Sprawozdanie LAB1

Magdalena Górka

Grupa Projektowa nr 1

Temat:

Budowa i działanie perceptronu

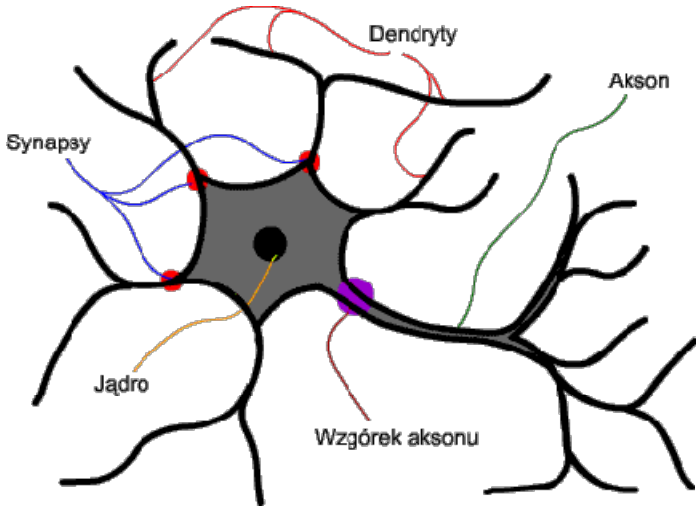
Cel ćwiczenia:

Celem ćwiczenia jest poznanie budowy i działanie perceptronu poprzez implementację oraz uczenie perceptronu realizującego wybraną funkcję logiczną dwóch zmiennych.

Opis Budowy neuronu oraz algorytmu:

Sztuczny Neuron –podstawowy element sieci neuronowych, która jest jedną z metod sztucznej inteligencji, odwzorowany na podstawie biologicznego neuronu, pozwala na przetwarzanie wielu wartości wejściowych na jedną wartość wyjściową.

Neuron biologiczny a sztuczny - RÓŻNICE:



Dendryty – wektor danych wejściowych

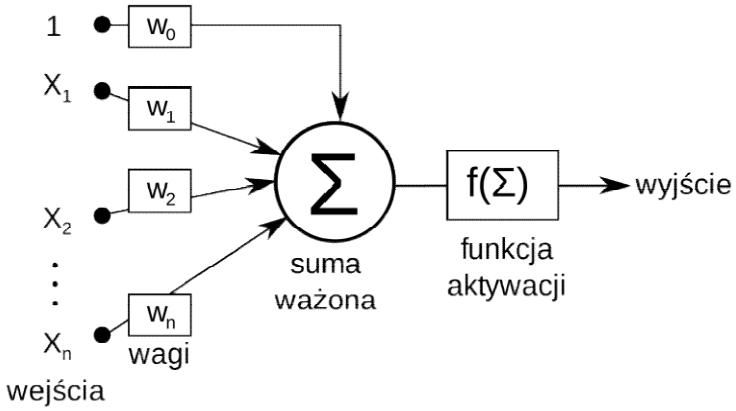
Synapsy – wektor wag

Jądro – blok sumujący

Wzgórek aksonu – blok aktywacji

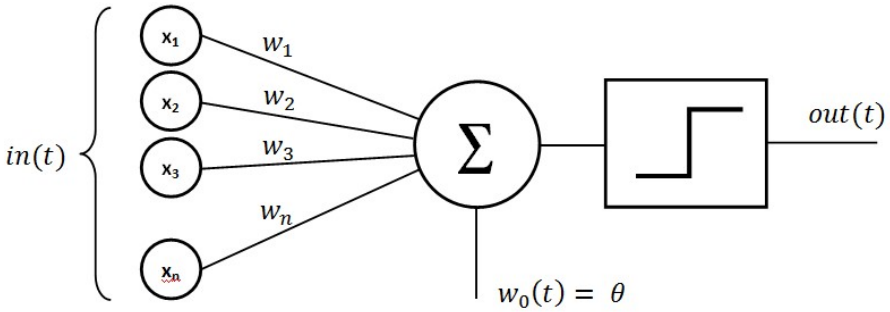
Akson – wyjście sztucznego neuronu

Schemat sztucznego neuronu McCullocha-Pittsa:



Perceptron – najprostsza sieć neuronowa, składająca się z jednego bądź wielu niezależnych neuronów McCullocha-Pittsa. Działanie perceptronu polega na klasyfikowaniu danych pojawiających się na wejściu i ustawianiu stosownie do tego wartości wyjścia. Przed używaniem perceptron należy wytrenować, podając mu przykładowe dane na wejście i modyfikując w odpowiedni sposób wagi wejść i połączeń między warstwami neuronów, tak aby wartość na wyjściu przybierała pożądane wartości.

