Wydział Matematyki
i Nauk Informacyjnych
POLITECHNIKA WARSZAWSKA

Automatyczne uczenie maszynowe

Sprawdzenie tunowalności hiperparametrów

Jakub Brzóskowski 313180, Jakub Knyspel 313282

21 listopada 2023

1 Wstęp

Celem eksperymentu jest sprawdzenie tunowalności hiperparametrów wybranych przez nas algorytmów uczenia maszynowego. W tym celu zostały wybrane trzy modele, które następnie były sprawdzane na czterech zbiorach danych. Do określenia tunowalności parametrów użyte zostały dwa podejścia: jedno korzystające z rozkładu jednostajnego, a drugie z techniki Bayesowskiej. Całość została opracowana przy użyciu modułów dostępnych w języku Python.

1.1 Modele uczenia maszynowego

Wybraliśmy trzy różne modele na których będą prowadzone eksperymenty. Każdy z nich jest dostępny w pakiecie *scikit-learn* i posiada zbiór hiperparametrów, które można tunować, aby sprawdzać jakość modelu.

Wybrane modele to:

- Elastic Net
- SVR
- SGD

1.2 Hiperparametry

Każdy z wymienionych wyżej algorytmów uczenia maszynowego charakteryzuje się pewnymi hiperparametrami. Celem tego ćwiczenia jest automatyczna optymalizacja części z nich.

1.2.1 Elastic Net

Elastic Net to model oparty o regresję liniową, z regularyzacją L1 i L2. Charakteryzują go dwa hiperparametry (oczywiście pomijając parametry implementacyjne): l_1 oraz α. Odpowiadają one sile metod regularyzacji - α definiuje siłę obu metod, natomiast l_1 definiuje różnicę sił między metodami.

Testowane wartości parametrów to:

1. l_1:
   0.1, 0.5, 0.9
2. α:
   0.0001, 0.05, 0.1, 0.5, 1.0

W optymalizacji bayesowskiej wartości numeryczne podane w tej sekcji zastąpiliśmy ciągłymi zakresami o rozkładzie jednostajnym, których krańcami są maksimum i minimum z podanych wyżej wartości danego parametru.

1.2.2 SVR

SVR to algorytm regresji bazujący na dopasowywaniu hiperpłaszczyzny do danych treningowych. Charakteryzują go następujące hiperparametry:

1. *kernel*
   Określa rodzaj hiperpłaszczyzny, która jest dopasowywana. Testowane przez nas wartości to:
   (a) *linear* - płaszczyzna liniowa,

(b) *poly* - płaszczyzna wielomianowa (trzeciego stopnia),

(c) *rbf* - płaszczyzna radialna (Radial Basis Function),

(d) *sigmoid* - Płaszczyzna sigmoidalna.

2. *gamma*
Określa lokalność wpływu każdej z wartości treningowych. Nie testowaliśmy bezpośrednio wartości tego parametru - zamiast tego eksperymentowaliśmy z różnymi sposobami automatycznego wyznaczania. Testowane przez nas metody to:

(a) *scale* - $gamma = \frac{1}{|X|}$, gdzie X to zbiór danych treningowych;

(b) *auto* - $gamma = \frac{1}{|X|var(X)}$, gdzie $var(X)$ to wariancja danych treningowych.

1.2.3 SGD

Stohastic Gradient Descent to metoda oparta na regresji liniowej. Charakteryzuje się następującymi hiperparametrami:

1. *loss*
Funkcja straty, którą model minimalizuje. Testowane przez nas wartości to:

(a) *squared_ error* - suma kwadratów błędów,
standardowy błąd metody najmniejszych kwadratów,

(b) *huber* - modyfikacja błędu kwadratowego, która ma na celu zmniejszyć błąd powodowany przez wartości najdalej od dopasowania,

(c) *epsilon_ insensitive* - suma tych błędów, które są większe niż $\epsilon = 0.1$,

(d) *squared_ epsilon_ insensitive* - suma kwadratów tych błędów, które są większe niż $\epsilon = 0.1$.

2. *penalty*
Metoda Rodzaj regularyzacji. Testowane przez nas wartości to *l1* (odpowiadające regularyzacji L1), *l2* (odpowiadające regularyzacji L2), oraz *elasticnet*, będące połączeniem poprzednich metod ($l_1 = 0.15$).

3. *alpha*
Określa siłę regularyzacji. Testowane przez nas wartości to 0.0001 i 0.05.

4. *learning_ rate*
Określa sposób uaktualniania wartości *learning rate* (która wyznacza wielkość zmian wartości parametrów zgodnie z gradientem w każdej iteracji) w miarę uczenia się. Testowane przez nas wartości to:

(a) *constant* - *learning rate* ma stałą wartość (0.01),

(b) *adaptive* - *learning rate* ma stałą wartość tak długo, jak długo wartość funkcji straty zmniejsza się w kolejnych iteracjach; gdy *loss* przestaje się zmniejszać, wartość *learning rate* jest zmniejszana pięciokrotnie.

1.3 Zbiory danych

Wybrane przez nas zbiory danych mają charakter ciągły szukanych danych, co oznacza, że testowane przez nas algorytmy bazują na regresji liniowej. Dodatkowo postanowiliśmy zachować ciągłość

tematyczną i wybrać jedynie dane związane z medycyną.

