Poglavje 9

Povezovalna pravila

Danes vse prodajalne, tako spletne kot te, kamor se moramo sprehoditi ali pa zapeljati z nekim prevoznim sredstvom, beležijo podatke o nakupih. Te, ki tega ne počno, so že propadle ali pa so žal na poti propada. Recimo, prodajalna prehrambenih izdelkov. Oglejmo si en zelo majhen vzorec nakupov (tabela 9.1), kjer vsaka vrstica prikazuje, kaj je bilo v nakupovalni košarici. V tem poglavju bomo zadeve zelo poenostavili: kupci bodo ostali anonimni, čas nakupa nas ne bo zanimal, prav tako ne bomo beležili števila nakupljenih stvari. Zanimali nas bodo samo tipi nakupljenih stvari v nakupovalnih košaricah (na primer, mleko, ne pa tri litre mleka, in ne pol štruce kruha).

Nakupovalne košarice smo označili s t_i za i-to nakupovalno košarico. Bolj strokovno vsaki vrstici v tabeli 9.1 pravimo tudi *transakcija*. Naj bo množica transakcij, katero bi želeli preučiti, $\mathcal{T} = \{t_1, t_2, \ldots t_N\}$. Množico stvari, ki lahko nastopajo v teh transakcijah, označimo z $\mathcal{I} = \{i_1, i_2, \ldots i_M\}$.

Tabela 9.1: Podatki o nakupovalnih košaricah.

nakup	stvari v košarici	poenostavljen zapis
t_1	mleko, kruh	MK
t_2	mleko, sir, zelenjava	MSZ
t_3	kruh, sir, zelenjava	KSZ
t_4	mleko, kruh, sir, zelenjava	MKSZ
t_5	mleko, kruh, sir	MKS
t_6	kruh, sir	KS
t_7	sir	S
t_8	kruh, zelenjava, sir	KZS
t_9	sir, zelenjava	SZ
t_{10}	kruh, sir, zelenjava	KSZ

9.1 Nabori in podnabori

Terkam stvari, kot so na primer {kruh, sir} ali pa {mleko, sir, zelenjava} pravimo *nabor*. Za par naborov X in Y pravimo, da je X podnabor nabora Y, če je vsaka stvar iz nabora X vsebovana v naboru Y. V tem primeru zapišemo, da velja $X \subseteq Y$. Nabor {kruh, sir} je na primer podnabor nabora {kruh, mleko, sir}. Številnost stvari v naboru X označimo $|X|$. V naboru $X = \{\text{kruh}, \text{mleko}, \text{sir}\}$ so tri stvari, $|X| = 3$. Nabor X s številnostjo $|X|$ ima $2^{|X|}$ možnih podnaborov, med katerimi je tudi prazen podnabor $\emptyset$.

9.2 Podpora in pogosti nabori

S $\sigma(X)$ označimo število transakcij, ki vsebuje podnabor X. Podnabor $X = \{\text{mleko}, \text{kruh}\}$ je vsebovan v transakcijah t_1, t_4 in t_5. Število podprtih transakcij za nabor X je zato enako tri, oziroma $\sigma(X) = 3$. Število $\sigma(X)$ formalno določimo s sledečim zapisom:

$$\sigma(X) = |\{t_i | X \subseteq t_i, t_i \in \mathcal{T}\}| \tag{9.1}$$

Iz števila podprtih transakcij za nabor X lahko izračunamo delež podprtih transakcij, torej delež transakcij, ki vsebujejo nabor X:

$$s(X) = \frac{\sigma(X)}{|\mathcal{T}|} = \frac{\sigma(X)}{N} \tag{9.2}$$

Zanima nas, kateri nabori so taki da za njih velja

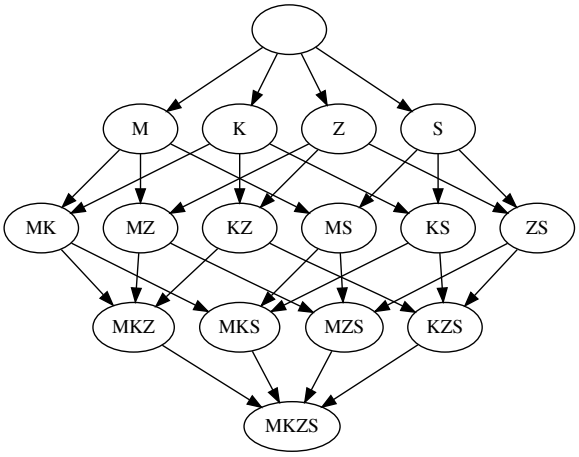
$$s(X) \geq \text{minsupp} \tag{9.3}$$

kjer je minsupp minimalna podpora oziroma parameter, ki ga določi uporabnik. Nabore, kjer velja zgornji izraz, imenujemo *pogosti nabori*.

