

Asocijativna pravila

Dubinska analiza podataka

8. predavanje

Pripremio: izv. prof. dr. sc. Alan Jović

Ak. god. 2023./2024.

Sadržaj

- Pronalaženje čestih obrazaca i asocijativna pravila
- Algoritmi asocijativnih pravila
 - Apriori
 - PCY
 - FP-Growth
 - EFIM
- Primjena asocijativnih pravila
 - Sustavi preporučivanja
 - Predobrada podataka

Pronalazak čestih obrazaca i asocijativna pravila

Pronalazak čestih obrazaca i asocijativna pravila

- Engl. *frequent pattern mining, frequent pattern discovery, association rules*
- Područje dubinske analize podataka (ne strojnog učenja) koje ima zadatak pronaći **česte i relevantne kombinacije vrijednosti značajki** (tzv. **obrasce**) u velikim skupovima podataka
 - Obrasci mogu biti:
 - **skupovi artikala** (engl. (i dalje) **itemsets**) – najčešći slučaj, za tablične podatke
 - podstrukture skupa podataka (kod modela podataka u obliku grafova)
 - podsekvence (kod vremenskih nizova podataka)
- Na temelju itemsetova najčešće se grade **asocijativna pravila**
- Vrlo široko područje s nizom potproblema i sličnih problema, npr. vidjeti:
 - <https://www.philippe-fournier-viger.com/spmf/index.php>

Primjena asocijativnih pravila

- U praksi, najčešća primjena je u **analizi afiniteta korisnika** (engl. *affinity analysis*), poznatijoj pod nazivom **analiza potrošačke košarice** (engl. *market basket analysis*)
 - Iskoristivo u **sustavima za preporučivanje** (engl. *recommender system*) u trgovini, industriji i medicini
- Može se iskoristiti i za **klasifikacijske probleme** te u sustavima za pronalazak **slijednih obrazaca u podacima** (engl. *sequential pattern mining*)
 - Npr. pravila ponavljanja određenih sekvenci nukleotida ili aminokiselina

Pronalazak čestih obrazaca – terminologija

- Pretpostavimo tablični oblik podataka u kojemu su retci (primjerci) **transakcije** koje su provedene, a stupci **artikli** (određenog naziva) koji su kupljeni, s mogućim vrijednostima 0 (nije kupljen) ili 1 (kupljen je)
- **Ne postoji ciljna klasa** u smislu nekog artikla (ili ona nije bitna)

| ID | Čaj | Kava | Čokolada | Mlijeko | Začin | Krafna |
|----|-----|------|----------|---------|-------|--------|
| T1 | 1 | 0 | 1 | 1 | 1 | 0 |
| T2 | 0 | 1 | 1 | 1 | 0 | 1 |
| T3 | 1 | 1 | 1 | 0 | 0 | 0 |
| T4 | 0 | 1 | 1 | 0 | 0 | 1 |

Itemset

| ID | Čaj | Kava | Čokolada | Mlijeko | Začin | Krafna |
|----|-----|------|----------|---------|-------|--------|
| T1 | 1 | 0 | 1 | 1 | 1 | 0 |
| T2 | 0 | 1 | 1 | 1 | 0 | 1 |
| T3 | 1 | 1 | 1 | 0 | 0 | 0 |
| T4 | 0 | 1 | 1 | 0 | 0 | 1 |

- Itemset je skup koji se sastoji od **jednog ili više artikala** (od ukupnog broja artikala)
 - Podrazumijeva se da svaki artikl naveden u itemsetu ima vrijednost 1
 - Veličinu itemseta označavamo s k -itemset, npr. 1-itemset sadrži 1 artikl, 2-itemset sadržava 2 artikla (par artikala), itd.
- Primjeri itemsetova:
 - {Kava}
 - {Čokolada, Mlijeko}
 - {Čaj, Kava, Čokolada}
 - {Čaj, Kava, Čokolada, Mlijeko, Začin}

Pronalazak čestih obrazaca – terminologija

- Apsolutna potpora itemseta A (ili samo: **potpora**, engl. *support*) = broj transakcija koje sadrže itemset A
- Itemset A je **čest** (engl. *frequent*) u skupu podataka ako je njegova potpora (engl. *support*) veća ili jednaka **minimalnom pragu potpore** (engl. *minimum support threshold*), ***min_sup***
- Minimalni prag potpore je hiperparametar algoritma za određeni skup, može se postaviti ovisno o algoritmu na apsolutnu potporu (češći slučaj) ili na relativnu potporu (rjeđi slučaj)
- Relativna potpora: $SUP(A) = \frac{\text{broj transakcija koje sadrže } A}{\text{ukupan broj transakcija}}$

Asocijativna pravila – terminologija

- **Asocijativno pravilo** definira se između **dva itemseta** **A** i **B** kao implikacija: $A \rightarrow B$ u značenju: ako u transakciji imamo itemset **A** onda imamo i itemset **B**
- Relevantnost asocijativnog pravila numerički je definirana pomoću mjere **pouzdanosti** (engl. *confidence*) pravila:
- $$CONF(A \rightarrow B) = \frac{SUP(A \text{ unija } B)}{SUP(A)} = \frac{\text{broj transakcija koje sadrže } A \text{ i } B}{\text{broj transakcija koje sadrže } A}$$
- Pojašnjenje: unija itemsetova je itemset koji se sastoji od svih artikala jednog i drugog itemseta, pravilo je to relevantnije što više transakcija sadrže artikle i iz itemseta **A** i iz **B**, a istovremeno čim manje sadrže artikle samo iz itemseta **A**
- Slično kao i kod potpore, obično se definira **minimalni prag pouzdanosti** (engl. *minimum confidence threshold*, **min_conf**) pravila za uključenje u konačni skup pronađenih obrazaca

Asocijativna pravila – cilj

- **Cilj:** za dani skup transakcija T otkriti sva asocijativna pravila ($A \rightarrow B$) čiji itemsetovi imaju apsolutnu potporu veću ili jednaku minimalnom pragu potpore min_sup i koji imaju pouzdanost veću ili jednaku minimalnom pragu pouzdanosti min_conf
- Računski zahtjevi na pretraživanje potpore itemsetova su veći nego na određivanje pouzdanosti mogućih asocijativnih pravila, stoga je naglasak algoritama na **pronalasku čestih itemsetova**
- Ukupan broj mogućih asocijativnih pravila **eksponencijalno** ovisi samo o broju artikala d u skupu podataka
- Teži se razvoju algoritama koji će čim lakše istražiti taj eksponencijalni prostor asocijativnih pravila

Primjer računanja potpore i pouzdanosti

| ID | Čaj | Kava | Čokolada | Mlijeko | Začin | Krafna |
|----|-----|------|----------|---------|-------|--------|
| T1 | 1 | 0 | 1 | 1 | 1 | 0 |
| T2 | 0 | 1 | 1 | 1 | 0 | 1 |
| T3 | 1 | 1 | 1 | 0 | 0 | 0 |
| T4 | 0 | 1 | 1 | 0 | 0 | 1 |

Potpora (relativna) itemsetova:

- $SUP(\{\text{Čaj}, \text{Kava}\}) = 1/4 = 0.25$
- $SUP(\{\text{Čokolada}\}) = 4/4 = 1$
- $SUP(\{\text{Čaj}, \text{Kava}, \text{Čokolada}\}) = 1/4 = 0.25$
- $SUP(\{\text{Čaj}, \text{Kava}, \text{Čokolada}, \text{Mlijeko}\}) = 0$

Pouzdanost asocijativnih pravila:

- $CONF(\{\text{Čokolada}\} \rightarrow \{\text{Kava}\}) = 3/4 = 0.75$
- $CONF(\{\text{Čaj}\} \rightarrow \{\text{Kava}\}) = 1/2 = 0.5$
- $CONF(\{\text{Kava}, \text{Čokolada}\} \rightarrow \{\text{Krafna}\}) = 2/3 = 0.67$
- $CONF(\{\text{Kava}\} \rightarrow \{\text{Mlijeko}, \text{Krafna}\}) = 1/3 = 0.33$

Algoritmi asocijativnih pravila

Algoritam Apriori

- engl. *Apriori*, R. Agrawal, 1996.
- **Temeljni iscrpni algoritam za učenje asocijativnih pravila** na temelju pronađenih čestih itemsetova, najšire korišten u praksi
- **Prednost:** garantirano pronalazi sve itemsetove koji imaju potporu veću od minimalnog praga potpore i sva pravila koja imaju pouzdanost veću od minimalnog praga pouzdanosti
- **Nedostatak:** u odnosu na ostale algoritme asocijativnih pravila, dosta je spor, jer ne optimira pretraživanje skupa mogućih itemsetova

Agrawal R, Mannila H, Srikant R, Toivonen H, Verkamo AI (1996) Fast discovery of association rules. In: Fayyad UM, Piatetsky-Shapiro G, Smyth P, Uthurusamy R (eds) Advances in knowledge discovery and data mining. AAAI Press, Menlo Park, pp 307–328

Algoritam Apriori – ideja

- Pretraga za itemsetovima počinje od najopćenitijih obrazaca – pojedinačnih artikala, **pretragom u širinu**
- Iterativno se računa **apsolutna potpora itemsetova** kandidata i pohranjuju se za iduću iteraciju **česti itemsetovi**
- U idućoj iteraciji, oni itemsetovi kandidati koji imaju **barem jedan nečesti podskup se ne razmatraju** (takvi itemsetovi ne mogu biti česti)
 - To omogućuje Aprioriju da pronađe relativno učinkovito sve česte itemsetove bez da troši previše vremena na nečeste itemsetove
- Konačno, algoritam **testira sva česta asocijativna pravila** i odabire ona koja su ujedno i pouzdana

Algoritam Apriori – pseudokod

- Ulaz: skup I pojedinačnih artikala, multiskup D svih podskupova od I , min_sup , min_conf
- razina = 1, česti_itemsetovi = {}
- Itemsetovi_kandidati = $\{\{i\} \mid i \in I\}$
- Dok itemsetovi_kandidati $\neq \emptyset$
 - Skeniraj D da se utvrdi apsolutna potpora svih skupova u itemsetovi_kandidati;
 - česti_itemsetovi := česti_itemsetovi $\cup \{C \in \text{itemsetovi_kandidati} \mid \text{aps. potpora}(C) \geq min_sup\}$;
 - razina := razina + 1;
 - itemsetovi_kandidati := $\{A \in D \mid \|A\| = \text{Razina} \text{ i } B \in \text{česti_itemsetovi } \forall B \subset A, \|B\| = \text{Razina} - 1$
i za prijelaz B u A razmatrati samo česte 1-itemsetove};
- Izlaz: česti_itemsetovi

Algoritam Apriori – pseudokod

- Za svaki $F \in \text{česti_itemsetovi}$:
 - Za svaki E pravi podskup od $F, E \neq \emptyset$:
 - Ako je $\frac{\text{Aps.potpota}(F)}{\text{Aps.potpota}(E)} \geq \text{min_conf}$ tada je izlaz **asocijativno pravilo** $E \rightarrow (F \setminus E)$

Algoritam Apriori – primjer

| ID | Čaj | Kava | Čokolada | Mlijeko | Začin | Krafna |
|----|-----|------|----------|---------|-------|--------|
| T1 | 1 | 0 | 1 | 1 | 1 | 0 |
| T2 | 0 | 1 | 1 | 1 | 0 | 1 |
| T3 | 1 | 1 | 1 | 0 | 0 | 0 |
| T4 | 0 | 1 | 1 | 0 | 0 | 1 |

- $min_sup = 2, min_conf = 0.7$

Razina = 1:

- itemsetovi_kandidati (+ potpora) = { {Čaj; 2}, {Kava; 3}, {Čokolada; 4}, {Mlijeko; 2}, {Začin; 1}, {Krafna; 2} }
- česti_itemsetovi = {{Čaj; 2}, {Kava; 3}, {Čokolada; 4}, {Mlijeko; 2}, {Krafna; 2}} // Začin je ispao

Razina = 2:

- itemsetovi_kandidati (+ potpora) = { {Čaj, Kava; 1}, {Čaj, Čokolada; 2}, {Čaj, Mlijeko; 1}, {Čaj, Krafna; 0}, {Kava, Čokolada; 3}, {Kava, Mlijeko; 1}, {Kava, Krafna; 2}, {Čokolada, Mlijeko; 2}, {Čokolada, Krafna; 2}, {Mlijeko, Krafna; 1} } // Začin se ne razmatra
- česti_itemsetovi = {{Čaj; 2}, {Kava; 3}, {Čokolada; 4}, {Mlijeko; 2}, {Krafna; 2}, {Čaj, Čokolada; 2}, {Kava, Čokolada; 3}, {Kava, Krafna; 2}, {Čokolada, Mlijeko; 2}, {Čokolada, Krafna; 2}} // Ispali su svi itemsetovi s potporom manjom od 2

Algoritam Apriori – primjer

| ID | Čaj | Kava | Čokolada | Mlijeko | Začin | Krafna |
|----|-----|------|----------|---------|-------|--------|
| T1 | 1 | 0 | 1 | 1 | 1 | 0 |
| T2 | 0 | 1 | 1 | 1 | 0 | 1 |
| T3 | 1 | 1 | 1 | 0 | 0 | 0 |
| T4 | 0 | 1 | 1 | 0 | 0 | 1 |

Razina = 3: (zadnja razina, nema daljnjih kandidata na razini 4)

- itemsetovi_kandidati (+ potpora) = $\{\{\text{Čaj, Čokolada, Kava; 1}\}, \{\text{Čaj, Čokolada, Mlijeko; 1}\}, \{\text{Čaj, Čokolada, Krafna; 0}\}, \{\text{Kava, Čokolada, Mlijeko; 1}\}, \{\text{Kava, Krafna, Čaj; 0}\}, \{\text{Kava, Čokolada, Krafna; 2}\}, \{\text{Kava, Mlijeko, Krafna; 1}\}, \{\text{Čokolada, Mlijeko, Krafna; 1}\}\}$
- Izlaz: česti_itemsetovi = $\{\{\text{Čaj; 2}\}, \{\text{Kava; 3}\}, \{\text{Čokolada; 4}\}, \{\text{Mlijeko; 2}\}, \{\text{Krafna; 2}\}, \{\text{Čaj, Čokolada; 2}\}, \{\text{Kava, Čokolada; 3}\}, \{\text{Kava, Krafna; 2}\}, \{\text{Čokolada, Mlijeko; 2}\}, \{\text{Čokolada, Krafna; 2}\}, \{\text{Kava, Čokolada, Krafna; 2}\}\}$

