STUDIA INFORMATICA 2013
Volume 34 Number 2B (112)

Wojciech MALARA, Marek SIKORA, Łukasz WRÓBEL
Politechnika Śląska, Instytut Informatyki

PROGRAM DO INDUKCJI I OCENY REGUŁ KLASYFIKACYJNYCH, ZINTEGROWANY Z PAKIETEM R

Streszczenie. Celem artykułu jest zaprezentowanie oprogramowania umożliwiającego indukcję i ocenę reguł klasyfikacyjnych w pakiecie R. Zaimplementowany algorytm indukcji realizuje strategię generowania kolejnych pokryć zbioru treningowego. Unikalną cechą algorytmu jest to, że może on wykorzystywać różne miary jakości, sterujące procesem wzrostu i przycinania reguł. Prezentowana implementacja jest jedną z pierwszych dostępnych dla środowiska R.

Słowa kluczowe: eksploracja danych, klasyfikacja, reguł klasyfikacyjne

AN R PACKAGE FOR INDUCTION AND EVALUATION OF CLASSIFICATION RULES

Summary: The primary goal of this paper is to present an R package for induction and evaluation of classification rules. The implemented rule induction algorithm employs a so-called covering strategy. A unique feature of the algorithm is the possibility of using different rule quality measures during growing and pruning of rules. The presented implementation is one of the first available for R environment.

Keywords: data mining, classification, classification rules

1. Wprowadzenie

Indukcja reguł to jedna z najstarszych i najbardziej popularnych metod odkrywania wiedzy w bazach danych. Spośród wielu metod reprezentacji wiedzy stosowanych przez algorytmy eksploracji danych i maszynowego uczenia się, reprezentacja regułowa jest najbliższa sposobom zapisu wiedzy przez ludzi. Szczególnym rodzajem regułowej reprezentacji wiedzy, obok m.in. reguł asocjacyjnych [1], reguł rozmytych [7] czy reguł regresyjnych [15, 25], są

tak zwane reguły klasyfikacyjne lub decyzyjne [2, 19]. Reguły decyzyjne znajdują zastosowanie w zadaniach opisu zbioru przykładów oraz klasyfikacji nowych przykładów. Jedną z najpopularniejszych strategii wyznaczania reguł klasyfikacyjnych jest strategia sekwencyjnego pokrywania [9, 17], pozwalająca w efektywny sposób tworzyć zbiory reguł opisujące cały zbiór uczący.

Strategia sekwencyjnego pokrywania wykorzystywana jest przez zdecydowaną większość algorytmów indukcji reguł klasyfikacyjnych, dlatego jednym z głównych czynników różnicującym poszczególne implementacje jest miara (heurystyka), sterująca procesem indukcji reguł. Jak wykazano w wielu pracach [3, 5, 14, 23, 24, 26, 27], ma ona duży wpływ na postać zbioru reguł, a przez to na jego zdolności opisowe i klasyfikacyjne. Takich miar zostało zdefiniowanych wiele, lecz nie jest możliwe przedstawienie jednolitej rekomendacji użycia danej miary w zależności od charakterystyki analizowanego zbioru danych [14, 24, 26, 27]. W pracach [26, 27] przedstawiono wyniki badania efektywności szerokiego spektrum miar oceny jakości reguł. Efektywność badano ze względu na zdolności klasyfikacyjne i opisowe otrzymanych zbiorów reguł, na tej podstawie zidentyfikowano zbiór miar najbardziej efektywnych. Stosowanie tych miar pozwala na osiągnięcie wyników lepszych niż uzyskują to najbardziej popularne algorytmy indukcji reguł [24, 26, 27].

Celem niniejszego artykułu jest przedstawienie oprogramowania *crules*, udostępniającego implementację pokryciowego algorytmu indukcji reguł, w którym proces tworzenia reguł jest nadzorowany za pomocą wspomnianego już zbioru miar najbardziej efektywnych. Unikalną cechą opracowanego algorytmu jest to, że pozwala on na wykorzystanie różnych miar na etapie wzrostu i przycinania reguły. Prezentowana implementacja dedykowana jest dla pakietu R [22], który jest obecnie jednym z wiodących darmowych narzędzi do analizy danych. Otwartość i popularność środowiska R powodują, że jest ono nieustannie rozwijane i wzbogacane o nowe funkcjonalności. Do chwili obecnej nie przedstawiono jednak implementacji pokryciowego algorytmu indukcji reguł, przeznaczonej stricte dla tego środowiska.

Dalsza część artykułu jest zorganizowana w następujący sposób. W rozdziale drugim, krótko omówiono implementacje algorytmów indukcji reguł w różnych narzędziach do analizy danych. W rozdziale trzecim omówiono zaimplementowany pokryciowy algorytm indukcji reguł oraz przedstawiono zawarte w pakiecie *crules* miary oceny jakości reguł. W rozdziale czwartym opisano pakiet *crules*, umożliwiający indukcję reguł i przeprowadzanie klasyfikacji w środowisku R, a następnie zobrazowano jego wykorzystanie w kilku przykładach analizy konkretnego zbioru danych. W rozdziale piątym zawarto podsumowanie i przedstawiono plany rozwoju pakietu.

