Lezione 3

Status	Done
Due date	@02/28/2025
Attach file	3_MiningInsiemiFrequenti.pdf

Mining di insiemi frequenti

Insiemi frequenti e regole d'associazione

Market-Basket Model

La market-basket analysis cerca di estrarre relazioni tra:

- **item** (oggetti) e **basket** (transazioni)
- basket= insieme di item (itemset)

L'obiettivo consiste nel trovare itemset frequenti, ovvero insiemi di item che vengono osservati assieme in molti basket.
A tal riguardo, definiamo supporto di un itemset il numero di basket in cui l'item è presente.
Sia σ la soglia di supporto fissata in fase di progetto. Diremo che un itemfrequente, se il suo supporto è maggiore o uguale a σ.

Dove trova applicazione?

- **Marketing:** carrelli del supermercato, trovare prodotti venduti assieme frequenti. Ciò può essere utile per applicare sconti mirati ai prodotti.
- **Ricerca di concetti correlati:** se gli item sono parole e i basket sono documenti trovare coppie o gruppi di parole che appaiono insieme in molti documenti.
- **Plagiarismo:** se gli item sono documenti e i basket sono frasi, si vogliono trovare documenti che hanno molte frasi in comune al fine di identificare dei casi di pla

Itemset frequenti - $\sigma = 2$

$B_1 = \{m, c, b\}$

$B_2 = \{m, p, j\}$

$B_3 = \{m, b\}$

$B_4 = \{c, j\}$

$B_5 = \{m, p, b\}$

$B_6 = \{m, c, b, j\}$

$B_7 = \{c, b, j\}$

$B_8 = \{b, c\}$

Itemset	Supp
{b}	6
{c}	5
{j}	4
{m}	5
{p}	2

Itemset	Supp
{b,c}	4
{b,j}	2
{b,m}	4
{b,p}	1
{c,j}	3
{c,m}	2
{c,p}	0
{j,m}	2
{j,p}	1
{m,p}	2

Itemset	Supp
{b,c,j}	2
{b,c,m}	2
{b,c,p}	0
{b,j,m}	1
{b,j,p}	0
{b,m,p}	1
{c,j,m}	1
{c,j,p}	0
{c,m,p}	0
{j,m,p}	1

Itemset	Supp
{b,c,j,m}	1
{b,c,j,p}	0
{b,c,m,p}	0
{b,j,m,p}	0
{c,j,m,p}	0

Itemset	Supp
{b,c,j,m,p}	0

Considero i prodotti che hanno almeno soglia 2. Poi faccio tutte le possibili coppie: tengo solo chi ha freq ≥ 2. Poi li raggruppo a 3, e così via.

La scelta della soglia determina il numero e la qualità dei risultati trovati:

- una soglia troppo alta potrebbe far perdere delle soluzioni interessanti
- una soglia troppo bassa implica un'esplosione di soluzioni (magari anche inutili)

In genere la soglia viene scelta pari all'1% del numero totale di basket.

Regole di associazione

La **ricerca** di itemset frequenti è generalmente finalizzata alla costruzione di rdi associazione (if-then).

> 💡 Una regola di associazione è denotata come un'implicazione I → j , dove I itemset e un j item.

La regola I → j indica che nei carrelli in cui si osserva l'itemset I è probabile osservare anche l'item j.

Confidenza di una regola

La probabilità dell'implicazione è quantificata dalla confidenza della regola. Indicato con $supp(I)$ il supporto dell'itemset I, allora la confidenza della regola è data da:

$$Conf(I \to j) = \frac{Supp(I \cup \{j\})}{Supp(I)}$$

La confidenza rappresenta la percentuale di carrelli dove si osserva sia I che j.

- $Supp(I \cup \{j\})$ è anche indicato come supporto alla regola
- $Supp(I)$ è definito come coverage della reagola, e misura quanto sia possibile applicare la regola

$B_1 = \{m, c, b\}$

$B_2 = \{m, p, j\}$

$B_3 = \{m, b\}$

$B_4 = \{c, j\}$

$B_5 = \{m, p, b\}$

$B_6 = \{m, c, b, j\}$

$B_7 = \{c, b, j\}$

$B_8 = \{b, c\}$

Regola	Supporto	Coverage	Confidenza
{b,c} → m	2	4	2/4 = 50%
{b,m} → c	2	4	2/4 = 50%
{c,m} → b	2	2	2/2 = 100%
{b,j} → m	1	2	1/2 = 50%
{c,j} → b	2	3	2/3 = 67%
{m,p} → b	1	2	1/2 = 50%

La confidenza è utile se il supporto di I è abbastanza alto.