Prema pseudokodu, za izgradnju **asocijativnih pravila** razmatraju se samo itemsetovi veličine 2 ili veće:

$\{\text{Čaj, Čokolada}\}, \{\text{Kava, Čokolada}\}, \{\text{Kava, Krafna}\}, \{\text{Čokolada, Mlijeko}\}, \{\text{Čokolada, Krafna}\}, \{\text{Kava, Čokolada, Krafna}\}$

- Npr. za $\{\text{Čaj, Čokolada}\}$, aps. potpora = 2, dok je aps. potpora ($\{\text{Čaj}\}$) = 2, aps. potpora ($\{\text{Čokolada}\}$) = 4 te je onda **$\text{CONF}(\text{Čaj} \rightarrow \text{Čokolada}) = 2 / 2 = 1 \geq 0.7$** , dok je **$\text{CONF}(\text{Čokolada} \rightarrow \text{Čaj}) = 2 / 4 = 0.5 < 0.7$**

Algoritam Apriori – primjer

| ID | Čaj | Kava | Čokolada | Mlijeko | Začin | Krafna |
|----|-----|------|----------|---------|-------|--------|
| T1 | 1 | 0 | 1 | 1 | 1 | 0 |
| T2 | 0 | 1 | 1 | 1 | 0 | 1 |
| T3 | 1 | 1 | 1 | 0 | 0 | 0 |
| T4 | 0 | 1 | 1 | 0 | 0 | 1 |

- Popis svih otkrivenih asocijativnih pravila (s pripadnom pouzdanosti):
 - Čaj -> Čokolada (1)
 - Kava -> Čokolada (1)
 - Čokolada -> Kava (0.75)
 - Krafna -> Kava (1)
 - Mlijeko -> Čokolada (1)
 - Krafna -> Čokolada (1)
 - Krafna -> Kava, Čokolada (1)
- Moguće strategije *offline* dućana
 - Ako su dosad kava, krafna i čokolada stajali blizu, pokušati ih odvojiti pa vidjeti hoće li neki novi artikal ući u čestu kupovinu (riskantno, možda netko prestane kupovati svo troje sada!)
 - Ako su dosad kava, krafna i čokolada stajali daleko, možda ih približiti da se klijentima ubrza kupovina i možda da i drugi klijenti uzmu istu kombinaciju

Algoritam PCY

- *PCY, Park et al., 1997.*
- Algoritam pronalaska čestih itemsetova s manjom vremenskom složenosti od Apriorija
- Razmatra problem kandidatnih itemsetova kod Apriorija – nakon inicijalnog pronalaska čestih 1-itemsetova, potrebno je razmotriti **sve** 2-itemsetove da se vidi koji su česti
 - Neki parovi artikala uopće se ne pojavljuju zajedno u transakcijama, ali ih Apriori svejedno razmatra
 - Složenost je kvadratna, što za veliki broj artikala može predstavljati ozbiljan problem
- Osnovna ideja: koristiti **funkciju raspršenog adresiranja** (engl. *hashing function*) prilikom prvog prolaska kroz skup – kada se otkrivaju česti 1-itemsetovi

Park, Jong Soo & Chen, Ming-Syan & Yu, Philip. (1997). An Effective Hash-Based Algorithm for Mining Association Rules. Proceedings of the 1995 ACM SIGMOD international conference on Management of data ACM SIGMOD. 24. 10.1145/568271.223813

Algoritam PCY

- U prvom prolasku razmatraju se transakcije redom te se:
 - pobrojavaju pojedinačni artikli (kao kod Apriorija) – **broji im se potpora** i
 - izgrađuju parovi **čestih** artikala (2-itemsetovi) u pojedinačnoj transakciji, na njima se primijeni funkcija raspršenog adresiranja i prati se broj pojavljivanja svake pojedinačne raspršene vrijednosti
 - specifičnije, za praćenje broja pojavljivanja vrijednosti parova koristi se **izmijenjena tablica raspršenog adresiranja** (engl. *modified hash-table*)
 - Svaka lokacija u tablici odgovara jednoj raspršenoj vrijednosti i sadrži jedan cijeli broj – koji broji koliko puta se došlo na tu lokaciju – **ne pamte se sami parovi, nego samo broj dolazaka na tu lokaciju u tablici**

Algoritam PCY – primjer

| ID | Čaj | Kava | Čokolada | Mlijeko | Začin | Krafna |
|----|-----|------|----------|---------|-------|--------|
| T1 | 1 | 0 | 1 | 1 | 1 | 0 |
| T2 | 0 | 1 | 1 | 1 | 0 | 1 |
| T3 | 1 | 1 | 1 | 0 | 0 | 0 |
| T4 | 0 | 1 | 1 | 0 | 0 | 1 |

- Neka jednostavna funkcija raspršenog adresiranja razmatra samo prva slova parova artikala abecednim redom kao ključ u tablici i neka je *min_sup* za artikle = 2
- Za T1 imamo česte 1-itemsetove: {Čaj}, {Čokolada}, {Mlijeko}, a ključevi parova su: {Č,Č}, {Č,M}, {Č,M}
- Za T2 imamo česte 1-itemsetove: {Kava}, {Čokolada}, {Mlijeko}, {Krafna}, a ključevi parova su: {Č,K}, {K,M}, {K,K}, {Č,M}, {Č,K}, {K,M}, itd.
- U tablici raspršenog adresiranja, na lokaciji (adresi) gdje se pohranjuje vrijednosti pamte se onda cijeli brojevi pojavljivanja:

{Č,Č}: 2

{Č,M}: 3

{Č,K}: 6

{K,K}: 2

...

Algoritam PCY

- Na početku je postavljen prag minimalne potpore *min_sup* koji itemset treba imati
- Na kraju prvog koraka dalje se razmatraju samo 1-itemsetovi (artikli) koji imaju potporu veću od *min_sup* te se dodatno tablica raspršenog adresiranja transformira
 - Svaka cjelobrojna vrijednost u tablici manja od *min_sup* postavlja se na **0**, a svaka veća ili jednaka *min_sup* na **1**
 - Ovakva transformirana tablica raspršenog adresiranja naziva se **bitmapa**
- Lokacije s potporom **manjom od *min_sup*** (tamo gdje je bitmapa daje vrijednost 0) su jako važne za drugi korak
- U drugom koraku **razmatrat će se samo oni parovi koji se sastoji od čestih 1-itemsetova i koji se u tablici raspršenog adresiranja zabilježeni kao **mogući česti par**** (nisu nužno česti par jer se raspršenim adresiranjem gubi cijela informacija o artiklu)
 - **Svi parovi koji imaju vrijednost u tablici jednaku 0 neće biti razmatrani**, jer tada par **sigurno** nije čest

