# Sztuczna inteligencja

## Sieć Hopfielda

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

# 1. Cel

Celem było zaimplementowanie sieci Hopfielda składającej się z jednego wektora posiadającego 64 o rozmiarze 8x8 która miała w sobie reprezentować obrazek bitowy o wartościach 0 lub 1.

# 2. Wykorzystane pojęcia oraz wzory

Wzór na obliczenie wagi:

dla i ≠ j,

$$W_{ij} \sum_{m=1}^{M} (2 * x_i^{(m)} - 1) * (2 * x_j^{(m)} - 1)$$

dla i = j,

0

Wzór na funkcję aktywacji:

$$\varphi(s) = \begin{cases} 1 & s > 0 \\ y(k) & s = 0 \\ 0 & s < 0 \end{cases}$$

gdzie y(k) określa element który został podany przez użytkownika.

# 3. Kod źródłowy programów

```
x = [1, 1, 1, 0, 1, 1, 1, 0,
     0, 0, 1, 0, 1, 0, 0, 1,
     0, 0, 1, 0, 1, 0, 0, 1,
     0, 0, 1, 0, 1, 0, 0, 1,
     0, 0, 1, 0, 1, 0, 0, 1,
     1, 0, 1, 0, 1, 0, 0, 1,
     1, 0, 1, 0, 1, 0, 0, 1,
     1, 1, 1, 0, 1, 1, 1, 0]

u1 = [0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0]

w = [[0 for j in range(64)] for i in range(64)] #wagi
ex = [0 for i in range(64)] #wyjscie
s = [0 for i in range(64)] #suma
```

1. Zadeklarowanie dwóch tablic w tym **x** określającej obrazek bitowy 8x8 o rozmiarze 64 elementów, **u1** reprezentującej podany obrazek przed użytkownika, również 8x8 mający 64 elementy.
2. Zadeklarowanie trzech tablic, **w** przechowująca wagi w tablicy dwuwymiarowej 64x64, **ex** tablica 64 elementowa przechowująca dane wyjściowe oraz **s** w której znajdują się wartości sumy.

```
for i in range(64):
    for j in range(64):
        if i != j:
            w[i][j] = (2*x[i] - 1)*(2*x[j] - 1)
        else:
            w[i][j] = 0

```

3. Zagnieżdżenie pętli w pętli po czym dodatnie do tablicy wag odpowiednich danych poprzez wykorzystaniem wzor na obliczenie wagi

```
30  for i in range(64):
31      for j in range(64):
32          s[i] = s[i] + u1[j] * w[i][j]
```

4. Zagnieżdżenie pętli w pętli po czym dodatnie do tablicy sumy wartości użytkownika po czym przemnożonej przez jeden cały element z tablicy wag.

```
34  for i in range(64):
35      if s[i] > 0:
36          ex[i] = 1
37      elif s[i] == 0:
38          ex[i] = u1[i]
39      else:
40          ex[i] = 0
```

5. Wykorzystanie funkcji aktywacji i przypisanie odpowiednich danych do tablicy ex.

```
42  print("Wektor x:")
43  k = 0
44  for i in x:
45      print(i, " ", end='')
46      k += 1
47      if (k % 8 == 0) & (k != 0):
48          print("")
49
```

```
50  print("")
51  print("Wektor u1:")
52  k = 0
53  for i in u1:
54      print(i, " ", end='')
55      k += 1
56      if (k % 8 == 0) & (k != 0):
57          print("")
```

```
59      print("")
60      print("Wyjście:")
61      k = 0
62      for i in ex:
63          print(i, " ", end='')
64          k += 1
65          if (k % 8 == 0) & (k != 0):
66              print("")
```

6. Wyświetlenie wektora x, wektora u1 i danych wyjściowych.

## 4. Przykładowe wykonania programu

```
Wektor x:
1  1  1  0  1  1  1  0
0  0  1  0  1  0  0  1
0  0  1  0  1  0  0  1
0  0  1  0  1  0  0  1
0  0  1  0  1  0  0  1
1  0  1  0  1  0  0  1
1  0  1  0  1  0  0  1
1  1  1  0  1  1  1  0

Wektor u1:
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0

Wyjście:
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
```