Ad esempio, un I osservato lo 0,1% dei carrelli, allora qualsiasi relazione risulterebbe poco interessante. Inoltre anche regole di associazione con supporto e confidenza elevata potrebbero risultare poco interessanti, se banali.

Interesse di una regola

L'interesse di una regola è definito come:

$$Int(I \to j) = Conf(I \to j) - \frac{Supp(j)}{N}$$

dove N è il numero di basket.

Se I non ha interesse su j, allora la percentuale di basket che contengono I e j insieme è circa uguale alla percentuale di basket che contengono j.

L’interesse può essere positivo o negativo: se negativo implica che se compro I, difficilmente compro j

Lift di una regola

Una misura alternativa all’influenza è il lift, definito come segue:

$$Lift(I \to j) = N \cdot \frac{Supp(I \cup \{j\})}{Supp(i) \cdot Supp(j)} = \\ = \frac{\#\text{supporto osservato dalla regola}}{\#\text{supporto atteso}}$$

dove N è ancora il numero di basket.

Il lift corrisponde al rapporto sopra nell’ipotesi in cui non ci sia alcuna dipendenza tra I e j, ovvero se il supporto di I non dipenda dal supporto di j.

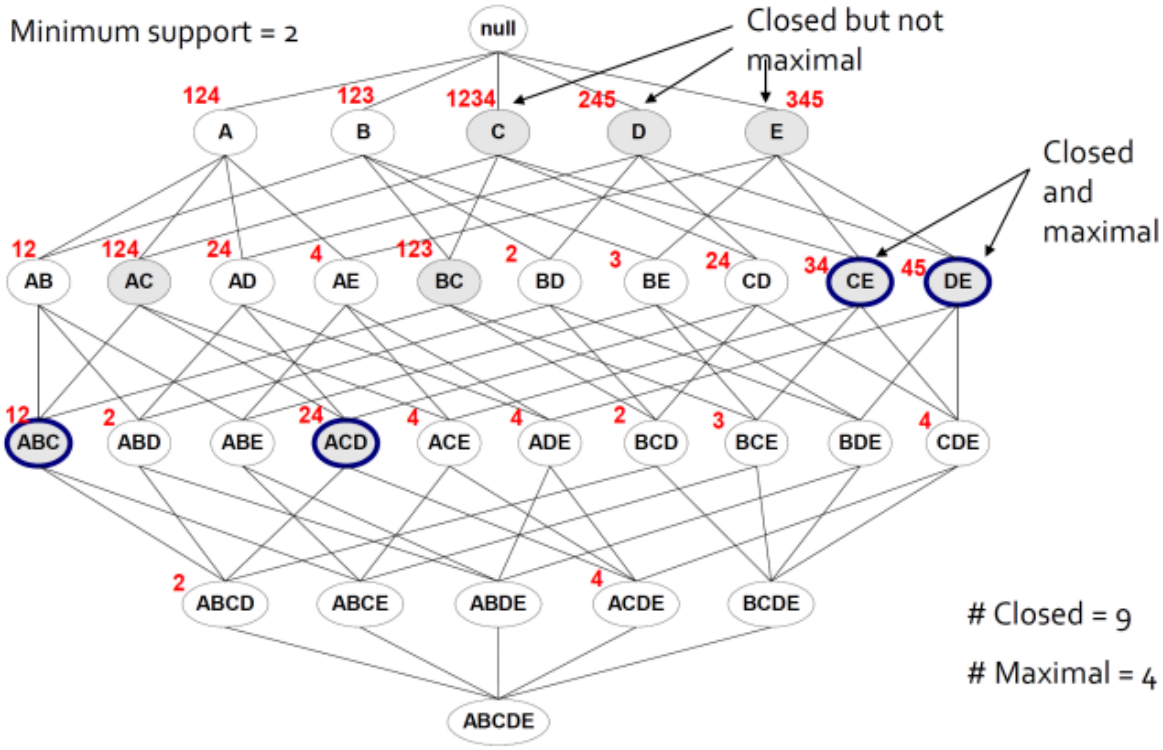
Se:

- lift>1, allora I ha un’influenza positiva su j: più è alto, maggiore è l’influenza
- lift<1, allora ha un’influenza negativa su j: più è basso, minore è l’influenza

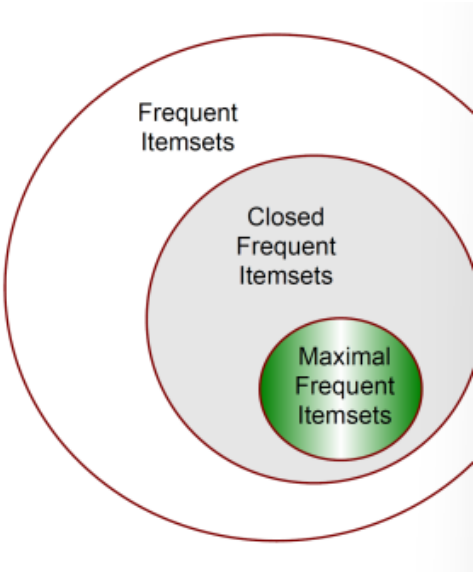
Insiemi frequenti massimali e chiusi

Varianti del problema consistono nella ricerca di:

1. Insiemi frequenti chiusi: se non esiste un super-insieme che ha lo stesso supporto
2. Insiemi frequenti massimali: se non esiste un super-insieme frequente
3. Insiemi frequenti massimali chiusi: se non esiste un super-insieme che abbia lo stesso supporto che è frequente.



In particolare, gli insiemi frequenti chiusi sono anche insiemi frequenti. Gli insiemi frequenti massimali sono anche insiemi frequenti chiusi. Qualsiasi algoritmo in grado di trovare insiemi frequenti massimali e chiusi.



Apriori

Approccio Naive

Il metodo naive per la ricerca degli itemset frequenti consiste nel generare un itemset, calcolare per ciascuno di essi il supporto e verificare se supera la soglia. Se ho tanti candidati è intrattabile: posso fare di meglio.

Anti-monotonicità del supporto

💡 Dato un itemset I, per ogni itemset $S \subseteq I,\ Supp(I) \leq Supp(S)$

Ovvero: il supporto di un itemset non supera mai quello dei suoi sottoinsiemi.

$$B_1 = \{m, c, b\}$$
$$B_2 = \{m, p, j\}$$
$$B_3 = \{m, b\}$$
$$B_4 = \{c, j\}$$
$$B_5 = \{m, p, b\}$$
$$B_6 = \{m, c, b, j\}$$
$$B_7 = \{c, b, j\}$$
$$B_8 = \{b, c\}$$

Itemset	Supporto
{m,c,b}	2
{m,c}	2
{m,b}	4
{c,b}	3
{m}	5
{c}	5
{b}	6

Principio apriori

Ottengo quindi il principio a Apriori:

> Se un itemset I è frequente, allora ogni sottoinsieme I è frequente.

O, equivalentemente:

> Se un itemset I non è frequente, allora nessun itemset che contiene I è frequente

Il viceversa non vale in generale: un itemset potrebbe non essere frequente anche se i suoi sottoinsiemi lo sono.

Il principio Apriori suggerisce un approccio **bottom-up**:

- gli itemset vengono ricercati per cardinalità crescente: partendo dai singoli item, poi le coppie, le triple e così via
- se un itemset di cardinalità k non è frequente, allora non occorre estenderlo e costruire un item k+1 che certamente non sarà frequente

Questo permette di fare un pruning dello spazio di ricerca ed esaminare molti candidati, anche in considerazione del fatto che itemset frequenti di cardinalità ≥ 4 sono rari.

Algoritmo Apriori

1. Costruisco l'insieme L_1 degli item frequenti
2. Per $k = 1, 2, ..$
   a. genero l'insieme C_{k+1} degli itemset candidati di cardinalità $k + 1$ a partire da L_k
   b. rimuovo da C_{k+1} gli itemset che contenfono sottoinsiemi con k item che non sono frequenti
   c. Calcolo il supporto di ogni itemset di C_{k+1}
   d. Costruisco l'insieme L_{k+1} formato da tutti gli itemset frequenti di C_{k+1}
3. Continuo fino a quando non ho più candidati da esaminare

Come genero i candidati?