9.3 Algoritem Apriori

Za iskanje vseh pogostih naborov v množici transakcij bomo razvili algoritem Apriori, ki sta ga sicer predlagala Agrawal in Srikant leta 1994. A preden zapišemo algoritem, razmislimo o nekaj lastnostih prostora vseh možnih naborov. Ti so za naš primer grafično prikazani v mreži naborov (slika 9.1), kjer so nabori določenega nivoja povezani z nabori prejšnjega nivoja, če so slednji njihovi podnabori. Tako je vozlišče MKZ povezano z vozlišči MK, MZ in KZ, a ne z vozliščem MS, saj MS ni podnabor MKZ.

Recimo, da iščemo pogoste nabore s podporo vsaj 0.6 (minsupp = 0.6). Najprej za vsak nabor v mreži naborov določimo število transakcij, v katerih je ta nabor prisoten. Za naš primer tako označeno mrežo prikazuje slika 9.2. Pogosti nabori morajo biti vsebovani vsaj v



Slika 9.1: Mreža vseh možnih naborov za množico stvari mleko (M), kruh (K), zelenjava (Z) in sir (S). V korenu grafa je prazen nabor.

šestih transakcijah. Iz mreže je razvidno, da to velja za šest naborov, med katerimi je eden prazen.

Iz mreže naborov lahko razberemo nekaj zanimivih zakonitosti. Recimo, nabor M je nepogost. Zaradi tega so nepogosti prav vsi nabori z mlekom, torej vsi nabori, katerih podnabor je M. To je razumljivo, saj vsi ti nabori zahtevajo, da je njhov podnabor tudi M, ta pa je vsebovan v pramajhnem številu transakcij, da bi bili nabori, ki ga vsebujejo, pogosti.

Prav tako razberemo, da podpora nabora ne more presegati podpore kateregakoli njegovega podnabora. Primer: KZS je vsebovan v štirih transakcijah, in to število ne sme presegati števila transakcij, v katerih so vsebovani vsi njegovi možni podnabori, torej tudi ti iz prejšnjega nivoja (KZ, KS, ZS).

Zgornja zapažanja zapišimo v obliki teoremov.

Teorem 1 *Če je nabor pogost, so pogoste tudi vse njegove podmnožice:*

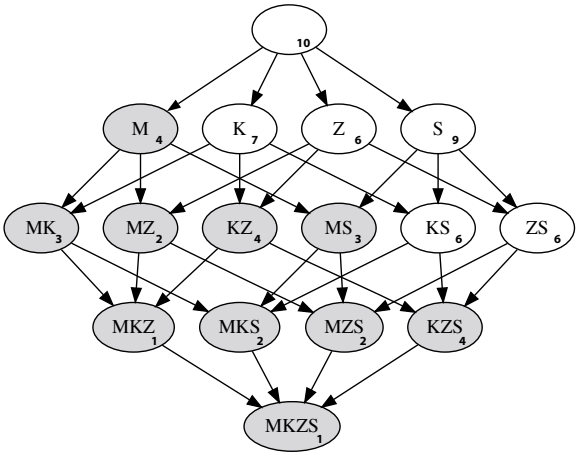
$$s(X) \geq \text{minsupp} \implies s(Y) \geq \text{minsupp}, Y \subseteq X$$

Teorem 2 *Če je nabor nepogost, so nepogosti tudi vsi nabori, ki ga vsebujejo:*

$$s(X) < \text{minsupp} \implies s(Y) < \text{minsupp}, X \subseteq Y$$

Teorem 3 *Podpora nabora nikoli ne presega podpore njegove podmnožnice (antimonotonost, $\mathcal{L} = 2^I$ je močnostna množica, torej množica vseh možnih naborov):*

$$\forall X, Y \in \mathcal{L} : X \subseteq Y \implies s(Y) \leq s(X)$$



Slika 9.2: Mreža pogostih naborov (naborov v praznih vozliščih) za minsupp = 0.6 oziroma za vse nabore, katerih število podpornih transakcij je vsaj 6 ($6 = 0.6 \times 10$). Vozlišča, ki predstavljajo nepogoste nabore, so osenčena.