2. Pokryciowe algorytmy indukcji reguł klasyfikacyjnych w narzędziach do analizy danych

W wielu narzędziach do analizy danych uzyskanie reprezentacji analizowanego zbioru w postaci reguł klasyfikacyjnych jest możliwe zwykle poprzez zamianę tak zwanego drzewa decyzyjnego [20] na zbiór reguł. Z sytuacją taką spotykamy się zwłaszcza w komercyjnych pakietach (np. SAS Enterprise Miner, Statistica Data Miner, GhostMiner) i programach (np. See5, Cubist). Pokryciowe algorytmy indukcji reguł dostępne są natomiast najczęściej w wolnym i/lub otwartym oprogramowaniu, takim jak: Weka [30], RapidMiner [18] czy RSES [4]. W Weka oraz RapidMiner standardowo zawarto implementację algorytmu pokryciowego RIPPER [6]. W środowisku RSES, przeznaczonym do analizy danych metodami oferowanymi przez teorię zbiorów przybliżonych, udostępniono implementację pokryciowego algorytmu LEM [11].

W R indukcję reguł zgodnie ze strategią sekwencyjnego pokrywania udostępnia jedynie pakiet RWeka [13], będący interfejsem do wybranych metod analitycznych zawartych w bibliotece Weka. Niedawno przedstawiono także pakiet Rule-Based Models [16], w którego skład wchodzą implementacje algorytmów indukcji drzew decyzyjnych C5.0 [20] oraz regresyjnych M5 [21].

3. Indukcja reguł klasyfikacyjnych

3.1. Reguły klasyfikacyjne

Niech dany jest skończony zbiór przykładów *Tr*, zwany zbiorem treningowym. Każdy przykład treningowy charakteryzowany jest przez zbiór $A \cup \{d\}$ cech. Dla każdego $a \in A \cup \{d\}$ $a : Tr \to V_a$. Zbiór V_a jest zbiorem wartości atrybutu *a*. Elementy zbioru *A* zwane są atrybutami warunkowymi, z kolei zmienna *d* nazywana jest atrybutem decyzyjnym, a jej wartości wskazuje na przynależność przykładu do pewnego pojęcia (klasy decyzyjnej). Atrybuty warunkowe mogą być typu symbolicznego (nominalnego), tj. przyjmować wartości dyskretne, lub typu numerycznego, tj. przyjmować wartości ze zbioru liczb rzeczywistych. Atrybut decyzyjny jest typu symbolicznego. Każdy przykład *x* należący do zbioru *Tr*, może zostać zapisany jako wektor $x = (x_1, x_2, ..., x_m, y)$, gdzie $a_i(x) = x_i$ dla każdego $i \in \{1,2,...,m\}$ oraz $y = d(x)$.

Regułą klasyfikacyjną (decyzyjną) nazywamy wyrażenie (1):

$$\textbf{\textit{jeżeli}}\ w_1 \wedge w_2 \wedge ... \wedge w_j\ \textbf{\textit{to}}\ d = v \tag{1}$$

Przesłanka reguły składa się z koniunkcji warunków elementarnych. Aby przykład spełniał warunek elementarny, musi on spełniać zapisane w nim ograniczenia. Reguła (1) reprezentuje za-

leżność mówiącą o tym, że przykład spełniający jednocześnie wszystkie ograniczenia zapisane w warunkach elementarnych należy do klasy decyzyjnej reprezentowanej przez wartość v.

Warunek elementarny w definiowany jest zazwyczaj jako wyrażenie o postaci $a\ op\ Z_a$. W wyrażeniu tym a jest atrybutem warunkowym, traktowanym tutaj jako zmienna mogąca przyjmować wartości należące do dziedziny atrybutu a. Operator relacyjny op wskazuje na jeden z symboli relacyjnych należących do zbioru $\{\leq,\geq,<,>,\neq,=,\in\}$, a Z_a jest tzw. zakresem warunku i w zależności od użytego operatora relacyjnego jest wartością lub podzbiorem zbioru V_a. Zbiór wszystkich przykładów z Tr, które mają identyczną wartość atrybutu decyzyjnego, nazywamy klasą decyzyjną.

Przykład spełniający wszystkie warunki wyspecyfikowane w przesłance reguły i należący do klasy decyzyjnej wskazywanej w jej konkluzji, nazywamy przykładem pozytywnym pokrywanym przez regułę. Przykład spełniający przesłankę reguły i nienależący do klasy wskazywanej w jej konkluzji nazywamy przykładem negatywnym pokrywanym przez regułę. Liczbę wszystkich przykładów pozytywnych pokrywanych przez regułę oznaczamy przez p. Liczbę wszystkich przykładów negatywnych pokrywanych przez regułę oznaczamy przez n. Dodatkowo, liczbę wszystkich przykładów pozytywnych w zbiorze uczącym oznaczymy przez P, a liczbę wszystkich przykładów negatywnych przez N.

