

Data oddania: \_\_\_\_\_

Ocena: \_\_\_\_\_

Mateusz Walczak 216911

Konrad Kajszczyk 216790

## Zadanie 2: Sieć neuronowa służąca do korygowania pomiaru systemu lokalizacji

### Wprowadzenie

Celem zadania było zaprojektowanie i zaimplementowanie sieci neuronowej, która pozwoli na korygowanie błędów uzyskanych z systemu pomiarowego. Projektując sieć neuronową należało odpowiednio dobrać [1]:

- liczbę warstw,
- liczebność neuronów w poszczególnych warstwach,
- funkcje aktywacji,
- liczbę próbek z poprzednich chwil czasowych.

### 1. Opis architektury sieci neuronowej

W tym rozdziale rozpoczniemy od opisu działania naszego programu oraz drogi jaką przebyliśmy, aby znaleźć taką sieć neuronową, która pozwoli na skuteczne korygowanie błędów uzyskanych z systemu pomiarowego. Następnie skoncentrujemy się na opisie architektury tej sieci neuronowej, która okazała się najskuteczniejsza.

#### 1.1. Historia wyboru odpowiedniej architektury sieci neuronowej

Nasz program został napisany w języku *Java*, bez wykorzystania wysokopoziomowych bibliotek do tworzenia sieci neuronowych. Program został napisany w taki sposób, aby w zależności od ustawień, tworzyć, inicjować a

następnie przeprowadzać proces nauki dla sieci neuronowych o różnych liczbach neuronów w poszczególnych warstwach a także różnych liczbach próbek z poprzednich chwil czasowych.

Po wielu nieudanych próbach poprawy błędów uzyskanych z systemu pomiarowego, z wykorzystaniem sieci 2-warstwowych ( $n$  neuronów w 1 warstwie i 2 neurony w warstwie wyjściowej), zdecydowano się na implementację 3-warstwowej sieci neuronowej.

Nasza aplikacja buduje 3-warstwową sieć neuronową na podstawie trzech parametrów, które na potrzeby tego sprawozdania będziemy nazywać  $n_1$ ,  $n_2$  oraz  $p$ . Kolejno, stanowią one:

- $n_1$  - liczba neuronów w pierwszej warstwie sieci,
- $n_2$  - liczba neuronów w drugiej warstwie sieci,
- $p$  - liczbę próbek z poprzednich chwil czasowych wykorzystywanych przez sieć neuronową.

W tym miejscu warto wspomnieć o tym, że trzecia warstwa za każdym razem składała się z 2 neuronów, ponieważ nasza sieć musi mieć 2 wyjścia, aby spełniała warunki zadania (powinna zwracać współrzędną x-ową i y-ową dla danej próbki pomiarowej).

Warstwa trzecia (wyjściowa) posiadała identycznościową funkcję aktywacji. Warstwy pierwsza i druga zaś funkcję hiperboliczną (tangens hiperboliczny), określoną wzorem:

$$\operatorname{tgh} x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \quad (1)$$

Wszystkie wagi, dla każdego z neuronów we wszystkich 3 warstwach, inicjalizowano wartością losową, należącą do przedziału  $\langle -1; 1 \rangle$ .

Program uruchamiano dla różnych kombinacji parametrów  $n_1$ ,  $n_2$  oraz  $p$ . W każdej z kombinacji powtarzano eksperymenty kilkakrotnie, w poszukiwaniu najbardziej optymalnego rozwiązania. Wartości osiągnęte przez poszczególne parametry każdorazowo należały do zbiorów liczb całkowitych spełniających warunki, odpowiednio:

$$2 \leq n_1 \leq 20, \quad (2)$$

$$2 \leq n_2 \leq 20, \quad (3)$$

$$1 \leq p \leq 10. \quad (4)$$

Na podstawie tysięcy iteracji programu, przeprowadzonych na przestrzeni kilkudziesięciu godzin, wyciągnięto wnioski dotyczące tego, jakie wartości

parametrów  $n_1$ ,  $n_2$  oraz  $p$  są optymalne dla zadanego problemu. Okazało się, że:

- najbardziej optymalne liczby neuronów zarówno w warstwie pierwszej jak i drugiej to te, należące do przedziału

$$n_1, n_2 \in \langle 6; 9 \rangle. \quad (5)$$

- liczby próbek z poprzednich chwil czasowych, dających najlepsze rezultaty są następujące:

$$p = 3 \vee p = 4 \vee p = 5 \quad (6)$$

W ten sposób bardzo zawęziliśmy różnorodność sieci neuronowych, wykorzystywanych do naszych eksperymentów. W następnym etapie badań, wykorzystywano już tylko takie sieci neuronowe, które stosowały się do powyższych wniosków. Program ponownie uruchomiono wiele razy. Tym razem jednak, regularnie udawało się osiągnąć zamierzony cel - nauczona sieć neuronowa znacząco korygowała błędy uzyskane z systemu pomiarowego.

Architektura sieci neuronowej, z wykorzystaniem której uzyskano najlepsze rezultaty została omówiona w następnym podrozdziale.

## 1.2. Najskuteczniejsza sieć neuronowa - szczegóły architektury

W wielu iteracjach, z wykorzystaniem różnych konfiguracji sieci (spełniających warunki (5) oraz (6)), udawało się nam korygować błędy uzyskane z systemu pomiarowego, a co za tym idzie poprawić dystrybucję błędów pomiarowych, a także zmniejszyć średni błąd pomiaru dla zbioru testowego.

Najlepsze wyniki zarówno dystrybucji jak i średniego błędów pomiarowych uzyskano dla 3-warstwowej sieci neuronowej, której parametry prezentują się następująco:

- liczebność neuronów w poszczególnych warstwach:  $n_1 = 6$  i  $n_2 = 7$ ,
- liczba próbek z poprzednich chwil czasowych  $p = 3$
- funkcje aktywacji: 1 i 2 warstwa - funkcja hiperboliczna, 3 warstwa - funkcja identycznościowa.

Omawiana sieć neuronowa posiada 8 wejść do każdego neuronu - przyjmuje 4 próbki - aktualną oraz 3 poprzednie (każda próbka składa się z dwóch wartości - x-owej i y-owej). Sieć nie posiada biasu, a co za tym idzie w pierwszej warstwie każdy neuron ma dokładnie 8 wag. Na podstawie powyższych rozważań należy zauważyć, że funkcja realizowana przez omawianą sieć jest funkcją 8 zmiennych, zwracającą w wyniku 2 wartości.

Zestaw wag nauczonej sieci neuronowej zestawiono w trzech poniższych tabelach (Skrót "Nn" oznacza neuron).

Waga	Nn 1	Nn 2	Nn 3	Nn 4	Nn 5	Nn 6
1	-0,8345	0,5208	0,5247	0,6965	-0,9008	0,8152
2	0,2971	-0,2796	0,0606	0,8703	-0,1564	0,7358
3	0,2577	0,4535	0,2143	-0,5176	0,2342	0,2574
4	-1,5628	0,5740	-1,2067	0,4588	-0,6185	-1,2189
5	0,3751	0,2277	-0,7225	0,6301	-0,6720	0,5789
6	0,8277	-0,3251	1,0052	-0,2776	-0,3657	0,4929
7	-1,1181	0,1962	0,9500	-0,1792	-1,1461	-0,5759
8	0,7334	-0,6795	-0,2298	-0,3676	0,8913	-0,8044

Tabela 1. Wagi dla neuronów warstwy 1

Waga	Nn 1	Nn 2	Nn 3	Nn 4	Nn 5	Nn 6	Nn 7
1	-0,6660	0,7435	-0,3241	0,4283	0,9674	-0,3338	-0,7312
2	-0,4454	-1,1011	0,4355	0,7745	0,9237	0,1524	0,1921
3	0,4469	-0,0431	0,1553	0,6801	0,5595	0,2300	0,0051
4	0,2453	-0,2336	-0,8740	0,7704	0,9656	0,1251	-0,1752
5	0,9365	-0,2782	-0,7875	-0,0719	0,4287	0,9144	-0,1206
6	1,0438	0,9676	0,4185	-0,4862	-0,5366	0,8669	-0,0850

Tabela 2. Wagi dla neuronów warstwy 2

Waga	Nn 1	Nn 2
1	-0,2437	0,0113
2	-0,5656	-0,1620
3	-0,1151	-0,0018
4	1,2181	0,3675
5	0,1185	0,7748
6	0,7158	-0,5857
7	-0,6539	-0,2363

Tabela 3. Wagi dla neuronów warstwy 3

## 2. Opis algorytmu uczenia sieci neuronowej

Metodą wykorzystywaną do nauki sieci neuronowej w naszym programie jest algorytm wstecznej propagacji błędów. Ogólny wzór na zmianę wag ( $\Delta w$ ), czyli wartość o jaką modyfikowane będą wagi po całej epoce nauki (prezentacji wszystkich danych treningowych - nauczamy w trybie *off-line*) prezentuje się następująco:

$$\Delta w = q \cdot d \cdot z, \quad (7)$$

gdzie  $q$  to współczynnik nauki,  $z$  - sygnał wchodzący do neuronu danym wejściem a  $d$  - pomocniczy współczynnik błędu.

