

Asszociációs szabályok keresése

Csima Judit

BME, VIK,
Számítástudományi és Információelméleti Tanszék

2016. április 29.

Alapfeladat

- adottak vásárlói kosarak (tranzakciók): miket vásároltak együtt
- cél: olyan szabályokat felállítani, hogy ha valaki vesz X -et, akkor esélyes, hogy vesz Y -t is
- X és Y lehet több elemből álló halmaz is
- egy ilyen szabály nem jelent ok-okozati összefüggést!
- de egy ilyen szabályból hasznot lehet húzni: pl. árazzuk le X -et kicsit, emeljük meg Y árát jobban

Association Rule Mining

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Market-Basket transactions

<i>TID</i>	<i>Items</i>
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Association Rules

$\{\text{Diaper}\} \rightarrow \{\text{Beer}\},$
 $\{\text{Milk, Bread}\} \rightarrow \{\text{Eggs, Coke}\},$
 $\{\text{Beer, Bread}\} \rightarrow \{\text{Milk}\},$

Implication means co-occurrence,
not causality!

Jelölések, alapfogalmak

- elem (item): amit lehet venni, pl. tej, pelenka
- tranzakció: amiket együtt vettek, egy vásárlói kosár
- egyszerű modell: darabszám nem számít, csak az, hogy szerepel-e egy adott termék a kosárban
- cél: $X \rightarrow Y$ szabályok találása, ahol X, Y nemüres, diszjunkt elemhalmazok ($X \cap Y = \emptyset$)
- elnevezés: ha $|X| = k$, akkor X -et k -elemű elemhalmaznak (k -item set) hívjuk

Mikor jó egy szabály?

- ha X és Y sok tranzakcióban szerepel együtt (különben nem érdekes, pl. Hello Kittys papucs és motorosfűrés)
- ha az X -et tartalmazó kosarak jelentős része tartalmaz Y -t is
- lesz még valami más is, de először nézzük ezeket

Támogatottság, support

- $X \rightarrow Y$ abszolút támogatottsága (support count): hány kosárban van X és Y is, jele $\sigma(X \cup Y)$
- $X \rightarrow Y$ támogatottsága (support):
$$\text{supp}(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\text{number of transactions}}$$
- csak olyan szabályokat akarunk, amikre a supp elég nagy (egy küszöbnél nagyobb, a küszöb neve min_sup)
- azért, mert ha kicsi a support, akkor arra nem lehet stratégiát építeni, az lehet, hogy véletlen egybeesés (Hello Kitty és fűrész)

Támogatottság, support

- $supp(X \rightarrow Y)$ csak $X \cup Y$ -től függ, attól nem, hogy X és Y hogy oszlik el a szabály két oldalára
- ha $supp(X \rightarrow Y)$ a küszöbnél nagyobb, akkor $X \cup Y$ neve gyakori elemhalmaz (frequent item set)
- az persze kérdés, hogy mi a küszöb

Definition: Frequent Itemset

- **Itemset**
 - A collection of one or more items
 - ◆ Example: {Milk, Bread, Diaper}
 - k-itemset
 - ◆ An itemset that contains k items
- **Support count (σ)**
 - Frequency of occurrence of an itemset
 - E.g. $\sigma(\{\text{Milk, Bread, Diaper}\}) = 2$
- **Support**
 - Fraction of transactions that contain an itemset
 - E.g. $s(\{\text{Milk, Bread, Diaper}\}) = 2/5$
- **Frequent Itemset**
 - An itemset whose support is greater than or equal to a *minsup* threshold

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Megbízhatóság, confidence

- $X \rightarrow Y$ megbízhatósága (confidence): $conf(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)}$
- azaz: az X -et tartalmazó kosarak mekkora részében van Y is
- az a szabály érdekes, aminél a conf egy küszöbnél (jele min_conf) nagyobb
- ez mutatja, hogy X eladásai befolyásolhatják Y eladásait

Definition: Association Rule

□ Association Rule

- An implication expression of the form $X \rightarrow Y$, where X and Y are itemsets
- Example:
 $\{\text{Milk, Diaper}\} \rightarrow \{\text{Beer}\}$

