Introducere în Data Science

Analiza asocierilor: concepte de bază

Lucian Sasu, Ph.D.

Universitatea Transilvania din Brașov, Facultatea de Matematică și Informatică

May 31, 2021

Outline

Noțiuni

- În cadrul vânzărilor din magazine se înregistrează conținutul coșurilor de cumpărături achiziționate = tranzacții
- Problemă: pentru un set de tranzacții să se determine regulile care dau predispoziția apariției unui obiect pe baza existenței altor obiecte într-un coș
- Exemplu de tranzacții:

TID	Obiecte
1	{pâine, lapte}
2	{pâine, scutece, bere, ouă}
3	{lapte, scutece, bere, suc}
4	{pâine, lapte, scutece, bere}
5	{pâine, lapte, scutece, suc}

Table 1: Tranzacții de tip coș de cumpărături

- TID = Transaction ID
- Exemplu de regulă ce se poate extrage: $\{scutece\} \longrightarrow \{bere\}$

Noțiuni

- Regula sugerează că ar exista o relație de dependență între vânzarea de scutece și cea de bere
- Remarcă: regula dată este direcțională; nu înseamnă neapărat și că $\{bere\} \longrightarrow \{scutece\}$
- Moduri de exploatare a unor astfel de reguli:
  - se scade prețul obiectelor din antecedentul regulii, se crește prețul celor din consecvent
  - se face cross-selling
  - se decide dispunerea pe raft a produselor
  - se face reclamă sau ofertă personalizată, gruparea produselor în cataloage de prezentare
- Alte medii de aplicare: bioinformatică, diagnostic medical, web mining, analiza științifică a datelor

Noțiuni

Probleme:

- descoperirea de pattern–uri în seturi mari de date este o problemă intensivă computațional
- o parte din relațiile descoperite pot fi pur și simplu rodul întâmplării
- evaluarea regulilor obținute este un pas absolut necesar
- semnul de implicație în acest context înseamnă apariție concomitentă, nu cauzalitate
- de citit: Correlation does not imply causation (Wikipedia)

Noțiuni, definiții

- Posibil mod de reprezentare a tranzacțiilor:

TID	Pâine	Lapte	Scutece	Bere	Ouă	suc
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

Table 2: Reprezentare binară a datelor tranzacției

- Reprezentarea binară este asimetrică: prezența unui produs este mai importantă decât lipsa lui
- Reprezentare voit simplistă, omițând cantitatea de achiziție

Noțiuni, definiții

- Mulțime de produse (simplu: mulțime; eng: itemset) — colecție de unul sau mai multe obiecte
- k–mulțime (eng: k-itemset) — o mulțime compusă din k obiecte (e.g. produse)
- Numărul de suport al unei mulțimi $\sigma(\cdot)$ (eng: support count) — frecvența de apariție a acelei mulțimi
- $\sigma(X) = |\{t_i : X \subseteq t_i, t_i \in T\}|$ unde T este mulțimea tuturor tranzacțiilor, $|U|$ este numărul de elemente ale (cardinalul) mulțimii U
- Exemplu: $\sigma(\{lapte, paine, scutece\}) = 2$

Noțiuni, definiții

- Regulă de asociere (regulă): o expresie de forma $X \longrightarrow Y$ unde X, Y sunt mulțimi de produse, $X \cap Y = \emptyset$

Definiție (Suportul unei reguli)

Suportul regulii $X \longrightarrow Y$ pentru o mulțime de N tranzacții este fracția de tranzacții care conțin produsele din $X \cup Y$:

$$s(X \longrightarrow Y) = \frac{\sigma(X \cup Y)}{N} \tag{1}$$

Definiție (Gradul de încredere al unei reguli)

Gradul de încredere al regulii $X \longrightarrow Y$ este numărul de tranzacții care conțin pe X și Y raportat la cele care conțin doar pe X:

$$c(X \longrightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)} \tag{2}$$

Noțiuni, definiții

Exemplu: pentru datele din tabelul de mai jos

TID	Obiecte
1	{pâine, lapte}
2	{pâine, scutece, bere, ouă}
3	{lapte, scutece, bere, suc}
4	{pâine, lapte, scutece, bere}
5	{pâine, lapte, scutece, suc}

- Considerăm regula $\{lapte, scutece\} \longrightarrow \{bere\}$
- Numărul de suport pentru $\{lapte, scutece, bere\}$ este 2; numărul total de tranzacții este 5, deci suportul este $2/5$
- Gradul de încredere: sunt 3 tranzacții care conțin $\{lapte, scutece\}$, deci gradul de încredere este $2/3$

Noțiuni, definiții

- De ce se folosește suportul?
  - o regulă cu suport mic poate însemna o legătură întâmplătoare
  - o regulă cu suport mic s–ar putea să fie neprofitabilă
  - suportul poate elimina regulile neinteresante
- De ce se folosește gradul de încredere?
  - măsoară încrederea în rezultatul aplicării unei reguli
  - pentru regula $X \longrightarrow Y$ un grad de încredere mare arată în ce măsură apariția mulțimii X va duce la apariția mulțimii Y
  - $c(X \longrightarrow Y)$ poate fi interpretată ca probabilitatea condiționată $P(Y|X)$

Noțiuni, definiții

Definiție (Descoperirea relațiilor de asociere)

Dându–se o mulțime de tranzacții T, să se găsească toate regulile cu suport $\geq$ minsup și grad de încredere conf $\geq$ minconf, unde minsup și minconf sunt praguri date.

Abordare brute-force:

- se generează toate regulile de asociere posibile
- se calculează suportul și gradul de încredere al fiecărei reguli
- se elimină regulile care nu respectă pragurile *minconf* și *minsup* date
- Complexitate de calcul prohibitivă: numărul de reguli pentru d produse este:

$$R = 3^d - 2^{d+1} + 1$$

  (temă pentru acasă)
- Pentru cele 6 produse din tranzacțiile date am avea 602 reguli; pentru *minsup* $= 20\%$ și *minconf* $= 50\%$, mai mult de 80% din regulile generate sunt eliminate!

Pași de lucru

- Observație importantă: suportul regulii $X \longrightarrow Y$ depinde doar de numărul suport al mulțimii $X \cup Y$
- Pentru mulțimea $\{bere, scutece, lapte\}$ se pot genera 6 reguli: $\{bere, scutece\} \longrightarrow \{lapte\}$, $\{bere\} \longrightarrow \{scutece, lapte\}$ etc.; toate acestea au același suport, indiferent de partiționarea în antecedent/consecvent
- Dacă o mulțime nu este frecventă, atunci toate regulile ce se pot construi pe baza ei pot fi eliminate fără a le mai calcula gradul de încredere
- Concluzie: se poate decupla calculul suportului și al confidenței
- Pașii de lucru:
  1. generarea mulțimilor frecvente, *i.e.* al celor pentru care suportul este cel puțin *minsup*
  2. generarea regulilor pe baza mulțimilor frecvente

Outline

Problema generării mulțimilor frecvente

Generarea mulțimilor frecvente este solicitantă computațional: pentru o mulțime de k produse se pot realiza $2^k - 1$ potențiale mulțimi frecvente (se exclude mulțimea vidă)

