

# DM2 - Miscellanea

In quest'ultima parte del corso analizzeremo diversi argomenti.

## PART1: Sequential Pattern Mining

### Introduzione

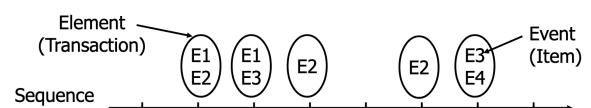
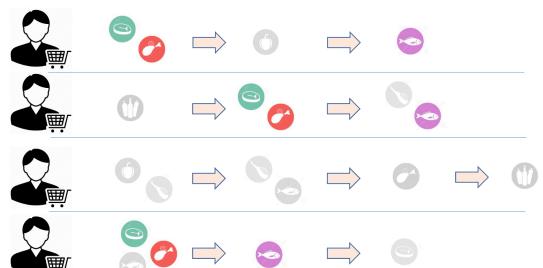
I **Frequent itemset** di DM1 non considerano la dimensione temporale, che andremo invece ad analizzare. Per farlo, in primo luogo dobbiamo trovare i **sequential pattern** (che ancora **non sono regole**).

Nell'immagine a lato, all'interno di una **sequenza** troviamo le varie **transazioni** (o elementi), che al loro interno degli **oggetti/item** (o eventi). Se la transazione ha un solo item è di tipo **singleton**, altrimenti è **complex**.

Il fattore temporale viene aggiunto nel momento in cui consideriamo anche il **timestamp** delle transazioni. Solitamente, lo si considera in modo "discretizzato", approssimando (v. rosso in figura).

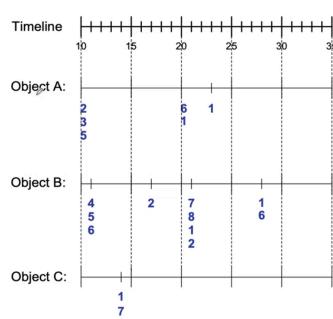
Objective: identify sequences that occur frequently

- Sequential pattern: { }



### Sequence Database:

Object	Timestamp	Events
A	10	2, 3, 5
A	20	6, 1
A	23	1
B	11	4, 5, 6
B	17	2
B	21	7, 8, 1, 2
B	28	1, 6
C	14	1, 8, 7



### Formal Definition of a Sequence

- A sequence is an ordered list of elements (transactions)  
 $s = < e_1, e_2, e_3, \dots >$
- Each element is attributed to a specific time or location
- Each element contains a collection of events (items)  
 $e_i = \{i_1, i_2, \dots, i_k\}$
- Length of a sequence,  $|s|$ , is given by the number of elements of the sequence
- A k-sequence is a sequence that contains k events (items)

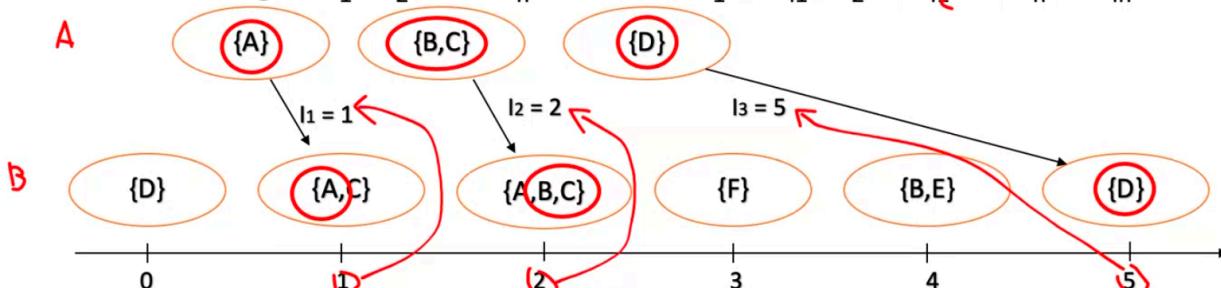
Notare che la lunghezza conta il numero di transazioni, il K quello degli item.