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

□ Rule Evaluation Metrics

- Support (s)
 - ◆ Fraction of transactions that contain both X and Y
- Confidence (c)
 - ◆ Measures how often items in Y appear in transactions that contain X

Example:

$\{\text{Milk, Diaper}\} \Rightarrow \text{Beer}$

$$s = \frac{\sigma(\text{Milk, Diaper, Beer})}{|T|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk, Diaper, Beer})}{\sigma(\text{Milk, Diaper})} = \frac{2}{3} = 0.67$$

Szabályok keresése

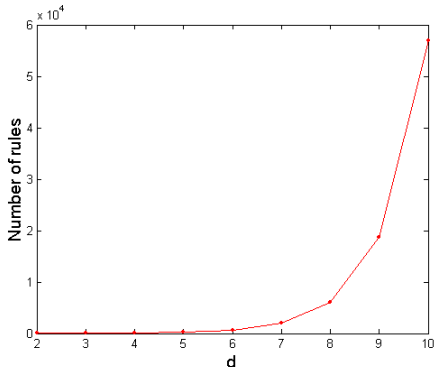
- olyan szabály kell, ahol $supp \geq min_sup$ és $conf \geq min_conf$
- $supp$ mindig kisebb, mint $conf$ egy adott szabály esetén
- két külön küszöb van, ez két külön feltétel
- egy szabály el tud bukni mindkét feltételen

Brute-force módszer

- minden olyan X és Y elemhalmaz végignézése, amikre $X \cap Y = \emptyset$
- minden ilyenre supp és conf számolása, rosszak kidobása
- ez sajnos túl sok: nagyjából $\sum_{k=1}^{d-1} \binom{d}{k} 2^{d-k}$, durván exponenciális (ahol d darab lehetséges item van)

Computational Complexity

- Given d unique items:
 - Total number of itemsets = 2^d
 - Total number of possible association rules:



$$R = \sum_{k=1}^{d-1} \left[\binom{d}{k} \times \sum_{j=1}^{d-k} \binom{d-k}{j} \right]$$
$$= 3^d - 2^{d+1} + 1$$

If $d=6$, $R = 602$ rules

- egy $X \rightarrow Y$ szabály akkor jó, ha elég nagy a *supp* és a *conf* is
- *supp* csak $X \cup Y$ -től függ, először ezt a lécezt kell megugrania a potenciális szabálynak
- ha $\text{supp}(Z = X \cup Y)$ elég nagy, akkor jön az, hogy hogyan legyen Z szétosztva a szabály két oldalára, hogy a *conf* is elég nagy legyen
- válasszuk szét a két ellenőrzést:
 - először keressünk gyakori elemhalmazokat (Z), csak ilyenekből lehet jó szabály
 - nézzük meg, hogy egy gyakori elemhalmazból milyen nagy megbízhatóságú szabály gyártható le

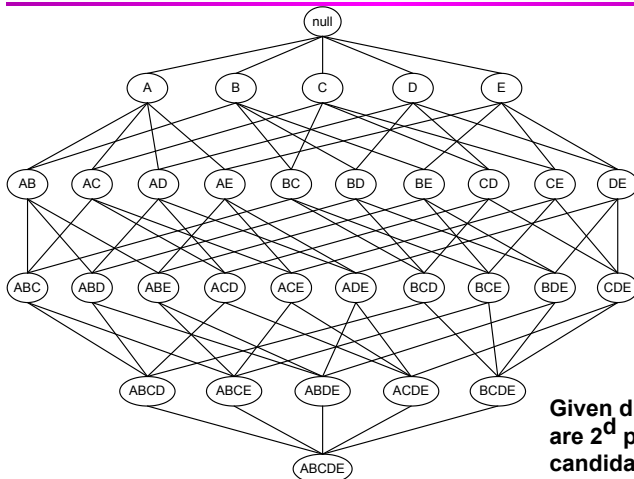
Általános algo

- először legenerálom az összes gyakori elemhalmazt (adott a *min_sup*)
- ezután minden egyes gyakori elemhalmazból megcsinálom a nagy megbízhatóságú szabályokat

