STUDIA INFORMATICA 2000
Volume 21 Number 3 (41)

Marek SIKORA, Paweł PROKSA
Politechnika Śląska, Instytut Informatyki

FUNKCJE OCENY JAKOŚCI REGUŁ W ALGORYTMIE GENEROWANIA REGUŁ APROKSYMACYJNYCH

Streszczenie. W pracy przedstawiono wpływ funkcji oceny jakości reguł, używanej podczas uczenia się reguł, na osiąganą dokładność klasyfikacji oraz na liczbę i złożoność reguł opisujących uczone klasy decyzyjne. Pod tym kątem rozpatrzono trzy funkcje jakości reguł. Podano pewne modyfikacje tych funkcji. Przedstawiono wyniki eksperymentów na trzech zbiorach danych.

RULES QUALITY FUNCTION FOR GENERATION APPROXIMATE RULES ALGORITHM

Summary. In this article one introduced influence of rules quality function used during learning rules on classification accuracy and on number of rules described decision classes. Three rule quality functions were considered. Some modifications of these functions were introduced. The article includes results of experiments conducted for three different data sets.

1. Wstęp

Ważnym problemem związanym z zastosowaniem algorytmów maszynowego uczenia do eksploracji danych jest takie modyfikowanie tych algorytmów, aby otrzymywać krótki i zwięzły opis klas decyzyjnych, których algorytm ma się nauczyć. Warunkiem akceptowalności takiej modyfikacji jest uzyskiwanie takiej samej lub niewiele mniejszej dokładności opisu klas decyzyjnych zwracanych przez algorytm przed i po jego modyfikacji.

Znanych jest kilka sposobów zmniejszania rozmiaru opisu klas decyzyjnych; jedne ingerują w algorytm uczący się w trakcie jego działania [1], inne działają po jego zakończeniu zmniejszając pierwotny opis klas [2].

W przypadku algorytmów uczących się reguł przez zmniejszanie opisu rozumiemy zmniejszanie liczby reguł potrzebnych do opisu klas decyzyjnych.

W artykule rozpatrzono modyfikacje algorytmu generowania wzorców [3], polegające na użyciu innych niż dokładność funkcji oceny jakości reguł oraz wykorzystanie tych funkcji jako kryterium zapamiętania wzorca. Zbadane zostało również jak użycie tych samych funkcji w procesie klasyfikacji wpływa na jej dokładności. Rozpatrywano funkcje wprowadzone przez Michalskiego, mające podłoże bardziej empiryczne niż teoretyczne oraz funkcję wywodzącą się ze statystyki, znaną pod nazwą funkcji Pearsona. Wprowadziliśmy pewne modyfikacje w zakresie określania wartości tych funkcji wtedy, kiedy ich wzór analityczny wskazuje, że są one nieokreślone.

Organizacja niniejszego artykułu jest następująca: w paragrafie drugim przedstawione zostały podstawowe definicje związane z pracą, paragraf trzeci zawiera omówienie używanych funkcji oceny reguł, w paragrafie czwartym przedstawiliśmy otrzymane wyniki, a w paragrafie piątym wnioski i kierunki dalszych prac.

2. Przestrzenie aproksymacyjne, reguły aproksymacyjne, klasyfikacja

2.1. Podstawowe definicje

Problem szukania wzorca w tablicy treningowej wyrażony może zostać jako problem szukania reguł w przestrzeni aproksymacji, poniżej podane zostaną definicje terminów używanych w niniejszej pracy.

Definicja 1

Parametryzowana przestrzeń aproksymacyjna

Parametryzowaną przestrzenią aproksymacyjną nazywamy system $AS_{\#,S}=(U,I_{\#},v_S)$, gdzie:

U jest niepustym zbiorem obiektów;

$I_{\#} : U \rightarrow P(U)$ jest funkcją niepewności;

$V_S: P(U)\times P(U) \rightarrow [0,1]$ jest przybliżoną funkcją przynależności.

Funkcja niepewności określa dla każdego obiektu $x \in U$ zbiór obiektów będących z x w pewnej relacji (podobieństwa, tolerancji, równoważności).

Cechy obiektów z U opisane są za pomocą zbioru atrybutów A, każdy z atrybutów $a \in A$ posiada swoją dziedzinę D_a taką, że dla każdego obiektu $x \in U$ $a(x) \in D_a$. Poza zbiorem A istnieje szczególny rodzaj atrybutu zwany atrybutem decyzyjnym d, każdy element z

dziedziny D_d tego atrybutu wyznacza jedną klasę decyzyjną, ogólnie przez i-tą klasę decyzyjną będziemy rozumieli taki zbiór $C_i \subseteq U$, że $C_i=\{x \in U: d(u)=i\}$.

Mając zadaną relację pomiędzy obiektami $x,y \in U$ względem zbioru cech $B \subseteq A$ możemy zdefiniować funkcję niepewności bardziej szczegółowo. Na przykład dla $x,y \in U$, $a \in A$, $D_a \subseteq R$, gdzie R jest zbiorem liczb rzeczywistych oraz relacji δ_a zdefiniowanej wzorem $\delta_a(a(x),a(y))=|a(x)-a(y)|$, możemy funkcję niepewności zdefiniować następująco: $y \in I_a^{\varepsilon}(x) \Leftrightarrow \delta_a(a(x),a(y) \leq \varepsilon$, w szczególności wartość ε może być funkcją zależną od a,x i y.

