Regole associative

Gianluca Amato

Corso di Laurea Specialistica in Economia Informatica
Università “G. D'Annunzio” di Chieti-Pescara
anno accademico 2005-2006
ultimo aggiornamento: 11/04/06

Regole associative

Database transazionali (1)

- Si chiama database transazionale un archivio nel quale ogni istanza corrisponde ad una transazione
- Una transazione è composta da
  - un identificatore
  - un elenco di oggetti (item) che compongono la transazione
  - eventuali ulteriori attributi quali la data della transazione, l'acquirente, etc..
- È possibile associare al file delle transazioni ulteriori archivi che forniscono informazioni correlate, quali
  - il prezzo degli oggetti
  - informazioni dettagliate sull'acquirente, etc..

Database transazionali (2)

- Esempio di database transazionale

Transaction ID	Acquirente	Oggetti
2000	Paolo	A,B,C
1000	Michele	A,C
4000	Carlo	A,D
5000	Giulia	B,E,F

- I database transazionali possono essere visti come database relazionali, con la possibilità di attributi multi-valore.
  - In un vero database relazionale, invece, la tabella di cui sopra diventa:

Transaction ID	Acquirente	Oggetto
2000	Paolo	A
2000	Paolo	B
2000	Paolo	C
1000	Michele	A
1000	Michele	C
...	....	....

Itemset

- Un item è una formula atomica del tipo attr(x,*v*)
  - *x* indica la generica istanza del nostro insieme di dati
  - *attr* è un attributo che compone il nostro insieme di istanze
  - *v* è uno dei possibili valori (o un insieme di valori) assunti da *attr*
- Sia *i* una istanza che fa parte del nostro insieme di dati e sia dato l'item attr(x,*v*).
  - *i* soddisfa attr(x,*v*) quando, nella istanza *i*, l'attributo attr assume il valore v
  - si scrive attr(i,*v*) per indicare che i soddisfa attr(x,v)
- Un itemset è una congiunzione di item
  - $attr_1(x,v_1)$, $attr_2(x,v_2)$ è un itemset
  - Una istanza soddisfa un itemset se soddisfa tutti gli item che lo compongono

Cos'è una regola associativa?

- Una regola associativa collega tra di loro gli attributi di un insieme di dati.
- Ha la forma: Corpo → Testa [supporto,confidenza]
  - Corpo e Testa sono itemset
  - supporto e confidenza sono due valori percentuali
- Regola associativa su un database di studenti:
  - laurea(x,"economia") ∧ incorso(x,true)→ voto(x,100..110) [s: 20%, c: 75%]
- Regole associative con attributi multi-valore:
  - facoltà(x,"economia") ∧ sostenuto(x,"crittografia") ∧ sostenuto(x,"crittografia") → corso_di_laurea(x,"clei") [s: 1%, c: 99%]
  - compra(x,"computer") → compra(x, "monitor") [s: 0.5%, c: 70%]

Cosa sono supporto e confidenza? (1)

- Sia data la regola Corpo → Testa
  - Supporto: la percentuale delle istanze che soddisfano sia il corpo che la testa:

$$\frac{\#\{i \mid i \ \ soddisfa\ Corpo\ e\ Testa\}}{n.\ totale\ istanze}$$

  - Confidenza: tra le istanze che soddisfano il Corpo, la percentuale di quelle che soddisfano anche la Testa:

$$\frac{\#\{i \mid i \ \ soddisfa\ Corpo\ e\ Testa\}}{\#\{i \mid i \ \ soddisfa\ Corpo\}}$$

  - se si vedono Corpo e Testa in termini probabilistici, allora si usa scrivere
    - supporto: P(Corpo)
    - confidenza: P(Testa | Corpo) = P(Testa ∩ Corpo) / P(Corpo)

Cosa sono supporto e confidenza (2)

- Si fissano dei valori di soglia minima per supporto e confidenza
  - si chiamano regole associative forti quelle regole per le quali supporto e confidenza superano i valori di soglia.
  - si vogliano determinare tutte le regole associative forti.