Brute-force módszer gyakori elemhalmazok keresésére

- minden nem-üres részhalmazt végignézek
- ez nem jó, túl sok van: $2^d - 1$
- észrevétel: a részhalmazok háló-struktúrát alkotnak
- sőt: ha M jelölt van a gyakori halmazra és N tranzakció, akkor minden jelöltet össze kell vetni minden tranzakcióval (bent van-e a jelölt az adott kosárban)
- ez $O(NMw)$, ahol w a tranzakciók nagysága (hány elem van benne) és már csak M maga $2^d - 1$ a brute-force esetben

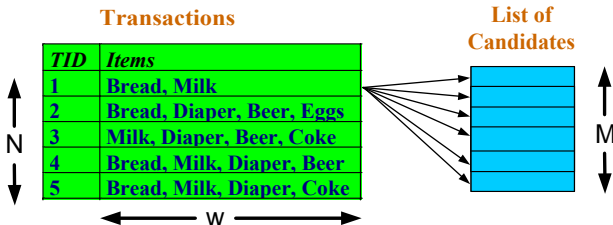
Frequent Itemset Generation



Given d items, there are 2^d possible candidate itemsets

Frequent Itemset Generation

- Brute-force approach:
 - Each itemset in the lattice is a **candidate** frequent itemset
 - Count the support of each candidate by scanning the database



- Match each transaction against every candidate
- Complexity $\sim O(NMw) \Rightarrow$ **Expensive since $M = 2^d$!!!**

Hogyan lehetne gyorsítani?

- csökkentsük M -et: ne az összes részhalmazt nézzük ész nélkül, hanem szűrjünk valahogy mielőtt elkezdjk őket összevetni a tranzakciókkal
- csökkentsük N -t (a tranzakciók számát) vagy a hosszukat (w -t)
- használjunk valami ügyes adatszerkezetet a jelöltek és a tranzakciók összevetésére
- most először az első lehetőséget nézzük: M csökkentése

Jelöltek számának csökkentése

- cél: gyakori elemhalmazok keresése
- hogyan: a részhalmazokból álló hálót úgy bejárni, hogy minél több elemhalmazt ki tudjunk zárni, minél előbb
- követelmények:
 - minden gyakort generáljunk végül
 - egyet csak egyszer
 - ne dolgozzunk túl sokat a generálás során
- egyszerűsítés: item-ek neve helyett számokat használunk

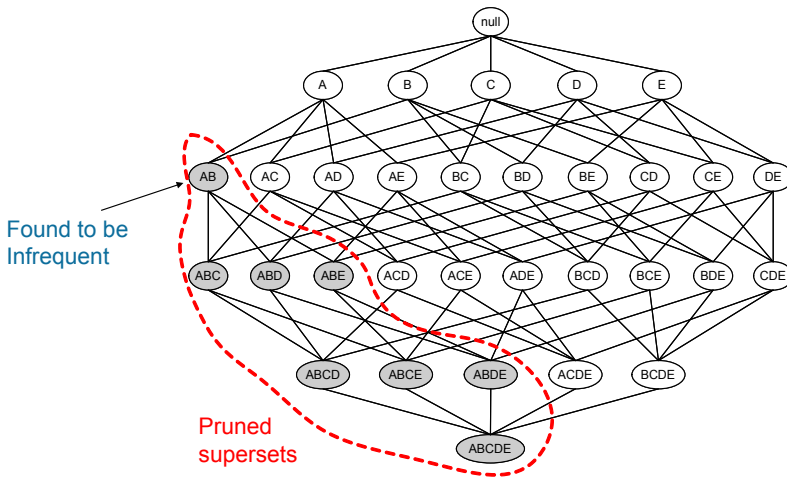
Apriori-elv

- Apriori-elv: ha X gyakori, akkor minden részhalmaza gyakori
- mert ha $Y \subseteq X$, akkor $supp(Y) = \frac{\sigma(Y)}{N} \geq \frac{\sigma(X)}{N} = supp(X)$ (itt N a tranzakciók száma)
- ugyanez máshogy: ha Y nem gyakori, akkor senki se gyakori, aki Y -t tartalmazza
- ezt úgy is szokták mondani, hogy a support függvény anti-monoton

Apriori algo

- haladjunk k szerint növeően $k = 1$ -től
- ha egy k elemű elemhalmaz nem gyakori, akkor minden nála bővebb elemhalmaz kizárható (infrequent)
- egy k elemű halmaz csak akkor lehet gyakori, ha minden $k - 1$ elemű részhalmaza gyakori