Przybliżona funkcja przynależności definiuje stopień zawierania się zbiorów (podzbiorów U) i zazwyczaj rozpatrywana jest jako standardowa funkcja przynależności (ozn. v_{SRI}), wyrażona wzorem:

$$v_{SRI}(X,Y)=\begin{cases} \dfrac{card(X \cap Y)}{card(X)} & X \neq \phi \\ 1 & wpp. \end{cases}$$

Im bardziej zbiór Y zawiera się w zbiorze X, tym wartość funkcji v_{SRI} rośnie, w szczególności jeżeli Y=X to $v_{SRI}(X,Y)=1$.

Definicja 2

Wzorzec

Niech A jest pewnym zbiorem atrybutów, dla każdego $a \in A$ oraz $V_a \subseteq D_a$ każde wyrażenie postaci $(a \in V_a)$ nazywamy deskryptorem, a zbiór wartości V_a zakresem deskryptora.

Jeżeli $B \subseteq A$, to każdą koniunkcję deskryptorów takich, że $a \in B$ i $V_a \subseteq D_a$ nazywamy wzorcem nad zbiorem atrybutów B i zapisujemy w postaci:

$$\bigwedge_{a \in B} (a \in V_a)$$

Zazwyczaj rozpatruje się dwa typy wzorców w zależności od tego, jak liczne są zbiory wartości deskryptorów. Jeżeli we wzorcu chociaż jeden deskryptor posiada zakres wartości większy od jeden, tzn. $card(V_a)>1$, to wzorzec taki nazywamy wzorcem uogólnionym, w przeciwnym przypadku wzorzec nazywamy prostym.

W naszych rozważaniach będziemy się zajmowali wzorcami związanymi z jedną klasą decyzyjną (opisującymi jedną klasę decyzyjną), wzorce takie nazywane są wzorcami decyzyjnymi, symbolicznie wzorce decyzyjne związane z C_i-tą klasą decyzyjną zapisane są w postaci:

$$\bigwedge_{a \in B} (a \in V_a) \rightarrow C_i$$

W przestrzeniach aproksymacyjnych zakres deskryptora jest wyznaczony przez funkcję niepewności. Budowę wzorca rozpoczyna się od wybrania obiektu, który będzie pełnił rolę tzw. generatora wzorca [6]. Postuluje się, aby generator był typowym przedstawicielem klasy decyzyjnej, do której należy, oraz by był on wystarczająco różny od obiektów z pozostałych

klas. Generator można znajdować nadając każdemu obiektowi pewną wagę odzwierciedlającą jego przydatność do bycia generatorem.

Aby ujednolicić słownictwo, wzorce decyzyjne będziemy odtąd nazywać regułami, przemawia za tym fakt, że pojęcie wzorca decyzyjnego jest równoważne pojęciu reguły w przestrzeni aproksymacyjnej oraz to że w literaturze używa się sformułowania funkcje oceny jakości reguł.

W pracy [4] podano pełną informację dotyczącą przestrzeni aproksymacyjnych i budowania w nich reguł.

Definicja 3

Podzbiór obiektów zbioru X wspierających regułę R

Niech $X \subseteq U$ będzie zbiorem obiektów, $C_i \subseteq U$ będzie klasą decyzyjną, $B \subseteq A$ oraz

$R := \bigwedge_{a \in B} (a \in V_a) \rightarrow C_i$ będzie regułą, wówczas zbiór:

$support_X(R) = \{x \in X: \forall a \in B\ a(x) \in V_a\}$

nazywamy zbiorem obiektów z X, które wspierają regułę R.

Łatwo zauważyć, że jeżeli zbiór $X = C_i$, pożądane jest, aby reguła wchodząca w skład opisu klasy C_i była wspierana przez jak największą liczbę obiektów z tej klasy i jak najmniejszą spoza niej. Można tę pożądaną cechę reguły wykorzystać jako funkcję przydatności generowanej reguły.

W algorytmie opisanym w [3] do oceny jakości generowanej reguły użyto miary zwanej dokładnością reguły.

Definicja 4

Dokładność reguły

Niech $C_i \subseteq U$ będzie klasą decyzyjną, $B \subseteq A$ oraz

$R := \bigwedge_{a \in B} (a \in V_a) \rightarrow C_i$ będzie regułą, wówczas liczbę:

$$accuracy(R) = \frac{card(support_{C_i}(R))}{card(support_U(R))}$$ nazywamy dokładnością reguły R.

Dokładność reguły może być miarą pokazującą, jak budować regułę. W algorytmie opisanym w [3] regułę buduje się startując z pustego zbioru deskryptorów, kolejny deskryptor dodaje się tak, aby maksymalizować liczbę obiektów wspierających regułę, jednocześnie minimalizując szerokość deskryptora (liczbę $card(V_a)$). Po każdorazowym dodaniu deskryptora sprawdza się, czy rozszerzona reguła ma lepszą dokładność. Stosując w tym celu inne funkcje oceny reguł, możemy wpływać na proces ich generowania.

Innym sposobem budowania reguł w przestrzeniach aproksymacyjnych jest otrzymywanie reguł z reduktów [4][7], jednak w tym przypadku funkcje oceny reguły były do tej pory używane dopiero po ostatecznym wygenerowaniu wszystkich reguł [2].

Rozważamy możliwość takiej modyfikacji otrzymywania reguł z reduktów, aby były generowane tylko takie reguły, które mają wysoką wartość funkcji oceny reguł, jednak nie jest to tematem niniejszej pracy.

