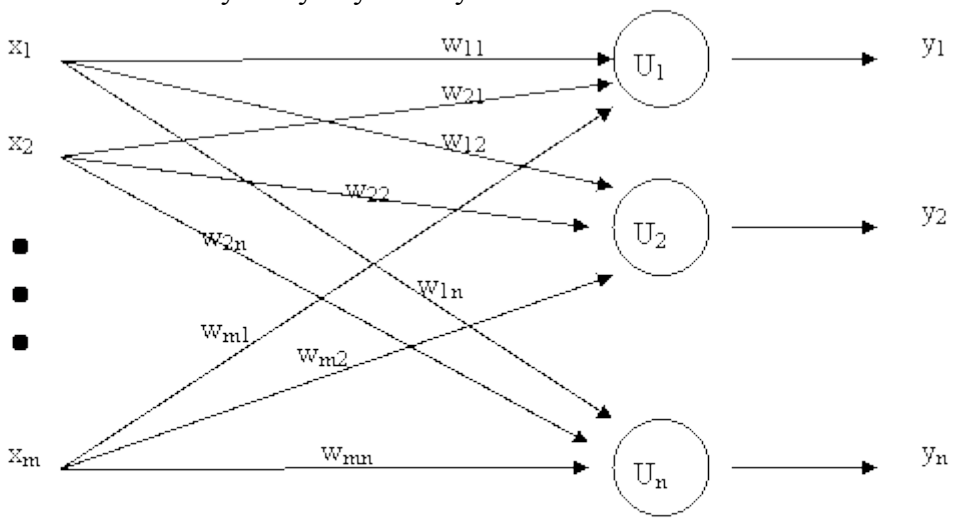
Krzysztof Kaflowski
IS
Podstawy Sztucznej Inteligencji

„Odwzorowanie charakterystycznych cech kwiatów za pomocą sieci typu Kohonen”

1. Celem ćwiczenia było stworzenie sieci neuronowej, która odwzorowuje charakterystyczne cechy kwiatów.
2. Opis sieci:

Jest to sieć typu Kohonen używająca algorytmu WTA.
Sieć kohonena jest to podstawowy typ sieci używającej samo-organizującej się mapy. Dzięki temu można uczyć ją bez nauczyciela. Jest to metoda najbliższa tej, która zachowi w naszych mózgach.
WTA – Winner Takes All – jest to algorytm, w którym neurony konkurują między sobą. Tylko jeden neuron może być aktywny w danym czasie.



Modyfikacja wag odbywa się tylko w neuronie, który wygra rywalizację. Oznaczo to, że neuron który zwróci największą wartość jako jedyny modyfikuje wartość wag według poniższego wzoru:

```
weight[i] = weight[i] + learningRate * (data.getX(i) - weight[i]);
```

Reguła uczenia konkurencyjnego powoduje stopniową zmianę kierunku wektora wag poszczególnych jednostek wyjściowych sieci w stronę statystycznie najczęściej występujących wejść.

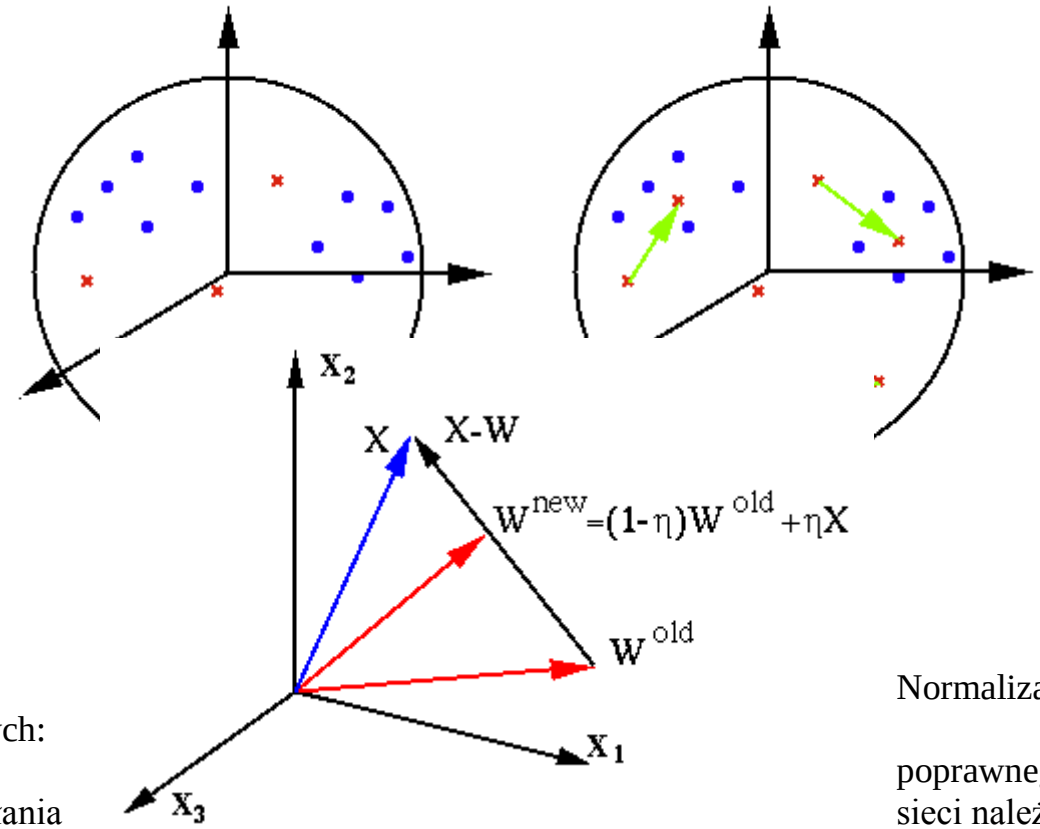
Regułę uczenia konkurencyjnego można zapisać w postaci uwzględniającej kolejne iteracje uczenia k, k+1, k+2,…

