Kaja Kwiatkowska
Inżynieria Obliczeniowa
286114

Podstawy sztucznej inteligencji
Sprawozdanie ćwiczenie 2 - Budowa i działanie sieci jednowarstwowej

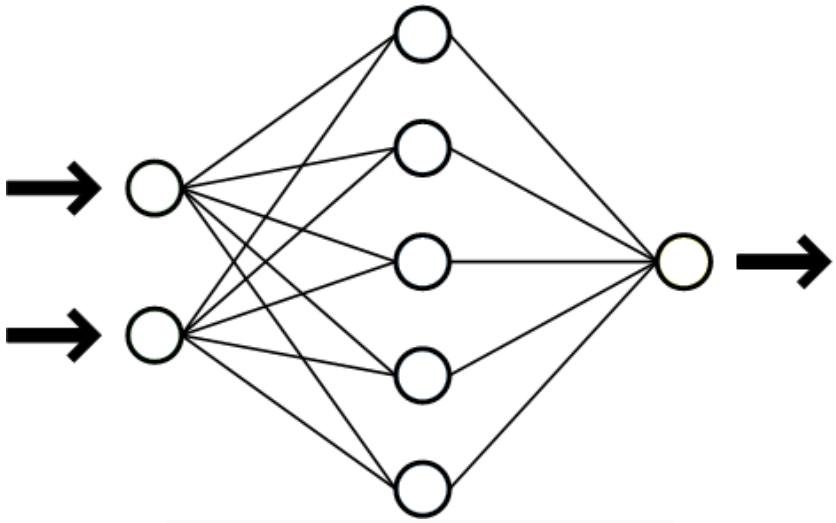
Cel:

Celem ćwiczenia było poznanie budowy i działania jednowarstwowej sieci neuronowej oraz nauczenie jej rozpoznawania wielkości liter.

Wiedza:

Sieć neuronowa jest strukturą matematyczną oraz ich programowymi lub sprzętowymi modelami, realizującymi obliczenia i przetwarzanie sygnałów poprzez rzędy elementów zwanymi sztucznymi neuronami, które wykonują pewną podstawową operację.

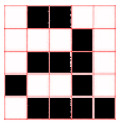
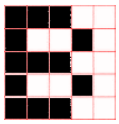
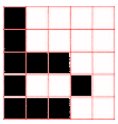
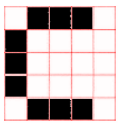
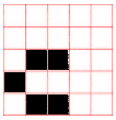
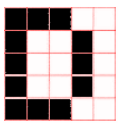
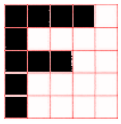
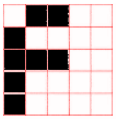
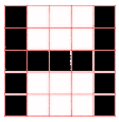
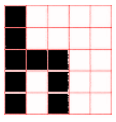
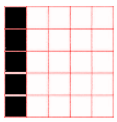
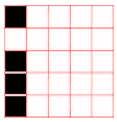
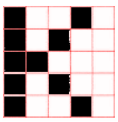
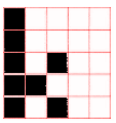
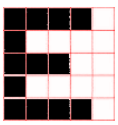
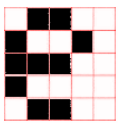
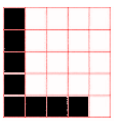
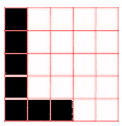
Sieć jednokierunkowa jest siecią neuronową, w której nie występuje sprzężenie zwrotne, czyli pojedynczy sygnał przechodzi przez każdy neuron dokładnie raz. Sieci jednokierunkowe dzielą się na jedno, dwu lub wielowarstwowe. **Sieci jednowarstwowe** mogą rozwiązać jedynie wąską klasę problemów.



Schemat jednokierunkowej sieci neuronowej.
Poszczególne kółka oznaczają sztuczne neurony.