Dobbiamo introdurre il **concepto di subseguenza**

### ESERCIZIO: Sequential Pattern

Lezione 19-1, Minuto 27:10.  
Altri esercizi nelle slide

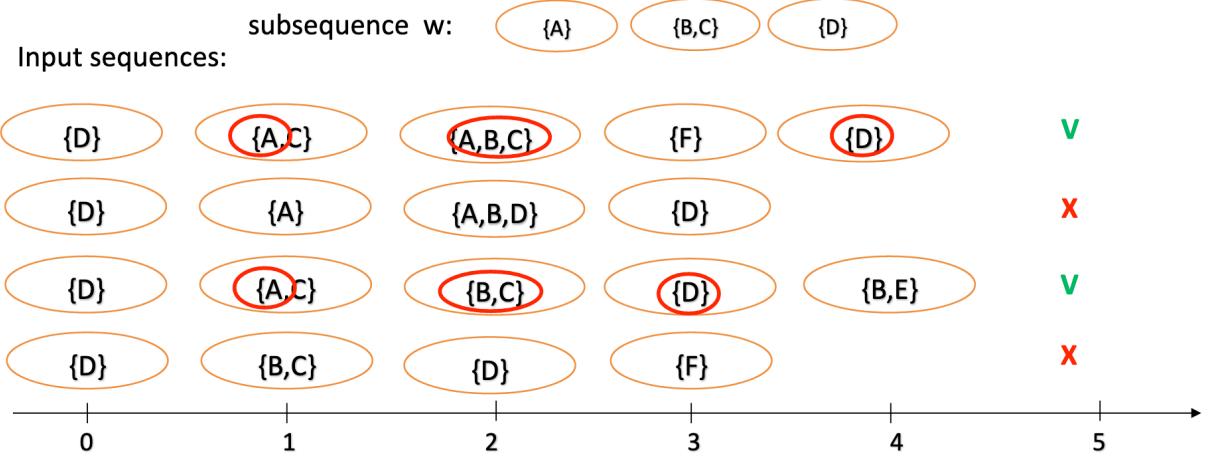
A sequence  $\langle a_1 a_2 \dots a_n \rangle$  is contained in another sequence  $\langle b_1 b_2 \dots b_m \rangle$  ( $m \geq n$ ) if there exist integers  $i_1 < i_2 < \dots < i_n$  such that  $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, \dots, a_n \subseteq b_{i_n}$



Data sequence	Subsequence	Contain?
$\langle \{2,4\} \{3,5,6\} \{8\} \rangle$	$\langle \{2\} \{3,5\} \rangle$	Yes
$\langle \{1,2\} \{3,4\} \rangle$	$\langle \{1\} \{2\} \rangle$	No
$\langle \{2,4\} \{2,4\} \{2,5\} \rangle$	$\langle \{2\} \{4\} \rangle$	Yes

Il sequential pattern fa anche uso del **support**: consideriamo solo le subsequences con un support superiore a un threshold.

The **support** of a subsequence  $w$  is the fraction of data sequences that contain  $w$



- **A sequential pattern**
  - is a **frequent** subsequence
  - i.e., a subsequence whose support is  $\geq \text{minsup}$

Il nostro scopo è trovare tutte le subsequences con support  $\geq \text{minsup}$ . Come farlo in modo efficiente?

### Generalized Sequential Pattern (GSP)

**Follows the same structure of Apriori**

- Start from short patterns and find longer ones at each iteration

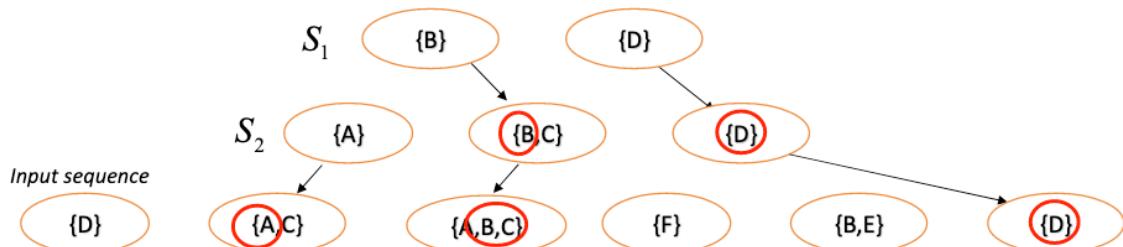
**Based on “Apriori principle” or “anti-monotonicity of support”**

