Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska

Uczenie się maszyn

Klasyfikacyjny las losowy

Dzmitry Kuksik

Warszawa, 2018

Spis treści

1. Treść zadania

1.1. Zadanie

Zaimplementować zmodyfikowaną wersję algorytmu generowania lasu losowego, w której do generowania kolejnych drzew losowane są częściej elementy ze zbioru uczącego, na których dotychczasowy model się mylił. W eksperymentach należy wykorzystać zadanie German Credit.

1.2. German Credit

Zbiór danych klasyfikuje osoby opisane przez zestaw atrybutów jako dobrych bądź złych kredytorów. Ilość atrybutów wynosi 20. Stosunek osób klasyfikowanych jako dobre do osób klasyfikowanych jako zły wynosi . Dodatkowo jest zdefiniowana funkcja kosztu: lepej klasyfikowac dobrego kredytora jako zły(1), niż złego jako dobry(5).

2. Opis algorytmu

2.1. Dane wejściowe

Zbiór danych został podzielony na zbiór trenujący oraz testowy na długości 900 i 100 odpowiednio. Wynika to z tego, że zbiór danych jest stosunkowo mały, natomiast liczba atrybutów wynosi 20 i każdy atrybut ma co najmniej dwie wartości.

Algorytm uruchamia się poleceniem: **python3 random_forest.py forest_size delta**, gdzie pierwszym argumentem jest liczba drzew decyzyjnych, drugim jest zmiana wagi dla elementów, na których algorytm się pomylił. Naprzykład, **python3 random_forest.py 30 0.0005**.

2.2. Schemat algorytmu

W klasycznym algorytmie lasów losowych każde kolejne drzewo budowane jest w następujący sposób.

1. Jak w algorytmie bagging wylosuj ze zwracaniem z oryginalnej n-elementowej próby uczącej n wektorów obserwacji do pseudopróby uczącej, na podstawie której zostanie zbudowane drzewo.
2. W każdym węźle budowanego drzewa podział podpróby, która dotarła do tego węzła odbywa się następująco: niezależnie od innych losowań wylosuj k spośród p atrybutów wektora obserwacji (jest to zatem losowanie bez zwracania k elementów spośród p elementów); następnie zastosuj przyjętą regułę podziału, jest to minimalizacja indeksu Gini(CART), do wylosowanych k atrybutów (podział jest zatem oparty na wylosowanych k, a nie wszystkich p atrybutach, i wśród tych k atrybutów odbywa się poszukiwanie najlepszego podziału na najlepszym atrybucie). W przypadku danego zadanie $k = 4$.

3. Drzewo jest budowane bez przycinania, jeśli to możliwe aż do otrzymania liści o elementach pseudopróby uczącej z tylko jej klasy.
4. Testowanie drzewa na całym zbiorze trenującym.
5. Modyfikacja wag dla budowania kolejnego drzewa: elementy, na których model drewa się pomylił, mają większe prawdopodobieństwo aby zostać wybranymi do budowania kolejnego drzewa.

2.3. Przetwrazanie zbioru danych

Niżej jest przedstawiony fragment kodu, w którym dokonywane jest wstępne przetwarzanie danych.

```
df = pd.read_csv('data.csv',sep=' ',names=columns,
                 header=None)
numeric2categoric(df,'duration_month',4)
numeric2categoric(df,'credit_amount',4)
numeric2categoric(df,'age',4)

[df_train,df_test] = split_dataframe(df, split_at)
```

2.4. Obliczanie najlepszego atrybutu

W programie zostały zaimplementowane dwie metody obliczania najlepszego atrybutu: Gini indeks oraz information gain, jednak użyta została metoda minimalizacji indeksu Gini.

2.5. Tworzenie lasu losowego

W tym mejscu tworzymy drzewa dla lasu losowego za pomocą metody bootstrapowej, testujemy drzewo na zbiorze trenującym oraz aktualizujemy wagi dla poszczególnych elementów.

```
for _ in range(size):
weights = set_weights(df_train,train_predictions,delta)
df_bootstrap = bagging(df_train,weights)
root_node = decision_tree(df_bootstrap,attr_amount,attributes)
roots.append(root_node)
train_predictions.append(test_tree(df_train,root_node))
```

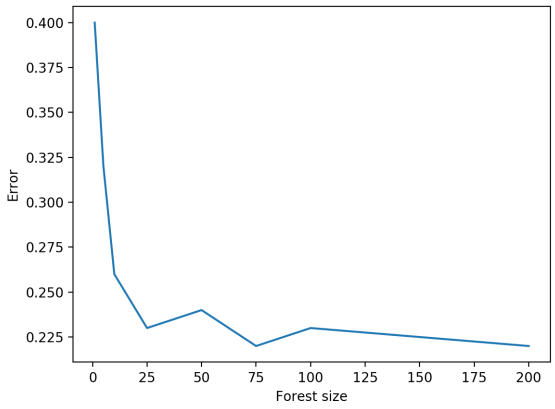
2.6. Testowanie lasu losowego

W tym fragmencie testujemy zbudowany las losowy na zbiorze testowym, robimy głosowanie oraz obliczamy głosy dla każdego elementu, obliczamy koszt oraz błąd klasyfikacji.

