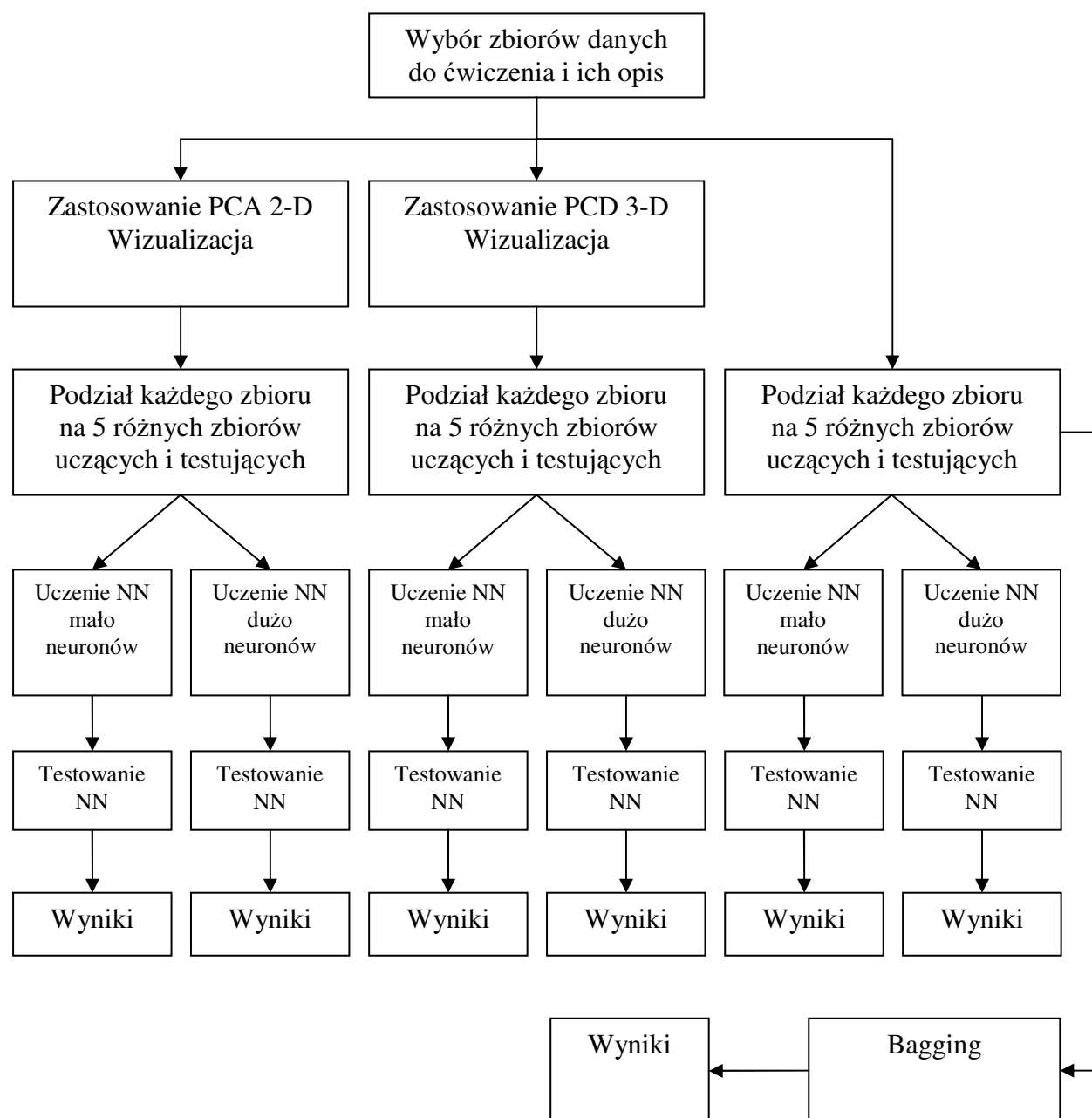


Ćwiczenie nr. 5

Celem ćwiczenia jest zaznajomienie się ze skutecznością klasyfikowania obiektów przy pomocy sztucznych sieci neuronowych a także implementacja i demonstracja działania klasyfikatora typu „bagging”.

Schemat realizacji ćwiczenia:



1. Wybór i opis analizowanych zbiorów

Do przeprowadzenia ćwiczenia wykorzystano przykładowe zbiór danych pochodzący ze strony <http://www.ics.uci.edu/~mlearn/MLRepository.html>

1.1. Zbiór wine.data

Plik ten zawiera dane opisujące 3 klasy:

- klasa: 1 ilość elementów: 59 33 %
- klasa: 2 ilość elementów: 71 40 %
- klasa: 3 ilość elementów: 48 27 %

Razem 178 próbek.

Dane opisane są na 13 numerycznych atrybutach, które zawierają chemiczną analizę winogron rosnących w tym samym rejonie Włoch ale dostarczonych przez różnych plantatorów.

1.2. Zbiór glass.data

Plik ten zawiera dane opisujące 7 klas:

- klasa: 1 ilość elementów: 70 33 %
- klasa: 2 ilość elementów: 76 35 %
- klasa: 3 ilość elementów: 17 8 %
- klasa: 4 ilość elementów: 0 0 %
- klasa: 5 ilość elementów: 13 6 %
- klasa: 6 ilość elementów: 9 4 %
- klasa: 7 ilość elementów: 29 14 %

Razem 214 próbek.

Dane opisane są na 9 numerycznych atrybutach, która zawierają informację o zawartości określonego składnika chemicznego w danym rodzaju szkła.

1.3. Zbiór iris.data

Plik ten zawiera dane opisujące 3 klasy:

- klasa 1 ilość elementów: 50 33 %
- klasa 2 ilość elementów: 50 33 %
- klasa 3 ilość elementów: 50 34 %

Razem 150 próbek.

Dane opisane są na 4 numerycznych atrybutach, które określają informację na temat wielkości liści i płatków pewnej rodziny kwiatów Iris.