3.2. Obiektywne miary jakości reguł klasyfikacyjnych

Dla ustalonej strategii indukcji reguł ich zdolności opisowe i klasyfikacyjne silnie zależą od miar stosowanych do oceny wyznaczanych reguł [5, 14, 24, 27]. Miary jakości znajdują zastosowanie do nadzorowania procesu indukcji w fazach wzrostu i przycinania oraz podczas rozstrzygania konfliktów klasyfikacji. Większość miar stosowanych do nadzorowania procesu indukcji reguł można wyznaczyć na podstawie informacji zawartej w tablicy kontyngencji (tabela 1), która charakteryzuje jakość reguły w kontekście ustalonego (najczęściej treningowego) zbioru danych. Miary definiowane na podstawie tablicy kontyngencji nazywane są miarami obiektywnymi, gdyż ocena reguły wykonywana jest jedynie na podstawie informacji i liczby pokrywanych przez nie przykładów pozytywnych (negatywnych) oraz rozkładu przykładów pomiędzy poszczególne klasy decyzyjne. W ocenie reguł nie są brane pod uwagę żadne subiektywne preferencje użytkownika.

W literaturze przedmiotu można znaleźć ponad 50 różnych miar stosowanych do nadzorowania indukcji reguł [5, 10, 12]. Jak wykazano m.in. w [15, 24, 27], niemożliwe jest przedstawienie jednolitej rekomendacji użycia danej miary w zależności od charakterystyki analizowanego zbioru danych.

Miara jakości to funkcja przypisująca regule pewną wartość rzeczywistą, która odzwierciedla jakość reguły w rozważanym zbiorze przykładów. Najważniejszymi miarami służącymi do oceny reguł decyzyjnych są precyzja ($p/(p+n)$) i pokrycie (p/P) reguły.

Tabela 1

Tablica kontyngencji dla reguły pokrywającej
p pozytywnych i n negatywnych przykładów

p	$P - p$	P
n	$N - n$	N
$p + n$	$P + N - p - n$	$P + N$

Algorytmy indukcji reguł dążą do tego, by indukować jedną regułę, pokrywającą wyłącznie wszystkie przykłady należące do danej klasy decyzyjnej, czyli precyzyjną, oraz o pełnym pokryciu. Gdy taka sytuacja nie jest możliwa, dąży się do utworzenia reguł maksymalizujących te dwie miary. Jest to trudne gdyż, zazwyczaj wraz ze wzrostem dokładności maleje pokrycie reguły. Definiuje się zatem różne miary jakości, które starają się oceniać jakość reguł równocześnie z punktu widzenia ich dokładności i pokrycia. Część miar dodatkowo analizuje jeszcze, jaki jest rozkład wszystkich przykładów pozytywnych i negatywnych w analizowanym zbiorze danych, dzięki czemu analizowane są względna dokładności i względne pokrycie reguł.

W pakiecie *crules* zawarto miary (tabela 2), których użycie – jak wykazano w pracach [26, 27] – pozwala na uzyskanie zbiorów reguł charakteryzujących się wysoką dokładnością klasyfikacji (*C1, C2, g-measure*), wysoką średnią dokładnością klas decyzyjnych (*wLap*, *LS*), umiarkowaną liczbą reguł, przy jednoczesnym zachowaniu dobrych zdolności klasyfikacyjnych (*Corr*, *Cohen*), oraz niewielką liczbą ogólnych reguł, o nieco gorszych zdolnościach klasyfikacyjnych (*RSS*).

Tabela 2

Miary oceniające jakość reguł zawarte w pakiecie *crules*

$\frac{p}{p+n+2}$ g-measure (g=2)	$\frac{(p+1)(P+N)}{(p+n+2)P}$ *wLap* (Weighted Laplace)	
$\frac{pN}{nP}$ *LS* (Logical sufficiency)	$\frac{p}{P}-\frac{n}{N}$ *RSS* (Rule specificity and sensitivity)	
$\frac{p}{p+n}-\frac{P-p}{P-p+N-n}$ *s* (s-Bayesian confirmation)	$\frac{pN-Pn}{\sqrt{PN(p+n)(P-p+N-n)}}$ *Corr* (Correlation)	
$\frac{(P+N)\left(\frac{p}{p+n}\right)-P}{\left(\frac{P+N}{2}\right)\left(1+\frac{\frac{p}{p+n}}{\frac{p}{P}}\right)-P}$ *Cohen*	$\left(\frac{(P+N)\frac{p}{p+n}-P}{N}\right)\left(\frac{2+Cohen}{3}\right)$ C1	$\left(\frac{(P+N)\frac{p}{p+n}-P}{N}\right)\left(\frac{1+\frac{p}{P}}{2}\right)$ C2

Oprócz wyżej wymienionych miar, algorytm zaimplementowany w ramach pakietu *crules* umożliwia także zastosowanie na etapie wzrostu reguł miary opartej na entropii warunkowej

[28]. Metoda ta zostanie omówiona w dalszej części pracy, po przedstawieniu ogólnego zarysu algorytmu indukcji.