Cechy

$$w_j(k+1) = \begin{cases} w_j(k) + \eta_j(k)[x - w_j(k)] & j = j^* \\ w_j(k) & j \neq j^* \end{cases}$$

charakterystyczne:
• zwykle sieci jednowarstwowe
• każdy neuron jest połączony ze wszystkim składowymi wektora wejściowego X
• neurony liniowe

Wartości do siebie zbliżone zwykle mają największą wartość na tym samym neuronie, dzięki czemu możemy wyznaczyć grupy podziału danych.



Normalizacja danych:
Dla poprawnego działania sieci należało znormalizować wagi oraz dane uczące i testujące. Wykonano to w następujący sposób:

```
double lengthSquared = weight[0]*weight[0] + weight[1]*weight[1] + weight[2]*weight[2] +
weight[3]*weight[3];
double length = Math.sqrt(lengthSquared);

weight[0] /= length;
weight[1] /= length;
weight[2] /= length;
weight[3] /= length;
```

Analogicznie zostało to wykonane dla danych uczących oraz testujących.

3. Opis Programu

Program został napisany w czystym języku Java. Napisane zostały następujące klasy:

- Network – obsługa tworzenia sieci neuronowej
- Layer – obsługa tworzenia warstwy neuronowej

- Neuron – obsługa tworzenia pojedynczego neuronu
- Data – kontener na dane uczące oraz testujące
- LoadData – obsługa wczytywania danych uczących oraz testujących
- Main – główne wywołania programu

Stowrzona sieć składa się z 4 wejść oraz 30 wyjść.

4. Dane uczące oraz testujące

Dane uczące zawierają się w pliku „trainingIris.txt”, a dane testujące zawierają się w pliku „testingIris.txt”. Pliki znajdują się w repozytorium na GitHub’ie.
Oba pliki zawierają w pierwszej linii ilość rekordów, a w drugiej linii ilość danych w każdym rekordzie. Oba pliki zawierają po 149 rekordów oraz 4 dane na każdy rekord. Każdy rekord posiada docelowy gatunek, który jednak nie jest brany pod uwagę podczas uczenia. Służy on jedynie do porównania wyników podczas testowania.
Różnica pomiędzy plikami jest jedynie taka, że dane uczące są wymieszane, a dane testujące nie i gatunki się nie przeplatają.
Ilość danych dla odpowiednich gatunków:

- Setosa – 50
- Versicolor – 50
- Virginica – 49

Przykład wyglądu danych uczących i testujących:

```
149
4
5.2	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5	3.6	1.4	0.3	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
```

