# Sztuczna inteligencja

## Nauka neuronu nieliniowego

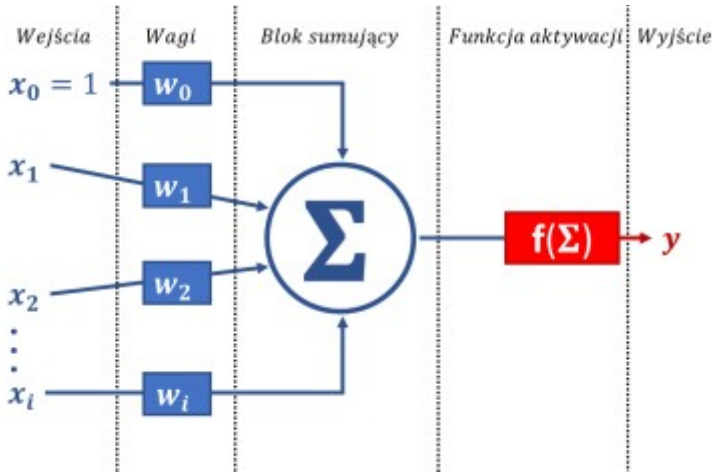
Marcin Wojtas
Nr. albumu: s153915
III rok Niestacjonarnie
Grupa: L3

# 1. Cel

Celem było stworzenie programu opartego o neuron nielinowy który miał za zadanie się uczyć poprzez zwracanie odpowiednich wartości z wykorzystanie operatora OR.

# 2. Wykorzystane pojęcia oraz wzory

Model sztucznego neuronu




Wzór na blok sumujący

$S = x_1w_1 + x_2w_2 + x_3w_3 + ... + x_nw_n$

Wzór na funkcję progową unipolarną

$$f(n) = \begin{cases} 1 & \text{jeśli } n \geq 0 \\ 0 & \text{wpp.} \end{cases}$$

Wzór na otrzymanie nowej wagi

$w_{i+1} = w_i + n * \delta_i * x_i$ , gdzie n to współczynnik uczenia, δ to błąd

# 3. Kod źródłowy programów

```
1   import random
2
3   w1 = random.random() * (1 - (-1)) + (-1)
4   w2 = random.random() * (1 - (-1)) + (-1)
5   w3 = random.random() * (1 - (-1)) + (-1)
```

1. Zaimportowanie biblioteki „random”
2. w1, w2, w3, reprezentują w programie wagi, wylosowanie dla nich różnych wartości z przedziału (-1,1)

```
7   x1 = [0,0,1,1]
8   x2 = [0,1,0,1]
9   x3 = [1,1,1,1]
10  d = [0,1,1,1]
11  b = [0,0,0,0]
12  y = [0,0,0,0]
13  sw = [0,0,0,0]
14  lk = 0
15  wsUcz = 0.70
```

3. Stworzenie zmiennych x1, x2, x3 oraz d i przypisanie im odpowiednich wartości w tablicy.
4. Stworzenie zmiennych
    4.1 b przechowującej określony błąd,
    4.2 zmiennej y przechowującej wyjścia z sztucznego neuronu,
    4.3 zmiennej sw przechowującej bloki sumujące,
    4.4 lk reprezentującej liczbę kroków neuronu uczącego się
    4.5 wsUcz, wskaźnik uczenia się neuronu

```
18  for x in range(4):
19      sw[x] = x1[x] * w1 + x2[x] * w2 + x3[x] * w3
20      if sw[x] >= 0:
21          y[x] = 1
22      else:
23          y[x] = 0
24      b[x] = d[x] - y[x]
```

5. Stworzenie w pętli funkcji progowej unipolarnej przed rozpoczęciem głównej pętli

5.1 sw oblicza blok sumujący za pomocą wzoru na blok sumujący

5.2 jeśli sw >= 0 y przyjmuje wartość 1

5.3 jeśli wartość sw < 0 y przyjmuje wartość 0

5.4 obliczenie błędu po przed odjęcie wartości wyjścia neuronu (y) od planowanego wyjścia (d)

```
print("Przed:")
print("\tBłąd 0: ", b[0])
print("\tBłąd 1: ", b[1])
print("\tBłąd 2: ", b[2])
print("\tBłąd 3: ", b[3])
```

6. Wypisanie błędów przed rozpoczęciem głównej pętli

```
while (b[0] is not 0) or (b[1] is not 0) or (b[2] is not 0) or (b[3] is not 0):
    for x in range(4):
        sw[x] = x1[x] * w1 + x2[x] * w2 + x3[x] * w3
        if sw[x] >= 0:
            y = 1
        else:
            y = 0
        b[x] = d[x] - y

        w1 = w1 + (wsUcz * b[x] * x1[x])
        w2 = w2 + (wsUcz * b[x] * x2[x])
        w3 = w3 + (wsUcz * b[x] * x3[x])
    lk += 1
```

7. Wejście do głównej pętli jeśli którykolwiek z błędów jest inny 0

7.1 sw oblicza blok sumujący za pomocą wzoru na blok sumujący

7.2 jeśli sw >= 0 y przyjmuje wartość 1

7.3 jeśli wartość sw < 0 y przyjmuje wartość 0

7.4 obliczenie błędu po przed odjęcie wartości wyjścia neuronu (y) od planowanego wyjścia (d)

7.5 przypisanej nowej wagi dla w1, w2, w3 poprzez wzór na nowe wagi
7.6 inkrementacja lk mająca na celu sprawdzenie ilości kroków

```
46    print("\n")
47    print("Po:")
48    print("\tBłąd 0: ", b[0])
49    print("\tBłąd 1: ", b[1])
50    print("\tBłąd 2: ", b[2])
51    print("\tBłąd 3: ", b[3])
52    print("")
53    print("Liczba kroków: ", lk)
```

8. Wypisanie błędów po wykonaniu programu
9. Wypisanie liczby kroków po wykonaniu programu

## 4. Przykładowe wykonania programu

1. Pierwszy przykład

```
Przed:
    Błąd 0:  -1
    Błąd 1:  0
    Błąd 2:  0
    Błąd 3:  0
```

```
Po:
    Błąd 0:  0
    Błąd 1:  0
    Błąd 2:  0
    Błąd 3:  0

Liczba kroków:  4
```

2. Drugi przykład

```
Przed:
    Błąd 0:  0
    Błąd 1:  1
    Błąd 2:  1
    Błąd 3:  1
```

```
Po:
    Błąd 0:  0
    Błąd 1:  0
    Błąd 2:  0
    Błąd 3:  0

Liczba kroków:  5
```

# 5. Podsumowanie

Programu pokazuje nam jak neuron nielinowy potrafi się uczyć poprzez podanie mu odpowiednich danych wyjściowych i jak wykorzystuje do tego operatora OR. Dodatkowo potrafi generować nowe wagi przez wzór na nowe wagi aby dążyć do określonych danych wyjściowych.

Pełen kod źródłowy: (python)

```python
import random

w1 = random.random() * (1 - (-1)) + (-1)
w2 = random.random() * (1 - (-1)) + (-1)
w3 = random.random() * (1 - (-1)) + (-1)

x1 = [0,0,1,1]
x2 = [0,1,0,1]
x3 = [1,1,1,1]
d = [0,1,1,1]
b = [0,0,0,0]
y = [0,0,0,0]
sw = [0,0,0,0]
lk = 0
wsUcz = 0.70


for x in range(4):
    sw[x] = x1[x] * w1 + x2[x] * w2 + x3[x] * w3
    if sw[x] >= 0:
        y[x] = 1
    else:
        y[x] = 0
    b[x] = d[x] - y[x]

print("Przed:")
print("\tBłąd 0: ", b[0])
print("\tBłąd 1: ", b[1])
print("\tBłąd 2: ", b[2])
print("\tBłąd 3: ", b[3])

while (b[0] is not 0) or (b[1] is not 0) or (b[2] is not 0) or (b[3] is not 0):
    for x in range(4):
        sw[x] = x1[x] * w1 + x2[x] * w2 + x3[x] * w3
        if sw[x] >= 0:
            y = 1
        else:
            y = 0
        b[x] = d[x] - y

        w1 = w1 + (wsUcz * b[x] * x1[x])
        w2 = w2 + (wsUcz * b[x] * x2[x])
        w3 = w3 + (wsUcz * b[x] * x3[x])
    lk += 1

print("\n")
print("Po:")
print("\tBłąd 0: ", b[0])
print("\tBłąd 1: ", b[1])
print("\tBłąd 2: ", b[2])
print("\tBłąd 3: ", b[3])
print("")
print("Liczba kroków: ", lk)
```

## Bibliografia

https://batmaja.com/sztuczny-neuron/ - wykorzystana grafika modelu neuronu