L'insieme L_k è rappresentato da una tabella con k righe (1 per item).

L'insieme dei candidati C_{k+1} si può ottenere mediante join di L_k con se stesso.

Nel combinare due righe A e B devo imporre che:

- i primi k-1 item di A e B siano uguali
- il k-esimo item di A sia minore del k-esimo item di B: così assicuro che non si siano ridondanze

Itemset	Supporto
{b,c}	4
{b,j}	2
{b,m}	4
{c,j}	3
{c,m}	2
{j,m}	2
{m,p}	2

⋈

Itemset	Supporto
{b,c}	4
{b,j}	2
{b,m}	4
{c,j}	3
{c,m}	2
{j,m}	2
{m,p}	2

=

Itemset
{b,c,j}
{b,c,m}
{b,j,m}
{c,j,m}

Ottimizzazione dell'apriori

Raggruppamento in bucket (PCY)

In una prima passata sui dati, conto il supporto degli item e li raggruppo le coppie in bucket usando una hash. Se il support del bucket è sotto soglia, allora nessuna coppia di item del bucket sarà frequente. L'insieme delle coppie candidate, formate dalle coppie (i,j) sono tali che:

- i e j sono frequenti
- la coppia (i,j) cade in un bucket frequente

Partizionamento del DB (SON)

Divido il dataset in chunk (partizioni). Dato s il supporto minimo e p la percentuale di basket in ogni chunk:

1. su ogni chunck eseguo l'Apriori, scalando il supporto minimo a $p \cdot s$
2. considero l'unione di tutti gli itemset che sono frequenti in 1 o più chunk
3. Per ciascuno di questi calcolo il supporto nel dataset iniziale

Campionamento dal DB e frontiera negativa (TOIVONEN)

Dato un campione S del dataset D, la frontiera negatica è l'insieme i itemset che NON sono frequenti in S, ma i cui immediati sottoinsiemi (ottenuti togliendo un singolo item) sono frequenti.

Ad esempio, se l'itemset {A,B,C} non è frequente in S , ma {A,B}, {A,C} e {B,C} lo sono, allora {A,B,C} appartiene alla frontiera negativa.
Come vedremo, il parametro σ determina il supporto minimo del campione S e determina la probabilità di successo dell'algoritmo.

Più basso è σ, maggiore è la memoria richiesta ma minore è il numero di passi richiesti per arrivare alla soluzione corretta.

1. Selezioniamo un campione random S dal dataset D, formato da p basket. Dato il supporto minimo in D, fissa la soglia di supporto minimo di S a σ x p x sσ è un parametro tra 0 e 1.
2. Trova gli itemset frequenti in S e quelli che stanno nella frontiera negativa
3. Calcola il supporto degli itemset trovati al passo precedente:
    a. Se nessun itemset della frontiera negativa è frequente in D, allora restituisci come risultato gli itemset che sono risultati frequenti in S;
    b. Se uno o più itemset della frontiera negativa risultano frequenti in D, allora ripeti l'algritmo su un nuovo campione. Questo perché potrebbero esserci falsi negativi, ovvero super-insiemi che non sono frequenti nel campione ma nel dataset si.

Algoritmo FP-GROWTH

Svantaggi dell'apriori

L'Apriori richiede la generazione dell'insieme dei candidati con k items ad ogni step dell'algoritmo e la verifica tramite scansioni ripetute nel DB delle transazioni. Nei casi in cui esistono parecchi itemset frequenti di cardinalità elevata o la soglia di supporto è molto bassa, ciò richiede un costo computazionale e un utilizzo di memoria elevati.
E' possibile trovare gli itemset frequenti in maniera veloce e senza generare i candidati?

FP-growth

FP-Growth è un algoritmo che permette di individuare itemset frequenti tramite la "crescita" (estensione) di itemset più piccoli frequenti.
Utilizza un prefix-tree chiamato **Frequent Pattern tree (FP-tree)**.
L'FP-tree serve a:

- Fornire una rappresentazione compatta del DB di transazioni che permetta di evitare ripetute
  scansioni del DB per la ricerca degli itemset frequenti;
- Decomporre il problema in un insieme di problemi più piccoli (approccio divide et impera).