Na koniec należy zwrócić uwagę, że w literaturze dotyczącej maszynowego uczenia istnieje wyraźne rozróżnienie pomiędzy regułą a regułą aproksymacyjną. Regułę zazwyczaj kojarzy się z tzw. regułą dokładną posiadającą taką własność, iż wspierają ją tylko te obiekty ze zbioru U, które należą do klasy decyzyjnej, jaką opisuje reguła. Reguły aproksymacyjne nie posiadają tej własności.

W przypadku przyjętego w niniejszym artykule zapisu reguły

$$R := \bigwedge_{a \in B} (a \in V_a) \rightarrow C_i$$

R byłaby regułą dokładną wtedy i tylko wtedy, gdy $support_U(R) \cap (U \setminus C_i) = \emptyset$ oraz R byłaby regułą aproksymacyjną, gdyby $support_U(R) \cap (U \setminus C_i) \neq \emptyset$.

W tym artykule rozważać będziemy algorytm i jego modyfikacje, który pozwala generować reguły aproksymacyjne. Opisy klas decyzyjnych tworzone przez takie reguły nazywamy opisami aproksymacyjnymi. Własności opisów aproksymacyjnych i ich przewagę w rzeczywistych zastosowaniach maszynowego uczenia można znaleźć w [5] [4].

2.2. Przyjęte założenia w algorytmie sekwencyjnego generowania reguł aproksymacyjnych

W algorytmie zaimplementowanym przez nas tworzymy opis klasy decyzyjnej budując regułę wokół obiektu, który w danej chwili jest najlepszym generatorem, po zbudowaniu reguły usuwamy ze zbioru treningowego wszystkie obiekty wspierające tę regułę (ale tylko te, które należą do tej samej klasy decyzyjnej co generator). Proces ten powtarzamy tak długo, dopóki nie usuniemy wszystkich obiektów ze zbioru treningowego.

Aby wyznaczyć generator, używaliśmy dwóch funkcji $weight: U \rightarrow R$, (gdzie R oznacza zbiór liczb rzeczywistych), nadających obiektowi wagę, odzwierciedlającą typowość obiektu dla klasy decyzyjnej, jaką reprezentuje oraz odległość od obiektów spoza tej klasy decyzyjnej:

$$weight(x) = \frac{1}{n} \sum_{x \neq y} dist(x, y), \qquad weight(x) = \sum_{d(y)=d(x)} dist(x, y) - \sum_{d(z) \neq d(x)} dist(x, z)$$

gdzie $x,y \in U$ $card(U)=n$, $dist(x,y)$ jest pewną miarą odległości. W naszym przypadku $dist(x,y)$ jest odległością Euklidesową lub Hamminga w zależności od tego, czy atrybut jest ciągły czy dyskretny. Pierwsza z funkcji zwana jest średnim dystansem (AVGD), druga dystansem statycznym (STATD) [6].

Jak już napisano, regułę buduje się startując z pustego zbioru deskryptorów, kolejny deskryptor dodaje się tak, aby maksymalizować liczbę obiektów wspierających regułę i minimalizować szerokość deskryptora (liczbę card(V_a)).

Sposób wyboru najlepszej szerokości deskryptora bazuje na zdefiniowaniu wokół każdego atrybutu $a \in A$ generatora x, funkcji niepewności. My robimy to używając strategii LSDD [6] działającej następująco: niech x jest bieżącym generatorem $a \in A$ i wartość atrybutu a dla obiektu x wynosi v (w skrócie a(x)=v), wartość $w \in D_a$ wejdzie w skład deskryptora, jeżeli dist($\mu_a(v)$,$\mu_a(w)$)$\leq\varepsilon$, gdzie dist jest Euklidesową funkcją odległości w R^m, m oznacza liczbę klas decyzyjnych, niech

$$P_{a=v}(C_i)=\frac{support_{C_i}(a=v)}{support_U(a=v)}$$

oznacza liczbę wyrażającą prawdopodobieństwo warunkowe zdarzenia, że losowo wybrany obiekt x o własności a(x)=v należy do klasy decyzyjnej C_i, natomiast

$$\mu_a(v)=<P_{a=v}(d_1),\ldots P_{a=v}(d_m)>$$

oznacza wektor rozkładu tych prawdopodobieństw dla wszystkich klas decyzyjnych.

Algorytm generowania reguły oparty na podanych wcześniej założeniach można podać za [6] w następującej postaci:

Algorytm

Sekwencyjne generowanie reguły aproksymacyjnej

1. $R=\emptyset; R_{best}=\emptyset$
2. **while** ($A\neq\emptyset$)
3. wybierz $a\in A$ i znajdź deskryptor ($a\in V_a$) (kryterium wyboru V_a jest to, aby po dodaniu deskryptora ($a\in V_a$) reguła R wspierała jak najwięcej obiektów z klasy generatora oraz to, aby liczba $|V_a|$ była jak najmniejsza).
4. $R:=R\wedge(a\in V_a)$;
5. **If** (quality(R_{best})<quality(R)) **then** $R_{best}:=R$;
6. $A:=A\setminus\{a\}$
7. **endwhile.**
8. **return** regułę R_{best}.

Można wprowadzić pewien niedeterminizm do algorytmu, wybierając w punkcie trzecim algorytmu deskryptor z prawdopodobieństwem wprost proporcjonalnym do tego, jak zwiększył on liczbę obiektów wspierających regułę i odwrotnie proporcjonalnym do jego szerokości (liczby card(V_a)), pozwala to unikać lokalnych ekstremów w trakcie generowania reguły.

