ACTA UNIVERSITATIS LODZIENSIS
FOLIA OECONOMICA 183, 2004

Sebastian Szamański, * *Ryszard Budziński* **

METODY EKSPLORACJI REGUŁ ASOCJACYJNYCH I ICH ZASTOSOWANIE

Wprowadzenie

Ogromny postęp technologiczny ostatnich lat oraz związane z nim rozpowszechnienie się systemów komputerowych w licznych przedsiębiorstwach zaowocowało nagromadzeniem się bardzo dużych zasobów danych przechowywanych w bazach danych. Ilość informacji z czasem przerosła ludzkie możliwości analizy oraz utrudniła podejmowanie decyzji na ich podstawie. Sytuacja taka sprawiała, że w ciągu ostatnich lat systemy informatyczne ewoluowały zwiększając swoją funkcjonalność i niezawodność.

Obecnie nową i ciągle rozwijającą się gałąź systemów informatycznych stanowią systemy automatycznie poszukujące wiedzy wśród zbiorów ogromnych ilości danych gromadzonych w bazach danych. Dziedzina informatyki badająca tego typu oprogramowanie znana jest pod nazwą „odkrywania wiedzy w bazach danych" (Knowledge Discovery in Databases), w skrócie **KDD**. Z **KDD** ściśle związane są technologie eksploracji danych (Data Mining). Odkrywanie wiedzy nie wymaga od użytkownika zakładania żadnych hipotez, gdyż są generowane automatycznie i weryfikowane przez system komputerowy. Zadaniem eksperta jest jedynie ocena wygenerowanej wiedzy, zazwyczaj w postaci reguł, sprowadzająca się do odrzucenia lub przyjęcia wyników na podstawie określonych wskaźników statystycznych.

Eksploracja danych (Data Mining) definiowana jest jako zbiór technik odkrywania nietrywialnych zależności i schematów w dużych zbiorach danych. Do metod eksploracji danych zalicza się :

- klasyfikację,
- klastrowanie,
- odkrywanie sekwencji,
- odkrywanie charakterystyk,
- wykrywanie zmian i odchyleń,

* Dr, adiunkt, Instytut Systemów Informatycznych, Wydział Informatyki, Politechnika Szczecińska.

** Prof. dr hab., Instytut Systemów Informatycznych, Wydział Informatyki, Politechnika Szczecińska, e-mail: R.Budzinski@man.szczecin.pl .

- odkrywanie podobieństw w przebiegach czasowych,
- odkrywanie asocjacji.

Ostatnia z wymienionych technik (będąca tematem niniejszego opracowania) – odkrywanie asocjacji – polega na znajdowaniu powtarzających się zależności wśród zadanego zbioru atrybutów wyselekcjonowanych z bazy danych. Zależności te określają, jakie i jak często występują powiązania (asocjacje) pomiędzy wartościami analizowanego zbioru danych. Eksploracja asocjacji odbywa się automatycznie, a użytkownik wskazuje jedynie dane wejściowe, na których przeprowadzone ma być odkrywanie wiedzy. Wynikiem działania algorytmu są wyspecyfikowane reguły wraz z odpowiednimi wskaźnikami w postaci **pewności** oraz **wsparcia** każdej z reguł.

Zastosowanie

Eksploracja reguł asocjacyjnych stosowana jest do rozwiązywania problemów w licznych dziedzinach nauki, ekonomii, marketingu oraz wielu innych związanych z naszym codziennym życiem. Najbardziej znanymi przykładami wykorzystania odkrywania asocjacji są :

- analiza koszyka zakupów (*basket analysis marketing*),
- projektowanie katalogów reklam/ofert do określonych grup odbiorców (*catalog design*),
- rozwiązywanie problemów klasteryzacji,
- analiza przyczyn spadku sprzedaży danego produktu, utraty prowadzenia na rynku (*loss leader analysis*).

