

Algorytmy eksploracji danych: Wykład 11

Copyright by Wojciech Kempa

Politechnika Śląska
Wydział Matematyki Stosowanej

- **Algorytm Eclat** (ang. *Equivalence Class Transformation*) jest typowym przykładem algorytmu eksploracji pionowej.

Copyright by Wojciech Kempa

- **Algorytm Eclat** (ang. *Equivalence Class Transformation*) jest typowym przykładem algorytmu eksploracji pionowej.
- Stosuje się w nim metodę podziału i grupowania zbiorów częstych (w rozłącznych klasach) na podstawie ich wspólnych podzbiorów, by ograniczyć przestrzeń poszukiwań zbiorów częstych.

Copyright by Wojciech Kempa

- **Algorytm Eclat** (ang. *Equivalence Class Transformation*) jest typowym przykładem algorytmu eksploracji pionowej.
- Stosuje się w nim metodę podziału i grupowania zbiorów częstych (w rozłącznych klasach) na podstawie ich wspólnych podzbiorów, by ograniczyć przestrzeń poszukiwań zbiorów częstych. **Copyright by Wojciech Kempa**
- Jak wskazują badania empiryczne, „wąskim gardłem” algorytmu Eclat jest znajdowanie 1- i 2-elementowych zbiorów częstych, stąd w praktyce stosuje się do ich wyznaczenia często inne algorytmy (np. algorytm Apriori).

- **Algorytm Eclat** (ang. *Equivalence Class Transformation*) jest typowym przykładem algorytmu eksploracji pionowej.
- Stosuje się w nim metodę podziału i grupowania zbiorów częstych (w rozłącznych klasach) na podstawie ich wspólnych podzbiorów, by ograniczyć przestrzeń poszukiwań zbiorów częstych. **Copyright by Wojciech Kempa**
- Jak wskazują badania empiryczne, „wąskim gardłem” algorytmu Eclat jest znajdowanie 1- i 2-elementowych zbiorów częstych, stąd w praktyce stosuje się do ich wyznaczenia często inne algorytmy (np. algorytm Apriori).
- Działanie algorytmu Eclat ilustruje poniższy przykład.

Zbiory maksymalne i domknięte (1)

- Jednym ze sposobów optymalizacji procesu wyszukiwania wszystkich zbiorów częstych jest wykorzystanie pojęć maksymalnego i domkniętego zbioru częstego.

Copyright by Wojciech Kempa

Zbiory maksymalne i domknięte (1)

- Jednym ze sposobów optymalizacji procesu wyszukiwania wszystkich zbiorów częstych jest wykorzystanie pojęć maksymalnego i domkniętego zbioru częstego.
- Zbiór X elementów nazywamy **domkniętym** w bazie danych D , jeżeli nie istnieje żaden bezpośredni nadzbiór Y tego zbioru (czyli nadzbiór o liczności o jeden większej), który miałby identyczne wsparcie w zbiorze D , tzn. dla którego $supp(X) = supp(Y)$.

Zbiory maksymalne i domknięte (1)

- Jednym ze sposobów optymalizacji procesu wyszukiwania wszystkich zbiorów częstych jest wykorzystanie pojęć maksymalnego i domkniętego zbioru częstego.
- Zbiór X elementów nazywamy **domkniętym** w bazie danych D , jeżeli nie istnieje żaden bezpośredni nadzbiór Y tego zbioru (czyli nadzbiór o liczności o jeden większej), który miałby identyczne wsparcie w zbiorze D , tzn. dla którego $supp(X) = supp(Y)$.
- Zbiór elementów nazywamy **domkniętym zbiorem częstym** w bazie danych D , jeżeli jest on zbiorem częstym i zbiorem domkniętym.