Schemat perceptronu złożonego z jednego neuronu McCullocha-Pittsa.



Algorytm użyty w zadaniu:

1. Dla danego neuronu, perceptron inicjuje losowe wagi z podanego przedziału(dla uproszczenia obliczeń).

2. Na podstawie danych wejściowych początkowych i zainicjowanych wag następuje obliczenie wyjścia neuronu oraz porównanie z oczekiwanym rezultatem także podanym we wzorcu.

3. Jeśli wyjście neuronu jest takie samo jak rezultat wyjściowy, następuje sprawdzenie kolejnych danych wzorcowych. Jeśli jest różny, następuje korekcja wag:

Waga1=waga1+stała_uczenia*(rezultat_oczekiwany-rezultat_otrzymany)*x1

Waga2=waga2+stała_uczenia*(rezultat_oczekiwany-rezultat_otrzymany)*x1

4. Następuje sprawdzenie kolejnych danych wzorcowych.

5. Ponieważ nastąpiła zmiana wag, algorytm ponownie zaczyna przeprowadzać sprawdzanie wszystkich danych wzorcowych.

6. Algorytm kończy się w momencie, gdy nie nastąpi zmiana wag.

Podany program sortuje owoce na podstawie wagi oraz koloru – wzór podany na wykładzie. W pliku tekstowym z danymi – pierwsza liczba to waga, druga to kolor, a trzecia – dana wyjsciowa dla owocu (1 lub 0).

Wyniki:

Naukę programu przeprowadzono w kilku wersjach by sprawdzić jaki wpływ na naukę oraz wyniki mają podane współczynniki:

-stała uczenia

-ilość danych uczących

Wersja I:

Wersja wyjściowa, do której będą porównywane pozostałe wersje.

Stała uczenia = 0.2

Liczba rekordów uczących z pliku = 4

Ilość epok, po których widoczny jest postęp nauczania (przybliżając wartości do 0.1 dla 0 oraz 0.99 dla 1), dla których możemy stwierdzić, że neuron się nauczył zależy od wygenerowanych wag.

Przykładowy program:

```
Podaj ilosc danych do 20: 4

WAGI poczatkowe: Dane z pliku:
0.81 0.22
0.2  0  0
1  1  1
0.25  0  0
1.4  1  1
_____________________ EPOKA 1_____________________
Funkcja aktywacji: 0.0323887  Rekord nauczony !
Funkcja aktywacji: 0.203135 w1: 0.969373 , w2: 0.373303
Funkcja aktywacji: 0.0484307  Rekord nauczony !
Funkcja aktywacji: 0.332899 w1: 1.15616 , w2: 0.497589
_____________________ EPOKA 2_____________________
Funkcja aktywacji: 0.0462135  Rekord nauczony !
Funkcja aktywacji: 0.319195 w1: 1.29232 , w2: 0.628902
Funkcja aktywacji: 0.0645263  Rekord nauczony !
Funkcja aktywacji: 0.452334 w1: 1.44567 , w2: 0.731739
_____________________ EPOKA 3_____________________
Funkcja aktywacji: 0.0577624  Rekord nauczony !
Funkcja aktywacji: 0.409892 w1: 1.56369 , w2: 0.845871
Funkcja aktywacji: 0.0780256  Rekord nauczony !
Funkcja aktywacji: 0.542017 w1: 1.69193 , w2: 0.932495
_____________________ EPOKA 4_____________________
Funkcja aktywacji: 0.0675739  Rekord nauczony !
Funkcja aktywacji: 0.481461 w1: 1.79563 , w2: 1.03305
Funkcja aktywacji: 0.0895412  Rekord nauczony !
Funkcja aktywacji: 0.610292 w1: 1.90475 , w2: 1.10723
_____________________ EPOKA 5_____________________
Funkcja aktywacji: 0.076043  Rekord nauczony !
Funkcja aktywacji: 0.538752 w1: 1.997 , w2: 1.19688
Funkcja aktywacji: 0.0995195  Rekord nauczony !
Funkcja aktywacji: 0.663218 w1: 2.0913 , w2: 1.26133
_____________________ EPOKA 6_____________________
Funkcja aktywacji: 0.0834574  Rekord nauczony !
Funkcja aktywacji: 0.585326 w1: 2.17423 , w2: 1.34211
Funkcja aktywacji: 0.108286 w1: 2.16882 , w2: 1.34211
Funkcja aktywacji: 0.704254 w1: 2.25163 , w2: 1.39896
_____________________ EPOKA 7_____________________
Funkcja aktywacji: 0.0898224  Rekord nauczony !
Funkcja aktywacji: 0.623137 w1: 2.327 , w2: 1.4725
Funkcja aktywacji: 0.115828 w1: 2.32121 , w2: 1.4725
Funkcja aktywacji: 0.737255 w1: 2.39478 , w2: 1.5232
_____________________ EPOKA 8_____________________
```

