Piotr Dubiel

Projekt MED-P3 - algorytm CHARM

Metody eksploracji danych w odkrywaniu wiedzy.

Opis zadania

Celem projektu jest zaimplementowanie algorytmu wyznaczania reguł decyzyjnych o minimalnych poprzednikach, które są częstymi zbiorami zamkniętymi. Do wyznaczania zbiorów zamkniętych używany jest zmodyfikowany algorytm CHARM.

Założenia

Funkcjonalne

- Dane wejściowe są odczytywane z pliku
- Dane wyjściowe są zapisywane w dwóch formatach: surowe dane w pliku csv oraz w sformatowanym pliku txt
- Aplikacja może używać 3 strategii sortowania identyfikatorów: rosnącą, malejącą i nieuporządkowaną.
- Aplikacji można przekazać minimalne wsparcie reguł (sup(r) >= min_sup)

Niefunkcjonalne

- Język programowania: C++.
- Obsługiwane systemy operacyjne: Linux, Windows, OSX.
- Aplikacja konsolowa.

Dane wejściowe

- Plik z danymi w formacie używanym w zbiorze *car.* Każda transakcja jest oddzielona znakiem nowej linii, a atrybuty oddzielone są przecinkami
    - Przykładowy plik:

,,H,,P
,M,H,,P
,,,,P
,,,T,P
S,,,,P
,M,,,P
S,M,,T,P

- (Opcjonalne) Plik z nagłówkiem zawierający nazwy atrybutów w formacie używanym w zbiorze *car.*
    - Przykładowy plik:

      unacc, acc, good, vgood
      buying: vhigh, high, med, low.
      maint: vhigh, high, med, low.
      doors: 2, 3, 4, 5more.
      persons: 2, 4, more.
      lug_boot: small, med, big.
      safety: low, med, high.

      Nazwa atrybutu to ciąg znaków do znaku dwukropka. Jeśli nie ma takiego znaku w linii, to do zbioru wartości przypisywana jest nazwa “decision”.

Parametry wywołania aplikacji

charm [opcje] PLIK_DANYCH NAZWA_PLIKU_WYNIKOWEGO

Opcje

- -a, --asc Sortuj identyfikatory według rosnącego wsparcia.
- -d, --desc Sortuj identyfikatory według malejącego wsparcia.
- -h, --header PLIK Plik z nagłówkiem,
- -s, --support SUP Minimalne wsparcie reguł. Domyślnie minimalne wsparcie to 0
- -c, --class COL Indeks atrybutu decyzyjnego z przedziału [0..liczba atrybutów - 1]. Domyślnie - ostatni atrybut ze zbioru danych.

Wynik

Wynikiem działania aplikacji są dwa pliki o nazwach powstałych przez doklejenie do argumentu NAZWA_PLIKU_WYNIKOWEGO rozszerzeń .txt i .csv.
Pliki te zawierają wygenerowane reguły odpowiednio w formacie przyjaznym dla człowieka w pliku txt i przygotowane do zaimportowania do arkusza kalkulacyjnego w formacie csv.

Plik .txt

Plik jest podzielony na sekcje według następnika reguły. W każdej sekcji występuje lista par atrybut-wartość należących do poprzednika reguły.

Przykład

==========
Attribute 4: P

Attribute 0(S), Attribute 1(M), Attribute 3(T)

==========
Attribute 4: N

Attribute 0(S), Attribute 2(H),

Jeśli został podany plik nagłówkowy, to nazwy atrybutów są zastąpione tymi występującymi w pliku.

Plik .csv

Pierwszy wiersz zawiera nazwy wszystkich atrybutów. W kolejnych wierszach występują reguły z atrybutem decyzyjnym na końcu.