- If one sequence  $S_1$  is contained in sequence  $S_2$ , then the support of  $S_2$  cannot be larger than that of  $S_1$ :

$$S_1 \subseteq S_2 \Rightarrow \text{sup}(S_1) \geq \text{sup}(S_2)$$

**Intuitive proof**

- Any input sequence that contains  $S_2$  will also contain  $S_1$



- **Follows the same structure of Apriori**
  - Start from short patterns and find longer ones at each iteration
- **Step 1:**
  - Make the first pass over the sequence database D to yield all the 1-element frequent sequences
- **Step 2:** Repeat until no new frequent sequences are found:
  - **Candidate Generation:**
    - Merge pairs of frequent subsequences found in the  $(k-1)$ th pass to generate candidate sequences that contain  $k$  items
  - **Candidate Pruning:**
    - Prune candidate  $k$ -sequences that contain infrequent  $(k-1)$ -subsequences
  - **Support Counting:**
    - Make a new pass over the sequence database D to find the support for these candidate sequences
  - **Candidate Elimination:**
    - Eliminate candidate  $k$ -sequences whose actual support is less than  $minsup$

- Given  $n$  events:  $i_1, i_2, i_3, \dots, i_n$ 
  - Candidate 1-subsequences:
  $\langle\{i_1\}\rangle, \langle\{i_2\}\rangle, \langle\{i_3\}\rangle, \dots, \langle\{i_n\}\rangle$

Le subsequenze sono composte da una sola transazione con un solo item.

- Candidate 2-subsequences:  
 $\langle\{i_1, i_2\}\rangle, \langle\{i_1, i_3\}\rangle, \dots, \langle\{i_1\} \{i_1\}\rangle, \langle\{i_1\} \{i_2\}\rangle, \dots, \langle\{i_{n-1}\} \{i_n\}\rangle$

Le subsequenze sono composte da una transazione con due item oppure da due transazioni con un solo item.

- Candidate 3-subsequences:  
 $\langle\{i_1, i_2, i_3\}\rangle, \langle\{i_1, i_2, i_4\}\rangle, \dots, \langle\{i_1, i_2\} \{i_1\}\rangle, \langle\{i_1, i_2\} \{i_2\}\rangle, \dots,$   
 $\langle\{i_1\} \{i_1, i_2\}\rangle, \langle\{i_1\} \{i_1, i_3\}\rangle, \dots, \langle\{i_1\} \{i_1\} \{i_1\}\rangle, \langle\{i_1\} \{i_1\} \{i_2\}\rangle, \dots,$

Le subsequenze sono composte da una transazione con tre item oppure da tre transazioni con un solo item oppure da due transazioni di cui 1+2 item.

## Candidate Generation

- **Base case ( $k=2$ ):**
  - Merging two frequent 1-sequences  $\langle\{i_1\}\rangle$  and  $\langle\{i_2\}\rangle$  will produce two candidate 2-sequences:  $\langle\{i_1\} \{i_2\}\rangle$  and  $\langle\{i_2\} \{i_1\}\rangle$
  - Special case:  $i_1$  can be merged with itself:  $\langle\{i_1\} \{i_1\}\rangle$
- **General case ( $k>2$ ):**
  - A frequent  $(k-1)$ -sequence  $w_1$  is merged with another frequent  $(k-1)$ -sequence  $w_2$  to produce a candidate  $k$ -sequence if the subsequence obtained by removing the **first event in  $w_1$**  is the same as the one obtained by removing the **last event in  $w_2$** 
    - The resulting candidate after merging is given by the sequence  $w_1$  extended with the **last event of  $w_2$** .
      - Simplest case:  $\langle\{d\}\{a\}\{b\}\rangle + \langle\{a\}\{b\}\{c\}\rangle = \langle\{d\}\{a\}\{b\}\{c\}\rangle$
      - If last two events in  $w_2$  belong to the same element  $\Rightarrow$  last event in  $w_2$  becomes part of the last element in  $w_1$ :  
 $\langle\{d\}\{a\}\{b\}\rangle + \langle\{a\}\{b,c\}\rangle = \langle\{d\}\{a\}\{b,c\}\rangle$
      - Otherwise, the last event in  $w_2$  becomes a separate element appended to the end of  $w_1$ :  
 $\langle\{a,d\}\{b\}\rangle + \langle\{d\}\{b\}\{c\}\rangle = \langle\{a,d\}\{b\}\{c\}\rangle$
    - Special case: check if  $w_1$  can be merged with itself
      - Works when it contains only one event type:  $\langle\{a\}\{a\}\rangle + \langle\{a\}\{a\}\rangle = \langle\{a\}\{a\}\{a\}\rangle$

Per la generazione di candidati si effettua il *merge* di due subsequenze. Procediamo eliminando il primo elemento della prima subseguenza e l'ultimo elemento della seconda subseguenza: se otteniamo subseguenza uguali si effettua il merge.

