Jak dyskretnie reprezentować wartości niedyskretne

Hung Son NGUYEN, Warszawa

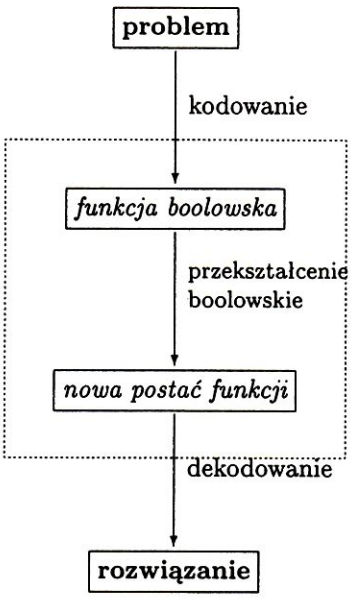
Praca została częściowo wsparta przez grant nr 8T11C02519 fundowany przez Komitet Badań Naukowych.

Jest to rozwinięty zapis odczytu wygłoszonego na XXVII Szkole Matematyki Poglądowej „Matematyka w informatyce i vice versa"; Grzegorzewice, sierpień 2001.

1. Wprowadzenie

Logika matematyczna dostarcza skutecznych narzędzi do modelowania i automatyzacji procesów wnioskowań. Znalazła ona zastosowania w wielu nowoczesnych dziedzinach. Do dziedzin takich należy sztuczna inteligencja (obejmująca np. robotykę, systemy uczące się, układanie harmonogramów zadań, planowanie) czy też gwałtownie rozwijająca się obecnie nowa dziedzina odkrywania wiedzy z danych.

Jedną z najprostszych logik jest logika rachunku zdań oparta na funkcjach boolowskich. Nazwisko George Boole'a (który żył i działał w XIX-tym wieku) najczęściej kojarzy się nam z algebrą Boole'a i jej udziałem w rozwoju komputerów. Rzadko jednak pamiętamy, że G. Boole zaproponował również zastosowanie pewnej metodologii rozwiązywania problemów w oparciu o wnioskowania symboliczne, do których jesteśmy teraz tak przywiązani. Schemat postępowania, jaki George Boole zaproponował w swojej pracy nazywamy wnioskowaniem boolowskim. Przedstawia go diagram na rysunku 1. Na schemacie tym kodowanie ma na celu wyznaczenie specyfikacji problemu w postaci funkcji boolowskiej. Ta specyfikacja jest przekształcana do nowej postaci, z której w wyniku dekodowania bezpośrednio można uzyskać rozwiązanie lub rozwiązania problemu.



Rys. 1. Schemat wnioskowania boolowskiego

Wielką zaletą zaproponowanego przez Boole'a podejścia jest łatwość jego implementacji komputerowej oraz szeroki zakres jego stosowalności do rozwiązywania złożonych problemów w różnych dziedzinach zastosowań. Obecnie istnieje wiele efektywnych algorytmów automatycznego przekształcenia funkcji boolowskich (czy też formuł rachunku zdań, opartego na funkcjach boolowskich). Formuły te, reprezentujące rozwiązywane problemy, są przekształcane w celu wyznaczenia rozwiązań tych problemów.

W tym miejscu warto dodać, że odkryto wiele nowych, w tym również „nieklasycznych", logik starając się rozszerzyć możliwości wyrażania w zwartej formie bardziej złożonych własności. Skonstruowane logiki mają niejednokrotnie znacznie szersze, w porównaniu z rachunkiem zdań, możliwości wyrażania, w bardziej zwartej postaci, złożonych własności różnych obiektów ze (skończonego) uniwersum. Cena, jaką przychodzi za to płacić, jest jednak zbyt wysoka, aby mogły one znaleźć zastosowanie praktyczne, przynajmniej przy obecnej technologii. To ograniczenie stosowalności wiąże się ze złożonością obliczeniową (np. wymaganym czasem obliczeń dla wyznaczenia rozwiązania) podstawowych problemów dotyczących tych logik. Z kolei możliwość rozwiązania tych podstawowych problemów decyduje o stosowalności odkrytych logik w praktyce. W przypadku rachunku zdań udało się dla wielu problemów pokonać tę, jakkolwiek również wysoką, barierę złożonościową. Stało się to możliwe dzięki opracowaniu efektywnych heurystyk umożliwiających skuteczne wyznaczanie (wprawdzie na ogół nie optymalnych, lecz o zadowalającej jakości) rozwiązań ze specyfikacji problemu, reprezentowanej przez formuły o nieraz bardzo dużym rozmiarze.

W artykule tym ograniczamy się do jednego typu przekształceń, które nazywa się sprowadzaniem do postaci DNF, oraz do przykładu zastosowania schematu wnioskowania boolowskiego do rozwiązywania problemu dyskretyzacji cech o wartościach rzeczywistych obiektów, reprezentowanych w tablicach danych.

2. Funkcje boolowskie

Funkcje postaci $f : \mathbb{B}^n \to \mathbb{B}$, gdzie $\mathbb{B} = \{0, 1\}$, nazywamy (n–argumentowymi) funkcjami boolowskimi.

