# Dokumentacja MED 17Z - wybór atrybutów

Artur M. Brodzki, Mateusz Orzoł

31 stycznia 2018

## 1 Wstęp

Zadaniem jest znalezienie na zbiorach danych atrybutów mających największy wpływ na wynik zadania klasyfikacji. Program ma działać w trybie wsadowym, z linii komend i mieć możliwość załadowania każdego z trzech zbiorów danych z repozytorium UCI:

- Adults: `:http://archive.ics.uci.edu/ml/datasets/Adult`
- Flags: `http://archive.ics.uci.edu/ml/datasets/Flags`
- PrimaryTumor: `http://archive.ics.uci.edu/ml/datasets/Primary+Tumor`

Dla zadanego zbioru danych, program ma wypisać wagę każdego atrybutu w zbiorze - im większa waga, tym lepiej używać danego atrybutu do klasyfikacji. Program ma mieć opcję uruchamiania pozwalającą wybrać jeden z dwóch algorytmów działania:

- Algorytm prof. Koronackiego
- Algorytm "Boruta"

Algorytmy te opisuję w sekcji 2

## 2 Opis wykorzystywanych algorytmów

Wybór optymalnego atrybutu w ogólności realizuje się poprzez zbudowanie lasu losowego zawierającego wiele drzew. Cały zbiór danych dzieli się na podzbiory i tworzy osobne drzewo dla każdego podzbioru. Każde drzewo z osobna tworzone jest algorytmem SPRINT, opisanym szczegółowo na notatkach z wykładu o klasyfikacji, w wersji z wykorzystaniem współczynnika Giniego. Z kolei sposób wyznaczania najlepszych atrybutów realizowany jest inaczej w zależności od algorytmu.

## 2.1 Oznaczenia

Oznaczmy zbiór rekordów pewnego zbioru danych przez $D$, a zbiór atrybutów tego zbioru przez $A$. Niech całkowita liczba rekordów w zbiorze danych wynosi $|D|$, a liczba atrybutów w tym zbiorze wynosi $|A|$.

## 2.2 Tworzenie drzewa decyzyjnego losowego algorytmem SPRINT

1. Inicjujemy algorytm ze zbiorem danych $D$ oraz progiem na współczynnik Giniego $gthr$.
2. Losujemy $n = \lfloor \sqrt{|A|} \rfloor$ atrybutów zbioru spośród $A$. Wylosowane atrybuty tworzą podzbiór $A_s$.
3. Dla każdego $a \in A_s$:
   (a) Jeżeli $a$ jest atrybutem typu dyskretnego:
      i. Tworzymy listę $vals(a)$ wszystkich wartości atrybutu $a$.
      ii. Dla każdej kombinacji bez powtórzeń $c = \{a_i, a_j, \ldots, a_m\}$ elementów zbioru $vals(a)$ tworzymy podzbiór $D_c = \{x \in D : x(a) \in c\}$ rekordów o wartości atrybutu $a$ należącej do kombinacji $c$ oraz podzbiór $D_n = \{x \in D : x(a) \notin c\}$ rekordów o wartości atrybutu $a$ nienależącej do kombinacji $c$.
   (b) Jeżeli $a$ jest atrybutem typu numerycznego:
      i. Sortujemy zbiór $D$ według wartości atrybutu $a$ w porządku niemalejącym.
      ii. Kolejne wartości $v$ atrybutu $a$ przyjmujemy jako progi odcięcia i tworzymy podział zbioru $D$ na dwa podzbiory: $D_c = \{x \in D : x(a) \leq v\}$ oraz $D_n = \{x \in D : x(a) > v\}$.
   (c) Dla każdego podziału $\{D_c, D_n\}$ wyznaczamy wartość współczynnika Giniego $gini(D_c, D_n)$ zbioru rekordów po podziale.
   (d) Wybieramy podział $p_{max}(a)$, który minimalizuje współczynnik Giniego na atrybucie $a$.
4. Spośród najlepszych podziałów dla wszystkich atrybutów $\{p_{max}(a_1), p_{max}(a_2), \ldots, p_{max}(a_n)\}$ wybieramy podział globalnie najlepszy $p_{max} = \{D_c, D_n\}$ i zapamiętujemy go w tworzonym węźle drzewa. Zapamiętujemy również warunek podziału.
5. Lewe i prawe poddrzewo tworzymy rekurencyjnie na zbiorach odpowiednio $D_c$ i $D_n$.
6. Rekursja kończy się, gdy współczynnik Giniego na zbiorze danych w węźle jest mniejszy od progu $gthr$ lub gdy w zbiorze pozostał tylko jeden element. Liść jest również tworzony, gdy wszystkie atrybuty decyzyjne

rekordów w zbiorze mają identyczne wartości, tzn. zbiór nie daje się podzielić na podzbiory - nawet, jeśli rekordy należą do różnych klas. Jako klasa zawarta w liściu przyjmowana jest klasa najliczniej występująca w zbiorze danych przypadających na liść. W przypadku, gdy kilka klas jest równolicznych, przyjmowana jest klasa występująca w zbiorze danych jako pierwsza.