2. Zastosowanie PCA 2-D oraz wizualizacja danych

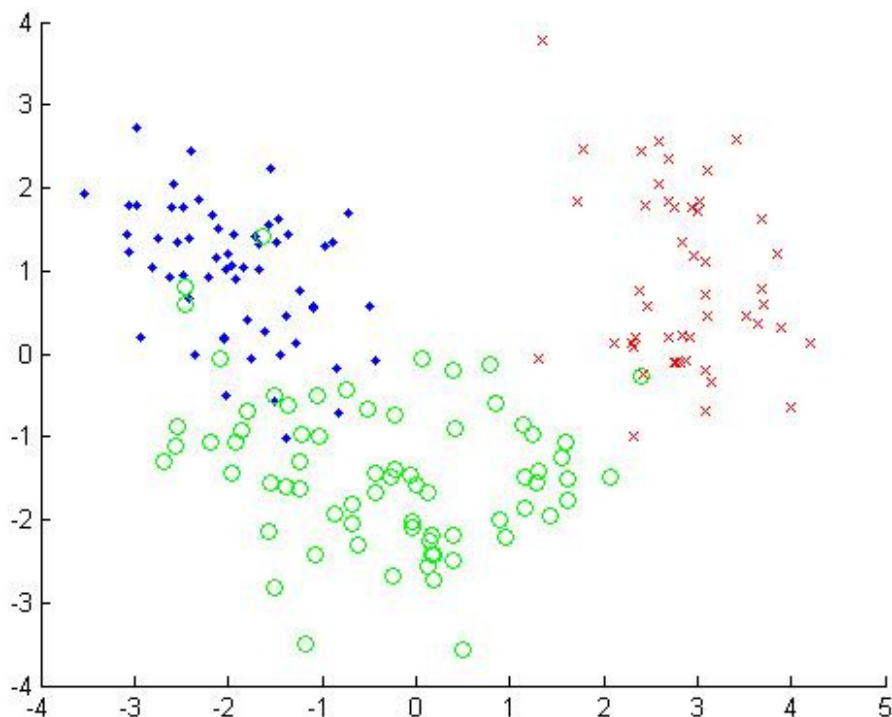
Dla każdego zbioru danych przeprowadzono operacje PCA 2-D oraz przeprowadzono wizualizację otrzymanych danych na wykresie. Zarówno PCA jak i wizualizację przeprowadzono na całym zbiorze danych, zawierającym zarówno dane uczące jak i dane testujące.

```
clear all;
D=load('iris.data'); %D=load('wine.data'); %D=load('glass.data');
klasy=D(:,1); %dane=D(:,2:13); %dane=D(:,2:9);
dane=D(:,2:4);

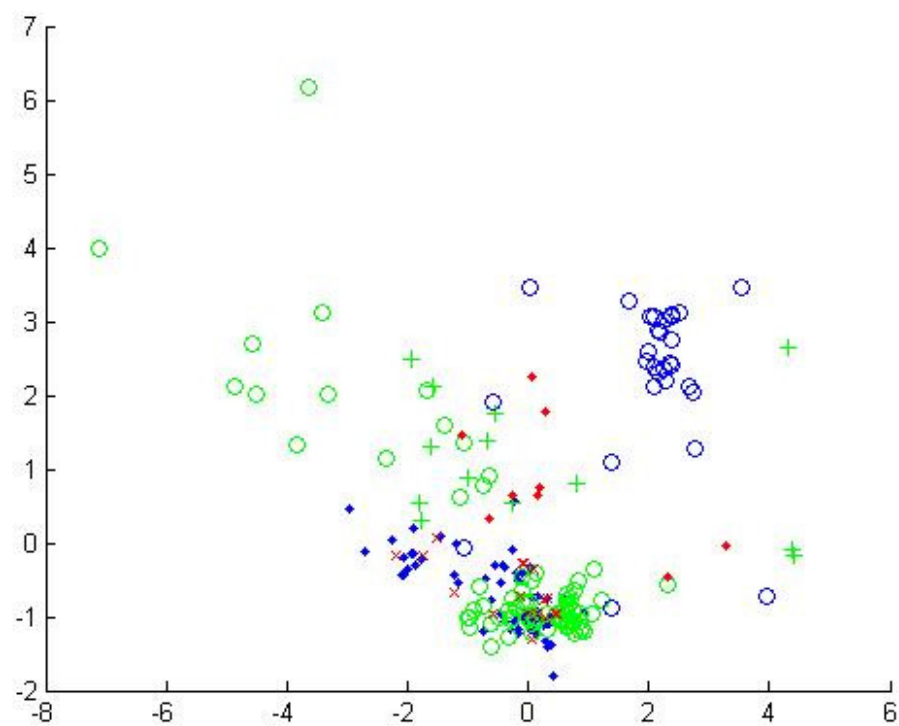
%normalizacja
[Ilosc, Rozmiar] = size(dane);
for h = 1:Rozmiar
    dane_znormalizowane(:,h) = (dane(:,h) - mean(dane(:,h)))/std(dane(:,h));
end

cov=pcacov(dane_znormalizowane);
dane_pca=dane_znormalizowane*cov;
dane2d=dane_pca(:,1:2);

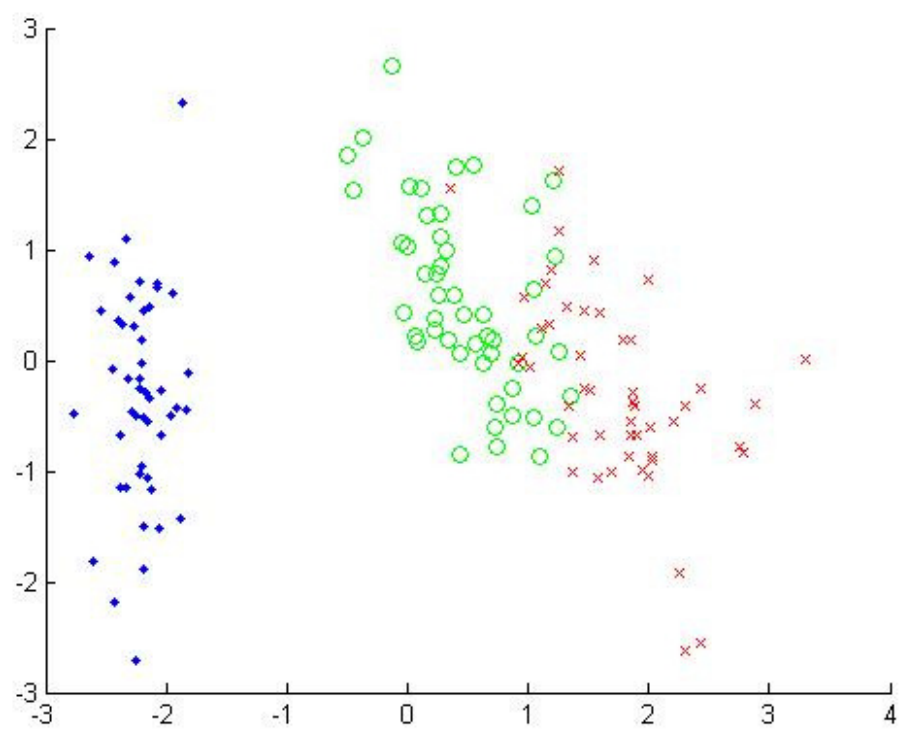
znaczniki = {'.b','og','xr','*c','om','sy','dk'};
hold all
for f = 1:Ilosc
    plot(dane2d(f,1),dane2d(f,2), znaczniki{klasy(f)});
end;
```



