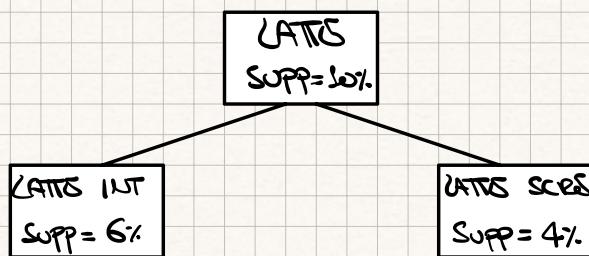


MULTI-LEVEL ASSOSATION RULES

Spesso gli item possono formare delle gerarchie, per esempio il latte può essere "intero" o "sciato".



Dove il supporto degli elementi ai livelli inferiori è minore.

Le regole associative utili per estrarre dei frequent pattern per item multi-livello sono dette **multi level association rule**.

Per estrarre queste regole abbiamo bisogno di item gerarchici, come introdotto prima.

Parametro di **min-sup flessibile**. Questo perché qualche item ha più valori ma è meno frequente. Per esempio si può adattare il min-sup al tipo di item o a un gruppo di item:

$$\begin{cases} \{\text{Diamante, telefono, etc}\} : 0,5\% \\ \{\text{Latte, caffè}\} : 5\% \end{cases}$$

Ridondanze fra regole:

Quelche regola può essere ridondante rispetto al suo antenato.

LATTE → PAN δ

(SUPP = 8%, CONF = 70%)

LATTE SCREMATO → PAN δ

(SUPP = 2%, CONF = 72%)

Una regola è ridondante se ha supporto e confidenza simile a un valore previsto, basato sull'anteneto.

MULTI-DIMENSIONAL ASSOCIATION RULES

Sono regole che impongono più di un predicato:

Esempio

Regola singola : LATTE \rightarrow CAFFÈ

Regola multi-d : LATTE \wedge STUDENTE \rightarrow CAFFÈ

QUANTITATIVE ASSOCIATIONS RULES

Sappiamo bene che possiamo avere a che fare con attributi categorici e attributi quantitativi, ovvero numerici con un ordine implicito.

Per categorizzare questi dati bisogna usare tecniche di discretizzazione:

1) DISCRETIZZAZIONE STATICA:

basata su un presupposto di gerarchia.

2) DISCRETIZZAZIONE DINAMICA:

basata sulla distribuzione dei dati

3) CLUSTERING

con associazioni distance-based.

NEGATIVES and RARE PATTERN

RARI

Sono itemset che hanno un basso supporto ma sono interessanti da analizzare. Per esempio si vuole analizzare chi compra una Ferrari.

NEGATIVES

Analizzare le correlazioni negative fra item, per esempio la correlazione fra chi compra una Ferrari e una Fiat insieme, sarà sicuramente negativa.

Definizione di correlazione negativa:

Se due itemset X e Y sono entrambi frequenti ma occorrono poco insieme, ovvero:

$$\text{supp}(X \cup Y) < \text{supp}(X) \cdot \text{supp}(Y)$$

allora X e Y sono negativamente correlate.

Definizione di strongly negative correlations:

Quando gli itemset X e Y :

nessuno dei
due presenti singolarmente

$$\text{supp}(\underline{X \cup Y}) \times \text{supp}(\bar{X} \cup \bar{Y}) \gg \text{supp}(\underline{X \cup Y}) \times \text{supp}(\bar{X} \cup \bar{Y})$$

\hookrightarrow X presente ma non Y

\hookrightarrow Transaction dove è presente solo X unito a quelle con solo Y

Problema delle transazioni nulle. Ovvio che se aumenta il numero di transazioni su cui definire una correlazione negativa, posso avere risultati diversi aggiungendo al DB transazioni senza i due itemset sotto analisi. Anche la definizione di Strongly negative correlation ne soffre.

Definizione di KULCZYNSKI MEASURE

Se due itemset X e Y sono frequenti ha:

$$\frac{P(X|Y) + P(Y|X)}{2} < \varepsilon$$

Allora X e Y sono negativamente correlati.

Dove ε è un parametro di threshold.

CONSTRAINT-BASED

Qui l'utente può impostare dei vincoli sugli itemset a cui fare l'analisi.

