Dyskretyzacja atrybutów w odkrywaniu ilościowych reguł asocjacyjnych

Jacek Sosnowski

14 stycznia 2015

Politechnika Warszawska
Wydział Elektroniki i Technik Informacyjnych
Instytut Informatyki

a

Spis treści

1 Wstęp

Otaczająca wszystkich rzeczywistość coraz częściej przenika się ze światem wirtualnym. Innymi słowy, Internet oraz komputery i narzędzia z nimi związane, już dawno stały się niezbędnym elementem codzienności. Próby przewidzenia dalszego rozwoju obecnej sytuacji spędzają sen z powiek naukowcom i wizjonerom. Niemniej jednak już teraz obserwuje się ciekawy skutek ekspansji technik komputerowych. Jest nim powstawanie społeczeństw informacyjnych. Takim mianem określa się cywilizację, w której szczególnym dobrem ekonomicznym staje się informacja. To ona pośrednio zaczyna być podstawą dochodu narodowego, a przez to głównym źródłem utrzymania wielu jednostek. Obecnie powszechnie akceptowalne jest stwierdzenie, że informacja jest bardzo cenna. Z tego powodu następuje ciągły rozwój usług informatycznych. Według autorów [17] z socjoekonomicznego punktu widzenia, sektor informacji dzieli się na: produkcję, przetwarzanie oraz przemysł dystrybucji informacji. Ostatni z nich obejmuje teleinformatykę, a więc przesyłanie danych. Natomiast produkcja odbywa się często w sposób naturalny. Kupując towary czy korzystając z dostępnych usług, konsumenci są producentami informacji. W tej pracy najwięcej odniesień będzie do przetwarzania, które jest naturalną konsekwencją gromadzenia danych.

Wyniki najnowszych badań statystycznych mogą wskazywać na prawdziwość doniesień o rozwoju wspomnianego typu społeczeństwa nawet w pięknym kraju nad Wisłą. Według danych GUS [18] dla Polski *„w 2012 r. liczba firm z sektora ICT wzrosła w stosunku do 2009 r. o 25,6 % (...) natomiast liczba pracujących w tym sektorze – o 11,6 %”*. Dodatkowo w tym samym okresie przychody netto dla tego sektora zwiększyły się o 30,8 %. Oznacza to wzrost zainteresowania świadczonymi usługami.

Efektem ubocznym dojrzewania społeczeństwa informatycznego jest wytwarzanie ogromnej ilości danych. Im więcej aspektów codziennego życia jest wspomagane przez technologie, tym więcej danych można zebrać. Przykładowo, jeszcze kilka dekad temu zwykłe zakupy były prostą wymianą dóbr (z wykorzystaniem środka płatniczego). Dzisiaj każda taka transakcja jest rejestrowana w systemie komputerowym, a następnie zapisywana w przygotowanej bazie danych. Jest to osiągalne dzięki postępowi techniki. A przede wszystkim dzięki łatwo dostępnej i taniej pamięci dyskowej. Realne jest więc gromadzenie dużych ilości danych, co czyni większość dzisiejszych instytucji.

Z upływem czasu coraz tańsza staje się również moc obliczeniowa. Taki kierunek zmian wprost świetnie wpisuje się w potrzeby współczesnej cywilizacji. Gdyż daje możliwość efektywnej analizy potężnych baz. Zadaniem tym zajmuje się dziedzina eksploracji danych. Dzięki jej metodom możliwe jest odkrycie interesujących zależności oraz nieznanej struktury zawartej w

zebranych danych. Można też pokusić się o stwierdzenie, że ta gałąź informatyki pozwala na wydobycie „wiedzy" ukrytej w posiadanych bazach.

Najbardziej popularną metodą eksploracji danych nieustannie jest wyszukiwanie reguł asocjacyjnych. Istnieje wiele algorytmów rozwiązujących to zadanie. Niestety większość z nich wymaga, by dane wejściowe były zorganizowane w postaci tabeli o wartościach wyłącznie binarnych. Natomiast współcześnie gromadzone dane zdecydowanie częściej mają bogatsze dziedziny wartości. Często spotykane są zbiory, które zawierają liczby rzeczywiste. W takim przypadku unikalnych wartości, zamiast dwóch może być nawet nieskończenie wiele. O takie właśnie zbiory opierają się ilościowe reguły asocjacyjne. Przykładem może być reguła:

$$waga : [80, 100] \wedge gorne\_cisnienie : [120, 140] \Rightarrow ryzyko = 1$$

Powyższa reguła jest oczywiście sztucznym przykładem, ale wiedza zawarta w takich implikacjach może być bezcenna. Dlatego celem niniejszej pracy jest zbadanie możliwości eksploracji ilościowych reguł asocjacyjnych. W tym celu połączone zostaną siły dwóch dyscyplin eksploracji danych: analizy skupień oraz odkrywania reguł asocjacyjnych. Cała praca nie aspiruje do miana przeglądu tych dziedzin. Przedstawia natomiast specyficzny tok zdobywania wiedzy i technologii dla rozwiązania problemu przedstawionego jako cel pracy. Dokumentuje tylko i wyłącznie te fragmenty teorii eksploracji danych, które aktywnie i bezpośrednio wpłynęły na rozwiązanie końcowe. Wszelkie nawiązania, które nie wyczerpują w pełni tematyki, ale dla kompletności rozważań znajdują się w tekście, zostały opatrzone komentarzem wskazującym dokładniejsze źródło.

2 Eksploracja reguł asocjacyjnych

Eksploracja reguł asocjacyjnych (ang. *association rule mining*) jest jednym z ważniejszych filarów eksploracji danych (ang. *data mining*). Głównym celem tej dziedziny informatyki jest wydobycie z dużych zbiorów danych informacji wyższego poziomu abstrakcji. Oznacza to odkrycie relacji czy struktur zawartych w analizowanym zestawie, najlepiej jeśli będą nieznane oraz interesujące. Manualna analiza nawet niewielkich baz danych może sprawiać problemy, a wraz ze wzrostem ich wymiarów, takie badanie staje się praktycznie niemożliwe dla człowieka. Dodatkowo najbardziej interesujące relacje ujawniają się dopiero w liczniejszych kolekcjach danych. Dobrym wprowadzeniem do wspomnianej tematyki jest słynna już praca [1]. Jednak dla kompletności rozważań zostanie przedstawiony tu bardzo krótki zarys teoretyczny problemu eksploracji danych.

Baza danych D to zbiór transakcji (próbek) $D = \{T_1, T_2, T_3, \ldots, T_n\}$. Każda próbka T_j ($j \in [1, n]$) zawiera ustaloną liczbę składowych opisanych zbiorem atrybutów $I = \{i_1, i_2, i_3, \ldots, i_k\}$.

Odkrycie związków pomiędzy atrybutami umożliwia eksploracja reguł asocjacyjnych. Ogólnie przez pojęcie reguły rozumie się wyrażenie postaci:

$$X \Rightarrow Y$$
$$X \in I,\, Y \in I,\, X \cap Y = \emptyset$$

Gdzie X oraz Y to zdarzenia. Przy czym kiedy w bazie występuje X to z pewnym poziomem ufności wystąpi też Y. Ten poziom nazywany jest wiarygodnością, lub ufnością (ang. *confidence*). Biorąc pod uwagę definicję bazy, każde ze zdarzeń, zarówno X jak i Y to podzbiory atrybutów I. Oznacza to, że reguła asocjacyjna opisuje relację pomiędzy występowaniem atrybutów X, a pojawianiem się atrybutów Y.

2.1 Rodzaje atrybutów

Mówiąc potocznie, bazę danych można potraktować jako tabelę. Wtedy każdy wiersz to rekord, próbka, lub po prostu element bazy. Natomiast każda kolumna nazywana jest cechą, atrybutem albo własnością obiektów przechowywanych w bazie. Każda cecha może zawierać wartości różnego typu, które można podzielić na kilka klas - zgodnie z opisem w rozdziałach 2.1.1 oraz 2.1.2.

2.1.1 Atrybuty kategoryczne

Atrybuty kategoryczne, dzieli się na *nominalne* oraz *porządkowe*[1]:

1. **Atrybuty nominalne** (dyskretne, skończone, wyliczeniowe) – wartości tworzą przestrzeń skończoną i zazwyczaj niewielką. Nie istnieje porządek pomiędzy wartościami atrybutu. Dobrym przykładem może być kolor pewnych produktów zawartych w bazie: biały, czerwony, zielony, niebieski i czarny. Lista dostępnych wartości dla danego atrybutu jest zazwyczaj w pełni znana jeszcze przed analizą danych.

2. **Atrybuty binarne** – są szczególnym przypadkiem atrybutów nominalnych, ponieważ przyjmują tylko dwie wartości. Zazwyczaj jest to 0 i 1. Najczęściej (choć nie zawsze) oznaczają występowanie, bądź brak danego atrybutu w transakcji. W [1] w rozdziale *„Formal Model"* można znaleźć formalną definicję atrybutów binarnych w kontekście bazy danych.

[1] Informacje zaczerpnięte między innymi z [5]

3. **Atrybuty porządkowe** (ang. *ordinal attributes*) – mają podobnie jak nominalne, skończoną oraz niezbyt liczną dziedzinę. Różnica polega na możliwości uporządkowania wartości tych atrybutów. Znajomość porządku nie implikuje jednak znajomości odległości pomiędzy wartościami. Przykładem może być następująca przestrzeń: bardzo mało, niewiele, dużo, bardzo dużo. Bez dodatkowych informacji ilościowych opisujących poszczególne dostępne wartości nie można określić relacji odległości pomiędzy próbkami tego atrybutu.

2.1.2 Atrybuty numeryczne

Atrybuty ciągłe (*numeryczne*) – definicja przestrzeni wartości dla takich atrybutów oparta jest na dobrze określonym zbiorze liczbowym, np. liczby rzeczywiste czy całkowite. Z tego powodu ilość tych wartości jest nieskończona (nie można przewidzieć ile różnych próbek znajduje się akurat w konkretnej bazie danych). Dodatkowo znany jest porządek pomiędzy wartościami oraz zazwyczaj dystans między nimi (według wybranej miary odległości). Takie atrybuty najczęściej są spotykane przy opisach właściwości fizycznych obiektów, czego przykładami mogą być: masa, temperatura, długość, wzrost, itp. Już ta krótka lista ukazuje, że pomimo teoretycznie nieskończonej liczby wartości jaką prezentują atrybuty numeryczne, czasami istnieje możliwość uściślenia dziedziny. Typowym wzorem jest wiek, który z punktu widzenia bazy danych może przyjmować dowolne dodatnie wartości, lecz w rzeczywistości nie spotykamy się z ludźmi żyjącymi 200 czy 300 lat. Tak zwany kontekst danych przechowywanych przez atrybut może pomóc przy manualnym grupowaniu przechowywanych w nim wartości.

2.2 Wsparcie i ufność

Z eksploracją reguł asocjacyjnych nierozłącznie związane są dwa pojęcia: wsparcia oraz ufności. Oba będą często gościły w niniejszej pracy, dlatego zostaną krótko przedstawione.

Wsparcie (ang. *support*) dla zbioru atrybutów X to liczba (lub procent) wszystkich transakcji bazy D, które zawierają atrybuty X. Oznaczane jest poprzez $sup(X)$.

Prostymi słowami można stwierdzić, że jest to miara, która określa jak często w bazie danych pojawiają się cechy ze zbioru X. Patrząc odwrotnie na powyższą definicję należy wprowadzić pojęcie wspierania:

Transakcja T_j wspiera zdarzenie X, wtedy kiedy zawiera wszystkie atrybuty zawarte w X (może zawierać ich więcej niż posiada X).

W tej pracy zostało założone, iż dziedziną funkcji wsparcia jest zbiór $\langle 0, 1\rangle$. Zatem miara ta wyznacza „ułamek" wszystkich transakcji. Terminu *wsparcie* używa się najczęściej nie w odniesieniu do zbioru atrybutów lecz w nawiązaniu do reguły asocjacyjnej:

Wsparcie reguły asocjacyjnej $A \Rightarrow B$ ($sup(A \Rightarrow B)$) to wsparcie zbioru $A \cup B$ ($sup(A \cup B)$).

Ta miara wskazuje jak silna jest przedstawiona reguła w konkretnym zbiorze danych. Im większa jest wartość omawianej funkcji tym częściej zbiór (na którym oparta jest reguła) pojawia się w bazie. W tym miejscu należy zaznaczyć, że to użytkownik lub badacz decyduje jak „silnych" reguł potrzebuje. W większości algorytmów taka decyzja jest respektowana poprzez ustalenie minimalnej wartości wsparcia. Celem jest nie generowanie reguł, dla których wsparcie znajduje się poniżej uzgodnionego progu.

Drugim niezbędnym pojęciem przy eksploracji jest **ufność** (lub **wiarygodność**, ang. *confidence*).

Ufność reguły asocjacyjnej $X \Rightarrow Y$ to część liczby transakcji, które wspierają Y wśród tych, które wspierają X. Oznaczana jest przez $conf(X \Rightarrow Y)$

$$\mathrm{conf}(X \Rightarrow Y) = \frac{\mathrm{supp}(X \cup Y)}{\mathrm{supp}(X)}$$

Sam problem wydobywania reguł jest zazwyczaj opisywany właśnie za pomocą tych dwóch miar poprzez podanie minimalnych ich wartości. To znaczy, że pożądane są tylko te asocjacje, których wsparcie jest wyższe niż minimalne wsparcie, a ufność jest większa niż minimalna ufność. Te wartości graniczne są oznaczane odpowiednio *minSup* oraz *minConf* (z ang. *minimumSupport* oraz *minimumConfidence*). Oba ograniczenia są ustawiane przez użytkownika, w celu sterowania procesem eksploracji.

Z przedstawionymi powyżej definicjami związane jest jeszcze jedno pojęcie. Każda reguła asocjacyjna jest stworzona w oparciu o pewien **zbiór częsty** (ang. *frequent itemset*). Jest to zbiór atrybutów, których wsparcie jest większe niż minimalne wsparcie minSup. Innymi słowy, problem eksploracji reguł można sprowadzić najpierw do zadania wyszukania zbiorów częstych. Następnie każdy z nich jest wykorzystywany do budowy kilku implikacji.

Wszystkie odsłonięte tutaj pojęcia będą wykorzystywane w kolejnych rozdziałach tej pracy. Natomiast ich dokładniejsze przedstawienie znajduje się w pracach [1, 3].

3 Ilościowe reguły asocjacyjne

Znakomita większość algorytmów odkrywających zbiory częste oparta jest na założeniu akceptacji baz danych o wyłącznie binarnych wartościach. Jest to prawdopodobnie spadek z przeszłości po popularnej analizie koszyka zakupowego (ang. *Market Basket Analysis*). Kiedyś stosowane były w większości dane binarne. Z kolei obecnie częściej zbierane są informacje o ciągłych dziedzinach. Przykładem mogą być wyniki badań nad kwiatem o wdzięcznej nazwie Kosaciec, które zaowocowały legendarną już bazą danych Iris[2]. Wiedza jakie relacje ukryte są pomiędzy atrybutami numerycznymi jest niezwykle przydatna. W tym celu buduje się ilościowe reguły asocjacyjne (ang. *quantitative association rules*)[3]. Mogą one opisywać implikację danych o dowolnych wartościach (w tym numerycznych). W literaturze można spotkać następujące konwencje zapisu:

$$A = [a_1, a_2] \wedge B = [b_1, b_2] \Rightarrow C = [c_1, c_2]$$

lub

$$A : [a_1, a_2] \wedge B : [b_1, b_2] \Rightarrow C : [c_1, c_2]$$

Formalnie: $a_1 \leq A \leq a_2$ oraz $b_1 \leq B \leq b_2$ oraz $c_1 \leq C \leq c_2$

Wszystkie zapisy służą pokazaniu, iż tym razem nie tylko stwierdzone jest, że atrybuty A, B oraz C biorą udział w implikacji, ale dodatkowo podane są przedziały wartości jakie wchodzą w jej skład. To jest jedyna różnica w stosunku do reguł binarnych, poza tym wszystkie definicje oraz miary (wsparcie, ufność) pozostają dalej uznawane.

W powyższych zapisach zastosowano przedziałowy **selektor danych**. To znaczy, że wybrane wartości należały do konkretnego przedziału (np. [a,b]). Istnieją też inne rodzaje selektorów (deskryptorów) takie jak większościowy oraz równościowy. Pierwszy z nich do selekcji danych używa operatorów porównania $<$ lub $>$ albo ich słabszych odpowiedników $\leq$ lub $\geq$. Przykładem niech będzie wyrażenie: $\{\text{wiek} < 7\}$. Ostatni deskryptor, oznaczany znakiem $=$ wybiera konkretne wartości. Wszystkie selektory są stosowane identycznie jak ich matematyczne odpowiedniki. Do atrybutów numerycznych można stosować wszystkie podejścia. Mimo to w dalszej części pracy wykorzystywany jest wyłącznie zapis przedziałowy.