- cholesterol - zbiór danych do przewidywania poziomu cholesterolu u pacjentów na podstawie ich wieku, płci i podstawowych danych medycznych
- liver-disorders - liczba wypitego alkoholu w zależności od parametrów krwii
- bodyfat - poziom tkanki tłuszczowej w ciele na podstawie miar człowieka jak wzrost, waga, wiek, gęstość itd.
- plasma_retinol - poziom retionolu w osoczu na podstawie diety oraz danych medycznych badanych pacjentów

Dane z wybranych zbiorów zostały poddane odpowiedniemu przygotowaniu, aby umożliwić używaniu ich przy testowaniu modeli. Brakujące dane zostały uzupełnione o średnie wartości w przypadku wartości liczbowych oraz o najczęściej występujące w przypadku danych kategorycznych. Następnie takie dane zostały zakodowane do potrzebnego formatu, tzn. liczby zostały sprowadzone do przedziału $[0, 1]$, a kategoryczne zostały zakodowane przy pomocy techniki *One Hot Encoding*.

2 Tunowalność

2.1 $R(\theta^*)$

Analiza tunowalności algorytmów została przeprowadzona według artykułu 4. Na początku estymowaliśmy doświadczalnie wartość $R(\theta^*) = \arg\min_{\theta\in\Theta} g(R^{(1)}(\theta), ..., R^{(m)}(\theta))$ dla każdego z algorytmów. Za funkcję g przyjęliśmy średnią, a $R^{(j)}(\theta)$ estymowaliśmy poprzez wartość funkcji straty po procesie uczenia modelu. Intuicyjnie $R(\theta^*)$ jest więc średnią po wszystkich zbiorach danych z wartości funkcji straty dla parametrów θ, dla których ta średnia wartość jest najmniejsza. Aby określić minimum z definicji szukanej wartości, sprawdziliśmy 16 losowo wybranych kombinacji parametrów (dla każdego datasetu).

Wyniki prezentują się następująco:

Model	$R(\theta^*)$
Elastic Net	0.102980
SVR	0.110227
SGD	0.114316

2.2 Tunowalność jednostajna

Aby policzyć tunowalność hiperparametrów na poszczególnych datasetach, potrzebowaliśmy minimalnej wartości funkcji straty modelu dla każdego zbioru danych (wartość $R^{(j)}(\theta^{(j)*})$).

Pierwszą metodą znalezienia hiperparametrów, które minimalizowałyby funkcję straty na danym zbiorze danych, było sprawdzenie wszystkich kombinacji parametrów (określonych w 1.2). Użyliśmy w tym celu modułu *GridSearchCV* z biblioteki *scikit-learn*. Następnie, na podstawie znalezionych wartości $R^{(j)}(\theta^{(j)*})$ obliczyliśmy tunowalności $d^{(j)} = R^{(j)}(\theta^*) - R^{(j)}(\theta^{(j)*})$. Wyniki przedstawione zostały poniżej.

Model	**Zbiór danych**	**Tunowalność**	**Średnia tunowalność**

Elastic Net	cholesterol	0.007242	0.004604
	liver-disorders	0.0	
	bodyfat	0.001789	
	plasma_retinol	0.009385	
SVR	cholesterol	0.010287	0.005308
	liver-disorders	0.000756	
	bodyfat	0.002580	
	plasma_retinol	0.007609	
SGD	cholesterol	0.001041	0.009282
	liver-disorders	0.013253	
	bodyfat	0.020677	
	plasma_retinol	0.002158	

2.3 Tunowalność Bayesowska

Drugą metodą znalezienia najlepszych hiperparametrów była optymalizacja bayesowska. W tym celu wybraliśmy narzędzie *BayesSearchCV* z pakietu *scikit-optimize*. Obliczone tunowalności zostały zaprezentowane poniżej:

Model	**Zbiór danych**	**Tunowalność**	**Średnia tunowalność**
Elastic Net	cholesterol	0.006304	0.004589
	liver-disorders	0.0	
	bodyfat	0.001789	
	plasma_retinol	0.010263	
SVR	cholesterol	0.000789	0.002933
	liver-disorders	0.000756	
	bodyfat	0.002580	
	plasma_retinol	0.007609	
SGD	cholesterol	0.001286	0.009675
	liver-disorders	0.013952	
	bodyfat	0.020999	
	plasma_retinol	0.002464	

3 Podsumowanie

Tunowalności algorytmów nie są zbyt duże. Może to być spowodowane wyborem modeli uczenia maszynowego - wybrane przez nas modele są dość proste. Powoduje to, że ich wyniki na zbiorach danych wykorzystanych w projekcie nie są najlepsze, a więc dopasowanie hiperparametrów do konkretnego zbioru nie powoduje dużej poprawy.

4 Bibliografia

- Tunability: Importance of Hyperparameters of Machine Learning Algorithms - https://jmlr.org/papers/volume20/18-444/18-444.pdf
- Dokumentacja *scikit-learn* - https://scikit-learn.org/stable/
- Dokumentacja *scikit-optimize* - https://scikit-optimize.github.io/stable/