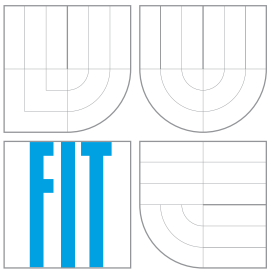


FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DOLOVÁNÍ ASOCIAČNÍCH PRAVIDEL Z DATABÁZÍ A DATOVÝCH PROUDŮ

VYBRANÉ PROBLÉMY INFORMAČNÍCH SYSTÉMŮ (VPD)

AUTOR PRÁCE Ing. MICHAL ŠEBEK

BRNO 2010

Obsah

Kapitola 1

Úvod

Prakticky všechny aplikace informačních technologií, které jsou dnes nasazeny, vytvářejí nějakou formu dat. V případě obchodních systému shromažďují data o proběhlých transakcích, klientech, v případě řídících systémů jsou pořizovány záznamy o činnosti, u bezpečnostních a dohledových systémů pak záznamy z příslušné sondy umístěné v daném prostředí. Data jsou ukládány, aby se v případě nutnosti dala dohledat patřičná informace. Avšak je možné i jejich druhotné využití. Formují se v nich totiž skryté závislosti napříč jednotlivými záznamy. Hledáním těchto závislostí, skrytých informací, se zabývá oblast *získávání znalostí z databází*. Typickým příkladem je využití pro marketingové účely, kdy hledáme závislosti mezi chováním jednotlivých zákazníků.

Algoritmy pro získávaní znalostí ze standardních databází jsou relativně prozkoumanou oblastí. Zmapovány jsou potřebné algoritmy pro shlukování, klasifikaci či asociační analýzu. Většinou se aplikují algoritmy známé z oblasti strojového učení. Avšak poměrně novým směrem je získávání znalostí, kde zdrojem dat je tzv. *proud dat*. Ten klade na algoritmy zcela nové požadavky a tato oblast je zájmem intenzivního výzkumu.

Struktura této práce je následující. Kapitola 2 se zabývá definováním problému získávání znalostí a asociačních pravidel. V kapitole 3 jsou představeny známé algoritmy pro dolování asociačních pravidel a frekventovaných množin. Kapitola 4 uvádí do problematiky dolování z proudů dat. Konkrétní algoritmy pro dolování asociačních pravidel z proudů jsou popsány v kapitole 5. Poznatky shrnuje kapitola 6.

Kapitola 2

Získávání znalostí z databází a asociační pravidla

Tato kapitola seznamuje s obecnými pojmy z oblasti **získávání znalostí z databází** (jinak též *dolování z dat*, angl. *Data Mining*). Dále se zaměřuje na definování problému dolování **asociačních pravidel**. Detaily k uvedené problematice jsou v [5, 2].

2.1 Získávání znalostí z databází

Získávání znalostí z databází je obecně definováno jako proces, kdy z databází získáváme zajímavé a potenciálně užitečné informace, které jsme do ní explicitně neukládali, v databázi vznikají implicitně.

Tento proces se sestává z následujících kroků:

- **Předzpracování dat** – v tomto počátečním kroku je nutné se především seznámit se strukturou a významem dat. Data jsou následně upravena pomocí čistění, transformací, výběru a integrace do podoby vhodné k dolování. Například je nutné odstranit z dat šum, který mohl vzniknout chyby při zadávání dat, příp. data vhodně redukovat.

- **Dolování z dat** – krok, který dal název celé disciplíně. Jedná se o aplikaci dolovacího algoritmu nad předzpracovanými daty. Zpravidla se jedná o vhodné algoritmy z oblasti strojového učení a umělé inteligence.

- **Vyhodnocení vzorů** – ne všechny vydolované informace jsou užitečné a použitelné. V rámci tohoto kroku jsou vybrané jen ty, které mají význam pro koncového uživatele.

- **Prezentace znalostí** – zpracování získaných informací do podoby akceptovatelné koncovým uživatelem. Nejčastěji jde o zpracování získaných vzorů do podoby čitelných pravidel, aplikace metod vizualizace, apod.