Rysunek 1 - wizualizacja wine.data



Rysunek 2 - wizualizacja glass.data



Rysunek 3 - wizualizacja iris.data

3. Zastosowanie PCA 3-D oraz wizualizacja danych

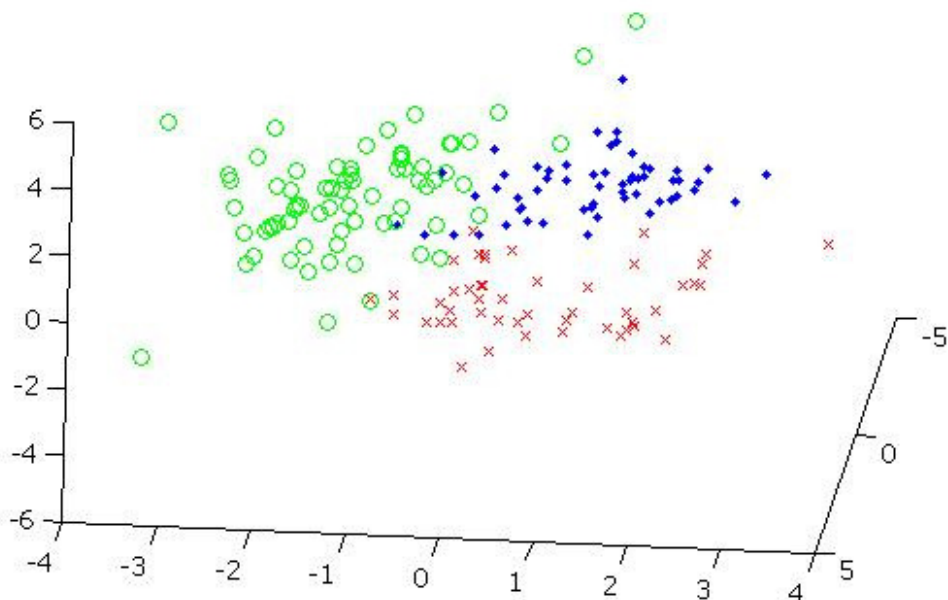
Dla każdego zbioru danych przeprowadzono operacje PCA 3-D oraz przeprowadzono wizualizację otrzymanych danych na wykresie. Zarówno PCA jak i wizualizację przeprowadzono na całym zbiorze danych, zawierającym zarówno dane uczące jak i dane testujące.

```
clear all;
D=load('iris.data'); %D=load('wine.data'); %D=load('glass.data');
klasy=D(:,1); %dane=D(:,2:13); %dane=D(:,2:9);
dane=D(:,2:4);

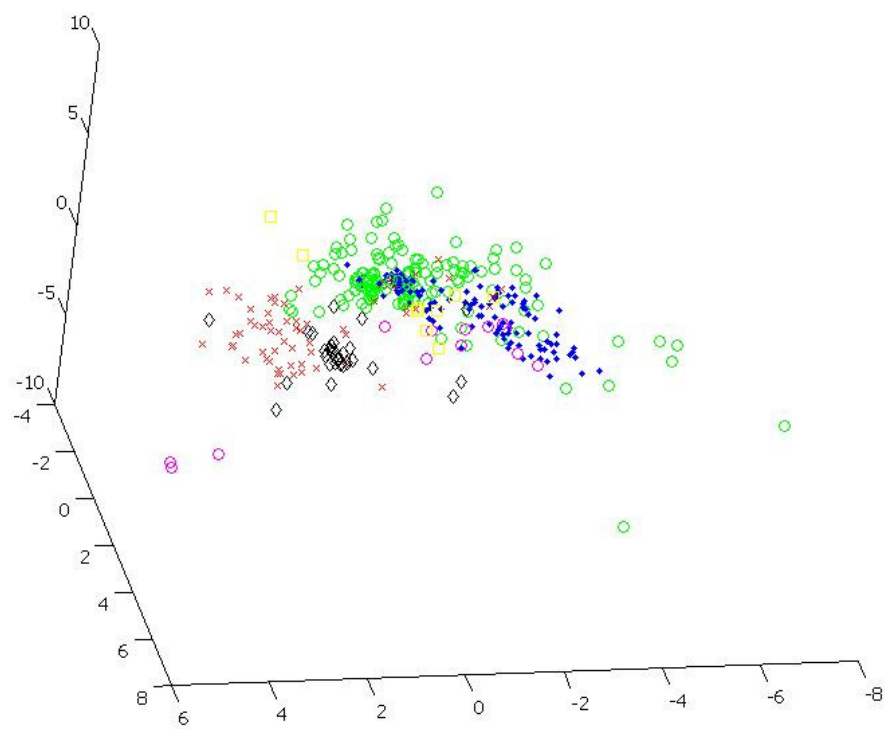
%normalizacja
[Ilosc, Rozmiar] = size(dane);
for h = 1:Rozmiar
    dane_znormalizowane(:,h) = (dane(:,h) - mean(dane(:,h)))/std(dane(:,h));
end

cov=pcacov(dane_znormalizowane);
dane_pca=dane_znormalizowane*cov;
dane2d=dane_pca(:,1:3);

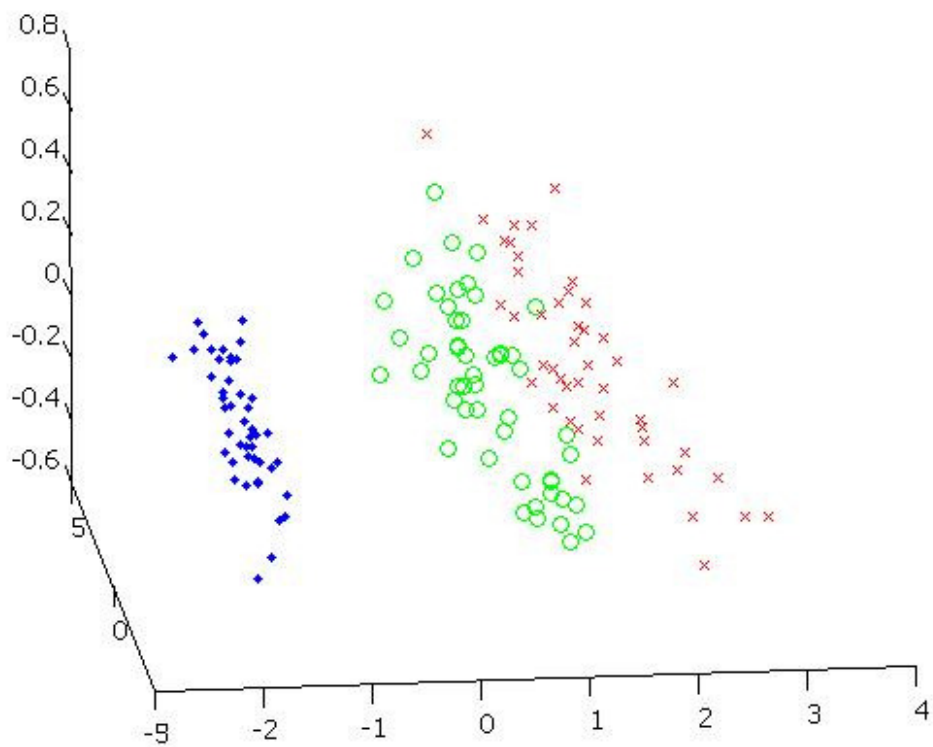
znaczniki = {'.b','og','xr','*c','om','sy','dk'};
hold all
for f = 1:Ilosc
    plot3(dane2d(f,1),dane2d(f,2),dane2d(f,3),znaczniki{klasy(f)});
end;
```



Rysunek 4 - wizualizacja wine.data



Rysunek 5 - wizualizacja glass.data



Rysunek 6 - wizualizacja iris.data

4. Podział plików na dane uczące oraz dane testujące

Pliki z danymi pięciokrotnie podzielono na zbiory uczące oraz zbiory testujące. Podczas podziału starano się zachować proporcje 85% danych uczących, 15% danych testujących. Podczas wybierania danych testujących starano się aby w kolejnych zestawach danych znajdowały się różne przypadki. Dodatkowo, do zbiorów testujących starano się wybrać trudniejsze przypadki, tzn. takie które znajdowały się na granicy klastrów. Podczas podziału posługowano się danymi uzyskanymi z metody kNN dla $k=9$.

