Dokumentacja – wersja 1.0
07.06.2021

Podstawy sztucznej inteligencji – projekt 2
Rozwody C4.5

Jakub Zieliński i Laura Wrońska

Spis treści

1. Cel projektu

Celem projektu jest przewidywanie rozwodów za pomocą implementacji algorytmu C4.5 w wersji przedstawionej na wykładzie.

2. Interpretacja treści zadania – przyjęte założenia

Algorytm został zaimplementowany według założeń przedstawionych na wykładzie. Drzewo decyzyjne konstruowane jest najpierw zgodnie z algorytmem ID3, a następnie poddawane „obcinaniu" przez algorytm C4.5.

3. Podział zadań w zespole

- wczytywanie danych – *Laura Wrońska*
- funkcje składowe algorytmu ID3 – *Laura Wrońska*
- główna funkcja algorytmu ID3 – *Jakub Zieliński*
- algorytm C4.5 – *Jakub Zieliński*
- walidacja krzyżowa – *Jakub Zieliński*
- agregacja testów – *Laura Wrońska*
- dokumentacja, opis badań i wnioski – *Laura Wrońska*

4. Wykorzystane narzędzia i biblioteki

Projekt został zrealizowany z użyciem środowiska PyCharm. (8.1)
Kod programu napisaliśmy w języku Python, wersja 3.8. Oprócz standardowych bibliotek, korzystaliśmy też z biblioteki Numpy (8.2) do wczytywania danych oraz z biblioteki Treelib (8.3), służącej do wizualizacji stworzonego drzewa.
Wyniki naszego algorytmu porównaliśmy z istniejącą implementacją drzewa decyzyjnego z biblioteki Scikit-learn (8.4).

5. Uruchomienie testów

Aby przetestować działanie kodu, wystarczy uruchomić plik `validate.py`, w którym znajduje się funkcja prezentująca zagregowane wyniki badań.

6. Badania

6.1 Zbiór danych

Zbiór danych pozyskaliśmy z repozytorium UCI Machine Learning (8.5). Zawiera on 170 rekordów o 54 atrybutach. Istnieją dwie klasy, jedna o wartości 0 i druga o wartości 1.
Po przejrzeniu go stwierdziliśmy, że nie zawiera brakujących wartości, zaś wszystkie atrybuty są liczbami całkowitymi od 0 do 4.

Zbiór dzielimy na część uczącą i testującą, przy czym zbiór testujący stanowi 10% pierwotnego zbioru z repozytorium.
Ze względu na rozkład klas, w którym pierwsze 84 rekordy mają wartość 1, zaś pozostałe 86 rekordów – wartość 0, rekordy są losowo, to jest nie po kolei, przydzielane do poszczególnych podzbiorów.

6.2 Plan testów

W ramach badań postanowiliśmy porównać działanie samego algorytmu ID3 z jego „rozszerzoną” wersją, czyli algorytmem C4.5.

Najpierw prezentowany jest przykładowy kształt drzewa uzyskanego za pomocą algorytmu ID3, a następnie kształt tego samego drzewa po tym, jak zostało one poddane przycinaniu. Jest to wizualizacja działania algorytmu C4.5 jedynie do celów poglądowych. Dla tej wizualizacji do algorytmu przekazujemy cały zbiór (170 rekordów), ponieważ w tym przypadku nie dokonujemy walidacji utworzonych modeli.

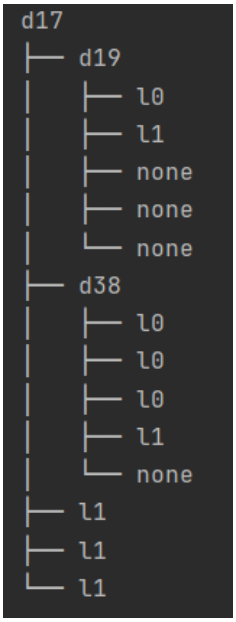
Następnie rozpoczynana jest właściwa część testów – porównanie wyników ID3 i C4.5 z walidacją krzyżową, gdzie liczba podzbiorów wynosi K = 10. W przypadku obu algorytmów, dla każdego z uzyskanych podzbiorów wyliczamy błąd, który pokazuje, jak często przewidywania drzewa decyzyjnego są zgodne z faktycznym wynikiem rekordów ze zbioru testującego. Aby zminimalizować ryzyko zaburzenia wyników przez losowość, procedura powtarzana jest N razy (na potrzeby testów przyjęto N=10). Dodatkowo, dla danego algorytmu z błędów podzbiorów wyliczany jest całkowity średni błąd.

Ponadto, średnie wartości błędów dla naszej implementacji porównujemy z wynikiem uzyskanym przez istniejącą implementacją, dostępną w Internecie.

