Algorytm lasu losowego (RandomForest)

Wykonali:
Bartłomiej Jankowski
Michał Górski

1. Treść zadania:

Zaimplementować algorytm lasu losowego (ang. Random Forest) do klasyfikacji. Zbiór danych do użycia: MNIST - http://yann.lecun.com/exdb/mnist/. Uzyskane rezultaty porównać z wynikami dla wybranej implementacji algorytmu ML z dostępnych bibliotek np. Scikit-learn, WEKA, MLlib, Tensorflow/Keras etc.

2. Przyjęte założenia, doprecyzowanie treści:

Ze względu na czasochłonność wykonanej przez nas implementacji wykonaliśmy testy dla okrojonego zbioru obiektów poddanych klasyfikacji (5000 próbek treningowych i 1000 testowych). Dane stanowią zdjęcia ręcznie zapisanych cyfr.
Początkowo wczytane dane to lista obrazów, z których każdy to lista liczb (o długości odpowiadającej liczbie pixeli) z zakresu <0,255> oraz lista etykiet z zakresu <0,9>. Przed rozpoczęciem obliczeń łączymy obraz i etykietę tworząc 785-elementowy obiekt (listę) będący reprezentacją cyfry. Zabieg ten jest robiony dla każdego obrazu.

3. Podział odpowiedzialności w zespole:

Michał Górski - architektura oraz implementacja algorytmu
Bartłomiej Jankowski - testy algorytmu oraz porównanie z implementacją z biblioteki (skylearn)

4. Zwięzły opis algorytmu/architektury i uzasadnienie sposobu realizacji:

Algorytm został zaimplementowany w pliku “randomforest.py” znajdują się tam funkcje niezbędne do zbudowania drzewa i na tej podstawie klasyfikacji.
Algorytm przewiduje określenie parametrów z jakimi będzie wykonany:

- n_trees - liczba drzew decyzyjnych jakie mają powstać w lesie losowym
- max_depth - maksymalna głębokość drzewa decyzyjnego liczona od korzenia
- sample_size - współczynnik wielkości próbki uwzględnianej w budowie drzewa
- min_size - minimalna ilość obiektów w drzewie, która spowoduje utworzenie liścia, stanu terminalnego
- n_features - liczba atrybutów brana pod uwagę podczas wyboru optymalnego podziału w węźle według danych z wykładu optymalny jest pierwiastek liczby wszystkich atrybutów. W naszym przypadku podczas wyboru optymalnego podziału uwzględniane jest 28 losowo wybranych atrybutów.
- k_validation - określa parametr k ,k-walidacji krzyżowej

Zasada działania:

1. Program przyjmuje niezbędne parametry, zbiór treningowy i testowy.
2. Budowa lasu losowego z zadaną wcześniej liczbą drzew
    a. Losowanie próbki danych do budowy drzewa.

    b. Wybór najlepszego podziału w korzeniu .
    c. Wywołanie rekurencyjne najlepszych podziałów w kolejnych węzłach jeśli nie spełniają warunków stopu.
3. Predykcja wyników na zbiorze testowym.
4. Obliczenie dokładności.

Drzewo decyzyjne jest słownikiem w którym w kolejnych węzłach znajdują się klucze {‘index’- numer atrybutu według, którego zostanie wykonany podział, ‘value’ - kryterium podziału, ‘left’ - lewy syn, ‘right’ - prawy syn}. Działanie algorytmu opiera się na rekursji.

W każdym węźle następuje optymalny podział ze względu na maksymalizację zysku informacyjnego InfGain. Definiowanego jako różnica Entropii zbioru przed podziałem i po podziale. Aby wybrać optymalny podział dla wylosowanych 28 atrybutów dla każdego obrazka w aktualnym zbiorze liczymy InfGain jeśli podział nastąpiłby w oparciu o dany atrybut i wartość osiąganą dla niego w danym obrazku.(dla wszystkich wylosowanych 28 atrybutów na każdym atrybucie obliczamy ten wskaźnik dla każdego obrazka w danym węźle).

Początkowo wybierany jest optymalny podział w korzeniu. Następnie rekurencyjnie wybierane są optymalne podziały w dzieciach danego węzła (korzenia) i tak dalej przez rekurencję. Wywołania rekurencyjne podziałów są zatrzymywane gdy:

- Osiągnięto maksymalną głębokość,
- Grupa powstała we wcześniejszym podziale jest pusta.
- Liczba obrazów w danej grupie(lewa, prawa) jest mniejsza od minimalnej liczby.

W takich przypadkach tworzony jest stan terminalny - węzeł staje się liściem, do którego przypisana jest cyfra, którą obrazki tej grupy wskazywały najczęściej.