Zbiory maksymalne i domknięte (1)

- Jednym ze sposobów optymalizacji procesu wyszukiwania wszystkich zbiorów częstych jest wykorzystanie pojęć maksymalnego i domkniętego zbioru częstego.
- Zbiór X elementów nazywamy **domkniętym** w bazie danych D , jeżeli nie istnieje żaden bezpośredni nadzbiór Y tego zbioru (czyli nadzbiór o liczności o jeden większej), który miałby identyczne wsparcie w zbiorze D , tzn. dla którego $supp(X) = supp(Y)$.
- Zbiór elementów nazywamy **domkniętym zbiorem częstym** w bazie danych D , jeżeli jest on zbiorem częstym i zbiorem domkniętym.
- Wykorzystanie domkniętych zbiorów częstych w procesie wyszukiwania wszystkich zbiorów częstych jest związane z pewną specyficzną własnością tak rozumianego domknięcia.

Zbiory maksymalne i domknięte (2)

- Mianowicie, aby stwierdzić, czy zbiór X jest zbiorem częstym, należy znaleźć najmniejszy domknięty zbiór częsty, który jest nadzbiorem X .

Copyright by Wojciech Kempa

Zbiory maksymalne i domknięte (2)

- Mianowicie, aby stwierdzić, czy zbiór X jest zbiorem częstym, należy znaleźć najmniejszy domknięty zbiór częsty, który jest nadzbiorem X .
- Jeżeli taki nadzbiór nie istnieje, to X nie jest zbiorem częstym.

Copyright by Wojciech Kempa

Zbiory maksymalne i domknięte (2)

- Mianowicie, aby stwierdzić, czy zbiór X jest zbiorem częstym, należy znaleźć najmniejszy domknięty zbiór częsty, który jest nadzbiorem X .
- Jeżeli taki nadzbiór nie istnieje, to X nie jest zbiorem częstym.
- W praktyce własność ta ma znaczenie przede wszystkim w przypadku dużych baz danych. Liczba domkniętych zbiorów częstych może być nawet o kilka rzędów mniejsza niż np. liczba wszystkich zbiorów kandydujących, generowanych w algorytmie Apriori.

Zbiory maksymalne i domknięte (2)

- Mianowicie, aby stwierdzić, czy zbiór X jest zbiorem częstym, należy znaleźć najmniejszy domknięty zbiór częsty, który jest nadzbiorem X .
- Jeżeli taki nadzbiór nie istnieje, to X nie jest zbiorem częstym.
- W praktyce własność ta ma znaczenie przede wszystkim w przypadku dużych baz danych. Liczba domkniętych zbiorów częstych może być nawet o kilka rzędów mniejsza niż np. liczba wszystkich zbiorów kandydujących, generowanych w algorytmie Apriori.
- Zbiór X elementów nazywamy **maksymalnym zbiorem częstym** w bazie danych D , jeżeli jest on zbiorem częstym oraz nie istnieje żaden nadzbiór Y tego zbioru, który byłby zbiorem częstym.

- Zbiór X nazywamy **maksymalnym domkniętym zbiorem częstym**, jeżeli jest on zarówno maksymalnym zbiorem częstym jak i domkniętym zbiorem częstym w D .

Copyright by Wojciech Kempa

Zbiory maksymalne i domknięte (3)

- Zbiór X nazywamy **maksymalnym domkniętym zbiorem częstym**, jeżeli jest on zarówno maksymalnym zbiorem częstym jak i domkniętym zbiorem częstym w D .
- Na podstawie maksymalnych domkniętych zbiorów częstych łatwo wygenerować wszystkie zbiory częste w danej bazie D : są to po prostu **wszystkie ich podzbiory**.

Zbiory maksymalne i domknięte (3)

- Zbiór X nazywamy **maksymalnym domkniętym zbiorem częstym**, jeżeli jest on zarówno maksymalnym zbiorem częstym jak i domkniętym zbiorem częstym w D .
- Na podstawie maksymalnych domkniętych zbiorów częstych łatwo wygenerować wszystkie zbiory częste w danej bazie D : są to po prostu **wszystkie ich podzbiory**.
- Zauważmy także, że jeśli X jest maksymalnym zbiorem częstym, to musi być również zbiorem domkniętym, bo niemożliwe jest, by istniał jego bezpośredni nadzbiór mający identyczne wsparcie.