Tipi di vincoli

- 1) DATA CONSTRAINT : Quali dati analizzare
- 2) DIMENSION CONSTRAINT : Quale categoria analizzare
- 3) RULE CONSTRAINT :
- 4) INTERESTING RULE CONSTRAINT:

META-RULES

Sono le ipotesi che l'utente vuole confermare e analizzare:

$$P_1(x, y) \wedge P_2(x, w) \Rightarrow \text{buy}(x, \text{iPad})$$

Dove si è interessati a capire le caratteristiche di chi compra un iPad.

Generalizzando:

$$P_1 \wedge P_2 \wedge \dots \wedge P_n \Rightarrow Q_1 \wedge Q_2 \wedge \dots \wedge Q_M$$

Pruning pattern search space or data space

Quando profondamente riusciamo a tagliare lo spazio dove ricercare le risposte alle nostre domande, garantendo poi la completezza delle risposte trovate?

- Search space

Prima valutare un pattern candidato e poi decidere se possiamo eliminarlo.

- Data space

Controllare il DB così per vedere se il pezzo sotto analisi contribuisce nelle generare subsequent frequent pattern.

PATTERN SPACES PRUNING CONSTRAINTS

- 1) Anti-monotonic: se un vincolo C viene violato, allora terminare le operazioni future
- 2) Monotonic: Se C è soddisfatto, non bisogna più controllarlo
- 3) Succint: C deve essere soddisfatto
- 4) Convertible: C non è né monotonic né anti-monotonic ma può essere convertito se gli item delle transazione sono propriamente ordinati

DATA SPACES PRUNING CONSTRAINT

- 1) Data Succint: potere D all'inizio del processo.
- 2) Data anti-monotonic: potere le transazione che non soddisfano C .

ANTI-MONOTONE

Un vincolo è anti-monotone se un super-pattern lo soddisfa. Di conseguenza tutti i sub-pattern del super-pattern in questione soddisfano C.

Esempio:

La somma dei prezzi di un set di prodotti deve essere sotto una certa soglia. Se un itemset viola queste condizioni allora sicuramente la somma dei prezzi di un super set è maggiore della threshold. Quindi questo vincolo è anti-monotone.

Il conto del supporto è un'operazione anti-monotone.

MONOTONE

Un vincolo è monotone se il pattern soddisfa C. Non c'è bisogno di controllare C per i super set di un set che rispetta C.

Esempio

$$\sum(\text{Prezzo})_i \geq v$$

Se AB soddisfa il vincolo, necessariamente lo soddisfa anche ABC ... Z.

SUCCINCTNESS (concisione)

È un vincolo che permette di generare direttamente tutti gli itemset che lo soddisfano, anche senza conoscere il supporto.

Esempio

prezzo minimo per oggetto = X

Ci basta scansionare il DB per generare direttamente gli insiemi che soddisfano questa condizione.

La succintness viene definita come:

Dato A_S itemset che soddisfa C , succinto

Allora un itemset I che soddisfa C è basato sugli elementi di A_S .

CONVERTABLE CONSTRAINTS

Trasformare un vincolo in uno anti-monotone o monotone, semplicemente ordinando appropriatamente i dati.

Esempio

Prendiamo C : $\text{AVG}(\text{Profitti}) \geq 25$.

Se ordiniamo gli items:

$\langle a, f, g, d, b, h, c, e \rangle$

$\langle 40, 30, 20, 10, 0, -10, -20, -30 \rangle$

L'itemset $a \cup b \Rightarrow \text{AVG}(ab) = 23,3$ non soddisfa C . Anche $ab \cup h$ o $ab \cup h^*$ non lo fanno. C diventa anti-monotone.

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

Se invece ordiniamo gli items in ordine crescente
C diventa monotone.

$\langle e, c, h, b, d, g, f, a \rangle$

Ora ho appena trovato un itemset che soddisfa C, i suoi super-set sulla base dell'ordinamento soddisfano
anche C.

Un vincolo che può diventare sia monotone che
anti-monotone è detto **STRONGLY CONSISTENT**.

COLOSSAL PATTERN

Con gli algoritmi visti fin ora non possiamo trovare frequent pattern di grosse dimensioni (50 o 100 o più item) questo per via delle **"downward closure"** dei frequent pattern, ovvero la proprietà per cui se un pattern è frequente allora lo sono anche i suoi sottopattern.

