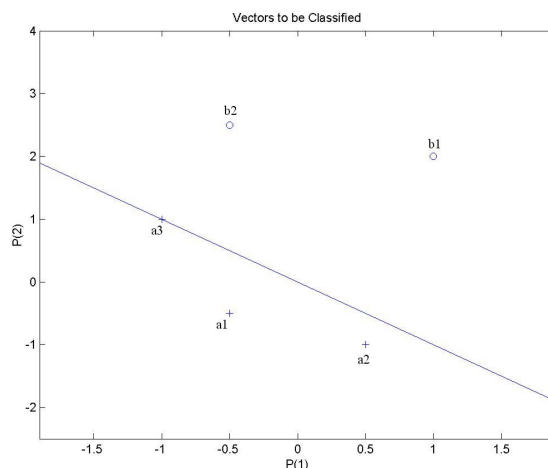


Uniwersytet Zielonogórski	Wykonali:	Grupa:	Nr ćwiczenia: 1	Ocena:
	Laboratorium systemów obliczeń inteligentnych			
Temat ćwiczenia: Sieci neuronowe – perceptron prosty.		Prowadzący:	Data wyk. ćw.	Data odd. spr.

Zadanie 1

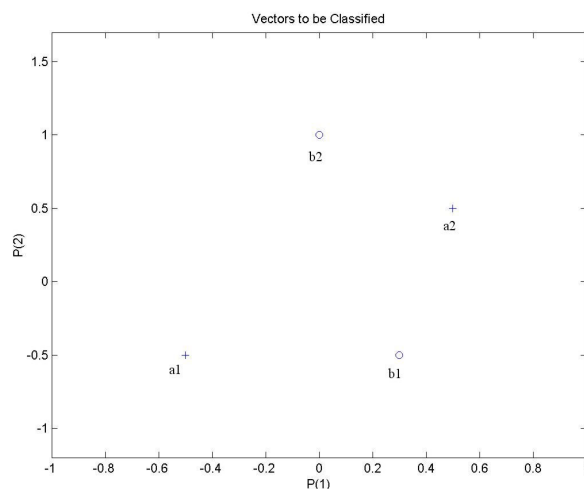
Zbadać czy poniższe zadania klasyfikacji wzorców możliwe są do realizacji przy użyciu perceptronu, jeżeli zadanie jest możliwe do realizacji wyznaczyć wektor wag w dla których neuron będzie klasyfikował elementy dwóch zbiorów A i B.

- a) $A=\{a_1,a_2,a_3\}, B=\{b_1,b_2\}$ gdzie $a_1=\{-0.5,-0.5\}$ $a_2=\{0.5,-1\}$ $a_3=\{-1,1\}$ $b_1=\{1,2\}$
 $b_2=\{-0.5,2.5\}$



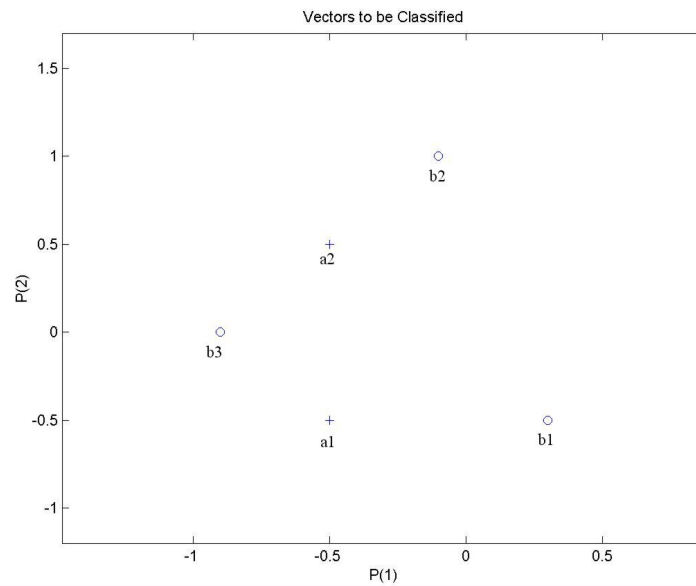
możliwe do realizacji

- b) $A=\{a_1,a_2\}, B=\{b_1,b_2\}$ gdzie $a_1=\{-0.5,-0.5\}$ $a_2=\{0.5,0.5\}$ $b_1=\{0.3,-0.5\}$ $b_2=\{0,1\}$