Wśród wielu różnych algorytmów odkrywania domkniętych zbiorów częstych wypada wymienić CLOSET, CLOSET+, CHARM, FPClose, TITANIC czy też LCM. Za najbardziej efektywny z nich uchodzi algorytm CLOSET+, stworzony w 2003 roku (J. Wang, J. Han, J. Pei). W algorytmie tym wykorzystuje się dwie ważne techniki:

- łączenie obiektów;
- przycinanie podzbiorów.

- **Łączenie obiektów** jest konsekwencją następującej własności. Załóżmy, że zbiór X jest zbiorem częstym. Jeżeli każda transakcja analizowanej bazy danych D zawierająca X zawiera też pewien zbiór Y , lecz nie zawiera żadnego bezpośredniego nadzbioru zbioru Y , to tworzymy domknięty zbiór częsty $X \cup Y$ i pomijamy w analizie wszystkie zbiory zawierające X .

Copyright by Wojciech Kempa

- **Łączenie obiektów** jest konsekwencją następującej własności. Załóżmy, że zbiór X jest zbiorem częstym. Jeżeli każda transakcja analizowanej bazy danych D zawierająca X zawiera też pewien zbiór Y , lecz nie zawiera żadnego bezpośredniego nadzbioru zbioru Y , to tworzymy domknięty zbiór częsty $X \cup Y$ i pomijamy w analizie wszystkie zbiory zawierające X .
- Rzeczywiście, jeżeli zbiór X jest zbiorem częstym i każda transakcja zawierająca X zawiera też zbiór Y , to zbiór $X \cup Y$ będzie domkniętym zbiorem częstym. Zakładamy bowiem, że transakcje zawierające $X \cup Y$ nie zawierają żadnego bezpośredniego nadzbioru zbioru Y . W konsekwencji nie może istnieć żaden bezpośredni nadzbiór $X \cup Y$, który miałby identyczne wsparcie.

- **Łączenie obiektów** jest konsekwencją następującej własności. Załóżmy, że zbiór X jest zbiorem częstym. Jeżeli każda transakcja analizowanej bazy danych D zawierająca X zawiera też pewien zbiór Y , lecz nie zawiera żadnego bezpośredniego nadzbioru zbioru Y , to tworzymy domknięty zbiór częsty $X \cup Y$ i pomijamy w analizie wszystkie zbiory zawierające X .
- Rzeczywiście, jeżeli zbiór X jest zbiorem częstym i każda transakcja zawierająca X zawiera też zbiór Y , to zbiór $X \cup Y$ będzie domkniętym zbiorem częstym. Zakładamy bowiem, że transakcje zawierające $X \cup Y$ nie zawierają żadnego bezpośredniego nadzbioru zbioru Y . W konsekwencji nie może istnieć żaden bezpośredni nadzbiór $X \cup Y$, który miałby identyczne wsparcie.
- **Przycinanie podzbiorów** wymaga stworzenia specjalnej struktury, zwanej **SE-drzewem** (ang. *Set-Enumeration Tree* ($S-ET$)), która jest następnie eksplorowana.

- SE-drzewo „numeruje” wszystkie elementy zbioru potęgowego (zbioru wszystkich pozdbiorów) $\mathcal{P}(L)$, gdzie L oznacza zbiór produktów występujących w bazie danych o transakcjach.

Copyright by Wojciech Kempa

- SE-drzewo „numeruje” wszystkie elementy zbioru potęgowego (zbioru wszystkich pozdbiorów) $\mathcal{P}(L)$, gdzie L oznacza zbiór produktów występujących w bazie danych o transakcjach.
- W pierwszym kroku numerowane są wszystkie elementy zbioru L za pomocą funkcji

$$ind : L \rightarrow \mathbb{N} \quad (1)$$

Copyright by Wojciech Kempa

- SE-drzewo „numeruje” wszystkie elementy zbioru potęgowego (zbioru wszystkich podzbiorów) $\mathcal{P}(L)$, gdzie L oznacza zbiór produktów występujących w bazie danych o transakcjach.
- W pierwszym kroku numerowane są wszystkie elementy zbioru L za pomocą funkcji

$$\text{Copyright by Wojciech Kempa} \quad ind : L \rightarrow \mathbb{N}. \quad (1)$$