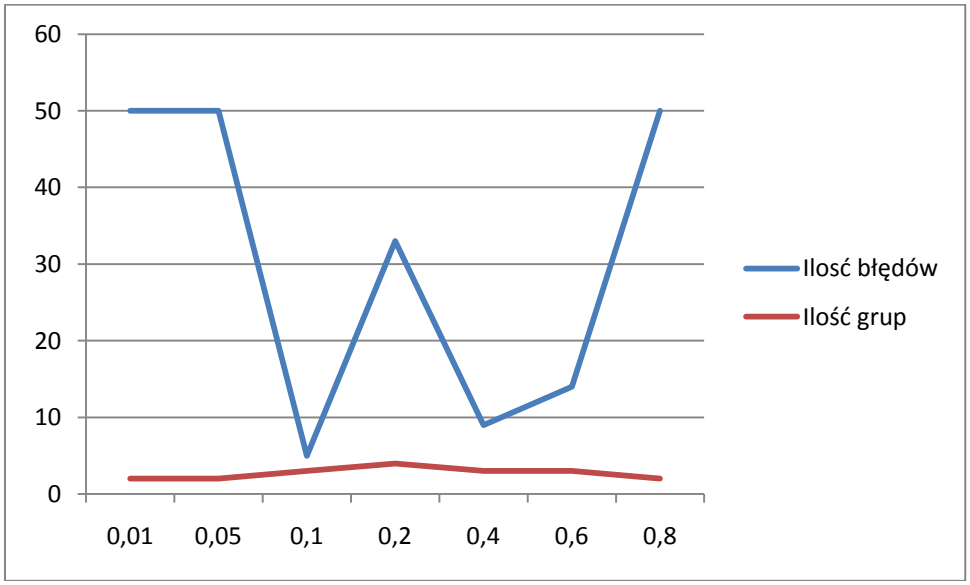
5. Wyniki działania programu:

W ramach przeanalizowania działania programu przeprowadzono testy z następującymi zmiennymi:

- Zmienny współczynnik uczenia
- Zmienna ilość epok
- Zmienna ilość neuronów

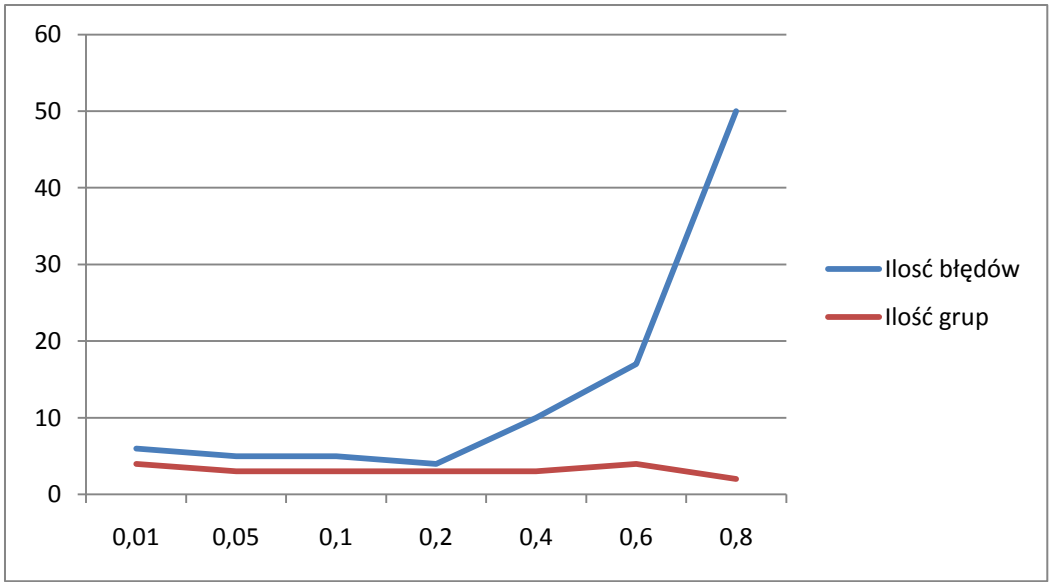
Dane testujące zawierają 149 rekordów, zatem maksymalna liczba błędów w następujących rozważaniach wynosi również 149.
Jako pierwsze w kolejności wykonano test ze zmiennym współczynnikiem uczenia dla różnych ilośći epok:

Współczynnik uczenia	Ilosć błędów	Ilość grup	Ilość iteracji
0,01	50	2	1000
0,05	50	2	1000
0,1	5	3	1000
0,2	33	4	1000
0,4	9	3	1000
0,6	14	3	1000
0,8	50	2	1000