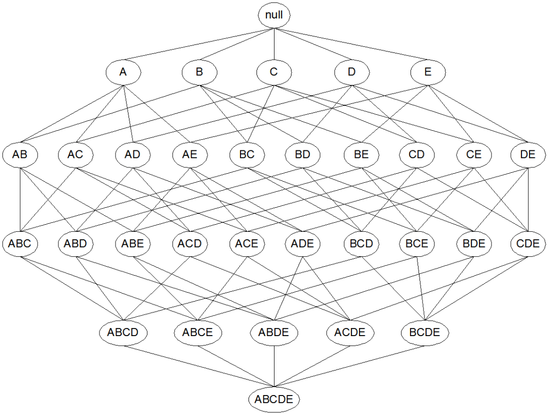
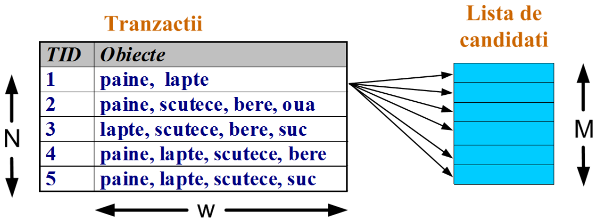


Figure 1: Latice de mulțimi

Varianta brute-force:

- Fiecare mulțime candidat din latice este considerat ca un candidat de mulțime frecventă
- Se calculează numărul suport al fiecărui candidat prin scanarea bazei de date
- Dacă un candidat e inclus într–o tranzacție se incrementează numărul suport



TID	Obiecte
1	paine, lapte
2	paine, scutece, bere, oua
3	lapte, scutece, bere, suc
4	paine, lapte, scutece, bere
5	paine, lapte, scutece, suc

Figure 2: Diferiți parametri ai datelor de intrare

Complexitate si remedii

- Complexitate: $O(NMw)$ unde: $N =$ numărul de tranzacții, $M = 2^k - 1$ este numărul de mulțimi candidat, w este numărul maxim de obiecte dintr-o tranzacție
- Modalități de reducere a complexității:
  - reducerea numărului de mulțimi candidat M – de exemplu prin principiul *Apriori*
  - reducerea numărului de comparații la confruntarea unei mulțimi cu o tranzacție prin structuri de date eficiente

Principiul *Apriori*

Teoremă (Principiul *Apriori*)

Dacă un set este frecvent, atunci oricare din subseturile sale este de asemenea frecvent.

Demonstrație: Pentru o tranzacție care conține mulțimea de obiecte $X = \{c, d, e\}$ este evident că ea conține și oricare din submulțimile lui X: $\{c, d\}$, $\{c, e\}$ etc. Mai mult, pentru o submulțime a lui X poate exista o tranzacție care să o conțină, dar să nu conțină și pe X. Astfel, numărul suport pentru o submulțime a lui X este cel puțin numărul suport al lui X.

- teorema afirmă că:

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

adică o proprietate de anti-monotonie a suportului

Eliminarea mulțimilor infrecvente

- Contrapoziția teoremei este utilă pentru a face eliminarea mulțimilor care nu pot fi frecvente: *Dacă un set nu este frecvent, atunci oricare superset al lui nu poate să fie frecvent.*

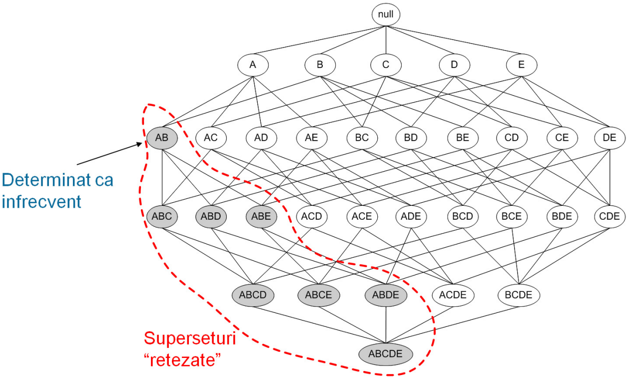


Figure 3: Retezarea mulțimilor infrecvente

Strategie

- Se pornește cu 1-mulțimi formate din câte un obiect
- 1-mulțimile nefrecvente se elimină
- Se generează 2-mulțimi combinând 1-mulțimi frecvente
- Pentru 2-mulțimile generate se calculează suportul; cele nefrecvente se elimină
- Se continuă procedeul pentru 3-mulțimi etc.
- Pe baza principiului *Apriori*, pentru generarea k-mulțimilor frecvente candidat se iau în considerare doar $(k-1)$-mulțimile frecvente

Exemplu de aplicare

Obiect	Numar
paine	4
suc	2
lapte	4
bere	3
scutece	4
oua	1

1-multimi

Multime	Numar
{paine,lapte}	3
{paine,bere}	2
{paine,scutece}	3
{lapte,bere}	2
{lapte,scutece}	3
{bere, scutece }	3

2-multimi

(Nu este nevoie sa se genereze candidati cu oua sau suc)

Valoarea suport minim ceruta: 3

3-multimi

Multime	Numar
{paine,lapte,scutece}	3

Daca se considera fiecare subset (brute force): ${}^6C_1 + {}^6C_2 + {}^6C_3 = 41$

Folosind principiul *Apriori*:
$6 + 6 + 1 = 13$,
reducere de 68%

Figure 4: Generarea mulțimilor frecvente folosind principiul *Apriori*

Schița algoritmului *Apriori*

- $k = 1$
- Se generează 1-mulțimi frecvente
- Repetă până când nu se mai pot identifica mulțimi frecvente:
  - se generează $(k+1)$-mulțimi candidat folosind k-mulțimile frecvente de la pasul anterior
  - se șterg $(k+1)$-mulțimile candidat care conțin k-mulțimi infrecvente (cu suportul sub prag)
  - se contorizează suportul fiecărei $(k+1)-$mulțimi prin parcurgerea tranzacțiilor
  - se șterg $(k+1)$-mulțimile candidat care nu sunt frecvente

Pseudocodul pentru algoritmul *Apriori*

$k = 1$.
$F_k = \{ i \mid i \in I \wedge \sigma(\{i\}) \geq N \times minsup\}$. {Find all frequent 1-itemsets}
repeat
    $k = k + 1$.
    C_k = apriori-gen(F_{k-1}). {Generate candidate itemsets}
    for each transaction $t \in T$ **do**
        C_t = subset(C_k, t). {Identify all candidates that belong to t}
        for each candidate itemset $c \in C_t$ **do**
            $\sigma(c) = \sigma(c) + 1$. {Increment support count}
        end for
    end for
    $F_k = \{ c \mid c \in C_k \wedge \sigma(c) \geq N \times minsup\}$. {Extract the frequent k-itemsets}
until $F_k = \emptyset$
Result = $\bigcup F_k$.