3.3. Pokryciowy algorytm indukcji reguł

Zaimplementowany w ramach pakietu *crules* algorytm indukcji reguł działa według strategii sekwencyjnego pokrywania [9, 17]. Przykłady z aktualnie rozpatrywanej klasy decyzyjnej stanowią klasę pozytywną (przykłady pozytywne), natomiast przykłady z pozostałych klas stanowią klasę negatywną (przykłady negatywne). Dla każdej klasy decyzyjnej reguły tworzone są dopóty, dopóki istnieją niepokryte przykłady pozytywne. Proces indukcji pojedynczej reguły składa się z dwóch faz: wzrostu i przycinania. W fazie wzrostu (realizowanej za pomocą strategii wspinaczki) do przesłanki reguły dodawane są warunki elementarne. W przypadku atrybutów nominalnych, warunki takie mają postać $(a=v_a)$. W przypadku atrybutów numerycznych, może to być jedna z postaci $(a<v_a)$ lub $(a\geq v_a)$. Dla atrybutów numerycznych wartość v_a jest średnią arytmetyczną z dwóch kolejnych wartości, z pewnego zakresu wartości atrybutu a. Zbiór wszystkich możliwych warunków elementarnych, które mogą zostać dodane do reguły, jest utworzony na podstawie zbioru przykładów pokrywanego przez tę regułę. Na danym etapie wzrostu reguły, do jej przesłanki zostaje dodany ten warunek, który powoduje największy wzrost wartości określonej miary jakości. Proces wzrostu kończy się z chwilą uzyskania reguły dokładnej lub gdy nie ma kolejnych warunków, które mogłyby zostać dodane do reguły.

Po etapie wzrostu reguły następuje etap jej przycinania. Algorytm przycinania stosuje również strategię wspinaczki – usuwane są te warunki elementarne, których usunięcie powoduje największy wzrost wartości określonej miary lub nie powoduje jej spadku. Przycięta reguła zostaje dodana do wynikowego zbioru reguł i proces indukcji reguł rozpoczyna się ponownie dla pozostałych, niepokrytych przez dotychczas utworzone reguły, przykładów pozytywnych.

Poniżej przedstawiono ogólny zarys algorytmu, zapisany w postaci pseudokodu:

```
RuleInduction(examples, growMeasure, pruneMeasure)
Wejście:
   examples: treningowy zbiór przykładów
   growMeasure: miara wykorzystywana we wzroście reguły
   pruneMeasure: miar wykorzystywana w przycinaniu reguły
Wyjście: wynikowy zbiór reguł
ruleSet := Ø
# class to zbiór przykładów o takiej samej wartości atrybutu decyzyjnego
foreach (class in examples)
   uncoveredPositives := class
   while (uncoveredPositives ≠ Ø)
      # reguła o pustej przesłance oraz
      # o konkluzji wskazującej docelową klasę decyzyjną
      rule := Ø
      # zbiór przykładów pokrywanych przez regułę
      # (początkowo reguła pokrywa wszystkie przykłady)
      covered := examples
```

```
        # faza wzrostu reguły
        repeat
            bestCondition = FindBestCondition(covered, uncoveredPositives,
            growMeasure)
            # dodaj wybrany warunek elementarny do przesłanki reguły
            rule := rule ∪ bestCondition
            covered := Covered(rule, examples)
        until rule is precise or bestCondition = ∅
        # faza przycinania reguły
        Prune(rule, examples, pruneMeasure)
        covered := Covered(rule, examples)
        uncoveredPositives := uncoveredPositives \ covered
        ruleSet := ruleSet ∪ {rule}
    end while
end foreach
return ruleSet
```

Funkcję *FindBestCondition* można przedstawić ogólnie w następujący sposób:

```
FindBestCondition(covered, uncoveredPositives, growMeasure)
# wygeneruj zbiór wszystkich możliwych warunków elementarnych
conditions := PossibleConditions(covered, uncoveredPositives)
bestQuality := -∞, bestCondition := ∅
foreach (c in conditions)
    # oceń jakość reguły z dodanym do jej przesłanki warunkiem c
    quality := Evaluate(rule ∪ c, growMeasure)
    if (quality > bestQuality)
        bestQuality := quality, bestCondition := c
    end if
end foreach
return bestCondition
```