O tym, jak będzie wyglądała reguła, decydują więc punkty 3 i 5 algorytmu. W dalszej części pracy podamy modyfikacje tych punktów.

Na koniec, aby wyrazić podobieństwo pomiędzy przestrzeniami aproksymacyjnymi a sekwencyjnym algorytmem generowania reguł aproksymacyjnych, można zauważyć, że znajdowanie deskryptorów wokół bieżącego generatora to definiowanie dla niego funkcji niepewności. Przybliżona funkcja przynależności (w postaci podanej wcześniej lub innej [5][4]) może zostać użyta do określania stopnia zawierania się zbioru obiektów wspierających daną regułę R w klasie decyzyjnej C_i, informacja ta może służyć do rozwiązywania konfliktów w procesie klasyfikacji [4][7].

2.3. Klasyfikacja

Problem klasyfikacji wiąże się z odpowiedzią na pytanie, do jakiej klasy decyzyjnej zakwalifikować obiekt testowy (o nie znanej wartości atrybutu decyzyjnego), bazując na opisach klas decyzyjnych stworzonych w trakcie uczenia się (analizy zbioru treningowego).

Każdy algorytm klasyfikacji posiada wbudowany mechanizm przyporządkowania obiektowi testowemu klasy decyzyjnej. Wartością weryfikującą jakość algorytmu uczącego się jest dokładność klasyfikacji uzyskana na obiektach testowych, czyli stosunek liczby obiektów klasyfikowanych prawidłowo (przyporządkowano je do dobrych klas decyzyjnych) do liczby obiektów klasyfikowanych w ogóle.

Istnieje wiele strategii klasyfikacji [7], w naszym algorytmie klasyfikujemy obiekt do takiej klasy decyzyjnej, jaką posiada reguła, którą obiekt testowy wspiera. W przypadku jeżeli żadna reguła nie jest wspierana przez obiekt, obliczamy odległości obiektu testowego od reguł i uznajemy, że obiekt wspiera te reguły, którym do niego najbliżej. W przypadku występowania konfliktu wyrażającego się tym, iż obiekt testowy wspiera reguły z różnych klas decyzyjnych, spór taki rozstrzygamy strategią tzw. Max-Max jakości. Polega ona na tym, że dla każdej klasy decyzyjnej C_i obliczamy jej wagę dla obiektu testowego x, zgodnie ze wzorem

$$weight(d_i) = \max_{R}\{quality(R)\},$$

gdzie R wchodzą w skład opisu klasy C_i i są wspierane przez obiekt x.

Klasę dla obiektu testowego wybieramy, znajdując wartość maksymalną wśród wszystkich wag. W tym przypadku badaliśmy, jak na dokładność klasyfikacji wpłynie użycie innej niż accuracy(R) funkcji quality(R).

3. Funkcje oceny jakości reguł

Funkcje oceny jakości reguły mają za zadanie wskazywanie na te reguły w zbiorze reguł, które posiadają najlepsze własności pożądane przez użytkownika.

Zbiorem argumentów dla funkcji oceny jakości reguły jest zbiór reguł RUL, zbiorem wartości jest zbiór liczb rzeczywistych $quality{:}RUL \rightarrow R$. Najlepiej, gdy funkcje można unormować tak, aby przyjmowały wartości ze zbioru [0..1], $quality{:}RUL \rightarrow R$.

W przeważającej ilości przypadków chcemy, aby reguła charakteryzowała się dobrymi parametrami dla dwóch własności: dokładności i pokrycia. Definicja dokładności reguły została wprowadzona wcześniej, obecnie podamy definicję pokrycia reguły.

Definicja 5

Pokrycie reguły

Niech C_i będzie klasą decyzyjną, $B \subseteq A$ jest podzbiorem atrybutów,

i R będzie regułą w postaci:

$$R := \bigwedge_{a \in B} (a \in V_a) \rightarrow C_i$$

wtedy pokryciem reguły R nazywamy liczbę

$$coverage(R) = \frac{support_{C_i}(R)}{card(C_i)}$$

Można używać również innych własności opisujących regułę, np. liczbę deskryptorów w regule lub koszt wyznaczenia wartości danego deskryptora [8], tym niemniej dwie własności opisane wcześniej wykorzystywane są najczęściej w funkcjach oceny reguł.

Wartości potrzebne do wyliczania dokładności, pokrycia oraz innych cech reguły zazwyczaj wylicza się z tablicy kontyngencji [9] budowanej dla każdej reguły.

Tablica kontyngencji jest macierzą $M^{2\times 2}$. Zakładając, że lewą część reguły R oznaczymy α, a prawą β (reguła R będzie miała postać $\alpha \rightarrow \beta$), tablicę kontyngencji można przedstawić w następujący sposób:

Tabela 1

Tablica kontyngencji

	U_β	$U_{\sim\beta}$	
U_α	$n_{\alpha\beta}$	$n_{\alpha\sim\beta}$	n_α
$U_{\sim\alpha}$	$n_{\sim\alpha\beta}$	$n_{\sim\alpha\sim\beta}$	$n_{\sim\alpha}$
	n_β	$n_{\sim\beta}$	$\lvert U \rvert$

Znaczenie przyjętych w tablicy oznaczeń jest następujące:

$n_\alpha = n_{\alpha\beta} + n_{\alpha\sim\beta}$ liczba (U_α - zbiór) obiektów wspierających lewą część reguły R,

$n_{\sim\alpha} = n_{\sim\alpha\beta} + n_{\sim\alpha\sim\beta}$ liczba ($U_{\sim\alpha}$- zbiór) obiektów nie wspierających lewej części reguły R,

$n_\beta = n_{\alpha\beta} + n_{\sim\alpha\beta}$ liczba (U_β - zbiór) obiektów wspierających prawą część reguły R,

$n_{\sim\beta} = n_{\alpha\sim\beta} + n_{\sim\alpha\sim\beta}$ liczba ($U_{\sim\beta}$ - zbiór) obiektów nie wspierających prawej części R,

$n_{\alpha\beta}$ liczba obiektów wspierających prawą i lewą część R.