Perchè usare questo approccio?

- Singola scansione del DB per memorizzare item frequenti e rispettivo supporto in una struttura compatta.
- Se due o più transazioni condividono parzialmente o interamente un itemset che risulta frequente (ad es. (a,c,l) e (a,c,m,n)), nella struttura dati (tipo prefix tree) viene memorizzata una sola volta la porzione condivisa avendo cura di registrare correttamente il supporto complessivo degli item coinvolti.

Fasi dell'algoritmo

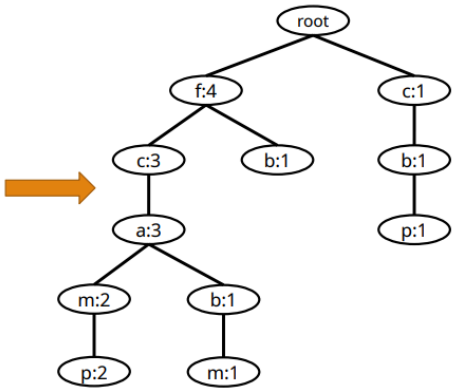
Ho due fasi principali:

1. Costruzione dell'FP-tree
2. Ricerca degli itemset frequenti

Costruzione dell'FP-tree

1. Scansiona il DB per calcolare la lista di item frequenti (con il loro supporto) ordinata per supporto decrescente.
2. Viene creata la radice dell'FP-Tree
3. Per ogni transazione presente nel DB:
   a. Gli item della transazione vengono ordinati per supporto decrescente
   b. Partendo dalla radice, aggiungo all'FP-Tree il cammino. Se un item non è presente, aggiungo un nuovo nodo con contatore settato a 1. Se è presente incremento tale contatore di 1.

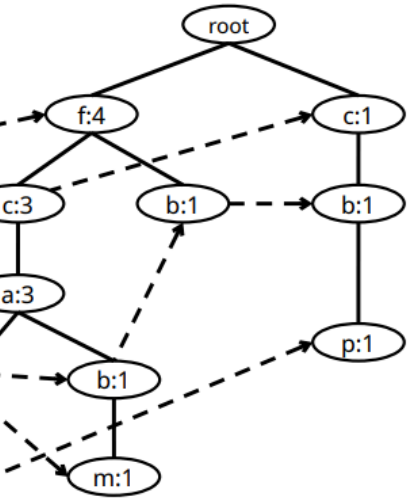
TID	Item frequenti ordinati
100	f, c, a, m, p
200	f, c, a, b, m
300	f, b
400	c, b, p
500	f, c, a, m, p



4. Collega l'FP-tree ad una tabella hash chiamata header table, in cui:
   a. Le chiavi della tabella sono gli item frequenti;
   b. Il valore corrispondente alla chiave X è un puntatore al primo nodo aggiunto nell'FPtree e contenente come etichetta l'item X.
5. Collega ogni nodo dell'FP-tree etichettato con l'item X al successivo nodo etichettato con X (se esiste) che è stato aggiunto all'FP-tree durante la sua costruzione.
   I puntatori aggiunti vengono anche chiamati node link.

Header table

Item	Head of node link
f	
c	
a	
b	
m	
p	



Si può dimostrare che l'FP-tree è una struttura dati:

- Completa: l'FP-tree rappresenta l'intero DB di transazioni e tutte le informazioni rilevanti per il mining degli insiemi frequenti;

- Compatta: Il numero di nodi dell'FP-tree è al più pari al numero di occorrenze degli item frequenti nel DB e la sua altezza è al più pari al numero di item frequenti. Nella pratica l'FP-tree è ancora più compatto, dato l'elevato numero di item condivisi tra varie transazioni.

Ricerca degli itemset frequenti:

La ricerca degli itemset frequenti viene effettuata usando esclusivamente l'FP-tree e i puntatori node link.
Dato un item X, tutti gli itemset frequenti che contengono X si possono ottenere:

- Partendo dai nodi dell'FP-tree puntati dai node link di X;
- Seguendo i cammini che, partendo da tali nodi, arrivano fino alla radice.