Każdą funkcję boolowską można przedstawić za pomocą zmiennych i operatorów logicznych takich jak $\vee, \wedge, \neg$ (czyli formuły rachunku zdań). Istnieje wiele zapisów tej samej funkcji boolowskiej. Wyróżniamy wśród nich dwie postacie

zwane postaciami normalnymi. Formuła jest w *normalnej postaci koniunkcyjnej* (lub CNF, z angielskiego Conjunctive Normal Form), jeśli jest ona koniunkcją czynników będących alternatywami zmiennych lub negacji zmiennych. Formuła jest w *normalnej postaci dysjunkcyjnej* (lub DNF, z angielskiego Dejunctive Normal Form), jeśli jest ona alternatywą składników będących koniunkcjami zmiennych lub negacji zmiennych. Np. formuły

$$\phi_1 = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2), \qquad \phi_2 = (x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2)$$

przedstawiają tę samą dwuargumentową funkcję boolowską

$$f(x_1, x_2) = x_1 \; XOR \; x_2.$$

Zapis ϕ_1 jest w postaci CNF, natomiast ϕ_2 jest w postaci DNF.

Niech $\mathbf{x} = (x_1, ..., x_n)$ i $\mathbf{y} = (x_1, ..., x_n)$ będą elementami zbioru $\mathbb{B}^n$. Mówimy, że $\mathbf{x} \ll \mathbf{y}$, jeśli dla każdego $i = 1, ..., n$ mamy $x_i \leq y_i$. Relacja $\ll$ jest częściowym porządkiem w zbiorze $\mathbb{B}^n$. Funkcję boolowską $f : \mathbb{B}^n \to \mathbb{B}$ nazywamy *monotoniczną*, jeśli dla każdych $\mathbf{x}, \mathbf{y} \in \mathbb{B}^n$, $\mathbf{x} \ll \mathbf{y}$ implikuje $f(\mathbf{x}) \leq f(\mathbf{y})$. Funkcja boolowska jest monotoniczna wtedy i tylko wtedy, gdy można ją zapisać używając formuł rachunku zdań bez użycia negacji. W tej pracy będziemy rozpatrywali wyłącznie monotoniczne funkcje boolowskie.

Jednomian $f' = x_{i_1} \wedge x_{i_2} ... \wedge x_{i_k}$ nazywamy *implikantem pierwszym* funkcji monotonicznej f, jeśli

- jest *implikantem*, tzn. $f'(\mathbf{x}) \leq f(\mathbf{x})$ dla każdego wektora $\mathbf{x}$
- każdy jednomian większy od f' nie jest implikantem.

W tej pracy, dla uproszczenia, będziemy omijali symbol $\wedge$ w formułach. Wówczas jednomian można zapisać jako $f' = x_{i_1} x_{i_2} ... x_{i_k}$.

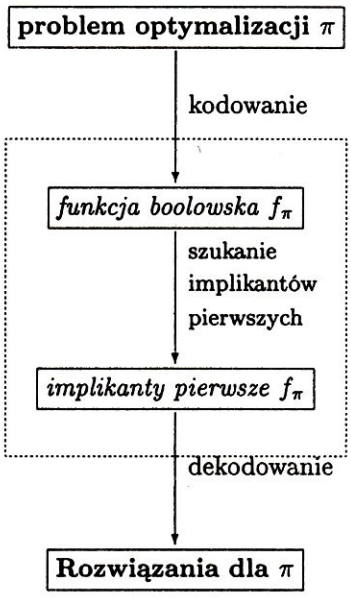
I tak np. funkcja

$$f(x_1, x_2, x_3) = (x_1 \vee x_2)(x_2 \vee x_3)$$

ma 2 implikanty pierwsze: $f_1 = x_2$ i $f_2 = x_1 x_3$.

Można pokazać, że dowolną funkcję boolowską można zapisać w postaci DNF, która jest alternatywą wszystkich swoich imlikantów pierwszych [1]. Oznacza to, że można szukać implikantów pierwszych poprzez sprowadzanie funkcji do postaci DNF.

Schemat wnioskowania boolowskiego przedstawiony na rysunku 1 przyjmuje szczególną postać, jeśli nałoży się wymaganie, aby dekodowana funkcja była sumą implikantów pierwszych funkcji otrzymanej w wyniku kodowania (por. rys. 2). Ta postać schematu wnioskowania boolowskiego jest szczególnie przydatna przy rozwiązywaniu problemów optymalizacyjnych.



Rys. 2. Schemat wnioskowania boolowskiego przy założeniu szczególnej postaci funkcji dekodującej

3. Problem dyskretyzacji w odkrywaniu wiedzy z danych

Odkrywanie wiedzy w bazach danych (lub w skrócie KDD, z angielskiego: Knowledge Discovery in Databases) jest stosunkowo młodym, interdyscyplinarnym kierunkiem badawczym, który powstał z potrzeb analizy i wyciągania wniosków z dużych zbiorów danych. KDD może być rozumiany jako sekwencja złożonych procesów przetwarzania informacji, obejmująca bazy danych, transformacje danych, systemy dialogowe, uczenie maszynowe itp. Każdy z tych procesów wykorzystuje ogromną wiedzę z wielu dziedzin naukowych np. statystyka, analiza harmoniczna, logika matematyczna, systemy uczące się, sieci neuronowe, algorytmy genetyczne. Typowy przebieg KDD składa się z następujących kroków:

- **zgromadzenie i udostępnienie danych**: ten proces jest ściśle związany z problematyką baz danych;
- **przekształcanie i transformacja danych**;
- **wykrywania reguł, wzorców z danych**;
- **interpretacja i ilustracja wyników**;

Dyskretyzacja cech o wartościach rzeczywistych występuje zarówno w drugim, jak i w trzecim kroku. Niżej opisujemy ten problem szczegółowo podając najpierw jego intuicyjny opis. W przestrzeni rzeczywistej skończenie wymiarowej zadana jest skończona liczba punktów etykietowanych różnymi decyzjami,

np. kolorami. Oprócz tego ustalona jest klasa powierzchni, np. hiperpłaszczyzn prostopadłych do osi współrzędnych, hiperpłaszczyzn w ogólnej postaci, czy też powierzchni drugiego stopnia. Zadanie polega na wyznaczeniu minimalnej liczby powierzchni tak, by w obszarach ograniczonych przez te powierzchnie znajdowały się punkty o tej samej decyzji (np. o tym samym kolorze).