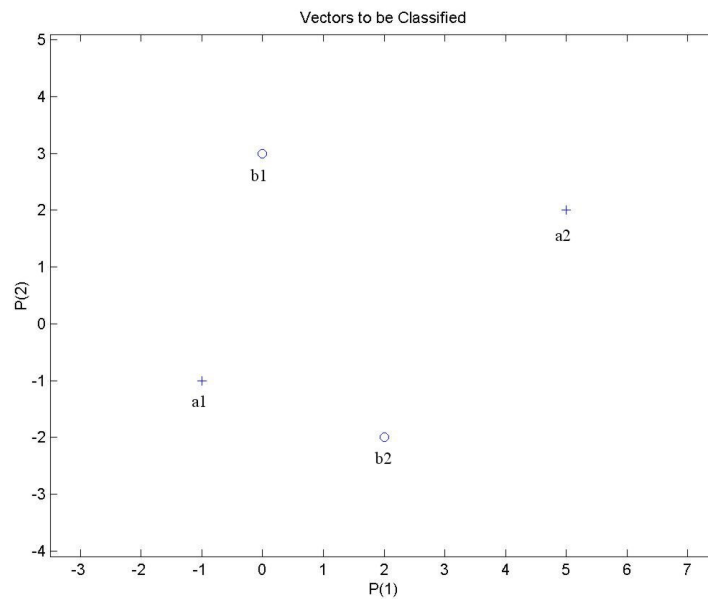
niemożliwe do realizacji

c) $A=\{a_1,a_2\},B=\{b_1,b_2,b_3\}$ gdzie $a_1=\{-0.5,-0.5\}$ $a_2=\{-0.5,0.5\}$ $b_1=\{0.3,-0.5\}$ $b_2=\{-0.1,1\}$
 $b_3=\{-0.9,0\}$



niemożliwe do realizacji

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



niemożliwe do realizacji

Zadanie 2

Dokonać uczenia perceptronu dla problemów z zadania 1 za pomocą funkcji **adapt** oraz **train**, porównać wyniki.

Uczenie perceptronu za pomocą funkcji **adapt** przeprowadziliśmy wykorzystując poniższy skrypt (TOOLBOX NEUTRAL NETWORKS).

```
p=[-0.5,0.5,-1,1,-0.5;-0.5,-1,1,2,2.5] %punkty wejsciowe
t=[1 1 1 0 0] %dane wzorcowe
net=newp([-1 1;-1 1],1); %tworzenie sieci składającej się z jednego neuronu i dwóch wejść
E=1; %roznica pomiedzy oczekiwana odpowiedzia neuronu a rzeczywista
while(sse(E)) %nauka perceptronu odbywa sie petli
    [net,Y,E]=adapt(net,p,t); %nuaka perceptronu
end;
Y %rzeczywista odpowiedz neuronu na wzorzec uczacy
E %roznica pomiedzy oczekiwana odpowiedzia neuronu a rzeczywista
```

Uczenie perceptronu za pomocą funkcji **train** przeprowadziliśmy wykorzystując poniższy skrypt (TOOLBOX NEUTRAL NETWORKS).

```
p=[-0.5,0.5,-1,1,-0.5;-0.5,-1,1,2,2.5] %punkty wejsciowe
t=[1 1 1 0 0] %dane wzorcowe
net=newp([-1 1;-1 1],1); %tworzenie sieci składającej się z jednego neuronu i dwóch wejść
E=1; %roznica pomiedzy oczekiwana odpowiedzia neuronu a rzeczywista
while(sse(E)) %nauka perceptronu odbywa sie petli
    [net,a,Y,E]=train(net,p,t); %nuaka perceptronu
end;
Y %rzeczywista odpowiedz neuronu na wzorzec uczacy
E %roznica pomiedzy oczekiwana odpowiedzia neuronu a rzeczywista
```

	t	adapt				train			
		Y	E	Y	E				
Przykład a	1 1 1 0 0	1 1 1 0 0	0 0 0 0 0	1 1 1 0 0	0 0 0 0 0				
Przykład b	1 1 0 0	Ucząc perceptron przy użyciu tej metody, nie jest możliwe określenie wyniku E i Y, ponieważ nigdy nie zostanie spełniony warunek pętli while.							
Przykład c	1 1 0 0 0								
Przykład d	1 1 0 0								

Ilość iteracji dla przykładu a:

Train :1 iteracja Adapt: 4 iteracje

Zadanie 3

Dane są dwa zbiory $A=\{a_1,a_2,a_3\}$, $B=\{b_1,b_2\}$, gdzie $a_1=\{-0.5,-0.5\}$ $a_2=\{-0.5,0.5\}$ $a_3=\{-80,100\}$, $b_1=\{0.3,-0.5\}$ $b_2=\{-0.1,1\}$, przeprowadzić na ich podstawie uczenie perceptronu. Jakie wnioski nasuwają się po analizie procesu uczenia.