Zamiast jednej z miar jakości prezentowanych w tablicy 2, algorytm może w fazie wzrostu wykorzystać entropię warunkową [28]. Rozważając dla danego atrybutu numerycznego a kolejne punkty graniczne v_a, wybierany jest ten, który minimalizuje wartość wyrażenia $\frac{|U_1|}{|U|}Entr(U_1)+\frac{|U_2|}{|U|}Entr(U_2)$, gdzie $Entr(U_i)$ oznacza entropię zbioru U_i, U_1 oznacza zbiór przykładów spełniających warunek $a<v_a$, a U_2 oznacza zbiór przykładów spełniających warunek $a \geq v_a$. Jako najlepszy warunek elementarny wybierany jest ten, dla którego – przy ustalonym już punkcie granicznym v_a – w odpowiednich zbiorach U_1, U_2 znajduje się więcej przykładów pozytywnych.

Wyznaczony zbiór reguł może zostać wykorzystany do opisu danych treningowych (reguły reprezentują zależności, jakie udało się odkryć w danych) oraz do klasyfikacji przykładów. Podczas klasyfikacji przykładów, które nie zostały użyte do wyznaczenia reguł, może dojść do sytuacji, w której przykład pokrywany jest przez kilka reguł wskazujących na różne klasy decyzyjne, wówczas potrzebna jest strategia rozwiązywania takich konfliktów. Najpopularniejszym rozwiązaniem jest głosowanie, które polega na przypisaniu do każdej reguły pewnej wartości numerycznej, zwanej stopniem zaufania, a następnie sumowaniu – w ramach reguł wskazujących na identyczną klasę decyzyjną – stopni zaufania reguł pokrywających klasyfikowany przykład i przyporządkowaniu go do tej klasy decyzyjnej, dla której uzyskano naj-

większą wartość sumy. W pakiecie *crules* stopień zaufania do reguły jest równy wartości jednej z wybranych miar jakości.

4. Implementacja i przykłady użycia

Zasadniczą częścią opisywanego w niniejszym artykule projektu jest implementacja pokryciowego algorytmu indukcji reguł oraz udostępnienie go jako pakietu programu R. W pakiecie tym przetwarzanie i obliczenia wykonywane są przede wszystkim przez skompilowany kod napisany w języku C++, a kod w języku R służy głównie do sprawdzenia poprawności danych wejściowych, a następnie przygotowania ich do przekazania do metod C++ i wywołania tych metod poprzez bibliotekę *Rcpp* [8]. Po zakończeniu przetwarzania, metody C++ zwracają wyniki, które następnie są przekształcane do postaci umożliwiającej przechowywanie ich w obiektach R.

Indukcja reguł odbywa się za pomocą funkcji *crules.* Funkcja ta przyjmuje jako parametr zbiór uczący wraz z formułą określającą zależność atrybutu decyzyjnego od atrybutów warunkowych. W wywołaniu funkcji *crules* należy także podać miary sterujące procesem wzrostu i przycinania reguł. Użytkownik ma do wyboru zestaw predefiniowanych miar z tabeli 2 oraz dodatkowo entropię warunkową jako miarę oceny warunków elementarnych we wzroście reguły. Użytkownik może również sam taką miarę określić poprzez zdefiniowanie funkcji wyznaczającej wartość miary na podstawie tablicy kontyngencji. Do przeglądania wygenerowanego zbioru reguł i ich statystyk służą przeciążone funkcje *print* oraz *summary*, a za pomocą metody *predict* można dokonywać klasyfikacji przykładów podanego zbioru oraz oceny klasyfikatora regułowego. Ocena algorytmu za pomocą metody k-krotnej walidacji krzyżowej odbywa się natomiast za pomocą funkcji *crules.cv*. Stosowana metoda walidacji krzyżowej zachowuje rozkład wartości atrybutu decyzyjnego (losowanie warstwowe) w *k* częściach, na jakie zostaje podzielony zbiór danych.

Pakiet *crules* został umieszczony na platformie R-Forge [29], wspomagającej tworzenie i utrzymywanie pakietów R. Mając dostęp do sieci Internet, instalacji *crules* można dokonać za pomocą polecenia `install.packages("crules", repos="http://R-Forge.R-project.org")`.

W kolejnej części artykułu zostaną przedstawione przykłady zastosowania pakietu *crules* do indukcji reguł, klasyfikacji i oceny klasyfikatora za pomocą metody walidacji krzyżowej. W przykładach zostanie wykorzystany zbiór danych *iris*, który jest standardowo dostępny w środowisku R.