Il *pruning* si basa sulla scomposizione. Ad esempio ABC è scomposto in AA, AB, BC. Nel caso ABC fosse stato generato da AB e BC con il metodo visto sopra, sappiamo già il supporto di AB e BC.

### ESERCIZIO: GSP

Lezione 19-2, Minuto 27.

## Candidate Pruning

- Based on Apriori principle:

- If a k-sequence W contains a (k-1)-subsequence that is not frequent, then W is not frequent and can be pruned

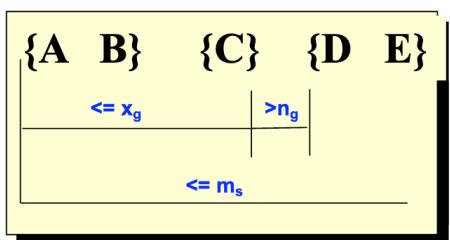
- Method:

- Enumerate all (k-1)-subsequence:
  - $\{a,b\}\{c\}\{d\} \rightarrow \{b\}\{c\}\{d\}, \{a\}\{c\}\{d\}, \{a,b\}\{d\}, \{a,b\}\{c\}$
- Each subsequence generated by cancelling 1 event in W
  - Number of (k-1)-subsequences = k
- Remark: candidates are generated by merging two “mother” (k-1)-subsequences that we know to be frequent
  - Correspond to remove the first event or the last one
  - Number of significant (k-1)-subsequences to test =  $k - 2$
  - Special cases: at step  $k=2$  the pruning has no utility, since the only (k-1)-subsequences are the “mother” ones

Si potrebbe tuttavia non essere interessanti a trovare un pattern che presenta una distanza temporale molto ampia, o molto minima, fra due transazioni. Si introducono quindi dei **Timing Constraints**.

### Define 3 types of constraint on the instances to consider

- E.g. ask that the pattern instances last no more than 30 days



- |                      |  |
|----------------------|--|
| $x_g$ : max-gap      | → Each element of the pattern instance must be <b>at most</b> $x_g$ time after the previous one  |
| $n_g$ : min-gap      | → Each element of the pattern instance must be <b>at least</b> $n_g$ time after the previous one |
| $m_s$ : maximum span | → The overall duration of the pattern instance must be <b>at most</b> $m_s$                      |

Ci sono due modi per applicare i timing constraints: come postprocessing o modificando il pruning. Nel primo caso si ha dispendio di tempo ed energie, nel secondo si crea qualche problema con l'applicazione dell'*a priori*. Il problema non si verifica con max-span o min-gap, ma con il max-gap. Per ovviare a questo problema, introduciamo la nozione di **contiguous subsequences**.

s is a contiguous subsequence of

$$w = \langle e_1 \rangle \langle e_2 \rangle \dots \langle e_k \rangle$$

if any of the following conditions hold:

1. s is obtained from w by deleting an item from either  $e_1$  or  $e_k$
2. s is obtained from w by deleting an item from any element  $e_i$  that contains more than 2 items
3. s is a contiguous subsequence of  $s'$  and  $s'$  is a contiguous subsequence of w (recursive definition)

Examples:  $s = \langle \{1\} \{2\} \rangle$

- is a contiguous subsequence of  $\langle \{1\} \{2\} \rangle, \langle \{1\} \{2\} \{3\} \rangle$ , and  $\langle \{3\} \{4\} \{1\} \{2\} \{2\} \{3\} \{4\} \rangle$
- is not a contiguous subsequence of  $\langle \{1\} \{3\} \{2\} \rangle$  and  $\langle \{2\} \{1\} \{3\} \{2\} \rangle$

Key point: avoids internal “jumps”  
Not interesting for our usage

Quando consideriamo il maxgap, la condizione di pruning è quindi modificata.

### ESERCIZIO: Pattern Mining

Lezione 20-1, Minuto 36.

#### Without maxgap constraint:

- A candidate  $k$ -sequence is pruned if at least one of its  $(k-1)$ -subsequences is infrequent

#### With maxgap constraint:

- A candidate  $k$ -sequence is pruned if at least one of its **contiguous**  $(k-1)$ -subsequences is infrequent
- Remark: the “pruning power” is now reduced
  - Less subsequences to test for “killing” the candidate

#### Riassuntone

-**Sequenza:** Lista ordinata di transazioni.