Nauka perceptronu zakończyła się pomyślnie. Czas trwania był jednak znacznie dłuższy (66 iteracji), niż w przypadku nauki perceptronu z punktu 2a (4 iteracje). Badanie przeprowadziliśmy przy użyciu skryptu z zadania 2.

Zadanie 4

Zbadać czy poniższe zadanie klasyfikacji możliwe jest do realizacji przy użyciu sieci dwóch perceptronów, jeżeli zadanie jest możliwe do realizacji wyznaczyć wagi w , dla których sieć będzie klasyfikował elementy czterech zbiorów A, B, C, D: $A=\{a_1, a_2, a_3\}$, $B=\{b_1, b_2\}$, $C=\{c_1, c_2, c_3\}$, $D=\{d_1, d_2\}$ gdzie $a_1=(1,12)$, $a_2=(7,18)$, $a_3=(8,16)$, $b_1=(8,6)$, $b_2=(10,8)$, $c_1=(3,5)$, $c_2=(0,2)$, $c_3=(-3,8)$, $d_1=(-5,-15)$, $d_2=(-15,-13)$.

Treść skryptu:

```
p=[1 7 8 8 10 3 0 -3 -5 -15;12 18 16 6 8 5 2 8 -15 -13]
t=[0 0 0 0 0 1 1 1 1 1; 0 0 0 1 1 0 0 0 1 1]
net=newp([-30 30;-30 30],2);
plotpv(p,t);
E=1;
i=0;
line=plotpc(net.IW{1},net.b{1})
while(sse(E))
    [net,Y,E]=adapt(net,p,t);
    line=plotpc(net.IW{1},net.b{1},line);
    drawnow;
    i=i+1;
end;
Y
E
i
```

Wynik działania programu (ostatnia iteracja):

Y =

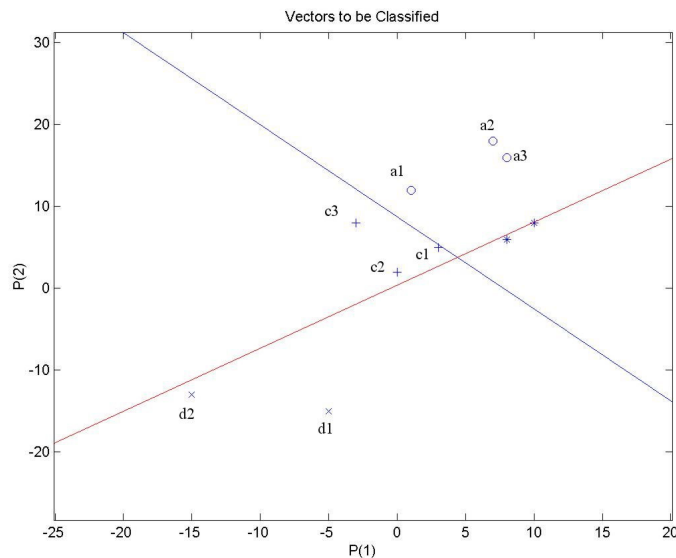
```
0 0 0 0 0 1 1 1 1 1
0 0 0 1 1 0 0 0 1 1
```

E =

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```

i =

105 (ilosc iteracji)



Zadanie 5

Wyznaczyć 10 wzorców uczących liniowo separowanych, podzielić je na 4 klasy oraz:

- Utworzyć perceptron składający się z 2 neuronów i nauczyć go klasyfikować dane wzorce do poszczególnych klas
- Zbadać zdolności uogólniające perceptronu przez podawanie sygnałów zbliżonych do sygnałów uczących

a)

A, B, C, D:

$A = \{a1, a2, a3\}$, $B = \{b1, b2\}$, $C = \{c1, c2, c3\}$, $D = \{d1, d2\}$

gdzie $a1 = (2.5, 7)$, $a2 = (3.5, 5)$, $a3 = (4, 4.5)$, $b1 = (7, 4)$, $b2 = (8, 2)$, $c1 = (3.5, 2.5)$, $c2 = (1.5, 1)$, $c3 = (0, 4)$, $d1 = (-1, -4.5)$, $d2 = (-1.5, -1.5)$.

