Projekt ZUM

Temat: Tworzenie zespołu modeli klasyfikacji przez wielokrotne stosowanie dowolnego algorytmu klasyfikacji do bootstrapowych prób ze zbioru trenującego z losowo wybranym podzbiorem atrybutów. Porównanie z algorytmami klasyfikacji tworzącymi pojedyncze modele oraz implementacjami algorytmów bagging i random forest dostępnymi w środowisku R lub Python

Dokumentacja

Autorzy: Jakub Szczepanowski, Oleh Hoba
Prowadzący: Stanisław Kozdrowski

1. Interpretacja tematu projektu

Idea modeli zespołowych polega na stworzeniu modelu będącego silnym klasyfikatorem na podstawie wielu składowych klasyfikatorów słabych (modele bazowe). Najbardziej popularnym, ale również skutecznym modelem tego typu jest las losowy. Las losowy to metoda uczenia zespołowego, w którym jak sama nazwa wskazuje stosuje się wiele drzew decyzyjnych do predykcji. Słowo „losowy'' w nazwie wskazuje na zastosowanie losowości w celu otrzymania różnych, słabszych modeli, które razem będą głosować do jakiej klasy należy przyporządkować dany przykład. Metody zespołowe charakteryzują się swobodą w zapewnianiu różnorodności modeli bazowych, dlatego można eksperymentować stosując różne algorytmy, zbiory trenujące itd. W tym projekcie zostaną zastosowane różne modele bazowe, które będą trenowane za pomocą różnych losowych podzbiorów zbioru uczącego powstałych poprzez losowanie próbek ze zwracaniem zwane agregacją. Do tego należy również dołożyć losowanie atrybutów dla każdego podzbioru, żeby osiągnąć jeszcze większą różnorodność. Ta druga opcja powinna zostać dodana jako hiperparametr regularyzacyjny estymatora. Liczba losowanych atrybutów może być różna, ale popularnym wyznacznikiem ilości jest zaokrąglona w dół wartość pierwiastka kwadratowego z liczby atrybutów. Drugą stosowaną techniką jest zlogarytmowanie liczby atrybutów z użyciem podstawy 2.

Niezbędnym z punktu widzenia modeli zespołowych elementem predyktora jest głosowanie. Może odbywać się ono na dwa różne sposoby. Jeżeli modele bazowe zwracają wynik wskazując na konkretną klasę użyte zostanie głosowanie twarde. W tym przypadku klasa, która otrzymała najwięcej głosów wygrywa. Natomiast jeżeli modele składowe są w stanie zwrócić wartość prawdopodobieństwa przynależności do danych klas zostanie użyte tzw. głosowanie miękkie. Dzięki technice głosowania miękkiego można uzyskać lepszą jakość, ponieważ zawiera się w nim miara pewności modelu za wskazaniem jakiejś konkretnej klasy. Samo głosowanie odbywa się poprzez uśrednianie prawdopodobieństw i zwracanie maksymalnego wyniku.

Elementem algorytmu uczenia maszynowego jest osłabianie modeli składowych za pomocą technik zwanych: agregacją bootstrapową, wklejaniem oraz stosowaniem podprzestrzeni losowych. Agregacja - jak to już zostało wspomniane - polega na losowaniu próbek ze zbioru trenującego ze zwracaniem, wklejanie natomiast na próbkowaniu bez zwracania. Zastosowanie podprzestrzeni losowej to po prostu wylosowanie atrybutów, do których ma być zawężone uczenie modelu bazowego.

2. Opis części implementacyjnej

Zakres prac projektowych zawęża się do zaimplementowania klasy modelu zespołowego, wraz z technikami agregacji bootstrapowej, głosowania oraz losowania atrybutów. Obiekty różnych technik klasyfikacji będą pochodziły z gotowej implementacji pochodzącej z pakietu Scikit-Learn. Cały projekt zostanie wykonany w języku Python. Następnym etapem będzie przeprowadzenie eksperymentów porównujących własną implementację z działaniem gotowej klasy stosującej bagging (agregowanie), lasu losowego oraz pojedynczego modelu.