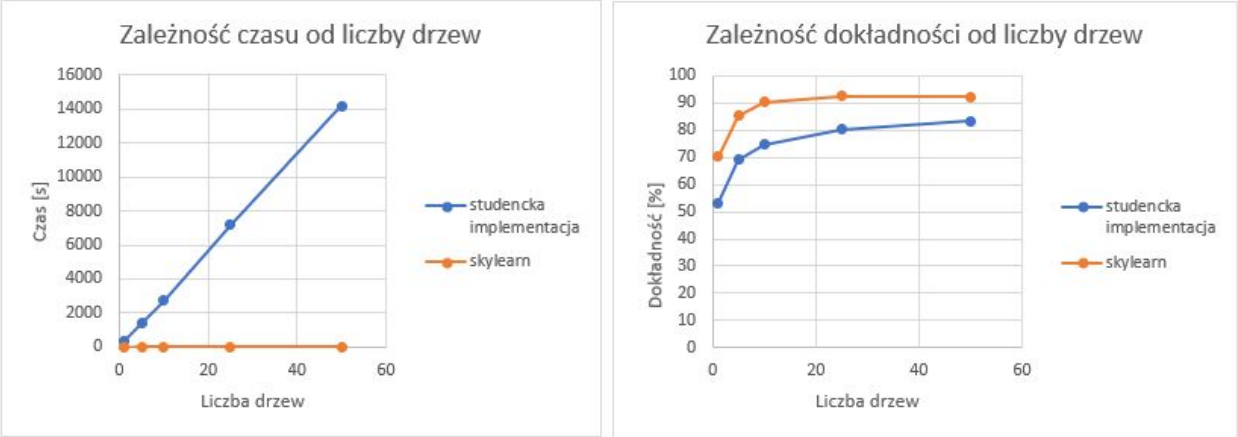
Proces predykcji drzewa decyzyjnego polega na przejściu każdego zdjęcia przez drzewo. Wynikiem predykcji jest cyfra wskazywana przez ostatni odwiedzony liść. W lesie losowym wynikiem predykcji jest najczęściej zwracany wynik predykcji dla drzew wchodzących w skład lasu losowego.

W celu przyspieszenia obliczeń w momencie wyboru optymalnego podziału w przypadku gdy liczba sprawdzanych obrazków jest większa od 255, zamiast iterować się po obrazkach iterujemy się po poziomach jasności.
Dodatkowo badano możliwość wybierania podczas podziału elementu losowego z jednostajnym prawdopodobieństwem. Czas obliczeń na 1000 obrazach i 5 drzewach spadł o ok. 40% natomiast średnia dokładność spadła o ok.8,7% z tego powodu nie zastosowano tego sposobu w dalszych obliczeniach.

5. Testy

Algorytm jest uruchamiany z k-walidacją. Predykcje dla zbioru testowego zostaną oparte o najlepszy model z tego procesu. Pierwsza część testów polega na sprawdzeniu zależności

dokładności oraz czasu obliczeń w zależności od liczby drzew. Druga część na porównaniu tych wyników w przypadku naszej implementacji oraz implementacji z biblioteki skylearn dla tych samych parametrów algorytmu. Obie części testów wykonano równolegle. Wyniki testów przedstawiono na rys. 1.



Rys.1 Wykresy przedstawiające zależność czasu i dokładności od liczby drzew dla studenckiej implementacji i skylearn.

Na podstawie wykresów przedstawionych na rys. 1 można zauważyć, że implementacja dokonana przez nas pozostawia wiele do życzenia pod względem wydajności. Jednak udało się osiągnąć dokładność zbliżoną do implementacji z biblioteki skylearn. Osiągnięta dokładność dla naszej implementacji dla 50 drzew to 83,4%. Dla biblioteki skylearn dokładność to 92%

Jedną z podstawowych obserwacji jest to, iż w przypadku naszej implementacji komputer bardzo długo obliczał parametry drzewa. W przypadku skylearn były to chwile. Dodatkowo zauważono (zgodnie z naszymi oczekiwaniami), że czas obliczeń naszej implementacji jest liniową funkcją liczby drzew. Jednak zasada ta nie obowiązuje w przypadku biblioteki skylearn. Tam czas rośnie w mniejszym tempie niż liczba drzew.

6. Wnioski

Zaimplementowany algorytm działa poprawnie, jednak wolno. Dalsze prace, mające na celu poprawić działanie algorytmu, szukałyby w pierwszej kolejności metod skrócenia czasu obliczeń (możliwe, że poprzez szybszy wybór feature'ów czy implementacje w innym języku programowania). W drugim kroku postaranoby się dopracować użyte metody obliczania błędu, aby zwiększyć precyzję. Możnaby też pomyśleć o dodaniu przycinania drzewa.

7. Uruchomienie

Należy pobrać wszystkie pliki z folderu implementation. otworzyć plik main.py i uruchomić standardowo skrypt. Warto dla skrócenia czasu ograniczyć liczbę danych do klasyfikacji w liniach 58 i 59.