Kacper Pietrzak, 242495
Robert Łaski, 242456

Rok akademicki: 2022/2023
Piątek, 10:45

Sztuczna inteligencja i systemy ekspertowe

Zadanie 2 – Poprawa lokalizacji UWB przy pomocy sieci neuronowych

1. Cel

Naszym celem było zaprojektowanie i zaimplementowanie sieci neuronowej, która pozwoliłaby na korygowanie uzyskanych błędów uzyskanych z systemu pomiarowego.

2. Wprowadzenie

Sieć neuronowa to informatyczny lub matematyczny model inspirowany ludzkim mózgiem oraz neuronami w nim zawartymi. Składa się co najmniej z 3 warstw, w których znajdują się neurony z odpowiednią wagą przetwarzające wartości w odpowiedni sposób. Sieci neuronowych używamy m. in. Do podejmowania decyzji na podstawie wcześniej wyuczonych schematów.

3. Opis sieci neuronowej

Nasza sieć używa interfejsu Keras z biblioteki Tensorflow. Wykorzystaliśmy również optymalizator „Adam”, natomiast za funkcję kosztu przyjęliśmy funkcję średniego błędu kwadratowego. Metodą prób i błędów testowaliśmy różne warianty architektury sieci. Postanowiliśmy wykorzystać 5-warstwową architekturę, przedstawioną poniżej:
Warstwa 1 – 10 neuronów, funkcja aktywacyjna ReLU
Warstwa 2 – 5 neuronów, funkcja aktywacyjna eLU
Warstwa 3 – 2 neurony, funkcja aktywacyjna SeLU

Testowaliśmy również różne ilości epok podczas trenowania. Do wybrania odpowiedniej ilości posłużyła nam klasa EarlyStopping z biblioteki Tensorflow. Nasza sieć została wytrenowana przez 29 epok. Podczas uczenia korzystamy z podziału zestawu treningowego na małe paczki danych po 32.

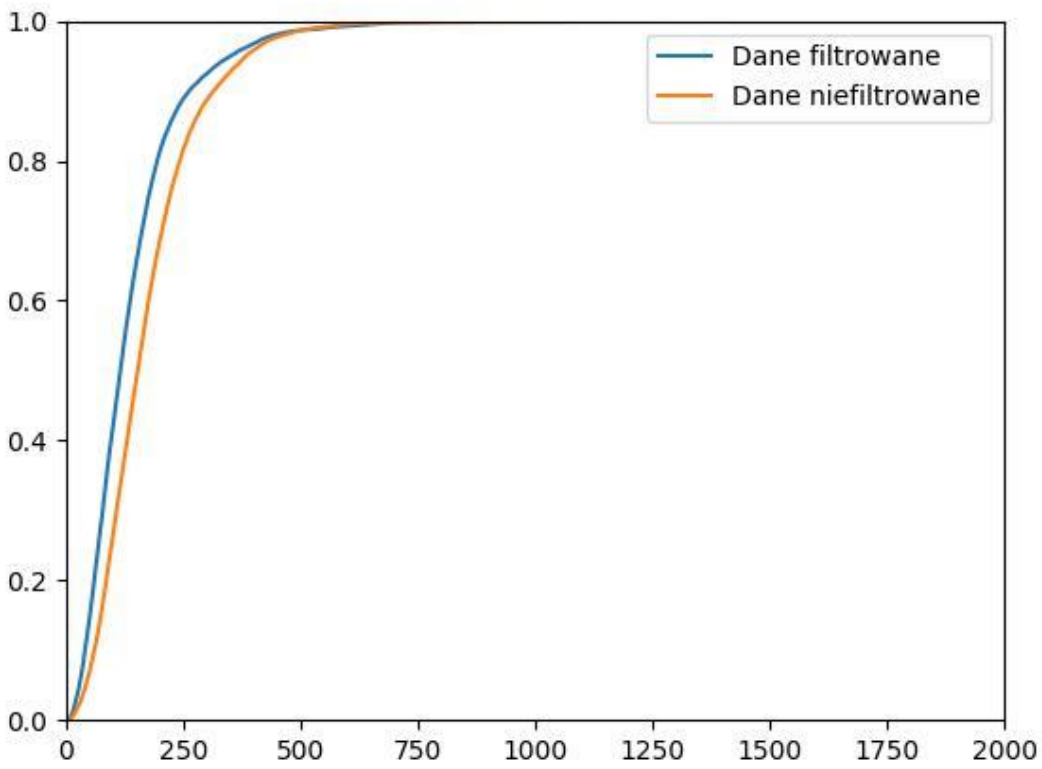
Nasza sieć przyjmuje na wejście wartości z kolumn data__coordinates__x, data__coordinates__y, reference__x oraz reference__x. Natomiast na wyjście przekazuje wagi neuronów, wykres z porównaniem dystrybuant błędów danych oraz plik xlsx z pojedynczą kolumną z wartościami dystrybuanty błędu uzyskanymi w wyniku filtracji przy użyciu sieci neuronowej.

W programie wykorzystaliśmy takie biblioteki jak wcześniej przytoczony tensorflow, glob(import danych), matplotlib(wykresy), math, pandas, numpy, sklearn(obliczenia i operacje).

4. Materiały

Do wytrenowania naszej sieci wykorzystaliśmy materiały zawarte na wikampie na stronie przedmiotu oznaczone jako dane pomiarowe statyczne. Następnie testowaliśmy sieć na danych pomiarowych dynamicznych.

5. Wyniki



Oś x-ów przedstawia odległość od rzeczywistych wartości, natomiast na osi y-ów znajduje się prawdopodobieństwo wystąpienia losowej zmiennej, która ma wartość odległości od rzeczywistej oczekiwanej wartości mniejszą lub równą wartości na osi odciętych.

Wagi neuronów:

Warstwa nr 1

```
[[-0.15275408 -0.26573285  0.27560118 -0.43067655  0.19848105  0.7453822
   0.7100555  -0.19326521 -0.00569332 -0.6255469 ]
 [ 1.768889   -0.08796337  0.5055051  -0.2259671  -0.12356191 -0.01726385
   0.54013705  0.05322075  0.89980394 -0.48255473]]
```

Warstwa nr 2

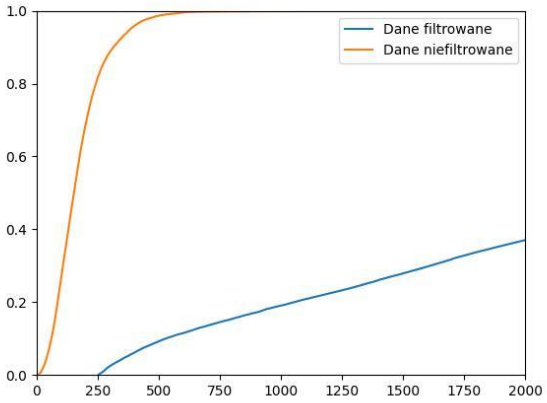
```
[[-0.33209255  0.23428528 -0.71674323  0.9357706  -0.59197736]
 [-0.616803    0.24577269 -0.10849349 -0.39996004  0.35227612]
 [-0.05440882  0.40188903 -0.1390272   0.09812798 -0.19663233]
 [ 0.23344046  0.08012994  0.07997707  0.15439732 -0.30246717]
 [ 0.26775494  0.20811218 -0.07487875 -0.44735697 -0.09690338]
 [ 0.86059004  0.4439413  -0.3720661   0.20393112 -0.16730128]
 [ 0.49613994 -0.43683836  0.54012966 -0.10140957 -0.32103834]
 [ 0.3451623  -0.36026523  0.12631756 -0.4684261  -0.11257467]
 [-0.36029145 -0.92055386 -0.18805644 -1.7660315   0.45624444]
 [ 0.08913024  0.45513776  0.16322033  0.6112008   0.53548795]]
```

Warstwa nr 3

```
[[ 0.8154269   0.13198552]
 [ 0.3175594   0.46331447]
 [-0.95217234 -0.14302832]
 [-0.65251034  0.7576328 ]
 [-0.44812912  0.42091712]]
```

6. Dyskusja

Podczas tworzenia sieci neuronowej testowaliśmy różne funkcje aktywacji. Sprawdziliśmy między innymi funkcję sigmoid, elu, selu, relu, softmax czy tanh. Zdecydowanie najgorzej poradził sobie softmax. Używając takiej samej architektury jak ostatecznie wybrana, ale z funkcjami aktywującymi softmax uzyskaliśmy takie wyniki:



Jasno to pokazuje, że funkcja softmax nie sprawdza się w naszym zadaniu. Pozostałe funkcje dawały zbliżone do siebie rezultaty(zdecydowanie lepsze od funkcji softmax). Uzyskane wyniki różniły się również w zależności od ilości zastosowanych warstw, ilości neuronów, czy liczby danych uczących.
Podczas tworzenia sieci nauczyliśmy się, że należy zwracać uwagę na zjawisko „przetrenowania”. Jest to sytuacja, gdy model jest zbyt dopasowany do zbioru uczącego. Sieć podejmuje wtedy bardzo dobre decyzje w obszarze próby, na której trenowała, ale drobna zmiana podawanych danych daje zupełnie nieprawdziwe rezultaty. Istnieje wiele metod walki z przeuczeniem, my zastosowaliśmy metodę wczesnego zakończenia, której implementacją jest funkcja EarlyStopping.
Analizując wykres przedstawiający dystrybuanty błędów danych testowych oraz uzyskanych w wyniku zastosowania sieci z punktu 5, widzimy, że przefiltrowane dane są dokładniejsze, co sugeruje, że nasza sieć spełnia swoje zadanie.

7. Wnioski

- wpływ na działanie sieci ma wiele czynników m. in. liczba neuronów, warstw, epok czy zastosowanie różnych funkcji aktywujących
- tworząc sieć neuronową musimy uważać na „przetrenowanie”
- nie każda funkcja aktywująca sprawdzi się w każdej sieci
- dobranie odpowiednich parametrów sieci bywa trudne i czasochłonne.

8. Literatura

[1] https://miroslawmamczur.pl/przykladowa-siec-neuronowa-mlp-w-tensorflow/
[2] https://keras.io/guides
[3] Inteligentna analiza danych, Jednokierunkowe wielowarstwowe sieci neuronowe, Bartłomiej Stasiak, 2012
[4] https://pl.wikipedia.org/wiki/Sie%C4%87_neuronowa