Algoritam PCY


- Odlučujući faktor u tome je li PCY brži ili ne u odnosu na Apriori je **postotak mogućih parova kandidata koje eliminira tehnika raspršenog adresiranja**
- **Najveći dobitak algoritma je u ranoj fazi – generiranje 2-itemsetova**, za kasnije faze PCY heuristički procjenjuje potrebu za izgradnjom tablice raspršenog adresiranja i ako ustanovi da nema potrebe, radi kao Apriori
- Važan je izbor funkcije raspršenog adresiranja – sve vrijednosti bi trebale imati jednaku vjerojatnost pojavljivanja
- Veličina tablice raspršenog adresiranja je također važna – mora biti dovoljno velika da ne dolazi do čestih kolizija artikala, a ne prevelika jer je to onda isto kao da se razmatraju svi kandidati
- Nedostatak algoritma je često veća potrošnja memorije u odnosu na Apriori, jer se tablica raspršenog adresiranja treba pamtit (ili više tablica s različitim funkcijama raspršivanja, ovisno o implementaciji)

Algoritam FP-growth

- *FP-Growth (od Frequent Pattern Growth)*, J. Han et al., 2000
- Spada u grupu tzv. *Pattern Growth* algoritama koji najprije skeniraju bazu kako ne bi razmatrali sve kandidate
- Algoritam koji koristi **stablatsu strukturu podataka FP-Tree** za sažeti prikaz ulaznih podataka i pronalaženje čestih itemsetova
 - FP-tree se konstruira čitanjem jedne po jedne transakcije i bilježenjem svake transakcije u nekoj grani stabla
- Budući da transakcije često imaju neke zajedničke artikle, njihovi putovi u stablu se dijelom preklapaju i time se ostvaruje sažimanje velike količine podataka
- FP-growth može biti i **nekoliko redova veličine brži od Apriorija** u pronalasku čestih itemsetova
- Za neke probleme koji nemaju dobar faktor sažimanja (engl. *compaction factor*) u stablo, FP-growth ima lošije performanse od Apriorija

J. Han, J. Pei, and Y. Yin. 2000. Mining frequent patterns without candidate generation. SIGMOD Rec. 29, 2 (June 2000), 1–12.
DOI:<https://doi.org/10.1145/335191.335372>

Algoritam FP-growth – primjer

| TID | Artikli u transakciji | 1. korak: | | Artikl | Apsolutna potpora |
|-----|-----------------------|--|---|--------|-------------------|
| 1 | {a,b} | <ul style="list-style-type: none">• Odrediti apsolutnu potporu svih pojedinačnih artikala u transakciji (1-itemsetova)• Sortirati artikle prema apsolutnoj potpori, od više prema nižoj i ukloniti one koji imaju apsolutnu potporu manju od npr. $min_sup = 2$ |  | {a} | 8 |
| 2 | {b,c,d} | | | {b} | 7 |
| 3 | {a,c,d,e} | | | {c} | 6 |
| 4 | {a,d,e} | | | {d} | 5 |
| 5 | {a,b,c} | | | {e} | 3 |
| 6 | {a,b,c,d} | | | | |
| 7 | {a} | | | | |
| 8 | {a,b,c} | | | | |
| 9 | {a,b,d} | | | | |
| 10 | {b,c,e} | | | | |

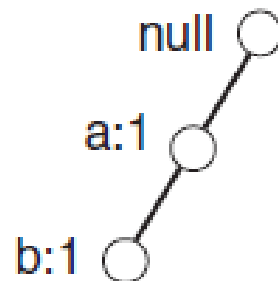
Algoritam FP-growth – primjer

2. korak:

- Čita se transakcija po transakcija, dodaju se artikli u FP-tree i pridružuje im se odgovarajuća vrijednost
- U svakoj transakciji sortiraju se pojedini artikli po apsolutnoj potpori u skupu podataka i tim redoslijedom ih se umeće u stablo
- Osim artikala i potpore, FP-stablo sadrži u čvorovima i pokazivače na već dotad izgrađene čvorove istog artikla, označene crtkanim strelicama
- Pokazivači služe za **optimizaciju pronalaska čestih itemsetova**

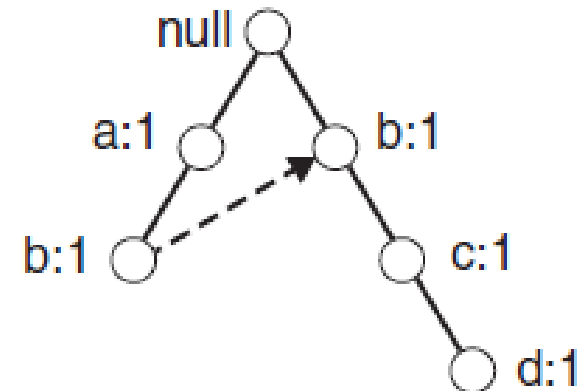


Nakon čitanja TID = 1



Najprije ubaci a, potom b.
Broj pojavljivanja artikala
u stablu je zasad jednak 1

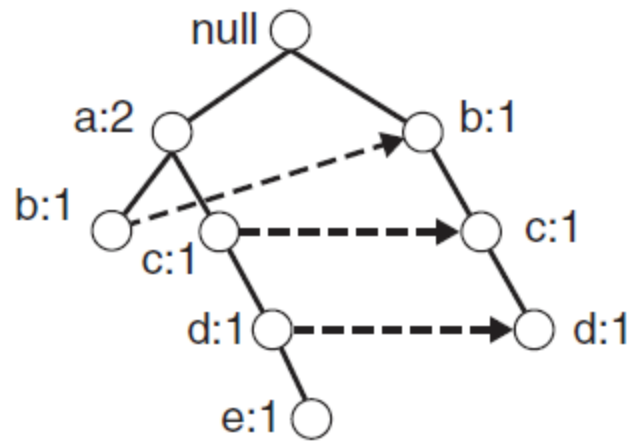
Nakon čitanja TID = 2



Uočava se da se u obje transakcije pojavljuje artikl **b**, no putovi se ne podudaraju jer imaju različite prefikse (**a** za TID=1 i null za TID=2)