Wyszukiwanie reguł asocjacyjnych ma za zadanie zwiększyć wydajność systemów baz danych pod względem funkcjonalności umożliwiając zadawanie złożonych zapytań, przykładowo w postaci:

- Znajdź wszystkie reguły, których konsekwencją jest sprzedaż produktu X. Znajomość reguł tego typu pozwoli podnieść sprzedaż produktu X.
- Znajdź wszystkie reguły, które powstały w wyniku sprzedaży produktu X. Odkryj te reguły, które powstały w konsekwencji sprzedaży produktu X. Powyższe reguły mają określić, jaki wpływ na sprzedaż wskazanych produktów może mieć zaprzestanie sprzedaży produktu X.
- Znajdź wszystkie reguły, które powstały w wyniku sprzedaży X, a które spowodowały, że wzrosła sprzedaż Y. Tego typu zapytania mogą wskazać, jakie produkty należy sprzedawać razem z X, tak by wzrosła sprzedaż Y.
- Znajdź wszystkie reguły, które wiążą ze sobą produkty umieszczone na półce A oraz półce B w sklepie.

– Znajdź k - najlepszych reguł, w wyniku których następuje sprzedaż produktu X. Podczas wyboru „najlepszych" reguł można posłużyć się współczynnikami pewności oraz wsparcia dla reguł.

Definicje i założenia związane z eksploracją reguł asocjacyjnych

Do określenia formalnej definicji problemu odkrywania reguł asocjacyjnych należy przyjąć następujące założenia:

I – zbiór literałów zwanych elementami (atrybutami),
$I = \{i_1, i_2, i_3, ..., i_m\}$,
D – zbiór transakcji, gdzie każda transakcja T jest zbiorem elementów takich, że $T \subseteq I$.

Z każdą transakcją związany jest unikalny numer, zwany często TID. Mówimy, że transakcja T zawiera X, zbór elementów ze zbioru I, jeśli $X \subseteq T$.

Reguła asocjacyjna zatem jest implikacją $X \Rightarrow Y$, gdzie $X \subseteq I, Y \subseteq I$ oraz $X \cap Y = \varnothing$.

Częstotliwością (*frequency*) występowania podzbioru atrybutów X w zbiorze transakcji D nazywamy stosunek wierszy (m), które zawierają atrybuty należące do X, do liczby wszystkich wierszy w tabeli (*dbsize*), co możemy zapisać w postaci:

$$c(X, D) = \frac{m}{dbsize}$$

Zbiór atrybutów X jest **częsty,** jeśli $c(X, D) \geq z$, gdzie z jest zadanym przez użytkownika progiem częstotliwości występowania podzbioru atrybutów w tabeli.

Reguła $X \Rightarrow Y$ zawarta w zbiorze transakcji D posiada **pewność c** (*confidence*) jeśli c% transakcji w zbiorze D zawierających X, zawiera także Y.

Pewność reguły można definiować również jako stosunek częstotliwości występowania sumy podzbiorów atrybutów $X \cup Y$ do częstotliwości występowania podzbioru atrybutów X.

$$conf(P,D) = \frac{c(X \cup Y, D)}{c(X,D)},$$

gdzie:

P – reguła $X \Rightarrow Y$.

Reguła $X \Rightarrow Y$ posiada **wsparcie s** (*support*) w zbiorze transakcji *D*, jeśli s% procent transakcji w zbiorze *D* zawiera $X \cup Y$.

Wsparcie może być również określone jako stosunek liczby wierszy w tabeli zawierających wszystkie atrybuty zawarte w przesłankach i konkluzjach reguły do liczby wszystkich wierszy w tabeli.

W algorytmie **Apriori** wykorzystano jeszcze inną definicję, przyjmującą, iż **wsparciem** reguły jest po prostu liczba wierszy w tabeli zawierających wszystkie atrybuty zawarte w przesłankach i konkluzjach reguły.