Zadania, które wykonałam w ramach ćwiczenia:

- Utworzyłam dane uczące i testujące, zawierające 10 dużych i 10 małych kolejnych liter alfabetu łacińskiego w postaci tablicy 5x5, z pominięciem liter G i J z powodu trudności z umieszczeniem ich w danym wymiarze tablicy

	0 1 1 1 0 1 0 0 0 1 1 1 1 1 1 1 0 0 0 1 1 0 0 0 1		0 1 1 0 0 0 0 0 1 0 0 1 1 1 0 1 0 0 1 0 0 1 1 1 1
	1 1 1 0 0 1 0 0 1 0 1 1 1 0 0 1 0 0 1 0 1 1 1 0 0		1 0 0 0 0 1 0 0 0 0 1 1 1 0 0 1 0 0 1 0 1 1 1 0 0
	0 1 1 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 1 1 0		0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 1 1 0 0
	1 1 1 0 0 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 1 1 0 0		0 0 0 1 0 0 0 0 1 0 0 1 1 1 0 1 0 0 1 0 0 1 1 1 0
	1 1 1 1 0 1 0 0 0 0 1 1 1 0 0 1 0 0 0 0 1 0 0 0 0		0 1 1 0 0 1 0 0 0 0 1 1 1 0 0 1 0 0 0 0 1 0 0 0 0
	1 0 0 0 1 1 0 0 0 1 1 1 1 1 1 1 0 0 0 1 1 0 0 0 1		1 0 0 0 0 1 0 0 0 0 1 1 1 0 0 1 0 1 0 0 1 0 1 0 0
	1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0		1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0
	1 0 0 1 0 1 0 1 0 0 1 1 0 0 0 1 0 1 0 0 1 0 0 1 0		1 0 0 0 0 1 0 0 0 0 1 0 1 0 0 1 1 0 0 0 1 0 1 0 0
	1 1 1 1 0 1 0 0 0 0 1 1 1 0 0 1 0 0 0 0 1 1 1 1 0		0 1 1 0 0 1 0 0 1 0 1 1 1 0 0 1 0 0 0 0 0 1 1 0 0
	1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 1 1 1 0		1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 1 1 0 0

Tablice składające się z 25 pól, czarnych - 1 lub białych - 0.
Czarne pole oznacza, że w danym miejscu występuje element litery.

- Zaimplementowałam jednowarstwową sieć dwoma metodami:
  - **metoda 1**
    Za pomocą funkcji ***newlin()***, która tworzy jednowarstwową sieć neuronową, która przyjmuje za argumenty: wartości minimalne i maksymalne elementów wejściowych dla funkcji tworzących sieć neuronową (min – 0, maks – 1) oraz ilość elementów na wyjściu sieci.
  - **metoda 2**
    Za pomocą funkcji ***newp()***, która tworzy prostą sieć neuronową, funkcja pobiera dane wejściowe, a następnie zwraca perceptron. Przyjmuje argumenty taka jak w przypadku funkcji *newlin()* oraz może dodatkowo przyjmować argument funkcji transferu (domyślnie przyjmuje *hardlim*) oraz funkcji uczenia (domyślnie *lernp*)

- Przeprowadziłam proces uczenia dla różnych wartości parametrów oraz przetestowałam działanie sieci

Błąd średniokwadratowy	0.001					
Współczynnik uczenia	0.001		0.01		0.1	
Litera\Funkcja	newlin	newp	newlin	newp	newlin	newp
A	0.9948	1	0.9948	1	0.9948	1
a	0.0044	0	0.0044	0	0.0044	0
B	1.0075	1	1.0075	1	1.0075	1
b	-0.0078	0	-0.0078	0	-0.0078	0
C	1.0032	1	1.0032	1	1.0032	1
c	-0.0124	0	-0.0124	0	-0.0124	0
D	0.9882	1	0.9882	1	0.9882	1
d	0.0075	0	0.0075	0	0.0075	0
E	0.9971	1	0.9971	1	0.9971	1
e	-0.0061	0	-0.0061	0	-0.0061	0
liczba epok	9741	8	9741	8	9741	8

Błąd średniokwadratowy	0.01					
Współczynnik uczenia	0.001		0.01		0.1	
Litera\Funkcja	newlin	newp	newlin	newp	newlin	newp
A	0.9510	1	0.9510	1	0.9510	1
a	0.0628	0	0.0628	0	0.0628	0
B	0.9369	1	0.9369	1	0.9369	1
b	0.0135	0	0.0135	0	0.0135	0
C	0.9878	1	0.9878	1	0.9878	1
c	-0.1745	0	-0.1745	0	-0.1745	0
D	0.9162	1	0.9162	1	0.9162	1
d	0.0588	0	0.0588	0	0.0588	0

E	1.0107	1	1.0107	1	1.0107	1
e	0.0300	0	0.0300	0	0.0300	0
liczba epok	1792	8	1792	8	1792	8

błąd średniokwadratowy	0.1					
współczynnik uczenia	0.001		0.01		0.1	
Litera\Funkcja	newlin	newp	newlin	newp	newlin	newp
A	0.9642	1	0.9642	1	0.9642	1
a	0.2351	0	0.2351	0	0.2351	0
B	0.5261	1	0.5261	1	0.5261	1
b	0.3152	0	0.3152	0	0.3152	0
C	0.7694	1	0.7694	1	0.7694	1
c	-0.1685	0	-0.1685	0	-0.1685	0
D	0.7457	1	0.7457	1	0.7457	1
d	0.1461	0	0.1461	0	0.1461	0
E	0.8434	1	0.8434	1	0.8434	1
e	0.3013	0	0.3013	0	0.3013	0
liczba epok	13	4	13	4	13	4

W tabelach powyżej umieściłam dokładne wartości zmiennej *efekt* podczas symulacji sieci dla pierwszych pięciu liter dla obu funkcji tworzenia sieci (*newlin(), newp()*) oraz różnych wartości błędu kwadratowego i współczynnika uczenia sieci.

Wnioski:

Na podstawie tabel z wartościami zmiennej efekt, zaobserwowałam, że dla danego błędu średniokwadratowego wartości współczynnika uczenia nie zmieniają efektu uczenia oraz liczba epok jest jednakowa.
Natomiast obie funkcje zależne są od błędu średniokwadratowego. Testowanie pokazało, że im mniejsza jego wartość tym większa liczba epok jest potrzebna do poprawnego uczenia sieci. Im większa wartość błędu średniokwadratowego tym wyniki stają się mniej dokładne.

Porównując obie użyte funkcje, *newp()* daje lepsze efekty niż *newlin()*, pokazuję to potrzebna do uczenia liczba epok i długość trwania procesu. Dodatkowo stosując funkcję *newp()* ograniczamy ilość błędu, ponieważ efekt uczenia daje dokładne wartości 1 lub 0, w przypadku stosowania funkcji *newlin()* konieczne było zaokrąglenie wartości.
Niestety obie funkcje nie są w stanie rozpoznać liter spoza danych uczących.

Listing kodu programu wraz z komentarzami:

```
close all; clear all; clc;

%zmienna wejściowa dla funkcji tworzących sieć neuronową
%dla tablicy pól obrazujących literę 5x5
%25 par wejść z dwiema możliwymi wartościami (minimalna - 0, maksymalna -1)
wartosci=[0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1;
    0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1;];

%ilość wyjść z sieci = 1
S=1;

%metoda 1.
%net = newlin(wartosci,S);
%metoda 2.
net = newp(wartosci,S);

%kolumnowa reprezentacja binarna 10 liter alfabetu dla tablicy 5x5 - dane uczące
         %A a B b C c D d E e F f H h I i K k L l
wejscie=[0 0 1 1 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1;
         1 1 1 0 1 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0;
         1 1 1 0 1 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0;
         1 0 0 0 1 0 0 1 1 0 1 0 0 0 0 0 1 0 0 0;
         0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0;%
         1 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 1 1 1 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 1 0 0 0;
         0 1 1 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0;
         1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0;%
         1 0 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1;
         1 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 1 0 0 0;
         1 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 1 0 0;
         1 1 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0;
         1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0;%
         1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1;
         0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0;
         0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0;
         0 1 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0;
         1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0;%
         1 0 1 1 0 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1;
         0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1;
         0 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0 1 1;
         0 1 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 1 0;
         1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0;];%

%wyjscie - zmienna przechowująca dane wyjściowe odpowiadające danym uczącym
%1 - duża litera, 0 - mała litera
        %A a B b C c D d E e F f H h I i K k L l
wyjscie=[1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0];

net.name='Rozpoznawanie wielkości liter';

%określenie parametrów treningu sieci
net.trainParam.epochs = 10000;      %maksymalna ilość epok
net.trainParam.goal = 0.0001;       %błąd średniokwadratowy
net.trainParam.mu = 0.001;          %współczynnik uczenia

%trening sieci - proces uczenia za pomocą danych wejściowych i wyjściowych
%net - zmienna, do której będzie przypisywana nowa sieć neuronowa
net = train(net, wejscie, wyjscie);

%dane testowe - litery
testA = [0; 1; 1; 1; 0;
        1; 0; 0; 0; 1;
        1; 1; 1; 1; 1;
        1; 0; 0; 0; 1;
        1; 0; 0; 0; 1];
```

```
testa= [0; 1; 1; 0; 0;
        0; 0; 0; 1; 0;
        0; 1; 1; 1; 0;
        1; 0; 0; 1; 0;
        0; 1; 1; 1; 1];
testB= [1; 1; 1; 0; 0;
        1; 0; 0; 1; 0;
        1; 1; 1; 0; 0;
        1; 0; 0; 1; 0;
        1; 1; 1; 0; 0];
testb= [1; 0; 0; 0; 0;
        1; 0; 0; 0; 0;
        1; 1; 1; 0; 0;
        1; 0; 0; 1; 0;
        1; 1; 1; 0; 0];
testC= [0; 1; 1; 1; 0;
        1; 0; 0; 0; 0;
        1; 0; 0; 0; 0;
        1; 0; 0; 0; 0;
        0; 1; 1; 1; 0];
testc= [0; 0; 0; 0; 0;
        0; 0; 0; 0; 0;
        0; 1; 1; 0; 0;
        1; 0; 0; 0; 0;
        0; 1; 1; 0; 0];
testD= [1; 1; 1; 0; 0;
        1; 0; 0; 1; 0;
        1; 0; 0; 1; 0;
        1; 0; 0; 1; 0;
        1; 1; 1; 0; 0];
testd= [0; 0; 0; 1; 0;
        0; 0; 0; 1; 0;
        0; 1; 1; 1; 0;
        1; 0; 0; 1; 0;
        0; 1; 1; 1; 0];
testE= [1; 1; 1; 1; 0;
        1; 0; 0; 0; 0;
        1; 1; 1; 0; 0;
        1; 0; 0; 0; 0;
        1; 1; 1; 1; 0];
teste= [0; 1; 1; 0; 0;
        1; 0; 0; 1; 0;
        1; 1; 1; 0; 0;
        1; 0; 0; 0; 0;
        0; 1; 1; 0; 0];
testF= [1; 1; 1; 1; 0;
        1; 0; 0; 0; 0;
        1; 1; 1; 0; 0;
        1; 0; 0; 0; 0;
        1; 0; 0; 0; 0];
testf= [0; 1; 1; 0; 0;
        1; 0; 0; 0; 0;
        1; 1; 1; 0; 0;
        1; 0; 0; 0; 0;
        1; 0; 0; 0; 0];
testH= [1; 0; 0; 0; 1;
        1; 0; 0; 0; 1;
        1; 1; 1; 1; 1;
        1; 0; 0; 0; 1;
        1; 0; 0; 0; 1];
testh= [1; 0; 0; 0; 0;
        1; 0; 0; 0; 0;
        1; 1; 1; 0; 0;
        1; 0; 1; 0; 0;
        1; 0; 1; 0; 0];
testI= [1; 0; 0; 0; 0;
        1; 0; 0; 0; 0;
        1; 0; 0; 0; 0;
```

```
        1; 0; 0; 0; 0;
        1; 0; 0; 0; 0];
testi= [1; 0; 0; 0; 0;
        0; 0; 0; 0; 0;
        1; 0; 0; 0; 0;
        1; 0; 0; 0; 0;
        1; 0; 0; 0; 0];
testK= [1; 0; 0; 1; 0;
        1; 0; 1; 0; 0;
        1; 1; 0; 0; 0;
        1; 0; 1; 0; 0;
        1; 0; 0; 1; 0];
testk= [1; 0; 0; 0; 0;
        1; 0; 0; 0; 0;
        1; 0; 1; 0; 0;
        1; 1; 0; 0; 0;
        1; 0; 1; 0; 0];
testL= [1; 0; 0; 0; 0;
        1; 0; 0; 0; 0;
        1; 0; 0; 0; 0;
        1; 0; 0; 0; 0;
        1; 1; 1; 1; 0];
testl= [1; 0; 0; 0; 0;
        1; 0; 0; 0; 0;
        1; 0; 0; 0; 0;
        1; 0; 0; 0; 0;
        1; 1; 1; 0; 0];

%testowanie działania sieci - symulacja
efekt=sim(net, teste);

%wypisanie efektu uczenia sieci
if round(efekt) == 0
    disp('Mała litera');
else
    disp('Duża litera');
end
```