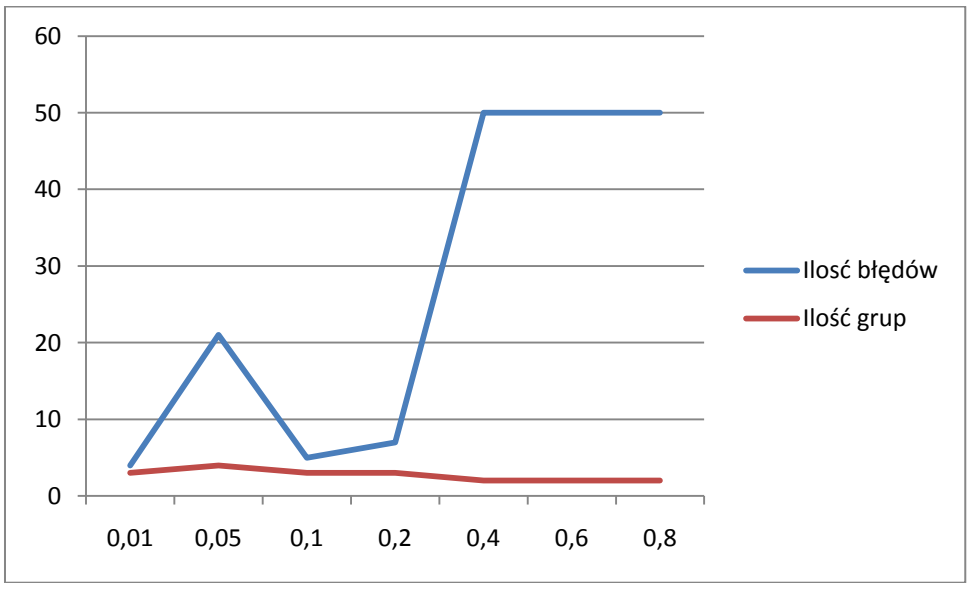
Dla 1000 iteracji wynik był mało przewidywalny i nie zaleca się korzystania z tak małej liczby epok.

Współczynnik uczenia	Ilosć błędów	Ilość grup	Ilość iteracji
0,01	6	4	100000
0,05	5	3	100000
0,1	5	3	100000
0,2	4	3	100000
0,4	10	3	100000
0,6	17	4	100000
0,8	50	2	100000



W przypadku wykorzystania 100000 epok do nauczania programu można zauważyć, że ze wzrostem współczynnika uczenia zwiększała się również liczba błędów.

Współczynnik uczenia	Ilosc błędów	Ilość grup	Ilość iteracji
0,01	4	3	10000000
0,05	21	4	10000000
0,1	5	3	10000000
0,2	7	3	10000000
0,4	50	2	10000000
0,6	50	2	10000000
0,8	50	2	10000000

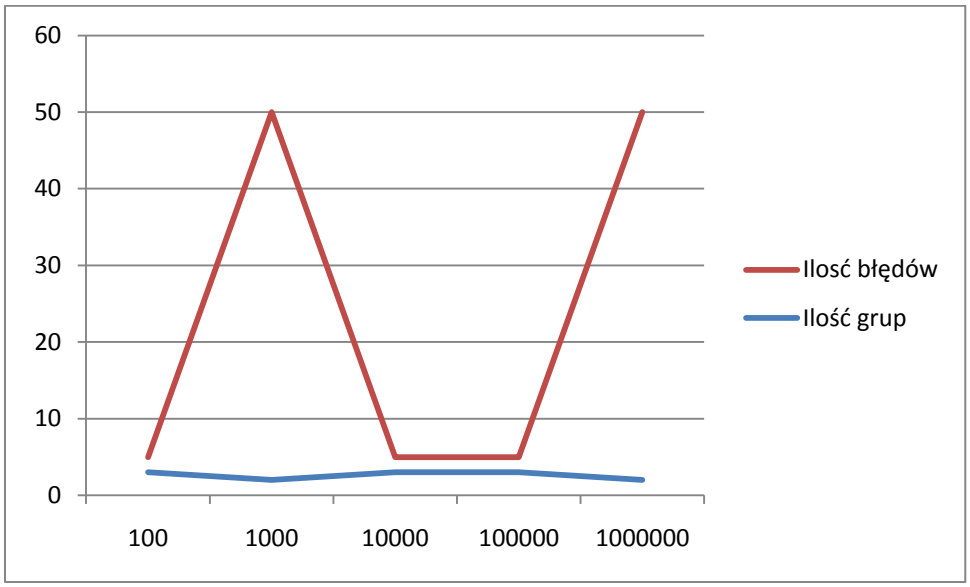


W tym przypadku można stwierdzić, że najlepiej sprawdziły się współczynniki 0,01; 0,1 oraz 0,2. W przypadku pozostałych liczba błędów była już zbyt wysoka do zaakceptowania.