Illustrating Apriori Principle



Apriori algo

- 1 egyszer végignézek minden tranzakciót és kigyűjtöm az egy-elemű gyakoriakat (ehhez minden x elemre kiszámolom $\sigma(x)$ -et), ez az F_1 halmaz
- 2 $k = 2$
 - $C_2 = 2$ -elemű esélyesek: akiknek mindkét tagja F_1 -ben van
 - $F_2 = C_2$ -beli jelöltek összevetése a tranzakciókkal, a gyakoriak F_2
- 3 $k = 3$
 - $C_3 = 3$ -elemű esélyesek: akiknek minden kételemű részhalmaza F_2 -ben van
 - $F_3 = C_3$ -beli jelöltek összevetése a tranzakciókkal, a gyakoriak F_3
- 4 k általában
 - $C_k = k$ -elemű esélyesek: akiknek minden $k - 1$ -elemű részhalmaza F_{k-1} -ben van
 - $F_k = C_k$ -beli jelöltek összevetése a tranzakciókkal, a gyakoriak F_k

Illustrating Apriori Principle

Item	Count
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

Items (1-itemsets)



Itemset	Count
{Bread,Milk}	3
{Bread,Beer}	2
{Bread,Diaper}	3
{Milk,Beer}	2
{Milk,Diaper}	3
{Beer,Diaper}	3

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3



Triplets (3-itemsets)

Itemset	Count
{Bread,Milk,Diaper}	3



If every subset is considered,
 ${}^6C_1 + {}^6C_2 + {}^6C_3 = 41$
 With support-based pruning,
 $6 + 6 + 1 = 13$

- úgy könnyű F_{k-1} -ből C_k képzése, ha az elemhalmazokban növekvő sorrendben vannak
- mert ekkor könnyű C_k -ba tartozó k -elemű jelölteket előállítani
- úgy, hogy két olyan (rendezett) $(k-1)$ -eleműt keressék F_{k-1} -ben, amiknek az első $k-2$ tagja ugyanaz
- így biztos, hogy minden k elemű gyakori bekerül C_k -ba, pontosan egyszer
- nem kell azzal foglalkozni, hogy C_k -ból kiszűrjük a duplikátumokat

Hogyan lesz tehát F_k F_{k-1} -ből?

- F_{k-1} -ben minden elemhalmazban rendezetten vannak az elemek
- két F_{k-1} -beli $k - 1$ elemű elemhalmazból akkor csinálunk egy k elemű jelöltet C_k -ba, ha
 - az első $k - 2$ tagjuk ugyanaz
 - az így létrejött k elemű elemhalmaz többi $k - 1$ elemű részhalmaza is F_{k-1} -ben van (ez még $k - 2$ ellenőrzés)
- az így kapott C_k minden elemhalmazát összevetjük minden tranzakcióval (ténylegesen leszámoljuk a σ -kat)

További gyorsítás

- láttuk, hogy a nagy meló a jelöltek és a tranzakciók összevetése (az egyes jelöltek előfordulásainak kiszámolásához)
- eddig: mielőtt összevetjük a jelölteket a tranzakciókkal, csökkentjük a jelöltek számát (minden k elemű részhalmaz helyett csak C_k -beliek)
- további lehetőségek:
 - csökkentjük a tranzakciók hosszát: a nem gyakori egy eleműeket dobjuk ki az elején minden tranzakcióból
 - csökkentjük a tranzakciók számát: F_k előállításában, akkor dobunk ki minden k -nál nem hosszabb tranzakciót

Szabályok generálása gyakori elemhalmazokból

- tegyük fel, hogy megvannak a gyakori elemhalmazok
- minden Z gyakori elemhalmazból le szeretnénk generálni az összes olyan $X \rightarrow Y$ szabályt, ahol
 - $Z = X \cup Y$, X és Y sem üres
 - $\text{supp}(X \rightarrow Y) \geq \text{min_sup}$
 - $\text{conf}(X \rightarrow Y) \geq \text{min_conf}$
- a min_sup -os dolog Z gyakorisága miatt megvan
- a conf -os feltételt kéne teljesíteni

Brute-force algo

- adott Z esetén minden lehetséges módon $X, Z \setminus X$ kiválasztása
- minden választásra $conf(X \rightarrow Z \setminus X)$ számolása
- ehhez $\sigma(X)$ kell
- de $2^{|Z|} - 2$ lehetőség van X -re, ez túl sok

Ha adott egy Z és ennek egy X részhalmazából, mint baloldalból származtatott szabály nem jó (conf -ja kisebb, mint min_conf), akkor az összes olyan X' baloldalból se lesz jó szabály, ahol $X' \subseteq X$.