Metody odkrywania reguł asocjacyjnych

Problem odkrywania reguł asocjacyjnych sprowadza się do wygenerowania wszystkich tych reguł, które posiadają współczynniki wsparcia oraz pewności wyższe od zadanych przez użytkownika (minimalne wsparcie i minimalna pewność).

Eksploracja reguł sprowadza się zatem do dwóch zasadniczych podproblemów:

1. Znalezienie wszystkich kombinacji atrybutów, które mają większe wsparcie od pewnego zadanego z góry progu zwanego minimalnym wsparciem (*minsupport*). Te kombinacje nazywamy dużymi zbiorami elementów l-zbiorami (*large itemsets*). Wszystkie kombinacje niespełniające powyższych kryteriów (poniżej zadanego progu) nazywane są małymi zbiorami elementów s-zbiorami (*small itemsets*).

Rozwiązanie powyższego problemu jest trudne szczególnie dla dużych baz danych i wymaga stosowania specjalnych algorytmów.

2. Dla danych dużych zbiorów l-zbiorów (zbiorów częstych) $Y = i_1, i_2, i_3, ..., i_k$ gdzie $k \geq 2$ generowane są wszystkie reguły, które wykorzystują elementy ze zbioru Y. Przesłanką dla wszystkich tych reguł powinien być podzbiór X ze zbioru Y, gdzie X posiada k-1 elementów, a konkluzją jest podzbiór atrybutów Y-X. Do wygenerowania reguły $X => i_j \mid conf$, gdzie $X = i_1, i_2, i_3, ..., i_{j-1}, i_{j+1}, ... i_k$ należy uwzględnić wsparcie zbioru Y i podzielić przez wsparcie zbioru X. Jeśli uzyskany stosunek jest większy od *conf*, wówczas reguła jest akceptowana ze współczynnikiem *conf*, w przeciwnym wypadku reguła jest odrzucana.

Mając określone zbiory częste, rozwiązanie drugiego podproblemu jest w większości przypadków zadaniem prostym.

Należy zauważyć, że jeśli zbiór Y jest zbiorem częstym, wówczas każdy jego podzbiór jest również zbiorem częstym.

Reguły wywiedzione ze zbioru Y spełniają wszystkie ograniczenia, gdyż zbiór Y jest sumą elementów będących konkluzjami bądź przesłankami każdej w ten sposób utworzonej reguły.

Do rozwiązania problemu pierwszego zostały opracowane specjalne algorytmy, wśród których możemy wymienić, takie jak:

algorytm AIS – jeden z pierwszych algorytmów odkrywania reguł asocjacyjnych omówiony w pracy R. Agrawala i T. Iminlinskiego [1];

algorytm SETM – algorytm zaprezentowany w pracy A. Swami i M. Houtsma [7];

algorytm Apriori – algorytm omówiony w pracy R. Agrawala i S. Ramakrishnana [2];

algorytm AprioriTID – modyfikacja algorytmu **Apriori** zaprezentowana razem z algorytmem Apriori w pracy [2].

Algorytm Apriori

Jednym z najpopularniejszych obecnie algorytmów jest algorytm **Apriori.** W algorytmie tym przyjmuje się następujące założenia:

$\mathbf{L_k}$ – zbiór k-elementowych l-zbiorów (spełniających warunek minimalnego wsparcia). Każdy element zbioru L_k ma 2 pola: zbiór atrybutów oraz wartość poparcia (liczba).

$\mathbf{C_k}$ – zbiór kandydatów – k-elementowych zbiorów atrybutów (potencjalnie l-zbiorów). Każdy element zbioru L_k ma 2 pola: zbiór atrybutów oraz wartość poparcia (liczba).

D – baza danych (zbiór transakcji).

Transakcja może być rozumiana jako wiersz tabeli utworzonej zapytaniem SQL. W algorytmie **Apriori** można przyjąć założenia upraszczające, które zakładają, że przechowywane dane są posortowane rosnąco. Nie jest istotne, jak jest zdefiniowana relacja porządku, może to być np. porządek leksykograficzny.