-**Transazione:** Un set di item.

-**Item:** I singoli componenti di una transazione.

-**Lunghezza di una Sequenza:** Il numero delle sue transazioni. Ma il  $k$  di una sequenza è il numero dei suoi item.

-**Timestamp:** Indicazione temporale della transaction.

-**Subsequenza:** Una sequenza  $\langle a_1, a_2, \dots, a_n \rangle$  è contenuta in un'altra sequenza  $\langle b_1, b_2, \dots, b_n \rangle$  se tutti gli elementi di  $a$  sono sottoinsieme dei rispettivi elementi di  $b$ .

-**Sequential Pattern:** Una subseguenza il cui support è  $>\text{minsup}$

-**Support di una subseguenza w:** La percentuale di frequenze che contengono  $w$ .

#### Generalized Sequential Pattern (GSP)

Se  $S_2$  è una subseguenza di  $S_1$ , tutte le sequenze che contengono  $S_1$  contengono anche  $S_2$ .

Segue lo stesso principio dell'Apriori.

**Apriori:** se  $S_2$  è una subseguenza di  $S_1$ , il support di  $S_2$  è minore o uguale a quello di  $S_1$ .

STEP1:

Troviamo tutte le sequenze frequenti con un solo elemento (es.: A, B, C)

STEP2:

*Candidate Generation:* Merge dei candidati con un solo elementi ( $A + B \Rightarrow AB$ )

*Candidate Pruning:* Eliminazioni dei candidati che contengono subseguenze non frequenti

*Support Counting:* Controllo del support dei candidati

*Candidate Elimination:* Eliminazione dei candidati con support < minsup

#### Timing Constraints:

-max-span

-min-gap

-max-gap

**maxgap crea problemi con l'Apriori.** Per l'Apriori non è possibile che una subseguenza abbia un support maggiore della "sequenza padre". Qui può invece accadere. Ad esempio:

<2, 5> con support 30%

<2, 3, 5> con support 60%

Questo perché altre occorrenze di <2, 5>, che avrebbero reso il support  $\geq$  60%, non sono state considerate poiché non rispettavano il maxgap.

## PART2: Advanced Clustering

Tratteremo adesso estensioni di algoritmi visti in Data Mining 1.

### Bisecting K-Means

Estensione del K-Means. È sempre un problema scegliere il buon K per il K-Means e distribuzione non gaussiane possono essere problematiche. Il Bisecting pone delle soluzioni, con un modello a metà strada con quello gerarchico (ma attenzione: il processo nel gerarchico è "bottom-up", qui si procede a dividere). All'inizio abbiamo un "2-Means", si procede a dividere in due quello dei due cluster con SSE più alto.

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

---

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

---

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

### X-Means

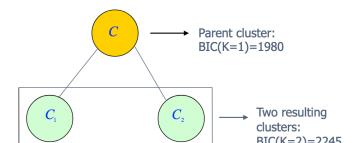
Poco conosciuto (non presente in SKLearn), ma molto utile: non richiede di fissare *a priori* il numero di cluster. Utilizza il Bayesian Information Criterion per stoppare lo *splitting* (e non andare così in *oversplitting* come può accadere con il B K-Means). L'utente deve precisare sono r1, rmax: il range in cui può essere K. Successivamente viene eseguito il bisecting k-means in ogni cluster: se il BIC del figlio è > del BIC del padre la divisione viene accettata, altrimenti no. Ad ogni modo arrivati a  $k > r_{\text{max}}$  l'algoritmo si ferma.

Se abbiamo settato K fra 3 e 10, dopo aver fatto  $K = 3$  e aver ottenuto *tot* cluster e un Global BIC, l'operazione verrà ripetuta per i successivi valori. Alla fine verrà scelta il modello con il BIC più alto.

### Bayesian Information Criterion (BIC)

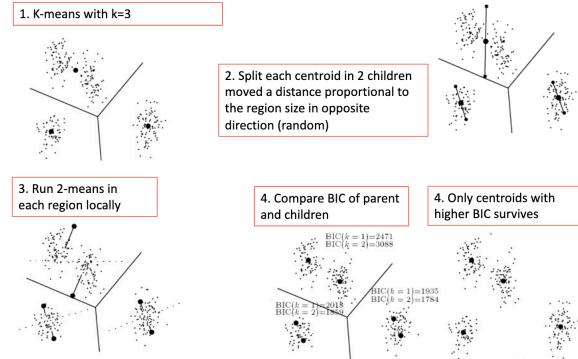
---

- 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.