Iz zgornjega intuitivno sledi algoritem Apriori. Ker nas prazna množica ne zanima, pričnemo lahko s pogostimi nabori prvega nivoja, torej nabori, ki vsebujejo natanko eno stvar (te imenujemo tudi *pogosti 1-nabori*). Od tu dalje s kombinacijo pogostih naborov generiramo *pogoste k-nabore*, kjer k iterativno povečujemo. Ustavimo se, ko na k-tem nivoju ni več pogostih naborov, saj bi to pomenilo, da teh tudi ni v vseh višjih nivojih.

Vhod: množica transakcij in minimalna podpora minsupp
Izhod: množica pogostih naborov

$k = 1$
$F_k = \{i | i \in I \wedge \sigma(i) \geq N \times \text{minsupp}\}$
ponavljaj
  $k \leftarrow k + 1$
  $C_k = \text{kandidati}(F_{k-1})$
  izračunaj podporo naborov v C_k
  $F_k = \{c | c \in C_k \wedge \sigma(c) \geq N \times \text{minsupp}\}$
dokler $F_k = \emptyset$
vrni $\cup_k F_k$

Algoritem Apriori uporablja funkcijo kandidati, ki na podlagi že odkritih pogostih naborov tvori pogoste k-nabore. S podrobnostmi implementacije te funkcije se ne bomo ukvarjali,

omenimo pa samo, da so za hitrost celotnega algoritma precej pomembne. Namreč, izogniti se moramo tvorjenju enakih kandidatov. Recimo, nabor KS lahko tvorimo iz pogostega nabora K in stvari S, ali iz pogostega nabora S in stvari K. Tu omenimo samo nekaj možnosti, ki jih lahko pri tvorjenju kandidatov lahko uporabimo:

- Kandidate za F_k tvorimo iz kombinacije pogostih naborov $F_{k-1} \times F_1$. Problem podvojenih kandidatov rešimo z leksikografsko ureditvijo naborov (na primer MKZ je z MZS, saj je K leksikografsko pred Z).

- Kandidate za F_k tvorimo iz kombinacije pogostih naborov $F_{k-1} \times F_1$, kjer nabora združimo le, če sta različna samo v zadnjem elementu. Tudi tu problem podvojenih kandidatov rešujemo z leksikografsko ureditvijo naborov.

9.4 Povezovalna pravila

Pravila tipa

$$X \rightarrow Y$$

kjer sta X in Y nabora stvari, imenujemo *povezovalna pravilna*. Zahtevamo, da je presečna množica obeh naborov prazna, torej $X \cap Y = \emptyset$. Primer takega pravila je recimo

$$\text{kruh} \rightarrow \text{mleko, sir}$$

pravilo pa pravi, da če bo v košarici kruh, pričakujemo, da bosta tam tudi mleko in sir. Še en primer povezovalnega pravila je

$$\text{mleko, sir} \rightarrow \text{zelenjava}$$

ki pravi, da če bo v košarici mleko, pričakujemo, da bosta tam tudi sir in zelenjava.

Pravila so lahko seveda bolj ali manj v soglasju z učno množico transakcij. Povezovalna pravila ocenjujemo z merami, med katerima sta najbolj znani podpora in zaupanje. *Podporo* že poznamo, poroča pa o delež transakcij, kjer najdemo vse stvari iz povezovalnega pravila:

$$s(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{N} \tag{9.4}$$

Podpora torej meri pogostost pravila v množici transakcij.

Drugačna mera je *zaupanje*, ki meri, kakšen je delež transakcij, ki vsebujejo desno stran pravila Y med transakcijami, ki vsebujejo levo stran pravila X:

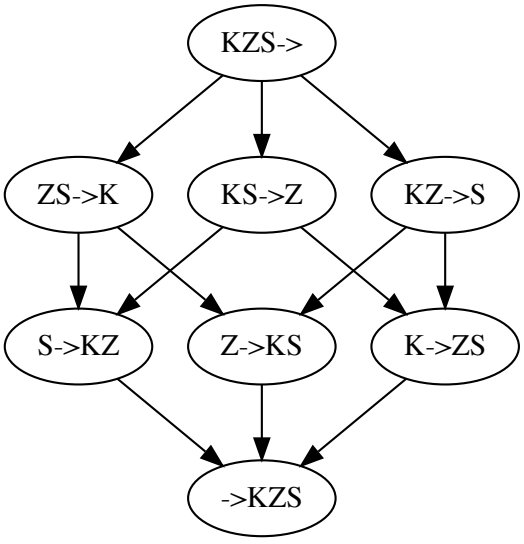
$$c(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)} \tag{9.5}$$

Za pravilo M → SK bo tako podpora enaka 0.2, zaupanje pa enako 0.5 (štiri transakcije vsebujejo mleko, od tega dve poleg mleka še sir in kruh). Podpora pravila Z → S bo 0.6, zaupanje pa 1.0.