Esempio: calcolo supporto e confidenza

- Problema: trovare tutte le regole del tipo
  - bought(x,_) → bought(x,_)
  - con supporto e confidenza minimi 50%
- Risultati
  - bought(x,A) → bought(x,C) [sup: 50%, conf: 66.6%]
  - bought(x,C) → bought(x,A) [sup: 50%, conf: 100%]

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Classificazione delle regole associative (1)

- Regole booleane e quantitative:
  - booleana se in tutti gli item *attr*(x,*v*), *v* è un singolo valore;
  - quantitativa se coinvolge attributi numerici e negli item *attr*(x,*v*), *v* è un insieme di valori, tipicamente un intervallo.
- Regole mono-dimensionali e multi-dimensionali: a seconda del numero di attributi diversi coinvolti.
  - Regola booleana mono-dimensionale:

    buys(x, "DB2"), buys(x, "DMBook") $\rightarrow$ buys(x, "DBMiner") [0.2%, 60%]
  - Regola quantitativa multi-dimensionale:

    age(x, "30..39"), income(x, "42..48K") $\rightarrow$ buys(x, "PC") [1%, 75%]
- Per le regole mono-dimensionali booleane si usa eliminare il nome del predicato, ottenendo forme del tipo:
  - DB2, DMBook $\rightarrow$ DBMiner

Classificazione delle regole associative (2)

- Si parla di analisi di associazioni a un singolo livello o a livelli multipli: a seconda che tutti gli item appartengano allo stesso livello di astrazione o no.
  - Una analisi multi-livello è in grado di trovare il seguente insieme di regole:
    - age(x,"30...39") => buys(x,"computer")
    - age(x,"30...39") => buys(x,"laptop computer")
- Le regole mono-dimensionali booleane sono le più comuni, e sono adoperate per la market-basket analysis (analisi del carrello della spesa).
- Ci sono varie possibili estensioni al concetto di regola di associazione. Ad esempio:
  - Analisi di correlazione
  - Analisi di maxpatterns e closed itemsets.

Regole associative booleane mono-dimensionali

Itemset frequenti

- Presentiamo uno degli algoritmi fondamentali per l'analisi di associazioni mono-dimensionali booleane: Apriori.
- Apriori si basa sul concetto di itemset frequente:
  - la frequenza di un itemset è il numero delle istanze che lo soddisfano.
  - un itemset frequente è un itemset la cui frequenza relativa (probabilità) supera la soglia di supporto minimo.
- Per trovare le regole associative forti:
  - Si determinano tutti gli itemset frequenti con supporto minimo *s*
  - Se X è un itemset frequente e $X=X_1 \cup X_2$, allora $X_1 \rightarrow X_2$ è una regola di associazione che supera la soglia minima di supporto.
  - Se la regola supera anche la soglia di confidenza minima, allora è una regola forte.

Esempio: itemset e regole associative

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Min. support 50%
Min. confidence 50%

Frequent Itemse	Support
{A}	75%
{B}	50%
{C}	50%
{A,C}	50%

Per la regola $A \Rightarrow C$:

support = P($\{A \cap C\}$) = 50%

confidence = P($\{A \cap C\}$)/P($\{A\}$) = 66.6%

Il principio Apriori

- Qualunque sottoinsieme di un itemset frequente è un itemset frequente.
- Si applica questo principio nel calcolo degli itemset frequenti:
  - Definizione: k-itemset è un itemset di k elementi.
  - In maniera iterativa, trovo tutti i k-itemset frequenti per k da 1 in poi.
    - Tutti i possibili (k+1)-itemset frequenti sono ottenuti dall'unione di due k-itemset frequenti.
    - Non è necessario controllare tutti i possibili sottoinsiemi di k+1 elementi per sapere se sono frequenti.

L'algoritmo Apriori: pseudo-codice

C_k: Candidate itemset of size k

L_k : frequent itemset of size k

L_1 = {frequent items}; // ottenuti da una scansione del database

for (k = 1; L_k !=∅; k++) **do begin**

C_{k+1} = candidati generati da L_k

for each transaction t in database do

incrementa il conteggio di tutti i candidati in C_{k+1} che sono contenuti in t

L_{k+1} = candidates in C_{k+1} with min_support

end

return $\cup_k L_k$;

Generazione dei candidati (1)

- Si basa su due passi fondamentali, eseguiti ripetutamente: Join e Prune.
- Passo Join: partendo da L_k, l'insieme dei k-itemset frequenti, genero C_{k+1}, l'insieme dei (k+1)-itemset candidati.
  - metto assieme le coppie di itemset che hanno k-1 elementi uguali
  - **ottimizzazione**: se esiste un ordinamento tra gli item, e gli itemset sono ordinati in maniera lessicografica, ci possiamo limitare a mettere insieme gli itemset che hanno i primi k elementi uguali
- Esempio:
  - se L_2={AB, AC, AF, BC, CF}, allora C_3={ABC, ABF, ACF}.
  - notare che, sfruttando l'ottimizzazione indicata sopra, non genero BCF