3. Lista algorytmów, które będą wykorzystane w eksperymentach

- Bagging
- Logistic Regression (regresja logistyczna)
- Drzewo decyzyjne
- KNN (k-nearest neighbors)
- SVM (Support Vector Machines)
- Random Forest
- Naive Bayes
- Można też rozważyć wykorzystanie innych algorytmów klasyfikacji

4. Plan badań

W celu porównania skuteczności różnych algorytmów klasyfikacji i tworzenia zespołów modeli klasyfikacji:

1) Przeniesienie danych do preferowanych struktur takich jak ramka danych pakietu Pandas, czy struktura tablicowa pakietu NumPy
2) Wydzielenie etykiet
3) W razie potrzeby zmapowanie danych kategorycznych na dane numeryczne w zależności od ich charakterystyki (nominalne, rangowane)
4) Sprawdzenie, czy dane są kompletne oraz uzupełnienie ewentualnych wartości null wartością najczęstszej kategorii (dane kategoryczne) lub mediany (dane numeryczne)
5) Standaryzacja danych
6) Podział zbiorów danych na zbiór treningowy (80%) i zbiór testowy (20%)
7) Stworzenie pojedynczych modeli klasyfikacji za pomocą algorytmów:
    - Drzewo decyzyjne
    - KNN (k-nearest neighbors)

    - Logistic Regression (regresja logistyczna)
    - Naive Bayes
    - SVM (Support Vector Machines)
8) Stworzenie zespołów modeli klasyfikacji za pomocą algorytmów:
    - Bagging z wykorzystaniem powyższych algorytmów
    - Własnej implementacji z wykorzystaniem powyższych algorytmów
    - Random Forest
9) Porównanie skuteczności pojedynczych modeli klasyfikacji, zespołów modeli klasyfikacji na podstawie metryk:
    - Accuracy (dokładność)
    - Precision (precyzja)
    - Recall (czułość)
    - F1 score
    - macierz pomyłek
    - krzywa ROC wraz z AUC
10) Analiza wyników i wnioski.
11) Opcjonalnie, porównanie czasu trenowania i predykcji każdego algorytmu.

5. Zbiór danych

W celu przeprowadzenia eksperymentów można wykorzystać popularne zbiory danych, takie jak:

- Iris dataset - zbiór danych dotyczący trzech gatunków irysów (Setosa, Versicolor, Virginica), których pąki i płatki zostały zmierzone pod względem długości i szerokości
- Breast Cancer dataset - zbiór danych zawierający informacje o guzach piersi, w tym ich rozmiar, kształt, konsystencję i inne czynniki
- Wine dataset - zbiór danych dotyczący cech wina, takich jak zawartość alkoholu, kwasowość, pH itp., dla trzech różnych gatunków wina
- Pima Indians Diabetes dataset - zbiór danych zawierający dane diagnostyczne w celu wykrywania cukrzycy u pacjenta

6. Opis implementacji

Implementację klasy modelu zespołowego (EnsembleClassifier) wykorzystującą bagging oparto o pola przechowujące obiekt bazowy dowolnego algorytmu pochodzącego z pakietu sklearn, listę indeksów wylosowanych atrybutów (przy randomizacji cech), hiperparametry: ilość estymatorów, flagę randomizacji cech; metodę fit (estymator), metodę predict (predyktor klasy), predict_proba (predyktor prawdopodobieństwa przynależności do klasy) oraz metody służące do głosowania miękkiego i twardego. Estymator przyjmuje macierz przykładów uczących oraz wektor etykiet do każdego z nich. Następnie, dla każdego modelu bazowego losowana jest próba bootstrapowa i w zależności od flagi rand_features losowany jest podzbiór cech równy pierwiastkowi kwadratowemu z liczby dostępnych atrybutów, przy czym zapisywane są indeksy. Po sklonowaniu obiektu bazowego jest on trenowany i zapisywany jako gotowy model bazowy. Poszczególne predyktory w zależności od rodzaju

używają swoich metod głosowania: predict używa głosowania twardego a predict_proba głosowania miękkiego. Dla głosowania twardego dla każdego przykładu wyznaczany jest licznik w postaci słownika oraz tupel zawierający nazwę najbardziej popieranej klasy wraz z liczbą głosów. Przechodząc w pętli przez wszystkie wytrenowane modele bazowe pobierane są predykcje. Jeżeli dokonano randomizacji cech dany przykład jest wycinany do atrybutów zastosowanych w procesie uczenia. Potem licznik dla danej predykcji jest inkrementowany a następnie porównywany z najbardziej popieraną klasą. Po zakończeniu pętli rezultat głosowania zapisywany jest do listy wynikowej. Proces głosowania miękkiego jest o wiele prostszy. Zamiast zliczać poszczególne klasy dla każdego z przykładów inicjalizowana jest zmienna oznaczająca sumę predykowanych prawdopodobieństw. Na zakończenie głosowania liczba ta jest dzielona przez liczbę modeli bazowych uzyskując w ten sposób średnią arytmetyczną ze wszystkich prawdopodobieństw.

