

Uniwersytet Zielonogórski	Wykonali:	Grupa:	Nr ćwiczenia: 3	Ocena:
Laboratorium				
Temat ćwiczenia: Sieci wielowarstwowe i algorytm wstecznej propagacji błędów.		Prowadzący:	Data wyk. ćw.	Data odd. spr.

1. Dokonać uczenia sieci dwuwarstwowej.

a) wybrać dla obu warstw neurony z sigmoidalną funkcją aktywacji

b) wybrać dla warstwy 1 sigmoidalną funkcję aktywacji a dla warstwy 2 liniową

c) wybrać dla warstwy 1 tangensoidalną funkcję aktywacji a dla warstwy 2 liniową

Uczenie przeprowadzić dla następujących danych: $P=[1\ 1\ 0\ 0; 1\ 0\ 1\ 0]$; $T=[0\ 1\ 1\ 0]$; (problem XOR) oraz dla: $P=[-0.5\ -0.5\ 0.3\ 0.1\ -0.3; -0.5\ 0.5\ -0.5\ 1\ 0]$; $T=[1\ 1\ 0\ 0\ 0]$;

Do realizacji zadania wykorzystaliśmy skrypt programu Matlab. Poniżej przedstawione są skrypty oraz otrzymane wyniki:

a)

```
P=[1 1 0 0 ;1 0 1 0];
T=[0 1 1 0];
net=newff([0 1;0 1],[2 1],{'logsig','logsig'});
net=train(net,P,T)
A=sim(net,P)
E=A-T
```

Otrzymane wyniki:

```
A = 0.0000 1.0000 1.0000 0.0000
E = 1.0e-005 * [ 0.3829 -0.3387 -0.5641 0.3984] (E - wartość błędu)
```

```
P=[-0.5 -0.5 0.3 0.1 -0.3;-0.5 0.5 -0.5 1 0];
T=[1 1 0 0 0];
net=newff([0 1;0 1],[2 1],{'logsig','logsig'});
net=train(net,P,T)
A=sim(net,P)
E=A-T
```

Otrzymane wyniki:

```
A = 1.0000 1.0000 0.0000 0.0000 0.0000
E = 1.0e-005 * [-0.1355 -0.3052 0.2254 0.4565 0.3965] (E - wartość błędu)
```

b)

```
P=[1 1 0 0 ;1 0 1 0];
T=[0 1 1 0];
net=newff([0 1;0 1],[2 1],{'logsig','purelin'});
net=train(net,P,T)
A=sim(net,P)
E=A-T
```

Otrzymane wyniki:

```
A = 0.0000 1.0000 1.0000 -0.0000
E = 1.0e-011 * [0.9396 -0.6516 -0.0173 -0.2631] (E - wartość błędu)
```

```
P=[-0.5 -0.5 0.3 0.1 -0.3;-0.5 0.5 -0.5 1 0];
```

```
T=[1 1 0 0 0];
net=newff([0 1;0 1],[2 1],{'logsig','purelin'});
net=train(net,P,T)
A=sim(net,P)
E=A-T
```

Otrzymane wyniki:

```
A = 1.0000    1.0000   -0.0000    0.0000    0.0000
E = 1.0e-006 *[-0.0131   -0.0131   -0.1302    0.0000    0.1302] (E - wartość błędu)
```

C)

```
P=[1 1 0 0 ;1 0 1 0];
T=[0 1 1 0];
net=newff([0 1;0 1],[2 1],{'tansig','purelin'});
net=train(net,P,T)
A=sim(net,P)
E=A-T
```

Otrzymane wyniki:

```
A = 0.0000    1.0000    1.0000   -0.0000
E = 1.0e-014 * [0.3331   -0.0222         0   -0.3109] (E - wartość błędu)
```

```
P=[-0.5 -0.5 0.3 0.1 -0.3;-0.5 0.5 -0.5 1 0];
T=[1 1 0 0 0];
net=newff([0 1;0 1],[2 1],{'tansig','purelin'});
net=train(net,P,T)
A=sim(net,P)
E=A-T
```

Otrzymane wyniki:

```
A = 1.0000    1.0000   -0.0000   -0.0000    0.0000
E = 1.0e-006 *[-0.0010   -0.0010   -0.0550   -0.0572    0.1123] (E - wartość błędu)
```

2. Wyprowadzić wzory na uogólnioną regułę delty i algorytm propagacji wstecznej błędu.

Dysponujemy siecią jednowarstwową z elementami przetwarzającymi o nieliniowej niemalejącej i różniczkowalnej funkcji aktywacji f . Zmiana wag wyraża się wzorem:

$$\Delta^{\mu} w_{ji} = -\eta \frac{\partial \xi_{\mu}}{\partial w_{ji}} = -\eta \frac{\partial \xi_{\mu}}{\partial \varphi_j^{\mu}} \frac{\partial \xi_{\mu}}{\partial w_{ji}} \quad (1)$$

Druga z pochodnych cząstkowych jest równa u_i^{μ} . Zdefiniujmy nową zmienną:

$$\delta_j^{f\mu} = -\frac{\partial \xi_{\mu}}{\partial \varphi_j^{\mu}} = -\frac{\partial \xi_{\mu}}{\partial y_j^{\mu}} \frac{\partial y_j^{\mu}}{\partial \varphi_j^{\mu}} \quad (2)$$

gdzie

$$\frac{\partial y_j^{\mu}}{\partial \varphi_j^{\mu}} = f'(\varphi_j^{\mu}) \quad (3)$$

oraz $f'(\varphi)$ oznacza pochodną funkcji aktywacji względem zmiennej φ .

Na podstawie wzorów (2), (3) zapiszemy:

$$\delta_j^{f\mu} = f'(\varphi_j^{\mu}) [y_j^{\mu} - y_j^{\mu}] = f'(\varphi_j^{\mu}) \delta_j^{\mu} \quad (4)$$

Ostatecznie wzór (1) przyjmuje postać:

$$\Delta^{\mu} w_{ji} = -\eta \delta_j^{f\mu} u_i^{\mu} \quad (5)$$

Wzór (5) nosi nazwę uogólnionej reguły delty dla nieliniowej sieci wielowarstwowej.

Rozważmy M-warstwową sieć elementów o jednakowych nierosnących i różniczkowalnych funkcjach aktywacji. Jeżeli przez $u_i^{m\mu}$ oznaczymy wartość wyjścia i-tego elementu m-tej warstwy przy prezentacji μ -tego wzorca, to aktywność j-tego elementu przetwarzającego m-tej warstwy wyrazi się wzorem:

$$u_j^{m\mu} = f(\varphi_j^{m\mu}) = f\left(\sum_{i=0}^{n_{m-1}} w_{ji}^m u_i^{(m-1)\mu}\right) \quad (6)$$

Zdefiniujemy błąd powstały w j-tym elemencie m-tej warstwy analogicznie do wzoru (2), tzn.

$$\delta_j^{m\mu} = -\frac{\partial \xi_\mu}{\partial \varphi_j^{m\mu}} = -\frac{\partial \xi_\mu}{\partial u_j^{m\mu}} \frac{\partial u_j^{m\mu}}{\partial \varphi_j^{m\mu}} = -\frac{\partial \xi_\mu}{\partial u_j^{m\mu}} f'(\varphi_j^{m\mu}) \quad (7)$$

Dla warstwy wyjściowej zapiszemy:

$$\frac{\partial \xi_\mu}{\partial u_j^{M\mu}} = \frac{\partial \xi_\mu}{\partial y_j^\mu} = -(y_j^\mu - y_j^\mu) = -\delta_j^\mu \quad (8)$$

Należy pamiętać, że $u_i^{m\mu}$ jest jednym ze składników sum ważonych liczonych we wszystkich elementach warstwy m+1. Możemy zatem napisać:

$$\begin{aligned} \frac{\partial \xi_\mu}{\partial u_j^{m\mu}} &= \sum_{l=1}^{n_{m+1}} \frac{\partial \xi_\mu}{\partial \varphi_l^{(m+1)\mu}} \frac{\partial \varphi_l^{(m+1)\mu}}{\partial u_j^{m\mu}} = \sum_{l=1}^{n_{m+1}} \frac{\partial \xi_\mu}{\partial \varphi_l^{(m+1)\mu}} \frac{\partial}{\partial u_j^{m\mu}} \sum_{i=1}^{n_m} w_{li}^{(m+1)} u_i^{m\mu} = \sum_{l=1}^{n_{m+1}} \frac{\partial \xi_\mu}{\partial \varphi_l^{(m+1)\mu}} w_{lj}^{(m+1)} = \\ &= -\sum_{l=1}^{n_{m+1}} \delta_l^{(m+1)\mu} w_{lj}^{(m+1)} \end{aligned} \quad (9)$$

Stąd na podstawie (7), (8), (9) możemy każdemu elementowi przetwarzającemu przypisać błąd:

- dla warstwy wyjściowej:

$$\delta_j^{M\mu} = f'(\varphi_j^{M\mu}) \delta_j^\mu = f'(\varphi_j^{M\mu}) (y_j^\mu - y_j^\mu) \quad (10)$$

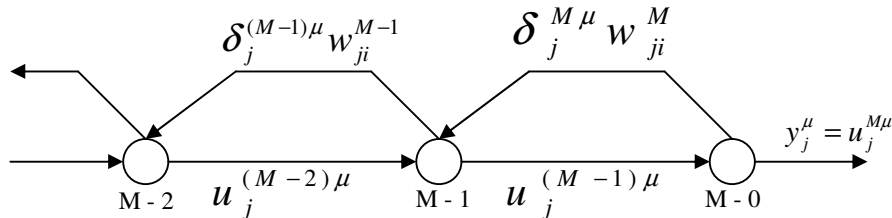
- dla warstw ukrytych:

$$\delta_j^{m\mu} = f'(\varphi_j^{m\mu}) \sum_{l=1}^{n_{m+1}} \delta_l^{(m+1)\mu} w_{lj}^{(m+1)} \quad (11)$$

Dla tak zdefiniowanego błędu możemy wprowadzić uogólnioną regułę delty:

$$\Delta^\mu w_{ji}^m = \eta \delta_j^{m\mu} u_i^{(m-1)\mu}$$

Poniższy rysunek przedstawia zasadę wstecznej propagacji błędów:



Rys.1. Ilustracja zasady wstecznej propagacji błędów.

3. Przy pomocy wielowarstwowej sieci jednokierunkowej dokonać zadania aproksymacji węzłów aproksymacji. Sprawdzić czy można polepszyć proces uczenia przy zastosowaniu technik momentu i adaptacyjnego kroku uczenia.

Do realizacji zadania wykorzystaliśmy skrypt zad1.m

```
% Dane uczace
load dane
u=wej';y=wyj';

% Tworzenie i inicowanie sieci
siec=newff([0 50],[5 1],{'tansig' 'purelin'});

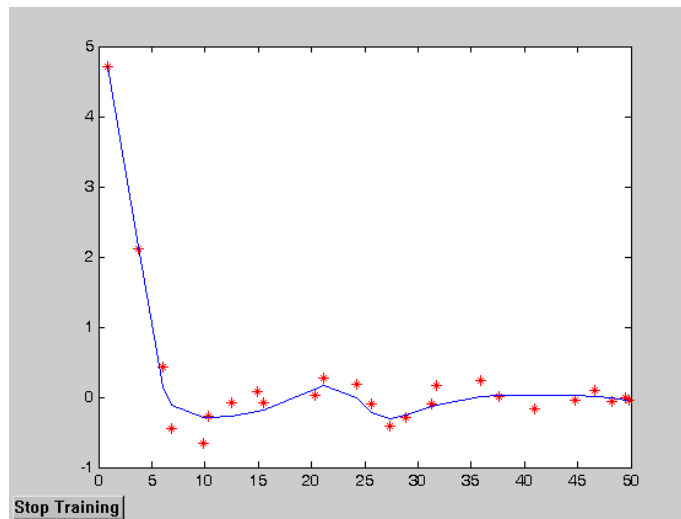
% Ustawienie parametrow uczacych
siec.trainfcn='traingd'; % Algorytm uczenia
siec.trainparam.epochs=10000; % Maksymalna liczba iteracji
siec.trainparam.lr=0.01; % Krok uczenia
siec.trainparam.goal=0.01; % Zadana jakosc
siec.trainparam.show=100; % Czestotliwosc wyswietlania

% Proces uczenia sieci
siec=train(siec,u,y);

% Proces testowania sieci
ys=sim(siec,u);

% Wykres wynikow
plot(u,y,'r*',u,ys);
```

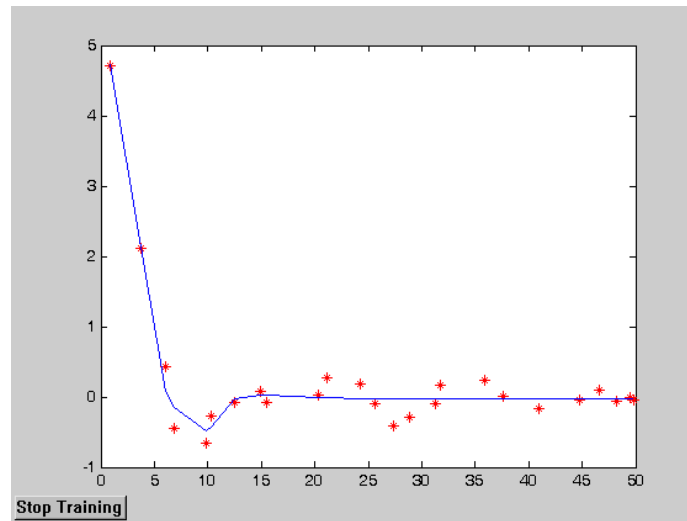
a) algorytm gradientowy



Rys.1. Wynik aproksymacji dla algorytmu gradientowego.

Po 10000 kroków otrzymaliśmy błąd: MSE 0.0294903

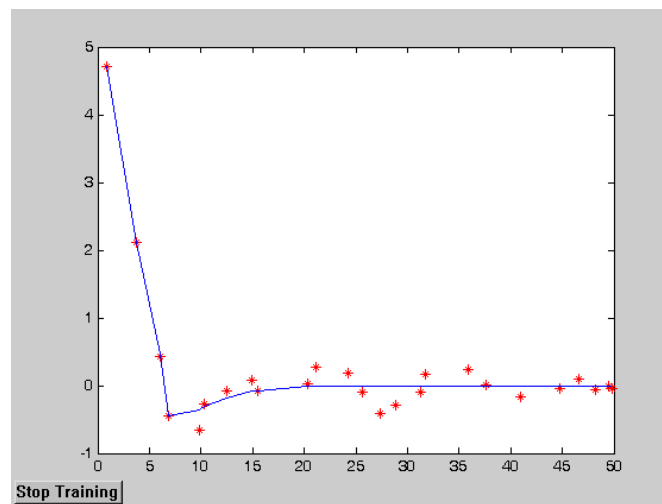
b) algorytm gradientowy z momentum



Rys.2. Wynik aproksymacji dla algorytmu gradientowego z momentum.

Po 10000 kroków otrzymaliśmy błąd: MSE 0.031524

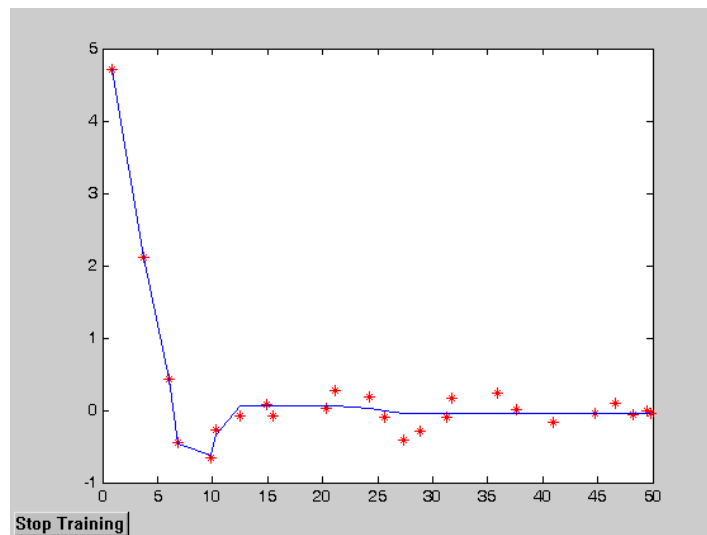
c) algorytm gradientowy z adaptacyjnym krokiem uczenia



Rys.3. Wynik aproksymacji dla algorytmu gradientowego z adaptacyjnym krokiem uczenia.

Po 10000 kroków otrzymaliśmy błąd: MSE 0.0261091

d) algorytm gradientowy z adaptacyjnym krokiem uczenia i momentum



Rys.4. Wynik aproksymacji dla algorytmu gradientowego z adaptacyjnym krokiem uczenia i momentum

Po 10000 kroków otrzymaliśmy błąd: MSE 0.0192865

4. W oparciu o wielowarstwową sieć zaprojektować system rozpoznawania cyfr lub liter. Dobrać odpowiednią ilość warstw i neuronów w sieci. Sprawdzić jak wpływają zakłócenia obrazów wyjściowych na pracę systemu.

W zadaniu tym zaproponowaliśmy system rozpoznawania cyfr. Stworzona przez nas sieć posiada dwie warstwy neuronów. Istnieje 10 wyjść. W zależności od rozpoznanej cyfry na odpowiadającym jej wyjściu pojawia się 1. Np. dla 0 na wyjściach powinno pojawić się:
1 0 0 0 0 0 0 0 0 0

Reprezentacja cyfr opiera się na macyzy o wymiarach 6x4.

Np. cyfrze 2 odpowiada:

```
z2=[1 1 1 1
    0 0 0 1
    0 0 1 0
    0 1 0 0
    1 0 0 0
    1 1 1 1];
```

Do realizacji ćwiczenia wykorzystaliśmy skrypt:

```
%reprezentacja poszczegolnych cyfr
z0=[1 1 1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 1 1 1 1];
z1=[0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 1];
z2=[1 1 1 1 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 1 1 1 1];
z3=[1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 1 0 0 0 1 1 1 1 1 1];
z4=[1 0 0 1 1 0 0 1 1 1 1 1 0 0 0 1 0 0 0 1 0 0 0 1 1];
z5=[1 1 1 1 1 0 0 0 1 1 1 1 0 0 0 1 0 0 0 1 1 1 1 1 1];
z6=[1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 1];
z7=[1 1 1 1 1 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 0];
z8=[1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 0 0 1 1 1 1 1 1];
z9=[1 1 1 1 1 0 0 1 1 1 1 1 0 0 1 0 0 1 0 0 1 0 1 1 0 0];

%zamiana na kolumny
```

```

z0=z0';
z1=z1';
z2=z2';
z3=z3';
z4=z4';
z9=z9';
z5=z5';
z6=z6';
z7=z7';
z8=z8';

Z=[z0,z1,z2,z3,z4,z5,z6,z7,z8,z9]; %polaczenie w reprezentacji cyfr w jedna macierz wzorcowa
T=eye(10); %oczekiwane wartości wyjsc

siec=newff(minmax(Z),[10 10],{'logsig' 'logsig'}); %tworzenie sieci

siec.trainFcn='traingdx'; % Algorytm uczenia
siec.trainParam.goal = 0.01; % Oczekiwana wartosc bledu
siec.trainParam.show = 20; % Czestotliwosc wyswietlania
siec.trainParam.epochs = 5000;% Maksymalna liczba iteracji
siec.trainParam.mc = 0.95; % Stala momentu

siec=train(siec,Z,T); %uczenie sieci
A=sim(siec,Z) %symulacja sieci
E=A-T %wyznaczenie bledu

```

Otrzymane wyniki:

```

A =
0.9948 0.0000 0.0000 0.0000 0.0000 0.0000 0.0005 0.0000 0.0000 0.0000
0.0000 0.9997 0.0000 0.0000 0.0003 0.0000 0.0003 0.0000 0.0000 0.0000
0.0001 0.0000 0.9932 0.0000 0.0000 0.0000 0.0000 0.0109 0.0003 0.0000
0.0000 0.0000 0.0000 0.9700 0.0002 0.0001 0.0000 0.0000 0.0271 0.0000
0.0000 0.0001 0.0000 0.0006 0.9996 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0006 0.0000 0.0007 0.0004 0.9988 0.0005 0.0000 0.0000 0.0001
0.0001 0.0001 0.0000 0.0000 0.0000 0.0000 0.0031 0.9977 0.0000 0.0054
0.0004 0.0001 0.0002 0.0001 0.0001 0.0000 0.0000 0.9979 0.0007 0.0002
0.0019 0.0014 0.0222 0.0359 0.0063 0.0001 0.0358 0.0221 0.9370 0.0010
0.0000 0.0002 0.0001 0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.9942

E =
-0.0052 0.0000 0.0000 0.0000 0.0000 0.0000 0.0005 0.0000 0.0000 0.0000
0.0000 -0.0003 0.0000 0.0000 0.0000 0.0000 0.0003 0.0000 0.0003 0.0000
0.0001 0.0000 -0.0068 0.0000 0.0000 0.0000 0.0000 0.0109 0.0003 0.0000
0.0000 0.0000 0.0000 -0.0300 0.0002 0.0001 0.0000 0.0000 0.0271 0.0000
0.0000 0.0001 0.0000 0.0006 -0.0004 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0006 0.0000 0.0007 0.0004 -0.0012 0.0005 0.0000 0.0000 0.0001
0.0001 0.0001 0.0000 0.0000 0.0000 0.0031 -0.0023 0.0000 0.0054 0.0000
0.0004 0.0001 0.0002 0.0001 0.0001 0.0000 0.0000 -0.0021 0.0007 0.0002
0.0019 0.0014 0.0222 0.0359 0.0063 0.0001 0.0358 0.0221 -0.0630 0.0010
0.0000 0.0002 0.0001 0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 -0.0058

```

Następnie wprowadziliśmy zakłócenia zmieniając w reprezentacji poszczególnych cyfr po 3 wartości. Otrzymane wyniki przedstawione są poniżej:

```

Z =
1 0 1 1 1 1 1 1 1 1
1 0 1 1 0 1 1 1 1 1
0 1 1 1 0 1 1 1 1 1
1 1 0 1 1 1 1 1 1 1
1 0 0 1 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
1 1 1 1 1 0 0 1 1 1
0 0 0 1 1 1 1 0 1 1
0 1 0 1 1 1 1 0 1 1
0 0 0 1 0 0 0 1 1 0
1 1 0 1 1 1 1 1 1 1
1 0 0 1 0 0 1 0 1 0
0 0 1 0 0 0 0 1 1 0
0 0 0 0 0 0 0 0 0 1
0 1 0 1 1 1 1 0 1 0
1 1 1 0 0 0 0 1 1 0
0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 1 0 0

```

```

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

```

A =

```

0.9923 0.0002 0.0006 0.0000 0.0000 0.0000 0.0000 0.0002 0.0000 0.0000 0.0000
0.0000 0.9521 0.0008 0.0000 0.1014 0.0001 0.0002 0.0000 0.0000 0.0000 0.0000
0.0454 0.0004 0.9996 0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0003 0.0000
0.0000 0.0000 0.0027 0.0261 0.0046 0.0001 0.0000 0.0000 0.0000 0.2262 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.9966 0.0000 0.0000 0.0000 0.0002 0.0000
0.0000 0.0008 0.0000 0.0000 0.0040 0.9964 0.0030 0.0000 0.0000 0.0000 0.0001
0.0000 0.0011 0.0026 0.0000 0.0000 0.0473 0.9940 0.0000 0.0000 0.0492 0.0000
0.0071 0.0001 0.1892 0.0003 0.0001 0.0000 0.0000 0.0000 0.9991 0.0006 0.0002
0.0171 0.0062 0.0703 0.3666 0.0130 0.0001 0.0173 0.1600 0.9675 0.0011
0.0196 0.0008 0.0000 0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.9905

```

E =

```

-0.0077 0.0002 0.0006 0.0000 0.0000 0.0000 0.0002 0.0000 0.0000 0.0000 0.0000
0.0000 -0.0479 0.0008 0.0000 0.1014 0.0001 0.0002 0.0000 0.0000 0.0000 0.0000
0.0454 0.0004 -0.0004 0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0003 0.0000
0.0000 0.0000 0.0027 -0.9739 0.0046 0.0001 0.0000 0.0000 0.0000 0.2262 0.0000
0.0000 0.0000 0.0000 0.0000 -0.0034 0.0000 0.0000 0.0000 0.0000 0.0002 0.0000
0.0000 0.0008 0.0000 0.0000 0.0040 -0.0036 0.0030 0.0000 0.0000 0.0000 0.0001
0.0000 0.0011 0.0026 0.0000 0.0000 0.0473 -0.0060 0.0000 0.0000 0.0492 0.0000
0.0071 0.0001 0.1892 0.0003 0.0001 0.0000 0.0000 -0.0009 0.0006 0.0002
0.0171 0.0062 0.0703 0.3666 0.0130 0.0001 0.0173 0.1600 -0.0325 0.0011
0.0196 0.0008 0.0000 0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 -0.0095

```

W kolejnym punkcie zwiększyliśmy poziom zakłóceń zmieniając w reprezentacji poszczególnych cyfr po 5 wartości. Otrzymane wyniki przedstawione są poniżej:

Z =

```

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

```

A =

```

0.9749 0.0000 0.6716 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.1340 0.0000 0.0000 0.0004 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.9984 0.0061 0.9977 0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0003 0.0001
0.0222 0.0000 0.0000 0.5170 0.0007 0.0230 0.0003 0.0000 0.1852 0.9044
0.0000 0.0000 0.0000 0.0002 0.9997 0.0024 0.0000 0.0000 0.0003 0.0001
0.0000 0.0000 0.0000 0.0000 0.0005 0.1382 0.9992 0.0001 0.0000 0.0015
0.0021 0.0000 0.0000 0.0001 0.0000 0.0000 0.0128 0.0000 0.0464 0.0000
0.0011 0.0008 0.2633 0.0002 0.0001 0.0000 0.0000 0.9942 0.0006 0.0001
0.0536 0.2498 0.0007 0.4307 0.0087 0.0022 0.0001 0.0005 0.9663 0.0247
0.0000 0.1799 0.0000 0.0000 0.0000 0.0000 0.0000 0.0327 0.0000 0.0008

```

E =

```

-0.0251 0.0000 0.6716 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 -0.8660 0.0000 0.0000 0.0004 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

```

0.9984	0.0061	-0.0023	0.0001	0.0000	0.0000	0.0000	0.0000	0.0003	0.0001
0.0222	0.0000	0.0000	-0.4830	0.0007	0.0230	0.0003	0.0000	0.1852	0.9044
0.0000	0.0000	0.0000	0.0002	-0.0003	0.0024	0.0000	0.0000	0.0003	0.0001
0.0000	0.0000	0.0000	0.0000	0.0005	-0.8618	0.9992	0.0001	0.0000	0.0015
0.0021	0.0000	0.0000	0.0001	0.0000	0.0000	-0.9872	0.0000	0.0464	0.0000
0.0011	0.0008	0.2633	0.0002	0.0001	0.0000	0.0000	-0.0058	0.0006	0.0001
0.0536	0.2498	0.0007	0.4307	0.0087	0.0022	0.0001	0.0005	-0.0337	0.0247
0.0000	0.1799	0.0000	0.0000	0.0000	0.0000	0.0000	0.0327	0.0000	-0.9992

Wnioski

Celem ćwiczenia było poznanie struktur sieci neuronowych zbudowanych z wielu warstw neuronów oraz zapoznanie się z algorytmem wstecznej propagacji błęd.

W zadaniu 1 dokonaliśmy uczenia sieci dwuwarstwowej. Okazuje się, że taka sieć doskonale radzi sobie z problemem XOR, który był nie do rozwiązania przy pomocy sieci jednowarstwowej. Sieć wielowarstwowa potrafi rozwiązać problem klasyfikacji danych, które nie są liniowo separowalne. Jest to duża zaleta sieci wielowarstwowych. Dodatkowo badaliśmy uczenie sieci dla różnych funkcji aktywacji. Okazuje się, że dla danych z zadania najlepsze wyniki daje zastosowanie funkcji tangensoidalnej w warstwie 1 i liniowej w warstwie 2.

W trakcie realizacji ćwiczenia zapoznaliśmy się z graficznym interfejsem użytkownika nntool. Służy on do tworzenia, uczenia oraz symulowania sieci neuronowych. Pozwala szybko i w stosunkowo prosty sposób zaimplementować sieć o żądanych parametrach.

W zadaniu 2 wyprowadziliśmy wzory na uogólnioną regułę delty oraz algorytm propagacji wstecznej błęd.

W zadaniu 3 dokonaliśmy aproksymacji węzłów przy pomocy wielowarstwowej sieci jednokierunkowej. Rysunek 1 pokazuje, że stworzona sieć dobrze aproksymuje zadane węzły. Otrzymaliśmy błąd na poziomie 0.0294903. Następnie sprawdziliśmy możliwość poprawienia rozwiązania przez zastosowanie technik momentum oraz adaptacyjnego kroku uczenia. Zastosowanie adaptacyjnego kroku uczenia pozwoliło zredukować poziom błęd do wartości 0.0261091. Jednak najlepszy rezultat dało zastosowanie algorytmu gradientowego z adaptacyjnym krokiem uczenia i momentum. Błąd na poziomie 0.0192865.

W ostatnim zadaniu w oparciu o wielowarstwową sieć jednokierunkową zaprojektowaliśmy system rozpoznawania cyfr. Jak się okazało sieci neuronowe świetnie nadają się do realizacji tego typu systemów. Doświadczalnie sprawdziliśmy właściwości uogólniające sieci. Wprowadziliśmy losowe zakłócenia na poziomie 12%. Okazało się, że 90% danych wyjściowych jest poprawnych. Następnie wprowadziliśmy zakłócenia na poziomie 21%. Na wyjściu 60% danych było poprawnych.