$n_{\sim\alpha\sim\beta}$, $n_{\sim\alpha\beta}$, $n_{\alpha\sim\beta}$ podobnie jak $n_{\alpha\beta}$.

3.1. Dokładność jako funkcja oceny reguły

Używając oznaczeń wprowadzonych w tablicy kontyngencji dokładność reguły R, można zapisać wzorem:

$$accuracy(R) = \frac{n_{\alpha\beta}}{n_\alpha}$$

dokładność jest tym większa, im mniejsza jest liczba $n_{\alpha\sim\beta}$, inaczej można powiedzieć, że dokładność reguły jest tym większa, im mniej obiektów spoza klasy decyzyjnej reguły, wspiera tę regułę. Wadą dokładności jako funkcji oceny jest to, że nie bierze ona pod uwagę pokrycia, jakie posiada reguła, stąd algorytm budujący reguły tylko na podstawie ich dokładności generuje w skrajnych przypadkach duże ilości reguł posiadających bardzo małe (czasami jednoelementowe) wsparcie.

3.2. Funkcje Michalskiego

Wprowadzone przez R.S.Michalskiego [1] funkcje oceny jakości reguł używane są w rodzinie programów maszynowego uczenia znanych pod nazwą AQ. Obie przytaczane przez nas funkcje biorą pod uwagę zarówno dokładność, jak i pokrycie reguły, dodatkowo jedna z nich rozpatruje ilościowy rozkład klas decyzyjnych w analizowanym zbiorze treningowym.

Pierwsza funkcja (nazwaliśmy ją standardową) jest postaci:

$$quality_{MichalskiStd}(R,w) = \frac{n_{\alpha\beta}}{n_\alpha} * w + \frac{n_{\alpha\beta}}{|U_\beta|}(1-w)$$

gdzie U_β oznacza zbiór obiektów w zbiorze treningowym posiadających taką wartość atrybutu decyzyjnego jak reguła R, stała *w* jest liczbą z przedziału [0,1] odzwierciedlającą preferencje użytkownika (czy bardziej zależy mu na dokładności czy na pokryciu). Łatwo zauważyć, że ułamek w drugim składniku sumy jest zgodny z wprowadzoną wcześniej definicją pokrycia reguły.

Druga funkcja bierze pod uwagę ilość przykładów reprezentujących poszczególne klasy decyzyjne i wartość informacyjną, jaką posiada rozpatrywana reguła [1] z punktu widzenia rozróżnienia przykładów z jej klasy decyzyjnej od pozostałych. U podstaw teoretycznych tej funkcji leży pojęcie entropii. We wzorze funkcji występuje również waga w. Oryginalnie funkcja ma postać:

$$quality_{Michalski}(R,w)=\left(\frac{n_{\alpha\beta}}{|U_\beta|}\right)^{w}*\left(\left(\frac{n_{\alpha\beta}}{n_\alpha}-\frac{|U_\beta|}{|U|}\right)*\left(\frac{|U|}{|U_{\neg\beta}|}\right)\right)^{(1-w)}$$

gdzie $U_{\neg\beta}$ oznacza wszystkie obiekty treningowe o wartości atrybutu decyzyjnego różnej niż ta, jaką ma reguła R.

Pierwszy składnik iloczynu to, tak jak w standardowej funkcji Michalskiego, pokrycie reguły, w drugim uwzględniana jest dokładność reguły, ale również rozkład liczebności poszczególnych klas decyzyjnych w zbiorze treningowym, dodatkowo wprowadzany jest współczynnik normalizujący.

Może się zdarzyć reguła, dla której różnica

$$\left(\frac{n_{\alpha\beta}}{n_\alpha}-\frac{|U_\beta|}{|U|}\right) \tag{1}$$

będzie liczbą ujemną, z tego powodu napotkamy trudności w określeniu wartości funkcji Michalskiego dla takiej reguły.

Postanowiliśmy tak zmodyfikować wzór analityczny tej funkcji, aby unikać takiej sytuacji, przyjęliśmy następującą postać funkcji:

$$quality_{Michalski}(R,w)=\left(\frac{n_{\alpha\beta}}{|U_\beta|}\right)^{w}*\left(\left(1+\left(\frac{n_{\alpha\beta}}{n_\alpha}-\frac{|U_\beta|}{|U|}\right)\right)\left(\frac{|U|}{|U|+|U_{\neg\beta}|}\right)\right)^{(1-w)}$$

nie zmienia to własności tej funkcji Michalskiego, gdyż różnica (1) ma odzwierciedlać, na ile dokładność reguły R:=$\alpha\rightarrow\beta$ wyrażona wzorem

$$\frac{n_{\alpha\beta}}{n_\alpha}$$

jest lepsza od wartości

$$\frac{|U_\beta|}{|U|}$$

czyli od dokładności klas decyzyjnych wynikającej z rozkładu klas w zbiorze treningowym.