4.1. Zbiór wine.data

Plik ten zawiera dane opisujące 3 klasy:

- | | | | |
|------------|---------------------|------------|---------------|
| • klasa: 1 | ilość elementów: 59 | uczące: 50 | testujące: 9 |
| • klasa: 2 | ilość elementów: 71 | uczące: 61 | testujące: 10 |
| • klasa: 3 | ilość elementów: 48 | uczące: 41 | testujące: 7 |

Razem 178 próbek podzielono na 5 różnych zbiorów zawierających:

- 152 próbki uczące
- 26 próbki testujące

4.2. Zbiór glass.data

Plik ten zawiera dane opisujące 7 klas:

- | | | | |
|------------|---------------------|------------|---------------|
| • klasa: 1 | ilość elementów: 70 | uczące: 60 | testujące: 10 |
| • klasa: 2 | ilość elementów: 76 | uczące: 65 | testujące: 11 |
| • klasa: 3 | ilość elementów: 17 | uczące: 15 | testujące: 2 |
| • klasa: 4 | ilość elementów: 0 | uczące: 0 | testujące: 0 |
| • klasa: 5 | ilość elementów: 13 | uczące: 11 | testujące: 2 |
| • klasa: 6 | ilość elementów: 9 | uczące: 8 | testujące: 1 |
| • klasa: 7 | ilość elementów: 29 | uczące: 25 | testujące: 4 |

Razem 214 próbek podzielono na 5 różnych zbiorów zawierających:

- 184 próbki uczące
- 29 próbki testujące

4.3. Zbiór iris.data

Plik ten zawiera dane opisujące 3 klasy:

- | | | | |
|-----------|---------------------|------------|--------------|
| • klasa 1 | ilość elementów: 50 | uczące: 42 | testujące: 8 |
| • klasa 2 | ilość elementów: 50 | uczące: 42 | testujące: 8 |
| • klasa 3 | ilość elementów: 50 | uczące: 42 | testujące: 8 |

Razem 214 próbek podzielono na 5 różnych zbiorów zawierających:

- 126 próbki uczące
- 24 próbki testujące

5. Opis zastosowanej sieci NN

Do realizacji niniejszego ćwiczenia wykorzystano oprogramowanie Matlab 7.1 oraz zaimplementowaną sieć neuronową typu LVQ.

Sieć neuronowa typu *LVQ* (ang. *Learning Vector Quantization*) stanowi wariant sieci klasyfikującej, opartej na zasadzie współzawodnictwa między neuronami. Warstwa ukryta (tzw. Warstwa Kohonena) dokonuje klasyfikacji prezentowanych sieci wektorów wejściowych, zaś warstwa wyjściowa składa się z tylu neuronów, ile klas podlega klasyfikacji. Warstwa Kohonena jest podzielona na grupy neuronów; każdej z grup przypisuje się jedną klasę wektorów wejściowych.

Wejściem klasyfikatora są zestawy danych wejściowych zawierające cechy danej próbki (w zależności od pliku zawierają różną ilość cech, dla PCA 2D są to 2 cechy, dla pełnego zbioru i pliku wine.data jest to 13 cech). Sieć uczona była w trybie nadzorowanym, czyli każdemu elementowi wejściowemu przypisano liczbę 1,2,...7 odpowiadającą przynależności do jednej z kilku grup.

Do treningu sieci wykorzystano dane przygotowane zgodnie z opisem w punkcie 4.

Poprawność klasyfikacji została zweryfikowana poprzez użycie zestawu danych testujących (przygotowanych zgodnie z opisem w punkcie 4.).

5.1. Konfiguracja sieci NN

Generalnie użyto standardowych parametrów sieci LVQ. Podczas inicjalizacji sieci za pomocą polecenia *newlvq* konfigurowano jedynie ilość neuronów oraz stopień prawdopodobieństwa przynależności do danej klasy. Stopień przynależności został obliczony i wskazany w punkcie 1. Ilość kroków uczących pozostawiono bez zmian, i wynosi ona 100.

Przykładowa inicjalizacja sieci neuronowej:

```
net = newlvq(minmax(P),10,[.33 .40 .27]);
```

5.2. Obserwacja działania sieci NN

Podczas procesu uczenia sieci generowany jest wykres postępów uczenia (przykładowe wykresy zostały umieszczone w kolejnych punktach opracowania).

Skrypt kończy się wyświetleniem wyniku dla symulacji przeprowadzonej na zbiorze testującym. Wynik zawiera informację o ilości elementów w zbiorze testującym, ilości elementów poprawnie rozpoznanych oraz procent poprawnie rozpoznanych danych. Przykładowy wynik działania skryptu wygląda następująco:

Wynik =
29.0000 15.0000 51.7241

6. Zastosowanie sieci NN na danych po PCA 2D

Dla każdego z przygotowanych par zbiorów (dane uczące, dane testujące) przeprowadzono PCA 2D następnie otrzymane dane posłużyły do uczenia sieci NN oraz do weryfikacji.

```
clear all;
u=load('wine_data_u1.txt');
t=load('wine_data_t1.txt');
%u=load('glass_data_u1.txt');
%t=load('glass_data_t1.txt');
%u=load('iris_data_u1.txt');
%t=load('iris_data_t1.txt');

[ile_u, r]=size(u);
[ile_t, r]=size(t);
c=cat(1,u,t);
klasy=c(:,1);
dane=c(:,2:r-1);

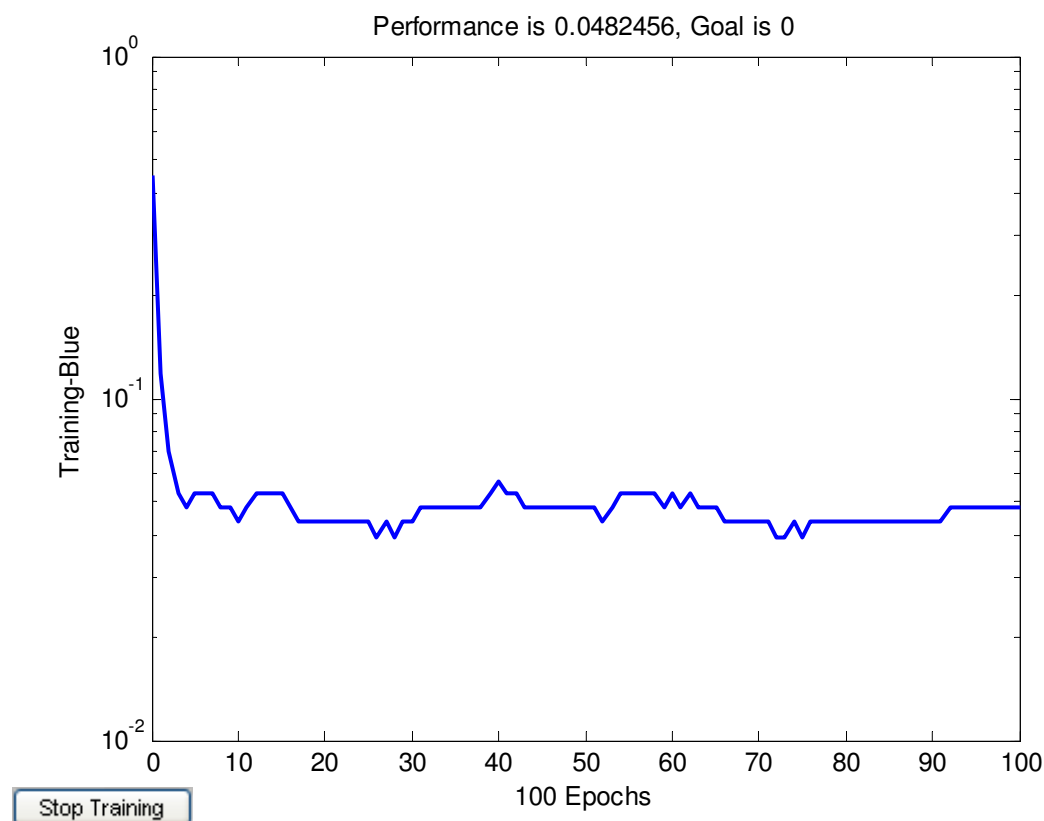
%normalizacja
[Ilosc, Rozmiar] = size(dane);
for h = 1:Rozmiar
    dane_znormalizowane(:,h) = (dane(:,h) - mean(dane(:,h)))/std(dane(:,h));
end

cov=pcacov(dane_znormalizowane);
dane_pca=dane_znormalizowane*cov;
dane=dane_pca(:,1:2); %dane=dane_pca(:,1:3);

dane_u=dane(1:ile_u,:);
dane_t=dane(ile_u+1:ile_u+ile_t,:);
klasy_u=klasy(1:ile_u,:);
klasy_t=klasy(ile_u+1:ile_u+ile_t,:);

P=dane_u;
Tc=klasy_u;
T = ind2vec(Tc);
P=shiftdim(P,1);
net = newlvq(minmax(P),10,[.33 .40 .27]); %wine
%net = newlvq(minmax(P),10,[.33 .35 .08 .00 .06 .04 .14]); %glass
%net = newlvq(minmax(P),10,[.33 .33 .34]); %iris
net = train(net,P,T);

PT=dane_t;
TTc=klasy_t;
PT=shiftdim(PT,1);
Y = sim(net,PT);
Yc = vec2ind(Y);
Yc=shiftdim(Yc,1);
W=cat(2,TTc,Yc);
c=0;
for h = 1:ile_t
    if (W(h,1)==W(h,2))
        c=c+1;
    end;
end
p=c/ile_t*100;
Wynik=[ile_t c p];
Wynik
```