Generazione dei candidati (2)

- Per controllare se $X \in C_{k+1}$ è veramente frequente, posso fare una scansione della base di dati e contare in quante transazioni appare X.
  - L'operazione è costosa, si riduce prima il numero di candidati con il passo Prune.
- Passo Prune: elimino da C_{k+1} tutti quegli itemset X per cui esiste un sottoinsieme di k elementi di X che non è in L_k.
  - ha senso solo per $k > 1$
- Esempio:
  - se L_2={AB, AC, AF, BC, BF}, allora C_3={ABC, ABF, ACF} dopo il passo Join
  - il passo Prune elimina ACF perché CF non non è in L_2

Pseudo-codice per Join e Prune

- Supponiamo che gli item in L_k siano ordinati in maniera lessicografica.
- Join:

  insert into C_k

  select $p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}$

  from $L_{k-1}\ p, L_{k-1}\ q$

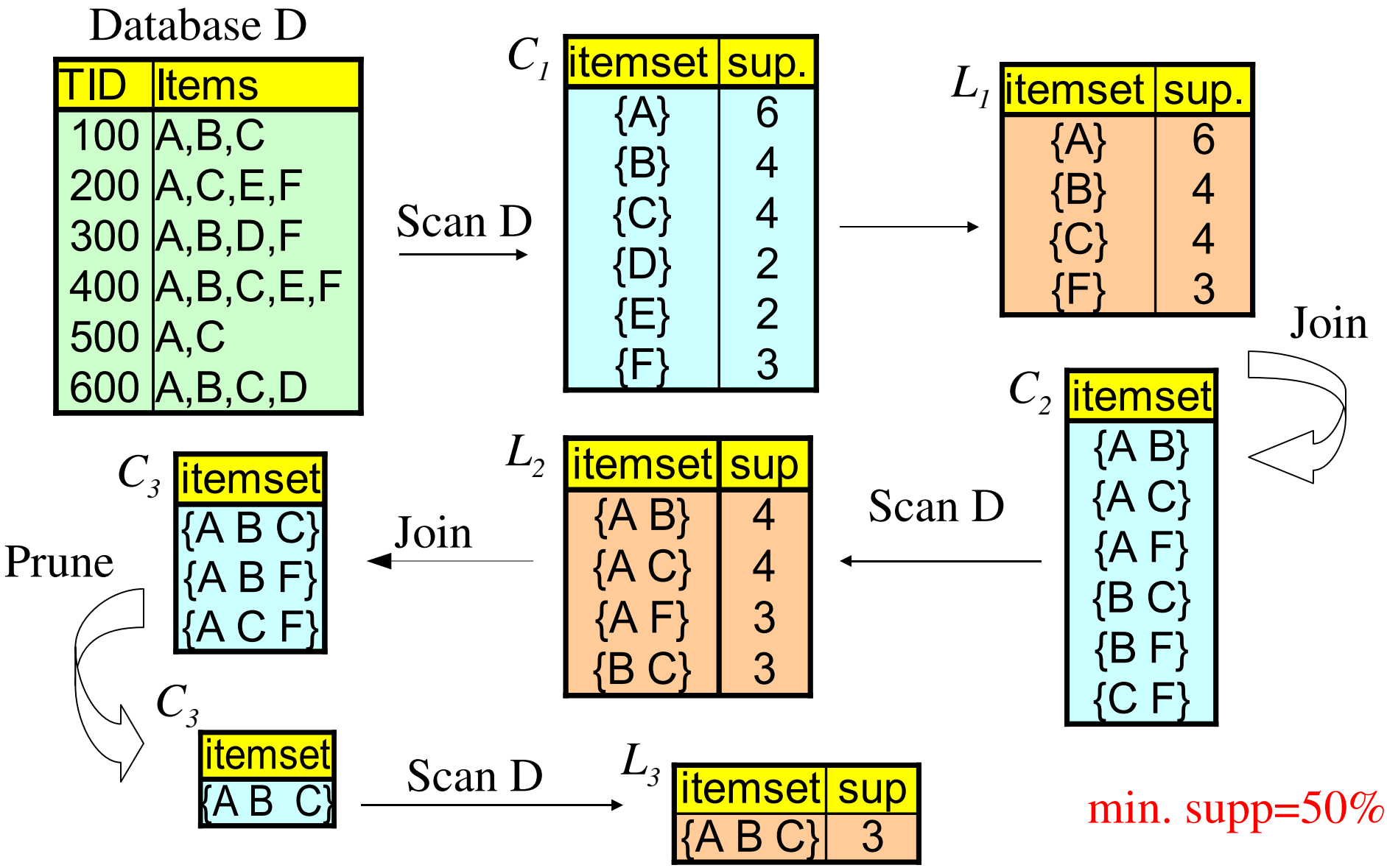
  where $p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$
- Prune:

  forall ***itemsets c in*** C_k do

  forall ***(k-1)-subsets s of c*** do

  if *(s is not in* L_{k-1}*)* **then delete** *c* **from** C_k

Esempio: algoritmo Apriori



Possibili ottimizzazioni

- Ne sono state studiate tante, vediamone due:
  - Transaction reduction: se una transazione non contiene nessun itemset in L_k non potrà contenere nessun itemset in L_{k+1} o successivi. Si può quindi eliminare e non considerare mai più.
  - Partizionamento: si divide il database in tante parti, ognuna delle quali può essere caricata in memoria. Calcolo gli itemset frequenti per le varie partizioni. Alla fine calcolo gli itemset frequenti globali, tenendo conto che un item frequente per tutto il database è frequente per almeno una delle sue partizioni.

Regole associative multi-livello

Regole associative a più livelli

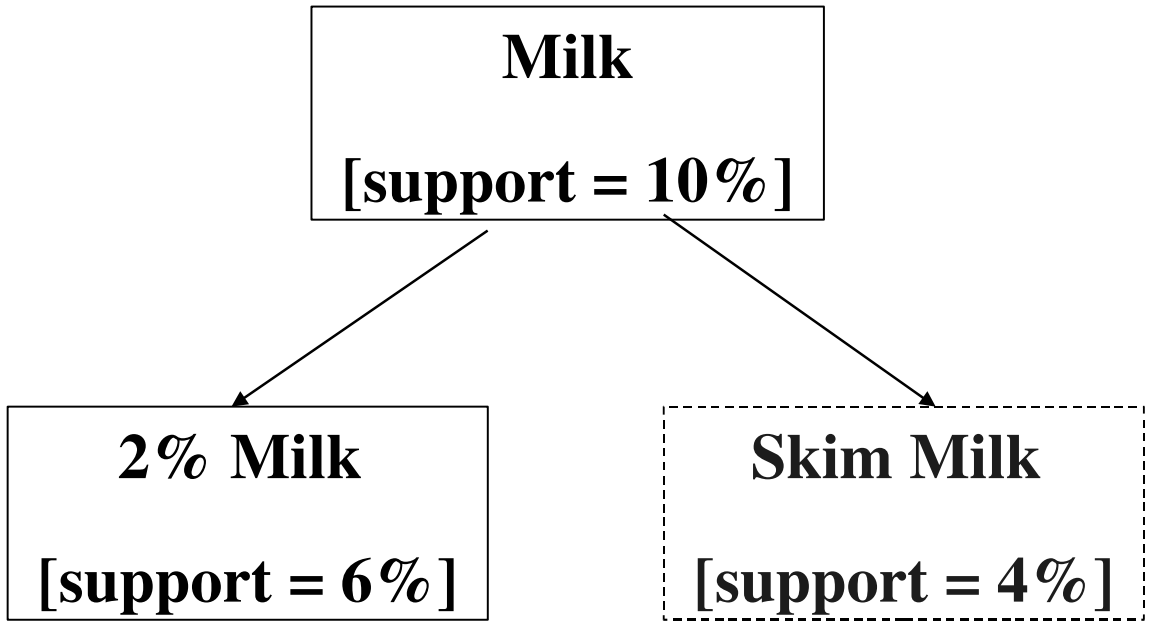
- Per molte applicazioni, è difficile trovare delle associazioni forti tra oggetti a un livello primitivo di dettaglio.
  - Occorre considerare regole associative a tutti i livelli della gerarchia di concetti.
- Approccio top-down:
  - Si calcolano gli itemset a partire dal livello di astrazione più generico;
    - latte → pane [20%,60%]
  - Si procede via via verso livello sempre più specifici.
    - latte scremato → pane integrale [6%, 50%]

Supporto uniformo e supporto ridotto

- Supporto uniforme: lo stesso livello di supporto minimo è usato a tutti i livelli di astrazione.
  - Efficiente! Se un itemset a un certo livello di astrazione non è frequente, non dobbiamo esaminarlo per nessuno dei livelli di dettaglio maggiori.
  - È difficile scegliere i livello di supporto appropriato:
    - Troppo basso: genera troppe regole associative
    - Troppo alto: si rischia di mancare del tutto regole associative ad alto livello di dettaglio.
- Supporto ridotto: valori di supporto minimo diversi per ogni livello.