Warto przyjrzeć się pojedynczemu składnikowi reguły: $X = [x_1, x_2]$. Zostało założone, iż rozpatrywany przedział jest domknięty z obu stron. Zasadność takiego warunku można zauważyć na prostym przykładzie. Mając zbiór

[2]C.L. Blake and C.J. Merz. UCI repository of machine learning databases. University of California, 1998 (https://archive.ics.uci.edu/ml/datasets/Iris)

[3]Termin ten wprowadza artykuł [2]

$Z = \{1, 147, 2, 151, 148, 3\}$ można bezspornie podzielić go na dwa podzbiory: $Z_1 = \{1, 2, 3\}$ oraz $Z_2 = \{147, 148, 151\}$. Brzegowe wartości obu zbiorów (uporządkowanych) mogą utworzyć granicę przedziałów dzielących dziedzinę danego zbioru: [1, 3] i [147, 151]. Jak widać, nie jest możliwe stworzenie ciągłego podziału (tak by suma przedziałów tworzyła ciągły przedział), ponieważ zbiór Z nie daje żadnych informacji na temat luki (3,147). Jakiekolwiek założenia mogłyby być sprzeczne z rzeczywistością. Dystans ten nie może zostać zniwelowany, a oba przedziały muszą być domknięte z obu stron aby dobrze opisywać zbiory Z_1 i Z_2. Ten fakt dobrze wpisuje się w idee całej eksploracji reguł, której zadaniem jest odkrywanie asocjacji już istniejących w danych, bez dokonywania żadnych założeń na temat próbek, które mogą pojawić się w przyszłości. Istnieją oczywiście również dziedziny eksploracji danych, które mają odmienną filozofię, jak choćby klasyfikacja.

Podział większych zbiorów wartości ciągłych (rzeczywistych, całkowitych, itp.) nie jest już tak oczywisty jak na powyższym prostym przykładzie. Wtedy zmiana granic wpływa na wsparcie tworzonego przedziału, co jest kluczowe w eksploracji reguł. To znaczy, że dysponując pewną regułą ilościową można zbudować regułę silniejszą, jeśli zwiększy się przedział jednego ze składników ilościowych. Niestety, czym silniejsza jest asocjacja, tym mniej może być interesująca dla użytkownika, ponieważ może być zbyt ogólna i dobrze znana.

Tu nieśmiało zarysował się problem, któremu czoła chce stawić ta praca. Ogólnie i w przybliżeniu zadanie polega na takim doborze zakresów poszczególnych atrybutów by uzyskać możliwie dobrą regułę. Choć nie ma formalnej definicji dla „dobrej reguły" to intuicyjnie może być ona pojmowana jako ta pomiędzy regułą bardzo silną (lecz jednocześnie pospolitą i nieciekawą), a bardzo słabą (przez co być może incydentalną i osobliwą, choć zapewne intrygującą).

3.1 Przykład

Dla zaprezentowania teorii wprowadzonej do tej pory, przedstawiona zostanie prosta reguła ilościowa. W tym celu stworzony został niewielki zbiór danych. Zawiera on wiek, kolor oczu oraz wynik pewnego testu dla dziewięciu podmiotów. Pierwszy atrybut jest numeryczny, drugi nominalny, ostatni binarny. Dane zawiera tabela 1.

Obserwując uważnie, w szczególności atrybut o wartościach całkowitych, można zauważyć, iż w zbiorze kryje się kilka reguł asocjacyjnych. Dla przykładu można przyjąć, że poszukiwane asocjacje powinny mieć wsparcie większe lub równe $\frac{4}{9}$ całego zbioru (to znaczy, że zbiór częsty będzie wspierany przez co najmniej 4 transakcje).

TID	wiek	kolor oczu	wynik testu
1	9	brązowy	1
1	48	zielony	1
3	4	niebieski	0
4	51	brązowy	1
5	5	niebieski	0
6	49	zielony	0
7	7	niebieski	1
8	5	niebieski	1
9	51	brązowy	1

Tablica 1: Przykładowe dane dotyczące oczu.

Analizując atrybut *wiek* widać naturalny wręcz podział na dwie kategorie: osobników młodych $\langle 4, 9\rangle$ oraz bardziej doświadczonych $\langle 48, 51\rangle$. Taki podział idealnie wpisuje się w wymaganie minimalnego wsparcia, ponieważ oba podzbiory go spełniają. Teraz wprost manifestuje swoją obecność reguła:

$$\underbrace{wiek : [4-9]}_{wsparcie=\frac{5}{9}} \Rightarrow \underbrace{kolor\_oczu = niebieski}_{wsparcie=\frac{4}{9}}$$

Pod obiema stronami implikacji zapisane są wsparcia poszczególnych podzbiorów. Natomiast cała reguła ma wsparcie $\frac{4}{9}$ co w przybliżeniu daje 44% i ufność $\frac{4}{5}$ czyli 80%.

4 Analiza skupień

Analiza skupień (ang. *cluster analysis*) lub inaczej grupowanie (ang. *data clustering*) to rozwiązanie problemu odkrywania struktury grup w kolekcji obiektów nieoznaczonych żadnymi etykietami klas. Jest metodą klasyfikacji bez nadzoru (ang. *unsupervised learning*). To znaczy, że wyników podziału na zbiory nie można porównać z rozwiązaniem „referencyjnym", ponieważ takie nie istnieje dla metod bez nadzoru. Dodatkowo ten fakt powoduje niejednoznaczność grupowania oraz brak obiektywnej i globalnej miary jego poprawności.

Zdefiniowanie problemu:

Celem analizy skupień jest taki podział obiektów, by te znajdujące się wewnątrz grup były maksymalnie podobne do siebie, natomiast podobieństwo pomiędzy tymi, które przynależą do różnych skupisk powinno być minimalne.

Innymi słowy, optymalizując powyższe warunki, poszukiwane są spójne podobszary danych.

Wygląda na to, że idea i kierunek działania dla rozwiązania tego problemu jest znany. Niemniej jednak w praktyce zastosowanie opisanej optymalizacji jest trudne. Sam problem grupowania jest uznawany za NP-zupełny (odniesienie w [12]). W dodatku jego przynależność do metod uczenia bez nadzoru sprawia, iż problematyczne jest zdefiniowanie kompletnego celu końcowego. A co za tym idzie, niełatwo znajduje się warunki zakończenia działania dla algorytmów rozwiązujących to zadanie.

4.1 Miara podobieństwa

Pod pojęciem grupy[4] kryje się więc zbiór obiektów, które są „podobne", wobec tego mają one porównywalne właściwości. W takiej formie jest to dość mgliste i niejednoznaczne wyjaśnienie. Dlatego w każdym konkretnym przypadku stosowania analizy skupień konieczne jest ścisłe zdefiniowanie miary podobieństwa (lub adekwatnie niepodobieństwa) rozpatrywanych obiektów.

Konieczność powołania takiej definicji może budzić zaniepokojenie w momencie napotkania danych o charakterze kategorycznym (podrozdział 2.1). Są to wartości ze skończonej przestrzeni, dla których nie sposób wyznaczyć funkcję odległości. Jednakże nawet takie warunki nie budują bariery nie do pokonania, czego dowodem są wyniki prac [7] oraz [6] - rozdział *„Distance Measures"*.

Zadanie wytyczenia funkcji podobieństwa przyjmuje formę bardziej przejrzystą i niebudzącą tylu wątpliwości jeśli grupowaniu podlegają tylko obiekty opisane właściwościami o wartościach ciągłych (numerycznych). Takie założenie uprawnia do wyboru jednej z wielu funkcji odległości i zbudowania w oparciu o nią adekwatnej funkcji podobieństwa. Praca [4] przedstawia odległość Euklidesową jako najczęściej stosowaną w takim przypadku.

$$d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\| = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Jest to uzasadnione dwoma czynnikami. Pierwszy to fakt, iż miara ta jest bardzo dobrze znana i powszechnie używana. Drugi, że dzięki temu jest intuicyjna albo sprawia wrażenie właśnie takiej. Powyższa funkcja stanowi szczególny przypadek miary Minkowskiego, która z kolei opisuje odległości w przestrzeniach o większej liczbie wymiarów:

[4]Pojęcia: grupa, skupisko, zbiór są tutaj używane wymiennie.

$$L_m(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^m \right)^{1/m}$$

Nadzwyczaj uciążliwą wadą tej ostatniej miary jest bardzo szybka utrata intuicyjności wraz ze wzrostem wymiarowości przestrzeni. O ile dla czterowymiarowych danych można odważyć się na pewne wnioski płynące z analizy odległości pomiędzy obiektami, o tyle już dla podwojonej ich liczby, czyli ośmiu wymiarów jest to karkołomne wyzwanie. Należy zauważyć, że pomimo iż nikogo nie przerażają wektory o kilkunastu elementach, to jednak przestrzenie wielowymiarowe mogą być kłopotliwe w interpretacji. To naturalne, że człowiek traci orientację i intuicję „matematyczną" po wyjściu poza przestrzeń, którą zna od dziecka (trójwymiar). Dlatego w większości przypadków, funkcja podobieństwa jest po prostu bezdyskusyjnie przyjmowana jako funkcja odległości. W praktyce rozważania są ograniczane do rozwiązania następnego kłopotu.

4.2 Normalizacja

Kolejnym problemem (tym razem niezależnym od liczby wymiarów) jest dominacja składników o „szerokich" dziedzinach. Innymi słowy takich, dla których zakres wartości jest większy niż w innych składnikach przestrzeni. Na całą miarę Minkowskiego wpływ ma suma „odległości" na poszczególnych wymiarach (atrybutach). Najlepiej, żeby wszystkie składniki miały wartości zbliżone do siebie (chociażby pod względem rzędu wielkości). W przeciwnym przypadku nawet pojedynczy składnik może zdominować zachowanie całej sumy. Można zauważyć to zjawisko dysponując nawet tylko dwoma atrybutami, z których jeden przyjmuje wartości z bardzo małego zakresu, a drugi rozciąga się na całą, bardzo szeroką dziedzinę. Można spostrzec, że w ramach drugiego wymiaru różnica odległości dowolnych dwóch punktów będzie zawsze dużo większa niż dla pierwszego. Tym samym więc ten drugi ma większy wpływ na wartość całej funkcji odległości. Zatem niewielkie zmiany położenia w ramach „szerokiego" atrybutu spowodują stosunkowo duże zmiany wartości pełnej miary Minkowskiego. Takie zachowanie obniża intuicyjność oraz uniemożliwia ustalenie warunków dla których można mówić o obiektach podobnych do siebie (co często czyni się na podstawie wyznaczenia progu dla odległości). Lekiem na wyżej wymienione dolegliwości może być **normalizacja** danych oddzielnie względem poszczególnych atrybutów. Zazwyczaj używane jest skalowanie przy wykorzystaniu zakresu lub wariancji [4]. W celu rozwiązania tego problemu adoptuje się również rozwiązania ze statystyki, w tym standaryzację:

Standaryzacja (ang. *standard score*) jest sposobem normalizacji zmiennej losowej. Po tym procesie zmienna posiada zerową wartość oczekiwaną ($\mu = 0$) oraz wariancję równą jeden ($\sigma = 1$). Najpopularniejsza jest *standaryzacja Z*, którą opisuje wzór:

$$z = \frac{x - \mu}{\sigma}$$

Podsumowanie

Istnieją też inne miary odległości wykorzystywane w analizie skupień, jak chociażby nieskomplikowana odległość Manhattan czy ambitna miara Mahalanobis. Jednakże w ramach tej pracy badane będą tylko i wyłącznie dane o charakterze liczb rzeczywistych oraz najczęściej jednowymiarowej dziedzinie, stąd prosty wniosek, że wystarczająca jest miara Minkowskiego.

Podsumowując, analizie skupień mogą podlegać dowolne obiekty pod warunkiem, że istnieje dla nich miara podobieństwa. Mimo to najczęściej pod pojęciem *obiektu* kryje się uporządkowany wektor cech (atrybutów) o wymiarze d. Dzięki temu można rozpatrywać go jako *punkt* w $d-$wymiarowej przestrzeni. W takim kontekście można mówić nawet o kształcie grup: wklęsłe, wypukłe, albo bardziej drobiazgowo: koliste, prostokątne, itp. Niektóre algorytmy jako swoje właściwości mają również specyfikowany kształt zbiorów wynikowych[5].

Daleko idący i atrakcyjny przegląd dziedziny analizy skupień został przedstawiony w pracy [4].

4.3 Rodzaje analizy skupień

Współczesna informatyka dostarcza szeroki wachlarz rozwiązań problemu analizy skupień. Każde z nich działa na swój sposób i ma wyjątkowe właściwości. Ta różnorodność jest cechą pozytywną ze względu na fakt, że nie wszystkie zadania grupowania są takie same. Oczywiste jest, iż w praktyce stosuje się różne miary podobieństwa. Ale dodatkowo stawiane są różnorodne wymagania odnośnie właściwości jakie mają spełniać nowo powstałe grupy. I chociażby z tego ostatniego powodu można podzielić wszystkie metody analizy skupień na kilka kategorii, oto lista pojęć jakimi są one identyfikowane:

4.3.1 Właściwości grup wynikowych

To jakie cechy będą mieć przedziały po analizie skupień zależy od zastosowanego algorytmu. Natomiast, dostępne opcje prezentuje poniższa lista:

[5]Przykładem jest algorytm dzielenia według siatki, który produkuje wyłącznie prostokątne podobszary.

1. **równe szerokości przedziałów** (ang. *equi-width*) – każdy interwał ma w przybliżeniu równą średnicę[6].

2. **równe głębokości** (ang. *equi-depth*) – uzyskane zbiory zawierają w przybliżeniu tyle samo elementów. Nazwa angielska i jej polskie tłumaczenie najprawdopodobniej są zainspirowane grupowaniem hierarchicznym, w którym to poziom głębokości definiuje również w pewnym sensie liczność grup.

3. **jednolite przedziały** (ang. *homogeneity-based bins*) – rozmiar grupy jest tak dobierany, by rozkład jej elementów był możliwie jednolity. Taką kategorię opisuje praca [12].

4. **grupowanie rozmyte** (ang. *fuzzy clustering*) – otrzymane grupy mogą mieć parami niepuste części wspólne. W potocznym znaczeniu - częściowo nakładają się na siebie. Kluczowe znaczenie ma dobór ograniczeń na wspólny podzbiór, czyli odpowiedź na pytanie jak bardzo grupy mogą się przenikać ze sobą. Drugim pytaniem jest, czy dopuszczalne jest zawieranie zbiorów (być może przydatne w pewnych specyficznych zastosowaniach).

   Formalnie: każdy obiekt zbioru może z pewnym prawdopodobieństwem należeć do każdego z powstałych przedziałów. Dla przykładu: dysponując zbiorem grup $\{G_1, G_2, G_3\}$, element x, dzięki swoim cechom, z prawdopodobieństwem $0,78$ powinien przynależeć do zbioru G_1, natomiast z $0,12$ do G_2 oraz z $0,1$ do G_3. Stosując grupowanie rozmyte nie dokonuje się rozstrzygnięcia do której z grup należy dany obiekt. Jest on pojmowany jako członek wszystkich grup z dodatkową informacją na temat „poziomu" tego członkostwa.

5. **grupowanie „twarde"** (ang. *hard clustering*) – każdy z analizowanych obiektów może należeć tylko i wyłącznie do jednej z grup. Dlatego utworzone zbiory nie posiadają części wspólnych. Jest to tradycyjne podejście do tematyki analizy skupień, a przez to jest najczęściej stosowane. Jego przeciwieństwem są metody rozmyte.

6. **grupowanie „naturalne"** (ang. *natural clustering*) - to koncepcja wydobycia ze zbioru takich skupisk, jakie tam się naturalnie znajdują. Sensem tego podejścia jest jak najmniejszy wpływ sztucznie i manualnie wybranych parametrów na końcowy podział. Do tej kategorii można

[6]Przez średnicę rozumie się tutaj maksymalną odległość pomiędzy dwoma dowolnymi elementami zbioru.

zaliczać metody gęstościowe, jak np. algorytm DBSCAN, czy metody jakościowe, jak np. algorytm Quality Threshold.

Zapoznanie się z taką kategoryzacją analizy skupień pozwala na optymalny dobór metody, a później konkretnego algorytmu dla przedstawionego rzeczywistego problemu. Użytkownik musi wskazać dokładnie jakich cech oczekuje. Należy jednak wspomnieć o tym, że istnieje możliwość mieszania powyżej przedstawionych opcji. Jest dozwolone zbudowanie takiego algorytmu, który będzie rozpatrywać zarówno kryterium równej szerokości tworzonego zbioru, ale również będzie czuły na ilość zawartych w nim obiektów lub ich rozkład. Wszystko zależy od warunków jakie są stawiane przed procesem grupowania.