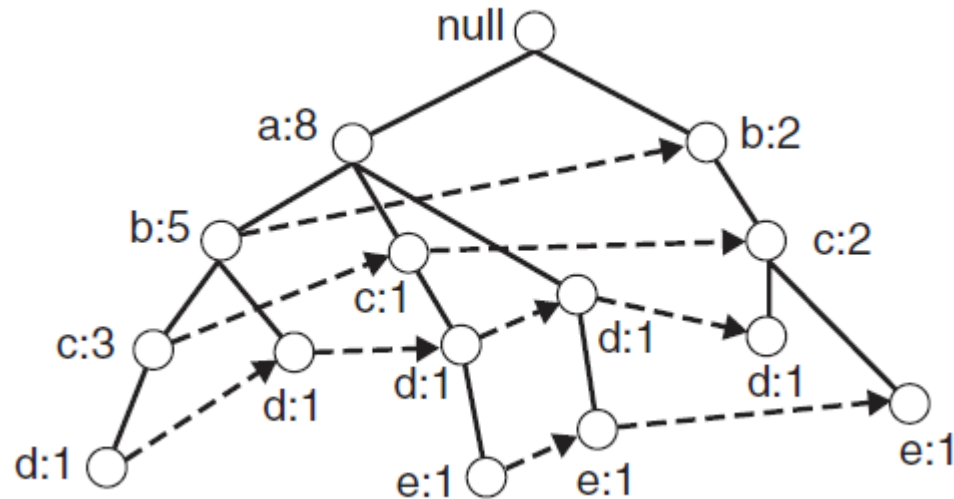
Algoritam FP-growth – primjer

Nakon čitanja TID = 3



Treća transakcija ima preklapanje s prvom pa se potpora za artikl **a** povećala

Nakon čitanja TID = 10



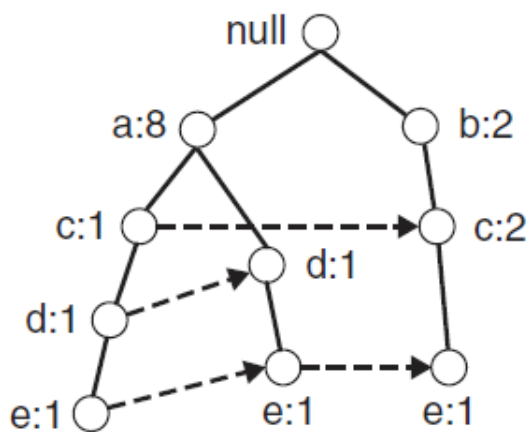
Svaka transakcija u postojećem skupu se preslikala u neki put u stablu; neki putovi dijele neke artikle, a najviše ih dijele artikl **a**

Algoritam FP-growth – veličina stabla

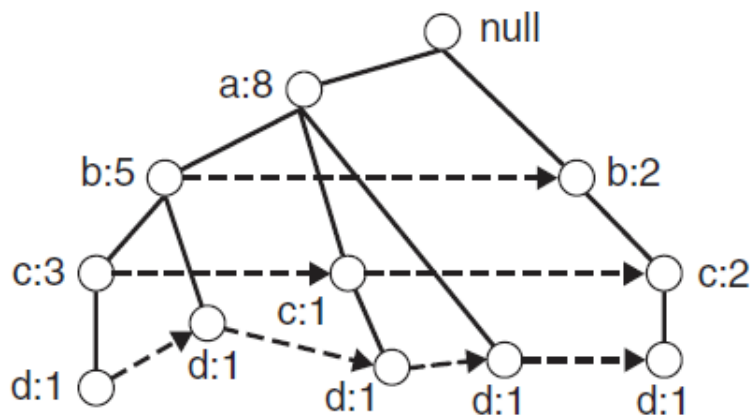
- Veličina stabla ovisi o tome koliko transakcije imaju istih artikala
 - Najbolji slučaj je kada sve transakcije imaju iste artikle – veličina stabla onda je samo jedna grana
 - Najgori slučaj je kada sve transakcije imaju sve različite artikle – veličina stabla je onda ista kao i tablica (+ dodatne informacije o pokazivačima na iduće artikle i informacija o potpori)
 - U prosječnom slučaju stablo značajno sažima skup podataka i omogućuje lakše pronalaženje čestih itemsetova
- Veličina stabla ovisi i o poretku na temelju kojeg se gradi
 - Može se graditi od najveće prema najmanjoj potpori (uobičajeno) ili obratno (neki puta je bolje)

Algoritam FP-growth – pronalazak čestih itemsetova – ideja

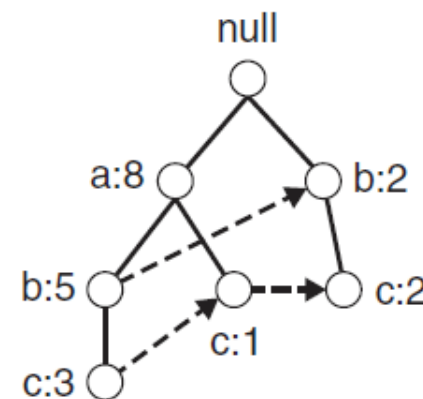
- FP-stablo pretražuje se od **dna prema vrhu** kako bi se pronašli česti itemsetovi
- Pretraživanje se dijeli na više **potproblema**, a svaki potproblem uključuje pronalazak čestih itemsetova koji završavaju sa svim čestim artiklima koji postoje u skupu (u našem primjeru s: {e}, {d}, {c}, {b}, {a})



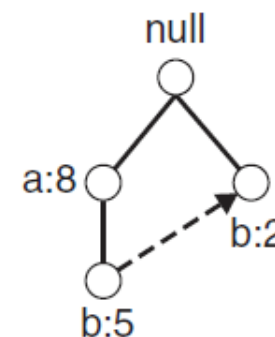
Putovi koji završavaju
s artiklom {e}
(prefiksni putovi)



Putovi koji
završavaju s
artiklom {d}



Putovi koji
završavaju s
artiklom {c}



Putovi koji
završavaju s
artiklom {b}



Putovi koji
završavaju s
artiklom {a}

Algoritam FP-growth – pronalazak čestih itemsetova – ideja

- Česti itemsetovi koji završavaju s pojedinačnim artiklima pronalaze se strategijom **podijeli-pa-vladaj**
- Npr. neka se želi pronaći sve česte itemsetove koji završavaju s {e}
 - U tom slučaju prvo se utvrdi je li 1-itemset {e} česti itemset – to se radi tako da se pobroje potpore čvorova {e} na krajevima **svih putova koji ga sadržavaju** (tzv. prefiksni putovi) u stablu
 - Ako jest, onda se razmatraju sljedeći itemsetovi koji redom završavaju u {e}, a to bi u čitavom skupu bili {d,e}, {c,e}, {b,e} i {a,e} te se za svakog od njih utvrđuje je li čest redukcijom stabla – tzv. **uvjetno stablo**
 - Uklanjanju se čvorovi {e} i odgovarajuće potpore koje dovode do {e} sve dok se ne dođe do odgovarajućeg drugog čvora koji se ispituje (npr. do čvorova {c})
 - U tom slučaju pokazuje se da {b,e} ne bi bio česti itemset, a ostali {d,e}, {c,e}, {a,e} bi bili
 - Iterativno se razmatraju sljedeće razine artikala za one itemsetove koji su bili česti u prethodnom koraku, sve do korijena stabla te se time pronalaze svi česti itemsetovi koji završavaju s {e}

Algoritam FP-growth – pronalazak čestih itemsetova

- Česti itemsetovi (podskupovi) su međusobno različiti jer imaju različiti sufiks:

| Sufiks | Česti podskupovi |
|--------|---|
| e | {e}, {d, e}, {a, d, e}, {c, e}, {a, e} |
| d | {d}, {c, d}, {b, c, d}, {a, c, d}, {b, d}, {a, b, d}, {a,d} |
| c | {c}, {b, c}, {a, b, c}, {a, c} |
| b | {b}, {a, b} |
| a | {a} |

- Generiranje **asocijativnih pravila** iz čestih itemsetova obično koristi minimalnu pouzdanost koju pravilo treba zadovoljiti te se razmatraju sve kombinacije antecedenta i konsekvensa pravila, slično kao i kod algoritma Apriori