Do głównych problemów badanych w ramach KDD należą problemy reprezentacji wiedzy i klasyfikacji obiektów. Problemy te można przedstawić następująco: Rozpatrujemy uniwersum obiektów $\mathbb{U}$. Obiekty z $\mathbb{U}$ są przydzielone do różnych kategorii zwanych klasami decyzyjnymi według pewnej nieznanej nam zasady. Określenie *funkcji decyzyjnej* (decyzja), która przyporządkowuje każdemu obiektowi jego klasę decyzyjną, jest głównym celem problemu klasyfikacji. Jedną z metod szukania tej funkcji jest wnioskowanie indukcyjne (wnioskowanie na podstawie skończonego zbioru przykładów). Problem klasyfikacji może być przedstawiony następująco:

> *Dana jest funkcja decyzyjna dec określona na skończonym podzbiorze (próbka treningowa) $U \subset \mathbb{U}$. Znaleźć rozszerzenie funkcji decyzyjnej na uniwersum $\mathbb{U}$ w postaci algorytmu decyzyjnego, który mając opis jakiegoś obiektu potrafi z dużą dokładnością twierdzić, do jakiej klasy decyzyjnej ten obiekt należy.*

Obiekty z uniwersum są opisane za pomocą atrybutów (lub cech), które też są funkcjami określonymi na obiektach. W problemie klasyfikacji zakładamy, że obiekty są identyfikowane przez ich opisy, czyli atrybuty są znane dla całego uniwersum.

Przykładem problemu klasyfikacji w realnym świecie jest prognozowanie: „czy jutro będzie padał śnieg?" lub przewidywanie „czy postawienie hurtowni w wybranym miejscu gwarantuje sukces finansowy?". W pierwszym problemie obiektami są dni, atrybutami są pomiary meteorologiczne, takie jak: temperatura, ciśnienie, wiatr, zachmurzenie itd., a funkcja decyzyjna określa, czy następnego dnia będzie padał śnieg. W drugim przykładzie, obiektami mogą być hurtownie, atrybutami są cechy opisujące lokalizację, a decyzją może być funkcja określajaca wynik finansowy jako: „dobry", „średni" lub „marny".

Algorytm decyzyjny powinien być przede wszystkim bardzo dokładny dla obiektów z próbki treningowej, zarazem powinien on być na tyle ogólny, by jego odpowiedzi dla nowych, nieznanych obiektów były wiarygodne.

Problem klasyfikacji może być opisany za pomocą *tablicy decyzyjnej*. Tablicą decyzyjną nazywamy strukturę $\mathbb{A} = (U, A \cup \{dec\})$ gdzie

- U nazywa się *zbiorem obiektów*

$$\boxed{U = \{u_1, ..., u_n\}}$$

- A jest zbiorem *atrybutów* postaci

$$\boxed{a_j : U \longrightarrow V_j}$$

- dec jest specjalnym atrybutem zwanym *decyzją*

$$\boxed{dec : U \longrightarrow \{1, ..., d\}}$$

$\mathbb{A}$	a_1	a_2	...	dec
u_1	100	27	...	1
u_2	120	86	...	1
u_3	70	52	...	1
u_4	95	18	...	1
...	...	...	...	...
u_{1200}	71	82	...	2
...	...	...	...	...

Rys. 3.

Przykład tablicy decyzyjnej jest pokazany na rysunku 3.

Wprowadzamy kilka dodatkowych pojęć:

- *Klasy decyzyjne: dec* definiuje podział $U = DEC_1 \cup ... \cup DEC_d$ gdzie

$$\boxed{DEC_k = \{x \in U : dec(x) = k\}}$$

  Zbiór DEC_i nazywamy i-tą klasą decyzyjną.
- *Rozróżnialność:* Dane są obiekty $x, y \in U$ oraz zbiór atrybutów $B \subset A$. Mówimy, że x, y są *rozróżnialne przez* B wtw, gdy istnieje $a \in B$ taki, że

$$\boxed{a(x) \neq a(y)}.$$

  Tablicę decyzyjną nazywamy *sprzeczną*, jeśli istnieją obiekty nierozróżnialne przez A, które należą do różnych klas decyzyjnych. W tym artykule ograniczamy się jedynie do tablic niesprzecznych.

- Zbiór atrybutów $B \subset A$ nazywamy *reduktem* tablicy $\mathbb{A}$ wtw, gdy
  - dla dowolnych obiektów $x, y \in U$
    jeśli $dec(x) \neq dec(y)$ i x, y są rozróżnialne przez A,
    to są również rozróżnialne przez B (B *zachowuje rozróżnialność* A)
  - B jest niezredukowalny (tzn. żaden właściwy podzbiór B nie zachowuje rozróżnialności zbioru A)

Przykład. *Poniższa tablica to przykład małej tablicy decyzyjnej opisującej hurtownie posiadane przez pewną korporację X. Wyniki finansowe* 12 *hurtowni (o numerach od* 1 *do* 12*) są znane. Korporacja X chce rozszerzyć swoją działalność poprzez utworzenie nowych hurtowni. Dostała ona ofertę wynajmu dwóch istniejących hurtowni (o numerach n_1 i n_2). Korporacja X chce prognozować wyniki finansowe tych nowych hurtowni na podstawie zgromadzonych doświadczeń.*

Hurtownia	Jakość obsługi	Jakość towaru	Obok autostrady?	Centrum?	zysk/ strata
ID	a_1	a_2	a_3	a_4	dec
1	dobra	dobra	nie	nie	strata
2	dobra	dobra	nie	tak	strata
3	bdb	dobra	nie	nie	zysk
4	słaba	super	nie	nie	zysk
5	słaba	niska	tak	nie	zysk
6	słaba	niska	tak	tak	strata
7	bdb	niska	tak	tak	zysk
8	dobra	super	nie	nie	strata
9	dobra	niska	tak	nie	zysk
10	słaba	super	tak	nie	zysk
11	dobra	super	tak	tak	zysk
12	bdb	super	nie	tak	zysk
n_1	bdb	dobra	tak	nie	?
n_2	słaba	super	nie	tak	?