4.3.2 Metody podziału

Znając już warunki jakie musi spełniać pożądane grupowanie, należy wybrać jeszcze jedną z wielu procedur grupujących[7]. Ponownie decyzję można oprzeć o kilka ogólnych kategorii. Procedury analizy skupień dzielą się na:

1. **hierarchiczne** (ang. *hierarchical*) – grupy są łączone ze sobą w celu utworzenia hierarchii – dwie mniejsze grupy mogą tworzyć ze sobą większą (pod warunkiem, że są do siebie „podobne"). Mówiąc inaczej, cały zbiór zawiera kilka mniejszych zbiorów, z kolei te zawierają w sobie jeszcze mniejsze i tak dalej. Tutaj należy wyróżnić dwa podejścia:

   (a) **wstępujące** – każdy pojedynczy obiekt początkowo traktowany jest jako grupa, następnie iteracyjnie grupy są łączone w pary. Taki proces kończy się zazwyczaj po połączeniu dwóch ostatnich grup w cały zbiór, albo po osiągnięciu założonego wcześniej celu dotyczącego właściwości podziału. W jednym i w drugim przypadku produktem końcowym jest hierarchia grup (struktura zawierania).

   (b) **zstępujące** – metoda rozpoczyna działanie od całego zbioru i dzieli go na mniejsze fragmenty, te nowo powstałe są rozcinane w podobny sposób rekurencyjnie. W tym przypadku warunek stopu też podlega ustaleniu. Absolutny koniec to uzyskanie wszystkich jednoelementowych zbiorów. W wyniku działania otrzymywana jest hierarchia grup (struktura dzielenia). Lecz tym razem, brak kompletnej struktury, to brak pełnej wiedzy na temat podobieństwa „małych" zbiorów, łącznie z jednoelementowymi.

[7] Obecne rozważania nadal nie nawiązują do żadnych konkretnych algorytmów.

Jeśli dowolna miara odległości zostanie wybrana jako funkcja podobieństwa, to dla obu powyższych podejść należy dokonać wyboru sposobu obliczania odległości pomiędzy grupami. Innymi słowy, zbiór obiektów jako taki nie stanowi pojedynczego punktu w przestrzeni, co powoduje niejednoznaczności przy obliczaniu funkcji odległości. Może być ona ustalona pomiędzy „środkami"[8] zbiorów. Ale również możliwe jest zastosowanie odległości maksymalnej, czy minimalnej. Zagadnienie to nazywane jest *metodami wiązania*, a więcej na ten temat w [8] w rozdziale Analiza skupień.

Algorytmy hierarchiczne to jedne z niewielu, które oprócz wyboru funkcji podobieństwa nie wymuszają dodatkowych parametrów. Natomiast na samą funkcję nie ma nałożonych żadnych ograniczeń ani wymagań (oprócz wartości zwracanej, która ma być miara określającą jak bardzo jeden obiekt jest podobny do drugiego). Budowana jest wtedy cała hierarchia, którą można analizować nawet wizualnie na dendrogramach, czy diagramach venna.

2. **metody oparte na podziałach** (ang. *partitional clustering*)[9] To metody stosujące pojedynczy i rozłączny podział całego zbioru zamiast hierarchii podziałów. Biorąc pod uwagę ich cechy można wyznaczyć pewne kategorie i są to kolejno procedury:

   (a) **deterministyczne** albo **niedeterministyczne** – niektóre algorytmy grupujące uznawane są za niedeterministyczne, ponieważ ich działanie opiera się w jakimś stopniu o „losowość". Przykładem może być algorytm k-means, który losowo ustala położenie pierwotne środków grup.

   (b) **minimalizujące błąd wynikowy** (ang. *Error Minimization Algorithms* [6]) – tak właściwie jest to raczej idea stojąca za niektórymi konkretnymi algorytmami. Polega ona na minimalizacji pewnego kryterium błędu, najczęściej opisanego funkcją odległości. Najbardziej znany jest błąd kwadratowy (ang. *squared error alorithms* [4]), a konkretnie błąd średniokwadratowy (ang. MSE - *Mean Squared Error*), czy suma kwadratów błędów (ang. *Sum of Squared Error*). Należy więc ustalić jakie odległości są rozpatrywane – najczęściej badana jest odległość punktów do środków

[8] To tak zwana metoda środka ciężkości zbioru. Można go wyznaczyć poprzez średnią (zwykłą lub ważoną) wartości we wszystkich punktach zbioru.

[9] Definicja zaczerpnięta z encyklopedii uczenia maszynowego [9] - hasło *partitional clustering*

grup ich zawierających. Cały proces polega wówczas na globalnej optymalizacji funkcji błędu.

Dobrą ilustracją tej procedury jest algorytm k-means, który początkowo losuje pakiet środków przyszłych grup, np. $\{S_1, S_2, S_3\}$. Niestety to jest jego wada, że liczbę i położenie tych środków należy ustalić apriori. Celem jest takie przemieszczenie środków względem obiektów aby zminimalizować błąd (np. średniokwadratowy sumy odległości punktów od środków). Rozwinięcie zagadnienia znajduje się w pracy [6] w podrozdziale „Partitioning Methods".

(c) **oparte na gęstościach** (ang. *density-based algorithms*) wykonują swoją pracę analizując gęstość rozłożenia punktów zbioru w przestrzeni. Tym razem, każdy obiekt z zadanego zbioru musi być opisany wektorem liczb (odpowiadającym kolejnym jego cechom, atrybutom obiektu). Tak stworzony wektor jest traktowany jako punkt w przestrzeni (liczba wymiarów odpowiada rozmiarom wektora). Tworzone grupy cechują się ustaloną i jednolitą wewnętrzną koncentracją. Natomiast dzięki kontrastowi gęstości można wydzielać kolejne zbiory. Punkty należące do obszarów o niskim nasyceniu są traktowane jako szum (zakłócenia lub próbki odstające od zbioru) i nie podlegają grupowaniu. Przykładem implementacji tej metody jest algorytm DBSCAN[10].

Algorytmy zbudowane w oparciu o tą idee mają spektakularne zalety. Po pierwsze i co najważniejsze nie wymagają dogłębnej znajomości dziedziny analizowanych danych (ani liczby ani położenia środków). Pozwalają także na wykrycie grup o bardzo dowolnym, a nawet wyrafinowanym kształcie. Te dwa argumenty już sprawiają, że można podchodzić do nich bardzo entuzjastycznie. Niemniej jednak takie atuty mają też swoje skutki uboczne. Problemem bywa ustalenie poziomu koncentracji wewnętrznej grup oraz poziomu gęstości, poniżej której punkty traktowane są jako szum. Procedura jest w stanie działać bardzo dobrze dla danych, w których skupiska punktów są dobrze odizolowane, co niestety nie zawsze zachodzi w świecie rzeczywistym.

(d) **oparte na grafach** (ang. *graph-theoretic clustering* lub *graph-based clustering*) – w zarysie polegają na rozpięciu grafu o węzłach w punktach zbioru. Najbardziej znane algorytmy wykorzystują

[10]Ester M. i in.: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, Proc. of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)

minimalne drzewo rozpinające (ang. *Minimal Spanning Tree*). Informacje szczegółowe zawiera praca [6] – rozdział „Graph-theoretic clustering".

(e) Istnieje jeszcze wiele innych podejść do metody grupowania opartej na podziałach. Są to między innymi procedury wykorzystujące sieci neuronowe ([6] – „Neural networks"), czy nawet techniki ewolucyjne. W nielicznych sytuacjach można również zastosować podział według narzuconej „siatki" (ang. *grid-based methods* [6]).

Jednakże dotychczas przedstawione metody oraz procedury stanowią wystarczającą bazę dla wyboru rozwiązania najlepiej spełniającego cel niniejszej pracy.

4.4 Rozważania na temat właściwości grupowania

Dysponując całym wachlarzem algorytmów grupujących, należy wybrać właściwości jakie są oczekiwane względem zbiorów końcowych. Pewną wskazówką może być klasyfikacja tych algorytmów naszkicowana w rozdziale 4.3 Rodzaje analizy skupień.

Odniesienie do właściwości podziałów (rozdz. 4.3.1)

Analiza skupień w przypadku tej pracy nie może ograniczyć się do grupowania ani o stałej szerokościach, ani o równej liczności. To specyfika i naturalny rozkład danych w bazie powinien narzucić rozpiętość i rozmiary poszczególnym grupom. Z drugiej strony zostaną wprowadzone zdroworozsądkowe ograniczenia oparte właśnie na tych parametrach. Wiadomo, że szerokość grupy określa poniekąd jej „jakość". Natomiast liczba elementów będzie niezmiernie ważna dla późniejszego odkrywania reguł.

Tworzenie pakietu rozłącznych grup to podejście tradycyjne i dobrze zakorzenione w intuicji wielu ludzi. Niemniej jednak warto wykorzystać potencjał grupowania rozmytego. Ta kwestia rozstrzygnie się w kolejnych rozdziałach.

Odniesienie do metod podziałów (rozdz. 4.3.2)

W kontekście dostępnych ogólnych schematów postępowania, pierwsze pytanie na jakie trzeba odpowiedzieć, to czy hierarchia grupowania jest potrzebna?

Zdecydowanie wymagany jest płaski podział, który można następnie potraktować jako atrybut nominalny dla reguł asocjacyjnych. Dlatego ani hierarchia grup, ani dokładność tego procesu nie są przydatne. Po przedstawieniu reguły, np. $wiek : [5 - 10] \Rightarrow wzrost = niski$ nie jest już istotne czy przedział $[5-10]$ został zbudowany w oparciu o dwa czy trzy inne przedziały. Nie jest też ważne jak grupują się obiekty pojedyncze.

W następstwie powyższych wniosków, algorytmu należy szukać wśród metod opartych na podziałach „płaskich". A w tych ramach, należy odpowiedzieć na pytanie, czy zachowanie losowe jest w tym zastosowaniu akceptowalne?

Grupowanie jest elementem transformacji, która jak było sugerowane wcześniej powoduje przekształcenie całej bazy (nawet fizycznie, z jednego pliku na drugi). Teoretycznie nie wymusza to deterministyczności całego procesu. Raz przekształcona baza może służyć do wielokrotnego poszukiwania reguł. Ostatecznie cały proces można powtórzyć kilka razy przed właściwą eksploracją reguł, żeby uniknąć negatywnego skutku „niefortunnej" sytuacji w momencie losowania (zbudowane modele ulegają wtedy pewnemu „uśrednianiu"). Z drugiej strony brak czynnika losowego, a w konsekwencji stabilność algorytmu może być kusząca i miło widziana w zastosowaniu do eksploracji danych.

Nie istnieje niestety możliwość dobrego zastosowania metody gęstościowej dla późniejszej eksploracji reguł asocjacyjnych. Dzieje się tak z kilku powodów, po pierwsze, jak zwykle problematyczne bywa ustalenie wartości wszystkich parametrów, z poziomem gęstości na czele. Dodatkowo w oryginalnych algorytmach nie ma możliwości sterowania wielkością grupy, ani zakresem jej wartości. Ostatnim problemem jest fakt, iż ta metoda pozwala na wyodrębnienie podobszarów, które są „kontrastowe" między sobą względem gęstości. Tymczasem dla eksploracji reguł cenny jest nawet podział przestrzeni spójnej. Oczywiście podział końcowy powinien opierać się na skupiskach możliwie odrębnych, ale jeśli zajdzie potrzeba rozcięcia zwartego skupiska na kilka części, to też powinno być wykonane.

Ani algorytmy grafowe, ani bezpośrednie zastosowanie sieci neuronowych nie są możliwe w przypadku baz przeznaczonych do eksploracji. Głównym powodem jest zazwyczaj duży rozmiar używanych kolekcji. Jednocześnie złożoność pamięciowa tych rozwiązań jest delikatnie mówiąc godziwa, co przyczynia się do tego, iż ich zastosowanie staje się mało zachęcające.

To krótkie podsumowanie uświadamia możliwości w wyborze algorytmu analizy skupień z zastosowaniem dla budowana reguł ilościowych. Pomijając wykluczone podejścia, w rozwiązaniu można zastosować metody oparte na podziałach w tym szczególnie deterministyczne, ale również te z elementami niedeterministycznymi. Dopuszczalne są metody z szeroko pojętej kategorii minimalizujących błąd wynikowy.

5 Zbiór wartości binarnych

Po wstępie teoretycznym musi pojawić się uzasadnienie, dlaczego tak, zdawałoby się, odmienne dziedziny eksploracji danych są przedstawiane w tej

pracy razem. Łączy je cel – odkrycie interesujących ilościowych reguł asocjacyjnych. Podążając za pracą [2] wyróżnia się dwa podstawowe typy reguł asocjacyjnych: binarne (ang. *boolean association rules*) oraz ilościowe (ang. *quantitative association rules*). Jak już zostało wspomniane wcześniej, do odkrywania asocjacji binarnych istnieje szereg algorytmów, natomiast w drugim przypadku jest ich niewiele oraz bywają mało efektywne. Stąd już w 1996 roku powstał pomysł, żeby zbiór danych formatu numerycznego przekształcić w kolekcję pozycji $\{0, 1\}$. Tym samym problem wyszukiwania ilościowych reguł staje się zadaniem odkrycia binarnych reguł asocjacyjnych. Ten rozdział zawiera przegląd możliwości zamiany pojedynczych atrybutów na binarne.

5.1 Transformacja na wartości binarne

W odniesieniu do **atrybutów kategorycznych** sposób przekształcenia jest oczywisty. Dla przykładu niech obiekty będą charakteryzowane przez *„atrybut1"*, który przyjmuje n wartości. Każdy element tej cechy powinien stworzyć nowy atrybut, klasycznie pod tytułem *„atrybut1:wartośćK"*. Tym razem jest on już dwuwartościowy, znów klasycznie, przyjmuje się dziedzinę $\{0, 1\}$. Jedynka oznacza, że pierwotna cecha *„atrybut1"* miała w danej transakcji wartość *„wartośćK"*, zero wstawiane jest w przeciwnym przypadku.

W odniesieniu do **kolekcji typu numerycznego** (np. liczb rzeczywistych) droga do zbioru dwuwartościowego nie jest tak bezdyskusyjna. Nie istnieje jednoznaczne i najlepsze rozwiązanie. Być może istnieją problemy, które będą wymagać w tym miejscu odwzorowania każdej unikalnej wartości w zbiór $\{0, 1\}$. Ale w znakomitej większości przypadków taka procedura doprowadzi do uzyskania ogromnej i bardzo rzadkiej macierzy.

Większość pozycji literatury dotykającej problemu reguł ilościowych sugeruje, iż transformacja atrybutów numerycznych powinna odbyć się poprzez dyskretyzację ich dziedziny wartości. Następnie tak podzielony atrybut można potraktować jako kategoryczny i przekształcić w dwuwartościowy tak jak to jest opisane wyżej.

5.2 Dyskretyzacja

Jest to procedura przetwarzająca informacje ciągłe w dyskretne (źródło: [4]). Formalnie, szczególnie w matematyce, pojęcie to dotyczy procesu przekształcania modeli ciągłych w dyskretne ich odpowiedniki. W topologii dotyczy przekształcania przestrzeni spójnych w przestrzenie dyskretne. Przestrzeń spójna intuicyjnie składa się z „jednego kawałka", natomiast dyskretna opiera się o punkty, które są niejako „oddzielone" od siebie.

W kontekście eksploracji danych, do dyspozycji jest kolekcja wartości konkretnego atrybutu ciągłego. Sam atrybut można potraktować jako pewną funkcję. Jej przeciwdziedzina tworzy przestrzeń spójną. Dlatego proces transformacji na dane binarne z tego punktu widzenia wygląda na zadanie dyskretyzacji przeciwdziedziny (po prostu w celu zamiany jej na topologiczną przestrzeń dyskretną). Uzyskany podział powinien pokryć całą przestrzeń. Niemniej jednak zazwyczaj o przeciwdziedzinie wiadomo niewiele. Znany jest wyłącznie skończony zbiór próbek tej przestrzeni. Dlatego w praktyce, mniej lub bardziej słusznie, mówi się o dyskretyzacji kolekcji próbek i w tym celu wykorzystuje się metody analizy skupień.

5.3 Właściwości dyskretyzacji

Sam proces grupowania niesie za sobą pewne konsekwencje. Przede wszystkich powoduje zmianę dziedziny wartości, a tym samym częściową utratę informacji o rozkładzie konkretnych wartości. Tracona jest wiedza o składzie wewnętrznym utworzonych grup. Czym szersze są nowe przedziały tym więcej informacji utajnionej.

Na szczęście, nie zawsze jest to proces szkodliwy. Metody dyskretyzacji są bardzo prostym sposobem ustanowienia pewnego poziomu prywatności danych pierwotnych. W uproszczeniu idea prywatności opiera się tutaj na ukryciu indywidualnych próbek, jednocześnie udostępniając dane zagregowane. Ma to praktyczne zastosowania w przypadku, kiedy przetwarzanie danych indywidualnych jest zabronione prawem lub pewnymi regulacjami, a dane łączne umożliwiają dalszą analizę i wnioskowanie.