Taka modyfikacja pozwoliła na obliczanie wartości funkcji Michalskiego dla każdej reguły. Aby używać funkcji Michalskiego, należy eksperymentalnie dobrać wartość wagi w, co może być uważane za wadę tych funkcji.

3.3. Funkcja χ^2 Pearsona

Ostatnią rozpatrywaną przez nas funkcją analizującą tablicę kontyngencji i mającą swoje źródło w statystyce matematycznej jest funkcja χ^2 Pearsona, mierzącą związki pomiędzy i-tym wierszem a i-tą kolumną w tablicy kontyngencji. Funkcja ma postać:

$$quality_{Pearson}(R)=\frac{(n_{\alpha\beta} * n_{\neg\alpha\neg\beta} - n_{\alpha\neg\beta} * n_{\neg\alpha\beta})^2}{n_\beta * n_{\neg\beta} * n_\alpha * n_{\neg\alpha}}$$

Poszukując reguły o możliwie największej wartości funkcji $quality_{Pearson}$, dążymy do maksymalizacji wartości $n_{\alpha\beta}$ i $n_{\neg\alpha\neg\beta}$, znajdujących się na głównej przekątnej macierzy kontyngencji.

W przypadku funkcji Pearsona łatwo zauważyć, że może zdarzyć się przypadek, w którym funkcja pozostanie nieokreślona. Sytuacja taka zdarzy się jedynie w przypadku, gdy $n_{\neg\alpha}=0$, gdyż z tego, że $n_{\alpha\beta}\geq 1$ (gwarantuje to obiekt będący generatorem), wynika, iż $n_\alpha>0$ i $n_\beta>0$. W przypadku gdyby $n_{\neg\beta}=0$, znaczyłoby to, że reguła ma dokładność 100% oraz pokrywa cały zbiór treningowy, co oczywiście nie ma sensu.

Pozostaje zatem przypadek, w którym $n_{\neg\alpha}=0$, wynika stąd, że $n_{\neg\alpha\beta}=0$ i $n_{\neg\alpha\neg\beta}=0$, co oznacza, iż wszystkie przykłady treningowe wspierają regułę, w takim przypadku zaproponowaliśmy, aby oceniać dokładność takiej reguły i dodatkowo karać ją za pokrywanie przykładów negatywnych (spoza jej klasy).

W przypadku kiedy $n_{\neg\alpha}=0$, wzór analityczny funkcji Pearsona przyjmować będzie postać:

$$quality_{Pearson}(R)=\frac{n_{\alpha\beta} - n_{\alpha\neg\beta}}{n_\alpha}$$

Ogólnie nie ma powszechnie przyjętych sposobów postępowania, w przypadku kiedy wartość jakiejś funkcji oceny jest niezdefiniowana. W literaturze reguły dla których wartość jest niezdefiniowana, albo traktuje się jako mające wartość maksymalną albo odrzuca się je [10]. W naszym przekonaniu oba te podejścia są skrajne i znacznie bardziej użyteczne jest używanie w takich przypadkach funkcji jak najbardziej podobnej do tej, której wartość nie może być określona.

Wszystkie opisane tu funkcje spełniają postulaty postawione w pracy [11], tzn. wartość funkcji rośnie wraz ze wzrostem liczby przykładów pozytywnych (z tej samej klasy co reguła) wspierających regułę, maleje wraz ze wzrostem liczby przykładów negatywnych wspierających regułę. Więcej o funkcjach oceny reguł można znaleźć w pracy [12].

4. Użycie funkcji oceny reguł jako kryterium generowania reguł

Obecnie zaprezentowane zostaną wyniki przeprowadzonych eksperymentów na danych udostępnianych przez Uniwersytet Kalifornijski (dane z różnych dziedzin) w celu porównywania efektywności algorytmów uczących się. Wybraliśmy trzy zbiory charakteryzujące się tym, iż zawierają one dane dyskretne, ciągłe oraz mieszane.

4.1. Zbiory danych

Pierwszy zbiór to dane dotyczące diagnostyki raka piersi znane jako Breast-cancer-wisconsin, zbiór zawiera 569 przykładów opisujących dwie klasy decyzyjne 357 przykładów klasy nowotwór łagodny i 212 przykładów klasy nowotwór złośliwy. Zbiór posiada 30 atrybutów warunkowych opisujących nowotwór, wszystkie atrybuty są typu ciągłego.

Drugi zbiór zawiera dane, udostępnione przez Instytut Onkologii w Lubljanie, dotyczące oceny stanu węzłów chłonnych na podstawie analizy zdjęć rentgenowskich, podobnie rozpatrywaliśmy dwie klasy. Zbiór zawierał 152 przykłady (rozkład klas 81 i 61). W zbiorze jest 19 atrybutów warunkowych. Wszystkie atrybuty są dyskretne.

Ostatnim rozpatrywanym zbiorem danych były dane dotyczące stosowania kart kredytowych. Dane zawierają 690 przykładów opisujących dwie klasy decyzyjne (307 i 383). Każdy przykład charakteryzuje 15 atrybutów warunkowych, wśród których znajdują się atrybuty ciągłe oraz dyskretne zarówno z małą, jak i liczną dziedziną wartości. Przykłady, dla których istniały atrybuty o nieznanych wartościach, zostały ze zbioru treningowego usunięte (5% przykładów).