Probleme ce trebuie rezolvate eficient pentru *Apriori*

- Generarea mulțimilor candidat și retezarea
  1. generarea de k-mulțimi folosind $(k-1)$-mulțimi frecvente
  2. eliminarea candidaților care nu au suportul peste pragul impus
- Calculul numărului suportul al unei mulțimi

Generarea mulțimilor candidat (var 1)

- Generarea mulțimilor candidat:
  - orice algoritm trebuie să evite generarea de prea multe mulțimi candidat
  - trebuie să asigure generarea unei familii complete de mulțimi candidat
  - trebuie să evite generarea aceleiași mulțimi candidat de mai multe ori ($\{a, b, c\}$ poate proveni din $\{a, b\} \cup \{c\}$ sau din $\{a\} \cup \{b, c\}$ etc.)
- Sunt mai mulți algoritmi în această direcție
- Metoda forței brute:
  - Se consideră fiecare posibilitate de a obține o k-mulțime
  - Se elimină candidații cu suport prea mic
  - Generarea e simplă, calculul suportului pentru fiecare candidat este costisitor
  - Complexitatea metodei: $O(d \cdot 2^{d-1})$

Generarea mulțimilor candidat (var 2)

- Metoda $F_{k-1} \times F_1$:
  - se pleacă de la fiecare $(k-1)$-mulțime frecventă și se extinde cu obiecte (1-mulțimi) frecvente
  - procedeul e complet
  - se poate ajunge la generarea multiplă a aceleiași k-mulțimi
  - evitare: fiecare mulțime este menținută sortată lexicografic: $\{a, b, c\}$ și nu $\{b, a, c\}$ sau altfel
  - extinderea unei mulțimi $F_{k-1} = \{ob_1, ob_2, \ldots, ob_{k-1}\}$ se face numai cu mulțimi $F_1 = \{x\}$ unde $ob_{k-1} < x$
  - complexitate: $O(\sum_k k|F_{k-1}||F_1|)$
  - încă se pot genera prea multe k-mulțimi candidat

Generarea mulțimilor candidat (var 3)

- Metoda $F_{k-1} \times F_{k-1}$:
  - se reunesc două $(k-1)$-mulțimi doar dacă primele $k-2$ elemente din ele sunt identice (se presupune ordinea lexicografică):
  - mai clar: dacă $A = \{a_1, a_2, \ldots a_{k-1}\}$ și $B = \{b_1, b_2, \ldots b_{k-1}\}$ sunt două $(k-1)$-mulțimi, atunci ele se reunesc doar dacă:

$$a_i = b_i, \ \forall i = 1, \ldots, k-2, \ a_{k-1} \neq b_{k-1}$$

- este varianta propusă în articolul ce introduce algoritmul Apriori

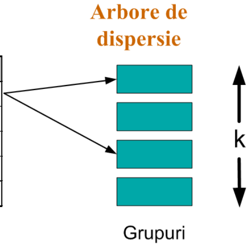
Reducerea numărului de comparații

- Varianta brute force: se scanează toată baza de date pentru a determina valoarea suport a fiecărei mulțimi candidat
  - Posibil, dar neeficient
- Pentru a reduce numărul de comparații stocăm candidații într–un arbore de dispersie (hash tree)
- Rezultat: în loc de a compara fiecare tranzacție cu fiecare mulțime candidat, se compară fiecare tranzacție cu grupuri de candidați din arbore; se vor actualiza doar valorile suport pentru mulțimi candidat din grupurile găsite

Tranzactii

N

TID	*Obiecte*
1	paine, lapte
2	paine, scutece, bere, oua
3	lapte, scutece, bere, suc
4	paine, lapte, scutece, bere
5	paine, lapte, scutece, suc



- Considerăm funcția de dispersie $h(p) = p \mod 3$
- Impunem restricția ca într–un nod frunză să nu avem mai mult de 3 mulțimi reprezentate; dacă este cazul, un nod va fi fragmentat în noduri copil
- Presupunem că avem mulțimile candidat: $\{1, 4, 5\}$, $\{1, 2, 4\}$, $\{4, 5, 7\}$, $\{1, 2, 5\}$, $\{4, 5, 8\}$, $\{1, 5, 9\}$, $\{1, 3, 6\}$, $\{2, 3, 4\}$, $\{5, 6, 7\}$, $\{3, 4, 5\}$, $\{3, 5, 6\}$, $\{3, 5, 7\}$, $\{6, 8, 9\}$, $\{3, 6, 7\}$, $\{3, 6, 8\}$
- Reprezentarea în arbore de dispersie:

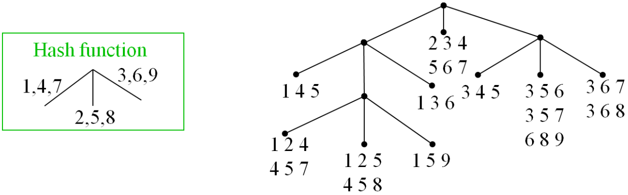
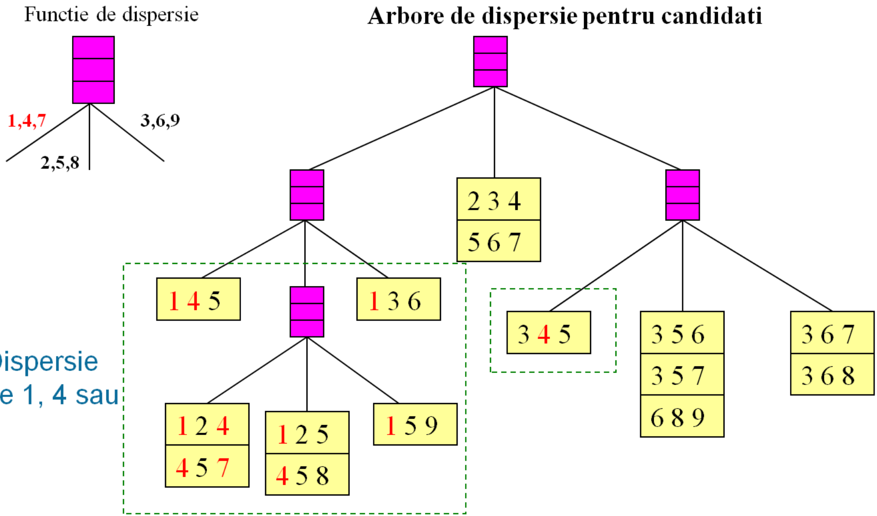


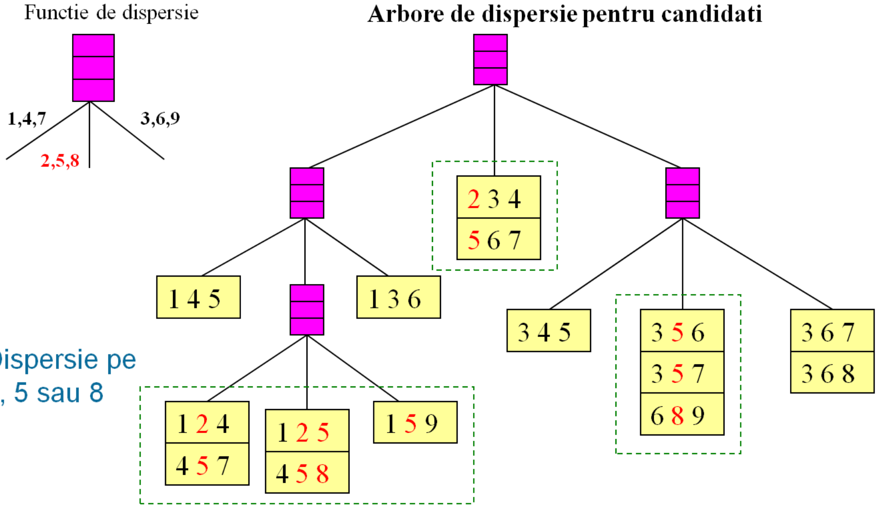
Figure 6: Arbore de dispersie pentru familia de mulțimi candidat