2.2 Frekventované množiny a asociační pravidla

Jedním z typů informací, které můžeme při získávání znalostí z databází zjišťovat, jsou tzv. *frekventované množiny* a *asociační pravidla*. Tato analýza je většinou řešena nad *transakční databází*, což je databáze, kde je vždy dvojice identifikátoru transakce TID a vektoru, který obsahuje výčet položek transakce, např. *(T100, {sůl, máslo, chléb})*. Příklad z maloobchodu není vybrán náhodou, tento typ dat je skutečně typickým příkladem komerčního nasazení dolování asociačních pravidel.

Formálně, mějme $I = \{I_1, I_2, \ldots, I_m\}$ množinu položek. Označme množinu všech dat jako D a uvažuje, že je složena z jednotlivých transakcí T, kde $T \subseteq I$. Uvažujeme množinu položek A a prohlásíme, že T obsahuje A právě tehdy, když platí $A \subseteq T$. Asociačním pravidlem pak nazveme pravidlo ve tvaru

$$A \Rightarrow B, \tag{2.1}$$

kde platí, že $A \subset T$, $B \subset T$ a $A \cap B = \emptyset$.

Abychom mohli vyjádřit míru, kdy lze prohlásit množinu za frekventovanou, zavedeme pojem **podpora**, která vyjadřuje procentuální zastoupení transakcí, které *obsahují* $A \cup B$. Míra podpory pravidla $A \Rightarrow B$ tedy lze vyjádřit jako

$$support(A \Rightarrow B) = P(A \cup B). \tag{2.2}$$

Míru, která určuje sílu implikace pravidla, nazveme **spolehlivostí**, která vyjadřuje poměr transakcí obsahujících $A \Rightarrow B$ k počtu transakcí obsahujících A. Toto lze vyjádřit

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support(A \cup B)}{support(A)}. \tag{2.3}$$

Asociační pravidla, která splňují podmínku podpory i spolehlivosti označujeme jako **silná asociační pravidla**. Příklad silného asociačního pravidla pak může být následující (pro hranice podpory 10 % a spolehlivosti 50 %):

$$koupi(chleb) \Rightarrow koupi(maslo) \quad [podpora = 20\%, spolehlivost = 70\%].$$

2.3 Typy asociačních pravidel

Asociační pravidla mají několik hledisek jejich typových vlastností. Zde jsou jsou představeny vybrané vlastnosti:

- **Úplnost množiny asociačních pravidel** vychází z použité metody. Metody poskytují následující možné druhy výsledků: *kompletní množina*, *uzavřená množina*, *aproximovaná množina* (určené počty jsou pouze přibližné, tento typ bude využitelný v případě dolování z proudu dat), atd.
- **Úroveň abstrakce pojmů** – výchozí z myšlenky, že v rámci jedné domény se můžeme nacházet na různé úrovni abstrakce pojmů (např. jedna úroveň by představovala pojmy *lev*, *tygr* a abstraktnější úroveň by byla reprezentována pojmem *šelmy*). Na tomto principu jsou pak získávány *víceúrovňová asociační pravidla*.
- **Počet dimenzí v pravidle** – určuje, zdali pravidlo je jednodimenzionální, případně jde vícedimenzionální na základě dimenzí, které pravidlo zahrnuje.

Kapitola 3

Dolování asociačních pravidel z databází a datových skladů

V těmto kapitole jsou představeny základní algoritmy pro dolování asociačních pravidel, kde zdrojová data jsou uložena v databázi nebo datovém skladu a lze k nim neomezeně přistupovat. U algoritmů jsou diskutovány jejich vlastnosti a případně používané optimalizační metody pro zvýšení efektivnosti.

3.1 Algoritmus Apriori

Základní algoritmus pro dolování frekventovaných množin a následně asociačních pravidel je algoritmus **Apriori**, jehož autory jsou Argawal a Srikant (1994). Algoritmus vytváří postupně množiny L_1, L_2, ... L_i tak, že vždy z množiny L_{k-1} je vygenerována množina L_k. Algoritmus využívá tzv. Apriori vlastnosti, kterou lze definovat následovně:

Vlastnost Apriori frekventovaných množin: Každá neprázdná podmnožina frekventované množiny je opět frekventovanou množinou.