Do implementacji klasy modelu zespołowego napisano także kilka funkcji pomocniczych do wykonywania eksperymentów. Są to funkcje tworzące: macierz pomyłek, krzywą ROC, wypisujące metryki dokładności, precyzji, czułości, f1 oraz do wykonania k-krotnego sprawdzianu krzyżowego z użyciem dowolnej metody podziału wraz z wypisywaniem metryk na każdy etap walidacji z ostatecznymi statystykami dla całego procesu oceny (średnia, odchylenie standardowe, wartość minimalna, wartość maksymalna).

7. Opis eksperymentów

Eksperymenty dzielą się na cztery wybrane zbiory. W ramach każdego zbioru przetestowano dla każdego algorytmu działanie: pojedynczego modelu, lasu losowego, własnej implementacji oraz gotowej implementacji modelu zespołowego na bazie danego algorytmu. Na wstępie warto zaznaczyć, że testy miały na celu porównanie działania poszczególnych rozwiązań, a nie uzyskanie jak najlepszych wyników. Tak, aby uzyskiwane wyniki były porównywalne posługiwano się domyślnymi hiperparametrami dla modeli bazowych, natomiast dla zespołów ustalono następujące wartości: liczba estymatorów - 10, liczba randomizowanych cech - pierwiastek kwadratowy z ilości cech. Każda sekcja notatnika Jupyter zawiera testy oparte o 3-krotną kroswalidację oraz porównanie predykowanych prawdopodobieństw między gotową i własną implementacją w oparciu o podział 2:1 na rzecz zbioru treningowego. Przed wykonaniem operacji na zbiorze poddawano go uprzedniej standaryzacji. W przypadku klasyfikacji binarnej oprócz macierzy pomyłek tworzona była również krzywa ROC.

8. Wnioski

Dla trzech pierwszych zbiorów będących częścią standardowej biblioteki Pythona wyniki były całkiem zadowalające. Każda metoda radziła sobie całkiem dobrze osiągając przy tym przyzwoite wyniki. Testy porównujące własną implementację z gotową przyniosły dobry rezultat. Wyniki obu modeli nie różniły się znacząco, a często bywało nawet tak, że własna implementacja ,,przebijała'' wynikami model sklearn'a. Jeśli chodzi o porównanie między pojedynczym modelem a modelem zespołowym wykorzystującym bagging - wyniki również

są porównywalne a czasami nawet gorsze po stronie zespołu. Wynika to z tego, że nie wszystkie algorytmy nadają się do ich osłabiania poprzez podprzestrzenie losowe czy bootstrapowe próby. Największy skok jakości było widać w przypadku drzewa decyzyjnego, gdzie zespół uzyskał duży skok jakości w porównaniu z pojedynczym drzewem. Bagging ma sens tylko w przypadku metod podatnych na przetrenowanie oraz nie skalibrowanych probabilistycznie. Takie niestabilne algorytmy będą osiągały lepsze wyniki w zespole, gdyż bagging poniekąd powstał po to, aby zaradzić tym konkretnym słabościom.

W przypadku ostatniego zbioru wyniki były zdecydowanie gorsze. Pojedyncze modele dawały bliskie sobie wyniki (średnia ~0.77 miary f1), natomiast wyniki bagging'owe były bardzo chaotyczne, a w niektórych przypadkach bliskie losowym. Brak regularyzacji, czy przetwarzania wstępnego (poza standaryzacją) dawał się we znaki już przy pojedynczym modelu, a model zespołowy jeszcze bardziej potęgował te błędy.

Podsumowując, bagging to przydatna metoda, lecz nie uniwersalna. Uzyskuje ona pożądane działanie, gdy opiera się o niestabilne algorytmy (nie bez powodu tak dobrą metodą jest las losowy, gdyż swoje działanie zawdzięcza drzewom decyzyjnym) wymagając przy tym dokładniejszej regularyzacji hiperparametrycznej dla bardziej skomplikowanych zadań, gdyż losowość zawarta w procesie uczenia może pogorszyć ostateczne rozwiązanie.