Ostali algoritmi za pronalazak čestih itemsetova

- Postoji veliki broj algoritama za pronalazak čestih itemsetova
- Glavne razlike između njih su u:
 - načinu pretrage prostora itemsetova
 - u širinu – generiranje kandidata, npr. Apriori, Apriori – TID, Eclat
 - u dubinu – *pattern growth*, npr. FP-growth, H-Mine, LCM
 - početnom predstavljanju baze podataka
 - horizontalan – navodi se koji artikli postoje u svakoj transakciji – većina algoritama
 - vertikalni – navodi se za svaki artikl u kojim transakcijama postoji – npr. Apriori – TID, Eclat
 - strukturama podataka i heuristikama koje se koriste
 - svaki algoritam ima neki svoj pristup

Detaljnije u: Philippe Fournier-Viger, Jerry Chun-Wei Lin, Bay Vo, Tin Truong Chi, Ji Zhang, Hoai Bac Le. A Survey of Itemset Mining. WIREs Data Mining and Knowledge Discovery, Vol. 7, Issue 4, 2017

Algoritam EFIM

- *EFIM (od EFicient high-utility Itemset Mining), Zida et al., 2015*
- Spada u grupu algoritama za **otkrivanje visokokorisnih itemsetova** (engl. *high-utility itemset mining*, **HUIM**), smatra se *state-of-the-art* algoritmom
- Za razliku od transakcija, koje imaju samo oznaku je li artikl kupljen ili ne, HUIM baze sadržavaju uz svaki kupljeni artikl i **informaciju o težini**, koja govori koliki je jedinični profit za dotični artikl (koliko se profita dobiva za svaki takav prodani pojedinačni artikl)
 - Zadatak HUIM-a je pronaći sve itemsetove koji imaju **korisnost** (ovisnu o težini) **veću od minimalno zadane**
 - Problem je što mjera korisnosti nije monotona a nije niti antimonotona – itemset može imati nadskup ili podskup s višom, jednakom ili nižom korisnosti – pa je potrebno pažljivo pristupiti pretrazi prostora stanja
- Glavna značajka: **EFIM koristi nekoliko optimizacija (heuristika) za pretraživanje prostora itemsetova** koje mu omogućuju bitno bolje rezultate od konkurentskih algoritama

Zida, S., Fournier-Viger, P, Lin, JC.W, Wu, CW, Tseng, VS. EFIM: A Highly Efficient Algorithm for High-Utility Itemset Mining. In: Proc. 14th Mexican Intern. Conf. Artificial Intelligence, Cuernavaca, Mexico, 25-31 October, 2015:530–546.

Algoritam EFIM – definicija problema HUIM

- Korisnost artikla i u transakciji T_c označava se s: $u(i, T_c)$ i iznosi $p(i) \times q(i, T_c)$ ako je $i \in T_c$, gdje je $q(i, T_c)$ **unutarnja korisnost** artikla i u transakciji T_c (tzv. kupljena količina), a $p(i)$ je **vanjska korisnost** artikla i (tzv. jedinična cijena, profit)
- Korisnost nekog itemseta $X \subseteq T_c$ u transakciji T_c iznosi $u(X, T_c) = \sum_{i \in X} u(i, T_c)$
- Korisnost itemseta X u svim transakcijama iznosi $u(X) = \sum_{T_c \in g(X)} u(X, T_c)$, gdje je $g(X)$ skup transakcija koje sadržavaju X
- Itemset X je visokokoristan (engl. *high-utility itemset*) ako je $u(X) \geq minutil$, a inače je niskokoristan
- **Problem HUIM:** potrebno je pronaći sve visokokorisne itemsetove u skupu podataka
 - Problem je puno teži od pronalaska svih čestih itemsetova
 - Rješenje ovog problema je vrlo korisno za trgovačka poduzeća

Algoritam EFIM – primjer

| TID | Transaction |
|-------|--|
| T_1 | $(a, 1)(c, 1)(d, 1)$ |
| T_2 | $(a, 2)(c, 6)(e, 2)(g, 5)$ |
| T_3 | $(a, 1)(b, 2)(c, 1)(d, 6)(e, 1)(f, 5)$ |
| T_4 | $(b, 4)(c, 3)(d, 3)(e, 1)$ |
| T_5 | $(b, 2)(c, 2)(e, 1)(g, 2)$ |

Transakcije s
unutarnjim
korisnostima
artikala

| Item | a | b | c | d | e | f | g |
|--------|-----|-----|-----|-----|-----|-----|-----|
| Profit | 5 | 2 | 1 | 2 | 3 | 1 | 1 |

Vanjska
korisnost
artikala

- Korisnost artikla a u transakciji T_2 iznosi $u(a, T_2) = 2 \times 5 = 10$
- Korisnost itemseta $\{a, c\}$ u transakciji T_2 iznosi $u(\{a, c\}, T_2) = u(a, T_2) + u(c, T_2) = 16$
- Korisnost itemseta $\{a, c\}$ u bazi podataka T je $u(\{a, c\}) = u(\{a, c\}, T_1) + u(\{a, c\}, T_2) + u(\{a, c\}, T_3) = u(a, T_1) + u(c, T_1) + u(a, T_2) + u(c, T_2) + u(a, T_3) + u(c, T_3) = 28$
- Ako bi *minutil* bio jednak 30, samo bi sljedeći itemsetovi bili visokokorisni (provjerite!): $\{b, d\}$, $\{a, c, e\}$, $\{b, c, d\}$, $\{b, c, e\}$, $\{b, d, e\}$, $\{b, c, d, e\}$, $\{a, b, c, d, e, f\}$.

Algoritam EFIM – TWU

- Budući da funkcija korisnosti nije monotona niti antimonotona, potrebno ju je **precijeniti** (engl. *overestimate*) kako bismo bili sigurni da tijekom pretrage pronalazimo česte itemsetove
 - Najčešći pristup je **korištenje itemseta X utežanog transakcijom** kao precijenjene mjere korisnosti itemseta X
 - Definira se **korisnost transakcije** (engl. *transaction utility*): $TU(T_c) = \sum_{x \in T_c} u(x, T_c)$
 - **Korištenje utežano transakcijom** (engl. *transaction-weighted utilization*, **TWU**) itemseta X je:

$$TWU(X) = \sum_{T_c \in g(X)} TU(T_c)$$

- TU transakcija $T_1 \dots T_5$ iznosi redom 8, 27, 30, 20 i 11. $TWU(\{a\}) = TU(T_1) + TU(T_2) + TU(T_3) = 65$
- Za svaki itemset X može se pokazati da vrijedi $TWU(X) \geq u(Y)$, $\forall Y$ nadskup od X , odnosno TWU je **gornja granica** korisnosti itemseta X i njegovih nadskupova – **itemsetovi višeg stupnja imaju manji TWU**