Początkowo algorytm Apriori generuje jednoelementowe zbiory częste sprawdzając dla każdego atrybutu z bazy danych, czy występuje on dostatecznie często. W kolejnym kroku dla każdego elementu ze zbiorów częstych (k-1)-elementowych wywoływana jest funkcja apriori-gen, która na podstawie zbioru częstego L_{k-1} generuje zbiór kandydujący C_k, zawierający wszystkie zbiory 1-elementowe, które pretendują do zbiorów częstych L_k. Funkcja

apriori-gen w pierwszym kroku dokonuje łączenia dwóch zbiorów kandydujących (*join step*) $p \cup q \in L_{k-1}$, które mają takie same k-2 pierwsze elementy. Suma tych zbiorów zostaje włączona do zbioru C_k. W tym kroku wykorzystuje się założenie, iż dane są posortowane rosnąco. W kolejnym kroku algorytm dokonuje filtrowania (*prune step*), co oznacza usuwanie tych elementów zbioru C_k, których jakikolwiek podzbiór k-1 elementowy nie należy do zbioru L_{k-1}. Następnie dla każdej transakcji z bazy danych tworzony jest podzbiór C_t ze zbioru C_k składający się z tych elementów, których atrybuty są zawarte w rozpatrywanej transakcji. Jednocześnie aktualizowany jest wskaźnik wsparcia dla danego elementu poprzez inkrementację o jeden jego wartości. Na koniec utworzony zostaje zbiór składający się z k-elementowych zbiorów częstych. W wyniku działania algorytmu otrzymuje się zbiór zawierający wszystkie zbiory częste o różnej liczbie elementów, które spełniły zadany warunek minimalnego wsparcia.

Algorytm AprioriTID

Algorytm **AprioriTID** jest modyfikacją algorytmu **Apriori**. Algorytm ten również wykorzystuje funkcję apriori-gen do określenia zbiorów kandydujących. Interesującą zmianą w tym rozwiązaniu jest to, że baza danych D nie jest wykorzystywana do obliczenia wsparcia podczas pierwszej iteracji, a tę rolę spełnia nowo wprowadzony zbiór kandydujący $\overline{C}_k$ wraz z korespondującymi numerami TID dla każdej transakcji. To na jego podstawie określane jest wsparcie dla każdego zbioru kandydującego. Każdy element zbioru $\overline{C}_k$ ma postać $< TID, \{X_k\} >$, gdzie każdy X_k jest potencjalnym zbiorem częstym występującym w transakcji o identyfikatorze TID. Dla k=1, $\overline{C}_1$ jest zborem zbiorów jednoelementowych pochodzących z każdej transakcji bazy danych D. Kolejne zbiory $\overline{C}_k$ dla k > 1 generowane są przez algorytm, gdzie dla każdej transakcji l określana jest para $< l.TID, \{c \in C_k \mid c \subset l\} >$. Jeśli transakcja nie zawiera żadnego zbioru kandydującego, wówczas zbiór $\overline{C}_k$ nie zawiera wpisu dla tej transakcji. Zatem liczba elementów zbioru $\overline{C}_k$ może być mniejsza niż liczba transakcji w bazie danych, szczególnie dla dużych wartości indeksu k. Ponadto dla dużych wartości indeksu k, każdy element może być mniejszy niż korespondująca z nim transakcja, ponieważ bardzo niewiele zbiorów kandydujących może zawierać się w danej transakcji.