Przykład 1

W przykładzie tym ze zbioru *iris* zostanie wydzielona w sposób losowy część ucząca, stanowiąca 2/3 oryginalnego zbioru. Wykorzystując część uczącą zbioru danych, przeprowadzona zostanie indukcja reguł decyzyjnych. Pod uwagę zostaną wzięte wszystkie atrybuty warunkowe ze zbioru. W fazie wzrostu, do oceny warunków elementarnych zostanie wykorzystana metoda oparta na entropii warunkowej, natomiast w fazie przycinania użytą miarą będzie: C2.

```
> library(crules)
> set.seed(123)
> tr <- sample(nrow(iris), 2/3 * nrow(iris))
> rules <- crules(Species~., iris[tr, ], "c2", "entropy")
```

Za pomocą przeciążonej funkcji *print* dla każdej z reguł można wyświetlić jej precyzję (*Precision*), pokrycie (*Coverage*) oraz ocenę istotności statystycznej (*P.value*):

```
> print(rules)
  Rule
1 IF Petal.Width < 0.8 THEN setosa
2 IF Petal.Length < 4.85 AND Petal.Width in [ 0.8 ; 1.7 ) THEN versicolor
3 IF Petal.Length >= 4.85 THEN virginica
4 IF Petal.Width >= 1.7 THEN virginica

Precision Coverage  P.value
1.0000000 1.0000000 0.000000e+00
1.0000000 0.9354839 0.000000e+00
0.9687500 0.9393939 2.473630e-22
0.9677419 0.9090909 5.511557e-21
```

Zbiorcze podsumowanie całego zbioru reguł dostępne jest natomiast poprzez funkcję *summary*:

```
> summary(rules)
                                                    value
Number of rules:                                4.000000e+00
Average number of elementary conditions per rule: 1.250000e+00
Average precision of rules:                     9.841230e-01
Average coverage of rules:                      9.459922e-01
Average p-value of rules:                       1.439730e-21
```

Przykład 2

Korzystając z utworzonego w przykładzie 1 zbioru reguł, przeprowadzona zostanie klasyfikacja przykładów, które nie zostały wylosowane do zbioru treningowego:

```
> predict(rules, iris[-tr,])
```

Funkcja *predict* zwraca strukturę zawierającą następujące elementy: *acc* – całkowita dokładność klasyfikacji, *bac* – średnia dokładność klasyfikacji klas decyzyjnych, *cov* – jaki procent przykładów ze zbioru testowego jest pokryty przez reguły, *predictions* – wartości klas decyzyjnych przypisanych przez klasyfikator regułowy poszczególnym przykładom, *confusionMatrix* – macierz pomyłek, w której rzeczywista przynależność do klas jest wskazana w wierszach, natomiast przewidywana – w kolumnach. Dodatkowo macierz ta zawiera kolumnę o nazwie *Uknown*, oznaczająca przykłady niezaklasyfikowane do żadnej z klas, oraz kolumnę *Class accuracy*, w której znajdują się wartości dokładności klasyfikacji dla poszczególnych klas decyzyjnych.

Przykład 3

Przykład ten ukazuje wykorzystanie zdefiniowanej przez użytkownika funkcji w środowisku R jako miary jakości reguły wykorzystywanej w fazie wzrostu lub/i przycinania reguł. Ponadto, przedstawiono postać argumentu *formula*, wskazującą na chęć wykorzystania wybranych atrybutów warunkowych w procesie indukcji:

```
> myMeasure <- function(P, p, N, n) {
>     ((p + 1) * (P + N)) / ((p + n + 2) * P)
> }
> crules(Species~Sepal.Width+Petal.Width, iris[tr,], myMeasure, "entropy")
```

Miara zdefiniowana przez użytkownika to funkcja środowiska R, zwracająca wartość rzeczywistą oraz przyjmująca cztery parametry (P, p, N, n) odpowiadające analogicznym symbolom z tablicy kontyngencji.

Przykład 4

W przykładzie tym przeprowadzona zostanie ocena klasyfikatora za pomocą metody walidacji krzyżowej. Przeprowadzonych będzie pięć przebiegów dziesięciokrotnej walidacji krzyżowej:

```
> cv <- crules.cv(Species~., iris, "c2", "entropy", folds = 10, runs = 5)
```

Do podsumowania wyników walidacji krzyżowej służy przeciążona funkcja *mean*:

```
> mean(cv)

$Statistics
                                                        Mean Std deviation
Accuracy:                                         8.906667e-01  5.775042e-02
Class accuracy:                                   8.906667e-01  5.775042e-02
Coverage:                                         9.573333e-01  5.926963e-02
Number of rules:                                  4.000000e+00  1.414214e+00
Average number of elementary conditions per rule: 1.651071e+00  1.491420e-01
Average precision of rules:                       9.660284e-01  1.306454e-02
Average coverage of rules:                        8.625714e-01  6.093491e-02
Average p-value of rules                          9.033438e-14  6.323406e-13

$`Average confusion matrix`
           Predicted
Actual      setosa versicolor virginica Unknown Class accuracy
  setosa         5       0.00      0.00    0.00          1.000
  versicolor     0       4.58      0.34    0.08          0.916
  virginica      0       0.66      3.78    0.56          0.756
```

Należy zauważyć, że średnia macierz pomyłek (*Average confusion matrix*) zawiera wartości średnie dla poszczególnych części walidacji krzyżowej, zatem jeśli zbiór zawiera 150 przykładów i wykonywana jest dziesięciokrotna walidacja krzyżowa, to elementy macierzy pomyłek sumują się do 15 (150/10).