Algoritmu za iskanje podpornih pravil predpišemo minsupp in minconf, torej minimalno podporo in minimalno zaupanje. V praksi pa pravila iščemo tako, da najprej poiščemo pogoste nabore, potem pa iz teh izluščimo pravila, katerih zaupanje je dovolj visoko. Pogosti nabori bodo namreč imeli enako podporo kot pravila, ki vsebujejo vse stvari iz pogostega nabora. Iz nabora KZS lahko tako tvorimo $2^3 - 2 = 6$ netrivialnih pravil, kjer je leva oziroma desna stran pravila neprazna:

$$KZ \rightarrow S,\ KS \rightarrow Z,\ ZS \rightarrow K,$$
$$S \rightarrow KZ,\ Z \rightarrow KS,\ K \rightarrow ZS$$

Za dani nabor stvari lahko tvorimo vsa možna povezovalna pravila na podoben način, kot smo tvorili vse možne podnabore. Z eno samo razliko. Tokrat podnabore v mreži uporabimo za desno stran povezovalnega pravila, na levo stran pa damo vse, kar je v naboru, ki ga cepimo v pravilo, še ostalo. Tako nam za nabor KZS mreža na sliki 9.3 podaja vse možne cepitve oziroma vsa možna pravila.



Slika 9.3: Mreža vseh možnih povezovalnih pravil, ki jih lahko tvorimo iz nabora KZS. Pravili v korenu (prazen nabor na levi strani) in na zadnjem nivoju mreže nista legalni, sta pa vseeno prikazani.

Ostane nam le, da tudi tu poiščemo finto oziroma pristop, kjer je mrežo povezovalnih

pravil graditi iz začetnega vozlišča tako, da kandidate klestimo glede na zahtevano zaupanje. V ta namen si oglejmo dve pravili, ki ju tvorimo iz nabora Z: $X \to Z - X$ in $X' \to Z - X'$. Vsakič smo torej desno stran pravila dobili tako, da smo od nabora Z odstranili stvari na levi strani pravila, ki so enkrat bile X in drugič X'.

Teorem 4 *Naj bosta* X *in* X'*, kjer je* X' *podnabor* X*. Velja:*

$$c(X \to Z - X) < \text{minconf} \implies c(X' \to Z - X') < \text{minconf}$$

Ta teorem moramo dokazati. Spomnimo se, da velja:

$$X' \subset X \implies \sigma(X') \geq \sigma(X)$$

Neenačbo na desni strani pomnožimo z $\sigma(Z)$ in delimo z $\sigma(X')$ ter $\sigma(X)$. Dobimo enačbi za zaupanje obeh pravil, in dokaz zgornjega teorema:

$$\frac{\sigma(Z)}{\sigma(X')} < \frac{\sigma(Z)}{\sigma(X)}$$

Z drugimi besedami: če $X \to Z - X$ ne presega minimalnega zaupanja, tudi $X' \to Z - X'$ ne bo, kjer je X' podnabor X. V naši mreži povezovalnih pravil s slike 9.3 vidimo, da so pravila s podnabori levih strani postavljena v spodnjih nivojih pravil nivoja nad njimi. Če pravilo ZS→K ne bo ustrezalo pogoju zaupanja, tudi pravili S→KZ in Z→KS, ki iz njega sledita, ne bodo.

Ravno smo torej "izumili" pristop gradnje povezovalnih pravil. Gradimo jih iz pogostega nabora. Uporabimo mrežo pravil, kjer pričnemo s prazno desno stranjo pravila in tej, na vsakem nivoju, dodamo po eno stvar ter na ta način tvorimo vse možne kombinacije. Pravila na nivoju k gradimo s kombinacijo pravil nivoja $k - 1$. Kot pri algoritmu Apriori tudi tu na vsakem nivoju hranimo samo pravila, ki ustrezajo kriteriju zaupanja.

Za velike množice transakcij gradnja povezovalnih pravil kaj lahko privede do nepreglednih množic pravil. To se seveda zgodi pri premalo ostrih pogojih, torej pri premalo visokih minsupp in minconf. Taka pravila je ročno nemogoče pregledati. Pomagamo si tako, da skušamo najprej zmanjšati množico pogostih naborov z dvigovanjem minsupp, potem pa iz nje tvorimo pravila tako, da najprej skušamo postavljati bolj ostre pogoje za zaupanje.