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

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



L'idea dietro il BIC è che riesca ad approssimare un certo clustering descrivendo i *real cluster* nei dati, attraverso la log-likelihood.

## Expectation Maximization Procedure - Le origini del K-Means

Per descrivere un dataset, possono servire più distribuzioni. Serve quindi un *mixture model*. A questa necessità risponde l'**Expectation Maximization Algorithm**. Si parte da alcuni parametri scelti casualmente. Nell'**E Step** si stima a quali di questi parametri i valori nel dataset si avvicinano. Nel **M Step** si stimano nuovi parametri. **È molto simili al K-Means, ma è un approccio soft, meramente probabilistico.**

In order to understand our data, we will assume that there is a **generative process** (a **model**) that creates/describes the data, and we will try to find the model that **best fits** the data.

- Models of different complexity can be defined, but we will assume that our model is a **distribution from which data points are sampled**
  - Example: the data is the height of all people in Greece
- In most cases, a single distribution is not good enough to describe all data points: **different parts of the data follow a different distribution**
- Example: the data is the height of all people in Greece and China
  - We need a **mixture model**
  - Different distributions correspond to different clusters in the data.

## Expectation Maximization Algorithm

- Initialize the values of the parameters in  $\Theta$  to some random values
- Repeat until convergence
  - E-Step: Given the parameters  $\Theta$  estimate the membership probabilities  $P(G_j)$ .
  - M-Step: Given the probabilities  $P(G_j|x_i)$ , calculate the parameter values  $\Theta$  that (in expectation) maximize the data likelihood
- Examples
  - E-Step: Assignment of points to clusters
    - K-Means: **hard** assignment, EM: **soft** assignment
  - M-Step: Parameters estimation
    - K-Means: Computation of centroids, EM: Computation of the new model parameters

## EM in K-Means

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

## Mixture Gaussian Model - Il fratello del K-Means

No, c'è un limite a tutto.

## OPTICS - Ordering Points To Identify the Clustering Structure

### - L'evoluzione del DBSCAN

Risolve i problemi del DBSCAN con diverse dimensionalità e forme. È **parameter-free!** Rispetto al DBSCAN c'è il concetto di **Core Distance**, valore minimo del raggio per classificare un punto come core AKA

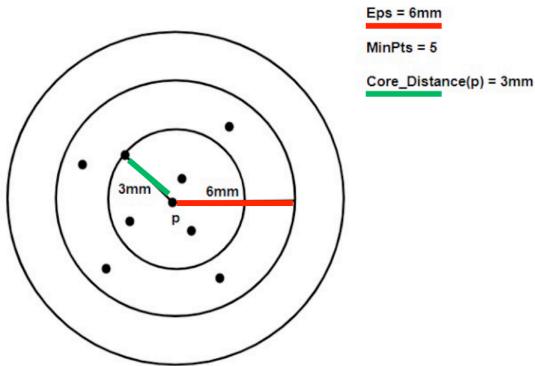
- Produces a special order of the dataset wrt its density-based clustering structure.
- This cluster-ordering contains info equivalent to the density-based clusterings corresponding to a broad range of parameter settings.
- Good for both automatic and interactive cluster analysis, including finding intrinsic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

OPTICS requires two parameters:

- $\epsilon$ , which describes the maximum distance (radius) to consider,
- MinPts, describing the number of points required to form a cluster

**Core point.** A point  $p$  is a core point if at least MinPts points are found within its  $\epsilon$ -neighborhood.

**Core Distance.** It is the **minimum** value of radius required to classify a given point as a core point. If the given point is not a Core point, then its Core Distance is undefined.



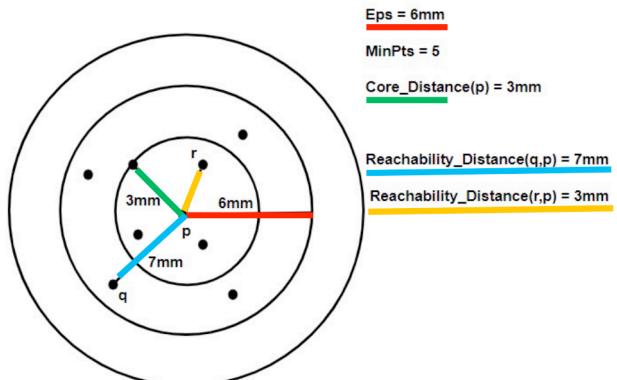
distanza massima da p e un punto nella vicinanza (altrimenti è core).