- Następnie dla każdego podzbioru $S \subseteq L$ definiujemy jego obraz SE-drzewa za pomocą funkcji *view* działającej w następujący sposób:

$$view(ind, S) \stackrel{def}{=} \{l \in L : ind(l) > \max_{l' \in S} ind(l')\}. \quad (2)$$

SE-drzewo

Niech K będzie rodziną zbiorów domkniętych ze względu na operację zawierania zbiorów (\subseteq). T nazywamy drzewem enumeracji zbiorów (SE-drzewem) dla rodziny K wtedy i tylko wtedy, gdy spełnione są następujące dwa warunki:

- 1 korzeniem drzewa T jest zbiór pusty;
- 2 następnikami wierzchołka S_i w drzewie T są wierzchołki postaci

$$\{S \cup \{i\} \in K : i \in \text{view}(\text{ind}, S)\}.$$

Dla przykładu, w przypadku wierzchołka $\{2\}$ mamy

$$\begin{aligned} \text{view}(\text{ind}, \{2\}) &= \{l \in L : \text{ind}(l) > \max_{l' \in \{2\}} \text{ind}(l')\} \\ &= \{l \in L : \text{ind}(l) > 2\} = \{3, 4\}, \end{aligned}$$

a zatem wierzchołkami potomnymi wierzchołka $\{2\}$ są wierzchołki $\{2\} \cup \{3\} = \{2, 3\}$ oraz $\{2\} \cup \{4\} = \{2, 4\}$.

Wykorzystanie techniki przycinania podzbiorów (związanej z konstrukcją SE-drzewa) uzasadnione jest następującym twierdzeniem.

Przycinanie podzbiorów (J. Wang, J. Han, J. Pei, 2003)

Niech X będzie zbiorem częstym. Jeśli X jest podzbiorem właściwym pewnego domkniętego zbioru częstego Y oraz zachodzi $\text{supp}(X) = \text{supp}(Y)$, to ani zbiór X , ani żaden z jego wierzchołków potomnych w SE-drzewie nie mogą być domkniętymi zbiorami częstymi

W CLOSET+ opracowano **hybrydową metodę projekcji drzewa**, która buduje warunkowe przewidywane bazy danych poprzez fizyczną projekcję drzewa „od dołu do góry” dla „gęstych” zbiorów danych i pseudoodwzorowanie drzewa „od góry do dołu” dla „rzadkich” zbiorów danych. Ogólny przebieg algorytmu CLOSET+ jest następujący.

- 1 Znajdujemy 1-elementowe zbiory częste w bazie danych D i sortujemy je według malejących wartości wsparcia. Posortowana lista zbiorów częstych tworzy zmodyfikowaną bazę danych \overline{D} .
- 2 Konstruujemy FP-drzewo dla bazy danych \overline{D} .
- 3 Oceniając wizualnie skonstruowane FP-drzewo, podejmujemy decyzję, czy zbiór (bazę) danych \overline{D} potraktować jako zbiór „gęsty” czy zbiór „rzadki”. Dla „gęstego” zbioru danych wybieramy metodę „od dołu do góry” fizycznej projekcji drzewa, natomiast dla „rzadkiego” zbioru danych stosujemy metodę „od góry do dołu” pseudoodwzorowania drzewa.

4. Korzystając z zasady „dziel i rządź” oraz wyszukiwania w głąb, budujemy FP-drzewo dla domkniętych zbiorów częstych (w sposób „od góry do dołu” dla „rzadkich” zbiorów danych lub „od dołu do góry” dla „gęstych” zbiorów danych).
5. Konstruując FP-drzewo używamy metody **łączenia elementów (obiektów)** oraz **przycinania podzbiorów** w celu zmniejszenia przestrzeni wyszukiwania.
6. Tworzymy warunkowe FP-drzewa. Dla „gęstych” zbiorów danych warunkowe projekcyjne bazy danych budujemy tak samo jak w algorytmie FP-Growth, natomiast dla zbiorów „rzadkich” wykorzystujemy metodę pseudoprojekcji „od góry do dołu”.
7. Kontynuujemy ten proces tak długo, dopóki nie będzie już możliwe „rozszerzanie” zbiorów i znajdowanie nowych domkniętych zbiorów częstych.