Supporto uniforme

Level 1
min_sup = 5%

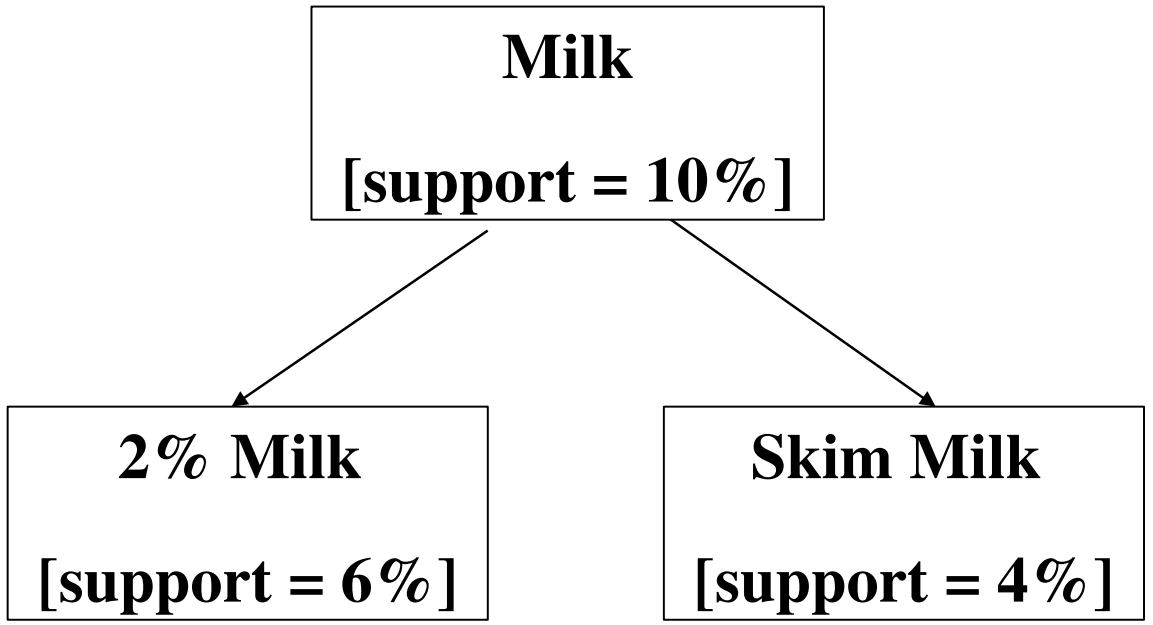


Level 2
min_sup = 5%

Supporto ridotto

Level 1
min_sup = 5%

Level 2
min_sup = 3%

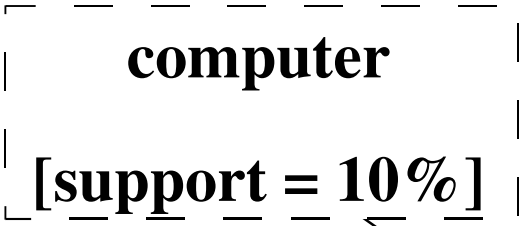


Strategie per il metodo di supporto ridotto

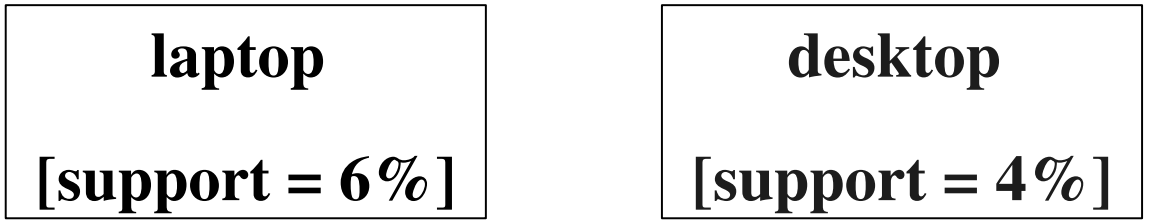
- Livelli indipendenti: gli itemset a differenti livelli di astrazione sono calcolati in maniera indipendente tra di loro.
  - Strategia completa: trova tutti gli itemset frequenti
- Filtraggio incrociato: un k-itemset a livello i viene considerato un candidato solo se al livello i-1 è frequente.
  - Strategia efficiente: ma può non trovare tutti gli itemset frequenti.