Tab. 1

Zanim przystąpimy do analizy tych danych, możemy sobie postawić pytania:

- *czy tablica posiada jakiś redukt, który nie zawiera wszystkich atrybutów? Odpowiedź na to pytanie jest pozytywna, gdyż takim reduktem jest zbiór* $\{a_1, a_2, a_4\}$. *To oznacza, że nie potrzebujemy wszystkich atrybutów do analizy. Ale powstaje przy tym inne pytanie:*
- *czy istnieje redukt zawierający* 2 *atrybuty? lub*
- *jak znaleźć redukt zawierający najmniejszą liczbę atrybutów (minimalny redukt)?*

Ogólniej, chcemy znaleźć rozwiązanie dla następujących problemów związanych z reduktami:

- Czy istnieje redukt zawierający k atrybutów?
- Znaleźć redukt o najmniejszej liczbie atrybutów.

Redukty mają wiele zastosowań w analizie danych, szczególnie w metodach opartych o teorię zbiorów przybliżonych [11, 8].

Istnieją rozmaite metody analizy tablic decyzyjnych, w tym również za pomocą reduktów, ale ze względu na złożoność problemu większość z nich ogranicza się do przypadku, gdy atrybuty mają mało możliwych wartości. W przypadku atrybutów o wartościach rzeczywistych, dane ulegają procesowi dyskretyzacji (lub kwantyzacji), w którym zbiór wartości każdego atrybutu zostaje podzielony na przedziały odpowiadające nowym wartościom. Po takim zabiegu zbiory wartości atrybutów stają się mniej liczne.

Dyskretyzacja atrybutu o wartościach rzeczywistych polega na znalezieniu zbioru końców przedziałów zwanych cięciami. Cięciem nazywamy każdą parę (a, c), gdzie a jest atrybutem, c zaś jest dowolną wartością rzeczywistą. Na rysunku 4 przedstawiamy przykład procesu dyskretyzacji.

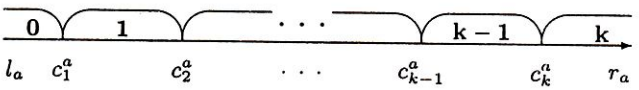
$\mathbb{A}$	a	b	d
u_1	0.8	2	1
u_2	1	0.5	0
u_3	1.3	3	0
u_4	1.4	1	1
u_5	1.4	2	0
u_6	1.6	3	1
u_7	1.3	1	1

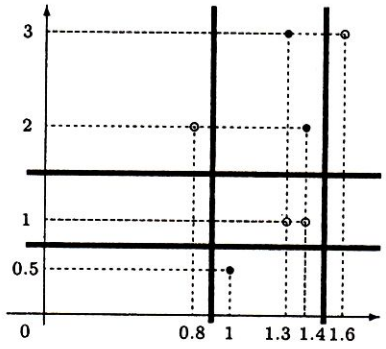
$\Longrightarrow$

$\mathbf{P} = \{(a, 0.9), (a, 1.5), (b, 0.75), (b, 1.5)\}$

$\mathbb{A}^P$	a^P	b^P	d
u_1	0	2	1
u_2	1	0	0
u_3	1	2	0
u_4	1	1	1
u_5	1	2	0
u_6	2	2	1
u_7	1	1	1

Rys. 4. Przykład dyskretyzacji atrybutów o wartościach rzeczywistych. Zbiór cięć $\mathbf{P}$ definiuje nową, dyskretną tablicę decyzyjną





Rys. 5. Ilustracja niesprzecznego zbioru cięć

Niestety, dyskretyzacji atrybutów rzeczywistych nie można przeprowadzać w dowolny sposób, ponieważ zawsze towarzyszy jej utrata informacji w danej tablicy decyzyjnej. Za duża utrata informacji powoduje często spadek jakości znalezionego później algorytmu decyzyjnego. Wielu autorów usiłowało rozwiązać to zagadnienie proponując liczne metody oparte na różnych dziedzinach takich, jak statystyka, teoria informacji, logika, rozpoznawanie wzorców (pattern recognition), uczenie maszyn (machine learning) (por. [2, 4]). Doświadczenia wykazują jednak, że żadna z tych metod nie jest absolutnie lepsza od pozostałych zaś wybór najlepszej metody zależy od natury danych. Niestety nie są znane kryteria wyboru właściwej metody dla konkretnego rodzaju danych.