Przykład

Attribute 0;Attribute 1;Attribute 2;Attribute 3;Attribute 4
S;M;;T;P
S;;H;;N

Implementacja

Wczytywanie danych

Jeśli podano plik z nagłówkiem, generowane są identyfikatory wartości atrybutów w kolejności podania ich w pliku. W przeciwnym przypadku wczytywany jest plik z danymi, a identyfikatory są przypisywane w kolejności pojawienia się wartości w danych.
Wybranie strategii sortowania wymaga policzenia wsparcia każdego z identyfikatorów, posortowania ich według wsparcia. Następnie ponownie przypisywane są identyfikatory do wartości atrybutów.

Modyfikacja algorytmu CHARM

Modyfikacja algorytmu CHARM [1] użyta w aplikacji polega na dodaniu do każdego węzła dwóch, dodatkowych atrybutów. Pierwszy to klasa decyzyjna pierwszej transakcji występującej w węźle. Drugi to atrybut logiczny mówiący, czy węzeł należy do jednej klasy decyzyjnej.

Budowanie reguł ze zbiorów zamkniętych

Dla każdego zbioru zamkniętego, który spełnia warunek jednolitej klasy decyzyjnej zostaje dodany do zbioru reguł, o ile nie istnieje już dodana reguła będąca jego podzbiorem oraz dająca tę samą decyzję.
Na tym etapie następuje przejście z identyfikatorów liczbowych na nazwy wartości podane w zbiorze danych. Zapisywane są też nazwy atrybutów, jeśli podano nagłówek.

Przykładowy przebieg działania algorytmu

Wykrywane będą reguły o minmalnym wsparciu >= 1, dla zbioru danych:

1			H		P
2		M	H		P
3					P
4				T	P
5	S				P
6		M			P
7	S	M		T	P
8		M	H	T	P
9					P
10	S		H		N
11	S		H	T	N
12				T	N
13	S	M	H		N
14		M	H	T	N

Wywołane polecenie

$ ./charm -s 1 test.data test_results

Wynik

Zawartość pliku test_results.txt to:
==========
Attribute 4: P

Attribute 0(S), Attribute 1(M), Attribute 3(T)

==========
Attribute 4: N

Attribute 0(S), Attribute 2(H),

Wygenerowane zostały dwie reguły:

- {S, M, T} -> {P}
- {S, H} -> {N}

Testy wydajnościowe

Badanie wydajności algorytmu przeprowadzono na następujących zbiorach danych:

- car,
- mushroom,
- nursery.

Brak uporządkowania

Minimalne wsparcie	Car	Mushroom	Nursery
1	0.391ms		9.9784s
2	0.298ms		8.833s
5	0.201ms		6.504s
10	0.189ms	3m16.202s	5.3312s
20	0.142ms	2m57.979s	4.329s
50	0.105ms	2m35.226s	3.38s
100	0.108ms	2m23.150s	2.8384s
150		2m13.499s	
200		2m7.167s	
300		1m54.754s	
500		1m34.790s	
700		1m20.436s	
1000		1m0.818s	