Algorytm AIS

W algorytmie **AIS** kandydujące zbiory generowane i zliczane są na bieżąco podczas przeszukiwania całej bazy danych. Po odczytaniu pojedynczej transakcji (wiersza) określa się, które atrybuty predysponują do bycia zbiorem częstym w odniesieniu do atrybutów zawartych w transakcji z poprzedniej iteracji. Nowe kandydujące zbiory generowane są poprzez rozszerzenie zbiorów częstych o nowe elementy z danej transakcji. Zbiór częsty rozszerzany jest jedynie o te elementy, które tworzą również zbiór częsty i występują w kolejności leksykograficznej. Zbiory kandydujące wygenerowane z danej transakcji są dodawane do zbiorów kandydujących utrzymywanych przez wszystkie iteracje lub zwiększany jest licznik, jeśli dany zbiór już wystąpił we wcześniejszej transakcji.

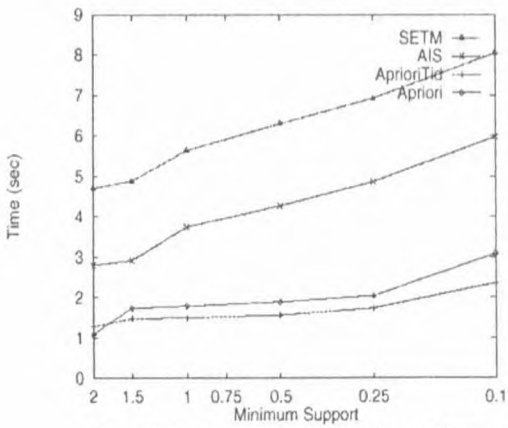
Algorytm SETM

Algorytm **SETM** powstał między innymi w wyniku potrzeby wykorzystania języka SQL do znajdowania zbiorów częstych. Razem ze zbiorem kandydujących zbiorów częstych przechowywany jest numer TID transakcji, z której wygenerowano dany zbiór. Każdy element takiego zbioru jest w postaci <TID, zbiór elementów>. Podobnie jak algorytm **AIS**, również **SETM** generuje kolejne zbiory kandydujące na bieżąco podczas iteracji przez całą bazę danych. Również **SETM** generuje i zlicza każdy kandydujący zbiór, tak jak to miało miejsce w algorytmie **AIS**. Jednakże, by móc wykorzystać operację złączenia JOIN z języka SQL dla zbiorów kandydujących, algorytm **SETM** oddziela generację zbiorów od ich zliczania. Algorytm zapamiętuje kopię zbioru kandydującego wraz z jego numerem TID transakcji w sekwencyjnej strukturze. Na koniec każdej iteracji wsparcie dla zbiorów kandydujących jest przeliczane, zbiory są sortowane i sumowane na podstawie wspomnianej sekwencyjnej struktury. By uniknąć operacji na zbiorach przechowywany jest wyżej wymieniony numer TID wraz z każdym zbiorem kandydującym. Zbiory które nie mają minimalnego wsparcia zostają usuwane. Wiedząc, że w bazie danych transakcje są posortowane według numerów TID, algorytm **SETM** może w prosty sposób znaleźć zbiory częste zawarte w transakcji poprzez posortowanie każdego zbioru częstego zgodnie z numerem TID. W rezultacie algorytm musi przejść przez każdy kandydujący zbiór częsty tylko jeden raz w kolejności wyznaczonej poprzez numery TID. Wadą tego rozwiązania jest wielkość zbiorów kandydujących. Każdy zbiór kandydujący ma tyle elementów, co liczba transakcji, w których występuje. Ponadto po każdej operacji wykluczenia zbiorów małych wszystkie zbiory ponownie muszą być posortowane.

Porównanie działania algorytmów SETM, AIS, AprioriTID oraz Apriori

Przykłady testów przeprowadzonych z wykorzystaniem algorytmów Apriori, AprioriTID, SETM oraz AIS zostały zaczerpnięte z prac [2, 3].

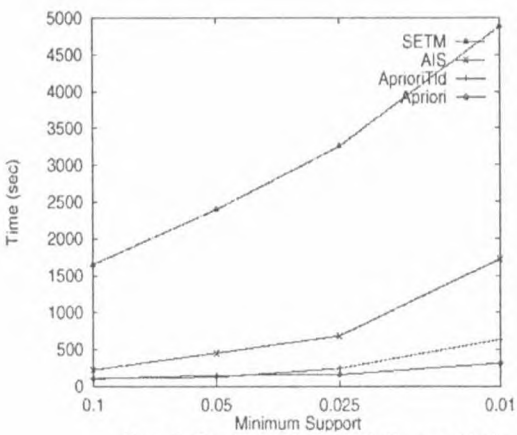
Doświadczenia zostały przeprowadzone na komputerze IBM RS/6000 530II Workstation procesor taktowany zegarem 33 MHz, 64MB RAM, system operacyjny AIX3.2, dysk twardy 2GB SCSI 3,5" z sekwencyjnym transferem ok. 2 MB/s. Dane zostały zaczerpnięte z publikacji grupy IBM Almaden Research Center z 1994 r.



Zakupy w poszczególnych działach	
Wielkość danych	0.65MB
Liczba atrybutów	63
Liczba transakcji	48873
Średni rozmiar transakcji	2,47

Rys. 1. Porównanie metod dla przykładu zakupów w poszczególnych działach

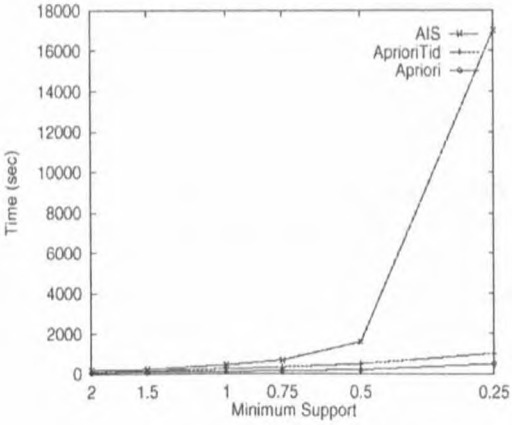
Ź r ó d ł o: [2].



Pojedyncze zlecenia pocztowe	
Wielkość danych	42MB
Liczba atrybutów	15836
Liczba transakcji	$29*10^6$
Średni rozmiar transakcji	2,62

Rys. 2. Porównanie metod dla przykładu pojedynczych zleceń pocztowych

Ź r ó d ł o: [2].



Zlecenia pocztowe poszczególnych osób	
Wielkość danych	42MB
Liczba atrybutów	15836
Liczba transakcji	$29*10^6$
Średni rozmiar transakcji	2,62

Rys. 3. Porównanie metod dla przykładu zleceń pocztowych poszczególnych osób

Źródło: [2].

Literatura

[1] Agrawal R., Iminlinski T., Swami A., *Mining association rules between sets of items in large Databases*, 1993.

[2] Agrawal R., Ramakrishnan S., *Fast Algorithms for Mining Association Rules*, IBM Almaden Research Center, IBM Research Raport RJ9839, June 1994.

[3] http://home.elka.pw.edu.pl/~mciesiel/, 27-12-2003

[4] Mannila H., Toivonen H., *Levelwise Search and Borders of Theoriesin Knowledge Discovery*, Department of Computer Science, Univ. Helisinki, Finland, January 28, 1997.

[5] Ordonez C., Omiecinski E., *Discovering Association Rules based on Image Content*, College of Computing Georgia Institute of Technology, 1999.

[6] Ramakrishnan S., Vu Q., Agrawal R., *Mining Association Rules with Item Constraint*, IBM Almaden Research Center, 1997.

[7] Swami A., Houtsma M., *Set-oriented mining of association rules*, 1993.

Sebastian Szamański, Ryszard Budziński

Exploration methods for association rules and their applications

Summary

This paper will contain a comparison of popular methods of discovering association rules between items in a large databases as a technique of knowledge discovery in databases. Article will introduce to problem of finding association rules, presents a several techniques and their application.