4.2. Wyniki eksperymentów

Prezentowane w tabelach wyniki zostały uzyskane przy użyciu metodologii testowania znanej jako *10-fold cross-validation*. Metodologia ta polega na losowym podziale zbioru danych na 10 równych podzbiorów i wykonaniu 10 przebiegów algorytmu, za każdym razem 9 z 10 podzbiorów danych służy jako zbiór treningowy, a 1 z 10 jako zbiór testowy. Wyniki w tabelach są średnią arytmetyczną ze wszystkich testów. W naszym przypadku dodatkowo z 9 podzbiorów przeznaczonych do uczenia była wybierana jedna (tzw. podzbiór strojący), na której dobieraliśmy najlepszą wartość ε dla funkcji niepewności (określaliśmy najlepszą szerokość deskryptorów). Wartość ε zmieniano z krokiem 0.05, wyliczano reguły i testowano je na podzbiorze strojącym, do ostatecznego testowania wybierane były reguły, dla których uzyskano najlepszą dokładność klasyfikacji na podzbiorze strojącym.

W przypadku funkcji Michalskiego wartość wagi *w* również dobierana była, tak aby maksymalizować dokładność klasyfikacji (tutaj krok wynosił 0.1).

Najważniejsze wyniki dotyczą liczby generowanych reguł oraz ich złożoności (liczby deskryptorów w regule).Pod uwagę brana była dokładność klasyfikacji uzyskana przy użyciu funkcji oceny reguł zarówno na etapie generowania reguł, jak i na etapie rozstrzygania powstałych konfliktów w czasie klasyfikacji. Rozpatrzono również wpływ użycia funkcji dokładności jako funkcji klasyfikującej na uzyskiwaną dokładność klasyfikacji. W tabelach podane są uzyskane wyniki, liczba podana w nawiasie obok nazwy funkcji oceny reguł oznacza wartość wagi *w*, z jaką obliczana była dana funkcja.

Tabela 2

Wyniki eksperymentów ze zbiorem BREAST 10-CV

Funkcja oceny reguły	Dokładność klasyfikacji	Liczba reguł	Ilość deskryptorów w regule
Dokładność	0,92	5	20,8
MichalskiStd(0.9)*	0,92	2	21,2
Michalski 0.9	0,92	2	22,6
Pearson	0,88	2,5	12,6

Gdyby jakość opisu wyrazić za pomocą funkcji, która byłaby iloczynem dokładności klasyfikacji, liczebności zbioru reguł i liczby, deskryptorów w regule, to użyte przez nas funkcje oceny reguł byłyby lepsze od funkcji, której wartością jest dokładność reguły (zwanej po prostu *dokładność reguły*) w stopniu zależnym od wagi, jaką przyznalibyśmy tym poszczególnym składnikom (dla takiej samej wagi wszystkich składników i po normalizacji składników w trzeciej i czwartej kolumnie otrzymalibyśmy odpowiednio wartości 2.84 2.25 2.32 1.93.). Modyfikacja mechanizmu głosowania przez zastąpienie funkcji oceny reguły dokładnością reguły nie zmieniała dokładności klasyfikacji.

Tabela 3

Wyniki eksperymentów ze zbiorem CREDIT 10-CV

Funkcja oceny reguły	Dokładność klasyfikacji	Liczba reguł	Ilość deskryptorów w regule
Dokładność	0,86	13,8	10
MichalskiStd(0.9)*	0,85	5,7	5
Michalski 0.9	0,86	5,2	4,5
Pearson	0,83	10	8
Pearson Dokładność	0,85	10	8

Podobnie jak dla raka piersi również dla tych danych bardziej sensowne na etapie generowania reguł jest użycie innych niż dokładność funkcji oceny reguły. Dokładność klasyfikacji jest wtedy niewiele mniejsza, lecz jej obniżenie rekompensuje znaczne zmniejszenie zbioru reguł i otrzymanie krótszych reguł. Ważną obserwacją jest to, że używanie w trakcie klasyfikacji dokładności reguły, w przypadku użycia na etapie generowania reguły funkcji Pearsona poprawia dokładność klasyfikacji. W pozostałych przypadkach użycie dokładności na etapie klasyfikacji nigdy nie pogarszało dokładności klasyfikacji.

Tabela 4

Wyniki eksperymentów ze zbiorem LYMPHO 10-CV

Funkcja oceny reguły	Dokładność klasyfikacji	Liczba reguł	Ilość deskryptorów w regule
Dokładność	0,75	41	10,8
MichalskiStd(0.9)*	0,76	6,1	1,6
MichalskiStd(0.9)* Dokładność	0,72	6,1	1,6
MichalskiStd(0.6)*	0,79	11,2	2,7
MichalskiStd(0.6)* Dokładność	0,78	11,2	2,7
Michalski 0.2	0,80	21,2	11,4
Michalski 0.2 Dokładność	0,82	21,2	11,4
Pearson	0,75	10	4,1
Pearson Dokładność	0,78	10	4,1

Ostatni analizowany zbiór danych potwierdza wcześniejsze spostrzeżenia. We wszystkich trzech przypadkach widać, że potencjalnie najlepszą funkcją oceny reguł jest funkcja Pearsona, gdyż stosując ją, uzyskuje się małe zbiory prostych reguł bez znaczącego spadku dokładności klasyfikacji. Dla funkcji tej nie trzeba ustalać żadnych dodatkowych parametrów oraz ma ona silne podłoże teoretyczne. Daje to możliwości do dalszego upraszczania opisu klas poprzez odrzucanie tych reguł, dla których wartość funkcji jest poniżej progu istotności odczytanego (lub wyliczonego) z tablicy rozkładu χ^2. Widać też, że w przypadku kiedy funkcja Pearsona generuje dużo reguł, używanie na etapie klasyfikacji dokładności tych reguł podnosi dokładność takiej klasyfikacji.

