

Uniwersytet Zielonogórski	Wykonali:	Grupa:	Nr ćwiczenia: 2	Ocena:
Laboratorium				
Temat ćwiczenia: Liniowe sieci neuronowe.		Prowadzący:	Data wyk. ćw.	Data odd. spr.

Zadanie 1. Przeprowadzić uczenie neuronu z linową funkcją aktywacji z jednym wejściem dla danych przedstawionych poniżej. Porównać wagi nauczonego neuronu przy użyciu reguły Widrowa-Hoffa (za pomocą funkcji train) z wagami wyliczonymi bezpośrednio, zbadać czy nauczony neuron dobrze działa dla danych które służyły jako wzorce uczące.

Do wyznaczania wag za pomocą funkcji train wykorzystaliśmy następujący skrypt:

```
net=newlin([0,8],1);
net=train(net,P,T);
A=sim(net,P)
E=A-T
W=net.IW{1,1}
B=net.b{1}
```

Do wyznaczania wag w sposób bezpośredni wykorzystaliśmy skrypt:

```
net=newlind(P,T);
net.IW{1,1}
net.b{1}
A=sim(net,P);
E=A-T
```

Wyniki przedstawione są poniżej:

a) $P=[1 \ -1.2]$ $T=[1 \ 0.5]$

Funkcja train:

A = 0.8540	0.4347	A - odpowiedź neuronu
E = -0.1460	-0.0653	E - błąd uczenia
W = 0.1906		W - wagi
B = 0.6634		B - bias

Sposób bezpośredni:

```
W= 0.2273
B= 0.7727
A=1 0.5
E= 0 0
```

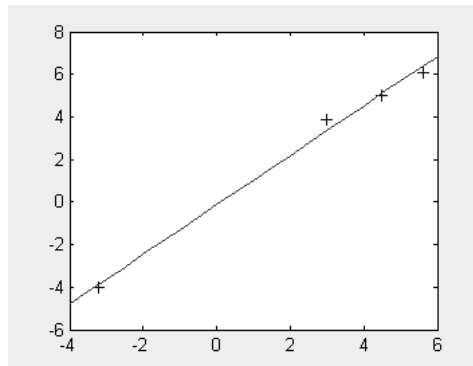
b) $P=[-3.2 \ 4.5 \ 5.6 \ 3]$ $T=[-4 \ 5 \ 6.1 \ 3.9]$

Funkcja train:

A = -3.8222 5.1141 6.3907 3.3733
E = 0.1778 0.1141 0.2907 -0.5267
W = 1.1606
B = -0.1084

Sposób bezpośredni:

W=1.1637
B=-0.1302
E=0.1460 0.1065 0.2866 -0.5391



Rys.1.

c) $P=[1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7]$ $T=[3 \ 5 \ 7 \ 9 \ 11 \ 13 \ 15]$

Funkcja train:

A = 2.8821 4.9119 6.9418 8.9716 11.0014 13.0313 15.0611
E = -0.1179 -0.0881 -0.0582 -0.0284 0.0014 0.0313 0.0611
W = 2.0298
B = 0.8522

Sposób bezpośredni:

W=2
B=1.0000
E=0 0 0 0 0 0

Zadanie 2. Przeprowadzić uczenie liniowej sieci jednowarstwowej przy użyciu funkcji adapt oraz train.

Funkcja train:

$P=[1 \ 1.5 \ 1.2 \ -0.3; -1 \ 2 \ 3 \ -0.5; 2 \ 1 \ -1.6 \ 0.9];$
 $T=[0.5 \ 3 \ -2.2 \ 1.4; 1.1 \ -1.2 \ 1.7 \ -0.4; 3 \ 0.2 \ -1.8 \ -0.4];$

Dla powyższych danych wykorzystaliśmy skrypt:

```
net=newlin([-2 2;-2 2;-2 2],3);  
i=0;  
while(sse(E))  
    i=i+1;  
    net=train(net,P,T);  
    if i==20 break; end //przerwanie po 20 iteracjach  
end
```

```
A=sim(net,P)
E=A-T
```

Otrzymane wyniki:

```
A =
    0.5014    2.9988   -2.1991    1.3991
    1.0989   -1.1990    1.6993   -0.3993
    2.9994    0.2005   -1.8004   -0.3996

E =
    0.0014   -0.0012    0.0009   -0.0009
   -0.0011    0.0010   -0.0007    0.0007
   -0.0006    0.0005   -0.0004    0.0004
```

Funkcja adapt:

```
net=newlin([-2 2;-2 2;-2 2],3);
i=0;
while(sse(E))
    i=i+1;
    net=adapt(net,P,T);
    if i==500 break; end //przerwanie po 500 iteracjach
end
A=sim(net,P)
E=A-T
```

```
A =
    0.7893    2.7435   -2.0087    1.2280
    0.8708   -0.9964    1.5481   -0.2646
    2.8749    0.3039   -1.8764   -0.3115
```

```
E =
    0.2893   -0.2565    0.1913   -0.1720
   -0.2292    0.2036   -0.1519    0.1354
   -0.1251    0.1039   -0.0764    0.0885
```

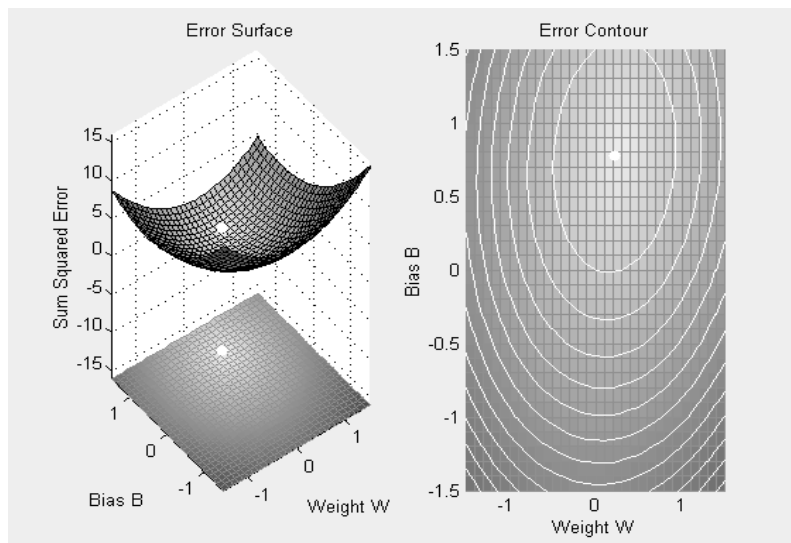
Zadanie 3. Dla zadań z punktu pierwszego wyznaczyć wykresy funkcji błędu wraz z zaznaczonym punktem określającym wartości nauczonych wag oraz błędu:

Do realizacji zadania wykorzystaliśmy poniższy skrypt:

```
w_range=-1.5:0.1:1.5;
b_range=w_range;
ES=errsurf(P,T,w_range,b_range,'purelin');
plotes(w_range,b_range,ES);
net=newlind(P,T);
A=sim(net,P);
E=T-A;
SSE=sumsq(E);
plotes(w_range,b_range,ES);
plotep(net.IW{1,1},net.b{1},SSE);
```

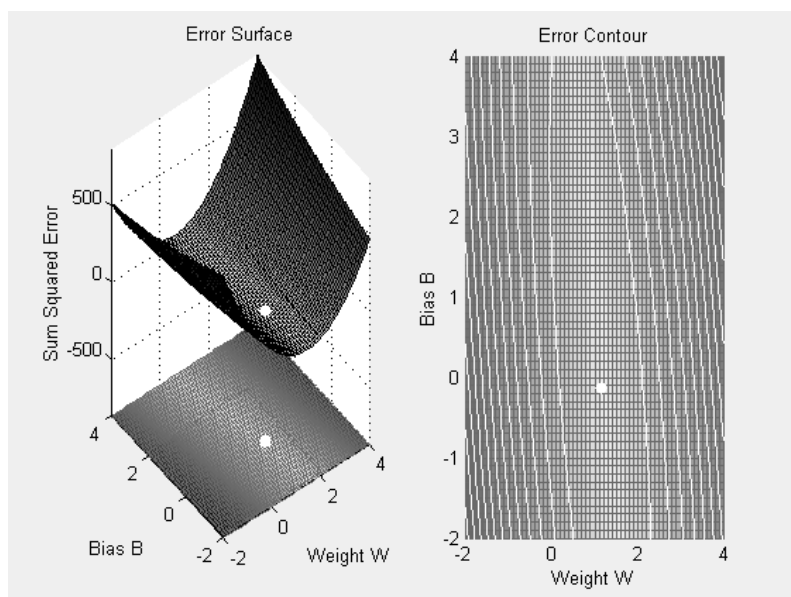
Otrzymane wykresy:

a)



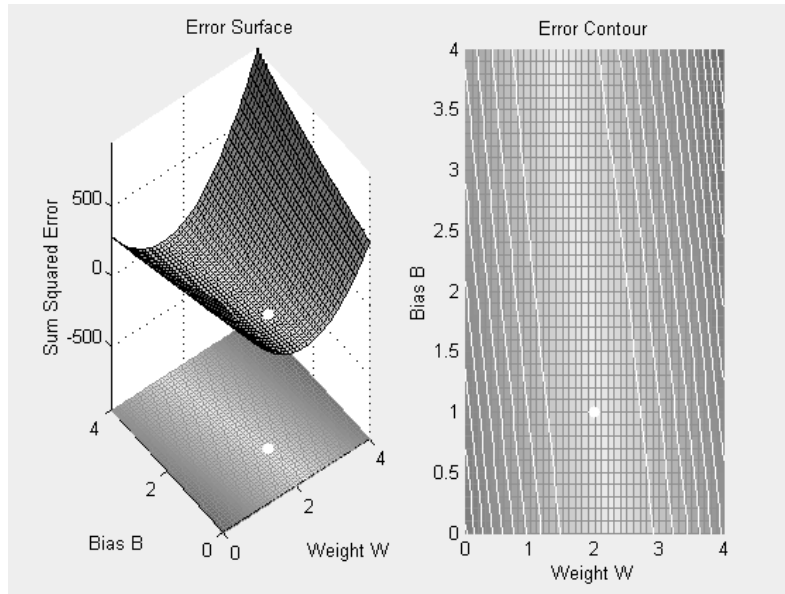
Rys.2.

b)



Rys.3.

c)



Rys.4.

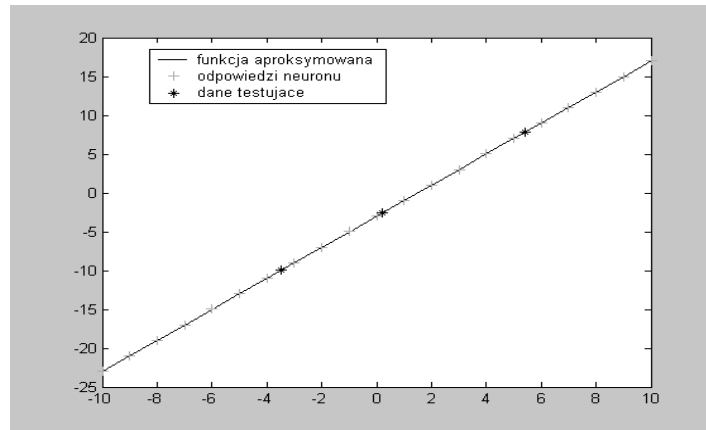
Zadanie 4. Dokonać aproksymacji funkcji w przedziale $(-10;10)$ przy użyciu linowego neuronu. Zbadać właściwości uogólniające neuronu.

a) $y=2x-3$

Zadanie zrealizowaliśmy z wykorzystaniem skryptu:

```
P=-10:1:10;
T=2.*P-3;
plot(P,T,'-b')
net=newlind(P,T);
A=sim(net,P);
hold on
plot(P,A,'+g')
E=A-T
dane_testowe=[0.2 5.4 -3.5];
At=sim(net,dane_testowe);
plot(dane_testowe,At,'*k')
```

Otrzymane wyniki przedstawione są na poniższym wykresie:



Rys. 5.

Wartosci bledu:

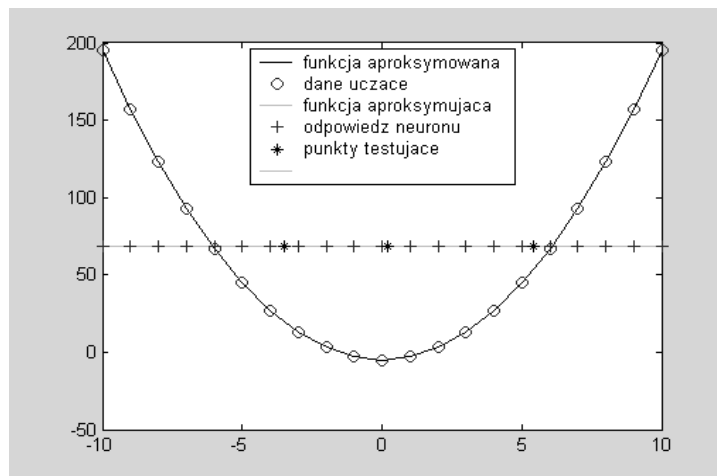
```
E= 1.0e-013 * [ 0.1066 0.1066 0.1066 0.1066 0.0711 0.0533 0.0533 0.0533
0.0355 0.0355 0.0222 0.0133 0.0044 -0.0044 -0.0178 -0.0178 -0.0355 -0.0533
-0.0533 -0.0533 -0.0355]
```

a) $y=2x^2-5$

Zadanie zrealizowaliśmy z wykorzystaniem skryptu:

```
P=-10:1:10;
T=2.*P.^2-5;
plot(P,T,'-b')
hold on
net=newlind(P,T);
plot(P,T,'or')
A=sim(net,P);
funkcja_aproksymujaca=net.IW{1,1}.*P+net.b{1};
plot(P,funkcja_aproksymujaca,'-g')
plot(P,A,'+r')
E=A-T
dane_testowe=[0.2 5.4 -3.5];
At=sim(net,dane_testowe);
plot(dane_testowe,At,'*k')
```

Otrzymane wyniki przedstawione są na poniższym wykresie:



Rys. 6.

Wartosci bledu:

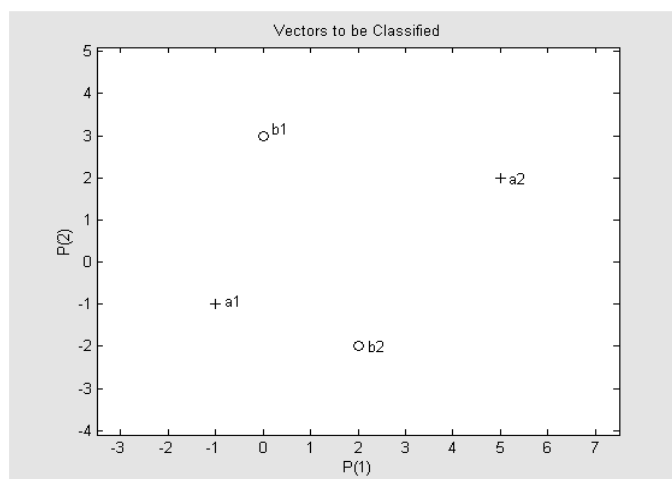
```
E=[ -126.6667 -88.6667 -54.6667 -24.6667 1.3333 23.3333 41.3333 55.3333  
65.3333 71.3333 73.3333 71.3333 65.3333 55.3333 41.3333 23.3333 1.3333  
-24.6667 -54.6667 -88.6667 -126.6667]
```

Zadanie 5. Znalezc zadanie klasyfikacji, którego liniowy neuron nie potrafi rozwiązać.

Przykładowe zadanie:

$A=\{a_1, a_2\}$, $B=\{b_1, b_2\}$, $a_1=(-1, -1)$, $a_2=(5, 2)$, $b_1=(0, 3)$, $b_2=(2, -2)$

Jak wynika z poniższego rysunku takie rozmieszczenie punktów nie pozwala zrealizować zadania klasyfikacji wzorców z wykorzystaniem liniowego neuronu.



Rys. 7.

Zadanie 6. Zbadac jak liniowy neuron uczy się dla roznych wielkości kroku uczenia dla następujących danych:

$P=[1.0 \ -1.2]$ $T=[0.5 \ 1.0]$

Zbadac uczenie się dla następujących wielkości kroku uczenia:

0.01, 0.1, 5, 10, 50

Porównać otrzymane wyniki z uczeniem przy uzyciu kroku uczenia wyznaczonego przez funkcje maxlinlr.

Do realizacji zdania wykorzystaliśmy skrypt:

```
net=newlin([-2 2], 1, [0], krok);  
net=train(net, P, T);  
A=sim(net, P)  
E=A-T
```

a) krok=0.01

Otrzymane wyniki:

A = 0.4105	0.9024	A - odpowiedź neuronu
E = -0.0895	-0.0976	E - błąd uczenia

b) krok=0.1

Otrzymane wyniki:

```
A = 0.5000 1.0000
E = 1.0e-009 *[-0.4111 -0.1591]
```

c) krok=5,10,50

Dla powyższych kroków przeprowadzenie uczenia było niemożliwe, ponieważ funkcja newlin przyjmuje wartości kroku tylko z przedziału 0..1

d) krok=maxlinlr(P,'bias')

```
krok = 0.3972
A = 0.5000 1.0000
E = 0 0
```

7. Przeprowadzić adaptacyjne uczenie liniowej sieci neuronowej tak aby dokonywała ona predykcji dla następującego systemu:

```
time=0:0.01:3;
```

Sygnal wejściowy dany jest następująco:

```
X=cos(time*10)+sin(time+4);
P=con2seq(X);
```

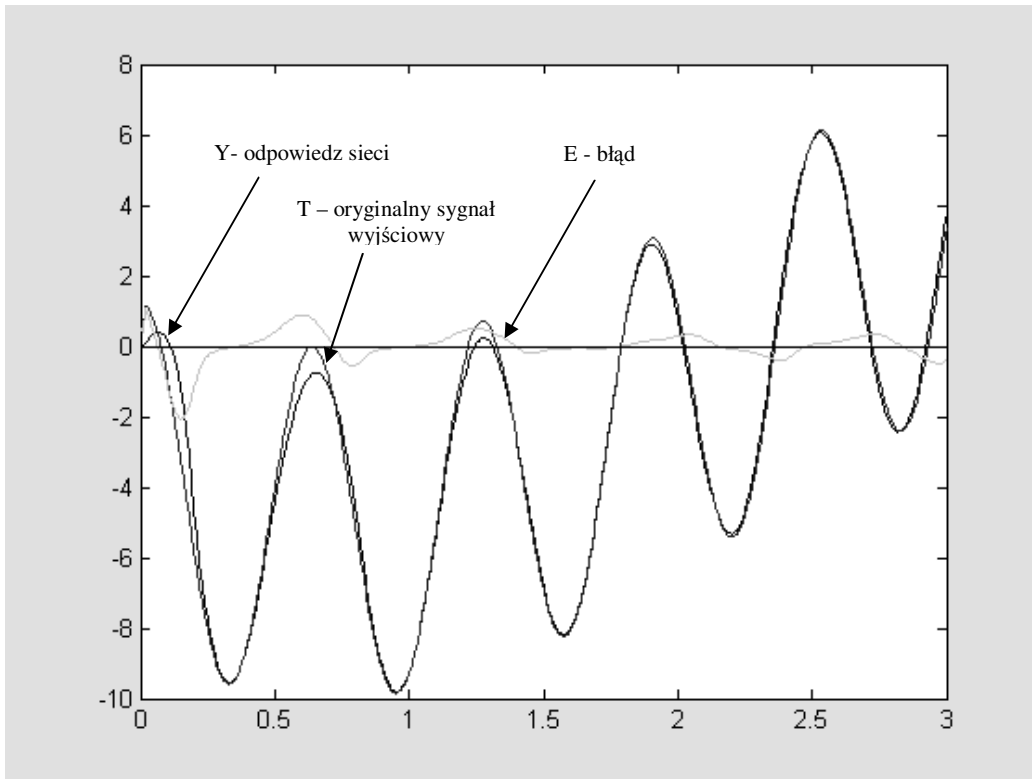
Oryginalny sygnal wyjściowy dany jest następująco:

```
T=con2seq(3*[0 0 X(1:(end-2))]+X.*2);
```

Wykorzystalismy ponizszy skrypt:

```
time=0:0.01:3;
X=cos(time*10)+sin(time+4);
P=con2seq(X);
T=con2seq(3*[0 0 X(1:(end-2))]+X.*2);
%%plot(time,cat(2,P{:}),time,cat(2,T{:}),'--');
net=newlin([-3 3],1,[0 1],0.1);
[net,Y,E,Pf]=adapt(net,P,T);
plot(time,cat(2,Y{:}),'b',time,cat(2,T{:}),'r',time,cat(2,E{:}),'g',[0
3],[0 0],'k');
```

Wyniki uczenia przedstawia ponizszy wykres:



Rys. 8.

Wnioski

Celem ćwiczenia jest poznanie struktur sieci neuronowych zbudowanych przy użyciu neuronów liniowych jak również poznanie metod ich uczenia oraz zbadanie zalet oraz ograniczeń niniejszych struktur.

W zadaniu 1 porównywaliśmy uczenie neuronu za pomocą reguły Widrowa-Hoffa z metodą bezpośredniego wyznaczania wag. We wszystkich przypadkach bezpośrednio wyznaczenie wag dało lepsze rezultaty (mniejsza wartość błędu). Tylko w podpunkcie b) wartości błędu były różne od zera. Wynika to z faktu iż nie było możliwości przeprowadzenia prostej przez wszystkie punkty (Rys.1.). Dlatego została przeprowadzona aproksymacja punktów za pomocą prostej.

Zadanie 2 polegało na uczeniu sieci jednowarstwowej przy użyciu funkcji `adapt` oraz `train`. Jak wynika z przeprowadzonych symulacji, funkcja `train` dała lepsze wyniki. Dla zminimalizowania wartości błędu, uczenie neuronu realizowaliśmy w pętli. Już po 20 iteracjach pętli funkcja `train` daje lepsze wyniki niż funkcja `adapt` po 500 iteracjach.

W zadaniu 3 wyznaczyliśmy wykresy błędu uczenia. Ponieważ funkcja błędu średniokwadratowego jest funkcją kwadratową względem wag, posiada ona tylko jedno minimum globalne. W takim przypadku wyznaczenie wartości wag wymaga tylko znalezienia minimalnej wartości błędu. Jest to możliwe do wykonania w jednym kroku (funkcja `newlind`). Rysunki 2,3,4 przedstawiają wykresy błędu wraz z zaznaczonymi punktami, w których przyjmuje on wartość minimalną.

W zadaniu 4 dokonaliśmy aproksymacji funkcji liniowej oraz kwadratowej za pomocą liniowego neuronu. Z wykonanych testów wynika, że odpowiednio nauczony neuron idealnie realizuje funkcję liniową. Błąd uczenia przyjmuje wartości zerowe. Neuron wykazuje właściwości uogólniające, dla danych innych niż dane uczące odpowiedzi sieci są zgodne z oczekiwaniami. Neuron potrafi także aproksymować funkcję nieliniową za pomocą funkcji

liniowej. Potwierdza to przypadek funkcji kwadratowej. Uczony neuron dokonał aproksymacji funkcją liniową.

Neuron liniowy ma podobną wadę jak perceptron prosty. Potrafi rozwiązywać tylko te problemy które są linowo separowalne. Podobnie jak perceptron, pojedynczy neuron liniowy nie jest w stanie rozwiązać problemu XOR(rysunek 7).

W zadaniu 6 badaliśmy uczenie neuronu dla różnych wartości kroku uczenia. Najlepsze wyniki uczenia(błąd równy 0) osiągnęliśmy dla kroku wyznaczonego przez funkcję `maxlinlr`. Nie udało się przeprowadzić testów dla kroku równego 5,10,50 ponieważ funkcja `newlin` przyjmuje jedynie wartości kroku z przedziału 0..1.

W kolejnym zadaniu przeprowadziliśmy adaptacyjne uczenie sieci neuronowej. W uczeniu adaptacyjnym wagi neuronu są modyfikowane po każdej prezentacji wzorca uczącego. Widać to na rysunku 8. W miarę postępu procesu uczenia wartość błędu maleje i odpowiedź neuronu jest prawie identyczna z odpowiedzią oczekiwaną.