1500		0m43.460s	

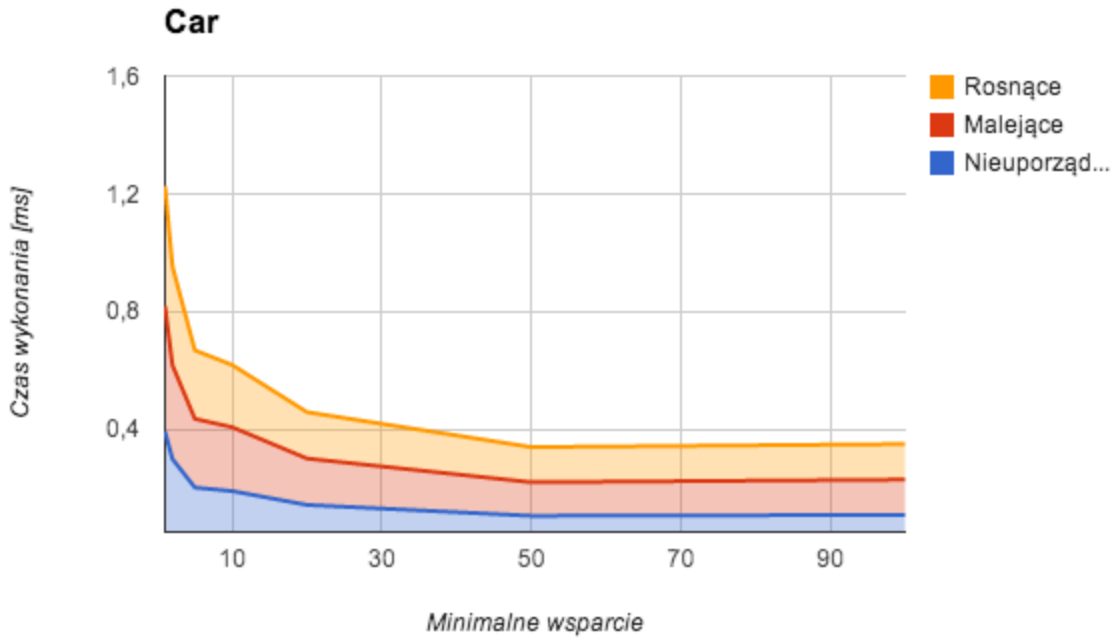
Malejące wsparcie

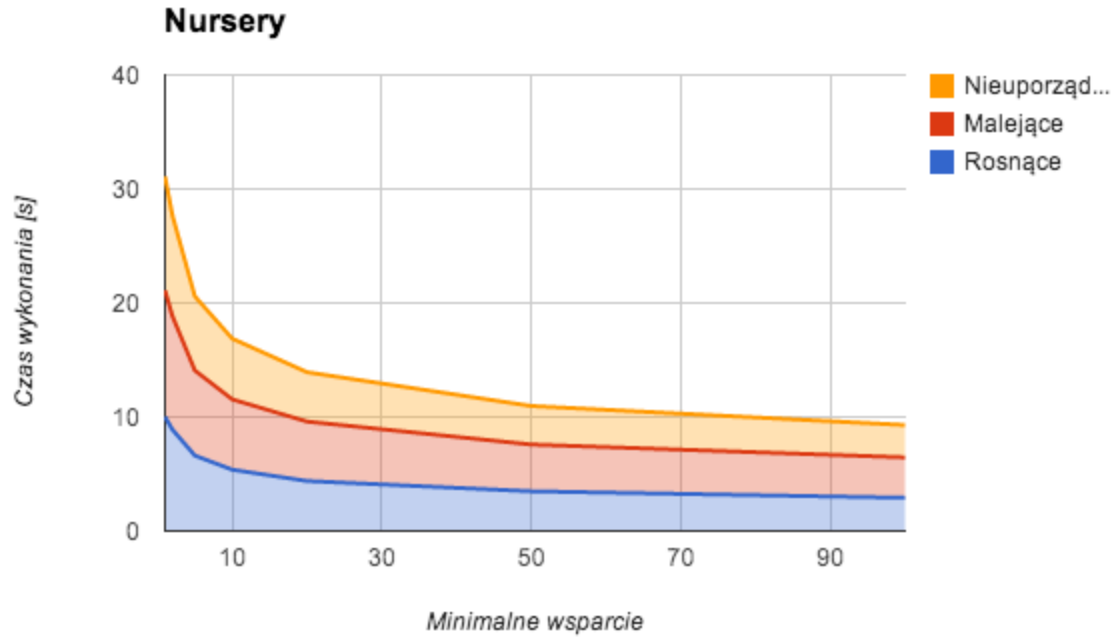
Minimalne wsparcie	Car	Mushroom	Nursery
1	0.4267ms		11.1016s
2	0.3187ms		9.948s
5	0.2335ms		7.4644s
10	0.2172ms	3m57.434s	6.1682s
20	0.1577ms	3m31.999s	5.2128s
50	0.1141ms	3m9.248s	4.1016s
100	0.12ms	2m38.073s	3.5416s
150		2m1.092s	
200		1m55.428s	
300		1m39.210s	
500		1m18.803s	
700		1m5.313s	
1000		0m47.916s	
1500		0m36.020s	