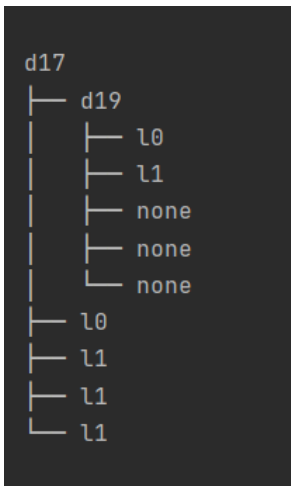
6.3 Wyniki

6.3.1 Wizualizacja drzew

Litera *d* oznacza węzeł decyzyjny, natomiast litera *l* – liść. Dla celów poglądowych węzły będące nullami wyświetlono jako *none*.



ID3



C4.5

6.3.2 Wyniki testów

Nasza implementacja algorytmu:

K=10	**Błędy w kolejnych podzbiorach**										**Średnia błędów**
	1	2	3	4	5	6	7	8	9	10	
ID3	0.047	0.024	0.047	0.012	0.035	0.047	0.047	0.053	0.041	0.024	**0.038**
C4.5	0.041	0.094	0.024	0.024	0.041	0.012	0.047	0.047	0.024	0.047	**0.04**

Średni błąd uzyskany przez gotową implementację algorytmu: 0.035

6.3.3 Wnioski

Dla obu algorytmów błędy mają ten sam rząd wielkości. Wartości błędów są niewielkie, a błędy średnie zawsze wynoszą ok. 0.039, co świadczy o poprawnym przewidywaniu wyników dla naszego zbioru testowego.

Ponadto, średni błąd istniejącej implementacji jest bardzo zbliżony do błędów uzyskanych przez naszą wersję. To również potwierdza poprawność jej działania.

W większości przypadków średni błąd przy algorytmie ID3 jest mniejszy, co może wynikać z faktu, że wówczas drzewo jest mniej ogólne niż dla C4.5, więc nie uwzględnia niektórych przypadków. Z drugiej strony, im bardziej drzewo jest szczegółowe, tym jednocześnie bardziej dopasowane do zbioru uczącego, co nie jest cechą pożądaną.

Patrząc na rozmiar drzewa powstałego w wyniku działania algorytmu, możemy wyciągnąć wniosek, że tak wytrenowany model nie może być dobrym klasyfikatorem dla przypadków nie pochodzących ze zbioru, którym dysponujemy. Według tego modelu wystarczy zadać 1 - 2 “pytania” o wartość konkretnych atrybutów, żeby z dobrym prawdopodobieństwem poprawnie zaklasyfikować dany przypadek. Na pierwszy rzut oka wyniki wydawały się być błędne, jednak po debuggingu zauważyliśmy, że do jednego liścia zostaje przyporządkowane po kilkadziesiąt rekordów uczących (44, 23, 10) dzięki pierwszemu warunkowi kończącemu rekurencje w ID3 (jeżeli zbiór składa się z rekordów tej samej klasy, zwróć liść tej klasy). To wyjaśniło mały rozmiar drzewa.

Trudno zgodzić się z faktem, że o sukcesie związku może decydować tak mało czynników. Gdybyśmy mieli do dyspozycji większy zbiór danych, bogatszy w informacje (tworzący bardziej rozbudowane drzewo), moglibyśmy otrzymać model bardziej dostosowany do klasyfikowania niestandardowych zestawów atrybutów.

7. Czego nauczyliśmy się dzięki projektowi?

Na potrzeby projektu szczegółowo zapoznaliśmy się z implementacją algorytmów ID3 oraz C4.5. Korzystaliśmy wtedy z informacji wyniesionych z wykładów, a także z udostępnionych materiałów do przedmiotu, dzięki czemu ugruntowaliśmy oraz pogłębiliśmy swoją wiedzę. Przekonaliśmy się również, jak ważnym składnikiem algorytmów sztucznej inteligencji jest zbiór danych wykorzystany do trenowania modelu.

Oboje mieliśmy okazję po raz pierwszy implementować większy projekt w Pythonie, dzięki czemu zdobyliśmy podstawowe umiejętności programowania w tym języku. Ponadto, poznaliśmy nieco bibliotekę Treelib, która jest przydatnym narzędziem do wizualizacji stworzonych struktur drzewiastych.

8. Źródła

8.1 PyCharm : https://www.jetbrains.com/pycharm/
8.2 Biblioteka Numpy : https://numpy.org/
8.3 Biblioteka Treelib : https://treelib.readthedocs.io/en/latest/
8.4 Biblioteka Scikit-learn, DecisionTreeClassifier: https://scikit-learn.org/stable/modules/tree.html
8.5 Repozytorium UCI ML, Divorce Predictors : http://archive.ics.uci.edu/ml/datasets/Divorce+Predictors+data+set