Aby obliczyć pomocniczy współczynnik błędu  $d$  dla ostatniej warstwy (wyjściowej) należy skorzystać z poniższego wzoru:

$$d = f'(s) \cdot (t - e), \quad (8)$$

gdzie  $f'(s)$  stanowi pochodną funkcji aktywacji od wartości wzbudzenia  $s$ ,  $t$  to poprawna odpowiedź dla danego neuronu - ponieważ mamy dwa neurony w warstwie wyjściowej będzie to odpowiednio wartość x-owa lub y-owa punktu treningowego,  $e$  - obliczona odpowiedź dla tego neuronu.

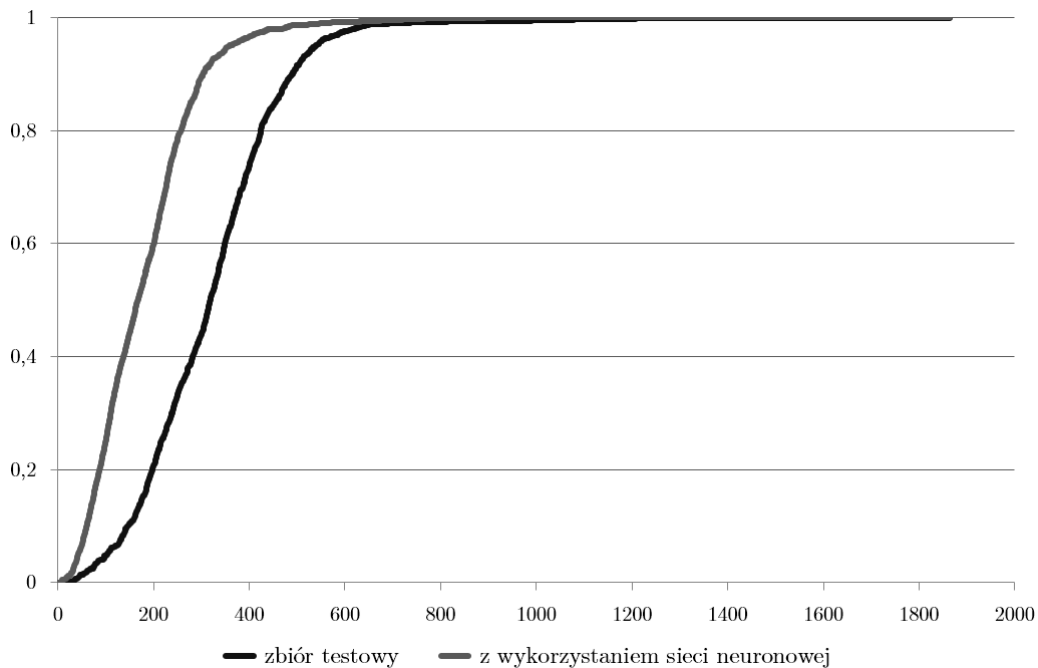
W celu obliczenia współczynnika  $d$  dla pozostałych warstw należy wykorzystać obliczony współczynnik dla warstwy wyjściowej, propagując w ten sposób błąd włąb sieci neuronowej, aż do warstwy pierwszej:

$$d = f'(s) \cdot \sum_{i=1}^n w_i d_i. \quad (9)$$

Na podstawie wzoru (9) łatwo zauważyć, że neuron w warstwie ukrytej dodaje do siebie błędy  $d_i$  z neuronów, z którymi jest połączony. Waga  $w_i$  to waga wejścia neuronu, do której "podłączony" jest neuron, dla którego aktualnie liczymy współczynnik  $d$ .

### 3. Porównanie dystrybuant błędu pomiaru

Na poniższym wykresie porównano dystrybuantę błędu pomiaru dla danych ze zbioru testowego oraz dla danych uzyskanych w wyniku filtracji przy użyciu sieci neuronowej.



Rysunek 1. Porównanie dystrybuant błędów pomiaru

## 4. Kod źródłowy programu

Program został napisany w języku *Java* z wykorzystaniem narzędzia *Maven* [2], służącego do automatyzacji budowy oprogramowania. Import danych oraz generowanie raportów w formacie *xlsx* zaimplementowano z wykorzystaniem biblioteki *poi – ooxml* [3].

Kod źródłowy programu dostępny w repozytorium *GitHub* [4].

## Literatura

- [1] Treść zadania drugiego  
<https://ftims.edu.p.lodz.pl/mod/page/view.php?id=73137>.
- [2] Narzędzie Maven  
<https://maven.apache.org/>.
- [3] Biblioteka poi-ooxml  
<https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml>.
- [4] Kod źródłowy programu  
<https://github.com/Walducha1908/sise2>.