Po przeanalizowaniu tych trzech przypadków można stwierdzić, że współczynnik uczenia 0,1 zawsze dawał satysfakcjonujące wyniki, dlatego w dalszych testach będzie on brany jako stała wartość współczynnika uczenia.

Test dla zmiennej liczby iteracji oraz stałego współczynnika ucznia:

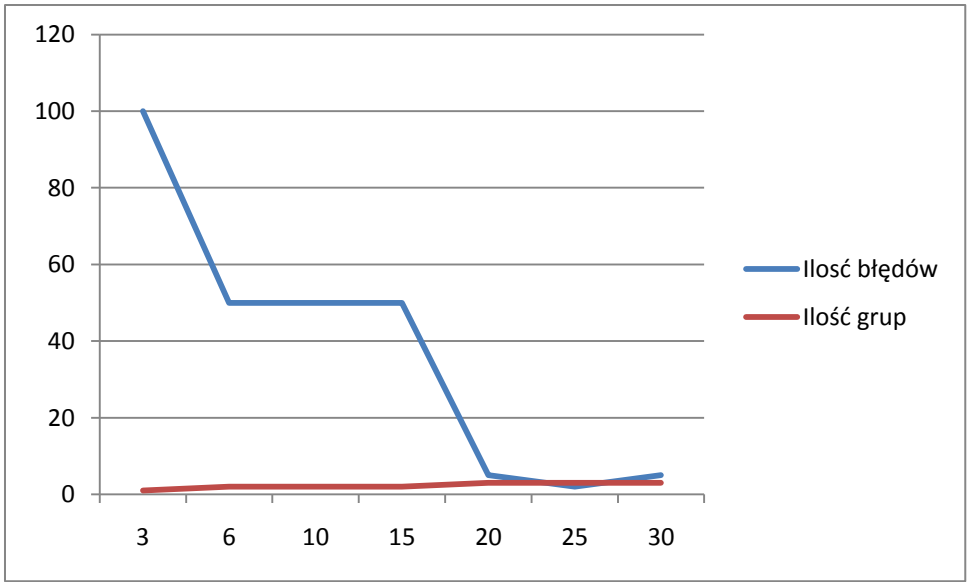
Współczynnik uczenia	Ilosc błędów	Ilość grup	Ilość iteracji
0,1	5	3	100
0,1	50	2	1000
0,1	5	3	10000
0,1	5	3	100000
0,1	50	2	1000000



W tym przypadku widać, że zbyt duża ilość epok powoduje błędy przeuczenia się sieci.

Ostatnim testem była zmienna ilość neuronów:

Ilość Neuronów	Współczynnik uczenia	Ilosc błędów	Ilość grup	Ilość iteracji
3	0,1	100	1	10000
6	0,1	50	2	10000
10	0,1	50	2	10000
15	0,1	50	2	10000
20	0,1	5	3	10000
25	0,1	2	3	10000
30	0,1	5	3	10000



Łatwo można zauważyć, że kiedy liczba neuronów spada poniżej 20 to tworzy się zbyt mało grup i dwa gatunki kwiatów przydzielane są do tej samej grupy.

3. Analiza wyników:

W działaniu tej sieci każdy element ma równie duże znaczenie dla poprawnego działania sieci.
Najlepiej sprawdziły się ustawienia:
Współczynnik uczenia 0,1
Ilość iteracji 100000
Ilość neuronów 30

Wartość ilości błędów wynoszący 50 oraz stworzenie 2 grup zamiast 3 jest prawdopodobnie spowodowane tym, że 2 gatunki są bardzo podobne do siebie i sieć klasyfikowało je jako jedno.

Dla małej ilości neuronów sieć tworzyła zbyt mało grup, aby program poprawnie rozpoznawał wzorce kwiatów.

4. Wnioski:
Dzięki sieci Kohonena wykorzystującej WTA (Winner Takes All) można rozpoznawać charakterystyczne wzorce.
Należy pamiętać o poprawnym dobraniu parametrów, gdyż mają one duży wpływ na działanie programu.
Normalizacja danych odgrywa dużą rolę w sieci neuronowej.
Sieć często miewa problemy przy rozpoznawaniu podobnych do siebie danych, które jednak należą realnie do innych grup, tak jak w tym przypadku Irysy Virgnica i Versicolor.

5. Listing kodu programu:

Kod programu znajduje się w repozytorium pod linkiem: