Algorytm ROCKET

Warsztaty z technik uczenia maszynowego

Wojciech Klusek, Aleksander Kuś

24 May 2022

1 Wstęp

Celem projektu było zaimplementowanie algorytmu ROCKET (RandOm Convolutional KErnel Transform) w języku R dla szeregów wielu zmiennych przy założeniu, że obserwacje mogą mieć różne długości. Algorytm ROCKET służy do klasyfikacji szeregów czasowych z wykorzystaniem dużych ilości losowych "kerneli", które mają parametry takie jak: "length", "weights, "bias", "dilation" oraz "padding".

2 Przygotowanie danych

Wstępne dane zostały pobrane ze strony timeseriesclassification.com. Do testów użyliśmy zestawów danych BasicMotions, Epilepsy, Handwriting oraz NATOP. Następnie pobrane dane przycięliśmy tak, aby obserwacje miały różne długości zgodnie z następującymi zasadami:

Dla każdej klasy

- 1/3 instancji będzie miała długość [10%,40%] oryginalnej długości
- 1/3 instancji będzie miała długość (40%,70%] oryginalnej długości
- 1/3 instancji będzie miała długość (70%,100%] oryginalnej długości

Skrócone instancje zostały następnie wypełnione średnią arytmetyczną wartości, które nie zostały przycięte dla danej instancji lub zerami.

3 Opis algorytmu ROCKET

Na wstępie generowana jest zadana ilość "kerneli" z odpowiednimi parametrami. Następnie są one aplikowane do danych treningowych oraz testowych. "Kernel" ma następujące parametry:

- "Length" - losowo wybrana liczba ze zbioru { 7, 9, 11 }
- "Weights" - wartości z rozkładu normalnego $X \sim \mathcal{N}(0, 1)$
- "Bias" - wartość z rozkładu jednostajnego ciągłego $\mathcal{U}(0, 1)$
- "Dilation" - jest próbkowana w skali wykładniczej $d = \lfloor 2^x \rfloor$, $x \sim \mathcal{U}(0, A)$, gdzie $A = \log_2 \frac{l_input-1}{l_kernel-1}$
- "Padding" - w momencie generacji "kernela" podejmowana jest losowa decyzja czy "padding" ma zostać użyty w momencie aplikowania "kernela" czy też nie. Bez paddingu "kernele" nie są wyśrodkowane w pierwszych i ostatnich $\lfloor l_{kernel} - 1 \rfloor$ punktów.

Każdy "kernel" (ω) z "Dilation" (d) jest aplikowany do każdego wejściowego szeregu czasowego (X) od pozycji i, według następującego wzoru:

$$X_i * \omega = \sum_{j=0}^{l_{kernel}-1} X_{i+(j \times d)} \times \omega_j.$$

Rocket oblicza dwie zagregowane cechy z każdej mapy cech, tworząc dwie liczby rzeczywiste dla każdego "kernela":

- maksymalna wartość
- odsetek wartości dodatnich (ppv)

4 Uruchomienie programu

Do uruchomienia programu wymagane są następujące biblioteki:

Dla R

- foreign
- reticulate

Dla pakietu reticulate wymagana jest instalacja programu "miniconda". Pakiet ten wykorzystywany jest dla funkcji "array_reshape()", której odpowiednika nie znaleźliśmy w czystym języku R.

Instalacja powyższych zależności:

```
install.packages(c("foreign", "reticulate"))
library("reticulate")
install_miniconda()
```

Dla Pythona

- sktime
- numpy

Z biblioteki sktime wykorzystywana jest klasa RidgeClassifierCV używana do oceniania poprawności algorytmu. Do uruchomienia projektu należy uruchomić plik run.py. Wyniki znajdują się w pliku results_score.txt, natomiast współczynniki cech poszczególnych klas znajdują się w pliku results_coefs.txt.

5 Przeprowadzone testy

Dla każdego wyżej opisanego zbioru danych przycięliśmy zbiory "train" i "test" według schematu opisanego powyżej oraz uruchomiliśmy nasz algorytm. Następnie wyniki przekazaliśmy do klasyfikatora w celu oceny. Proces ten powtórzyliśmy 10 razy z uwagi na brak determinizmu etapu przycinania danych, a otrzymane wyniki uśredniliśmy. Wyniki przedstawiono poniżej.

6 Wyniki

Wyniki zostały wyznaczone dla 100 "kerneli"

BasicMotions

Parametry zbioru danych:

- długość instancji TRAIN: 40
- długość instancji TEST: 40
- Długość szeregów: 100

Numer próby	Wynik dla średniej	Wynik dla zer	Wynik referencyjny
1	0.95	0.975	0.9
2	0.95	0.975	0.9
3	0.95	0.95	0.9
4	1.0	1.0	0.9
5	0.975	0.975	0.9
6	0.975	0.975	0.9
7	0.975	0.975	0.9
8	0.975	0.975	0.9
9	0.95	0.95	0.9
10	0.975	0.975	0.9
Średnia	0.967	0.972	0.9
Odchylenie standardowe	0.016	0.013	0
Średni czas iteracji	14.81s	13.15s	16.37s

Epilepsy

Parametry zbioru danych:

- długość instancji TRAIN: 137
- długość instancji TEST: 138
- Długość szeregów: 206

Numer próby	Wynik dla średniej	Wynik dla zer	Wynik referencyjny
1	0.949	0.913	0.942
2	0.934	0.876	0.942
3	0.927	0.905	0.942
4	0.934	0.934	0.942
5	0.920	0.905	0.942
6	0.927	0.891	0.942
7	0.942	0.913	0.942
8	0.913	0.884	0.942
9	0.942	0.913	0.942
10	0.913	0.913	0.942
Średnia	0.93	0.905	0.942
Odchylenie standardowe	0.0117	0.016	0
Średni czas iteracji	48.82s	44.11s	47.45s

Handwriting

Parametry zbioru danych:

- długość instancji TRAIN: 150
- długość instancji TEST: 850
- Długość szeregów: 152

Numer próby	Wynik dla średniej	Wynik dla zer	Wynik referencyjny
1	0.223	0.224	0.248
2	0.222	0.222	0.248
3	0.232	0.232	0.248
4	0.231	0.229	0.248
5	0.236	0.231	0.248
6	0.217	0.223	0.248
7	0.242	0.241	0.248
8	0.241	0.236	0.248
9	0.227	0.218	0.248
10	0.215	0.214	0.248
Średnia	0.229	0.227	0.248
Odchylenie standardowe	0.0089	0.0078	0
Średni czas iteracji	144.22s	142.09s	144.88s

NATOPS

Parametry zbioru danych:

- długość instancji TRAIN: 180
- długość instancji TEST: 180
- Długość szeregów: 51

Numer próby	Wynik dla średniej	Wynik dla zer	Wynik referencyjny
1	0.816	0.688	0.827
2	0.805	0.677	0.827
3	0.788	0.716	0.827
4	0.816	0.755	0.827
5	0.811	0.744	0.827
6	0.822	0.772	0.827
7	0.800	0.700	0.827
8	0.805	0.794	0.827
9	0.811	0.761	0.827
10	0.827	0.805	0.827
Średnia	0.810	0.741	0.827
Odchylenie standardowe	0.0106	0.0419	0
Średni czas iteracji	55.58s	55.02s	59.41s

7 Wnioski

Z przeprowadzonych eksperymentów wynika, że dokładność algorytmu dla danych nieobciętych jest największa. Obcinanie danych i wypełnianie obciętych wartości zerami daje lepsze rezultaty niż wypełnianie wartością średnią pozostałych elementów. Dla zbioru "Handwriting", dla którego ilość instancji w zbiorze treningowym jest dużo mniejsza od tych w zbiorze testowym, dokładność była najmniejsza. Obcinanie danych ma różny wpływ na dokładność w różnych zbiorach. W zbiorze NATOPS, gdzie długość szeregów była najmniejsza, obcinanie danych miało największy wpływ.

8 Bibliografia

[1] https://github.com/alan-turing-institute/sktime/blob/main/sktime/transformations/panel/rocket/_rocket.py

[2] https://github.com/alan-turing-institute/sktime/blob/main/examples/rocket.ipynb

[3] https://github.com/angus924/rocket