de, def}

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

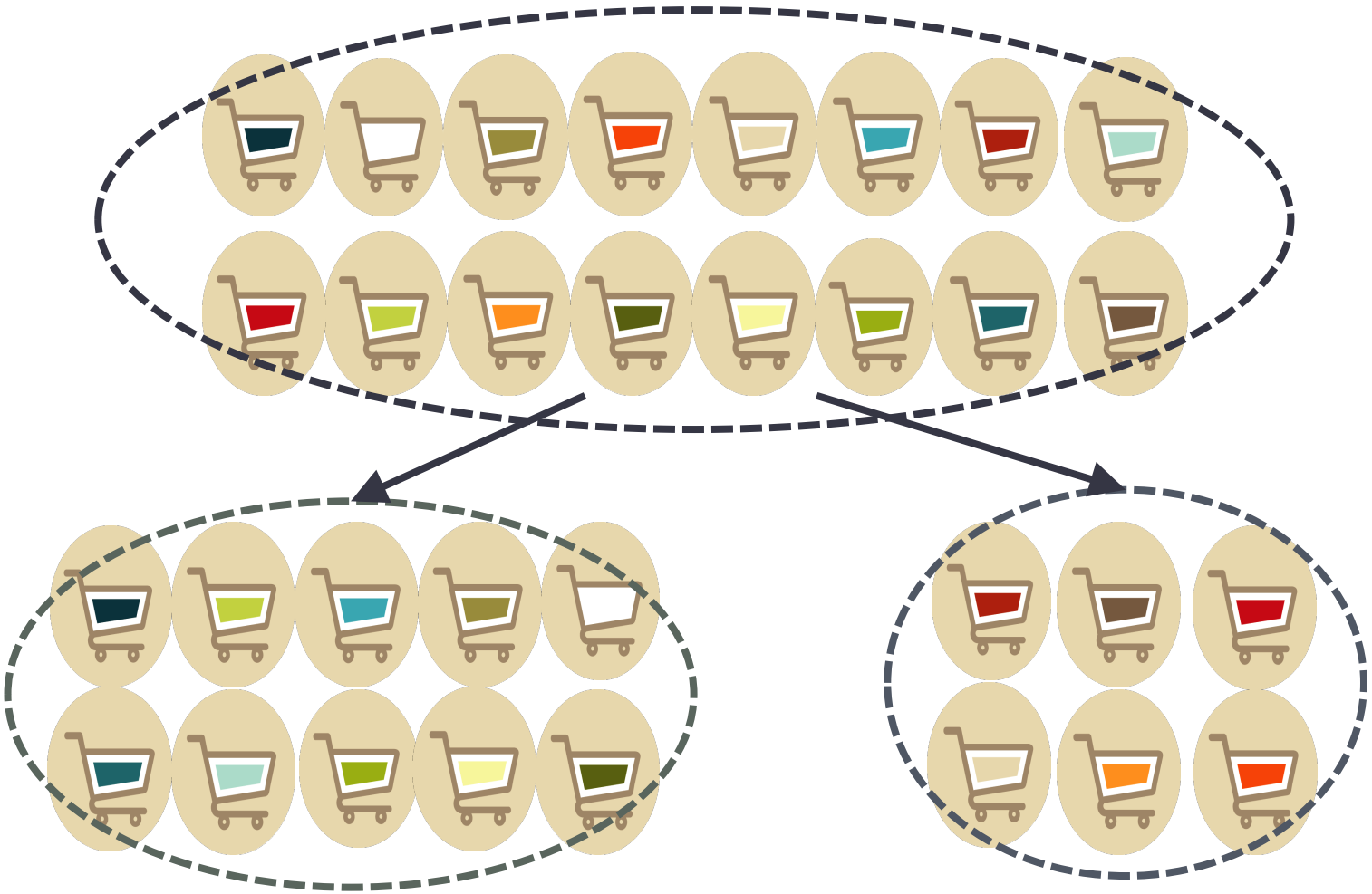
```
    /* Phrase 1 - Initialization */
1:  while not end of the database file
2:      read the next transaction ⟨t, unknown⟩;
3:      put t in an existing cluster or a new cluster C_i
          that maximize profit;
4:      write ⟨t, i⟩ back to database;

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

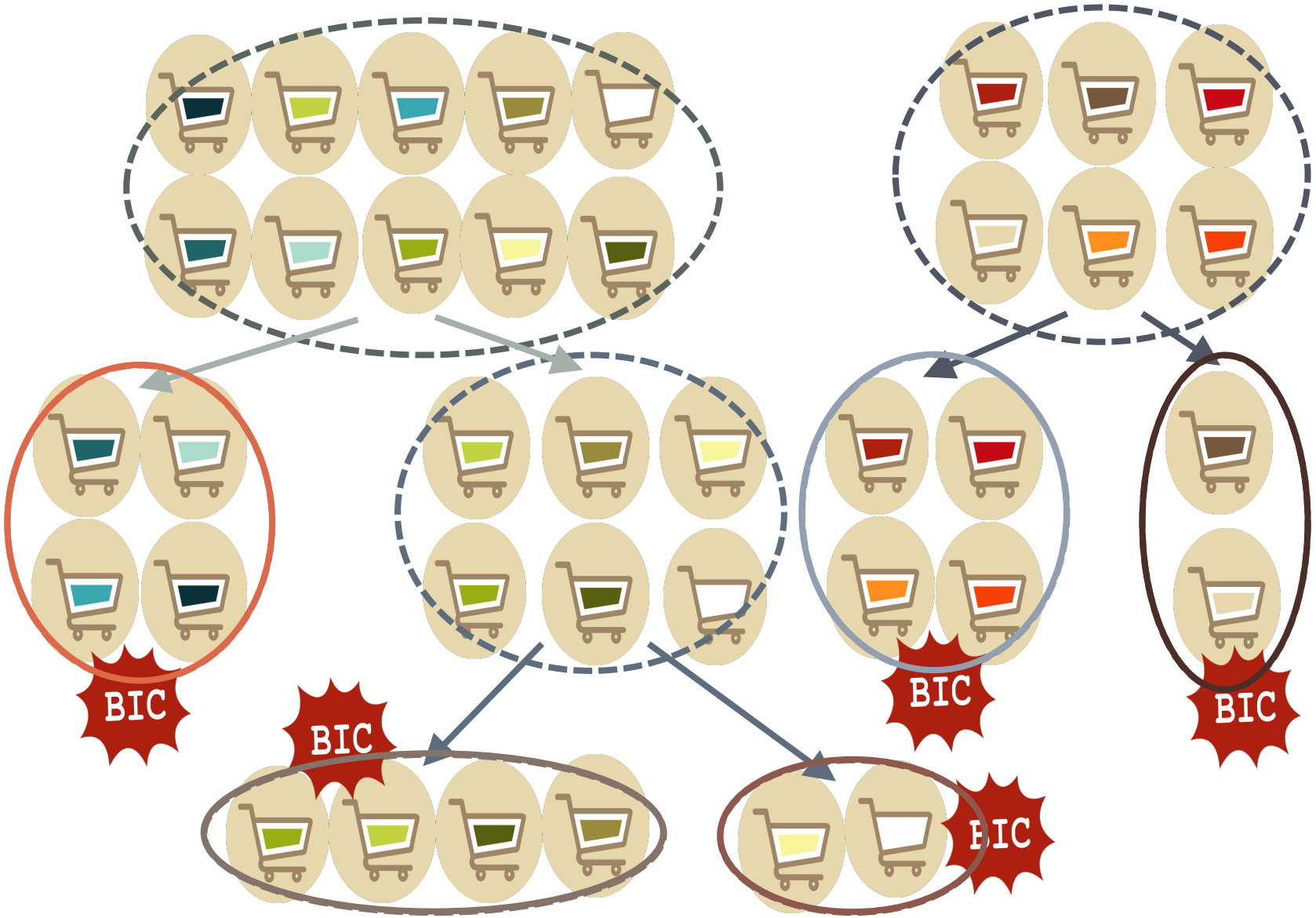
TX-MEANS

- 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