Filtraggio incrociato

Level 1
min_sup = 12%



Level 2
min_sup = 3%



non esaminati con il filtraggio incrociato

Cross-level association rules

- Finora abbiamo considerato regole i cui i vari item hanno tutti lo stesso livello di dettaglio.
- Cross-level association rules: quando i livelli di dettaglio dei vari item differiscono.
  - Computer $\rightarrow$ Stampante BN [sup: 4%, conf. 30%]

Associazioni multi-livello ridondanti

- Alcune regole possono essere ridondanti a causa di regole più astratte.
  - R1) Latte → Pane integrale [sup: 8%, conf: 70%]
  - R2) Latte scremato → Pane integrale [sup: 2%, conf: 72%]
- Diciamo che la regola R1 è antenato della R2: la R1 si ottiene da R2 rimpiazzando alcuni item con altri item a un livello di astrazione superiore.
- Supponiamo che ¼ delle vendite di latte sia da latte scremato
  - Supporto atteso "Latte scremato → Pane integrale" è ¼*8%=2%
  - La confidenza attesa è il 70%
  - Seconda regola ridondante!!

Regole associative multidimensionali

Regole associative multi-dimensionali

- Regole associative multi-dimensionali:
  - Regole inter-dimensionali (ogni predicato appare una sola volta):
    - age(x,"19--25") $\wedge$ occupation(x,"student") -> buys(x,"coke");
  - Regole ibride (predicati ripetuti)
    - age(x,"19--25") $\wedge$ buys(x,"pop corn") -> buys(x,"coke")
- Regole multi-dimensionali booleane
  - Si usa un algoritmo simile ad Apriori
- Regole multi-dimensionali quantitative
  - Gli attributi quantitativi vanno discretizzati. Ci sono tre approcci:
    - discretizzazione statica, eseguita prima di applicare l'algoritmo di ricerca delle regole associative
      - una volta discretizzat i datii, si applica un algoritmo standard come Apriori.
    - discretizzazione dinamica, eseguita in parallelo alla ricerca di regole associative

Discretizzazione statica e dinamica

- La discretizzazione dinamica produce regole “migliori”, tipicamente con un supporto e confidenza più elevati.
- Supponiamo di discretizzare age negli intervalli “1-18”, “19-36”, “37-48”,...
- Applicando Apriori possiamo ottenere regole del tipo

  age(X,”1-18”) ∧ buys(X,”computer”) -> buys(X,”joystick”)
  age(X,”19-36”) ∧ buys(X,”computer”) -> buys(X,”joystick”)
- Un algoritmo che usa una discretizzazione dinamica è in grado di ottenere:

  age(X,”12-28”) ∧ buys(X,”computer”) -> buys(X,”joystick”)

  con confidenza e supporto maggiori.
  - l'intervallo “12-28” è generato ad-hoc per migliorare le prestazioni della regola associativa.

Lift di una regola associativa

Regole interessanti (1)

- Supporto e confidenza sono buone misure oggettive per determinare se le regole sono interessanti?
- Abbiamo bisogno di una misura di "correlazione tra eventi"
- Esempio 1:
  - tra 5000 studenti
    - 3000 giocano a basket
    - 3750 mangiano cereal
    - 2000 giocano a basket e mangiano cereali
  - *play basketball* $\Rightarrow$ *eat cereal* [40%, 66.7%] è una regola forte, ma è fuorviante, in quanto la percentuale di studenti che mangiano cereali è del 75%
  - *play basketball* $\Rightarrow$ *not eat cereal* [20%, 33.3%] è molto più accurata, sebbene con valori di confidenza e supporto inferiori.

Regole interessanti (2)

- Esempio 2:
  - X e Y sono positiv. correlati
  - X e Z sono neg. correlati
  - X=>Z è più forte di X=>Y

X	1	1	1	1	0	0	0	0
Y	1	1	0	0	0	0	0	0
Z	0	1	1	1	1	1	1	1

Rule	Support	Confidence
X=>Y	25%	50%
X=>Z	37.50%	75%

Lift di una regola associativa (1)

- Dati due eventi A, B si definisce il coefficiente di correlazione tra i due come

$$corr_{A,B} = \frac{P(A \cup B)}{P(A)P(B)} = \frac{P(B|A)}{P(B)}$$

  - Se $corr_{A,B} = 1$ i due eventi sono indipendenti
  - Se $corr_{A,B} > 1$ i due eventi sono positivamente correlati
  - Se $corr_{A,B} < 1$ i due eventi sono negativamente correlati
- Se A → B è una regola associativa, il valore $corr_{A,B}$ è detto lift o (interesse)
  - è la confidenza diviso il supporto della conclusione
  - notare che il lift di A→ B e B→A è uguale