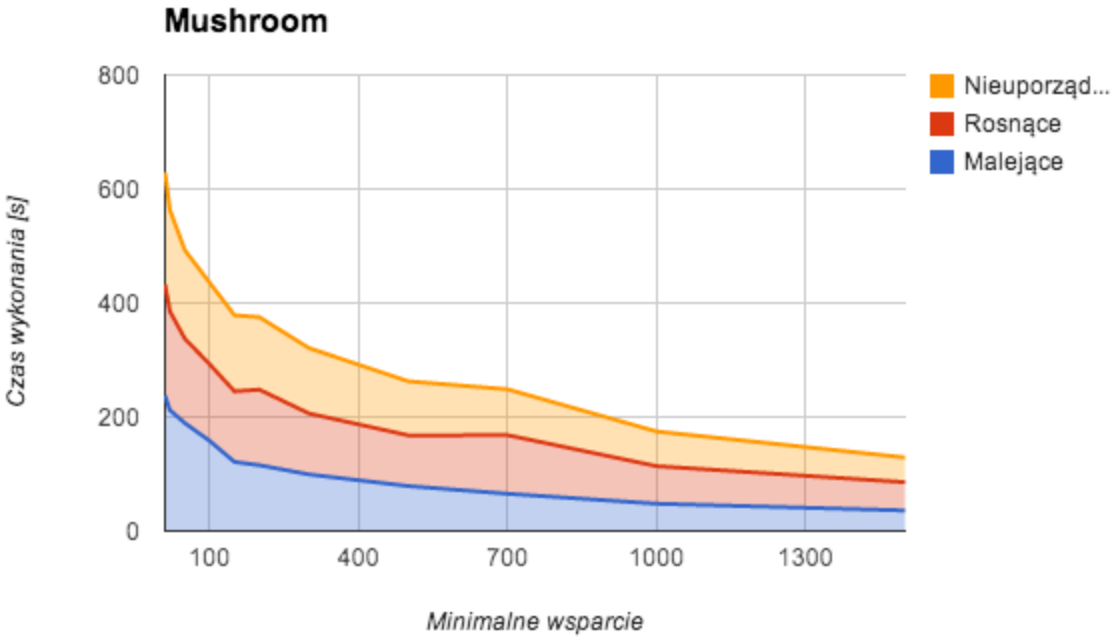
Rosnące wsparcie

Minimalne wsparcie	Car	Mushroom	Nursery
1	0.4079ms		10.014s
2	0.3361ms		8.891s
5	0.233ms		6.6138s
10	0.2115ms	3m15.203s	5.376s
20	0.1576ms	2m53.152s	4.3858s

50	0.1195ms	2m28.206s	3.4836s
100	0.1209ms	2m14.331s	2.9158s
150		2m3.913s	
200		2m12.588s	
300		1m46.928s	
500		1m29.025s	
700		1m42.784s	
1000		1m5.835s	
1500		0m49.579s	




Nursery
40
30
20
10
0
Czas wykonania [s]
10
30
50
70
90
Minimalne wsparcie
Nieuporząd...
Malejące
Rosnące


Mushroom
800
600
400
200
0
Czas wykonania [s]
100
400
700
1000
1300
Minimalne wsparcie
Nieuporząd...
Rosnące
Malejące

Wnioski

Opisana w [1] metoda sortowania według rosnącego wsparcia okazała się wydajniejsza tylko dla zbioru *mushroom.* Podejrzewam, że jest spowodowane małym rozmiarem pozostałych zbiorów, przez co szybkość działania samego algorytmu stała się porównywalna z czasem wczytywania i zapisywania danych. Możliwe, że etap sortowania elementów według wsparcia wydłużał czas działania na tyle, że nie da się zauważyć zysku z sortowania.
Sama implementacja algorytmu nie jest jeszcze optymalna ze względu na długi czas sprawdzania relacji zawierania zbiorów transakcji.

Bibliografia

[1] CHARM: An Efficient Algorithm for Closed Itemset Mining, Zaki M., Hsiao C.