Dana jest niesprzeczna tablica decyzyjna $\mathbb{A} = (U, A \cup \{dec\})$

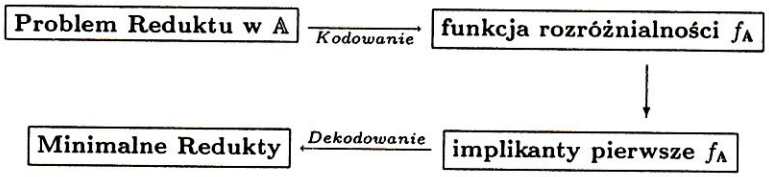
- Mówimy, że cięcie (a, c) *rozróżnia* obiekty x, y, jeśli albo $a(x) \leq c < a(y)$, albo $a(y) \leq c < a(x)$.
- Zbiór cięć $\mathbf{P}$ nazywamy *niesprzecznym* z $\mathbb{A}$, jeśli dla każdej pary $x, y \in U$, takich że $d(x) \neq d(y)$, istnieje cięcie $(a, c) \in \mathbf{P}$ rozróżniające x i y.
- Niesprzeczny zbiór cięć $\mathbf{P}$ nazywamy *nieredukowalnym*, jeśli każdy właściwy podzbiór zbioru $\mathbf{P}$ jest sprzeczny.
- Zbiór cięć $\mathbf{P}_{opt}$ nazywamy *optymalnym* dla $\mathbb{A}$, jeśli $\mathbf{P}_{opt}$ posiada najmniejszą liczbę cięć wśród niesprzecznych zbiorów cięć.

Na rysunku 5 przedstawiamy ilustrację obiektów z tablicy decyzyjnej z rysunku 4 i niesprzecznego (nawet nieredukowalnego) dla niej zbioru cięć.

4. Metoda wnioskowania boolowskiego dla problemów reduktu i dyskretyzacji

4.1. Szukanie reduktu metodą boolowską

Przestawimy przykład zastosowania schematu wnioskowania boolowskiego dla problemu reduktu. W tym celu stosujemy diagram z rysunku 2



Decydującą trudnością w tym schemacie jest etap kodowania. Opiszemy algorytm skonstruowania funkcji boolowskiej $f_{\mathbb{A}}$ dla zadanej tablicy decyzyjnej $\mathbb{A}$ (patrz [12]):

1. *skonstruowanie macierzy rozróżnialności*:
   Ogólnie, macierzą rozróżnialności dla tablicy decyzyjnej $\mathbb{A}$ nazywamy macierz o rozmiarach $n \times n$ (n jest liczbą obiektów w tablicy), w której element w i-tym wierszu i j-tej kolumnie zawiera warunek zapewniający

rozróżnialność między i-tym, a j-tym obiektem. W przypadku problemu reduktu macierzą rozróżnialności jest macierz

$$\mathbb{M}(\mathbb{A}) = [C_{ij}]_{i,j=1}^{n},$$

gdzie element C_{ij} jest zbiorem atrybutów rozróżniających i-ty od j-tego obiektu, czyli $C_{ij} = \{a \in A : a(x_i) \neq a(x_j)\}$.

2. *skonstruowanie funkcji rozróżnialności:*
Każdy atrybut $a_i \in A$ utożsamiamy ze zmienną boolowską α_i, która ma wartość logiczną „PRAWDA" wtedy i tylko wtedy, gdy a_i należy do reduktu. Niech $A = \{a_1, ...a_k\}$, wówczas $X = \{\alpha_1, ..., \alpha_k\}$ jest zbiorem wszystkich zmiennych boolowskich. Dla każdego podzbioru $B \subset A$ definiujemy wartościowanie zmiennych $v_B : X \to \{0, 1\}$ jako funkcję charakterystyczną zbioru B, czyli $v_B(\alpha_i) = 1 \Leftrightarrow a_i \in B$.
Zbiór C_{ij} koduje warunek: *aby odróżnić i-ty obiekt od j-tego obiektu, trzeba mieć co najmniej jeden atrybut ze zbioru C_{ij}*. Warunek ten można opisać funkcją boolowską:

$$\psi_{ij} = \bigvee_{a_m \in C_{ij}} \alpha_m = \bigvee_{a_m(x_i) \neq a_m(x_j)} \alpha_m.$$

A rozróżnialność między obiektami z różnych klas decyzyjnych może być zapewniona za pomocą funkcji boolowskiej:

$$f_{\mathbb{A}} = \bigwedge_{dec(x_i) \neq dec(x_j)} \psi_{ij} = \bigwedge_{dec(x_i) \neq dec(x_j)} \left(\bigvee_{a_m(x_i) \neq a_m(x_j)} \alpha_m \right).$$

Dla każdego podzbioru atrybutów $B \in A$, zdefiniowana funkcja $f_{\mathbb{A}}$ jest spełniona przy wartościowaniu v_B wtedy i tylko wtedy, gdy B zachowuje rozróżnialność między obiektami z różnych klas decyzyjnych.

Mamy twierdzenie

Twierdzenie 4.1 *Zbiór atrybutów $B \subset A$ jest reduktem wtedy i tylko wtedy, gdy funkcja*

$$\psi_B = \bigwedge_{a_i \in B} \alpha_i$$

jest implikantem pierwszym funkcji $f_{\mathbb{A}}$.

Przykład. *Opiszemy działania przedstawionej metody na przykładzie tablicy decyzyjnej w Tablicy* 1.