Funkcje wprowadzone przez Michalskiego są również efektywne, trudnością jest dobranie odpowiedniej wartości wagi *w*. Z naszych eksperymentów wynika, że należy używać dużych wartości *w*, czyli kłaść nacisk na pokrycie reguły. Ogólnie jednak wartość *w*

należy dobrać do konkretnych danych. Dla wszystkich trzech zbiorów danych lepsze wyniki uzyskano dla standardowej funkcji Michalskiego, ale należy zaznaczyć, że rozkład klas decyzyjnych w danych przez nas analizowanych nie był dramatycznie różny, co nie pozwoliło w pełni pokazać możliwości drugiej funkcji Michalskiego. We wszystkich przypadkach, w których użyto innej niż dokładność funkcji oceny reguł, uzyskano krótszy czas obliczania reguł.

5. Podsumowanie

Z otrzymanych wyników wynika, że sensowne jest stosowanie funkcji oceny reguł już na etapie ich generowania. Można w ten sposób otrzymać mniejszy opis klas decyzyjnych bez zmniejszenia dokładności uzyskanej klasyfikacji. Pożądane jest aby w przypadku wybrania algorytmu klasyfikacji podobnego do naszego, w trakcie klasyfikacji używać dokładności reguł do decydowania, jaką klasę przyznać obiektowi testowemu.

Spośród wielu funkcji oceny reguł wybraliśmy te, które przez wielu autorów uznawane były za obiecujące [12], należałoby jednak potwierdzić statystycznie przewagę jednych funkcji nad innymi, gdyż, jak autorzy sprawdzali, dla większości funkcji nie można sprawdzić teoretycznie, że dane funkcje ustalają na zbiorze reguł identyczny porządek (tzn. reguły po posortowaniu ich malejąco według wartości przyznanym im przez funkcje oceny reguł stworzą dwa identyczne ciągi reguł).

LITERATURA

1. K.A.Kaufman, R.S.Michalski: Learning in an inconsistent world. Rule selection in STAR/AQ18. MLI 99-2, May 1999.
2. T. Agotnes: Filtering large propositional rule set while retaining classifier performance. Knowledge Systems Group, Norwegian Univ. of Science and Technology, MSc., 1999.
3. S.H.Nguyen, A.Skwron, P.Synak: Discovery of Data Patterns with Applications to Decomposition and Classification Problems. In Polkowski L., Skowron A.: Rough Stes in Knnowledge Discovery 1, Methodology and Application. Physica-Verlag, 1998.
4. J.Stepaniuk: Knowledge discovery by application of Rough Set Models. Prace IPI PAN, Warszawa, October 1999.
5. G.Drwal, A.Mrózek: System Rclass - software implementation of the Rough Classifier. Intelligent Information Systems, Proceedings of the Workshop, Malbork, June 15-19, 1998.

6. S.A.Nguyen: Regularity analysis and its application in data minig. Doctoral Thesis, Warsaw University, July 1999.
7. J.G.Bazan: Metody wnioskowań aproksymacyjnych dla sysntezy algorytmów decyzyjnych. Praca doktorska, Uniwersytet Warszawski, 1998.
8. A.Mrózek: Modele wnioskowania operatorów i ich wykorzystanie w komputeryzacji obiektów technologicznych. Zeszyty Naukowe Politechniki Śl., Z.14 Informatyka, 1989.
9. M.Sobczyk: Statystyka. Wydawnictwo Naukowe PWN, Warszawa 1997.
10. A.Ohrn: ROSETTA: Technical Reference Manual. The Norwegian Univ. of Science and Technology, Trondheim, Norway, 1998.
11. G.Piatetsky-Shapiro: Discovery, analysis and presentation of strong rules. Knowledge Discovery in Databases, Melno Park, CA:AAAI Press, pp.229-248, 1991.
12. I. Bruha: Quality of decision rules: Definitions and classification schemes for multiple rules. Machine Learning and Statistics:The Interface, John Wiley and Sons, 1995

Recenzent: Prof. dr hab. inż. Antoni Niederliński

Wpłynęło do Redakcji 20 maja 2000 r.

Abstract

In this paper we proposed the modification of sequential algorithm generating approximate rules. The modification involves the use of rule quality functions which are different than accuracy. The original algorithm we considered was proposed by Nguyen, Skowron and Synak. We studied how the accuracy of classification and the complexity of decision classes description depend on rule quality function, which was used in rules generation and in the classification process. We considered three rule quality functions, two introduced by R. S. Michalski and one based Pearson's formula. In some instances values of considered functions are undefined, therefore we proposed indispensable modifications. We conducted experiments with three data sets available in the „Machine Learning Databases" repository: „Breast cancer", „Credit screening" and „Lymphography". As the result of experiments we formulated two conclusions. First, when using the rule quality functions different than accuracy, we can often get simpler decision classes description, meaning less rules, and also less descriptors in rule. Second, the accuracy function used during classification in voting process always gave the best results regardless of a function used during the rules generation.