Dodatkowo utrata konkretnych wartości ma kilka zastosowań ogólnych. Są to między innymi próby uproszczenia danych, na przykład w celu ich prezentacji wizualnej. Innym przykładem jest chęć przygotowania wstępnej analizy lub próba szybkiego wyodrębnienia ogólnej wiedzy.

5.4 Przykład transformacji

W celu zaprezentowania działania transformacji atrybutów różnego rodzaju, użyty zostanie zbiór danych zawarty w tabeli 1 omawiany już w rozdziale 3.1. Zawiera on zarówno wartości całkowite jak i nominalne. Minimalne wsparcie niech tym razem wynosi 40%. Pierwszy atrybut (kolumna wiek) zostanie poddany analizie skupień, czego wynikiem będzie utworzenie dwóch przedziałów $\langle 4 - 9\rangle$ oraz $\langle 48 - 51\rangle$. Wartości w dotychczasowych rekordach zostaną zamienione na wspomniane przedziały. Ten krok prezentuje tabela 2, która jest już bazą danych wyłącznie kategorycznych. W tym momencie należy zastosować binaryzację. Dla przykładu, drugi atrybut jest przekształcony na

TID	**wiek**	**kolor oczu**	**wynik testu**
1	$\langle 4-9 \rangle$	brązowy	1
1	$\langle 48-51 \rangle$	zielony	1
3	$\langle 4-9 \rangle$	niebieski	0
4	$\langle 48-51 \rangle$	brązowy	1
5	$\langle 4-9 \rangle$	niebieski	0
6	$\langle 48-51 \rangle$	zielony	0
7	$\langle 4-9 \rangle$	niebieski	1
8	$\langle 4-9 \rangle$	niebieski	1
9	$\langle 48-51 \rangle$	brązowy	1

Tablica 2: Wyniki dyskretyzacji atrybutu wiek z tabeli 1.

TID	wiek: $\langle 4-9 \rangle$	wiek: $\langle 48-51 \rangle$	oczy: niebieski	oczy: brązowy	oczy: zielony	wynik testu
1	1	0	0	1	0	1
2	0	1	0	0	1	1
3	1	0	1	0	0	0
4	0	1	0	1	0	1
5	1	0	1	0	0	0
6	0	1	0	0	1	0
7	1	0	1	0	0	1
8	1	0	1	0	0	1
9	0	1	0	1	0	1

Tablica 3: Wyniki przekształcenia danych z tabeli 2 do postaci binarnej.

tyle kolumn ile zawiera unikalnych wartości, ostatni pozostanie bez zmian. Wynik prezentuje tabela 3 zawierająca binarną bazę danych.

6 Analiza skupień, a wyszukiwanie reguł asocjacyjnych

Jak zostało zaznaczone już nawet we wstępie, w celu odkrycia ilościowych reguł asocjacyjnych zostanie przeprowadzone grupowanie wartości atrybutów numerycznych. W tym celu należy uściślić właściwości tego procesu i wymagania odnośnie wynikowych grup. Sam proces eksploracji podzielony jest na dwa etapy. Pierwszy to znalezienie zbiorów częstych, drugi to budowa reguł

z odkrytych zbiorów.

6.1 W kontekście wyszukiwania zbiorów częstych

Rozważania warto rozpocząć od przypomnienia, że zbiór częsty to taki zbiór atrybutów bazy danych, który w całym zbiorze transakcji ma wsparcie większe niż minSup. Ogólnie znane jest też twierdzenie, które mówi:

Każdy podzbiór zbioru częstego jest zbiorem częstym. To oznacza, że żaden zbiór nieczęsty nie może stać się częstym poprzez dodanie do niego nowych atrybutów. A wniosek ostateczny brzmi:

Jeśli pojedynczy atrybut jest nieczęsty, to nigdy nie wejdzie w skład reguły asocjacyjnej.

Z ostatniego stwierdzenia można wysnuć już prosty postulat: cechę numeryczną należy tak przekształcać, by nowo powstałe atrybuty były częste. Innymi słowy utworzenie grupy która nie będzie częsta nie przyniesie korzyści z punktu widzenia wyszukiwania reguł. Z tego powodu na grupowanie należy nałożyć pierwsze kryterium – powinno znać i respektować minimalne wsparcie ustalane przed procesem eksploracji. A konkretniej:

Wynikowe grupy powinny zawierać co najmniej tyle elementów ile wynosi iloczyn minimalnego wsparcia i liczby wszystkich transakcji w bazie:

$min\_rozmiar\_grupy = minSup * liczba\_transakcji$

Pod warunkiem, że minSup jest dane jako ułamek, a nie jako konkretna liczba próbek.

Ten sam problem został zaprezentowany w pracy [2] pod nazwą problemu „MinSup" i zdefiniowany nieco inaczej. Tam obecna definicja mówi, że jeśli liczba przedziałów będzie duża to wsparcie pojedynczego przedziału będzie małe. A wtedy niektóre reguły zawierające tą cechę mogą zostać nieodkryte z powodu braku minimalnego wsparcia.

Bardzo trudno jest uzasadnić sens powoływania przedziałów tworzących atrybuty, które nie wezmą udziału w żadnej regule asocjacyjnej. Jednym z pomysłów jest pomijanie takich grup podczas procesu binaryzacji. Próbki, które są w nich zawarte otrzymają zestaw samych zer - co oznacza, że dana próbka nie wspiera żadnego atrybutu (nie należy do grup, które stworzyły nowe atrybuty). Z drugiej strony jest to jednak jawne niewykorzystanie pełnego zbioru danych, dlatego w tej pracy takie podejście nie jest stosowane.

Podsumowując, z punktu widzenia problemu „MinSup" im szersze zakresy przedziałów, tym lepiej. Innymi słowy, zwiększając liczbę obiektów w grupie, zwiększa się liczbę zbiorów częstych (w których składzie będzie dana grupa). Jest to zjawisko pożądane ale niestety może być przyczyną zmniejszenia liczby reguł asocjacyjnych, co prezentuje rozdział 6.2.

6.2 W kontekście powoływania reguł asocjacyjnych

Żeby z dowolnego zbioru częstego utworzyć regułę asocjacyjną, należy podzielić go na dwie części i jedną potraktować jako lewą stronę implikacji, a drugą jako prawą. Jeśli taka reguła spełnia warunek minimalnej ufności to może być spokojnie zaprezentowana jako jeden z wyników całego procesu eksploracyjnego. W przeciwnym przypadku zostaje odrzucona. Jak to odnosi się do procesu analizy skupień? Otóż, niski poziom wiarygodności pojawia się wtedy, kiedy wsparcie poprzednika implikacji jest duże w stosunku do wsparcia całej reguły (czyli tych transakcji, które wspierają poprzednik i następnik jednocześnie). Jeśli w poprzedniku znajduje się atrybut utworzony z pierwotnie numerycznego, to zwiększając szerokość przedziału możliwe jest pogorszenie wiarygodności całej reguły. A w konsekwencji reguła może zostać odrzucona z powodu niespełnienia kryterium minimalnej wiarygodności (minConf). Innymi słowy, im mniejsze przedziały, tym lepiej z punktu widzenia poziomu ufności.

Dla przykładu warto rozważyć regułę $A \Rightarrow B$ która ma wsparcie w. Dodatkowo niech zarówno A jak i B oddzielnie też mają wsparcia w. To oznacza, że reguła ma ufność na poziomie 100%. Jeśli jednak zwiększy się zakres A, a więc jednocześnie zacznie go wspierać więcej transakcji, to poziom ufności całej reguły spadnie. Przykładowo, jeśli wsparcie A wzrośnie dwukrotnie to poziom wiarygodności zmaleje aż do 50%.

Rozwiązywanie problemów minSup i minConf działa antagonistycznie. Poprawa jednego może pogorszyć sytuację drugiego. Dlatego właśnie tak trudno jest ustalić optymalny rozmiar grup.

7 Algorytm Quality Threshold Clustering

Quality Threshold Clustering (w skrócie QT Clustering albo QTC) to algorytm grupowania pierwotnie stworzony do analizy genów [15][11]. Jego priorytetem jest zapewnienie odpowiedniego poziomu jakości dla tworzonych grup. Nie wymaga specyfikowania potencjalnej liczby skupisk jakie wystąpią w bazie danych. Wręcz przeciwnie, pozwala na odkrycie ich naturalnej liczby. Wielkością grup wynikowych steruje parametr maksymalnej średnicy skupiska. Podstawowa idea opiera się na znalezieniu grup kandydujących na podstawie każdego punktu ze zbioru. Każda z nich musi spełniać wymaganie jakościowe i co ważniejsze każda z nich jest budowana w oparciu o pełny zbiór danych (jeszcze niepogrupowanych). Ostatecznie spośród wszystkich

[11] Informacje są zaczerpnięte również z encyklopedii uczenia maszynowego [9] (hasło: „Quality Threshold Clustering")

kandydatów wybierany jest najlepszy. Elementy w nim zawarte są usuwane z grupowanego zbioru (jako już przydzielone), a cały proces jest powtarzany. Cała procedura jest zapisana w pseudokodzie poniżej – Algorytm 1.

Algorytm 1 Procedura grupowania Quality Threshold Clustering

```
funkcja QTClustering(G, d)
    if |G| ≤ 1 then return G
    end if
    for all i ∈ G do
        zbiór A_i ← {i}                          ▷ A_i jest i-tym kandydatem
        while A_i ≠ G do
            znajdź j ∈ (G − A_i) dla którego średnica(A_i ∪ j) jest minimalna
            if średnica(A_i ∪ j) < d then
                A_i ← A_i ∪ {j}
            else
                break while
            end if
        end while
    end for
    zbiór C ← największy_zbiór_z( A_1, A_2, A_3, ..., A_|G| )
    return { C, QTClustering(G − C, d) }
end funkcja
```

7.1 Właściwości

W oryginalnym algorytmie mianowanie najlepszego kandydata opiera się na wyborze największego pod względem liczby elementów. Niemniej jednak modyfikacja tego warunku może być wskazana w zależności od zastosowania algorytmu. Idea niezależnego generowania kandydatów z całej dostępnej puli próbek sprawia, że ta metoda ma wiele unikalnych zalet. Przede wszystkim jest niezależna od kolejności budowania grup ani występowania danych. Dodatkowo nie zawiera żadnego elementu losowego. To wszystko sprawia, że jest to jeden z niewielu, w pełni deterministyczny algorytm grupujący. Uruchamiając go wielokrotnie, zawsze jest pewność tego samego wyniku. Algorytm gwarantuje też, że wszystkie grupy wynikowe będą spełniały przedstawione wymaganie jakościowe (w tym przypadku maksymalną średnicę). Na dodatek wybierane są skupiska w kolejności od najlepszego (np. największego) do najsłabszego (np. najmniejszego). Daje to gwarancję, że najbardziej istotna struktura danych zostanie odkryta poprawnie.

Najważniejszą zaletą algorytmu jest fakt, że odkrywa on naturalne skupiska w zadanym zbiorze z dokładnością do ustawionej jakości. Oznacza to, że możliwe jest poznanie faktycznego obrazu struktury wartości, niesfałszowanej i niczym nie wymuszonej. Jednocześnie parametr sterujący jest dość intuicyjny dla badaczy. Dla kontrastu wystarczy wspomnieć np. konieczność ustalenia poziomu gęstości dla algorytmu DBSCAN (rozdz. Metody podziału). Tutaj ustalenie średnicy może w pierwszym momencie nie być łatwe, ale szybkie zapoznanie z dziedziną wartości (chociażby zakresu min-max) wystarczy by zapanować nad sytuacją.

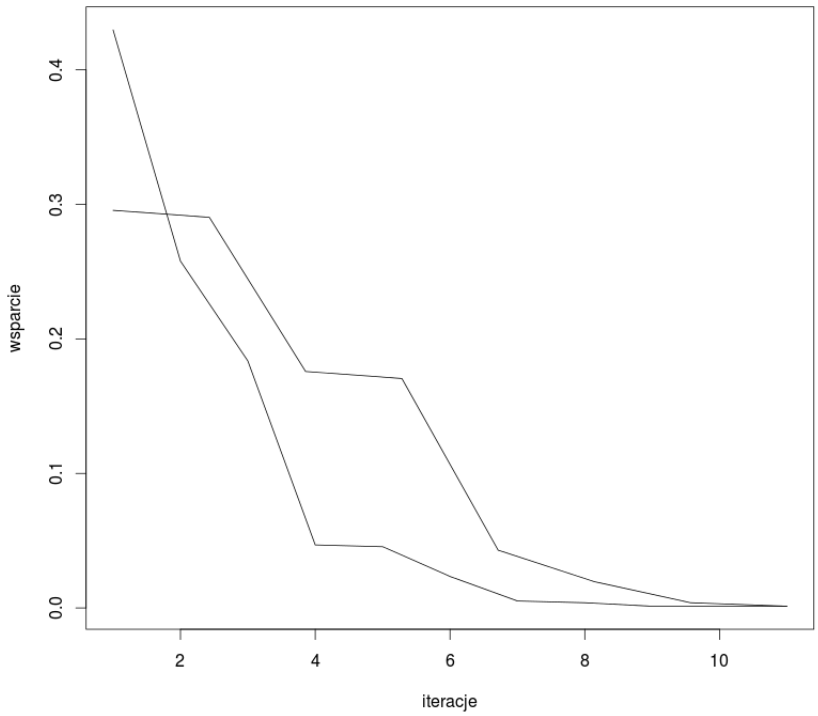
Algorytm ten wykazuje delikatne podobieństwo do grupowanie hierarchicznego z kompletnym łączeniem (ang. *complete linkage hierarchical clustering*), ale uśredniając produkuje zdecydowanie większe grupy. Jak zaznaczają jego autorzy: L. Heyer, S. Kuglyak oraz S. Yooseph [15], lokalne decyzje podczas budowy kandydatów nie mają dużego wpływu na końcowy wynik. Jedynie grupa w danym kroku najsilniejsza jest istotna dla całej analizy skupień. Autorzy przypuszczają, że metoda jest mniej wrażliwa na niewielkie zmiany w danych, niż grupowanie hierarchiczne. Co akurat było istotną zaletą w przypadku ówczesnego zastosowania z powodu konieczności filtrowania i usuwania niektórych próbek genów.

7.2 Parametr

Algorytm QT wymaga zdefiniowania jakości grup wynikowych w postaci ograniczenia na ich średnicę. Ten parametr algorytmu nazywany jest **progiem jakości**. Celem grupowania w kontekście reguł ilościowych ma być dostarczenie zbiorów, które będą miały liczność przynajmniej na granicy minimalnego wsparcia. Zakłada się więc, że przed analizą skupień znane są parametry dalszej eksploracji (w tym minSup). Według wymagań, grupowaniu będą poddawane poszczególne atrybuty osobno, czyli dane jednowymiarowe. Konieczny jest krok ich wstępnego przetworzenia, aby poznać dziedzinę wartości. W tym przypadku wystarczająca jest znajomość wartości minimalnej (niech będzie oznaczona przez atrMin) oraz maksymalnej (atrMax). Pierwszym podejściem do ustalenia wartości współczynnika jakości jest wyrażenie:

$$progJakosci = (atrMax - atrMin) \cdot minSup$$

Gdzie *minSup* to wartość minimalnego wsparcia dla reguł asocjacyjnych zdefiniowana w postaci ułamka. Celem było takie wykorzystanie dostępnego parametru algorytmu, aby zaspokoić ograniczenie minimalnego wsparcia reguł ilościowych. Powyższe wyrażenie stanowi próbę realizacji tego celu. Wybór takiego progu sprawia wrażenie dzielenia zakresu dziedziny na równomierne przedziały. Należy zauważyć, iż algorytm działający w oparciu o tak



Rysunek 1: Wsparcie grup znajdowanych w kolejnych iteracjach algorytmu Quality Threshold dla dwóch atrybutów ilościowych zbioru *Diabetes*.

ustaloną wartość będzie mieć największą skuteczność w przypadku założenia rozkładu danych zbliżonego do rozkładu jednostajnego. Jeśli zbiór danych wejściowych spełniałby taki warunek, to algorytm dostarczyłby grup o podobnej liczności i wystarczających wsparciach. Mimo to wynik byłby znacząco różny od grup uzyskanych przez algorytmy równomiernego podziału (ang. *equi-width* oraz *equi-depth*). Wynika to z iteracyjności algorytmu QTC i wyboru pierwszej grupy w dowolnym miejscu, a nie zaczynając od próbek najmniejszych (czy największych).

7.3 Algorytm QTC w kontekście reguł ilościowych

Algorytm z ustalonym powyżej progiem jakościowym ma niezaprzeczalną zaletę braku nacisku na wygląd grup wynikowych. Prezentują się one w sposób naturalny, co niestety okazuje się delikatnie kłopotliwe.

Brak gwarancji minimalnego wsparcia

Zbiory występujące w realnym świecie są nieregularne. Dlatego wewnętrz-

nie sprzeczne jest założenie, iż wynikowe grupy powinny posiadać minimalne wsparcie i jednocześnie, że podział powinien odbywać się w oparciu o parametr przestrzenny. Jakim jest próg jakości - ograniczenie średnicy przedziału. Naturalne dane wejściowe nigdy nie posiadają cech rozkładu nawet w przybliżeniu jednostajnego, zazwyczaj bliższe są rozkładowi naturalnemu. To utwierdza w przekonaniu, iż koncentracja próbek będzie znacząco różna w poszczególnych grupach wynikowych.

Problem pojawia się również na styku wymagań procesu wyszukiwania reguł i żądania naturalności. Otóż w przypadku danych dobrze separowalnych (względem wybranego parametru jakościowego) algorytm QTC pokaże naturalną strukturę. Mimo to próbki, które nie będą należeć do przedziałów o wystarczającym wsparciu, nie będą mogły brać udziału w dalszym procesie eksploracyjnym.

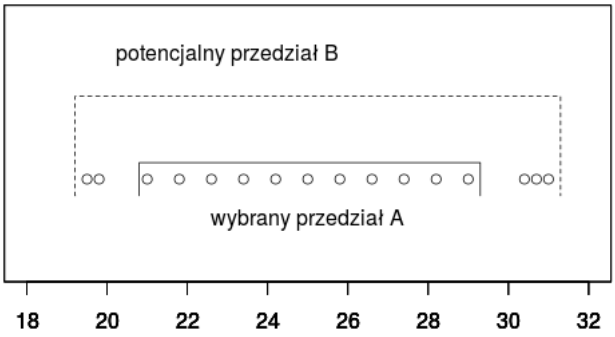
Właściwość monotoniczności

Dodatkowo z cech algorytmu wynika gwarancja, że każdy kolejny wyznaczony przedział będzie mniej liczny. Monotoniczność tego procesu sprawia, iż jeśli w pewnym momencie rozmiar grupy spadnie poniżej minimalnego wsparcia, to następne grupy z całkowitą pewnością nie spełnią wymagań. Dlatego zbiory końcowe są praktycznie nieprzydatne z punktu widzenia eksploracji reguł asocjacyjnych. Potwierdzają to doświadczenia, których skromnym przykładem może być przedstawienie zależności wsparcia zbiorów generowanych w kolejnych iteracjach procedury QTC (rysunek 1). Przykład przedstawia tylko dwa atrybuty, lecz taką tendencję można zaobserwować dla każdego zestawu danych. Jak widać w ostatnich iteracjach wsparcie jest bliskie zeru. Powodem tego jest fakt, iż ostatnie grupy zazwyczaj składają się z pojedynczych próbek.

Zwiększanie progu jakościowego

Należy uzasadnić, że prosty atak na powyższe problemy nie przynosi zwycięstwa. Pierwszym przychodzącym na myśl rozwiązaniem jest zwiększanie maksymalnej rozpiętości średnicy zbioru od razu w przypadku, kiedy uzyskany przedział przestanie zaspokajać minimalne wsparcie. Teoretycznie ponowne wyszukanie grupy z większym parametrem progu powinno zakończyć się znalezieniem grupy, której rozmiar będzie co najmniej większy od poprzednio niezadowalającego. Ten intuicyjny wniosek został zweryfikowany w praktyce i niestety nie jest prawdziwy.

Pierwszy błąd to fakt, że progu nie można zwiększyć ani trochę. Żeby to uzasadnić należy wyobrazić sobie sytuację zobrazowaną rysunkiem 2. Opisuje ona analizę skupień, której wyniki będą wykorzystane dla wyszukania



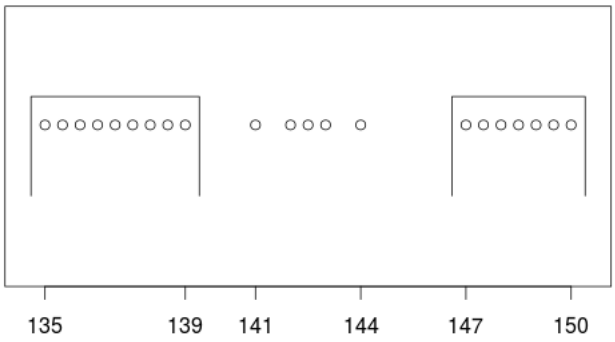
Rysunek 2: Szkic wskazujący zagrożenie zwiększania parametru algorytmu QT w trakcie działania.

reguł ilościowych, których minimalne wsparcie to 5 obiektów (tutaj wyjątkowo jednostką są elementy, a nie ułamek czy procent całości). Początkowy próg jakości to 10 (średnica przedziału nie może być większa niż 10). Dotychczas wyznaczony jest już przedział na zakresie $\langle 21; 29 \rangle$. Jego prawo do istnienia opiera się na fakcie, iż w jego granicach skupia się duża część punktów. Jednocześnie włączenie w jego szeregi punktów sąsiednich naruszyłoby warunek średnicy. Niestety w następnej iteracji algorytmu próba znalezienia kolejnego przedziału kończy się fiaskiem. Stosując w tym momencie procedurę rozluźnienia ograniczenia średnicy można wpaść w tarapaty. Konkretnym problemem jest wtedy groźba zawierania się przedziałów. Wracając do omawianego przykładu, jeśli próg zostałby zwiększony do wartości 12 (o 20%) to prawdopodobny byłby wybór zakresu $\langle 20; 31 \rangle$ (o średnicy 11). Jest on nie do przyjęcia ze względu na to, że zawiera w sobie odkryty już wcześniej zbiór.

Przedstawiony przypadek stanowi kontrprzykład dla próby uzasadnienia pomysłu zwiększania progu jakościowego. Dodatkowo uzmysławia, iż w przypadku algorytmów iteracyjnych problem niezamierzonego zawierania przedziałów jest istotny. Rozwiązanie końcowe musi unikać powyższego zagrożenia, ale istnieje jeszcze jeden problem. Zaimplementowany oryginalny algorytm QTC nie jest w stanie wytworzyć wyłącznie zadowalających rozmiarem przedziałów z powodu „problemu skrawków", który jest opisany w rozdziale 8. Ten problem dotyczy wszystkich metod, które dokonują grupowania iteracyjnie.

8 Problem skrawków

Skrawek (ang. *snippet*) to według słownika języka polskiego „resztka, pozostałość po cięciu". W grupowaniu algorytmem QTC jest to grupa wartości, która



Rysunek 3: Szkic zbioru z dwoma znalezionymi przedziałami $\langle 135; 139\rangle$ oraz $\langle 147; 150\rangle$. Natomiast punkty w $\langle 139; 147\rangle$ tworzą przykład skrawka, gdyż jest ich zbyt mało na utworzenie przedziału o minimalnym wsparciu.

nie może uzyskać żądanego minimalnego wsparcia z powodu sąsiadujących z nią utworzonych już wcześniej grup.

Przykład

Niech w wyniku działania algorytmu wyszukującego grupy w sposób iteracyjny powstaną dwa przedziały $G_1 = \langle a_1, a_2\rangle$ oraz $G_2 = \langle a_3, a_4\rangle$, przy czym $a_2 < a_3$. Jeśli dokładnie pomiędzy nimi znajduje się zbiór obiektów $S = \langle a_2, a_3\rangle$ to jest tak jakby odizolowany od punktów bazy danych przez sąsiedztwo G_1 oraz G_2. Zbiór S można nazwać skrawkiem, wtedy jeśli jego wsparcie będzie niższe od zakładanego minimum, a więc dalszy jego podział będzie daremny. Przykład zbioru tego typu można zaobserwować na szkicu 3. Przedział [139, 147] jest otoczony przez zbiory silne, ale sam nie spełnia wymogu wsparcia.

8.1 Rozciąganie przedziałów

Jedną z najprostszych metod radzenia sobie z problemem opisanym wyżej jest polityka rozciągania przedziałów. W przypadku napotkania skrawka może on być włączony do jednego z dwóch swoich sąsiadów. Istnieją dwa przypadki tego procesu:

1. Przedział rozpatrywany ma tylko jednego sąsiada - taka sytuacja zdarza się „na końcach" dziedziny. Czyli dla wartości najwyższych i najniższych. Jak wynika z przeprowadzonych eksperymentów, skrajne elementy dziedziny prawie zawsze tworzą skrawki. Jest to uzasadnione chociażby tym, że to właśnie ekstrema dziedziny wartości zawierają najczęściej próbki „fałszywe" i odstające od typowych obiektów.

Przedział znajdujący się w takiej konfiguracji musi być przyłączony do swojego sąsiada. Innymi słowy, znaleziona wcześniej grupa zostanie tak rozciągnięta, aby obejmować też dany skrawek.

2. Przedział rozpatrywany leży pomiędzy dwoma oznaczonymi już grupami. Ten przypadek jest omawiany od samego początku i przedstawiony na rysunku 3. Teraz rozwiązanie nie jest aż tak oczywiste jak w podpunkcie wyżej. Rozsądnie jest podzielić rozpatrywany przedział na dwa obszary, tak żeby rozciągnąć obie sąsiadujące grupy. Należy w tym przypadku oprzeć się o pewną heurystykę, ponieważ błędnym rozwiązaniem byłby podział na dwie równoliczne połowy. Wydaje się, że najlepiej sprawuje się rozpatrywanie bliskości. Niech każdy nieprzydzielony punkt trafi do tej grupy, do której odległość jest najmniejsza. Ze wszystkich znanych metod wiązania [8] skuteczna będzie metoda pojedynczego wiązania. W jej ramach odległość pomiędzy dwoma grupami jest równa dystansowi pomiędzy dwoma najbliższymi punktami. W tym przypadku będzie to odległość rozpatrywanego punktu do granicy przedziału. To wiązanie może być zastosowane tylko pod warunkiem, że granica grupy zostanie rozciągnięta dopiero po rozdzieleniu wszystkich elementów skrawka.

Równie dobrym wiązaniem może być metoda środków - obliczanie dystansu do środka grupy. Chociaż to rozwiązanie ma w wyjątkowych okolicznościach gorsze werdykty niż nakazywałaby intuicja, to jednak jest stabilniejsze od wiązania pojedynczego. Stabilność jest skutkiem wysokiej bezwładności środka grupy.

8.2 Nakładanie przedziałów

Całkowicie oryginalnym wyjściem z impasu wprowadzonego przez skrawki może być odpowiednie zaadoptowanie techniki grupowania z nakładaniem (ang. *overlapping clustering*). Ogólnie technika ta pozwala na nierozłączne dzielenie dziedziny. Innymi słowy, przedziały mogą się ze sobą „zazębiać". Takie podejście można wprost wykorzystać w przypadku problematycznych skrawków. Wystarczy rozszerzyć zbyt mały przedział korzystając z elementów, które już były użyte (przydzielone). Zapis przedziałów nakładających umożliwia tylko algorytm analizy skupień, który od razu dokonuje binaryzacji atrybutów (pojedyncze atrybuty transformowane od razu do zbioru kolumn binarnych). Oryginalna wartość atrybutu ciągłego będzie należeć do dwóch przedziałów jednocześnie. Rekord ją zawierający, po transformacji będzie wspierać dwa binarne atrybuty zamiast jednego.

Nakładanie przedziałów ma ważną zaletę: Nie narusza struktury odkrytych wcześniej grup, które bardzo silnie odpowiadają skupiskom naturalnie zawartych w zbiorze. Mimo to, przedziały którym pierwotnie brakowało tro-

chę wsparcia mogą też wziąć udział w procesie eksploracji i wpłynąć na budowę nieoczekiwanych reguł.

Jedynym minusem metody, jest konieczność unikania całkowitego zawierania zbiorów. Powołanie dwóch takich atrybutów mogłoby doprowadzić do powstania reguły między nimi, co byłoby sprzeczne z logiką reguł asocjacyjnych.

9 Koncepcja rozwiązania

9.1 Zmodyfikowany algorytm Quality Threshold

Do tego momentu przedstawiany był tok tworzenia rozwiązania problemu postawionego tej pracy. Oryginalny algorytm QTC nie spełnił wszystkich oczekiwań. Niemniej jednak na jego podstawie można było zbudować nowy, który idealnie sprawdza się w odkrywaniu naturalnych skupisk. Nadal grupowane są dane jednowymiarowe, lecz zmianie uległ szereg cech samego algorytmu.

Algorytm 2 Procedura grupowania MQTC (ang. Modified Quality Threshold Clustering)

```
funkcja MQTC(G, minSup)
    SORTUJ(G)
    h ← minSup * ROZMIAR(G) / 2                    ▷ h to połowa przedziału
    set wynik ← ∅
    while G ≠ ∅ do
        set best ← ZAKRES(G.minID, G.maxID)                ▷ cały zakres
        for all i ∈ G do
            cand ← ZAKRES(i - h, i + h)                ▷ h elem. na lewo i prawo
            ROZSZERZIDENTYCZNEWARTOSCINABRZEGACH(cand)
            if ZAKRESWART(G, cand) < ZAKRESWART(G, best) then
                best ← cand
            end if
        end for
        wynik ← { wynik ∪ best }
        G ← G – best
    end while
    wynik ← ROZLICZSKRAWKI(G, wynik)
    return wynik
end funkcja
```

9.2 Algorytm MQTC

Nazwa *MQTC* pochodzi od angielskiego sformułowania *Modified Quality Threshold Clustering*, co w wolnym tłumaczeniu można potraktować albo jako *Zmodyfikowany algorytm QTC* lub *Ulepszony algorytm QTC*. Pseudokod znajduje się w ramce 2 powyżej.

Pierwszą widoczną zmianą jest wprowadzenie wstępnego sortowania danych wejściowych. Ten krok jest kosztowny, ale po pierwsze ułatwi kolejne kroki, a po drugie priorytetem algorytmu jest jakość, co niestety zazwyczaj odbywa się kosztem wydajności.

Najistotniejszej modyfikacji poddane zostało kryterium jakości. Tym razem zamiast ograniczenia na szerokość zbioru, użyte zostało miękkie ograniczenie na jego liczność. Procedura zna próg minimalnego wsparcia potrzebnego dla reguł ilościowych (parametr minSup). Dla każdego punktu ze zbioru budowany jest przedział kandydujący poprzez zebranie minSup elementów „najbliższych" temu punktowi (wokół niego). Zakłada się, że połowa to najbliższe elementy z lewej, a druga połowa z prawej strony. Oczywiście tak wybrane punkty nie będą najbliższe w sensie matematycznym. To zostanie wyjaśnione w poniżej.

Zbiór punktów, o których mowa tworzy zarazem pewien zakres oryginalnych danych. Nawiązując do formalnego zapisu algorytmu 2 chodzi o wartości ze zbioru G dla indeksów $i - h$ oraz $i + h$ (czyli $G(i - h)$ i $G(h + i)$). W ramach tej funkcji taki zakres oznaczany jest poprzez wywołanie $ZakresWart(G, cand)$. Zarówno z lewej strony takiego zakresu jak i z prawej może wydarzyć się sytuacja iż graniczny element ma taką samą wartość jak najbliższy sąsiad nie objęty już w przedziale o rozmiarze minSup. Formalnie:

$$G(i - h - 1) == G(i - h) \vee G(i + h) == G(i + h + 1)$$

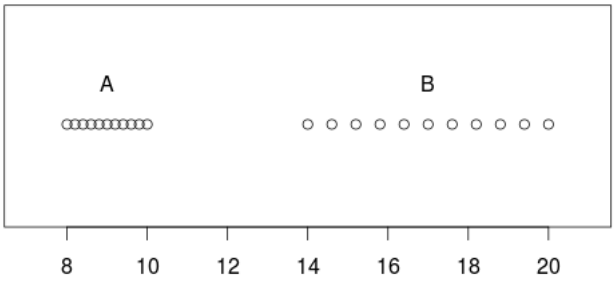
$$h = minSup * rozmiar(G)/2$$

Takie sytuacje zdarzają się dość często, bo ciężko o zbiór danych posiadający wyłącznie unikalne wartości. Naturalnym postępowaniem jest poszerzanie przedziału w obie strony tak długo, aż na brzegach osiągnie się różnicę wartości:

$$G(i - k - 1) \neq G(i - k) \wedge G(i + k + 1) \neq G(i + k), \quad k > h$$

Pomijany jest też fakt, że tak wybierane otoczenie nie musi być zbiorem najbliższych sąsiadów i-tego punktu. Nawiązując do powyższej formuły można spotkać przypadek, kiedy:

$$\exists x \in \langle 0, k \rangle, \quad G(i) - G(i - k - 1) < G(i + k - x) - G(i)$$



Rysunek 4: Szkic prezentujący dwa zbiory o tej samej liczbie elementów, ale różnym zakresie wartości.