Reprezentarea mulțimilor candidați în arbore de dispersie



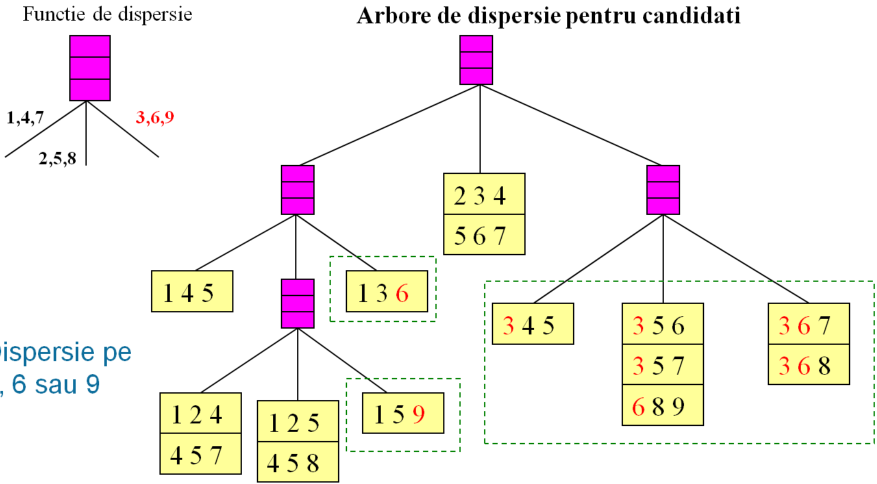
Dispersie pe 1, 4 sau 7

Reprezentarea mulțimilor candidați în arbore de dispersie



Dispersie pe 2, 5 sau 8

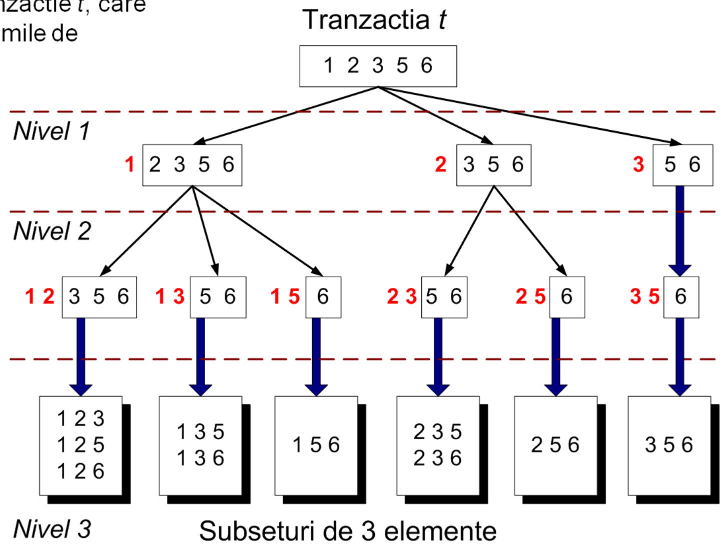
Reprezentarea mulțimilor candidați în arbore de dispersie



Dispersie pe 3, 6 sau 9

Generarea 3-submulțimilor unei tranzacții

Data fiind o tranzactie *t*, care sunt toate multimile de cardinal 3?



Căutarea potrivirilor între tranzacții și mulțimi candidat

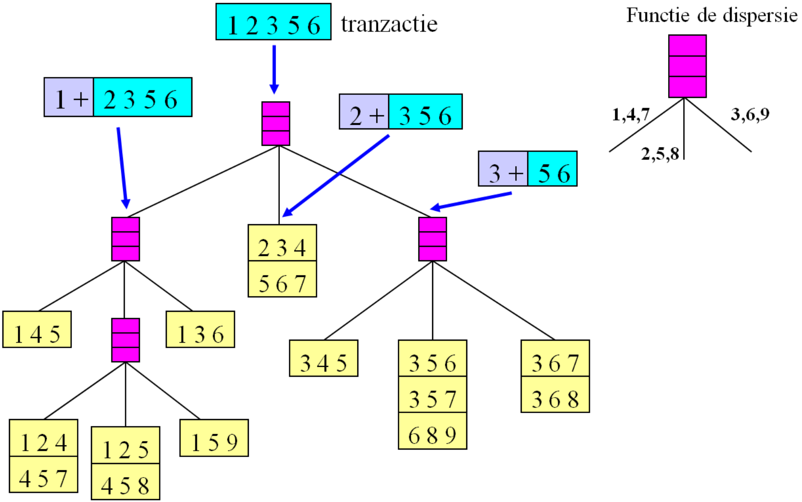


Figure 7: Se ia in considerare primul obiect al unei posibile 3-multimi ce se regaseste in tranzactie

Căutarea potrivirilor între tranzacții și mulțimi candidat

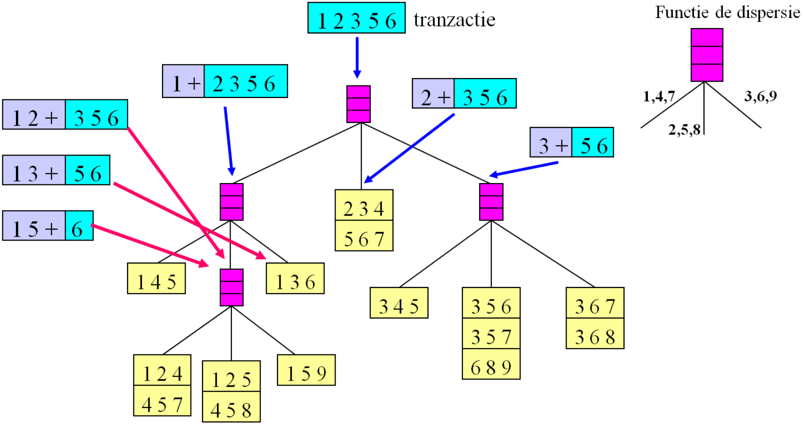


Figure 8: Se ia in considerare al doilea obiect al unei posibile 3-multimi ce se regaseste in tranzactie

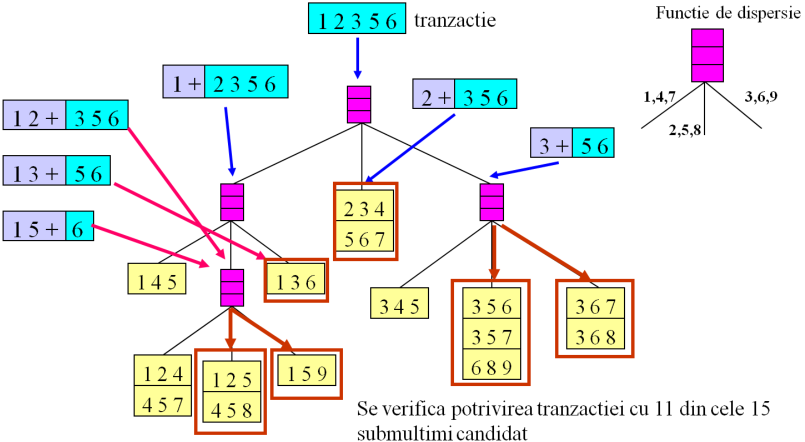


Figure 9: Se ia in considerare al treilea obiect al unei posibile 3-multimi ce se regaseste in tranzactie

Complexitatea computațională a generării de mulțimi frecvente

Factorii care influențează complexitatea generării:

- Alegerea valorii de *minsup*
  - micșorarea lui *minsup* duce la mai multe mulțimi declarate ca frecvente
- Numărul de obiecte din setul de date
  - poate duce la mărirea cardinalului maxim de mulțime frecventă
  - dacă numărul de mulțimi frecvente crește și el atunci cresc atât efortul computațional cât și costul I/O
- Dimensiunea bazei de date
  - fiecare tranzacție se compară cu mulțimile candidat; număr mare de tranzacții $\Rightarrow$ timp crescut pentru eliminarea mulțimilor candidat nefrecvente

Complexitatea computațională a generării de mulțimi frecvente

- Generarea 1-mulțimilor frecvente: $O(Nw)$
- Generarea mulțimilor candidat:

$$\sum_{k=2}^{w}(k-2)|C_k| < \text{costul unific. a 2 } k-1\text{-mulţimi } < \sum_{k=2}^{w}(k-2)|F_{k-1}|^2$$

- Eliminarea de mulțimi candidat infrecvente:

$$O\left(\sum_{k=2}^{w} k(k-2)|C_k|\right)$$

- Calcularea suportului

$$O\left(N\sum_{k} C_w^k \alpha_k\right)$$

unde α_k este costul actualizării unei k-mulțimi din arborele de dispersie.