1. *konstrukcja macierzy rozróżnialności: dla uproszczenia, zostawiamy tylko kolumny związane z obiektami z klasy decyzyjnej „zysk".*

$\mathbb{M}$	1	2	6	8
3	α_1	α_1, α_4	$\alpha_1, \alpha_2, \alpha_3, \alpha_4$	α_1, α_2
4	α_1, α_2	$\alpha_1, \alpha_2, \alpha_4$	$\alpha_2, \alpha_3, \alpha_4$	α_1
5	$\alpha_1, \alpha_2, \alpha_3$	$\alpha_1, \alpha_2, \alpha_3, \alpha_4$	α_4	$\alpha_1, \alpha_2, \alpha_3$
7	$\alpha_1, \alpha_2, \alpha_3, \alpha_4$	$\alpha_1, \alpha_2, \alpha_3$	α_1	$\alpha_1, \alpha_2, \alpha_3, \alpha_4$
9	α_2, α_3	$\alpha_2, \alpha_3, \alpha_4$	α_1, α_4	α_2, α_3
10	$\alpha_1, \alpha_2, \alpha_3$	$\alpha_1, \alpha_2, \alpha_3, \alpha_4$	α_2, α_4	α_1, α_3
11	$\alpha_2, \alpha_3, \alpha_4$	α_2, α_3	α_1, α_2	α_3, α_4
12	$\alpha_1, \alpha_2, \alpha_4$	α_1, α_2	$\alpha_1, \alpha_2, \alpha_3$	α_1, α_4

2. *skonstruowanie funkcji rozróżnialności*

$$\begin{aligned} f \;=\; & (\alpha_1)(\alpha_1 \vee \alpha_4)(\alpha_1 \vee \alpha_2)(\alpha_1 \vee \alpha_2 \vee \alpha_3 \vee \alpha_4)(\alpha_1 \vee \alpha_2 \vee \alpha_4) \\ & (\alpha_2 \vee \alpha_3 \vee \alpha_4)(\alpha_1 \vee \alpha_2 \vee \alpha_3)(\alpha_4)(\alpha_2 \vee \alpha_3)(\alpha_2 \vee \alpha_4) \\ & (\alpha_1 \vee \alpha_3)(\alpha_3 \vee \alpha_4)(\alpha_1 \vee \alpha_2 \vee \alpha_4); \end{aligned}$$

3. *szukanie implikantów pierwszych: na ogół, ten krok jest związany ze sprowadzeniem do postaci* CNF. *Można stosować następujące kroki:*

   - *usuwanie alternatyw regułą* pochłaniania*(t.j. $p \wedge (p \vee q) \equiv p$):*

$$f \;=\; (\alpha_1)(\alpha_4)(\alpha_2 \vee \alpha_3);$$

- *sprowadzanie funkcji f z postaci* CNF *(koniunkcja alternatyw) do postaci* DNF *(alternatywa koniunkcji)*

$$f = \alpha_1\alpha_4\alpha_2 \vee \alpha_1\alpha_4\alpha_3;$$

- *każdy składnik jest reduktem! Zatem mamy 2 redukty:* $R_1 = \{\alpha_1, \alpha_2, \alpha_4\}$ *i* $R_2 = \{\alpha_1, \alpha_3, \alpha_4\}$.

4.2. Heurystyka zachłanna

Problem szukania minimalnego reduktu należy, niestety, do klasy problemów NP-trudnych. To oznacza, że prawdopodobnie nie istnieje algorytm rozwiązujący ten problem w czasie wielomianowym (chyba, że P=NP). Przestawiony wyżej algorytm, w najgorszym przypadku, ma złożoność wykładniczą. Dla dużych tablic decyzyjnych taki czas obliczeń nie jest akceptowany, dlatego w praktyce stosujemy różne algorytmy aproksymacyjne, które w krótkim czasie dają rozwiązanie bliskie optymalnego.

Niżej przedstawiamy prosty algorytm aproksymacyjny zwany „heurystyką zachłanną" dla problemu szukania minimalnego reduktu:

1. wybierz atrybut, który występuje najczęściej w macierzy rozróżnialności;
2. usuń pola macierzy rozróżnialności zawierające wybrany atrybut;
3. powtórz te kroki dopóki wszystkie pola macierzy są puste;
4. zwróć zbiór wybranych atrybutów jako wynik szukania.

Przykład. *Zilustrujemy ten algorytm na podstawie przykładu z tablicy 3:*

- *W macierzy rozróżnialności z poprzedniego przykładu*
  A_1 *- występuje* 23 *razy,* A_2 *- występuje* 23 *razy,*
  A_3 *- występuje* 18 *razy,* A_4 *- występuje* 16 *razy.*
- *Jeśli wybieramy* A_1*, to po usunięciu pól zawierających* A_1 *zostało* 9 *niepustych pól macierzy rozróżnialności. Wśród nich:*
  A_2 *- występuje* 7 *razy,* A_3 *- występuje* 7 *razy,* A_4 *- występuje* 6 *razy.*
- *Jeśli wybieramy tym razem* A_2 *to zostały* 2 *niepuste pola i wśród nich* A_4 *jest zdecydowanym faworytem.*
- *Możemy dostać w ten sposób redukt:* $R_1 = \{A_1, A_2, A_4\}$.

4.3. Dyskretyzacja metodą boolowską

Dana jest niesprzeczna tablica decyzyjna $\mathbb{A} = (U, A \cup \{dec\})$. Analogicznie do przypadku problemu reduktu, problem szukania optymalnego zbioru cięć można kodować za pomocą funkcji boolowskiej jak następuje:

- Niech $\mathbf{C}$ będzie zbiorem proponowanych cięć dla tablicy $\mathbb{A}$. Każde cięcie $(a, c) \in \mathbf{C}$ jest związane ze zmienną boolowską $p_{(a,c)}$.
- Niech $\psi_{x,y}$ będzie funkcją rozróżnialności dla x, y:

$$\psi_{x,y} = \bigvee \{p_{(a,c)} : (a, c) \text{ rozróżnia } x, y\}.$$

- Funkcja

$$\Psi_{\mathbb{A}} = \bigwedge \{\psi_{x,y} : dec(x) \neq dec(y)\}$$

koduje problem dyskretyzacji.