Co oznacza, że istnieje taki element należący do prawej połowy przedziału, który jest oddalony od środkowego dalej niż pierwszy element poza lewą granicą. Taka asymetria jest akceptowalna z dwóch powodów. Po pierwsze analizowani są wszyscy kandydaci, po drugie jest nowe kryterium wyboru:

Drugą zmodyfikowaną i niezwykle istotną cechą z oryginalnego QTC jest warunek **wyboru najlepszego z kandydatów**. Pierwotnie był to zbiór najliczniejszy, teraz najcenniejszy jest zbiór najgęstszy. Można uzasadnić taką decyzję w sposób intuicyjny przyglądając się dwóm przedziałom o takiej samej liczbie elementów – rysunek 4. Z dwóch kandydatów o podobnym wsparciu lepszy jest ten, który zajmuje mniejszą przestrzeń dziedziny atrybutu (ma mniejszą średnicę). Powodem takiego wniosku jest fakt, iż skupisko z natury jest grupą gęstszą od reszty otoczenia. Zbiór A z rysunku 4 wypada lepiej w porównaniu do B.

9.3 Rozwiązanie problemów

Zmodyfikowany algorytm radzi sobie dobrze z odkrywaniem skupisk do czasu, kiedy w całej bazie pozostaną tylko skrawki niespełniające warunku minimalnego wsparcia. Wtedy osobna procedura o zabawnej nazwie *Rozlicz-Skrawki(G, wynik)* jeden raz przegląda cały zbiór i dla każdego skrawka dokonuje jednaj z dwóch decyzji:

1. Nałożenie nowego przedziału w oparciu o nieprzydzielony kawałek - w przypadku, gdy skrawek zawiera co najmniej minSup/2 elementów. Oraz pod warunkiem, że nałożony przedział nie zakryje całkowicie już istniejącego (unikanie zawierających się przedziałów). Opis w rozdziale 8.2.

2. Rozciągnięcie istniejących przedziałów - w pozostałych przypadkach. Przede wszystkim wtedy, kiedy liczność skrawka jest niewielka. Próba

powołania nowego przedziału, którego znakomita większość punktów będzie współdzielona z innymi przedziałami jest ruchem ryzykownym. Powodem jest niebezpieczeństwo powstania reguł wewnątrz pojedynczego atrybutu oraz nadmierna ilość reguł redundantnych. Dlatego w takiej sytuacji dokonywana jest procedura przesunięcia granic sąsiednich przedziałów tak aby obejmowały próbki zawarte w analizowanym skrawku. Granica pomiędzy wykorzystaniem strategii nakładania, a przesuwaniem granic przedziałów jest pewną cechą algorytmu i jednocześnie podlega kontroli. Wprowadzenie do tego sposobu było w rozdziale 8.1.

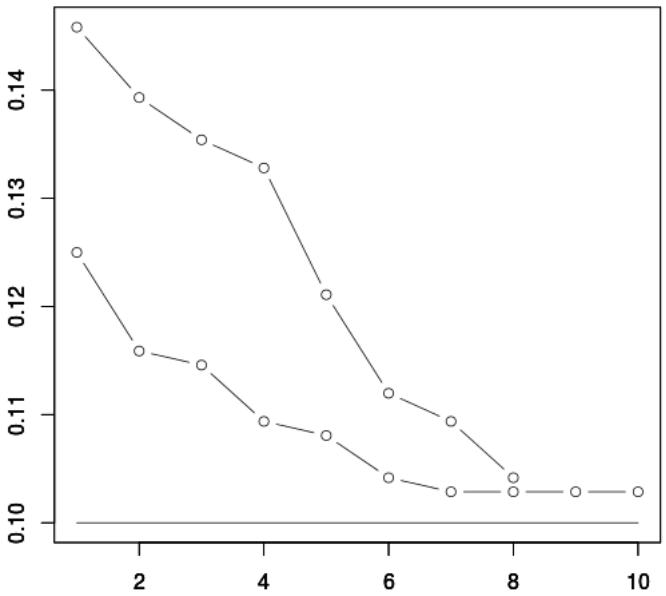
9.4 Właściwości

Modyfikacja algorytmu miała na celu rozwiązanie problemów oryginału z jednoczesnym zachowaniem jego indywidualnych właściwości. Ten cel został osiągnięty.

Zalety

Oprócz skutecznego rozwiązania zagadnienia skrawków, którego szeroki opis znajduje się w powyższych rozdziałach, rozstrzygnięty został też problem malejącego wsparcia zbiorów wynikowych. Prezentuje to rysunek 5, który jest nawiązaniem do wykresu 1 z rozdziału *7.3 Algorytm QTC w kontekście reguł ilościowych*. Wykres 5 również pokazuje tendencję spadkową wraz z kolejnymi iteracjami, lecz tym razem jest ona kontrolowana przez parametr algorytmu i nie przyjmuje wartości poniżej zadeklarowanego progu (w przykładzie 0.1).

Zmodyfikowany algorytm nadal posiada zalety oryginalnego. W początkowych etapach działania algorytmu grupowanie można ocenić jako zgodne z naturalną strukturą danych. Ta zgodność jest rozumiana tym razem w kontekście czynnika częstotliwościowego, jakim jest parametr algorytmu. Pierwotnie było to ograniczenie przestrzenne. Należy pokreślić, że taka restrykcja nie umniejsza naturalności wyników. Można powiedzieć, że wpływa na ich „rozdzielczość", czyli na rozmiary zbiorów względem całości danych. Niemniej jednak, dalsze kroki związane z analizą skrawków wprowadzają już realną sztuczność. Niestety zaspokajanie celów postawionych na wstępie to działanie antagonistyczne - chcąc uzyskać maksymalną naturalność wyniku, zmniejszana jest ilość próbek biorących udział w eksploracji reguł i na odwrót. Algorytm MQTC stanowi dobry kompromis dla tego sporu. Maksymalizuje ilość próbek, które mogą być przetwarzane w procesie eksploracji (generacja kandydatów) oraz kieruje się strukturą danych przy konstruowaniu podziału (kryterium jakości oraz sposób analizy niewielkich zbiorów).



Rysunek 5: Wsparcie grup znajdowanych w kolejnych iteracjach algorytmu MQTC dla atrybutów ilościowych zbioru *Diabetes* (nawiązanie do rysunku 1). Założone *minSup = 0.1*. Oś pozioma - iteracje, pionowa - wsparcie.

Nadal nie ma potrzeby samodzielnego przewidywania ilości grup z góry przed analizą skupień. Wręcz przeciwnie - można skorzystać z intuicyjnego parametru, który jest jednocześnie spójny z parametrem minimalnego wsparcia przy wyszukiwaniu reguł asocjacyjnych.

Wady

Szczera zasada głosi, że w informatyce nie ma rozwiązań bez wad. Analogicznie nie istnieje idealny sposób grupowania - stąd tak wielka różnorodność algorytmów analizy skupień. Podstawową wadą procedury MQTC jest jej kwadratowa złożoność obliczeniowa. Jednak jest to cecha akceptowalna ponieważ algorytm był planowany dla niewielkiej ilości danych. Kolejną wadą jest fakt, iż algorytm nie wyróżnia się na tle innych w przypadku danych o słabo zarysowanych skupieniach. Jeśli wszystkie próbki mają unikalne wartości, albo tworzą rozkład bardzo zbliżony do znanego rozkładu sztucznego (np. jednostajnego) to warto użyć innego algorytmu dyskretyzacji.

10 Opis systemu

Po stworzeniu koncepcji rozwiązania przyszedł moment na część praktyczną. W ramach niniejszej pracy został stworzony system eksploracji reguł ilościowych. Oprócz wytwarzania pożądanych asocjacji, pozwala on również na prześledzenie poszczególnych etapów budowy takich reguł. Warto podkreślić, że nie był on projektowany jako produkt dla potencjalnego użytkownika, lecz jako system prezentujący skuteczność przyjętej koncepcji.

10.1 Architektura systemu

Eksploracja reguł ilościowych, przedstawiona w tej pracy, opiera się na schemacie wprowadzonym w rozdziale 5. Zakłada się, że dane dostarczone są w postaci pewnej bazy danych, w której mają być znalezione interesujące asocjacje. Pełny proces analizy danych wygląda następująco:

1. Normalizacja i przygotowywanie danych.
2. Analiza skupień i dyskretyzacja.
3. Binaryzacja bazy w oparciu o grupy znalezione w poprzednim kroku.
4. Wyszukiwanie binarnych reguł asocjacyjnych.

Pierwszy krok wykonywany jest manualnie w zależności od potrzeb. Natomiast kolejne kroki odpowiadają konkretnym modułom stworzonego systemu.

Dane wejściowe trafiają do modułu analizy skupień, który jest zrealizowany przez implementację algorytmu MQTC. W ramach tego kroku budowany jest podział dla każdego atrybutu numerycznego, a następnie jest on zastępowany przez zestaw atrybutów nominalnych[12]. W wyniku powstaje nowa baza o atrybutach kategorycznych.

Kolejnym etapem jest binaryzacja – proces zamiany wszystkich atrybutów w bazie na binarne. Dysponując danymi z poprzedniego kroku, to zadanie można wykonać w prosty sposób opisany w rozdziale 5.1.

Należy zauważyć, że dyskretyzacja i binaryzacja są procesami zależnymi od siebie. Pierwszy z nich produkuje atrybuty kategoryczne (nominalne), natomiast drugi zamienia je na binarne. Taka zależność sprawia, że traktowanie

[12]W zasadzie nic nie stoi na przeszkodzie, żeby tworzyć atrybuty porządkowe (patrz definicja: rozdział 2.1.1). Ale w przypadku niniejszej pracy takie rozróżnienie nie daje wymiernych korzyści.

ich oddzielnie nie jest konieczne – stworzony system umożliwia transformację na wartości binarne również w jednym kroku.

Ostatnim etapem jest eksploracja reguł asocjacyjnych za pomocą dowolnego algorytmu. Istnieje wiele znanych rozwiązań tego zadania, najsłynniejsze to algorytmy: Apriori, Eclat oraz FP-Growth. W opisywanym systemie użyty został algorytm Apriori.

Podsumowując, szczegółowy schemat „przepływu" danych wygląda następująco:

dane wejściowe
⇓
MQTC
⇓
dyskretyzacja i binaryzacja
⇓
Apriori
⇓
ilościowe reguły asocjacyjne

10.2 Implementacja

Poszczególne elementy systemu są stworzone za pomocą różnych narzędzi. Zanim zostaną omówione, należy wspomnieć, że dane wejściowe są przyjmowane w postaci pliku formatu ARFF [11].

Format danych ARFF

Jest to standardowy sposób przechowywania danych stosowany głównie w słynnym pakiecie Weka [11]. Mimo to, korzysta z niego również wiele innych projektów. Jest formatem znakowym, to znaczy, że dane są przechowywane w postaci znaków – poszczególne wartości rozdzielone przecinkami. Z tego powodu jest to podejście podobne do równie popularnego standardu CSV (ang. *Comma Separated Values* – wartości rozdzielone przecinkiem). Jednakże w praktyce właśnie ARFF okazuje się bardziej pożyteczny. A to z powodu posiadania części nagłówkowej, w której można uściślić właściwości poszczególnych atrybutów.

Podsumowując, łatwość podglądu danych (postać znakowa) oraz możliwość specyfikacji nazw atrybutów w części nagłówkowej były głównymi powodami, dla których to ARFF został wybrany jako standard przechowywania danych w opisywanym tutaj systemie.

Implementacja algorytmu MQTC

Algorytm grupowania został zaimplementowany przy użyciu języka C++, z wykorzystaniem standardu C++11. Implementacja używa również funkcji oferowanych przez biblioteki STL oraz Boost. Kod podlega bezproblemowej kompilacji za pomocą kompilatorów GNU g++ 4.8 oraz Microsoft Visual Studio. Natomiast działanie programu jest potwierdzone na systemach Windows 7 oraz Ubuntu 14.04 LTS. Wybór takich narzędzi nie był przypadkowy, język C++ jest polecanym rozwiązaniem do przetwarzania danych z powodu wysokiej wydajności jaką osiągają napisane w nim aplikacje. Natomiast wspomniane kompilatory są obecnie powszechnie uważane za najlepsze i najbardziej zgodne ze standardem C++.

Implementacja MQTC tworzy program bez graficznego interfejsu użytkownika. Wszystkie argumenty przekazywane są z linii poleceń konsoli systemowej. Obsługę tych argumentów wewnątrz aplikacji zapewnia biblioteka TCLAP[13] (ang. *Templatized C++ Command Line Parser Library*). Takie podejście umożliwia przetwarzanie danych przez skrypty.

Jednym z parametrów uruchomienia aplikacji MQTC jest plik z danymi w formacie ARFF. Analizator syntaktyczny tego formatu został zaimplementowany od podstaw w celu precyzyjnego sterowania wydajnością analizy. Na przykład, nie ma konieczności dokładnie przetwarzać atrybutów, które nie będą grupowane. Dodatkowo przyjęto schemat polegający na przepisywaniu (ang. *rewriting*) zmodyfikowanych danych. Oryginalny plik jest przepisywany z jednoczesnym zastępowaniem atrybutów numerycznych.

Dyskretyzacja i binaryzacja

Proces przekształcenia danych wejściowych w zbiór binarny jest domyślnie wykonywany przez aplikację MQTC. Można temu zapobiec wybierając opcję *-c* (wszystkie opcje opisane są w rozdziale 10.4). Wtedy samą binaryzację można wykonać wykorzystując funkcję *binarize* napisaną w ramach tej pracy w języku R.

Wyszukiwanie reguł

Do zbudowania części eksploracyjnej wykorzystany został język R. Jest to bardzo dobre środowisko do obliczeń statystycznych oraz posiada szeroką gamę użytecznych pakietów. Najpoważniejszym argumentem przemawiającym za użyciem R jest istnienie pakietu Arules, którego autorem jest Michael Hahsler [14]. Z tego właśnie pakietu pochodzi wykorzystywana implementacja algorytmu Apriori.

[13] Źródło: http://tclap.sourceforge.net [dostęp 10.01.2015]

Należy jednak wspomnieć, że funkcja *apriori* z pakietu Arules jest tylko adapterem prawdziwej implementacji napisanej w języku C, którą stworzył Christian Borgelt[14]. Dzięki takiemu połączeniu proces odkrywania reguł zyskuje elastyczność języka R oraz wysoką wydajność języka C.

Idąc za przykładem funkcji *apriori* z Arules, stworzona została funkcja realizująca grupowanie, która wywołuje program MQTC napisany w języku C++. Jej zadaniem jest odczyt danych, przeprowadzenie analizy skupień i zapis nowego binarnego zbioru danych.

10.3 Proces wytwórczy

Proces wytwórczy oprogramowania dla niniejszej pracy powinien spełniać specyficzne wymagania, które są zdecydowanie inne niż w większości projektów informatycznych. Największą różnicą jest jednoosobowe prowadzenie projektu. Prowadzenie oznacza tutaj zarówno planowanie i projektowanie jak i samą implementację – aż po testy akceptacyjne. Większość technik wytwarzania oprogramowania, czy to tradycyjnych czy zwinnych, zalecana jest dla zespołów kilku osobowych, w których musi następować podział obowiązków.

W niniejszym przypadku całość pracy jest wykonana przez jedną osobę, stąd najlepsza byłaby metoda wytwarzania oprogramowania, która jednocześnie wspiera kilka etapów, na przykład projektowanie i testowanie. Od każdej aplikacji wymagana jest skuteczność (w postaci niskiej awaryjności). Można to osiągnąć, poprzez wysoką jakość kodu i użytych rozwiązań. Dla osiągnięcia takich cech najczęściej wykorzystuje się technikę inspekcji kodu (ang. *code review*). Metoda ta polega na przekazywaniu implementacji pomiędzy członkami zespołu w celu sprawdzenia poprawności przyjętego rozwiązania. Takie przeglądy powinny odbywać się możliwie jak najczęściej. Mimo wysokiej sprawności inspekcji można z powodzeniem zastąpić ją wyczerpującymi testami.

Nawiązując do powyższych rozważań, dla niniejszej pracy została wybrana metoda TDD (ang. *Test-driven development*), którą można określić metodą rozwijania oprogramowania w oparciu o testy. Jest to znana od wielu lat technika oparta o trzy podstawowe kroki:

1. Stworzenie zestawu testów dla nowej funkcjonalności, który powinien zwracać komunikaty błędów (z racji braku tej funkcjonalności).

2. Dostarczenie minimalnej ilości kodu potrzebnej na zaspokojenie poprawnej kompilacji testów.

[14] Christian Borgelt, *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 437-456, J. Wiley & Sons, Chichester, United Kingdom 2012.

3. Właściwa implementacja pełnej funkcjonalności.

Jakość oprogramowania jest oparta o wyczerpujące testy. Ale tutaj przejawia się ważniejsza zaleta: testy mogą być budowane jeszcze przed etapem właściwej implementacji. Oznacza to, że już tworząc architekturę aplikacji można dostarczyć pewien zestaw weryfikujący zgodność z koncepcją. Można to potraktować jako zwinne łączenie kilku etapów budowy, co powoduje oszczędność czasu oraz zwiększenie czytelności projektu. Ponadto takie podejście zachęca do porządkowania nawet działającego kodu (refaktoryzacja, ang. *refactoring*). W ramach niniejszej pracy proces porządkowania był kilkakrotnie wykonywany.

Problem jakości kodu i całego oprogramowania został niejako zaspokojony dzięki przyjętemu procesowi wytwórczemu. Jednakże takie podejście wiąże się z wysokim kosztem. W tym przypadku mowa o koszcie w postaci czasu poświęconego projektowi, który był niestety ograniczony. Mimo to, nie był to kłopot uciążliwy, z powodu przyjęcia zasady intensywnego wykorzystywania dostępnych narzędzi. W ramach projektów jednoosobowych warto przyjmować założenie maksymalnego wykorzystywania bibliotek i dostępnych pakietów narzędziowych, nawet jeśli własnoręczne stworzenie rozwiązania kusi wyższą wydajnością lub innymi zaletami. Tak też powstawał system w ramach tej pracy. Stąd właśnie wykorzystanie biblioteki Boost oraz pakietów języka R, które nie tylko przyspieszyły implementację, ale jednocześnie zwiększyły jej niezawodność.

Pomysłodawcą metody TDD jest Kent Beck. Więcej informacji można znaleźć w jego książce [16]. Celem tego rozdziału było wyłącznie uzasadnienie wyboru metody oraz ukazanie na jakich podstawach można mówić o jakości stworzonego systemu.

Narzędzia wykorzystane w ramach procesu wytwórczego

Język C++ jest użyty do stworzenia implementacji aplikacji MQTC. Ten wybór jest uzasadniony w rozdziale 10.2. Dlatego dla testów jednostkowych wykorzystano bibliotekę testującą z Boost (ang. Boost Test Library) Zaletą jej użycia jest przenośność pomiędzy większością platform systemowych (szczególnie Unix oraz Windows), a także wysoka wydajność podczas uruchomienia (kosztem wydłużenia czasu kompilacji).

Od strony samego wytwarzania implementacji wykorzystany został szereg narzędzi wspomagających. Od środowiska programistycznego przez system kontroli wersji (wybrano Git). Warto również dodać fakt użycia automatycznego generatora dokumentacji - Doxygen[15]. Jego zadaniem jest tworzenie

[15]Witryna domowa projektu Doxygen to: http://www.doxygen.org [dostęp 09.01.2015].

dokumentacji technicznej na podstawie specjalnie napisanych komentarzy bezpośrednio w kodzie aplikacji. Dzięki temu proces tworzenia dokumentacji technicznej można połączyć z etapem tworzenia kodu. To zapewnia lepsze utrzymanie spójności dokumentacji z realną implementacją.

Ostatnim narzędziem wartym dłuższego opisu jest system automatycznej kompilacji. Chodzi o narzędzie porządkujące sam proces kompilacji kodu źródłowego i budowania plików wynikowych. Trudność tych zadań polega na tym, iż różnią się w zależności od użytego kompilatora i systemu operacyjnego. Narzędzie CMake[16] pokonuje wszystkie te utrudnienia. Dodatkowo zapewnia automatyczne wykrywanie zależności jednostek translacji kodu źródłowego (plików źródłowych). Dzięki temu kompilacji podlegają tylko te jednostki, w których nastąpiła zmiana od czasu ostatnio przeprowadzonej kompilacji.

Jak widać, proces wytworzenia systemu, dostarczonego razem z tą pracą, nie był chaotyczny, lecz przemyślany i oparty na solidnych argumentach. Ten rozdział miał na celu uzasadnienie użycia przyjętych metod oraz narzędzi.

Testowanie

Tworzenie aplikacji MQTC było mocno związane z procesem tworzenia testów. Generalnie, stworzone zostały dwie grupy zestawów testujących. Pierwsza zawiera testy jednostkowe, a druga testy akceptacyjne (akceptujące wyniki analizy skupień).

Testy jednostkowe (ang. *unit tests*) to sposób weryfikacji tworzonego oprogramowania poprzez wykonywanie testów dla pojedynczych elementów programu. Sprawdzeniu zostały poddane metody stworzonych klas. Dla aplikacji MQTC są to w większości testy wartości brzegowych. Co oznacza, że weryfikacja skupiona jest na sytuacjach poprawnych (z wartościami skrajnymi, ale jeszcze dopuszczalnymi) oraz niepoprawnych (które niewiele odbiegają od tych dopuszczalnych). Przykładem może być wartość minimalnego wsparcia przyjmowana w postaci ułamka z przedziału $(0; 1)$. W takim przypadku jako wartości brzegowe może posłużyć zbiór: $\{-0.1, 0, 0.1, 0.9, 1, 1.1\}$. Tu postawione jest niejawne założenie, że jeśli badana jednostka poprawnie zachowuje się dla 0.1 oraz 0.9 to, że poprawne zachowanie będzie również dla całego przedziału pomiędzy tymi wartościami.

Celem **testów akceptacyjnych** nie było odkrywanie błędów, lecz potwierdzenie jakości rozwiązania. Testy zbudowane w ramach implementacji MQTC pokazują poprawne działanie analizy skupień i binaryzacji. Dzięki czemu można mieć przekonanie, iż aplikacja będzie funkcjonować poprawnie

[16] Więcej informacji o narzędziu CMake można znaleźć na: http://www.cmake.org [dostęp 09.01.2015].

również dla innych zestawów danych.

Między innymi zbadane zostały takie przypadki:

- odpowiednia liczność grup w zależności od minimalnego wsparcia
- brak transakcji, które nie należą do żadnej z grup
- sprawdzenie części wspólnej grup (zawierania)
- reakcja na błędne parametry uruchomienia

Dane dla testów akceptacyjnych pochodzą z generatora plików ARFF o losowych wartościach (stworzonego razem z platformą testową) oraz z repozytorium UCI [10]. Wszystkie przypadki testowe zostały zaimplementowane w języku R.

10.4 Sposób użycia

Działaniem aplikacji MQTC (opis: rozdział 10.2) sterują argumenty podawane z linii poleceń konsoli systemowej. Każdy z nich posiada wersję pełną (jedno słowo) oraz skróconą (pojedyncza litera). Niektóre posiadają wartość, a inne nie – są przełącznikami. Wspomniana wcześniej biblioteka TCLAP wymusza styl wpisywania parametrów zbliżony do stylu standardu POSIX. Z tego powodu konieczne jest poprzedzanie każdego argumentu myślnikiem. Kolejność wpisywania jest bez znaczenia. Popełnienie błędu lub wywołanie program bez argumentów, powoduje wyświetlenie komunikatu informującego o poprawnym sposobie użycia. Oto pełna lista opcji:

- `input` (`i`) – plik ARFF z danymi wejściowymi
- `output` (`o`) – plik ARFF dla danych wyjściowych
- `algorithm` (`a`): `qtc / mqtc` – wybór algorytmu, który zostanie użyty
- `qtcMinSup` (`s`) – minimalne wsparcie (wykorzystywane tylko w przypadku wyboru MQTC)
- `categoricalCluster` (`c`) – (bez wartości) użycie tej opcji blokuje binaryzację – dane zostaną zapisane w formie atrybutów kategorycznych (nominalnych)
- `version` (*bez skrótu*) – wyświetla wersję aplikacji
- `help` (`h`) – pełny wykaz dokładnie opisanych opcji oraz komentarz na temat użycia programu

11 Eksperymenty

Użytkownik stający przed problemem dyskretyzacji, może pokusić się o wybór metody łatwo dostępnej i jednocześnie skutecznej dla eksploracji reguł ilościowych. Takie wymagania spełniają między innymi algorytmy k-means oraz podziału na „równe głębokości" (ang. *equi-depth*). Są to metody efektywne i szeroko wykorzystywane, choć obie mają pewne wady. Podstawowa to oczywiście niepewność przy ustalaniu parametrów – liczby przedziałów. Metoda MQTC miała walczyć z tym problemem, stąd wniosek, że warto dokonać porównania tych trzech algorytmów. Jednak aby MQTC mógł stanąć w szranki z przeciwnikami należy ustanowić zasady wyboru zwycięzcy.

11.1 Kryterium porównawcze

Jako kryterium porównania algorytmów grupujących należy przyjąć dowolny sposób porównania jakości reguł asocjacyjnych budowanych na podstawie binarnego zbioru danych. Powodem takiej decyzji jest założenie, iż ich działanie ma na celu uzyskanie dobrych reguł - co nie musi być równoznaczne z uzyskaniem dobrego podziału.

W rzeczywistości liczba generowanych reguł zazwyczaj jest bardzo duża. Problemem jest jednak to, że większość z nich to reguły redundantne. Z pracy [13] zostało zaczerpnięte pojęcie reguły redundantnej, które jest przedstawione poniżej:

> Reguła $A \rightarrow B$ jest opisana na silnym zbiorze AB, natomiast reguła $C \rightarrow D$ na silnym zbiorze CD. Jeśli $AB \subset CD$ oraz $lift(A \rightarrow B) \geq lift(C \rightarrow D)$ to regułę $C \rightarrow D$ można uznać za **redundantną**.
>
> Gdzie *lift* jest miarą korelacji pomiędzy poprzednikiem i następnikiem reguły asocjacyjnej wyrażoną wzorem:
>
> $$lift(A \rightarrow B) = \frac{conf(A \rightarrow B)}{sup(B)}$$

Innymi słowy, reguła jest redundantna, jeśli dowolny podzbiór jej elementów tworzy regułę o nie mniejszym współczynniku *lift*. Jest to jeden z wielu sposobów na wykrycie niechcianych asocjacji. Skuteczność takiego podejścia do wykrywania reguł redundantnych nie będzie w tej pracy oceniana, dlatego należy wspomnieć o tym, że istnieją bardziej wyrafinowane metody, których efekty mogą być inne.

Jako współczynnik służący do porównania jakości dwóch różnych zbiorów reguł asocjacyjnych w niniejszej pracy zaproponowano i użyto następującą proporcję:

$$nredRel = \frac{allRules - redundantRules}{allRules} \tag{1}$$

Gdzie *redundantRules* to liczba reguł redundantnych, *allRules* to liczba wszystkich znalezionych reguł. Natomiast *nredRel* to przyjęta nazwa współczynnika, nawiązuje do angielskiego zwrotu *relative nonredundant rules*, co oznacza relatywną liczbę reguł nieredundantnych, które będą traktowane w tej pracy jako reguły interesujące użytkownika. Współczynnik *nredRel* został powołany wyłącznie dla tej pracy w celu przeprowadzenia eksperymentów.

Z dwóch grup reguł asocjacyjnych, lepsza jest ta, która zawiera więcej interesujących asocjacji. Dlatego porównując różne podejścia do dyskretyzacji można sprawdzić wartości współczynników *nredRel* dla wyznaczonych zbiorów reguł. Większa wartość tego parametru wskazuje na lepszy zbiór.

11.2 MQTC w porównaniu z Equi-depth

Algorytm *Quality Treshold*, a szczególnie jego modyfikacja mogą być błędnie utożsamiane z procedurą *equi-depth* – podziału o „równych głębokościach" (inaczej: równej częstotliwości). Jednak sposób działania obu metod jest zupełnie inny, a wyniki choć mają wspólne cechy to jednak prezentują się również inaczej. W tym rozdziale zostanie przedstawione porównanie wpływu stosowania obu podejść przy eksploracji reguł ilościowych.

Rysunki 6 i 7 przedstawiają wyniki eksperymentów na czterech zbiorach danych z repozytorium UCI [10]. Te zbiory to w kolejności:

0. Zbiór	Liczba próbek	Liczba atrybutów
1. Abalone	1000	9
2. Diabetes	768	9
3. Ecoli	336	8
4. Glass	214	10

Wszystkie zbiory zawierają atrybuty ilościowe i co najmniej jeden nominalny. Każdy ze zbiorów najpierw został poddany dyskretyzacji, a jej wynik posłużył algorytmowi *Apriori* do znalezienia zbiorów częstych i reguł ilościowych. Każdy diagram zawiera dwa wykresy. Linią ciągłą z zaznaczonymi punktami oznaczone są wyniki algorytmu *Equi-depth*, natomiast linia bez

wsparcie	algorytm	l. reguł	l. niered.	nredRel
0.2	MQTC	122	53	0.4344262
	Equi-depth	118	40	0.3389831
0.15	MQTC	142	65	0.4577465
	Equi-depth	158	53	0.3354430
0.1	MQTC	246	105	0.4268293
	Equi-depth	245	69	0.3070866
0.05	MQTC	494	180	0.3643725
	Equi-depth	470	126	0.2680851
0.01	MQTC	2423	572	0.2360710
	Equi-depth	2097	370	0.1764425

Tablica 4: Dane dotyczące reguł asocjacyjnych uzyskanych ze zbioru Ecoli. Oznaczenia: *l. niered* oznacza liczbę reguł nieredundantnych, *nredRel* to kryterium porównawcze opisane równaniem (1). (uwagi w rozdziale 11.2).

punktów reprezentuje *MQTC*. W obu przypadkach prezentowany jest ułamek reguł nieredundantnych zgodnie z równaniem (1).

Parametry uruchomienia

Algorytm MQTC jest uruchamiany z parametrem równym wartości minimalnego wsparcia dla reguł. Natomiast problem wskazania ilości grup na jakie ma być podzielony cały zbiór algorytmem Equi-depth został rozwiązany na jego korzyść. Parametr ten jest równy średniej liczbie przedziałów znalezionych przez algorytm MQTC. Dzięki temu oba algorytmy mają równe szanse. Gdyby nie takie postanowienie, to rozwiązaniem przybliżonym byłaby wartość:

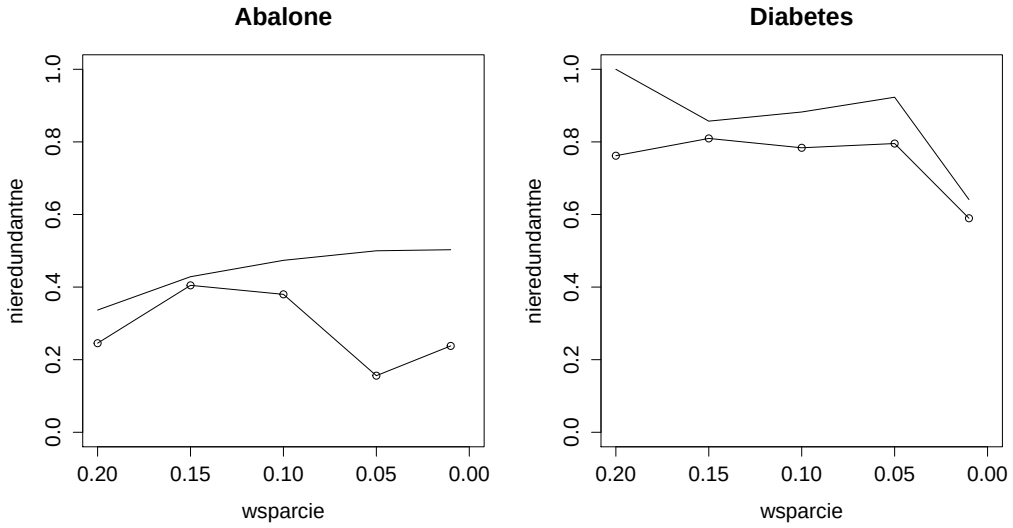
$$liczba\_przedzialow = \frac{1}{minimalne\_wsparcie} \tag{2}$$

Niestety taka wartość sprawia, iż każdy powstały podzbiór ma wsparcie na granicy minimum, dlatego konieczne są dalsze modyfikacje. Oparcie tego parametru na wynikach MQTC pozwala zrezygnować z tych szacowań.

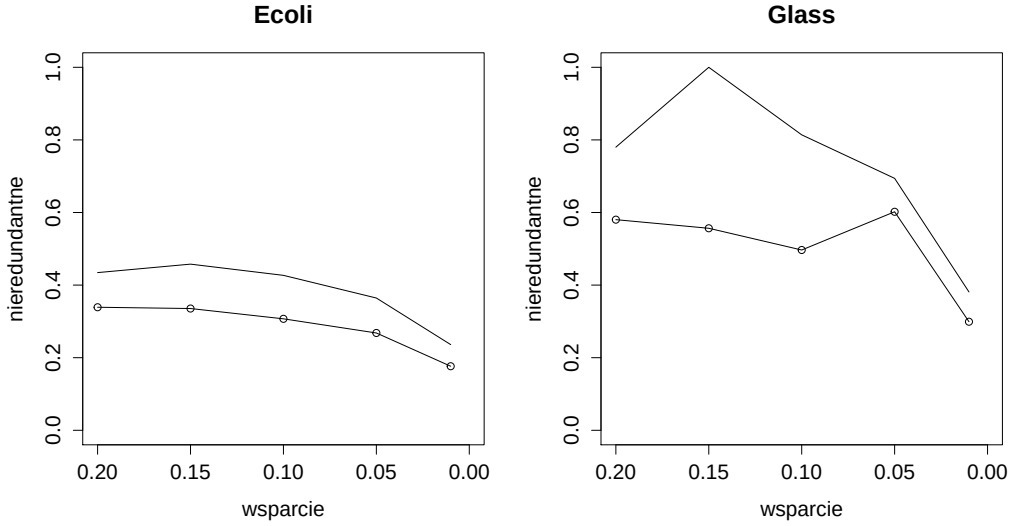
Wyniki

Zmodyfikowany algorytm Quality Threshold sprawuje się lepiej na przedstawionych zbiorach niż prosty podział na równe zbiory. Ta wyższość dotyczy oczywiście wybranego kryterium, czyli względnej liczby interesujących reguł.