Outline

Generarea regulilor

- Enunț: dându–se o mulțime frecventă L, să se găsească toate submulțimile nevide $f \subset L$ astfel încât regula $f \longrightarrow L - f$ să aibă gradul de încredere minim cerut
- Pentru mulțimea frecventă $\{A, B, C, D\}$ regulile candidat ce se pot obține sunt: $ABC \longrightarrow D$, $ABD \longrightarrow C$, $ACD \longrightarrow B$, $BCD \longrightarrow A, A \longrightarrow BCD, B \longrightarrow ACD, C \longrightarrow ABD$, $D \longrightarrow ABC, AB \longrightarrow CD, AC \longrightarrow BD$, $AD \longrightarrow BC$, $BC \longrightarrow AD, BD \longrightarrow AC$, $CD \longrightarrow AB$
- Pentru $k = |L|$ sunt $2^k - 2$ reguli care se pot genera (ignorăm regulile cu antecedent sau consecvent nul)

Generarea regulilor

- Nu avem nicio proprietate de tip (anti)monotonie pentru gradul de încredere al regulilor
  - Pentru o regulă $X \longrightarrow Y$ și $\tilde{X} \subset X$, $\tilde{Y} \subset Y$ nu avem nicio relație permanent valabilă între $c(X \longrightarrow Y)$ și $c(\tilde{X} \longrightarrow \tilde{Y})$
- Dar avem o teoremă $\ddot\smile$

Teoremă

Dacă o regulă $X \longrightarrow Y - X$ nu satisface condiția de grad de încredere minim $c(X \longrightarrow Y - X) \geq minconf$ atunci nicio regulă $X' \longrightarrow Y - X'$ cu $X' \subset X$ nu va avea nici ea gradul de încredere minim.

Demonstrație: pentru regulile $X' \longrightarrow Y - X'$ și $X \longrightarrow Y - X$ confidențele sunt $s_1 = \sigma(Y)/\sigma(X')$ respectiv $s_2 = \sigma(Y)/\sigma(X)$. Pentru $X' \subset X$ avem că $\sigma(X') \geq \sigma(X)$. Ca atare, $s_1 \leq s_2$ și deci prima regulă nu poate avea un grad de încredere mai mare decât a doua.

Generarea regulilor: strategie de retezare

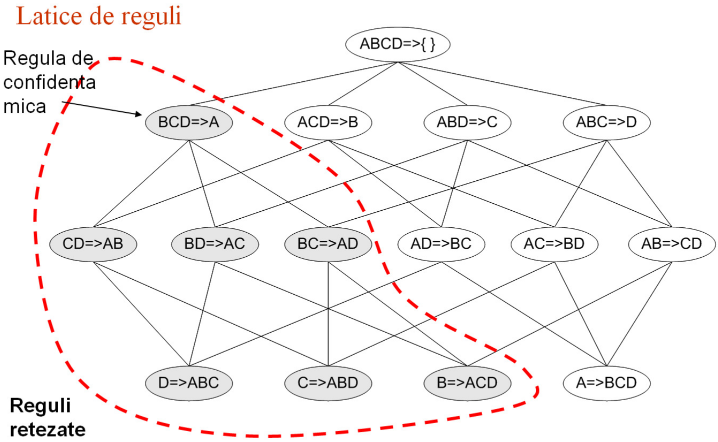


Figure 10: Retezarea regulilor aplicand teorema 2

Retezarea regulilor în algoritmul *Apriori*

- Se generează toate regulile care au doar un element în antecedent
- Se combină reguli care au ceva comun în sufix: de exemplu, din $\{a, c, d\} \longrightarrow \{b\}$ și $\{a, b, d\} \longrightarrow \{c\}$ se generează $\{a, d\} \longrightarrow \{b, c\}$
- Se fac eliminările de reguli conform teoremei 2

Generarea regulilor în algoritmul *Apriori*

Algorithm 6.2 Rule generation of the *Apriori* algorithm.

```
for each frequent k-itemset f_k, k ≥ 2 do
    H_1 = {i | i ∈ f_k}        {1-item consequents of the rule.}
    call ap-genrules(f_k, H_1.)
end for
```

Algorithm 6.3 Procedure ap-genrules(f_k, H_m).

```
k = |f_k|    {size of frequent itemset.}
m = |H_m|    {size of rule consequent.}
if k > m + 1 then
    H_{m+1} = apriori-gen(H_m).
    for each h_{m+1} ∈ H_{m+1} do
        conf = σ(f_k)/σ(f_k − h_{m+1}).
        if conf ≥ minconf then
            output the rule (f_k − h_{m+1}) ⟶ h_{m+1}.
        else
            delete h_{m+1} from H_{m+1}.
        end if
    end for
    call ap-genrules(f_k, H_{m+1}.)
end if
```

Outline

Reprezentarea compactă a mulțimilor frecvente

- Numărul de mulțimi frecvente poate să fie prohibitiv
- Se poate identifica o familie reprezentativă de mulțimi frecvente din care se pot obține toate celelalte mulțimi frecvente
- Variante: mulțimi frecvente maximale și mulțimi frecvente închise

Mulțimi frecvente maximale

Definiție

O mulțime frecventă maximală este o mulțime frecventă pentru care toate superseturile imediate sunt infrecvente.

(superset imediat al lui X este mulțime $X \cup \{y\}$, $y \notin X$)

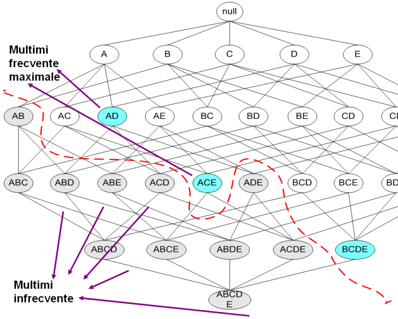


Figure 11: Familia de mulțimi frecvente maximale este reprezentată cu verde

Utilitatea mulțimilor frecvente maximale

- Toate mulțimile frecvente sunt generate de mulțimile frecvente maximale
- Ex: mulțimile frecvente din figura anterioară sunt într-una din situațiile:
  1. mulțimi care încep cu litera a și conțin c, d sau e
  2. mulțimi care încep cu b, c, d sau e.
- Există algoritmi care pot exploata eficient mulțimile frecvente maximale, fără a genera toate submulțimile
- Problemă: mulțimile frecvente maximale nu dau o modalitate de calcul al suportului submulțimilor frecvente pe care le generează

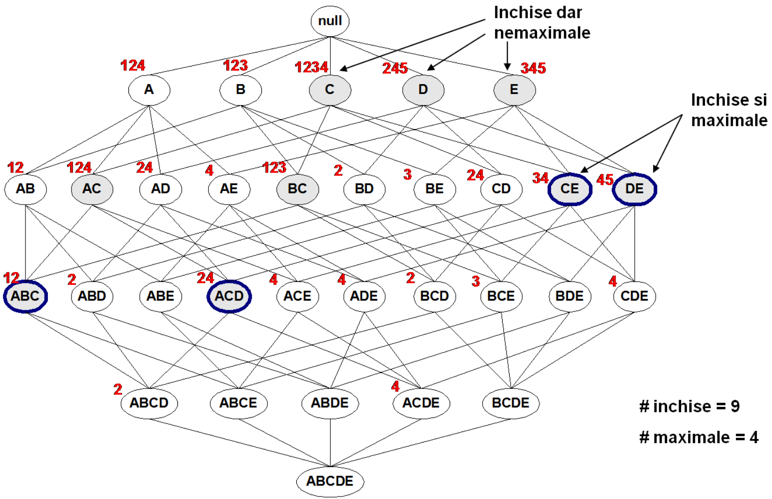
Mulțimi frecvente închise

Definiție (Mulțimi închise)

O mulțime X este închisă dacă niciunul din superseturile imediate ale sale nu are același suport ca ea.