Mamy

Twierdzenie 4.2 *Minimalny implikant pierwszy* $\Psi_{\mathbb{A}}$ *koduje optymalny zbiór cięć.*

Przykład. *Rozpatrujemy tablicę decyzyjną z rysunku* 4. *Cięciami kandydackimi są następujące:*

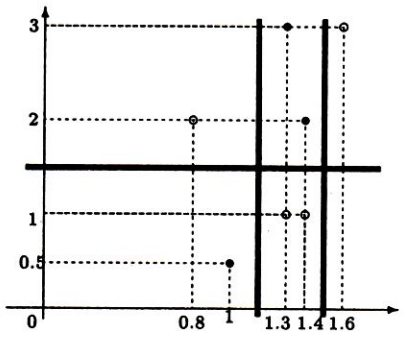
$$(a, 0.9); (a, 1.15); (a, 1.35); (a, 1.5); \quad (b, 0.75); (b, 1.5); (b, 2.5).$$

Oznaczmy przez $p_1^a, p_2^a, p_3^a, p_4^a, p_1^b, p_2^b, p_3^b$ *zmienne boolowskie odpowiadające tym cięciom. Wówczas*

$$\begin{aligned}
&\psi(2,1) = p_1^a \vee p_1^b \vee p_2^b; && \psi(2,4) = p_2^a \vee p_3^a \vee p_1^b;\\
&\psi(2,6) = p_2^a \vee p_3^a \vee p_4^a \vee p_1^b \vee p_2^b \vee p_3^b; && \psi(2,7) = p_2^a \vee p_1^b;\\
&\psi(3,1) = p_1^a \vee p_2^a \vee p_3^b; && \psi(3,4) = p_2^a \vee p_2^b \vee p_3^b;\\
&\psi(3,6) = p_3^a \vee p_4^a; && \psi(3,7) = p_2^b \vee p_3^b;\\
&\psi(5,1) = p_1^a \vee p_2^a \vee p_3^a; && \psi(5,4) = p_2^b;\\
&\psi(5,6) = p_4^a \vee p_3^b; && \psi(5,7) = p_3^a \vee p_2^b;
\end{aligned}$$

$\mathbb{A}^*$	p_1^a	p_2^a	p_3^a	p_4^a	p_1^b	p_2^b	p_3^b	d^*
(u_1, u_2)	1	0	0	0	1	1	0	1
(u_1, u_3)	1	1	0	0	0	0	1	1
(u_1, u_5)	1	1	1	0	0	0	0	1
(u_4, u_2)	0	1	1	0	1	0	0	1
(u_4, u_3)	0	0	1	0	0	1	1	1
(u_4, u_5)	0	0	0	0	0	1	0	1
(u_6, u_2)	0	1	1	1	1	1	1	1
(u_6, u_3)	0	0	1	1	0	0	0	1
(u_6, u_5)	0	0	0	1	0	0	1	1
(u_7, u_2)	0	1	0	0	1	0	0	1
(u_7, u_3)	0	0	0	0	0	1	1	1
(u_7, u_5)	0	0	1	0	0	1	0	1

Tab. 2. Każdy redukt tej tablicy odpowiada jednemu niesprzecznemu zbiorowi cięć tablicy z rysunku 4



Rys. 6. Ilustracja optymalnego zbioru cięć

Funkcja $\Phi_{\mathbf{A}}$ definiowana przez koniunkcję funkcji $\psi_{i,j}$:

$$\Phi_{\mathbf{A}} = (p_1^a \vee p_1^b \vee p_2^b) \wedge (p_1^a \vee p_2^a \vee p_3^b) \wedge (p_1^a \vee p_2^a \vee p_3^a) \wedge (p_2^a \vee p_3^a \vee p_1^b) \wedge p_2^b \wedge (p_2^a \vee p_2^b \vee p_3^b) \wedge (p_2^a \vee p_3^a \vee p_4^a \vee p_1^b \vee p_2^b \vee p_3^b) \wedge (p_3^a \vee p_4^a) \wedge (p_4^a \vee p_3^b) \wedge (p_2^a \vee p_1^b) \wedge (p_2^b \vee p_3^b) \wedge (p_3^a \vee p_2^b)$$

jest funkcją boolowską kodującą problem dyskretyzacji. Po sprowadzaniu do postaci DNF *mamy:*

$$\Phi_{\mathbf{A}} = (p_2^a \wedge p_4^a \wedge p_2^b) \vee (p_2^a \wedge p_3^a \wedge p_2^b \wedge p_3^b) \vee (p_3^a \wedge p_1^b \wedge p_2^b \wedge p_3^b) \vee (p_1^a \wedge p_4^a \wedge p_1^b \wedge p_2^b).$$

Zatem mamy 4 niereduktowalne zbiory cięć, a optymalnym zbiorem cięć jest $\{(a, 1.15), (a, 1.5), (b, 1.5)\}$. *Ilustracja tego zbioru cięć jest pokazana na rysunku* 6.

Można stosować zachłanną heurystykę opisaną w poprzednim rozdziale dla funkcji $\Phi_{\mathbf{A}}$. Pokazaliśmy w pracy [7], że szukanie optymalnego zbioru cięć dla tablicy $\mathbb{A}$ jest równoważne szukaniu minimalnego reduktu pewnej innej tablicy decyzyjnej $\mathbb{A}^*$ skonstruowanej z tablicy $\mathbb{A}$. Przykład tablicy decyzyjnej $\mathbb{A}^*$ skonstruowanej dla tablicy decyzyjnej z rysunku 3 podajemy w Tablicy 2. Formalną konstrukcję tej tablicy i dowód tego faktu zostawiamy Czytelnikowi do sprawdzenia. Tablica $\mathbb{A}^*$ jest jedynie inną prezentacją funkcji $\Phi_{\mathbf{A}}$, ale jest ona bardzo pomocna przy konstrukcji efektywnych algorytmów. Niżej przedstawiamy prosty algorytm zwany „MD-heurystyką".