Warto dokonać dokładniejszej analizy informacji zebranych podczas eksperymentów. Jako przykład posłuży zbiór Ecoli opisany w tabeli 4. Za-



Rysunek 6: Porównanie liczby reguł nieredundantnych dla algorytmu MQTC (linia ciągła) oraz Equi-depth (linia z punktami) dla dwóch zbiorów: Abalone oraz Diabetes. (uwagi w rozdziale 11.2)



Rysunek 7: Porównanie liczby reguł nieredundantnych dla algorytmu MQTC (linia ciągła) oraz Equi-depth (linia z punktami) dla dwóch zbiorów: Ecoli oraz Grass. (uwagi w rozdziale 11.2)

wiera ona liczbę wszystkich znalezionych reguł, reguł nieredundantnych oraz względną liczbę reguł nieredundantnych dla obu algorytmów i czterech poziomów minimalnego wsparcia.

Zbiór *Ecoli* nie jest zbiorem dużym, ale samych reguł byłoby dużo więcej gdyby nie ograniczenie procedury *Apriori*, która nie szukała reguł dłuższych niż 5. Taka restrykcja nie ma wpływu na wyniki eksperymentu ale znacząco ułatwia go pod względem wydajności. Wąskim gardłem całego procesu jest właśnie wyszukiwanie reguł.

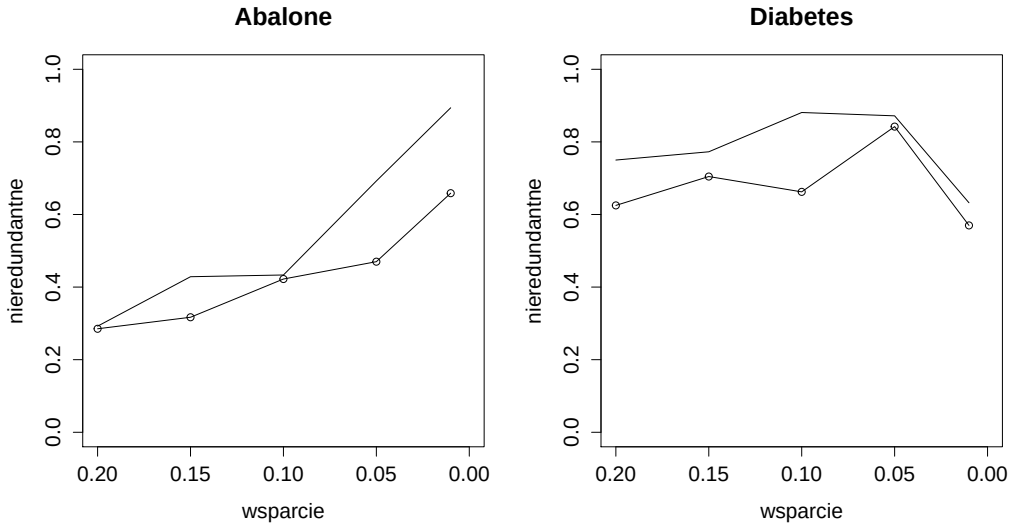
W tabeli 4 można zaobserwować wzrost liczby znajdowanych nieredundantnych reguł wraz ze spadkiem wartości zakładanego minimalnego wsparcia. Podobną tendencję zachowuje suma wszystkich reguł, lecz stosunek tych dwóch liczb staje się coraz mniejszy. Pomimo, że wykresy 6 przedstawiają miejscowe wzrosty tej proporcji, to jednak należy zwrócić uwagę, że ogólna tendencja jest zachowana. To oznacza, że im mniejsze wsparcie tym gorsze pojawiają się reguły.

Wynikom można zarzucić brak zdecydowanej przewagi nowego algorytmu. Można dojść do błędnego wniosku, że skoro algorytm *Equi-depth* jest wydajniejszy, a jego działanie daje wyniki niewiele gorsze niż MQTC, to że tego drugiego można nie brać pod uwagę przy projektowaniu dyskretyzacji dla eksploracji reguł ilościowych. Jednak warto przypomnieć, że użyty tutaj algorytm *Equi-width* wie ile przedziałów znajduje się w zbiorze. Jest tak dzięki temu, że MQTC dokonuje analizy danych jako pierwszy. Pokazuje ile przedziałów jest potrzebnych by zaspokoić warunki eksploracji. Gdyby nie taki schemat działania, manualne ustalenie parametru liczby podziałów wpłynęłoby na wyniki *Equi-width*, które byłyby gorsze gdyby użyto wartości ze wzoru (2).

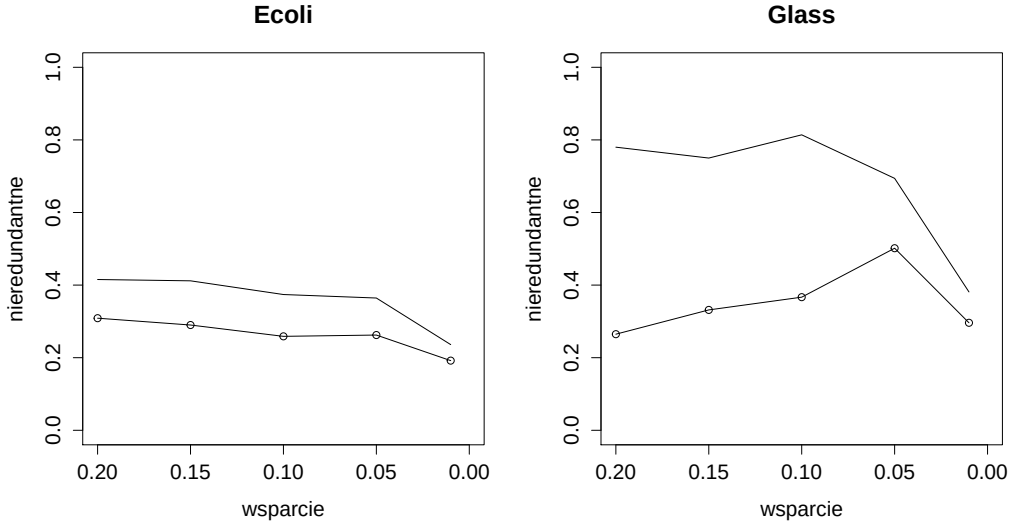
11.3 MQTC w porównaniu z K-means

W testach wykorzystano implementację K-means z pakietu języka R. Eksperymenty ze zwykłym algorytmem K-means byłyby zbyt narażone na wpływ czynników losowych. Rozwiązaniem tego problemu byłoby kilkukrotne uruchamianie grupowania i uśrednianie wyników (albo wybór najlepszego). Na szczęście język R dysponuje implementacją K-means o wzmocnionej „odporności" na losowość procedury.

Wykresy 8 oraz 9 ponownie wskazują wyższość algorytmu MQTC, jednak tym razem dystans między konkurującymi procedurami jest jeszcze mniejszy. Powodem takiej sytuacji jest fakt, że K-means w wersji stabilnej dąży do minimalizacji niepodobieństwa członków grup. Equi-depth troszczył się wyłącznie o odpowiednią liczbę elementów w podgrupie, stąd jego wyniki zawierały więcej reguł redundantnych.



Rysunek 8: Porównanie liczby reguł nieredundantnych dla algorytmu MQTC (linia ciągła) oraz K-means (linia z punktami) dla dwóch zbiorów: Abalone oraz Diabetes. (uwagi w rozdziale 11.3)



Rysunek 9: Porównanie liczby reguł nieredundantnych dla algorytmu MQTC (linia ciągła) oraz K-means (linia z punktami) dla dwóch zbiorów: Ecoli oraz Grass. (uwagi w rozdziale 11.3)

Procedura K-means zna liczbę przedziałów przy której spełnione są warunki procesu eksploracyjnego. Parametr jest wyznaczany podobnie jak w poprzednim przypadku - jest to średnia liczba przedziałów dla wszystkich atrybutów. Dzięki takiemu ustaleniu oba algorytmy mają równe szanse na zwycięstwo. Mimo to zastosowanie MQTC prowadzi do uzyskania bardziej interesujących reguł ilościowych.

12 Podsumowanie

Celem pracy była eksploracja asocjacyjnych reguł ilościowych. Jako koncepcję rozwiązania wybrano metodę wykorzystującą dyskretyzację, która sprowadza całe zadanie do problemu eksploracji reguł binarnych. Sam proces ich wyszukiwania jest dobrze znany. Istnieje szereg algorytmów skutecznych i mających szeroką gamę indywidualnych cech. Inaczej przedstawia się problem dyskretyzacji, ponieważ w kontekście reguł asocjacyjnych może być niedocenionym etapem całego procesu. Brak poświęcenia mu większej uwagi może doprowadzić do stracenia cennej wiedzy. Prosta i łatwa dyskretyzacja doprowadza do dwóch skrajności: gubienia reguł, albo produkcji wielu nieinteresujących.

Warto rozważyć użycie zmodyfikowanego algorytmu Quality Threshold (MQTC), jeśli jedną z pożądanych cech reguł ilościowych jest ich wysoka jakość i naturalność. Jakość jest tutaj rozumiana jako ilość reguł nieredundantnych (co wykazały eksperymenty z rozdziału 11.2 i 11.3). Również istotny jest wygląd uzyskanych przedziałów, ponieważ granice każdej z grup są elementem nazwy nowego atrybutu, a następnie wchodzą w skład reguł ilościowych.

Prosty podział zbioru na przedziały algorytmem *Equi-depth* niesie wyłącznie informacje o koncentracji próbek w podgrupach. Dla przykładu niech próbki będą uporządkowane, a ich identyfikatory niech tworzą zakres $\langle 0; 100\rangle$. Jeśli podzbiorów ma być pięć to zawsze należy zadać sobie pytanie: skąd pewność, że powinno być ich akurat pięć? Oto przykład: $\{\langle 0; 19\rangle$, $\langle 20; 39\rangle$, $\langle 40; 59\rangle$, $\langle 60; 79\rangle$, $\langle 80; 100\rangle\}$. Każda podgrupa zawiera w przybliżeniu 20 próbek. Jeśli wsparcie wymagane przez algorytm eksploracji reguł nie jest większe niż 20 próbek to tak podzielony atrybut może być elementem reguł, na przykład reguły opartej na podzbiór o identyfikatorach $\langle 20; 39\rangle$:

$$A : \langle 243, 278\rangle \rightarrow B = 1$$

gdzie:

$$A(20) = 243; A(39) = 278$$

Niestety użytkownika nie interesuje to, że granice $\langle 243; 278\rangle$ są wymuszone przez algorytm dyskretyzacji. Obserwując regułę użytkownik będzie

myślał, że akurat ten przedział ($\langle 243; 278\rangle$) był najsilniejszy wśród innych przedziałów i dlatego mogła powstać powyższa reguła. Niestety prawda jest inna – to procedura analizy skupień wymusiła taki podział.

Natomiast algorytm MQTC uwzględnia strukturę danych co może doprowadzić do innego podziału, na przykład następującego: $\{\langle 0; 24\rangle, \langle 25; 50\rangle, \langle 51; 71\rangle, \langle 72; 100\rangle\}$, który prezentuje dodatkową informację na temat samej ilości grup (nie pięć, a cztery) i ich rozkładu próbek (nie równo po 20).

Efektem powstania niniejszej pracy jest nowy deterministyczny algorytm o intuicyjnym parametrze sterującym. Ta metoda analizy skupień została stworzona z myślą o regułach ilościowych. Dlatego reguły asocjacyjne, powstałe w oparciu o wyprodukowany dzięki MQTC zbiór binarny, przedstawiają nie tylko wiedzę na temat powiązań pomiędzy atrybutami, ale również informację na temat natury zakresów ich dziedzin.

12.1 Dalsze prace

Stworzenie algorytmu MQTC nie powinno być końcem pracy nad udoskonaleniem oryginalnego algorytmu QTC. Należy przypomnieć, że wprowadzone modyfikacje polegały przede wszystkim na dostosowaniu tego grupowania do celów eksploracji reguł. Natomiast dalsze prace będą dotyczyć zwiększenia wydajności algorytmu. W ramach każdej jego iteracji analizowany jest cały (dotychczas niewykorzystany) zbiór i wybierany jest najlepszy kandydat. Można zauważyć, że w momencie odrzucenia reszty kandydatów tracone jest wiele cennych informacji na temat rozkładu próbek w nich zawartych. Pomysłem na zwiększenie wydajności jest próba ograniczenia zbioru próbek, które będą analizowane w następnej iteracji na podstawie informacji z aktualnego kroku.

Kolejnym kierunkiem dalszych prac jest dziedzina rozmytej analizy skupień (ang. *fuzzy clustering* - rozdział: 4.3.1). Chodzi o połączenie cech algorytmu QTC oraz zbioru rozmytego, tak by zbudować w pełni deterministyczny algorytm grupowania rozmytego sterowanego parametrem jakościowych (jak na przykład minimalne wsparcie). Następnym krokiem jest konieczność opracowania zabezpieczenia na wypadek tworzenia reguł z kilkoma składnikami jednego atrybutu ilościowego. Jednym z pomysłów jest automatyczne wyznaczenie progu odcięcia. W uproszczeniu: jeśli próbka będzie posiadać prawdopodobieństwo przynależności do zbioru $A[2-14]$ mniejsze od wartości progu odcięcia to dana próbka jest wykluczana z tego zbioru (formalnie otrzymuje wartość przynależności równą zero). Niestety istnieje ryzyko, że takie podejście da podział bardzo zbliżony do wyników samego MQTC, co należy również zbadać.

Literatura

[1] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami, *Mining association rules between sets of items in large databases.* In Peter Buneman and Sushil Jajodia, editors, SIGMOD Conference, pages 207–216. ACM Press, 1993

[2] Rakesh Agrawal, Ramakrishnan Srikant, *Mining Quantitative Association Rules in Large Relational Tables*, SIGMOD Conference, ACM Press, 1996

[3] Tadeusz Morzy, *Eksploracja danych. Metody i algorytmy, PWN, 2013*

[4] Jain, A.K., Murty M.N., and Flynn P.J. (1999): Data Clustering: A Review, ACM Computing Surveys, Vol 31, No. 3, 264-323. http://www.cs.rutgers.edu/ mlittman/courses/lightai03/jain99data.pdf

[5] Ying Yang, Geoffrey I. Webb, Xindong Wu, Discretization Methods, Springer, 2005

[6] Lior Rokach, *A survey of Clustering Algorithms*, Data Mining and Knowledge Discovery Handbook, Springer, Londyn, 2010

[7] Shyam Boriah, Varun Chandola, Vipin Kumar, *Similarity Measures for Categorical Data: A Comparative Evaluation*, 8 SDM SIAM, Atlanta, 2008

[8] StatSoft (2006). *Elektroniczny Podręcznik Statystyki PL*, Krakow, WEB: http://www.statsoft.pl/textbook/stathome.html.

[9] Encyclopedia of Machine Learning, Springer, 2010

[10] Bache K. Lichman M. *UCI Repozytorium danych dla uczenia maszynowego (ang. Machine Learning Repository).* Irvine, CA: University of California, School of Information and Computer Science, 2013. [http://archive.ics.uci.edu/ml]

[11] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); *The WEKA Data Mining Software: An Update*; SIGKDD Explorations, Volume 11, Issue 1.

[12] Brian Lenty, Arun Swamix, Jennifer Widom, *Clustering Association Rules*

[13] S. Jaroszewicz, D Simovici, *Pruning redundant association rules using maximum entropy principle*, Springer, 2002

[14] Michael Hahsler, Bettina Grün, and Kurt Hornik. *Arules – A computational environment for mining association rules and frequent item sets.* Journal of Statistical Software, 14(15):1–25, October 2005.

[15] Heyer, L., Kruglyak, S., Yooseph, S. (1999). *Exploring expression data: Identification and analysis of coexpressed genes.* Genome Research, 9, 1106–1115

[16] Kent Beck, *Test Driven Development: By Example*, Addison-Wesley, Boston, USA 2002

[17] Kazimierz Krzysztofek, Marek Szczepański, *Zrozumieć rozwój - od społeczeństw tradycyjnych do informacyjnych*, strony 186–189, Wydawnictwo Uniwersytetu Śląskiego, Katowice, 2002

[18] Justyna Berezowska, Michał Huet, *Społeczeństwo informacyjne w Polsce. Wyniki badań statystycznych z lat 2009-2013*, strony 20–22, Główny Urząd Statystyczny, Szczecin, 2014