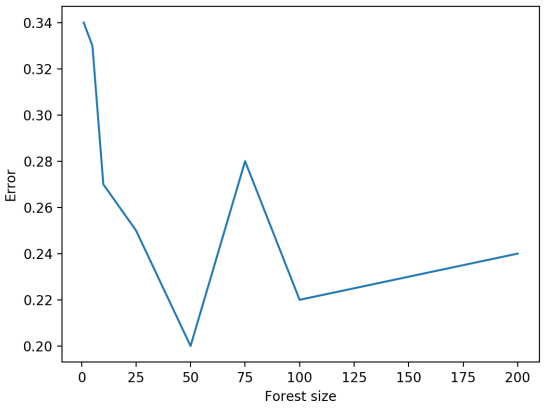
```
for root in roots:
  test_predictions.append(test_tree(df_test,root))

votes_table = ensemble_voting(df_test,test_predictions)
forest_classification = count_votes(votes_table)

for i,fc in enumerate(forest_classification):
  if df_test.iloc[[i]]['decision'].values[0] == fc:
    correct_classification += 1
  elif df_test.iloc[[i]]['decision'].values[0] == 1 and fc == 2:
    cost += 1
  elif df_test.iloc[[i]]['decision'].values[0] == 2 and fc == 1:
    cost += 5
```

3. Wyniki eksperymentów

Na początku kwestia eksperymentów polegała na doborze optymalnej liczby drzew dla lasu losowego. Ponieważ algorytm jest całkowice losowy, program został uruchomiony kilka razy.Niżej są przedstawiony wybrane wykresy zależności błędu klasyfikacji od liczby drzew w lasie.



Rys. 3.1: Błąd klasyfikacji dla różnej liczby drzew



Rys. 3.2: Błąd klasyfikacji dla różnej liczby drzew

Otóż poniżej jest umieszczona tabela dla pierwszego eksperymentu, w której jest porównywane błędy oraz koszty dla różnej liczby drzew.

Liczba drzew	Liczba poprwanie klasyfikowanych	Błąd	Koszt
1	60	0,4	112
5	68	0,32	80
10	74	0,26	102
25	77	0,23	95
50	76	0,24	100
75	78	0,22	102
100	77	0,23	99
200	78	0,22	94

Tab. 3.1: Wartości kosztów i błedów dla różnej ilości drzew

Tabela dla ekperymentu drugiego:

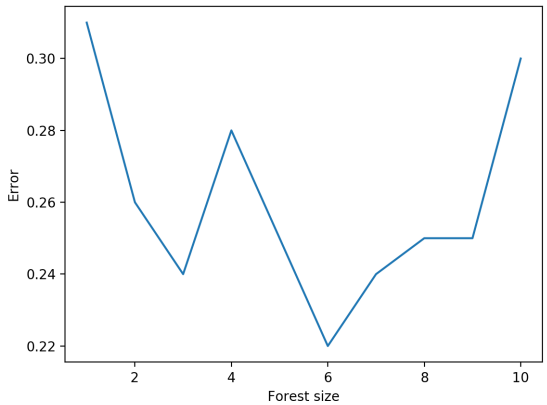
Liczba drzew	Liczba poprwanie klasyfikowanych	Błąd	Koszt
1	66	0,34	102
5	67	0,33	89
10	73	0,27	87
25	75	0,25	105
50	80	0,20	84
75	72	0,28	112
100	78	0,22	98
200	76	0,24	104

Tab. 3.2: Wartości kosztów i błedów dla różnej ilości drzew

Widać, że koszt jest mniejszy im większa jest liczba poprawnie klasyfikowanych elementów. Dla pierwszego eksperymentu najlepsze wyniki pokazały las strorzony z 75 drzew oraz z 200, otóż dla drugiego eksperymentu najlepszy wynik był osiągnięty dla liczby drzew równej 50.

Z zwiększeniem liczby drzew w lasie losowym w pewnym momencie różnica błędów jest nie zbyt istotna, natomiast nakład obliczeń jest mniejszy. Dla kolejnych eksperymentów zostanie wybrana liczba drzew równa 50.

W następnym kroku został przeprowadzony eksperyment dla liczby drzew decyzyjnych w lasie równej 10. Ponieważ algorytm nie jest deterministyczny, eksperyment został uruchamiony 10 razy dla tych samych zbiorów trenujących oraz testowych. Poniżej znajduje się wykres zależności iteracji i błędu klasyfikacji dla każdej iteracji.



Rys. 3.3: Błąd klasyfikacji dla różnej liczby drzew

Tabela porównująca wyniki uzyskane w każdej iteracji.

Liczba drzew	Liczba poprwanie klasyfikowanych	Błąd	Koszt
1	69	0,31	123
2	74	0,26	106
3	76	0,24	104
4	72	0,28	116
5	75	0,25	105
6	78	0,22	98
7	76	0,24	96
8	75	0,25	105
9	75	0,25	109
10	70	0,30	126

Tab. 3.3: Wartości kosztów i błedów dla różnej ilości drzew

Widać, że algorytm w różnych iteracjach wydaje różne wyniki, zależy to wówczas losowości metody bootstrapowej oraz losowości wyboru atrybutów w każdym węzle.

4. Podsumowanie

Z przeprowadzonych eksperymentów można powiedzieć, że las losowy jest dobrym klasyfikatorem. Wynika to z losowości algorytmu i małej korelacji między kolejnie tworzonymi drzewami.Jedną z zalet tej metody jest stosunkowo szybki czas uczenia modelu. Lasy losowe mają szeroki spektr zastosowania, wyżej opisany projekt pokazuje,że klasyfikator radzi sobie z problemami w takiej sferze jak bankowość.