MD-Heurystyka. Stosując zachłanną heurystyką dla tablicy $\mathbb{A}^*$ mamy algorytm szukania małego zbioru cięć. Niżej przedstawimy kilka właściwości tego algorytmu:

- W algorytmie zachłannym preferujemy cięcia, które rozróżniają najwięcej par obiektów. Dlatego ten algorytm nazywamy *heurystyką maksymalizującą rozróżnialność* lub MD-*heurytyką*, z angielskiego Maximal Discernibility heuristics.
- Można realizować MD-heurystykę w czasie $O(nk \log n|\mathbf{P}|)$, gdzie n jest liczbą obiektów, k jest liczbą atrybutów, $\mathbf{P}$ jest zbiorem cięć znalezionych przez algorytm.

5. Podsumowanie

Przestawiliśmy metodę rozumowania boolowskiego i jej przykładowe zastosowanie w problemach selekcji istotnych atrybutów (redukty) i dyskretyzacji. W praktyce znalazła ona zastosowanie w wielu innych dziedzinach takich, jak: rozpoznawania wzorców (np. rozpoznawania obrazów, sygnałów,...), sztuczna inteligencja, odkrywanie wiedzy (np. odkrywanie reguł asocjacyjnych [10], generowanie drzew decyzyjnych [9], odkrywanie nowych cech [8], ...). We wszystkich tych zastosowaniach metody oparte o rozumowanie boolowskie zawsze wykazywały wysoką skuteczność i niską złożoność obliczeń w porównaniu z innymi metodami.

Po wielu analizach i obserwacjach przy zastosowaniu podejścia boolowskiego, jesteśmy przekonani, że złożoność funkcji kodującej może stanowić miarę oceniającą trudność aproksymacji dla problemu. Pracujemy intensywnie nad tym zagadnieniem. Poza tym, pracujemy również nad metodą „aproksymacyjnego rozumowania boolowskiego", która polega na zastępowaniu funkcji kodującej w standardowym rozumowaniu boolowskim prostszą funkcją boolowską o tej własności, że jej implikanty pierwsze są przybliżonymi rozwiązania problemu. Pierwsze próby okazują się bardzo obiecujące i zachęcają do dalszego badania.

Podziękowanie: Pragnę wyrazić ogromną wdzięczność kolegom, którzy mi pomogli przy powstaniu tego artykułu, zwłaszcza Prof. A. Skoronowi, Dominikowi Ślęzakowi za cenne uwagi i korekty językowe. Największe podziękowanie kieruję do Profesora Andrzeja Skowrona, który jest moim wieloletnim opiekunem naukowym.

Literatura

[1] Brown, F.M., *Boolean reasoning*, Kluwer, Dordrecht 1990.

[2] Catlett, J.: On changing continuous attributes into ordered discrete attributes. In Y. Kodratoff (ed.), *Machine Learning-EWSL-91, Proc. of the European Working Session on Learning, Porto, Portugal*, Lecture Notes in Artificial Intelligence, Springer–Verlag, Berlin, 1991, pp. 164–178.

[3] Chmielewski, M. R., Grzymala-Busse, J. W.: Global discretization of attributes as preprocessing for machine learning. In. T.Y. Lin, A.M. Wildberger (eds.). *Soft Computing.* Rough Sets, Fuzzy Logic Neural Networks, Uncertainty Management, Knowledge Discovery, Simulation Councils, Inc., San Diego, CA, 1995, pp. 294–297.

[4] Dougherty J., Kohavi R., Sahami M.: Supervised and unsupervised discretization of continuous features. In *Proceedings of the Twelfth International Conference on Machine Learning*, Morgan Kaufmann, San Francisco, CA, 1995, pp. 194–202.

[5] Fayyad, U. M., Irani, K.B.: On the handling of continuous-valued attributes in decision tree generation. *Machine Learning* **8**, 1992, pp. 87–102.

[6] V.M. Fayad, G. Piatetsky Shapiro, P. Smyth, R. Uthurusamy (eds.): *Advanced in Knowledge Discovery and Data Mining*, AAAI/MIT Press, 1996.

[7] Nguyen, H. Son: Discretization Methods in Data Mining. In L. Polkowski, A. Skowron (eds.): *Rough Sets in Knowledge Discovery* **1**, Springer Physica-Verlag, Heidelberg, 1998, pp. 451–482.

[8] Nguyen H. Son, Skowron A.: Boolean reasoning for feature extraction problems. In Z.W. Raś and A. Skowron (eds.): Proceedings of Tenth International Symposium on Foundation of Intelligent Systems, ISMIS'97, NC, USA, *Foundation of Intelligent Systems* **LNAI 1325**, Springer Verlag, 1997, pp. 117-126.

[9] H.S. Nguyen and S.H. Nguyen: From Optimal Hyperplanes to Optimal Decision Trees, *Fundamenta Informaticae* **34**No 1–2, 1998, pp. 145–174.

[10] H.S. Nguyen and S.H. Nguyen: Rough Sets and Association rule Generation. *Fundamenta Informaticae,* Vol. 40/4, IOS Press, 1999. pp. 310-318.

[11] Pawlak Z.: *Rough sets: Theoretical aspects of reasoning about data.* Dordrecht, Kluwer, 1996.

[12] Skowron, A., Rauszer, C.: The discernibility matrices and functions in information systems. In R. Słowiński (ed.). *Intelligent Decision Support - Handbook of Applications and Advances of the Rough Sets Theory*, Kluwer Academic Publishers, Dordrecht, 1992, pp. 311–362.