Definiție (Mulțimi frecvente închise)

O mulțime X este frecventă închisă dacă este închisă și frecventă.



inchise = 9

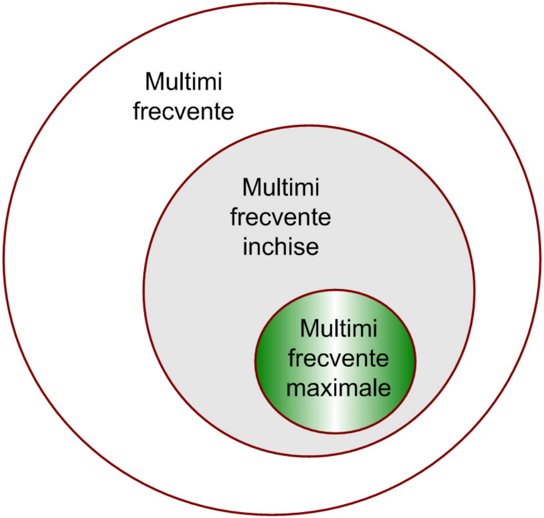
maximale = 4

Set de tranzacții:

TID	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

- 3 grupuri: $\{A1, \dots, A5\}$, $\{B1, \dots, B5\}$, $\{C1, \dots, C5\}$
- Pentru *minsup* $= 20\%$ avem număr total de mulțimi frecvente $= 93$
- Dar există doar 3 mulțimi frecvente închise: $\{A1, \dots, A5\}$, $\{B1, \dots, B5\}$, $\{C1, \dots, C5\}$

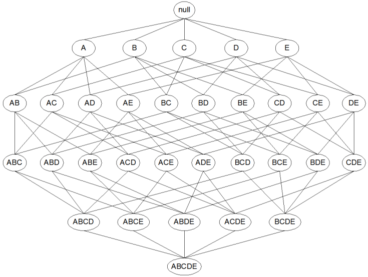
Relația între diferite tipuri de mulțimi



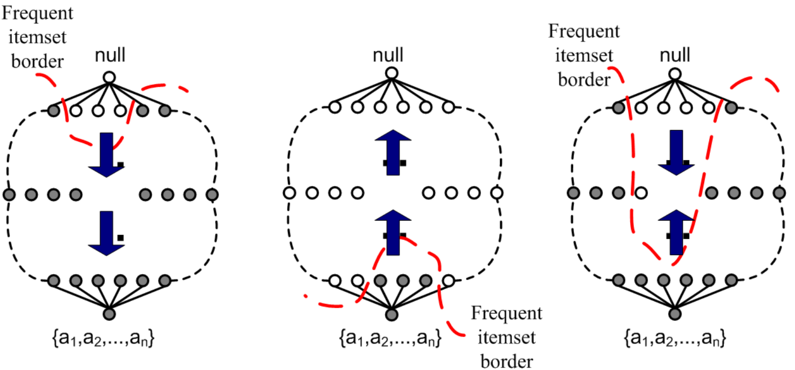
Outline

Metode alternative pentru generarea de mulțimi frecvente

- *Apriori* este unul din primii algoritmi eficienți care evită explozia combinatorială a generării seturilor frecvente
- Principiul de bază: retezarea conform teoremei de la pagina 40
- Deficiență: numărul mare de operații de I/O
- Deficiență: pentru seturile de tranzacții dense performanța scade mult
- Metode alternative: “de la general la specific”, “de la specific la general”, căutare bidirecțională
- Ideea de bază: determinarea mulțimilor frecvente este o problemă de căutare în graful laticii mulțimilor



Metode alternative pentru generarea de mulțimi frecvente



(a) "De la general la specific"

(b) "De la specific la general"

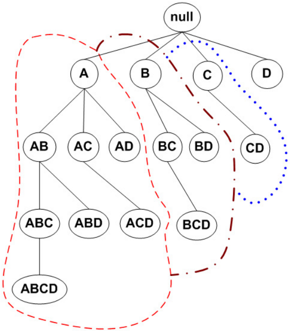
(c) Bidirectional

Metode alternative pentru generarea de mulțimi frecvente

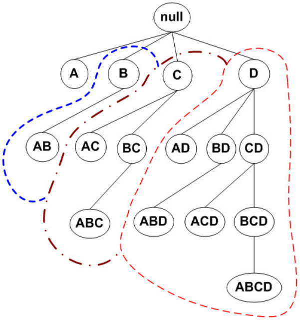
- "De la general la specific": în stilul algoritmului *Apriori*, de la o $(k-1)$-mulțime se ajunge la o k-mulțime; strategia e bună dacă lungimea maximă a unei mulțimi frecvente nu este prea mare
- "De la specific la general": se poate adapta principiul *Apriori*
- Căutare bidirecțională: combinație a precedentelor două, necesită mai multă memorie, dar permite determinarea rapidă a zonei de delimitare

Metode alternative pentru generarea de mulțimi frecvente

- Clase de echivalență: se partiționează mulțimea nodurilor din latice în clase de echivalență; se trece la o altă partiție numai când s–a terminat de explorat partiția curentă
- Exemple de partiționare: arbori de tip prefix/sufix



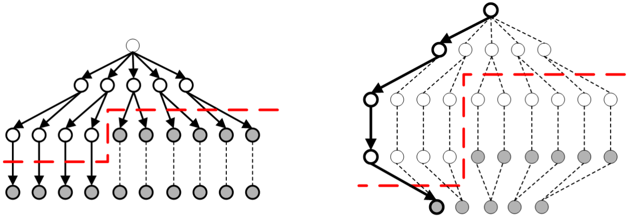
(a) Arbore prefix



(b) Arbore sufix

Metode alternative pentru generarea de mulțimi frecvente

- Căutarea "mai întâi în lățime": algoritmul *Apriori* funcționează astfel, trecând la k-mulțimi numai după ce s–au epuizat toate $(k-1)$-mulțimile
- Căutarea în adâncime este o variantă folosită pentru a determina mulțimile frecvente maximale
- Odată găsită o mulțime maximală, se poate face retezare



(a) Cautare "mai intai in latime"

(b) Cautare "mai intai in adancime"

Figure 12: Parcugeri alternative

Outline

Problema evaluării asocierilor

- Algoritmii pot duce la producerea unui număr mare de reguli
- Multe pot fi neinteresante sau redundante
- Exemplu de redundanță: dacă regulile $\{A, B, C\} \longrightarrow \{D\}$ și $\{A, B\} \longrightarrow \{D\}$ au același suport și grad de încredere
- Se pot folosi funcții de măsurare a gradului de interes care să reducă /sorteze regulile
- În cele prezentate până acum, doar suportul și gradul de încredere erau considerate

Schema unui proces de extragere de cunoștințe

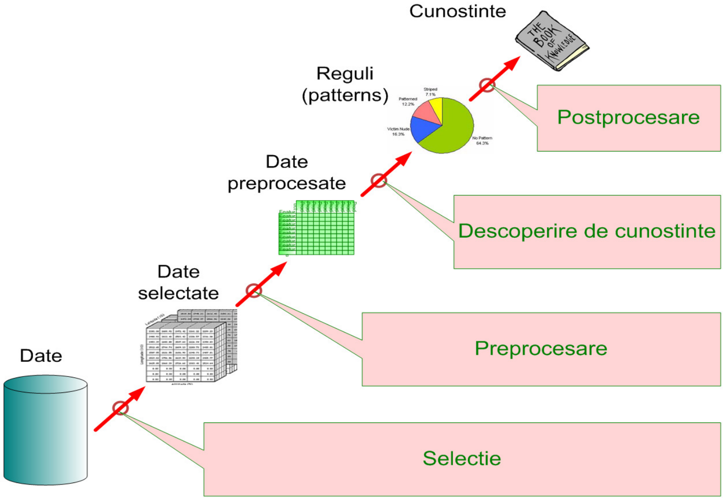


Figure 13: Pașii unui proces de extragere de cunoștințe. Postprocesarea conține evaluarea regulilor

Moduri de cuantificare al gradului de interes

- Funcții obiective:
  - folosesc statistici derivate din date pentru a determina gradul de interes
  - suport, grad de încredere, <u>corelație</u>
- Funcții subiective:
  - se referă la grad de interes pentru cunoașterea umană
  - exemplu: $\{unt\} \longrightarrow \{paine\}$ este de așteptat și deci neinteresant; dar $\{scutece\} \longrightarrow \{bere\}$ este ceva surprinzător
  - modalități de încorporare a subiectivismului:
    - vizualizare
    - filtrare bazată pe șabloane
    - ierarhie de concepte

Măsura obiectivă a interesului

- Măsură obiectivă: modalitate dependentă de date
- Necesită intervenție minimă din partea utilizatorului
- Punct de plecare pentru diferite măsuri, pe perechi de variabile binare: tabel de contingență

	Y	$\overline{Y}$	Total
X	f_{11}	f_{10}	f_{1+}
$\overline{X}$	f_{01}	f_{00}	f_{0+}
Total	f_{+1}	f_{+0}	N

Table 3: Tabel de contingență pentru regula $X \longrightarrow Y$. O valoare de forma $\overline{(\cdot)}$ reprezintă lipsa obiectului asociat în tranzație. f_{1+} (f_{+1}) reprezintă valoarea suport pentru X (respectiv Y).

Limitări ale cuplului suport-grad de încredere

- Considerăm studiul legăturii între cei care beau ceai sau cafea:

	Cafea	$\overline{Cafea}$	Total
Ceai	150	50	200
$\overline{Ceai}$	650	150	800
Total	800	200	1000

Table 4: Preferințe de consum.

- Considerăm regula: $\{Ceai\} \longrightarrow \{Cafea\}$: suport 15%, grad de încredere 75%
- Remarcăm însă că procentul celor care beau cafea este de 80%, mai mult decât gradul de încredere anterior
- Deci regula $\{Ceai\} \longrightarrow \{Cafea\}$ dă o indicație greșită față de starea actuală a datelor; știind că o persoană bea ceai, asta va scădea șansa ei ca să bea cafea
- Cauza: măsura de grad de încredere ignoră suportul mulțimii consecvent

Alte funcții obiective de măsurare a gradului de interes

- factor de ridicare (eng: lift)

$$lift(A \longrightarrow B) = \frac{c(A \longrightarrow B)}{s(B)}$$

- interes:

$$I(A,B) = \frac{s(A,B)}{s(A) \cdot s(B)} : \begin{cases} = 1 & \text{pentru } A \text{ şi } B \text{ independente} \\ > 1 & \text{pentru } A \text{ şi } B \text{ pozitiv corelate} \\ < 1 & \text{pentru } A \text{ şi } B \text{ negativ corelate} \end{cases}$$

- corelația Pearson pentru variabile binare:

$$\phi = \frac{f_{11} f_{00} - f_{01} f_{10}}{\sqrt{f_{1+} f_{+1} f_{0+} f_{+0}}} \in [-1, 1]$$

#	Measure	Formula
1	ϕ-coefficient	$\frac{P(A,B)-P(A)P(B)}{\sqrt{P(A)P(B)(1-P(A))(1-P(B))}}$
2	Goodman-Kruskal's (λ)	$\frac{\sum_j \max_k P(A_j,B_k)+\sum_k \max_j P(A_j,B_k)-\max_j P(A_j)-\max_k P(B_k)}{2-\max_j P(A_j)-\max_k P(B_k)}$
3	Odds ratio (α)	$\frac{P(A,B)P(\overline{A},\overline{B})}{P(A,\overline{B})P(\overline{A},B)}$
4	Yule's Q	$\frac{P(A,B)P(\overline{AB})-P(A,\overline{B})P(\overline{A},B)}{P(A,B)P(\overline{AB})+P(A,\overline{B})P(\overline{A},B)} = \frac{\alpha-1}{\alpha+1}$
5	Yule's Y	$\frac{\sqrt{P(A,B)P(\overline{AB})}-\sqrt{P(A,\overline{B})P(\overline{A},B)}}{\sqrt{P(A,B)P(\overline{AB})}+\sqrt{P(A,\overline{B})P(\overline{A},B)}} = \frac{\sqrt{\alpha}-1}{\sqrt{\alpha}+1}$
6	Kappa (κ)	$\frac{P(A,B)+P(\overline{A},\overline{B})-P(A)P(B)-P(\overline{A})P(\overline{B})}{1-P(A)P(B)-P(\overline{A})P(\overline{B})}$
7	Mutual Information (M)	$\frac{\sum_i \sum_j P(A_i,B_j)\log\frac{P(A_i,B_j)}{P(A_i)P(B_j)}}{\min(-\sum_i P(A_i)\log P(A_i), -\sum_j P(B_j)\log P(B_j))}$
8	J-Measure (J)	$\max\Big(P(A,B)\log(\frac{P(B\|A)}{P(B)}) + P(A\overline{B})\log(\frac{P(\overline{B}\|A)}{P(\overline{B})}),$ $P(A,B)\log(\frac{P(A\|B)}{P(A)}) + P(\overline{A}B)\log(\frac{P(\overline{A}\|B)}{P(\overline{A})})\Big)$
9	Gini index (G)	$\max\Big(P(A)[P(B\|A)^2 + P(\overline{B}\|A)^2] + P(\overline{A})[P(B\|\overline{A})^2 + P(\overline{B}\|\overline{A})^2]$ $-P(B)^2 - P(\overline{B})^2,$ $P(B)[P(A\|B)^2 + P(\overline{A}\|B)^2] + P(\overline{B})[P(A\|\overline{B})^2 + P(\overline{A}\|\overline{B})^2]$ $-P(A)^2 - P(\overline{A})^2\Big)$
10	Support (s)	$P(A,B)$
11	Confidence (c)	$\max(P(B\|A), P(A\|B))$
12	Laplace (L)	$\max\left(\frac{NP(A,B)+1}{NP(A)+2}, \frac{NP(A,B)+1}{NP(B)+2}\right)$
13	Conviction (V)	$\max\left(\frac{P(A)P(\overline{B})}{P(A\overline{B})}, \frac{P(B)P(\overline{A})}{P(B\overline{A})}\right)$
14	Interest (I)	$\frac{P(A,B)}{P(A)P(B)}$
15	cosine (IS)	$\frac{P(A,B)}{\sqrt{P(A)P(B)}}$
16	Piatetsky-Shapiro's (PS)	$P(A,B) - P(A)P(B)$
17	Certainty factor (F)	$\max\left(\frac{P(B\|A)-P(B)}{1-P(B)}, \frac{P(A\|B)-P(A)}{1-P(A)}\right)$
18	Added Value (AV)	$\max(P(B\|A) - P(B), P(A\|B) - P(A))$
19	Collective strength (S)	$\frac{P(A,B)+P(\overline{AB})}{P(A)P(B)+P(\overline{A})P(\overline{B})} \times \frac{1-P(A)P(B)-P(\overline{A})P(\overline{B})}{1-P(A,B)-P(\overline{AB})}$
20	Jaccard (ζ)	$\frac{P(A,B)}{P(A)+P(B)-P(A,B)}$
21	Klosgen (K)	$\sqrt{P(A,B)}\max(P(B\|A) - P(B), P(A\|B) - P(A))$