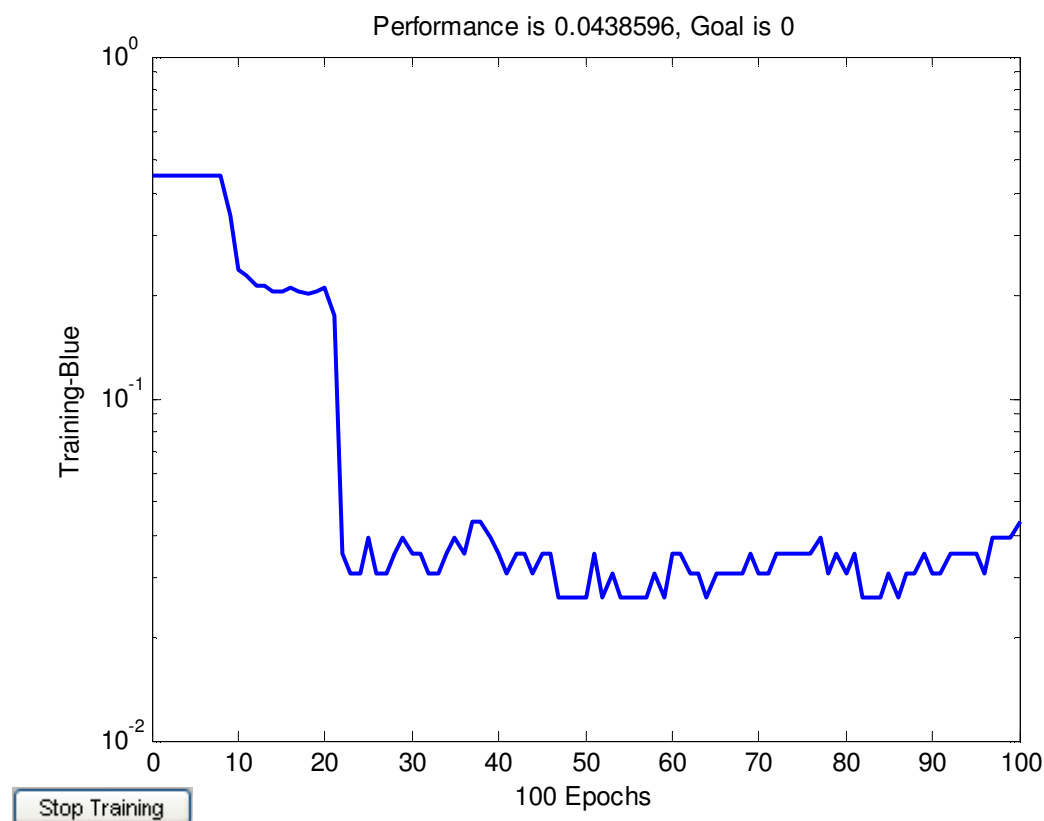
Rysunek 7 - przykładowy wykres uczenia się sieci NN dla 10 neuronów

Tabela 1 - tabela wyników z przeprowadzonych eksperymentów NN dla PCA 2D

Dane			Próbka										Wyniki			
Plik	Zbiór testowy	Ilość neuronów	1		2		3		4		5		Średnia		Wariancja	
wine	26	10	23,0	88,5%	24,0	92,3%	25,0	96,2%	24,0	92,3%	25,0	96,2%	24,2	93,1%	0,7	0,1%
		100	24,0	92,3%	24,0	92,3%	25,0	96,2%	24,0	92,3%	25,0	96,2%	24,4	93,8%	0,3	0,0%
		1000	23,0	88,5%	24,0	92,3%	25,0	96,2%	25,0	96,2%	25,0	96,2%	24,4	93,8%	0,8	0,1%
		10000	17,0	65,4%	18,0	69,2%	18,0	69,2%	18,0	69,2%	17,0	65,4%	17,6	67,7%	0,3	0,0%
		Średnia	21,8	83,7%	22,5	86,5%	23,3	89,4%	22,8	87,5%	23,0	88,5%	22,7	87,1%	0,3	0,0%
glass	29	10	19,0	65,5%	5,0	17,2%	15,0	51,7%	19,0	65,5%	13,0	44,8%	14,2	49,0%	33,2	3,9%
		100	19,0	65,5%	16,0	55,2%	16,0	55,2%	16,0	55,2%	13,0	44,8%	16,0	55,2%	4,5	0,5%
		1000	19,0	65,5%	15,0	51,7%	14,0	48,3%	15,0	51,7%	13,0	44,8%	15,2	52,4%	5,2	0,6%
		10000	15,0	51,7%	12,0	41,4%	12,0	41,4%	12,0	41,4%	12,0	41,4%	12,6	43,4%	1,8	0,2%
		Średnia	18,0	62,1%	12,0	41,4%	14,3	49,1%	15,5	53,4%	12,8	44,0%	14,5	50,0%	5,7	0,7%
iris	24	10	18,0	75,0%	19,0	79,2%	19,0	79,2%	20,0	83,3%	20,0	83,3%	19,2	80,0%	0,7	0,1%
		100	17,0	70,8%	21,0	87,5%	20,0	83,3%	19,0	79,2%	18,0	75,0%	19,0	79,2%	2,5	0,4%
		1000	17,0	70,8%	19,0	79,2%	18,0	75,0%	19,0	79,2%	20,0	83,3%	18,6	77,5%	1,3	0,2%
		10000	16,0	66,7%	14,0	58,3%	16,0	66,7%	16,0	66,7%	14,0	58,3%	15,2	63,3%	1,2	0,2%
		Średnia	17,0	70,8%	18,3	76,0%	18,3	76,0%	18,5	77,1%	18,0	75,0%	18,0	75,0%	0,3	0,1%

7. Zastosowanie sieci NN na danych po PCA 3D

Dla każdego z przygotowanych par zbiorów (dane uczące, dane testujące) przeprowadzono PCA 2D następnie otrzymane dane posłużyły do uczenie sieci NN oraz do weryfikacji.



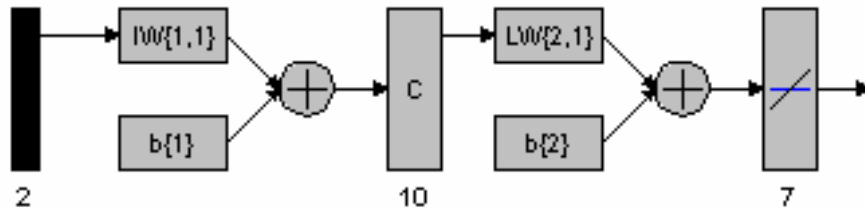
Rysunek 8 - przykładowy wykres uczenia się sieci NN dla 1000 neuronów

Tabela 2 - tabela wyników z przeprowadzonych eksperymentów NN dla PCA 3D

Dane			Próbka										Wyniki			
Plik	Zbiór testowy	Ilość neuronów	1		2		3		4		5		Średnia		Wariancja	
wine	26	10	20,0	76,9%	26,0	100,0%	24,0	92,3%	22,0	84,6%	25,0	96,2%	23,4	90,0%	5,8	0,9%
		100	22,0	84,6%	26,0	100,0%	24,0	92,3%	24,0	92,3%	25,0	96,2%	24,2	93,1%	2,2	0,3%
		1000	23,0	88,5%	25,0	96,2%	24,0	92,3%	23,0	88,5%	24,0	92,3%	23,8	91,5%	0,7	0,1%
		10000	16,0	61,5%	16,0	61,5%	17,0	65,4%	18,0	69,2%	17,0	65,4%	16,8	64,6%	0,7	0,1%
Średnia		20,3	77,9%	23,3	89,4%	22,3	85,6%	21,8	83,7%	22,8	87,5%	22,1	84,8%	1,3	0,2%	
glass	29	10	20,0	69,0%	16,0	55,2%	16,0	55,2%	18,0	62,1%	12,0	41,4%	16,4	56,6%	8,8	1,0%
		100	19,0	65,5%	15,0	51,7%	17,0	58,6%	17,0	58,6%	13,0	44,8%	16,2	55,9%	5,2	0,6%
		1000	20,0	69,0%	16,0	55,2%	15,0	51,7%	16,0	55,2%	12,0	41,4%	15,8	54,5%	8,2	1,0%
		10000	11,0	37,9%	15,0	51,7%	12,0	41,4%	16,0	55,2%	13,0	44,8%	13,4	46,2%	4,3	0,5%
Średnia		17,5	60,3%	15,5	53,4%	15,0	51,7%	16,8	57,8%	12,5	43,1%	15,5	53,3%	3,7	0,4%	
iris	24	10	18,0	75,0%	20,0	83,3%	20,0	83,3%	19,0	79,2%	20,0	83,3%	19,4	80,8%	0,8	0,1%
		100	19,0	79,2%	23,0	95,8%	20,0	83,3%	20,0	83,3%	18,0	75,0%	20,0	83,3%	3,5	0,6%
		1000	21,0	87,5%	20,0	83,3%	21,0	87,5%	20,0	83,3%	17,0	70,8%	19,8	82,5%	2,7	0,5%
		10000	16,0	66,7%	14,0	58,3%	16,0	66,7%	16,0	66,7%	14,0	58,3%	15,2	63,3%	1,2	0,2%
Średnia		18,5	0,0%	19,3	80,2%	19,3	80,2%	18,8	78,1%	17,3	71,9%	18,6	77,5%	0,7	12,2%	

8. Zastosowanie sieci NN na danych bez użycia PCA

Dla każdego z przygotowanych par zbiorów (dane uczące, dane testujące) przeprowadzono uczenie sieci NN (na zbiorach uczących) oraz do weryfikację sieci (na zbiorach testujących).



Rysunek 9 - wizualizacja sieci LVQ dla 7 klas i 10 neuronów

```
clear all;
u=load('wine_data_u1.txt');
t=load('wine_data_t1.txt');

[ile_u, r]=size(u);
[ile_t, r]=size(t);

P=u(:,2:r);
Tc=u(:,1);

T = ind2vec(Tc);
P=shiftdim(P,1);
net = newlvq(minmax(P),10,[.33 .40 .27]); %wine
%net = newlvq(minmax(P),1000,[.33 .35 .08 .00 .06 .04 .14]); %glass
%net = newlvq(minmax(P),1000,[.33 .33 .34]); %iris
net = train(net,P,T);

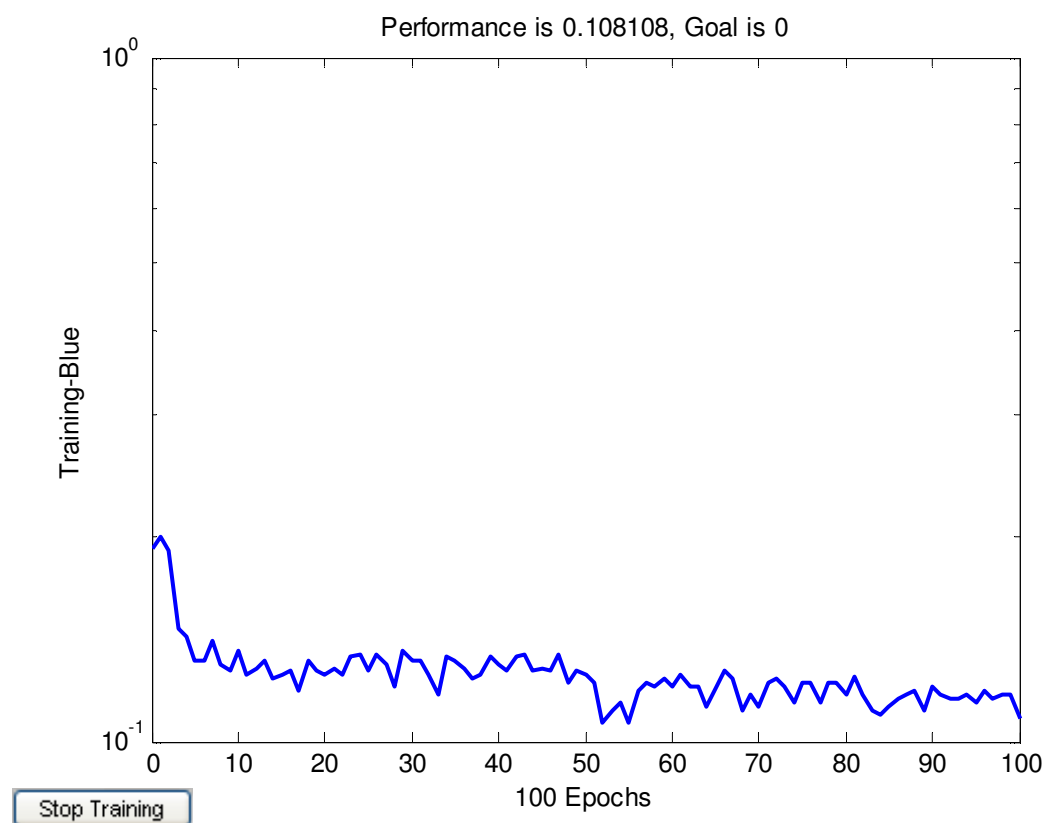
PT=t(:,2:r);
TTc=t(:,1);

PT=shiftdim(PT,1);

Y = sim(net,PT);
Yc = vec2ind(Y);

Yc=shiftdim(Yc,1);

W=cat(2,TTc,Yc);
c=0;
for h = 1:ile_t
    if (W(h,1)==W(h,2))
        c=c+1;
    end;
end
p=c/ile_t*100;
Wynik=[ile_t c p];
Wynik
```



Rysunek 10 - przykładowy wykres uczenia się sieci NN dla 100 neuronów

Tabela 3 - tabela wyników z przeprowadzonych eksperymentów dla NN bez PCA

Dane			Próbka										Wyniki			
Plik	Zbiór testowy	Ilość neuronów	1		2		3		4		5		Średnia		Wariancja	
wine	26	10	18,0	69,2%	19,0	73,1%	16,0	61,5%	18,0	69,2%	17,0	65,4%	17,6	67,7%	1,3	0,2%
		100	18,0	69,2%	16,0	61,5%	16,0	61,5%	16,0	61,5%	15,0	57,7%	16,2	62,3%	1,2	0,2%
		1000	18,0	69,2%	17,0	65,4%	16,0	61,5%	17,0	65,4%	17,0	65,4%	17,0	65,4%	0,5	0,1%
		10000	16,0	61,5%	16,0	61,5%	16,0	61,5%	17,0	65,4%	17,0	65,4%	16,4	63,1%	0,3	0,0%
		Średnia	17,5	67,3%	17,0	65,4%	16,0	61,54%	17,0	65,4%	16,5	63,5%	16,8	64,6%	0,3	0,0%
glass	29	10	18,0	62,1%	14,0	48,3%	14,0	48,3%	17,0	58,6%	16,0	55,2%	15,8	54,5%	3,2	0,4%
		100	21,0	72,4%	15,0	51,7%	14,0	48,3%	17,0	58,6%	16,0	55,2%	16,6	57,2%	7,3	0,9%
		1000	22,0	75,9%	16,0	55,2%	14,0	48,3%	18,0	62,1%	16,0	55,2%	17,2	59,3%	9,2	1,1%
		10000	12,0	41,4%	11,0	37,9%	10,0	34,5%	15,0	51,7%	12,0	41,4%	12,0	41,4%	3,5	0,4%
		Średnia	18,3	62,9%	14,0	48,3%	13,0	44,8%	16,8	57,8%	15,0	51,7%	15,4	53,1%	4,5	0,5%
iris	24	10	22,0	91,7%	22,0	91,7%	23,0	95,8%	21,0	87,5%	21,0	87,5%	21,8	90,8%	0,7	0,1%
		100	22,0	91,7%	21,0	87,5%	23,0	95,8%	21,0	87,5%	21,0	87,5%	21,6	90,0%	0,8	0,1%
		1000	23,0	95,8%	21,0	87,5%	23,0	95,8%	21,0	87,5%	22,0	91,7%	22,0	91,7%	1,0	0,2%
		10000	16,0	66,7%	16,0	66,7%	16,0	66,7%	16,0	66,7%	16,0	66,7%	16,0	66,7%	0,0	0,0%
		Średnia	20,8	86,5%	20,0	83,3%	21,3	88,5%	19,8	82,3%	20,0	83,3%	20,4	84,8%	0,4	0,1%

9. Zastosowanie metody „Bagging”

Bagging (Bootstrap Aggregating) to metoda umożliwiająca polepszenie działania klasyfikatora poprzez ogólnie mówiąc zwielokrotnienie użytych predyktorów.

Zasada działania metody „Bagging”:

- metodą bootstrap generujemy m różnych zbiorów (próbek) wybierając k pozycji z k-elementowego zbioru danych treningowych z zastępowaniem – część pozycji może pojawić się więcej niż raz, innych może nie być w ogóle
- trenujemy każdy model na innej takiej próbce zbioru treningowego (imitacja uczenia na różnych zbiorach treningowych)
- wyjście z systemu otrzymujemy przez proste uśrednienie lub głosowanie

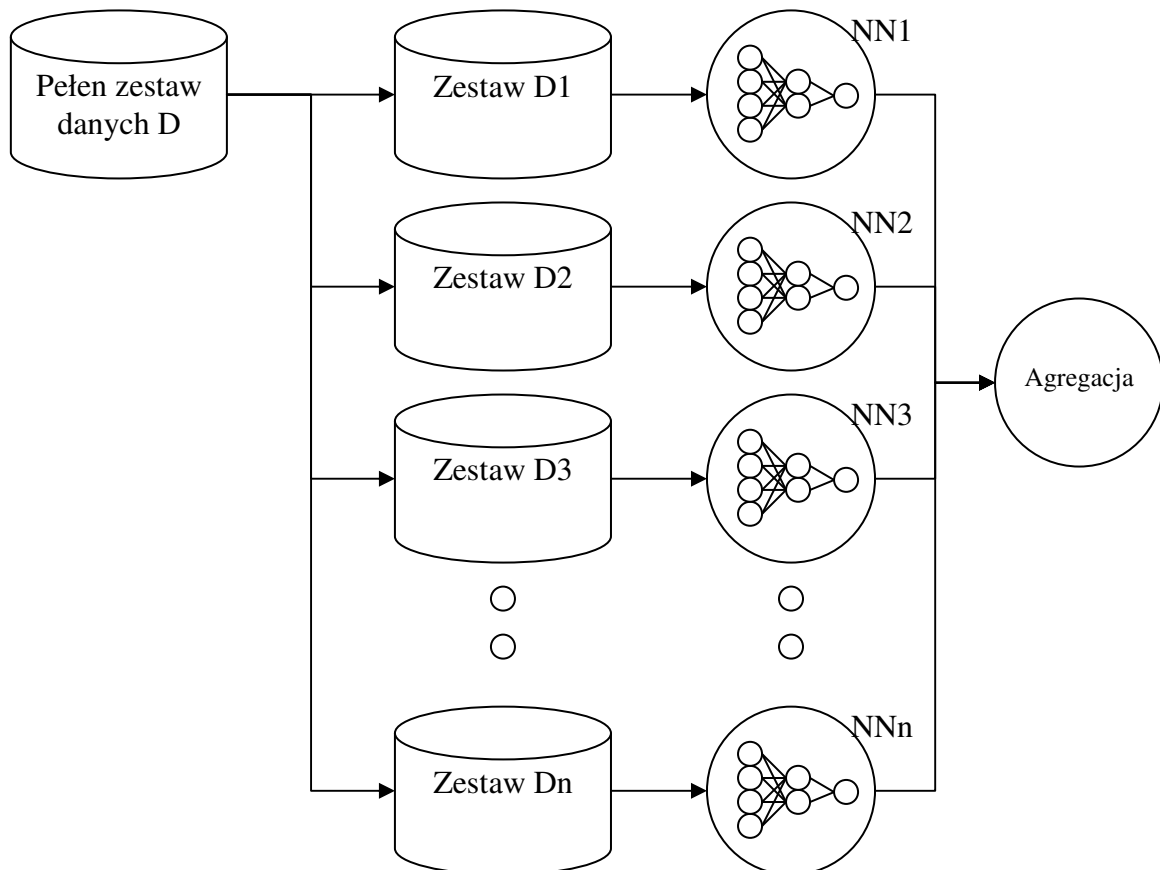


Tabela 4 - przykładowy zestaw danych oraz wylosowane zestawy uczące

D	1	2	3	4	5	6	7	8	9
D1	4	3	1	3	1	6	8	4	7
D2	3	9	9	7	3	5	1	3	2
D3	5	5	3	2	9	8	6	4	7
Dn	3	1	2	4	7	4	5	8	7

9.1 Implementacja generowania podzbiorów uczących

Do prawidłowego działania klasyfikatora „bagging” niezbędne było opracowanie metody umożliwiającej generowanie podzbiorów uczących na podstawie zadanego zbioru uczącego. Zadanie to realizuje poniżej przedstawiona funkcja:

```
function [P, Tc]=podzial_pliku(nazwa_pliku, prog)
u=load(nazwa_pliku);

[ile_u, r]=size(u);

P=[];
Tc=[];

c=0;
while c<ile_u
    for h = 1:ile_u
        if rand < prog
            P=cat(1,P,u(h,2:r));
            Tc=cat(1,Tc,u(h,1));
            c=c+1;
            if c==ile_u
                break;
            end
        end
    end
end
end
```

Parametrem wywołania funkcji jest nazwa pliku zawierającego dane uczące oraz prawdopodobieństwo wyboru próbki. Główna pętla programu przeprowadza iteracje tworzenia podzbioru do momentu, kiedy utworzony nowy podzbiór uczący będzie zawierał taką samą ilość elementów jak wejściowy zbiór uczący. Pętla wewnętrzna kilkakrotnie iteruje wszystkie elementy zbioru uczącego, dla każdego elementu sprawdzane jest prawdopodobieństwo czy element ten zostanie dodany czy też nie dodany do tworzonego podzbioru testowego. W ten sposób w nowo utworzonym podzbiorze testowym niektóre elementy mogą się powtórzyć, niektóre zaś nie będą występować.

9.2 Implementacja funkcji „bagging”.

Implementacja polega na wygenerowaniu pewnej ilości podzbiorów uczących, które zostają wykorzystane do procesu nauczania takiej samej ilości sieci neuronowych. Zadanie to realizowane jest w głównej pętli programu. Po przeprowadzeniu uczenia dla każdej z sieci neuronowych przeprowadzany jest procesy symulacji z wykorzystaniem danych testowych. Uzyskane wyniki zapisywane są do macierzy.

W ostatnim kroku działania algorytmu na podstawie danych zgromadzonych w macierzy wyników przeprowadzana jest agregacja otrzymanych wyników z poszczególnej sieci a następnie obliczany jest stopień poprawności klasyfikacji zbioru testowego.

Poniżej zamieszczono implementację w środowisku Matlab:

```
clear all;

plik_u='wine_data_u4.txt';
plik_t='wine_data_t4.txt';
ilosc_klas=3;

u=load(plik_u);
t=load(plik_t);
[ile_u, r]=size(u);
[ile_t, r]=size(t);

PT=t(:,2:r);
TTc=t(:,1);
PT=shiftdim(PT,1);

ilosc_probek=50;
YY=[];
for h = 1:ilosc_probek
    h
    [P,Tc]=podzial_pliku(plik_u, 0.2);
    T = ind2vec(Tc);
    P=shiftdim(P,1);
    net = newlvq(minmax(P),10,[.33 .40 .27]); %wine
    net.trainParam.epochs=100;
    net.trainParam.show=NaN;
    net = train(net,P,T);

    Y = sim(net,PT);
    Yc = vec2ind(Y);
    YY=cat(1,YY,Yc);
    Yc=shiftdim(Yc,1);
end

YY=shiftdim(YY,1);

B=zeros(ile_t,ilosc_klas);

for h=1:ile_t
    for k=1:ilosc_probek
        B(h,YY(h,k))=B(h,YY(h,k))+1;
    end
end
B=shiftdim(B,1);
[k i]=max(B);
i=shiftdim(i,1);

W=cat(2,TTc,i);
c=0;
for h = 1:ile_t
    if (W(h,1)==W(h,2))
        c=c+1;
    end;
end
p=c/ile_t*100;
Wynik=[ile_t c p];
Wynik
```


9.3 Wyniki z symulacji metodą „bagging”

Symulację przeprowadzono dla każdego z 5 różnych zestawów danych uczących testujących z każdego z 3 plików z danymi. Otrzymane wyniki zamieszczono w poniższej tabeli:

Tabela 5 - tabela wyników z przeprowadzonych eksperymentów dla metody "bagging"

Dane			Próbka										Wyniki			
Plik	Zbiór testowy	metoda	1		2		3		4		5		Średnia		Wariancja	
wine	26	bag	19,0	73,1%	18,0	69,2%	19,0	73,1%	18,0	69,2%	18,0	69,2%	18,4	70,8%	0,3	0,0%
glass	29	bag	20,0	69,0%	15,0	51,7%	14,0	48,3%	18,0	62,1%	18,0	62,1%	17,0	58,6%	6,0	0,7%
iris	24	bag	22,0	91,7%	21,0	87,5%	23,0	95,8%	22,0	91,7%	22,0	91,7%	22,0	91,7%	0,5	0,1%

10. Podsumowanie, wnioski

Łącznie przeprowadzono 195 symulacji. 180 ze standardowym wykorzystaniem sieci neuronowych oraz 15 symulacji z wykorzystaniem metody „bagging”. Otrzymane dane przedstawiono w poniższej zbiorczej tabeli:

Tabela 6 - zbiorcze zestawienie wyników z przeprowadzonych eksperymentów

Dane			Próbka										Wyniki			
Plik	Zbiór testowy	metoda	1		2		3		4		5		Średnia		Wariancja	
wine	26	2D	21,8	83,7%	22,5	86,5%	23,3	89,4%	22,8	87,5%	23,0	88,5%	22,7	87,1%	0,3	0,0%
		3D	20,3	77,9%	23,3	89,4%	22,3	85,6%	21,8	83,7%	22,8	87,5%	22,1	84,8%	1,3	0,2%
		NN	17,5	67,3%	17,0	65,4%	16,0	61,54%	17,0	65,4%	16,5	63,5%	16,8	64,6%	0,3	0,0%
		bag	19,0	73,1%	18,0	69,2%	19,0	73,1%	18,0	69,2%	18,0	69,2%	18,4	70,8%	0,3	0,0%
glass	29	2D	18,0	62,1%	12,0	41,4%	14,3	49,1%	15,5	53,4%	12,8	44,0%	14,5	50,0%	5,7	0,7%
		3D	17,5	60,3%	15,5	53,4%	15,0	51,72%	16,8	57,8%	12,5	43,1%	15,5	53,3%	3,7	0,4%
		NN	18,3	62,9%	14,0	48,3%	13,0	44,83%	16,8	57,8%	15,0	51,7%	15,4	53,1%	4,5	0,5%
		bag	20,0	69,0%	15,0	51,7%	14,0	48,3%	18,0	62,1%	18,0	62,1%	17,0	58,6%	6,0	0,7%
iris	24	2D	17,0	70,8%	18,3	76,0%	18,3	76,0%	18,5	77,1%	18,0	75,0%	18,0	75,0%	0,3	0,1%
		3D	18,5	0,0%	19,3	80,2%	19,3	80,21%	18,8	78,1%	17,3	71,9%	18,6	77,5%	0,7	12,2%
		NN	20,8	86,5%	20,0	83,3%	21,3	88,54%	19,8	82,3%	20,0	83,3%	20,4	84,8%	0,4	0,1%
		bag	22,0	91,7%	21,0	87,5%	23,0	95,8%	22,0	91,7%	22,0	91,7%	22,0	91,7%	0,5	0,1%

- Zbyt mała ilość neuronów powoduje wzrost błędu klasyfikacji (sieć niedouczona),
- Zbyt duża ilość neuronów powoduje istotny wzrost błędu klasyfikacji (sieć przeuczona),
- Dla danego zbioru danych powinna się znaleźć optymalną ilość neuronów, dająca najlepsze dopasowanie sieci do danych,
- Zastosowanie PCA przed użyciem klasyfikatora NN powoduje wzrost błędu klasyfikacji, przy czym PCA 2D generuje większy błąd niż PCA 3D,
- Zastosowanie metody „bagging” powoduje zmniejszenie błędu klasyfikacji,
- Metoda „bagging” wykorzystuje wiele sieci z małą ilością neuronów,
- Metoda „bagging” lepiej penetruje dane uczące.