|  |  |  |
| --- | --- | --- |
| aGrupa ćwicz. **2** | Data wykonania 06.12.2017 | Nr. Scenariusza  **5** |
| **Temat ćwiczenia:** Budowa i działanie sieci Kohonena dla WTA. | | |
| Imię i nazwisko  **Kamil Szczurkowski** | | Ocena i Uwagi |

**Cel ćwiczenia:**

Celem ćwiczenia jest poznanie budowy i działania sieci Kohonena przy wykorzystaniu reguły

WTA do odwzorowywania istotnych cech kwiatów.

**Wykonane zadania:**

1. Wygenerowano dane uczące, zawierające numeryczny opis cech kwiatków.
2. Wykorzystano narzędzie Matlab do stworzenia sieci (competlayer) oraz reguły WTA
3. Uczono sieć dla różnych współczynników uczenia.
4. Testowano poprawność działania sieci

**Specyfikacja wykorzystanych funkcji:**

**train** - Trening sieci neuronowej.

TRAIN Funkcja realizuje trening sieci neuronowej, wykorzystując funkcję treningu, której

nazwa została podana w polu NET.trainFcn, zaś wartości niezbędnych parametrów – w polu NET.trainParam. Funkcja treningu jest uniwersalna, wywoływana w jednolity sposób dla wszystkich typów sieci neuronowych, stąd też niektóre argumenty wejściowe nie mają znaczenia dla sieci jednokierunkowych (bez sprzężeń zwrotnych i opóźnień w torach sygnałów wejściowych)

*Wywołanie funkcji:* **[NET, TR, Y, E, Pf, Af] = TRAIN(NET, P, T, Pi, Ai)**

*lub:* **[NET, TR, Y, E, Pf, Af] = TRAIN(NET, P, T, Pi, Ai, VV, TV)**

**competlayer ( numClasses, kohonenLR, conscienceLR ) –**Warstwa konkurencyjna (współzawodnicząca)

Warstwy konkurencyjne uczą się klasyfikować wektory wejściowe do danej liczby klas, zgodnie z podobieństwem między wektorami, z preferencją dla jednakowej liczby wektorów na klasę.

**numClasses -** Liczba klas do sklasyfikowania danych wejściowych (domyślnie = 5)

**kohonenLR -** Wskaźnik uczenia się dla wag Kohonena (domyślnie = 0,01)

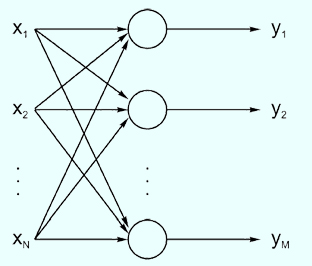
**conscienceLR -** Wskaźnik uczenia się dla błędu sumienia (domyślnie = 0,001)

zwraca warstwę konkurencyjną z neuronami numClasses.

**configure(net,x) -** Skonfiguruje wejścia i wyjścia sieciowe, tak aby jak najlepiej dopasować dane wejściowe i docelowe. konfiguruje tylko wejścia.

Uczenie rywalizacyjne (*competitive learning*) jest również metodą uczenia sieci samoorganizujących. Neurony na początku mają przypadkowe wartości wag. Następnie, na podstawie prezentowanych danych, „uczą się” rozpoznawać te dane i zbliżają się odpowiednio do obszarów zajmowanych przez te dane. Po każdej prezentacji wzorca x zwycięzcą zostaje tylko jeden neuron, najbliższy prezentowanemu wzorcowi.

Neurony rywalizują między sobą, a „neuron wygrywając” jest jedynym pobudzanym przy konkretnej obserwacji wejściowej.



Rysunek 1- Siec Kohonena

Grupa współzawodniczących neuronów otrzymuje te same sygnały wejściowe *x*.  
W zależności od aktualnych wartości wag, sygnały wyjścioweróżnią się między sobą. Wartości tych sygnałów są porównywane i zwycięża ten neuron, którego wartość jest największa. Neuron „zwycięzca” przyjmuje na swoim wyjściu stan 1, a pozostałe neurony (neurony „przegrywające”) stan 0. Wagi neuronu „*zwycięzcy*” są dostrajane w danym kroku uczenia. Taka metoda uczenia jest przykładem tzw. nauki z rywalizacją (*competitive learning*) bez nauczyciela.

Zasada **WTA** - tylko neuron zwycięski (oznaczany indeksem *c*, od *conqueror*) uaktualnia swe wagi, tzn. zbliża się do wektora *x*. Wagi zwycięskiego neuronu mogą zostać uaktualnione na podstawie wzoru Kohonena:



Rysunek 2 - wzór Kohonena

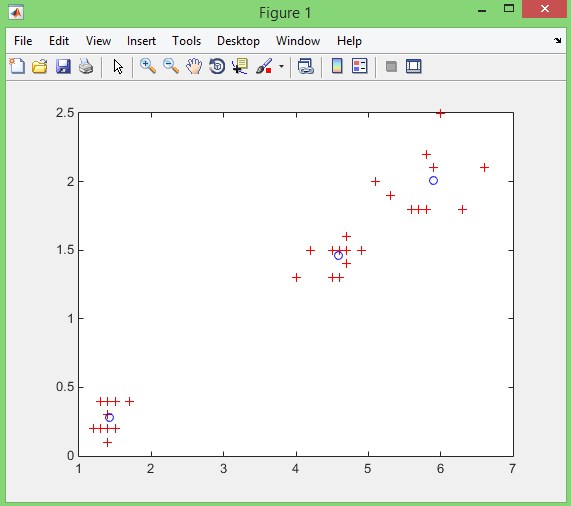
**Wykonanie zadania:**

Do wykonania zadania użyłem zastawu danych zawierające numeryczny opis cech kwiatków.

**Wyniki:**

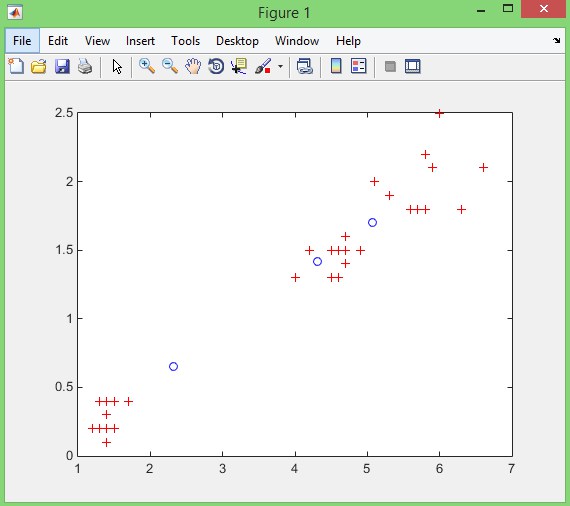
Przeprowadzenie kilku pomiarów dla rożnego wskaźnik uczenia się wag Kohonena oraz stałej liczby iteracji = 10:

Wskaźnik uczenia się wag Kohonena = 0.1



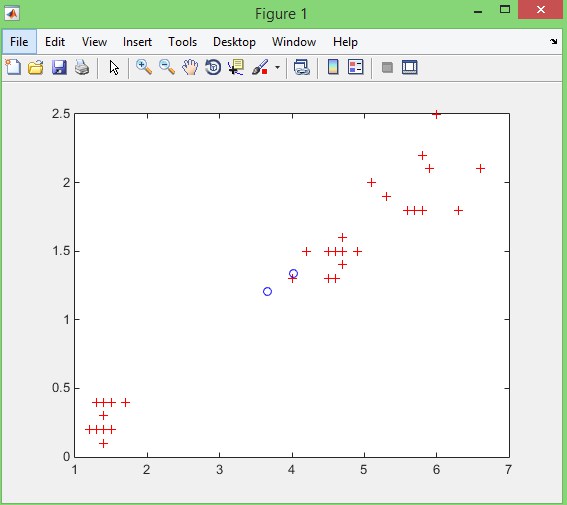
Rysunek 3 - Otrzymany wynik

Wskaźnik uczenia się wag Kohonena = 0.01



Rysunek 4- Otrzymany wynik

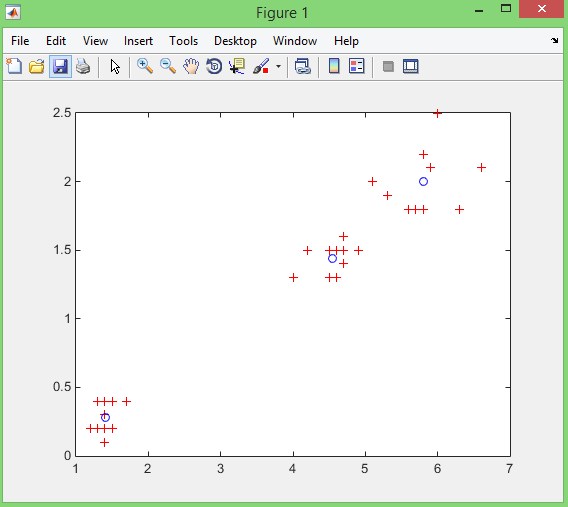
Wskaźnik uczenia się wag Kohonena = 0.001