Figure 14: Măsuri propuse

- Pe baza funcțiilor se pot face ordonări ale regulilor
- Ordinea poate să difere de la o funcție de interes la alta

Exemplu	f_{11}	f_{10}	f_{01}	f_{00}
E1	8123	83	424	1370
E2	8330	2	622	1046
E3	9481	94	127	298
E4	3954	3080	5	2961
E5	2886	1363	1320	4431
E6	1500	2000	500	6000
E7	4000	2000	1000	3000
E8	4000	2000	2000	2000
E9	1720	7121	5	1154
E10	61	2483	4	7452

Figure 15: 10 exemple de tabele de contingență

#	ϕ	λ	α	Q	Y	κ	M	J	G	s	c	L	V	I	IS	PS	F	AV	S	ζ	K
E1	1	1	3	3	3	1	2	2	1	3	5	5	4	6	2	2	4	6	1	2	5
E2	2	2	1	1	1	2	1	3	2	2	1	1	1	8	3	5	1	8	2	3	6
E3	3	3	4	4	4	3	3	8	7	1	4	4	6	10	1	8	6	10	3	1	10
E4	4	7	2	2	2	5	4	1	3	6	2	2	2	4	4	1	2	3	4	5	1
E5	5	4	8	8	8	4	7	5	4	7	9	9	9	3	6	3	9	4	5	6	3
E6	6	6	7	7	7	7	6	4	6	9	8	8	7	2	8	6	7	2	7	8	2
E7	7	5	9	9	9	6	8	6	5	4	7	7	8	5	5	4	8	5	6	4	4
E8	8	9	10	10	10	8	10	10	8	4	10	10	10	9	7	7	10	9	8	7	9
E9	9	9	5	5	5	9	9	7	9	8	3	3	3	7	9	9	3	7	9	9	8
E10	10	8	6	6	6	10	5	9	10	10	6	6	5	1	10	10	5	1	10	10	7

Figure 16: Sortarea pe baza diferitelor reguli

Proprietăți ale funcțiilor obiective

De văzut din bibliografie:

- Proprietatea de inversiune
- Proprietatea de adăugare nulă
- Proprietatea de scalare

De citit: paradoxul lui Simpson și necesitatea stratificării datelor înaintea extragerii de reguli

Outline

Distribuții oblice

- Uneori datele au următoarea formă: foarte multe obiecte cu suport mic, puține cu suport mare
- O atare distribuție este puternic neechilibrată (eng: skewed) și nu poate fi tratată uniform
- Dacă pragul *minsup* este ales prea mic atunci algoritmul *Apriori* (sau oricare altul) poate genera excesiv de multe mulțimi frecvente $\Rightarrow$ consum mare de memorie, posibil relații întâmplătoare
- Dacă *minsup* este prea mare se pot rata niște reguli utile
  - exemplu: bijuterii - se cumpără rar în comparație cu alte produse, dar profitul adus e considerabil
- Pentru setul de date Public Use Microarray Sample census data distribuția datelor este:

Group	G_1	G_2	G_3
Support	$< 1\%$	$1\% - 90\%$	$> 90\%$
Number of Items	1735	358	20

Distribuții oblice

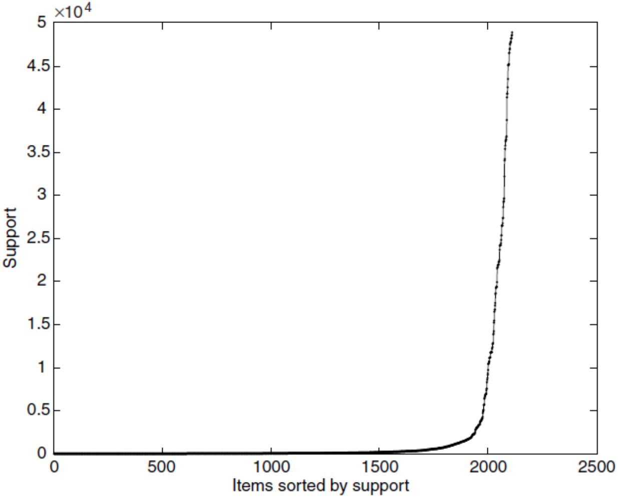


Figure 17: Distribuție oblică pentru setul de date PUMS census data

Distribuții oblice

- Dacă se lasă valoare *minsup* mică, atunci o mulțime poate să combine obiecte cu suport mare și mic
- Obiectele din mulțime însă pot avea o corelație mică
- Atfel de mulțimi se numesc șabloane cu suport încrucișat (eng: cross-support patterns)
- Pentru $minsup = 0.05\%$ avem 18847 perechi frecvente, din care 93% sunt cu obiecte din G1 și G3; corelația maximă este însă 0.029 - prea puțin
- Chiar mărirea pragului *minconf* poate fi inefectivă; consecventul unei reguli poate avea suport mare deci șabloane cu suport înclucișat pot încă să apară

Distribuții oblice

Definiție (Șabloane cu suport încrucișat)

Un șablon cu suport încrucișat este o mulțime $X = \{i_1, i_2, \ldots, i_k\}$ *pentru care raportul suporturilor*

$$r(X) = \frac{\min[s(i_1), s(i_2), \ldots, s(i_k)]}{\max[s(i_1), s(i_2), \ldots, s(i_k)]} \tag{3}$$

este mai mic decât un prag specificat de utilizator h_c.

- Un alt mod de detectare a șabloanelor cu suport încrucișat: se examinează gradul de încredere minim care se poate extrage dintr–o mulțime dată, *i.e.* măsura *h-grad de încredere*:

$$\frac{s\left(i_1, i_2, \ldots, i_k\right)}{\max\left[s(i_1), s(i_2), \ldots, s(i_k)\right]}$$

- Criteriul de eliminare a unui șablon încrucișat X: șablonul se elimină dacă

$$h - confidenta(X) \leq \frac{\min[s(i_1), s(i_2), \ldots, s(i_k)]}{\max[s(i_1), s(i_2), \ldots, s(i_k)]} \leq h_c$$

- Valoarea peste h_c a h-confidenței ne asigură că obiectele din mulțime sunt puternic corelate între ele