Algoritmus má 2 základní kroky:

1. nagenerování množiny kandidátů s využitím Apriori vlastnosti frekventovaných množin,
2. ořezání množiny kandidátů na množiny, které jsou frekventovanými množinami.

Výše uvedené kroky, konkrétně 2. krok, vyžaduje v každém cyklu algoritmu jedno projítí databáze. Důležité úseky algoritmu jsou uvedeny v algoritmu 3.1.

Algoritmus 3.1 Apriori

Vstupy: transakční databáze D, minimální podpora min_supp

Výstupy: množina frekventovaných množin L

Metoda:

$L_1 = \text{nalezni\_frekventovane\_1\_polozky}(D)$;
for $(k = 2; L_{k-1} \neq \emptyset; k++)$
$C_k = \text{generuj\_mnoziny}(L_{k-1})$;
foreach $t \in D_k$
foreach $c \in C_k$
if $c \subseteq t$ **then** $c.count++$;
$L_k = \{c \in C_k | c.count \geq min_supp\}$
return $L = \bigcup_{i=1}^{k} L_i$

function generuj_mnoziny(L_{k-1})
foreach $l_1 \in L_{k-1}$
foreach $l_2 \in L_{k-1}$
if $(l_1[1] = l_2[1]) \wedge l_1[2] = l_2[2]) \wedge \cdots \wedge l_1[k-2] = l_2[k-2]) \wedge$
$\wedge l_1[k-1] < l_2[k-1])$ **then**
$c = l_1 \bowtie l_2$;
if obsahuje_frekventovane_podmnoziny(c, L_{k-1}) **then**
$C_k = C_k \cup \{c\}$;
return C_k;

function obsahuje_frekventovane_podmnoziny(c, L_{k-1})
foreach $(k-1)$-podmnozina s z c
if $s \notin L_{k-1}$ **then**
return false;
return true;

K uvedenému algoritmu je důležité podotknout, že vyžaduje, aby nad jednotlivými položkami frekventovaných množin existovalo uspořádání (matematické, lexikografické, ...) a v tomto pořadí musí být položky množin seřazeny. Tím je dosaženo efektivního porovnávání při spojování 2 frekventovaných množin v nadmnožinu.

Generování asociačních pravidel probíhá jednoduchým způsobem – pro všechny frekventované množiny $l \in L_k$: pro každou neprázdnou podmnožinu s takovou, že $s \subseteq l$, generuj pravidla tvaru $s \Rightarrow (l - s)$ tak, aby platil vztah 2.3.

3.2 Optimalizace algoritmu Apriori

Algoritmus Apriori tak, jak je uveden v předešlé kapitole, není příliš použitelný pro dolování ve větší databázi. Jeho velkou nevýhodou je procházení vždy celé databáze. Proto vznikla řada optimalizací tohoto algoritmu, které minimalizují přístupy k databázi, případně prohledávaný stavová prostor. Některé vybrané optimalizace jsou uvedeny zde:

- **Využití hašování** pro omezení generování množiny kandidátů. Například při prvním průchodu můžeme počítat všechny L_2 množiny, ukládat je použitím hašování do hash tabulky a počítat výskyty těchto podmnožin. Tento krok je využitelný především pro $k = 2$.
- **Odstranění transakcí** z databáze pro další cykly algoritmu Apriori, které neobsahují frekventované k-množiny. Je zřejmé, že tyto transakce není třeba v dalších cyklech testovat.
- **Rozdělování dat** umožňuje aplikovat algoritmus pouze ve 2 průchodech celou databází. Nejdříve je databáze rozdělena na n částí. Pro každou část jsou nalezeny lokálně frekventované množiny. Absolutní minimální podpora pro tyto množiny je $min_supp * pocet\_transakci\_casti$. Speciální datová struktura umožňuje nalézt k-frekventované množiny v jednom průchodu. Potom se provede sjednocení frekventovaných množin z těchto n částí a ověří se dalším průchodem, zdali jsou frekventované i v celé databázi.
- **Vzorkování** je jednoduchou metodou, kdy vytvoříme podmnožinu transakcí S z databáze D. Frekventované množiny tak můžeme vyhledávat například v objemu dat, který je dostupný v hlavní paměti. Tato metoda nemusí najít všechny frekventované množiny v D.

3.3 Algoritmus FP-Tree

Slabinou algoritmu Apriori je generování kandidátních množin. Je třeba si uvědomit, že se tomto kroku nageneruje řádově více množin, než je pak skutečně frekventovaných. Proto je snaha tento krok v dolování zcela vynechat.

Algoritmem, který vynechává krok generování kandidátních množin, je algoritmus nazvaný **FP-Tree**. Tento algoritmus využívá pouze 2 průchodů databází:

1. V prvním průchodu databází jsou podobně jako v Apriori zjištěny 1-frekventované množiny a určeny absolutní četnosti jednotlivých položek transakcí. Tyto položky jsou podle počtu výskytů seřazeny sestupně.

2. V druhém průchodu databází sestavujeme FP strom. Princip je zhruba takový, že kořenem stromu označíme uzel *null* a na něm v sestupném pořadí určeném z předchozího kroku vytváříme synovské uzly a jejich poduzly. Každý uzel obsahuje čítač a pokud již daná položka ve stromu existuje, je jen inkrementován čítač. Na základě takto vytvořené struktury pak lze vygenerovat frekventované množiny.

3.4 Přístupy pro dolování víceúrovňových pravidel z datových skladů

Doposud diskutované algoritmy nepočítaly s víceúrovňovostí pravidel, resp. jednotlivých položek frekventovaných množin. V této části je zmíněna okrajově problematika, kdy máme různé úrovně abstrakce pojmů v databází, resp. v OLAP systému. Pro algoritmy je především nutné zvolit vhodnou strategii v pohledu na minimální podporu. Není vhodné mít nastavenou stejnou úroveň podpory pro různé úrovně transakce. Patrně se v OLAP systému bude lišit četnost výskytů prodeje automobilů a četnost prodeje jednotlivých značek. Možné přístupy k úrovním minimální podpory:

- Pro každou úroveň specifikovat minimální podporu pro frekventované množiny.
- Testování položek na $i + 1$ úrovni pouze, pokud položky na i-té úrovni jsou frekventované.
- Použití předchozího pravidla v aplikaci na množiny, ne na položky.

Kapitola 4

Dolování v proudech dat

V předešlých kapitolách jsme se zabývali dolováním, kde byla zdrojem dat standardní databáze. Tyto algoritmy ale nejsou přímo použitelné pro analýzu *proudů dat*. Proudy dat mají oproti běžným databázím některá specifika, která musí algoritmy pro dolování respektovat. Je tedy přirozené, že je snaha stávající algoritmy modifikovat nebo navrhnout nové tak, aby bylo možné řešit úlohy hledání asociací, klasifikace, predikce a shlukování i nad proudy dat. Tato kapitola představuje úvod do problematiky proudů dat a dolováním v nich.

4.1 Proudy dat

Za **proud dat** typicky považujeme potenciálně nekonečnou sekvenci dat. Data v proudu obvykle přicházejí velkou rychlostí. Lze si představit například data z videokamery či sondy umístěné v síti.

Pro dolovací algoritmy z toho plynou některé specifické požadavky. Hlavním požadavkem je, že by algoritmus neměl pracovat s **prostorovou složitostí** $O(N)$ obvyklou pro klasické algoritmy dolování. Tyto algoritmy většinou vyžadují více průchodů daty a data musí mít uložená. Toto vzhledem k nekonečnosti (objemnosti) proudu není možné. U algoritmů nad proudy dat požadujeme prostorovou složitost v nejhorším případě polylogaritmickou, uváděno bývá $O(log^k N)$ [5], kde N je počet vzorků na vstupu a $k \in N$ konstanta. Této složitosti algoritmy dosahují postupem, že nepotřebují ukládat a procházet všechna data, ale z příchozích dat shromažďují pouze přehledy (*angl. synopses*) sumarizující příchozí data. Z toho plyne i fakt, že není možné data zpracovávat víceprůchodově – není možné všechny příchozí data uchovávat pro další zpracování. Algoritmy by rovněž neměly být příliš časově složité, vzorky z proudu dat je nutné zpracovávat a vyhodnocovat v reálném čase.