Algoritam EFIM – strategije podrezivanja prostora stanja

- **1. strategija podrezivanja prostora stanja:** za svaki itemset X , ako je $TWU(X) < minutil$, onda je X itemset niske korisnosti, zajedno sa svim svojim nadskupovima te ih je moguće zanemariti
- Neka je $s „>”$ označen potpuni poredak artikala u skupu artikala (npr. abecedni poredak)
- **Preostala korisnost** (engl. *remaining utility*) za itemset X u transakciji T_c je definirana kao $reutil(X, T_c) = \sum_{i \in T_c, i \succ x, \forall x \in X} u(i, T_c)$
- **Lista korisnosti** (engl. *utility-list*) itemseta X u skupu podataka D je skup trojki takav da svaka transakcija T_c koja sadržava X ima **trojku** $(T_c, iutil, reutil)$, gdje je $iutil$ korisnost od X u T_c ($iutil = u(X, T_c)$), dok je $reutil$ preostala korisnost $reutil(X, T_c)$
- U našem primjeru, neka je potpuni poredak artikala $g > f > e > d > c > b > a$. Lista korisnosti od itemseta $\{a, e\}$ je $\{(T_2, 16, 5), (T_3, 8, 5)\}$
- Gornja granica preostale korisnosti itemseta X u skupu podataka D je $reu(X) = u(X) + reutil(X)$ i može se dobiti sumirajući $iutil$ i $reutil$ vrijednosti u listi korisnosti itemseta X

Algoritam EFIM – strategije podrezivanja prostora stanja

- **2. strategija podrezivanja prostora stanja:** ako je $reu(X) < minutil$ onda je X itemset niske korisnosti, kao i svi itemsetovi dobiveni dodavanjem artikla $i > x, \forall x \in X$ u itemset X
- Neka je X itemset i z artikl koji može proširiti X prema pretrazi u dubinu: $z \in E(X), E(X) = \{z \mid z \in I \text{ i } z > x, \forall x \in X\}$. Korisnost podstabla od z u odnosu na X je:

$$su(X, z) = \sum_{T \in g(X \cup \{z\})} [u(X, T) + u(z, T) + \sum_{i \in T \text{ i } i \in E(X \cup \{z\})} u(i, T)]$$

- U našem primjeru neka je itemset $X = \{a\}$. Razmotrimo artikal $z = \{c\}$ i proširenje na $\{a, c\}$. Transakcije koje sadrže $\{a, c\}$ su T_1, T_2, T_3 . Imamo $su(X, c) = (5 + 1 + 2) + (10 + 6 + 11) + (5 + 1 + 20) = 61$
- **3. strategija podrezivanja prostora stanja:** Za itemset X i artikl $z \in E(X)$, ako je $su(X, z) < minutil$ onda je proširenje s artiklom z i svim daljnjim proširenjima niskokorisno te se podstablo može podrezati

Algoritam EFIM – strategije podrezivanja prostora stanja

- Neka je X itemset i z artikl koji može proširiti X prema pretrazi u dubinu: $z \in E(X)$, $E(X) = \{z \mid z \in I \text{ i } z > x, \forall x \in X\}$.
- Lokalna korisnost od z u odnosu na X je:

$$lu(X, z) = \sum_{T \in g(X \cup \{z\})} [u(X, T) + reutil(X, T)]$$

- U našem primjeru neka je itemset $X = \{a\}$. Razmotrimo artikl $z = \{c\}$ i proširenje na $\{a, c\}$. Imamo $lu(X, c) = (8 + 27 + 30) = 65$
- **4. strategija podrezivanja prostora stanja:** Za itemset X i artikl $z \in E(X)$, ako je $lu(X, z) < minutil$ onda su sva proširenja od X s artiklom z niskokorisna. Drugim riječima, artikl z može se ignorirati kad se istražuju podstabla od X

Algoritam EFIM – brzo računanje korisnosti

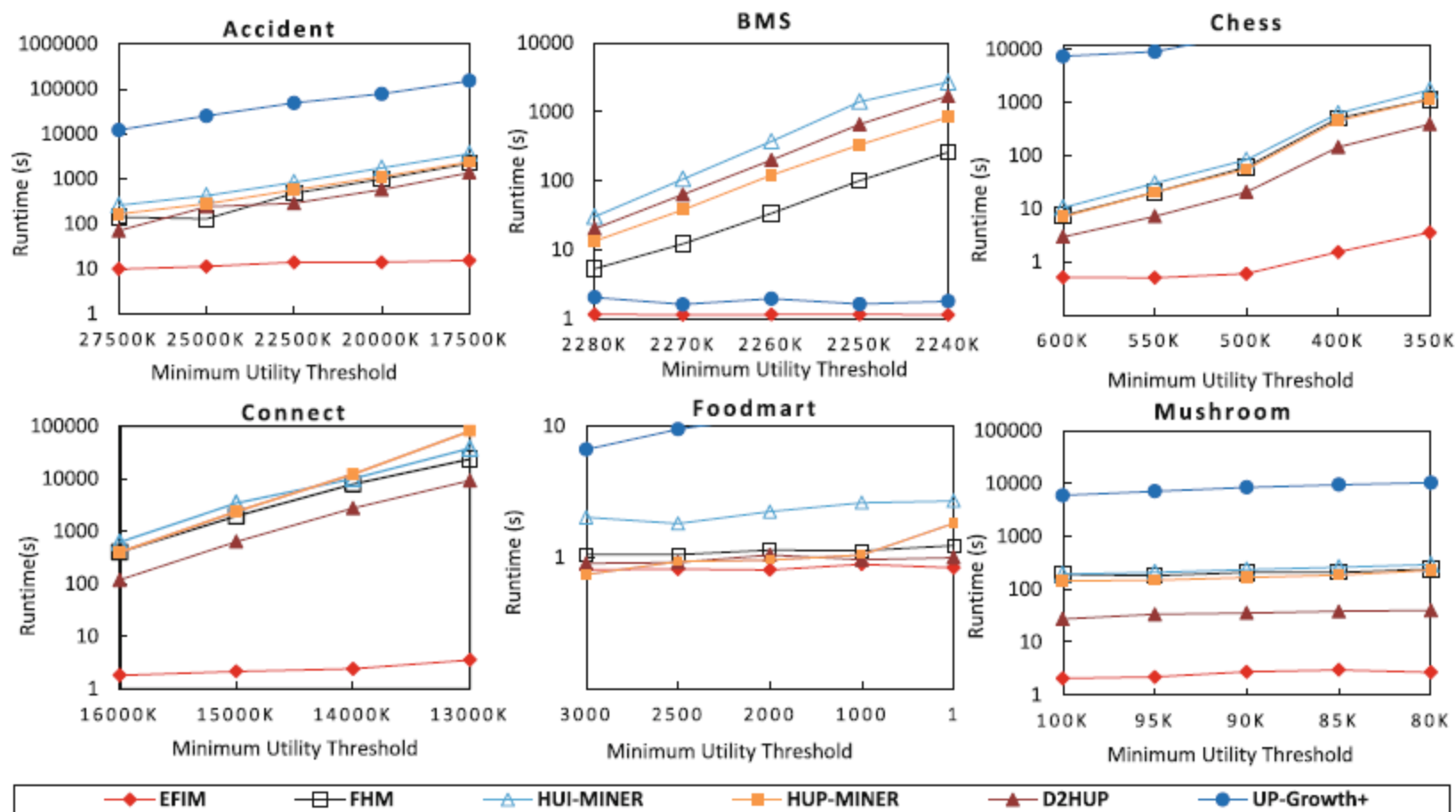
- **Polje košara korisnosti U** za bazu transakcija D
 - **Polje** dimenzije $|I|$, gdje je I skup artikala u D , s elementima $U[z]$, $\forall z \in I$, a svaki element naziva se **košara korisnosti** (engl. *utility-bin*) i služi za pohranu vrijednosti korisnosti artikala (cijeli broj, inicijalno jednak 0)
- U se može iskoristiti za brzo računanje sljedećih **gornjih granica korisnosti** (u vremenu $O(n)$):
 - **Računanje TWU svih artikala**: za svaku transakciju T u bazi, $U[z]$ se revidira za svaki artikl $z \in T$ na način da je $U[z] := U[z] + TU(T)$, na početku je $U[z] = 0$. Po prolasku kroz bazu, $\forall k \in I$, $U[k] = TWU(k)$
 - **Računanje korisnosti podstabla u odnosu na itemset X** : za svaku transakciju T u bazi, $U[z]$ se revidira za svaki artikl $z \in T \cap E(X)$ tako da $U[z] := U[z] + u(X, T) + u(z, T) + \sum_{i \in T \text{ i } i >_z} u(i, T)$. Po prolasku kroz bazu, $\forall k \in I$, $U[k] = su(X, k)$
 - **Računanje lokalne korisnosti u odnosu na itemset X** : za svaku transakciju T u bazi, $U[z]$ se revidira za svaki artikl $z \in T \cap E(X)$ na način da $U[z] := U[z] + u(X, T) + reutil(X, T)$. Po prolasku kroz bazu, $\forall k \in I$, $U[k] = lu(X, k)$