## 2.3 Algorytm prof. Koronackiego

1. Podział na podzbiory: decydujemy się na $|A|$ drzew w lesie. Do każdego drzewa przydzielamy $|T| / |A|$ rekordów wybieranych drogą losowania ze zwracaniem (czyli rekordy dla jednego drzewa mogą się powtarzać). Ponieważ rekordy się powtarzają, to niektóre nie zostaną wybrane do żadnego drzewa, za to użyjemy ich do testowania skuteczności nauczonych drzew.

2. Każde drzewo uczymy z wykorzystaniem przydzielonych mu rekordów oraz testujemy za pomocą (tego samego dla każdego drzewa) zbioru rekordów testujących. Zapamiętujemy dokładność (ang. *accuracy*) klasyfikacji drzewa $i : \varphi(i)$.

3. Na każdym z $|A|$ drzew wybieramy jeden z $A$ atrybutów. Wartości tego atrybutu na zbiorze rekordów uczących są losowo permutowane i na tak zmienionym zbiorze ponownie uczymy drzewo i wykonujemy testowanie. Zapamiętujemy nową (niższą, bo zbiór uczący został zmieniony) dokładność klasyfikacji $\varphi'(i)$.

4. Różnica $w(i) = \varphi(i) - \varphi'(i)$ oznacza wagę danego atrybutu.

## 2.4 Algorytm *Boruta*

1. Dokonujemy podziału na podzbiory identycznie jak w algorytmie prof. Koronackiego.

2. Replikacja: wszystkie kolumny-atrybuty w danych sa replikowane. W nowym zbiorze danych mamy więc $2\,|A|$ atrybutów: każdy atrybut $a_i$ ma swoją kopię $a'_i$.

3. Kopie atrybutów mają losowo permutowane wartości.

4. Następnie obliczamy wagę każdego atrybutu $w(i) = \varphi(i) - \varphi'(i)$ (replikowanych i niereplikowanych) tak samo jak w algorytmie Koronackiego.

5. Jako wagę atrybutu $i$ przyjmujemy różnicę pomiędzy wagą Koronackiego jego zwykłej wersji $w(i)$ a wagą jego zreplikowanej wersji $w'(i)$.

# 3 Podział zadań pomiędzy członków zespołu

Artur Brodzki:

1. Implementacja algorytmu SPRINT w wersji dostosowanej do tworzenia lasów losowych
2. Testowanie oraz pisanie dokumentacji końcowej projektu

Mateusz Orzoł:

1. Implementacja algorytmów prof. Koronackiego oraz algorytmu *Boruta*
2. Implementacja modułu parsowania plików z danymi z formacie CSV
3. Interfejs CLI programu - parsowanie parametrów linii komend

# 4 Obsługa programu

Program napisany został jako projekt środowiska Visual Studio 2015. Kod źródłowy programu znajduje się na repozytorium GitHuba pod adresem `https://github.com/ArturB/MED` Do jego skompilowania wymagany jest pakiet Windows SDK w wersji 10.0.14393.0 oraz kompilator MSVC 2015 w wersji 140. Prekompilowany plik wykonywalny MED.exe na architekturę x86-64 znajduje się w podkatalogu x64/Release. Dostępny jest równiez intalator programu w formie archiwum ZIP, zawierający tylko plik wykonywalny oraz przykładowe zbiory danych: można go pobrać z repozytorium GitHuba w zakładce Releases: `https://github.com/ArturB/MED/releases/tag/1.0.2`.

Program uruchamia się, podając kilka parametrów:

```
MED.exe <datafile> <decision-attr> <algorithm>
```

- *datafile*: nazwa pliku z danymi. W podkatalogu x64/Release znajdują się już trzy pliki z danymi:
  - adult.txt
  - flag.txt
  - tumor.txt

  Umożliwia to bezpośrednie testowanie programu na trzech wymienionych w sekcji 1 zbiorach danych.
- *decision-attr*: indeks atrybutu, który określa klasę rekordu. Atrybuty numerowane są od 0. Atrybut określający klasę rekordu musi być atrybutem typu dyskretnego.
- *algorithm*: algorytm tworzenia lasu losowego. Dla $algorithm = 1$ wykonywany jest algorytm prof. Koronackiego, a dla $algorithm = 2$ wykonywany jest algorytm *Boruta*.

| Koronacki | Boruta |
|---|---|
| 25.995 s | 37.659 s |
| 25.505 | 34.928 |
| 23.171 | 42.025 |
| 22.417 | 33.551 |
| 17.539 | 39.937 |
| 21.763 s | 37.819 s |
| 24.898 | 33.199 |
| 21.67 | 29.482 |
| 19.06 | 34.335 |
| 28.045 | 41.076 |

Tabela 1: Przykładowe czasy wykonania programu na zbiorze danych Adults.