Warto dodać, że jeśli badacz wykonuje serię eksperymentów, przydatna może być także konwersja wyników walidacji krzyżowej do tabeli danych za pomocą funkcji *as.data.frame*. Środowisko R oferuje wiele sposobów eksportu danych z tej struktury do pliku.

5. Podsumowanie

Długodystansowym celem przedstawionych prac jest opracowanie otwartego i ogólnodostępnego oprogramowania udostępniającego wiele metod do regułowej analizy danych. Prace rozpoczęto od implementacji pokryciowego algorytmu indukcji reguł klasyfikacyjnych. Unikalną cechą opracowanego algorytmu jest możliwość wykorzystania różnych miar do sterowania procesem wzrostu i przycinania reguł. Użytkownik ma do wyboru zestaw predefiniowanych miar, które w pracy [27] zostały zidentyfikowane jako najbardziej efektywne. Ponadto, zaproponowane rozwiązanie umożliwia w prosty sposób zdefiniowanie własnej miary, dzięki czemu może ono również służyć jako narzędzie badawcze w zakresie nowych miar oceny jakości reguł.

Całość implementacji została udostępniona w postaci biblioteki dla pakietu R, będącego obecnie jednym z wiodących darmowych narzędzi do analizy danych. Użytkownik dostaje zatem możliwość połączenia przedstawionych w niniejszym artykule metod z ogromem funkcji, jakie udostępnia R. Warto przy tym nadmienić, że pomimo bardzo szerokiej gamy różnego rodzaju metod analizy dostępnych w środowisku R, zaprezentowana implementacja pokryciowego algorytmu reguł jest jedną z pierwszych dla tej platformy.

Dalszy rozwój pakietu *crules* będzie obejmował z jednej strony doskonalenie oprogramowania od strony technicznej – jak na przykład optymalizację czasową algorytmu indukcji reguł, a z drugiej strony, dodanie nowych funkcjonalności do pakietu, takich jak: algorytmy filtracji [23, 27, 32], metody adaptacyjnego doboru miar do zbioru treningowego [26, 27] czy też rozszerzenie algorytmu o możliwość indukcji reguł z danych regresyjnych [25] i cenzurowanych [31].

BIBLIOGRAFIA

1. Agrawal R., Srikant R.: Fast algorithms for mining association rules, [in:] Bocca J. B., Jarke M., Zaniolo C. (eds.): Proceedings of 20th International Conference on Very Large Data Bases. VLDB, Morgan Kaufmann, 1994, s. 487÷499.
2. Amin T., Chikalov I., Moshkov M., Zielosko B.: Dynamic programming approach to optimization of approximate decision rules. Information Sciences, No. 221, 2013, s. 403÷418.
3. An A., Cercone N.: Rule quality measures for rule induction systems: Description and evaluation. Computational Intelligence, Vol. 17, No. 3, 2001, s. 409÷424.
4. Bazan J., Szczuka M., Wróblewski J.: A new version of rough set exploration system. Rough Sets and Current Trends in Computing, Springer, 2002, s. 397÷404.

5. Bruha I., Tkadlec J.: Rule quality for multiple-rule classifier: Empirical expertise and theoretical methodology. Intelligent Data Analysis, Vol. 7, No. 2, 2003, s. 99÷124.
6. Cohen W. W.: Fast effective rule induction. Proceedings of the Twelfth International Conference on Machine Learning, Morgan Kaufmann, 1995, s. 115÷123.
7. Duch W., Adamczak R., Grabczewski K.: A new methodology of extraction, optimization and application of crisp and fuzzy logical rules. IEEE Transactions on Neural Networks, Vol. 12, No. 2, 2001, s. 277÷306.
8. Eddelbuettel D., François R.: Rcpp: Seamless R and C++ integration. Journal of Statistical Software, Vol. 40, No. 8, 2011, s. 1÷18.
9. Fürnkranz J.: Separate-and-conquer rule learning. Artificial Intelligence Review, Vol. 13, No. 1, 1999, s. 3÷54.
10. Fürnkranz J., Flach P. A.: ROC 'n' rule learning – towards a better understanding of covering algorithms. Machine Learning, Vol. 58, No. 1, 2005, s. 39÷77.
11. Grzymala-Busse J. W.: A new version of the rule induction system LERS. Fundamenta Informaticae, Vol. 31, No. 1, 1997, s. 27÷39.
12. Hilderman R. J., Hamilton H. J.: Knowledge Discovery and Measures of Interest. Kluwer Academic Publishers, Norwell, MA, USA 2001.
13. Hornik K., Buchta C., Zeileis A: Open-source machine learning: R meets Weka. Computational Statistics, Vol. 24, No. 2, 2009, s. 225÷232.
14. Janssen F., Fürnkranz J.: On the quest for optimal rule learning heuristics. Machine Learning, Vol. 78, No. 3, 2010, s. 343÷379.
15. Janssen F., Fürnkranz J.: Heuristic rule-based regression via dynamic reduction to classification, [in:] Walsh T. (ed.): Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-11). 2011, s. 1330÷1335.
16. Kuhn M., Weston S., Keefer C., Coulte N.: Rule-based models, http://rulebasedmodels.r-forge.r-project.org.
17. Michalski R. S.: Discovering classification rules using variable-valued logic system VL. Proceedings of the 3rd international joint conference on Artificial intelligence, Morgan Kaufmann Publishers Inc., 1973, s. 162÷172.
18. Mierswa I., Wurst M., Klinkenberg R., Scholz M., Euler T.: Yale: Rapid prototyping for complex data mining tasks. Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2006, s. 935÷940.
19. Pawlak Z.: Rough sets: theoretical aspects of reasoning about data. Kluwer Academic Publishers, Dordrecht Boston 1991.
20. Quinlan J. R.: C4.5: Programs for Machine Learning, 1st ed. Morgan Kaufmann, San Mateo, CA, USA 1992.