Lift di una regola associativa (2)

- Riprendiamo l'esempio 2
- Otteniamo i seguenti valori di interesse

X	1	1	1	1	0	0	0	0
Y	1	1	0	0	0	0	0	0
Z	0	1	1	1	1	1	1	1

Itemset	Support	Lift
X,Y	25%	2
X,Z	37.50%	0.9
Y,Z	12.50%	0.57

- Si vede chiaramente che, considerando il lift, la regola X→Y è più interessante di X → Z

Regole associative vincolate

Metaregole

- È possibile focalizzare la ricerca solo di determinate regole
  - Ad esempio, vogliamo capire il profilo dei clienti della AllElectronics che comprano software educativo.
- Alcuni sistemi consentono di specificare una metaregola del tipo
  - $P_1(x,Y) \wedge P_2(x,W) \Rightarrow$ buys(x,"software educativo")
  - P_1 e P_2 sono "variabili di predicato" e possono essere instanziate con un qualunque attributo
- Il sistema di analisi cerca solo regole che "matchano" la meta-regola:
  - age(x,"30..39") $\wedge$ income(x,"41K..60K") => buys(x,"software educativo")

Vincoli sul contenuto delle regole

- Altri tipi di vincoli possono essere imposti sul contenuto delle regole, piuttosto che sulla forma.
  - ad esempio, supponiamo di avere un database con due tabelle
    - (1) trans (TID, Itemset) (2) itemInfo (Item, Type, Price)
  - ci si può restringere a considerare quelle regole della forma $S_1 \Rightarrow S_2$ tali che:
    - $sum(S_1.price) < 100 \wedge min(S_1.price) > 20 \wedge count(S_1) > 3 \wedge sum(S_2.price) > 1000$
    - ho usato la notazione S.price per indicare i prezzi associati agli item in S ricavabili dalla tabella itemInfo.
- Una "constrained association query" (CAQ) è una stringa del tipo $\{(S_1, S_2) \mid C\}$ dove C è un insieme di vincoli che coinvolgono S_1 ed S_2 (lato sinistro e destro della regola associativa).

Classificazione dei vincoli

- I vincoli che si possono esprimere sono i seguenti
  - vincoli di dominio
    - $S \subseteq V$ oppure $S \supseteq V$
    - Esempio: $\{snacks, sodas\} \subseteq S$
  - vincoli di aggregazione
    - $agg(S)\ \theta\ v$ dove $agg \in \{min, max, sum, count, avg\}$, and $\theta \in \{=, \neq, <, \leq, >, \geq\}$
    - Esempio: $count(S.Type) = 1\ ,\ \ avg(S.Price) < 100$

Ottimizzazione delle CAQ

- Data una CAQ = { (S1, S2) | C } l'algoritmo è
  - corretto: se trova solo insiemi frequenti che soddisfano i vincoli
  - completo: se trova tutti gli insiemi frequenti che soddisfano i vincoli
- Una soluzione naïve:
  - applicare Apriori per trovare tutti gli insiemi frequenti e poi eliminare quelli che non soddisfano C
- Una soluzione migliore
  - analizzare i vincoli e tentare di anticipare il più possibile il loro utilizzo, in modo da ridurre la complessità della operazione.

Monotonia ed anti-monotonia

- Un vincolo C_a è anti-monotono se, quando S non soddisfa C_a, nessun sopra-insieme di S soddisfa C_a
  - il vincolo supporto(S) >= s è un vincolo anti-monotono
  - ad ogni passo di Apriori si può controllare C_a ed eliminare gli itemset che non lo soddisfano
- Un vincolo C_m è monotono se, quando S soddisfa C_m, tutti i sopra-insiemi di S soddisfano C_m
  - se un itemset S soddisfa C_m non è necessario controllarlo per gli itemset derivati da S

Vincoli convertibili (1)

- Si supponga che tutti gli item siano ordinati secondo un determinato ordinamento totale
- Un vincolo C è convertibile monotono sse quando un itemset S soddisfa C, ogni item-set ottenuto aggiungendo elementi in coda ad S soddisfa C
  - Esempio: avg(I.price) >= 100 è convertibile monotono, con l'ordinamento standard dei numeri.
  - Siccome {50,100,500} soddisfa avg(I.price) >=100, allora anche {50,100, 500, 1500} lo soddisfa