W przypadku uruchomienia programu bez parametrów, przyjmowane są domyślne wartości:

```
MED.exe adult.txt 1 1
```

Zatem klasyfikacja wykonywana jest na zbiorze Adult-UCI, z atrybutem *workclass* jako klasą rekordu, metodą prof. Koronackiego.

Wyniki działania programu wypisywane są na standardowe wyjście, można je przekierować do pliku operatorem $>$.

# 5 Testy

## 5.1 Wydajność

Czas działania programu zależy oczywiście od rozmiaru analizowanych danych oraz od wyboru algorytmu tworzenia lasu losowego. Przykładowe czasy wykonania programu na zbiorze Adults-UCI, na komputerze klasy PC z procesorem Intel i5-3570 3,4GHz znajdują się w tabeli 1.

Przeciętny czas wykonania programu wynosi metodą prof. Koronackiego wynosi 23.11 sek, z odchyleniem standardowym 3.15 sek, natomiast metodą *Boruta* wynosi 36.4 sek, z odchyleniem 3.77 sek. Na uwagę zasługuje fakt, że algorytm *Boruta* wykonuje się ok. 1,57 razy dłużej od algorytmu prof. Koronackiego - wynika to niewątpliwie z faktu, że algorytm *Boruta* wymaga skopiowania wszystkich atrybutów na zbiorze danych - poza tym, algorytmy są do siebie bardzo podobne. Co prawda liczba atrybutów z zbiorze danych jest podwajana, ale w algorytmie lasu losowego nie są analizowane wszystkie atrybutu tylko ich część równa $\lfloor\sqrt{|A|}\rfloor$. Dla 15 atrybutów wybieramy więc zawsze 3, a dla 30 - 5. Stosunek tych liczb 1.6 odpowiada niemal dokładnie róznicy wykonania algorytmu Koronackiego i *Boruta* (1.57). Niewielka róznica wynika zapewne z faktu, że oprócz samego tworzenia lasu losowego, w programie występuje pewien czynnik stały, związany z parsowaniem pliku CSV z danymi.

## 5.2 Wyniki

Skuteczność algorytmu SPRINT wypada zadowalająco. Przeprowadzono następujące testy budowanych klasyfikatorów:

- Przewidywanie typu pracodawcy na zbiorze Adult: przeciętny współczynnik skuteczności (ang. *accuracy*) dla stworzonego drzewa wynosił pomiędzy 0.6 a 0.8, choć rzadko pojawiały się też obserwacje odstające ze skutecznością 0.3 - 0.4. Skuteczność klasyfiatora losowego wynosiłaby 0.125.

- Przewidywanie kontynentu państwa na podstawie koloru flagi: skuteczność średnio wyniosła między 0.15 a 0.3, czyli tylko nieco więcej niż losowa 0.17. Wynika to jednak z niewielkiej liczności zbiorów uczących dla drzew: tworząc 30-elementowy las losowy i mając 194 rekordy w zbiorze danych, na jedno drzewo przypada jedynie ok. 5-6 przykładów uczących.

- Przewidywanie typu nowotworu na podstawie jego umiejscowienia: skuteczność średnio wyniosła pomiędzy 0.3 a 0.6, przy 0.05 losowej (są aż 22 typy nowotworów).

Wyniki w każdym przypadku są dość mocno zmienne, co wynika z losowości wprowadzonej do algorytmu SPRINT na potrzeby generowania lasów losowych - randomizowanego doboru analizowanych w każdym węźle atrybutów.

Mniej zadowalająco wypadły testy samych algorytmów prof. Koronackiego i *Boruta*. Niezależnie od zbioru danych oraz niezależnie od atrybutu, uzyskiwane wagi atrybutów są niemal zawsze równe 0. Czasami pojawiają się niewielkie wartości dodatnie - zazwyczaj dla atrybutu determinującego klasę rekordu w danym zbiorze. Biorąc pod uwagę uzyskane wyniki, nie daje się wyciągnąć konkluzywanych wniosków na temat róznic w rezultatach generowanych przez algorytmy Koronackiego i *Boruta*.