4.2 Koncept v proudech dat

Jedním z hlavním aspektů, který musí zohledňovat algoritmy pro dolování v proudech, je měnící se statistická informace v datech. Tato problematika se nazývá **koncept v proudu dat** (*angl. concept drift*). Jde o situaci, kdy se postupně mění informace sledovaná procesem dolování. Při dolování z videa z průmyslové kamery například rozdílná povaha obrazu během dne a noci.

Změna konceptu dat bývá v algoritmech zohledněna následujícími možnými způsoby [7]:

- **Výběr vzorků** – princip metody spočívá ve výběru pouze několika vzorků, u kterých se předpokládá spojitost s aktuálním konceptem. Nejběžnější metodou, která tento přístup využívá, je metoda pohybujícího se okna přes množinu přicházejících prvků.
- **Váhování vzorků** – metoda je založena na přidělení váhy každému vzorku dat a idea spočívá v tom, že se váha snižuje se stářím vzorku. Tento přístup je dobře použitelný v klasifikátorech založených na *Support Vector Machines (SVM)*.
- **Souborné učení** – kombinuje více popisů konceptu, predikce jsou kombinovány na základě vah či „hlasování“ a vybere se nejvíce pravděpodobný popis.

4.3 Přístupy pro dovolání v proudech dat

Z teoretického pohledu můžeme algoritmy používané pro zpracování proudů dat rozdělit do 2 skupin [4, 5] **techniky založené na datech** a **techniky založené na úloze**.

Techniky založené na datech

Tato skupina algoritmů pracuje s využitím sumarizačních údajů nad daty, nebo používá pouze některé (vybrané) vzorky dat. Tato podkapitola představuje některé přístupy používané touto skupinou algoritmů.

- **Vzorkování** (*angl. Sampling*) – jde o intuitivní metodu, kdy ze vstupních dat zpracováváme pouze náhodně vybrané vzorky. Algoritmy vzorkování často pracují jako tzv. vzorkování se zásobníkem (*angl. reservoing sampling*).

- **Vynechání dat** (*angl. Load Shedding*) – metoda zahodí shluky vstupních dat. Užití této metody je však problematické, neboť může dojít k nepříznivému zkreslení výsledků.

- **Projekce podmnožiny vlastností a přehledy** (*angl. Sketching, Synopsis*) – metoda pracuje nad daty v jednom průchodu následovně. Uvažujme sekvenci dat $S = \{x_1, \ldots, x_N\}$, kde každý vzorek x_i je z univerza $D = \{1, \ldots, d\}$. Dále předpokládejme momenty frekvencí výskytů F_k definované jako

$$F_k = \sum_{i=1}^{d} m_i^k, \tag{4.1}$$

  kde d je velikost domény $|D|$ a m_i je frekvence vzorku i v datech a $k \geq 0$. Jednotlivé hodnoty momentů lze pak interpretovat následovně. F_0 odpovídá počtu různých prvků dat, F_1 počtu vstupních dat a F_2 se označuje jako *Gini index homogenity*. Dále předpokládejme, že pracujeme s omezenou pamětí. Pokud je rozsah domény větší, než velikost použitelné paměti, spočítáme přehledy (synopse) momentů F_k, které se označují jako *sketches*. Aplikace tohoto přístupu aproximují hodnoty F_2 v prostorové složitosti $O(log(d) + log(N))$. Podrobně se problematikou zabývá [1].

Techniky založené na úloze

Jedná se o techniky úpravy stávajících algoritmů a návrhu zcela nových.

- **Aproximační algoritmy** – silnou vlastností těchto algoritmů je, že jsou schopny aproximovat řešení v definovaných mezích přesnosti – tedy s ohraničenou chybou.

- **Posuvné okno** – metoda rovněž jako vzorkování se zásobníkem používá paměť pro uložení určitého počtu vzorků. Narozdíl od vzorkování ale nepoužívá náhodně vybraná data, ale data nedávná. Pokud předpokládáme okno velikosti w a čas příchodu vzorku t, pak je platnost vzorku po dobu $t + w$. Pak je vzorek z okna odstraněn a nahrazen následujícím vzorkem. Aby metoda zohlednila celou historii proudu dat, jsou nejnovější prvky v okně analyzovány důkladně a starší data jsou reprezentovány pouze souhrnnými statistikami.

Následující kapitola se bude věnovat aplikaci výše uvedených technik v algoritmech, které jsou nyní používané pro dolování asociačních pravidel, příp. frekventovaných množin, v proudech dat.

Kapitola 5

Dolování frekventovaných množin a asociačních pravidel z proudů dat

Tato kapitola se zaměřuje na vysvětlení některých algoritmů pro dolování frekventovaných množin, frekventovaných položek a asociačních pravidel v proudech dat. V kapitole 4 byly uvedeny některé hlavní vlastnosti proudů dat, které dolovací algoritmy musí zohledňovat. Připomeňme především nekonečnost proudu dat a z toho plynoucí nutnost zpracovávat data jednoprůchodově a to v okamžiku příchodu jednotlivých vzorků.

Tato kapitola vychází především z metod pro dolování frekventovaných množin popsaných souhrnně v [3, 5, 4]. Algoritmy pro dolování frekventovaných položek a množin se rozdělují do tří skupin:

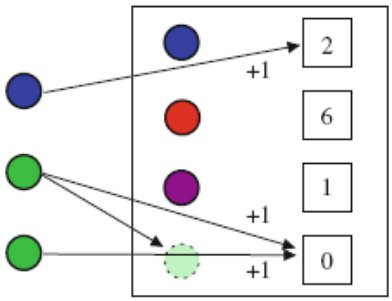
- algoritmy založené na čítači,
- kvantilové algoritmy,
- a algoritmy založené na lineární projekci *(angl. sketches)*.

5.1 Hledání frekventovaných množin v proudech

Zájem o tuto oblast projevil už v osmdesátých letech Moore. Tehdy byl vysloven v podstatě problém nalezení většiny a hledal se algoritmus, na základě kterého by mohla bát spočtena většina. Pro řešení tohoto problému byl použit algoritmus **Majority**. Algoritmus pracuje na principu čítačů pro jednotlivé položky. Algoritmus zpracovává položky, pro vyskytující se položku čítač inkrementuje, pro ostatní je dekrementován.

Problém byl znovuotevřen až na přelomu století, kdy je algoritmus Majority upraven tak, aby uchovával vždy maximálně k vzorků proudu a je publikován pod

názvem **Frequency**. Algoritmus opět pracuje s dvojicemi položka, čítač. Algoritmus čte vzorky ze vstupu. Pokud aktuální vzorek na vstupu je již uložen, je mu inkrementován čítač o 1. Jinak, pokud ještě není počet uložených dvojic k, pak je vzorek uložen a čítač je mu nastaven na 1. Pokud je již uloženo k vzorků, jsou dekrementovány všechny čítače a vyřazeny položky, které mají čítač roven nule. Postup algoritmu je naznačen na obrázku 5.1.



Obrázek 5.1: Znázornění činnosti algoritmu založeného na čítačích. Převzato z [3].

5.2 LossyCounting algoritmus

Algoritmus **LossyCounting** byl představen v [6] a je jedním ze základních algoritmů pro zpracování proudu dat. Jde o metodu používající čítače pro získání frekventovaných položek a frekventovaných množin s ohraničenou chybovostí.

Algoritmus má následující uživatelské parametry:

- *minimální podporu* označenou jako σ
- a *chybovou mez* ϵ.

Princip algoritmu je následující. Proud dat je nejdříve rozdělen na skupiny o w položkách tak, že $w = \lceil 1/\epsilon \rceil$. Algoritmus ukládá jednotlivé položky do seznamu, kde záznamy v seznamu jsou trojice:

$$(\textit{položka } e, \textit{ frekvence } f, \textit{ maximální chyba frekvence } \Delta).$$

Algoritmus zpracovává skupiny položek takto. Přidává jednotlivé položky do seznamu. Pokud položka existuje, je inkrementován její čítač frekvence výskytů o 1. V opačném případě je vložena do seznamu a f nastavena na 1, Δ položky je nastaveno následovně: uvažujme, že položka je ze skupiny v pořadí b_{curr}, potom Δ bude mít hodnotu $B_{curr} - 1$. Aby algoritmus měl omezené paměťové nároky, zahrnuje

ještě odstraňování položek ze seznamu, pokud je splněna pro daný záznam podmínka $f + \Delta \leq b_{curr}$. Procedura aktualizace seznamu při příchodu položky je uvedena v mírně obměněném algoritmu 5.1, který ukládá $f + \Delta$ společně.

Algoritmus má následující vlastnosti:

- Pracuje s ohraničenou prostorovou složitostí. Lze ukázat, že maximální požadovaný prostor je $O(\frac{1}{\epsilon} log \epsilon N)$, kde N je délka proudu.
- Vždy najde *všechny* frekventované položky v proudu dat.
- Algoritmus může prohlásit i nefrekventovanou položku za frekventovanou, avšak pouze v rámci ohraničené nepřesnosti na frekvenci. Frekvence takové položky je minimálně $\sigma N - \epsilon N$.
- Chyba frekvence položky v proudu dat je vždy maximálně ϵN.

Uvedený algoritmus lze použít úpravou položek za množiny na dolování frekventovaných množin.

Algoritmus 5.1 ***LossyCounting(k)*** $n \leftarrow 0; \Delta \leftarrow 0; T \leftarrow 0;$

```
foreach i do
    n ← n + 1
    if i ∈ T then
        c_i ← c_i + 1;
    else
        n ← T ∪ {i};
        c_i ← 1 + Δ;
    if ⌊n/k⌋ ≠ Δ then
        Δ ← ⌊n/k⌋;
        forall j ∈ T do
            if c_j < Δ then T ← T\{j};
```

5.3 SpaceSaving algoritmus

V principu jde o velmi podobný algoritmus k algoritmu LossyCounting. Opět využívá čítače k uložení frekvence položky v proudu dat. Algoritmus ukládá maximálně $O(k)$ položek seznamu, má tedy ohraničenou prostorovou složitost. Tuto složitost algoritmus vynucuje případnou náhradou staré položky za novou v případě, že je seznam položek již zaplněn k záznamy. Zpracování položky je uvedeno v algoritmu 5.2

Algoritmus 5.2 ***SpaceSaving(k)***

$n \leftarrow 0; T \leftarrow 0;$
foreach i **do**
  $n \leftarrow n + 1$
  if $i \in T$ **then**
    $c_i \leftarrow c_i + 1;$
  else if $|T| < k$ **then**
    $n \leftarrow T \cup \{i\};$
    $c_i \leftarrow 1$
  else
    $j \leftarrow argmin_{j \in T} c_j;$
    $c_i \leftarrow c_j + 1$
    $T \leftarrow T \cup \{i\} \backslash \{j\};$

5.4 Kvantilové algoritmy, GK algoritmus

V algoritmech založených na kvantilech se na problém nahlíží odlišně. Jde o problém nalezení Φ-tého kvantilu nad uspořádanými vzorky ze stupu. Nadefinujeme ohodnoceni pro i-tý element jako $rank(i) = \sum_{j<i} f_i$. Potom pro Φ-kvantil platí $rank(i) \leq \Phi n$ a $rank(i+1) > \Phi n$. Pro přibližné řešení s chybou ϵ lze definovat následující platnost $rank(i) \leq (\Phi + \epsilon)n$ a $rank(i+1) > (\Phi + \epsilon)n$.