Rysunek 5- Otrzymany wynik

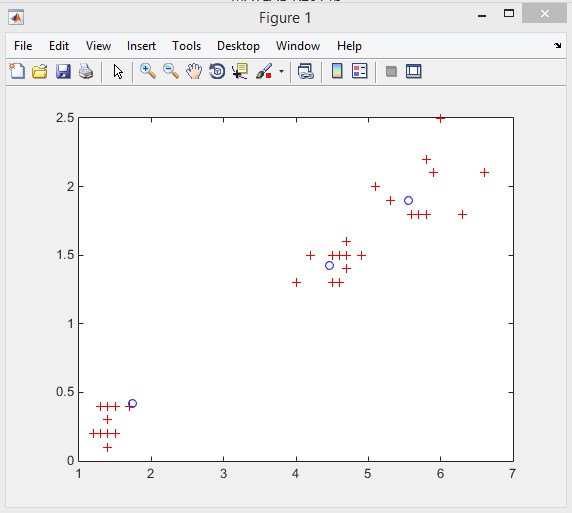
Przeprowadzenie kilku pomiarów dla rożnego wskaźnik uczenia się wag Kohonena oraz różnej liczby iteracji:

Wskaźnik uczenia się wag Kohonena = 0.01 ilość iteracji = 100



Rysunek 6- Otrzymany wynik

Wskaźnik uczenia się wag Kohonena = 0.01 ilość iteracji = 20



Rysunek 7- Otrzymany wynik

**Analiza:**

Cały program został zbudowany na competlayer. Jest to sieć rywalizująca która ma w sobie wskaźnik wag Kohonena oraz działa jako WTA – zwycięzca bierze wszystko. Na początku została stworzona tablica zawierająca numeryczny opis cech kwiatków które później grupowaliśmy według odpowiednich cech mianowicie długości sepii, szerokości sepa. Na zamieszczonych wcześniej obrazkach można zobaczyć jak wypadało grupowanie. Badania zostały przeprowadzone dla różnej ilości iteracji oraz różanego wskaźnika wag.

**Wnioski:**

Na podstawie otrzymanych wyników można było zauważyć ze dobór wag Kohonena oraz liczba iteracji są ze sobą powiazanie. Kiedy ustaliliśmy wysoki wskaźnik wag to wystarczała mała ilość iteracji aby otrzymać ten sam efekt który był przy małym wskaźniku wag i dużej ilości iteracji.

**Bibliografia:**

<https://www.mathworks.com/help/nnet/ref/competlayer.html>

<https://www.mathworks.com/help/nnet/ref/configure.html>

<https://en.wikipedia.org/wiki/Iris_flower_data_set>

<http://www.mbfys.ru.nl/~robvdw/CNP04/LAB_ASSIGMENTS/LAB05_CN05/MATLAB2007b/nnet/nndemos/html/democ1.html>

**Listing:**

close all; clear all; clc;

% I. setosa I. versicolor I. virginica

x =[5.2 4.9 4.7 4.6 5.0 5.4 4.6 5.4 5.7 5.8 7.0 6.4 6.9 5.5 6.5 5.7 6.3 6.6 5.9 6.1 6.3 6.7 7.1 6.3 6.5 7.6 7.3 6.7 6.5 6.4;

3.5 3.0 3.2 3.1 3.6 3.9 3.4 3.9 4.4 4.0 3.2 3.2 3.1 2.3 2.8 2.8 3.3 2.9 3.0 2.9 3.3 2.5 3.0 2.9 3.0 3.0 2.9 2.5 3.2 2.7;

1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.3 1.5 1.2 4.7 4.5 4.9 4.0 4.6 4.5 4.7 4.6 4.2 4.7 6.0 5.8 5.9 5.6 5.8 6.6 6.3 5.7 5.1 5.3;

0.2 0.4 0.2 0.2 0.3 0.4 0.1 0.4 0.4 0.2 1.4 1.5 1.5 1.3 1.5 1.3 1.6 1.3 1.5 1.5 2.5 1.8 2.1 1.8 2.2 2.1 1.8 1.8 2.0 1.9];

plot(x(3,:),x(4,:),'+r');

net = competlayer(3,.01);

net = configure(net,x);

w = eye(4,2)

plot(x(3,:),x(4,:),'+r');

hold on;

net.trainParam.epochs = 20;

net = train(net,x);

w = net.IW{1};

plot(w(:,3),w(:,4),'ob');

x1 = [0; 0.2];

y = net(x1)