Vincoli convertibili (2)

- Un vincolo C è convertibile anti-monotono sse quando un itemset S non soddisfa C, ogni itemset ottenuto aggiungendo elementi in coda ad S non soddisfa C.
  - Esempio: avg(I.price) <= 100
- Se si considera l'ordinamento inverso (si ordinano gli elementi nell'itemset dal più grande al più piccolo) allora:
  - avg(I.price) >= 100 è convertibile anti-monotono
  - avg(I.price) <= 100 è convertibile monotono
- I vincoli $sum(S) \leq v$ e $sum(S) \geq v$ non ricadono in nessuna delle categorie viste prima: sono vincoli inconvertibili.
  - Notare che se vale che tutti gli elementi in S sono positivi, allora $sum(S) \leq v$ è antimonotono e $sum(S) \geq v$ è monotono.

Vincoli succinti (1)

- Informalmente: un vincolo C_s è succinto se è possibile trovare una procedura che generi direttamente tutti e soli gli itemset che soddisfano C_s (senza controllare effettivamente il vincolo)
  - gli itemset J tali che min(J.price)$\geq$ 500 sono generabile direttamente come sotto-insiemi degli item con prezzo superiore a 500
  - gli itemset J tali che min(J.price) $\leq$ 500 è generabile direttamente come unione di due itemset $S_1 \cup S_2$ dove $S_1 \neq \emptyset$ contiene solo item con prezzo inferiore di 500 ed S_2 contiene item con prezzo superiore a 500
  - invece, il vincolo sum(J.price) $\leq$ 2000 non è succinto

Vincoli succinti (2)

- Formalmente: dato un insieme di oggetti I
  - un sottoinsieme I_s di I è succinto se è esprimibile come $\sigma_p(I)$, ovvero come proiezione di I rispetto a un determinato predicato esprimibile nell'algebra relazionale.
    - {S | S.price <= 500} è succinto
    - {S | S.price è primo} non è succinto
  - $SP \subseteq 2^I$ è succinto se esistono $I_1,...,I_k \subseteq I$ succinti tali che ogni elemento di SP si può scrivere a partire da sottoinsiemi di $I_1,...,I_k$ usando le operazioni di unione e differenza di insiemi.
    - {SP | min(SP).price <=500} è succinto perchè ogni elemento di SP si scrive come $I_1 \cup I_2$ con $I_1 \subseteq$ {I | I.price <= 500} ed $I_2 \subseteq$ {I | I.price > 500}.

Vincoli succinti (3)

- Ancora formalmente:
  - Un vincolo C è succinto se l'insieme degli item che lo soddisfa è succinto
    - min(S.price) <= 500 è succinto per quanto vista prima
- Alcune proprietà dei vincoli succinti
  - se S_1 ed S_2 soddisfano C e C è succinto, anche $S_1 \cup S_2$ soddisfa C

Conclusioni

- Determinazione di regole associative
  - probabilmente il più grosso contribuito della comunità di ricercatori nelle basi di dati al KDD
  - un gran numero di pubblicazioni relative
- Direzioni di ricerca
  - determinazione di regole associative in tipi di dati complessi
    - dati spaziali
    - serie temporali

Bibliografia (1)

Algoritmo Apriori:

[1] R. Agrawal and R. Srikant. *Fast algorithms for mining association rules*. VLDB '94 487-499, Santiago, Chile.

Regole Associative Multilivello

[2] R. Srikant and R. Agrawal. *Mining generalized association rules*. VLDB'95, 407-419, Zurich, Switzerland, Sept. 1995.

[3] J. Han and Y. Fu. *Discovery of multiple-level association rules from large databases*. VLDB'95, 420-431, Zurich, Switzerland.

Regole Associative Quantitative

[4] B. Lent, A. Swami, and J. Widom. *Clustering association rules*. ICDE'97, 220-231, Birmingham, England.

[5] R. Srikant and R. Agrawal. *Mining quantitative association rules in large relational tables*. SIGMOD'96, 1-12, Montreal, Canada.

[6] R.J. Miller and Y. Yang. *Association rules over interval data*. SIGMOD'97, 452-461, Tucson, Arizona.

Bibliografia (2)

Regole associative vincolate

[7] M. Kamber, J. Han, and J. Y. Chiang. *Metarule-guided mining of multi-dimensional association rules using data cubes*. KDD'97, 207-210, Newport Beach, California.