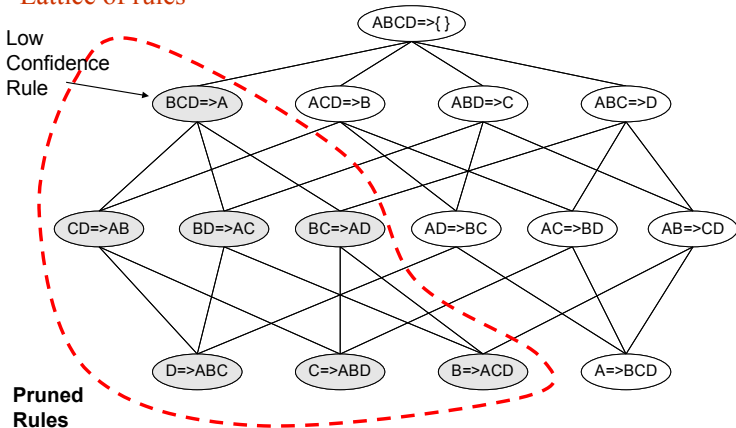
Biz.
$$\text{conf}(X' \rightarrow Z \setminus X') = \frac{\sigma(Z)}{\sigma(X')} \leq \frac{\sigma(Z)}{\sigma(X)} < \text{min_conf}$$
A középen álló egyenlőtlenség azért igaz, mert $X' \subseteq X$ miatt $\sigma(X') \geq \sigma(X)$.

Észrevétel másként

- ha egy adott Z -ből generálok szabályokat és egy Y jobboldalú szabály rossz, akkor minden olyan szabály is rossz, ahol a jobboldal Y -nál bővebb
- ez hasonló az Apriori-elvhez
- csináljuk ugyanazt, amit az Apriori-algoban:
 - adott Z esetén először legeneráljuk az 1-elemű jobboldalú jó szabályokat
 - növeljük a szabályok jobboldalának hosszát, csak olyan jobboldalak jönnek be, amiknek minden eggyel kisebb részalmazáéhoz tartozó szabály jó volt

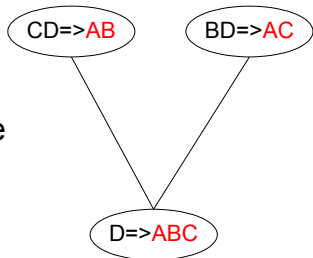
Rule Generation for Apriori Algorithm

Lattice of rules



Rule Generation for Apriori Algorithm

- Candidate rule is generated by merging two rules that share the same prefix in the rule consequent
- $\text{join}(CD \Rightarrow AB, BD \Rightarrow AC)$ would produce the candidate rule $D \Rightarrow ABC$
- Prune rule $D \Rightarrow ABC$ if its subset $AD \Rightarrow BC$ does not have high confidence



Apriori-elven működő szabálygenerálás Z -ből

- egyelemű jobboldalú szabályokra *conf* számolása, csak a jók maradnak
- minden szabály jobboldalán rendezve tartjuk az elemeket
- $k - 1$ hosszú jobboldalról k hosszú jobboldalra:
 - ha van két olyan $k - 1$ hosszú jobboldal, akiknek az első $k - 2$ tagja megegyezik, akkor ezekből unióval k hosszú jobboldalt képezünk (ezt minden lehetséges módon meg tesszük)
 - leellenőrizzük, hogy a két, generáló $k - 1$ hosszú részhalmazon kívüli többi $k - 2$ darab $k - 1$ elemű részhalmazhoz is jó szabály tartozott
 - aki ezen a szűrőn is átmegy, arra *conf*-ot számolok, aki ezt is túléli az lesz jó, k hosszú jobboldalú szabály