GK algoritmus – je podobný algoritmu LossyCounting. Algoritmus ukládá n-tice obsahující element, g a Δ, kde g vyjadřuje rozdíl mezi nejmenším možným ohodnocením $rank$ aktuálního elementu a elementu předcházejícího, Δ je pak rozdíl mezi minimálním a maximálním možným ohodnocením. Algoritmus je velmi podobný výše uvedeným a lze dohledat v literatuře [3] společně s dalšími možnými algoritmy založenými na principu kvantilů, např. **QDigest**.

5.5 Sketches, CountSketch algoritmus

Pro tzv. *sketche* se využívá lineární projekce vstupu. Proud je reprezentován vektorem, kde i-tý záznam je f_i. Pro linearizaci se používá hašovací funkce.

CountSketch algoritmus pracuje s maticí $d \times w$ čítačů. Pro každý element proudu se provede právě d projekcí postupně do všech řádků matice. Hašovací funkce h_j zobrazuje element do konkrétního sloupce $[w]$. Hašovací funkce $g_j(i)$ zobrazuje elementy

na hodnoty $\{-1, +1\}$. Detailně lze najít aktualizaci v algoritmu 5.3. Algoritmus dosahuje prostorové složitosti $O(\frac{1}{e^2} log \frac{1}{\delta})$, kde δ určuje pravděpodobnost splnění ohraničení chyby.

Algoritmus 5.3 ***CountSketch(w,d)***

$C[1,1] \dots C[d,w] \leftarrow 0$;
for $j \leftarrow 1$ **to** d **do**
    Initialize g_j, h_j;
foreach i **do**
    $n \leftarrow n + 1$;
    for $j \leftarrow 1$ **to** d **do**
        $C[j, h_j(i)] \leftarrow C[j, h_j(i)] + g_j(i)$;

Kapitola 6

Závěr

Tato práce shrnuje podstatu dolování asociačních pravidel a frekventovaných množin. V první části shrnuje dosažené poznatky a používané algoritmy pro dolování tohoto typu znalostí z klasických databází. Diskutuje používané algoritmy Apriori a FP-Tree s jejich optimalizacemi.

V druhé části předkládá aktuální oblast výzkumu dolování v proudech dat. Nastiňuje problematiku dolování v proudech dat obecně, neboť je nutné pochopit, jaké požadavky dolování v proudech ve své podstatě na algoritmy klade. Následně představuje základní algoritmy pro dolování asociačních pravidel z proudu dat, které byly objeveny a používají se. Přesto tato oblast není zdaleka uzavřeným tématem a nabízí mnoho možností k navazujícímu výzkumu.

Literatura

[1] Babcock, B.; Babu, S.; Datar, M.; aj.: Models and issues in data stream systems. In *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, New York, NY, USA: ACM, 2002, ISBN 1-58113-507-6, s. 1–16.

[2] Cios, K. J.; Pedrycz, W.; Swiniarski, R. W.; aj.: *Data Mining: A Knowledge Discovery Approach*. Springer, 2007, ISBN 978-0-387-36795-8, 606 s.

[3] Cormode, G.; Hadjieleftheriou, M.: Methods for finding frequent items in data streams. *The VLDB Journal*, ročník 19, č. 1, 2010: s. 3–20, ISSN 1066-8888.

[4] Gaber, M. M.; Zaslavsky, A.; Krishnaswamy, S.: Mining data streams: A review. *SIGMOD Rec.*, ročník 34, č. 2, 2005: s. 18–26, ISSN 0163-5808.

[5] Han, J.; Kamber, M.: *Data Mining: Concepts and Techniques*. Elsevier Inc., druhé vydání, 2006, ISBN 978-1-55860-901-3, 770 s.

[6] Manku, G. S.; Motwani, R.: Approximate frequency counts over data streams. In *VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases*, VLDB Endowment, 2002, s. 346–357.

[7] Tsymbal, A.: The problem of concept drift: definitions and related work. Technická zpráva, Department of Computer Science, Trinity College Dublin, Ireland, 2004.