Esempio:

Se abbiamo $(a_1 \dots a_{100})$ itemset frequente, allora anche $(a_1, a_{100}), (a_2, a_{100}), (a_1, a_2, a_3) \dots$ fino ad arrivare ad avere 2^{100} frequent pattern diversi. **Troppi da esaminare.**

C'è da dire che spesso i colossal pattern sono **nessuno** da trovare, e rappresentano una minima parte sul totale dei frequent pattern.

Ma spesso più importanti dei piccoli pattern.

NESSUNA SPERANZA DI COMPLETÀZZA

Impossibile cercare colossal pattern usando la filosofia vista fin ora di set completo, ovvero di cercare un intero colossal pattern.

L'idea è quella di aggiungere più item a ogni passo dell'algoritmo. Se pensiamo a come Apriori lavora, a ogni iterazione aggiunge un solo item ai frequent itemset, non supportabile per cercare Colossal Pattern. D'onde a ogni iterazione aggiungiamo N item? Questa è la strategia.

$I_4 \xrightarrow{\text{Apriori}} I_5$ VS $I_4 \longrightarrow I_8$

In altre, invece che trovare tutti i Colossal Pattern abbiamo bisogno di un modo più veloce, che però trovi solo alcuni Colossal Pattern, non tutti.

STRATEGIA "PATTERN FUSION"

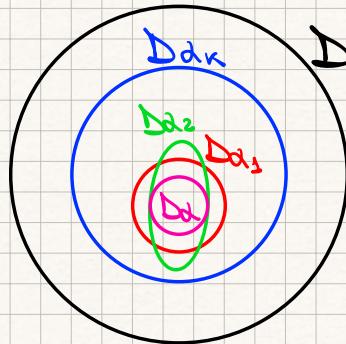
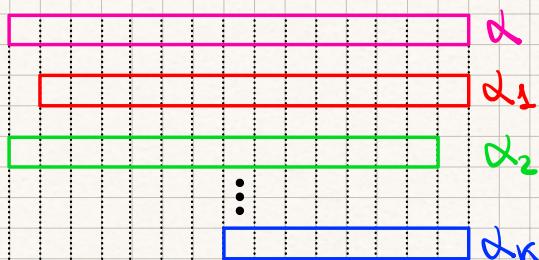
Partendo dal FP-Tree, andiamo alla ricerca dei pattern, ma si procede con passi più grandi di un passo, questo viene fatto combinando i pezzi trovati in un certo livello dell'albero per così saltare verso itemset più grandi. Questo procedimento viene ripetuto.

Non è garantito che tutti i Colossal Pattern vengano fuori.

Osservazione.

Presso un colossal pattern α , il suo support è tipicamente molto basso. Se noi consideriamo ora i sotto-pattern $\alpha_1 = \alpha_2$ entrambi con dimensioni simili ad α , il loro support è molto vicino a quello di α .

Questi sotto-pattern sono detti core patterns of α .



T-CORE PATTERN

Presso il colossal pattern α , un sotto-pattern β è un t-core pattern se ha un support simile a α :

$$\frac{|D_\alpha|}{|D_\beta|} > \tau \quad \text{con } 0 < \tau \leq 1$$

Dove $|D_d|$ è il numero di pattern che contengono d .

ROBUSTNESS of Colossal Pattern

Quando ha molti più core-pattern che piccoli pattern

Un pattern d è (d, τ) -robust se d è il massimo numero di items che possono essere rimossi da d per cui il risultato di questa operazione è un τ -core pattern di d .

I colossal pattern tendono ad avere molti core-pattern (2^d nel caso di d (d, τ) -robust).

DISTANZA FRA PATTERN

$$\text{Dist}(d, \beta) = 1 - \frac{|D_d \cap D_\beta|}{|D_d \cup D_\beta|}$$

Se sia d che β sono core-pattern dello stesso colossal pattern, allora la loro distanza ha un limite superiore pari a una sfera di raggio pari al core-ratio τ .

$$\text{Dist}(d, \beta) \leq 1 - \frac{1}{\frac{2}{\tau} - 1} = \gamma(\tau)$$

ALGORITMO

L'idea che sta dietro PATTERN-FUSION è:

- 1) Generare completamente tutti i frequent pattern di una grandezza piccola (diciamo grandezza g)
- 2) Prendiamo un pattern random β , esso avrà un'alta probabilità di essere un core-pattern di un colossal.