Wejście użytkownika:
same zera

Wejście użytkownika:
nie pokrywające się jedynka

```
Wektor x:
1  1  1  0  1  1  1  0
0  0  1  0  1  0  0  1
0  0  1  0  1  0  0  1
0  0  1  0  1  0  0  1
0  0  1  0  1  0  0  1
1  0  1  0  1  0  0  1
1  0  1  0  1  0  0  1
1  1  1  0  1  1  1  0

Wektor u1:
0  0  0  1  0  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0

Wyjście:
0  0  0  1  0  0  0  1
1  1  0  1  0  1  1  0
1  1  0  1  0  1  1  0
1  1  0  1  0  1  1  0
1  1  0  1  0  1  1  0
0  1  0  1  0  1  1  0
0  1  0  1  0  1  1  0
0  0  0  1  0  0  0  1
```

Wejście użytkownika:
pokrywające się jedynka

```
Wektor x:
1  1  1  0  1  1  1  0
0  0  1  0  1  0  0  1
0  0  1  0  1  0  0  1
0  0  1  0  1  0  0  1
0  0  1  0  1  0  0  1
1  0  1  0  1  0  0  1
1  0  1  0  1  0  0  1
1  1  1  0  1  1  1  0

Wektor u1:
0  0  0  0  1  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0

Wyjście:
1  1  1  0  1  1  1  0
0  0  1  0  1  0  0  1
0  0  1  0  1  0  0  1
0  0  1  0  1  0  0  1
0  0  1  0  1  0  0  1
1  0  1  0  1  0  0  1
1  0  1  0  1  0  0  1
1  1  1  0  1  1  1  0
```

# 5. Podsumowanie

Sieć Hopfielda pozwala na zapamiętywanie wzorów i ich odtwarzania nawet po otrzymaniu tylko jednego poprawnego bitu. Ta sieć również pozwala na otrzymanie negatywu obrazka po podaniu przynajmniej jednego nie poprawnego bitu. W przypadku podania zer lub takiej samej ilości poprawnej bitów jak i nie poprawniej program zwróci wartości jako zera.

Załączniki – cały kod źródłowy (python):

```python
x = [1, 1, 1, 0, 1, 1, 1, 0,
     0, 0, 1, 0, 1, 0, 0, 1,
     0, 0, 1, 0, 1, 0, 0, 1,
     0, 0, 1, 0, 1, 0, 0, 1,
     0, 0, 1, 0, 1, 0, 0, 1,
     1, 0, 1, 0, 1, 0, 0, 1,
     1, 0, 1, 0, 1, 0, 0, 1,
     1, 1, 1, 0, 1, 1, 1, 0]

u1 = [0, 0, 0, 0, 1, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0]

w = [[0 for j in range(64)] for i in range(64)] #wagi
ex = [0 for i in range(64)] #wyjscie
s = [0 for i in range(64)] #suma

for i in range(64):
    for j in range(64):
        if i != j:
            w[i][j] = (2*x[i] - 1)*(2*x[j] - 1)
        else:
            w[i][j] = 0

for i in range(64):
    for j in range(64):
        s[i] = s[i] + u1[j] * w[i][j]

for i in range(64):
    if s[i] > 0:
        ex[i] = 1
    elif s[i] == 0:
        ex[i] = u1[i]
    else:
        ex[i] = 0

print("Wektor x:")
k = 0
for i in x:
    print(i, " ", end='')
    k += 1
    if (k % 8 == 0) & (k != 0):
        print("")

print("")
print("Wektor u1:")
k = 0
for i in u1:
    print(i, " ", end='')
    k += 1
    if (k % 8 == 0) & (k != 0):
        print("")

print("")
print("Wyjście:")
k = 0
for i in ex:
    print(i, " ", end='')
    k += 1
    if (k % 8 == 0) & (k != 0):
        print("")
```