I contatori associati ad ogni nodo incontrato permettono di risalire al supporto degli itemset analizzati durante l'attraversamento dell'FP-tree.

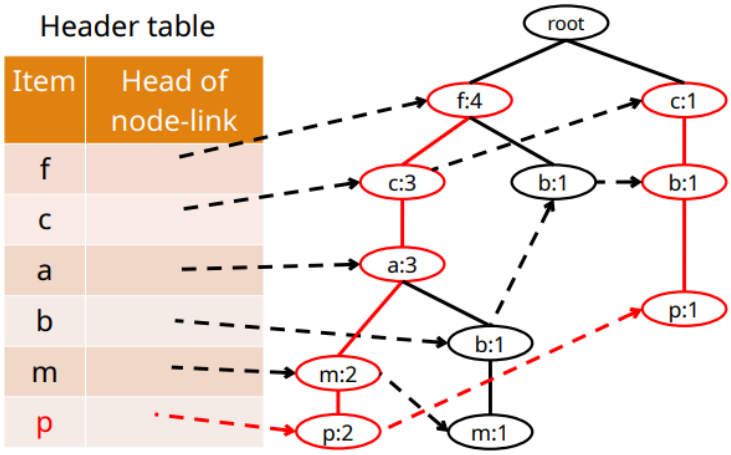
Per ogni item X nell'header table:

1. Sia P l'insieme dei cammini dell'FP-tree che partono dai nodi puntati dai node link di X
   e raggiungono la radice;
2. Per ogni cammino p P: ∈
   a) Considera tutte le possibili combinazioni di item in p e associa ad ogni combinazione C un supporto pari al
   contatore minimo tra quelli associati agli item di C;
   b) Aggiungi all'insieme delle soluzioni solo le combinazioni con supporto maggiore della soglia.

ESEMPIO (SOGLIA 3)

Header table

Item	Head of node-link
f	
c	
a	
b	
m	
p	



Consideriamo il nodo **p**

Seguendo i node link di p, **troviamo due cammini** che arrivano fino alla radice:

<f:4, c:3, a:3, m:2, p:2>
<c:1, b:1, p:1>

Il primo cammino, implica che l'itemset **{f,c,a,m} → p** occorre **2** nel DB

Il secondo, invece che **{c,b}→p** occorre **una** sola volta nel DB

Entrambi gli **itemset** sono **formati** da **item frequenti** (per come costruito l'FP-Tree.

Poichè la **soglia** di supporto è **3**, i due itemset sono **non frequenti**

RICERCA DI ALTRI ITEMSET FREQUENTI

Nell'esempio, il cammino trovato **<f:4, c:3, a:3, m:2, p:2>** ci dice non solo che l'itemset **{f,c,a,m,p}** ha **supporto 2**, ma che nel DB **esistono** anche i seguenti **itemset più piccoli**, alcuni dei quali risultano invece frequenti.

Per trovarli è sufficiente calcolare tutte le possibili **combinazioni** di **item del cammino**.

Il **supporto** di una combinazione è dato dal **contatore più piccolo** tra quelli associati agli item della combinazione.

TID	Support
f, c, a, m	2
f, c, a	3
f, c, m	2
f, a, m	2
c, a, m	2
f, c	3
f, a	3
f, m	2
c, a	3
c, m	2
a, m	2
f	4
c	3
a	3
m	2

In sintesi — il "succo" vero:

- Dal percorso completo puoi generare tutti i sottoinsiemi.
- Ogni sottoinsieme ha come supporto **il valore minimo** dei contatori dei suoi item.
- Tra questi sottoinsiemi, quelli con supporto ≥ *min_support* sono dichiarati **frequenti**.

- In pratica, se il sottoinsieme compare x volte, quello sopra compare almeno x volte

Principali vantaggi rispetto ad apriori

- Nessuna generazione di candidati
- Il DB delle transazioni viene scansionato solo due volte (una per la ricerca degli
  item frequenti e una per la costruzione dell'FP-tree)
- L'FP-tree è una struttura dati compatta che è costruita soltanto a partire dagli itemfrequenti e sfrutta eventuali ridondanze presenti nel DB (ad es. transazioni molto simili)
- Scala molto meglio rispetto ad Apriori rispetto sia al numero di transazioni che rispetto al supporto