3) Possiamo quindi individuare tutti i pattern discendenti del colossal pattern α , partendo dagli itemset completi g-Itemset trovati inizialmente.

Da qui generiamo discendenti di α molto più vicini ad α per grandezza. Questi discendenti saranno core-pattern.

4) Da questi pattern più grandi ne selezioniamo K da cui inizierà l'iterazione successiva.

Inizializzazione

Usiamo un algoritmo conosciuto per trovare i g-Itemset.

Iterazioni

1) Sceglie K pattern random che faranno da seme per la creazione dei nuovi pattern

2) Per ogni seed pattern, troviamo tutti i pattern che sono confinati dentro la sfera centrata nel seed pattern corrente.

3) Fondere tutti i pattern trovati col seed pattern che li confina nella sfera.

Terminazione

Quando il pool di pattern corrente, contiene non più di K pattern all'inizio dell'iterazione.

Poche^{re} pattern-fusion est efficiente?

- 1) Evita di far esplodere la ricerca per pattern di medie dimensioni
- 2) Sa individuare "Short-cut" e fare salti.

MINING COMPRESSION

Vogliamo comprimere i vari pattern perché spesso i pattern sono composti da troppi item e spesso sono anche meno significativi

PATTERN DISTANCE

$$\text{Dist}(P_1, P_2) = 1 - \frac{|T(P_1) \cap T(P_2)|}{|T(P_1) \cup T(P_2)|}$$

Dove con $T(P_i)$ si intende tutte le transazioni in cui è presente il pattern P_i .

δ -CUSTOR

Per ogni pattern P , cerchiamo tutti i pattern S che possono essere espressi tramite P , e per loro distanze da P sia sotto la soglia δ ($\text{Disp}(P, S_i) \leq \delta$).

In questo modo definiamo un cluster di pattern S che sono vicini fino a δ rispetto a P .

Esempio

ID	Item-Sets	Support
P1	{38, 16, 18, 12}	205227
P2	{38, 16, 18, 12, 17}	205211
P3	{39, 38, 16, 18, 12, 17}	101758
P4	{39, 16, 18, 12, 17}	161563
P5	{39, 16, 18, 12}	161576

Il seguente dataset può essere compresso, se rappresentato dai soli P_2, P_3 e P_4 . Questo perché calcolando

$$\text{Dist}(P_1, P_2) = \frac{2}{3} \quad \left\{ \begin{array}{l} \text{bassa} \end{array} \right.$$

Si noti che P_2 contiene P_1 e il suo support è molto simile, perciò invece di dare come risultato P_1 e P_2 possiamo considerare solo P_2 . Stesso ragionamento possiamo applicarlo anche per P_4 e P_5 .

TOP-K PATTERNS

Mostrare i soli top-k risultati è una strategia per mostrare in output meno risultati.

Bisogna stare attenti, perché i vari pattern non sono mutualmente indipendenti, ma spesso possono essere posizionati in vari cluster in una regione ridotta (Δ come considerare solo $k=20$ città in tutto il mondo. Considerando le sole città più popolose, saranno tutte o quasi in Cina e India. L'intera Italia sarebbe vista come una o due città)

Per questo l'idea di considerare solo i Top-K non va bene.

Obiettivo: HIGH SIGNIFICANCE & LOW REDUNDANCY

Maximal Marginal Significance.

Serve per misurare la significatività di un pattern set. Perché vogliamo selezionare i migliori pattern per ogni cluster. Per far funzionare le cose dobbiamo introdurre il concetto di significance measure S , ovvero di una funzione che mappa i vari pattern con numeri reali in modo tale che $S(p)$ è il grado di utilità di un pattern.

Possiamo avere una divisione in termini di:

- 1) Misure oggettive: dipendenti dalla sola struttura del pattern e dai dati usati in fase di costruzione
- 2) Misure soggettive: basate sulla convinzioni degli utenti.

RIDONDANZA fra due pattern:

$$R(p, q) = S(p) + S(q) - s(p, q)$$

La ridondanza ideale è difficile da raggiungere, possiamo però approssimarla con la distanza fra pattern.