Algoritam EFIM – opis algoritma

- Sam algoritam ne raspisujemo u detalje, jer ima niz optimizacija
- Ukratko, razmatraju se proširenja osnovnog artikla (sortiranih prema TWU) koja su lokalno korisna ($lu > minutil$) i koja su korisna u smislu podstabla ($su > minutil$) počevši od praznog skupa artikala pretragom u dubinu
- Iz koraka u korak u dubinu, iz baze se uklanjaju oni artikli nisu lokalno korisni, a baza transakcija se preslaguje i smanjuje (transakcije se spajaju) dok se ne pronađu svi korisni itemsetovi

Algoritam EFIM – usporedba s drugim HUIM algoritmima

- EFIM prolazi kroz bazu u jednoj fazi, bez generiranja itemsetova kandidata, pretraživanjem u dubinu za razliku od većine ostalih algoritama
- Zbog niza optimizacija, daje bolje rezultate od ostalih algoritama



Zida, S., Fournier-Viger, P, Lin, JC.W, Wu, CW, Tseng, VS. EFIM: A Highly Efficient Algorithm for High-Utility Itemset Mining. In: Proc. 14th Mexican Intern. Conf. Artificial Intelligence, Cuernavaca, Mexico, 25-31 October, 2015:530–546.

Primjena asocijativnih pravila

Sustavi preporučivanja

- Sustavi za preporučivanje sadržaja (npr. namirnice, knjige, glazba, video) koji najčešće razvijaju tvrtke u svrhu povećanja prometa – obično vrlo veliki skupovi podataka
- Glavne vrste sustava preporučivanja
 - **Suradničko filtriranje** (engl. *collaborative filtering*) – predviđanje interesa korisnika na temelju informacija o preferencama tog korisnika i informacija o preferencama drugih korisnika
 - **Filtriranje zasnovano na sadržaju** (engl. *content-based filtering*) – sadržaj je opisan određenim karakteristikama te se predviđanje interesa temelji na tom opisu i preferencama u profilu korisnika
 - **Sustavi preporučivanja zasnovani na znanju** (engl. *knowledge-based recommender system*) – slično kao filtriranje zasnovano na sadržaju, ali su zahtjevi i ograničenja korisnika jasno zadani
 - **Sustavi preporučivanja zasnovani na dubokom učenju** (engl. *deep learning-based recommender system*) – korisno za slike, tekst, audio, video, društvene mreže
 - **Hibridni sustavi** – kombiniranje više pristupa

C. C. Aggarwal, Recommender Systems, Springer Nature Switzerland AG, 2016, <https://doi.org/10.1007/978-3-319-29659-3>

Sustavi preporučivanja

- Asocijativna pravila se najčešće koriste za **suradničko filteriranje zasnovano na artiklima** (engl. *item-based collaborative filtering*)
- Postupak
 - Prikupljen je povijesni skup podataka korisničkih transakcija s kupljenim artiklima
 - Iskoriste se algoritmi za asocijativna pravila da se izgrade relevantna pravila (min_sup, min_conf)
 - Sustav preporuča određenom korisniku konsekvence pravila za one antedecense koje je odabrao (ili kupio)
 - Artikli koji su preporučeni obično su rangirani prema nekom kriteriju, kao što je popularnost kod drugih korisnika
- Nedostatci
 - **Problem „hladnog starta”** (engl. *cold start*) – kako krenuti s preporučivanjem ako još nema podataka u sustavu ili je neki artikl nov?
 - Što ako preference korisnika ne odgovaraju uobičajenim preferencama ostalih korisnika (npr. čovjek ne ide u tjednu kupovinu u dućan kao većina, nego mu samo nedostaju neke namirnice)?

Asocijativna pravila za predobradu podataka

- Algoritmi asocijativnih pravila mogu se koristiti kao **metoda predobrade** za brzi pronalazak razumno velikog i korisnog skupa čestih obrazaca u velikim skupovima podataka
- U idućem koraku, dobiveni skup obrazaca može se iskoristiti za različite primjene strojnog učenja s jasnim tumačenjem
 - Klasifikacija s ciljem maksimizacije točnosti – postupak **CBA** (engl. *Classification Based on Associations*)
 - B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In KDD, 1998.
 - Asocijativna pravila za odabir značajki – postupak koristi grupiranje nad mrežom asocijativnih pravila **ARN** (engl. association rule network) za odabir značajki
 - S. Chawla. Feature Selection, Association Rules Network and Theory Building, JMLR: Workshop and Conf. Proceedings 10: 14-21, 2010
 - Induktivna pravila s ciljem maksimizacije točnosti i mogućnosti tumačenja – postupak **IDS** (engl. *Interpretable Decision Sets*)
 - H. Lakkaraju, S. H. Bach, and J. Leskovec. 2016. Interpretable Decision Sets: A Joint Framework for Description and Prediction. In KDD, 2016. 1675–1684. <https://doi.org/10.1145/2939672.2939874>

Zaključak

- Pronalazak čestih itemsetova i generiranja asocijativnih pravila vrlo su česti problemi koji se rješavaju u analizi ponašanja klijenata
- Apriori je temeljni i najpoznatiji algoritam za pronalazak čestih itemsetova i generiranje asocijativnih pravila koji se može poboljšavati na različite načine
- FP-growth je brzi algoritam za pronalazak čestih itemsetova temeljen na strukturi stabla
- EFIM je suvremeni i brzi algoritam koji koristi niz optimizacija kako bi riješio problem pronalaska visokokorisnih itemsetova
- Asocijativna pravila se primjenjuju za razne svrhe