21. Quinlan J. R.: Learning with continuous classes. Proceedings of the 5th Australian joint Conference on Artificial Intelligence, Singapore 1992, s. 343÷348.
22. R Development Core Team.: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria 2011.
23. Sikora M.: Rule quality measures in creation and reduction of data rule models. Lecture Notes in Computer Science, Vol. 4259, Springer, Berlin-Heidelberg 2006, s. 716÷725.
24. Sikora M.: Wybrane metody oceny i przycinania reguł decyzyjnych. Studia Informatica, Vol. 33, No. 3B (108), Gliwice 2012.
25. Sikora M., Skowron A., Wróbel Ł.: Rule quality measure-based induction of unordered sets of regression rules. Lecture Notes in Computer Science, Vol. 7557, Springer, Berlin-Heidelberg 2012, s. 162÷171.
26. Sikora M., Wróbel Ł.: Data-driven adaptive selection of rules quality measures for improving the rules induction algorithm. Lecture Notes in Computer Science, Vol. 6743, Springer, Berlin-Heidelberg 2011, s. 278÷285.
27. Sikora M., Wróbel Ł.: Data-driven adaptive selection of rule quality measures for improving rule induction and filtration algorithms. International Journal of General Systems, 42(6) (w druku, 2013).
28. Stefanowski J.: Algorytmy indukcji reguł decyzyjnych w odkrywaniu wiedzy. Wydawnictwo Politechniki Poznańskiej, Poznań 2001.
29. Theußl S., Zeileis A.: Collaborative Software Development Using R-Forge. The R Journal, Vol. 1, No. 1, 2009, s. 9÷14.
30. Witten I. H., Frank E., Hall M. A.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, Amsterdam 2011.
31. Wróbel Ł.: Tree-based induction of decision list from survival data. Journal of Medical Informatics & Technologies, No. 20, 2012, s. 73÷78.
32. Wróbel Ł., Sikora M., Skowron A.: Algorithms for filtration of unordered sets of regression rules. Lecture Notes in Computer Science, Vol. 7694, Springer, 2012, s. 284÷295.

Wpłynęło do Redakcji 16 stycznia 2013 r.

Abstract

In this paper we present an R package for induction and evaluation of classification (decision) rules. Rule induction is one of the oldest and one of the most popular methods of

knowledge discovery in databases. The implemented rule induction algorithm employs a so-called covering strategy and it is one of the first of such implementations available for R environment. The covering approach consists in learning a rule which covers some part of the training set. Next, the examples covered by the learned rule are removed and the rule learning process starts recursively for the remaining examples. A unique feature of the developed algorithm is the possibility of using different rule quality measures during growing and pruning of rules. Rule quality measures guide the rule learning process and as numerous empirical studies [3, 5, 14, 23, 24, 26, 27] show, they are of great importance for predictive and descriptive abilities of the output set of rules. In the presented software a user can choose from a set of built-in measures (table 2) which were identified in [27] as a set of measures the most effective according to classification and descriptive abilities of the induced rules. Finally, we show several examples which illustrate the usage of the package.

Adresy

Wojciech MALARA: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, malara.wojciech@gmail.com.

Marek SIKORA: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, marek.sikora@polsl.pl; Centrum Elektryfikacji Automatyzacji Górnictwa EMAG, ul. Leopolda 31, 40-189 Katowice, Polska.

Łukasz WRÓBEL: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, lukasz.wrobel@polsl.pl; Autor jest Stypendystą w Projekcie „SWIFT (Stypendia Wspomagające Innowacyjne Forum Technologii)" POKL.08.02.01-24-005/10 współfinansowanym ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.