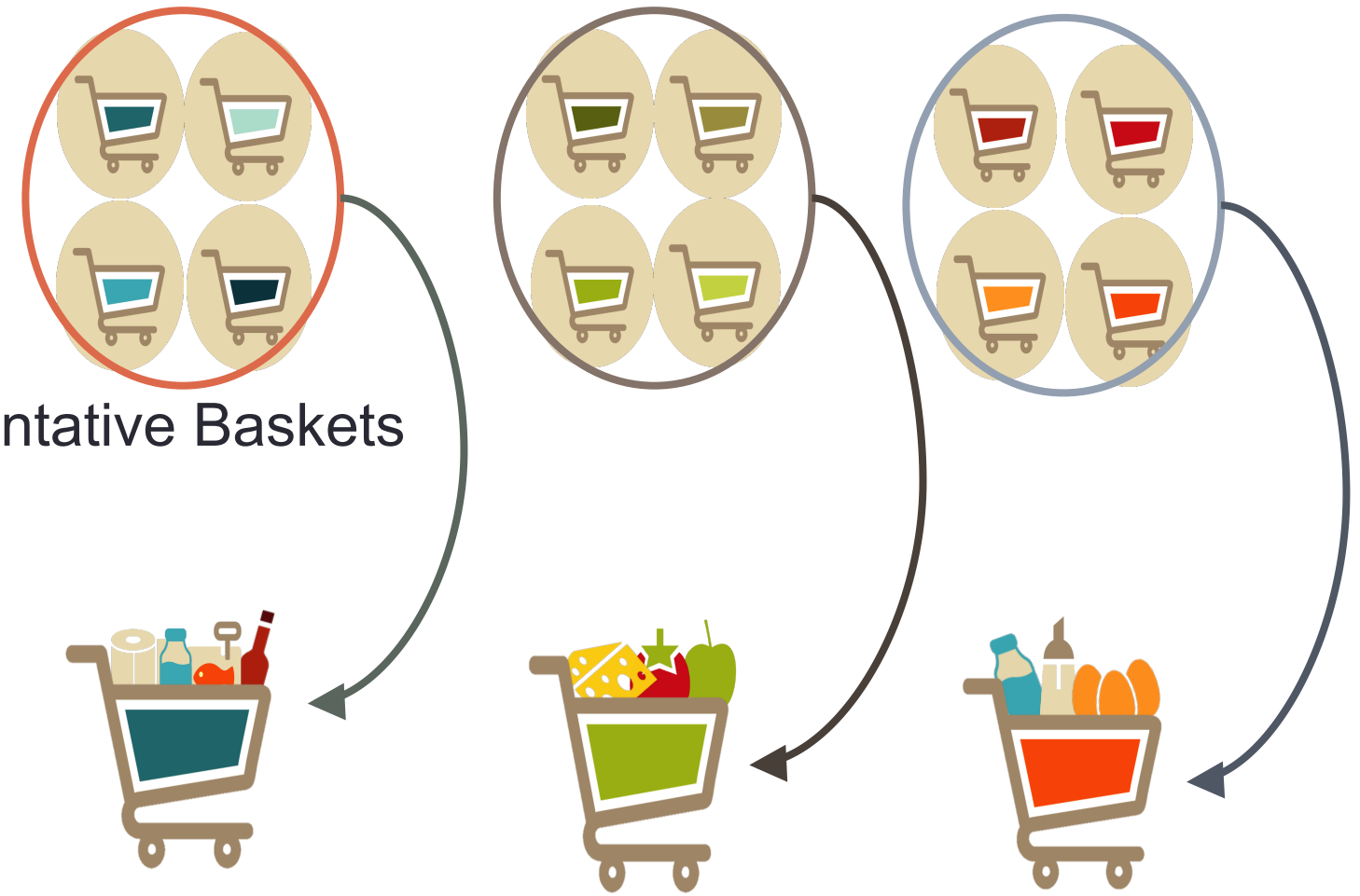


How It Works 2/3



How It Works 3/3

- Clusters
- Representative Baskets



TX-Means Algorithm

```
TXMEANS(B: baskets):
• r <-- 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;
```

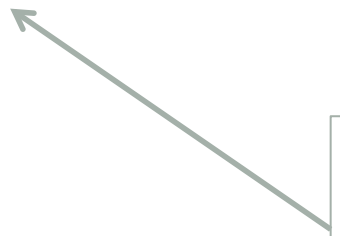
representative basket

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)

Termination & Complexity

- TX-Means terminates for any input:
  - `GETREPR` terminates because `I` becomes empty
  - `BISECTBASKET` terminates because 2-means terminates: the loop stops when the SSE does not strictly increase
  - `TXMEANS` terminates because at each iteration replace a cluster with strictly smaller ones, at worst all singletons are returned

- The complexity of TX-Means is **O(It • N • K • D):**
  - **It** is the number of iterations required to convergence by **`bisectBaskets`**,
  - **N** is the number of transactions in input,
  - **D** is the number of distinct items in the dataset, and
  - **K** is the number of clusters detected.

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.