Naukę perceptronu przeprowadziliśmy wykorzystując skrypt z [zadania 4](#)

Wynik działania programu (ostatnia iteracja):

Y =

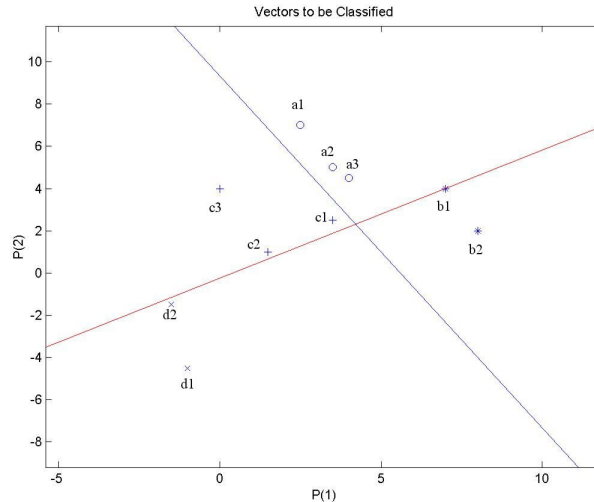
0	0	0	0	0	1	1	1	1	1
0	0	0	1	1	0	0	0	1	1

E =

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

i =

27 (ilosc iteracji)



b) Badanie przeprowadziliśmy dla punktów:

```
point={ [5; 6] [6;-4] [1 ;4] [10;-2] }
```

```
r=sim(net,point) %sprawdzenie
```

```
r= 0 1 1 0
    0 1 0 1
```

Wnioski:

Celem tego ćwiczenia było przeprowadzenie badań sieci neuronowych. Badane sieci zbudowane były z jednego lub dwóch perceptronów.

W zadaniu pierwszym sprawdzaliśmy czy zadania klasyfikacji wzorców możliwe są do realizacji przy użyciu perceptronu. Do sprawdzenia użyliśmy skryptu ze specjalistycznymi funkcjami do badania sieci neuronowych (plotpc, plotpv,sse). Zadanie możliwe było do realizacji, jeśli dwa zbiory punktów można było oddzielić linią prostą. Tylko w przypadku a) zadanie okazało się wykonalne. W pozostałych przypadkach zadanie, aby poprawnie dokonać klasyfikacji wzorców należałoby użyć jeszcze jednego perceptronu, tak jak to miało miejsce w zadaniu 4.

W zadaniu drugim należało przeprowadzić naukę perceptronu dla problemów z zadania pierwszego za pomocą funkcji adapt oraz train, a następnie porównać efektywność tych dwóch metod. W naszym przypadku naukę perceptronu można było przeprowadzić tylko dla podpunktu a, bowiem w pozostałych przypadkach program działałby w nieskończoność, nigdy bowiem błąd perceptronu nie będzie równy 0, a taki był warunek zakończenia pętli wykorzystanej do nauki. Porównanie obu metod wypadło na korzyść train, bowiem pętla zakończyła swoje działanie już po jednej iteracji, podczas gdy funkcja adapt potrzebowała aż cztery iteracje. W rzeczywistości przewaga metody train nie jest tak znacząca, bowiem ne zawsze gwarantuje ona nauczenie perceptronu w skończonej liczbie iteracji.

W zadaniu trzecim dokonaliśmy uczenia perceptronu podobnie jak w zadaniu poprzednim. Tym razem jednak jeden punkt, podany jako wzorzec znajdował się w znacznej odległości od pozostałych co znacząco wydłużyło naukę perceptronu(66 iteracji), chociaż ilość punktów wzorcowych była taka sama jak w zadaniu 2.

W zadaniu 4 należało wykorzystać sieć zbudowaną z dwóch perceptronów, umożliwiło to bowiem podział punktów na cztery różne grupy i oddzielenie ich za pomocą dwóch prostych. Fakt ten znacznie wydłużył czas działania programu.

W zadaniu piątym, należało wykonać polecenie z zadania 4, tyle tylko, że dla własnych punktów. Ciekawym zjawiskiem był fakt, że dla następujących punktów: $a_1(1,2)$, $a_2(2,3)$, $a_3(2,15)$, $b_1(5,4)$, $b_2(7,2)$, $c_1(8,-2)$, $c_2(4,-5)$, $c_3(6,-7)$, $d_1(-4,-4)$, $d_2(-3,-3)$ program nie mógł znaleźć szukanych prostych, podczas gdy bez problemów zrobił to dla punktów opisanych w zadaniu 5. Świadczy to o niedoskonałości metody uczenia sieci perceptronów.