Un'altra nozione è quella di **reachability distance**. Per un punto Q è il massimo fra (Core Distance P) e (Distanza P-Q). Ne segue che sarà la Core Distance se il punto Q interessato è fra i vicini di P, altrimenti è pari alla distanza P-Q.

**Reachability Distance.** The reachability distance between a point  $p$  and  $q$  is the **maximum** of the Core Distance of  $p$  and the Distance between  $p$  and  $q$ .

The Reachability Distance is not defined if  $q$  is not a Core point. Below is the example of the Reachability Distance.

In other words, if  $q$  is within the core distance of  $p$  then use the core distance, otherwise the real distance.



Nella slide c'è un errore: is not defined if  $P$  is not...

### Riformulo vedendo Mirò:

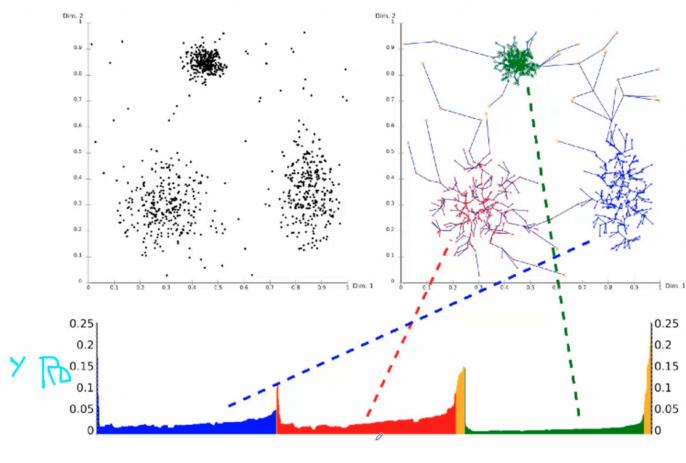
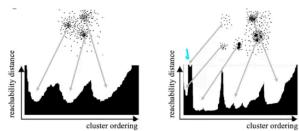
1. Si fissano i MinPts (es =5) e il Raggio massimo
2. La core distance è la distanza dal MinPts più lontano, si crea così un "sottocerchio" (core distance neighborhood)
3. La Reach Dist sarà uguale alla Core Distance per i punti dentro il sottocerchio, o uguale alla distanza P - punto per i punti fuori

## OPTICS Output

Nel grafico su X ci sono i vari punti, su Y la RechDist. Quando finisce un cluster si forma una montagnola. I punti arancione sono rumori, punti al di sopra di un certo threshold ai "bordi" dei cluster.

- Clusters are extracted

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



Ma come funziona esattamente l'OPTICS?

For each point  $p$  in the dataset

- Initialize the reachability distance of  $p$  as undefined

For each unprocessed point  $p$  in the dataset

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

Il raggio invero non è necessario, solo per stoppare l'algoritmo per piccoli cluster.

## OPTICS: The Radius Parameter

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

## PART3: Transactional Clustering

## K-Modes

Stessa procedura del *K-Means*, mira a minimizzare l'inter-cluster e massimizzare l'extra-cluster distance. La distanza è calcolata però in modo molto diverso: è **data dal mismatch**. Qui è 2.

	O <sub>C</sub>	O <sub>H</sub>	M	W	MB
X	1	1	0	1	0
Y	1	0	1	1	0

Al posto della media usiamo poi la **moda** per ogni colonna. Nel caso sopra la prima colonna avrebbe **1**, l'ultima **0**. Si tratta quindi di calcolare la frequenza ed usare la moda.

## K-Modes: Distance

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

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

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

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

## K-Modes: Mode

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

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

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

A	B	C	D	A	B	C	D	E	F
1	1	1	1	1	1	1	1	0	0
A	B	C	.	1	1	1	0	0	0
A	C	D	.	1	0	1	1	0	0
A	E	F	.	1	0	0	0	1	1
A	B	C	F	1	1	1	0	0	1
				1	1	1	1	0	0

## TX-Means

Analogia a X-Means. Fornisce un transazione rappresentativa.

Parte da un unico grande cluster, poi splitta in due, poi ancora... finché il BIC non indica di fermarsi, ottenendo così una gerarchia. **A differenza dell'X-Means non abbiamo il K iniziale** e non abbiamo quindi più modello comparati dal BIC.

Viene prima estratto un basket

rappresentativo da tutte le transazioni e lo mette in coda. Quando in coda c'è una transazione, rimuove gli elementi in comune fra più transazioni / basket. Si passa poi ad applicare il bisecting. A seconda del BIC si riporta il cluster nella coda, o fra i definitivi.

La funzione GETREP trova poi la transazione rappresentativa, partendo da quella presente in tutte le transazioni del cluster e aggiungendone man mano, finché la somma delle distanze (fra la transazione prototipica che stiamo costruendo e ogni singola transazioni del dataset) non cessa di decrescere. Si può *runnare* il TX-Means su un sample, e assegnare i valori non analizzati al cluster con il prototipo più simile.

## ROCK - RObust Clustering using linK

La similarità è calcolata usando Jaccard.

Per il ROCK due punti sono simili se  $\text{sim}(A, B) \geq \text{tot}$ .

Mentre per definire due transazioni come appartenenti allo stesso cluster si usa il concetto di link.

## 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

## 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.

## 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
- 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.

## Esercizio ROCK

Threshold = 0.3; Cluster = 2

P1={ judgment, faith, prayer, fair}

P2={ fasting, faith, prayer}

P3={ fair, fasting, faith}

P4={ fasting, prayer, pilgrimage}

Partiamo dalla transazioni e calcoliamo la similarità con Jaccard (elementi in comune / totale)

Ad esempio  $P1-P2 = 2/5 = 0.4$  Tutti i risultati sotto la threshold diventano 0, altrimenti 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

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

Controlliamo per ogni transazione gli elementi in comune con l'altra. **Otteniamo la matrice dei link.** Usiamo la formula della Goodness Measure: numero di link / numero di cluster atteso.

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

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

Possibili merge: (P1+P2), (P2+P3), (P3+P1). Proviamo P1+P2. Ripetiamo l'operazione.

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

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

Mergiamo C(P1+P2)+P3 in C(P1,P2,P3). Otteniamo due cluster: P1+P2+P3 e P4 da solo.

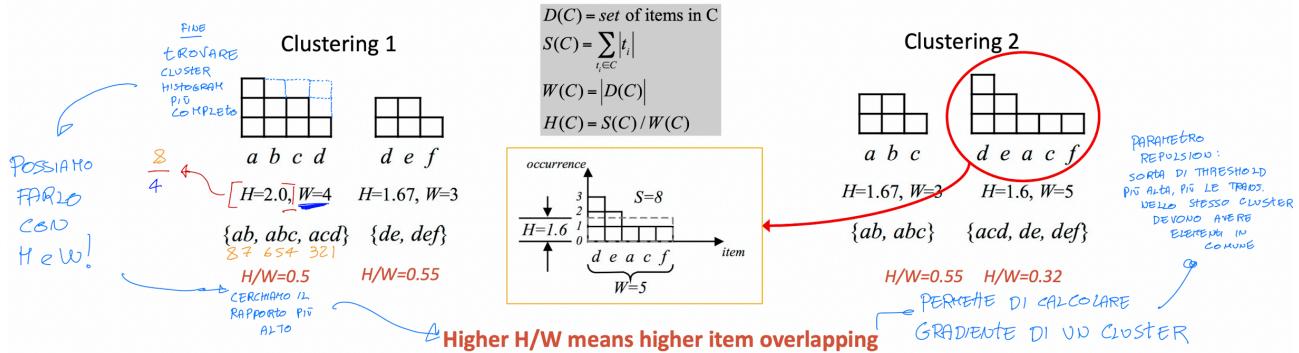
q.e.d.

## CLOPE

# 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}



Input: dataset, repulsion, maximum number of clusters

### • Phase 1

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

### • Phase 2

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

Abbiamo la transazione A B C D E F e tre possibili cluster. All'inizio sono tutti vuoti e arbitrariamente poniamo A in c1. Proviamo poi B in c1 con A, oppure da solo in c2. Scegliamo in base a dove si riesce a massimizzare il profit (cioè H/W), poniamo in c2. Quindi adesso abbiamo: A in c1, B in c2. Con la stessa logica poniamo anche C, D, E, F e otteniamo: A C F in c1, B D in c2, E in c3. Finisce così la fase 1. Nella fase 2 proviamo a spostare ogni transazioni (ad esempio A in c2 o in c3).