.....

```
_____________________ EPOKA 446_____________________
Funkcja aktywacji: 0.0803404  Rekord nauczony !
Funkcja aktywacji: 0.900149  Rekord nauczony !
Funkcja aktywacji: 0.100304 w1: 2.00783 , w2: 5.35217
Funkcja aktywacji: 0.926435  Rekord nauczony !
_____________________ EPOKA 447_____________________
Funkcja aktywacji: 0.0801411  Rekord nauczony !
Funkcja aktywacji: 0.899958 w1: 2.02784 , w2: 5.37203
Funkcja aktywacji: 0.101046 w1: 2.02279 , w2: 5.37203
Funkcja aktywacji: 0.927582  Rekord nauczony !
_____________________ EPOKA 448_____________________
Funkcja aktywacji: 0.0807354  Rekord nauczony !
Funkcja aktywacji: 0.901274  Rekord nauczony !
Funkcja aktywacji: 0.100796 w1: 2.01775 , w2: 5.37203
Funkcja aktywacji: 0.927385  Rekord nauczony !
_____________________ EPOKA 449_____________________
Funkcja aktywacji: 0.0805352  Rekord nauczony !
Funkcja aktywacji: 0.901084  Rekord nauczony !
Funkcja aktywacji: 0.100547 w1: 2.01272 , w2: 5.37203
Funkcja aktywacji: 0.927188  Rekord nauczony !
_____________________ EPOKA 450_____________________
Funkcja aktywacji: 0.0803354  Rekord nauczony !
Funkcja aktywacji: 0.900895  Rekord nauczony !
Funkcja aktywacji: 0.100298 w1: 2.00771 , w2: 5.37203
Funkcja aktywacji: 0.926991  Rekord nauczony !
_____________________ EPOKA 451_____________________
Funkcja aktywacji: 0.0801361  Rekord nauczony !
Funkcja aktywacji: 0.900706  Rekord nauczony !
Funkcja aktywacji: 0.100049 w1: 2.0027 , w2: 5.37203
Funkcja aktywacji: 0.926793  Rekord nauczony !
_____________________ EPOKA 452_____________________
Funkcja aktywacji: 0.0799373  Rekord nauczony !
Funkcja aktywacji: 0.900517  Rekord nauczony !
Funkcja aktywacji: 0.0998019  Rekord nauczony !
Funkcja aktywacji: 0.926793  Rekord nauczony !
Neuron nauczyl sie za 452 razem
```

Gdy neuron nauczy się rekordu pojawia się informacja oraz na samym końcu również widoczny jest komunikat, że neuron naczył się całości za odpowiadającej liczbię epok razem.

Obserwacje:

Ilość iteracji utrzymywała się w zakresie 420-470.

W przeprowadzonym teście nie napotkano żadnych błędów.

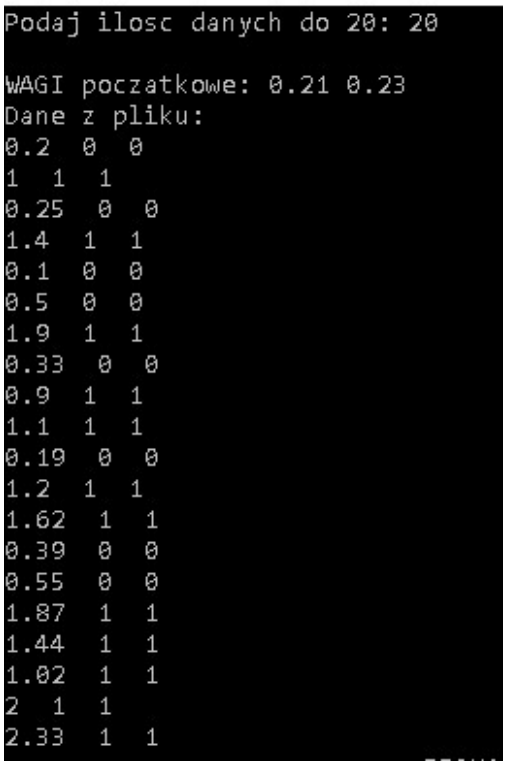
Wersja II:

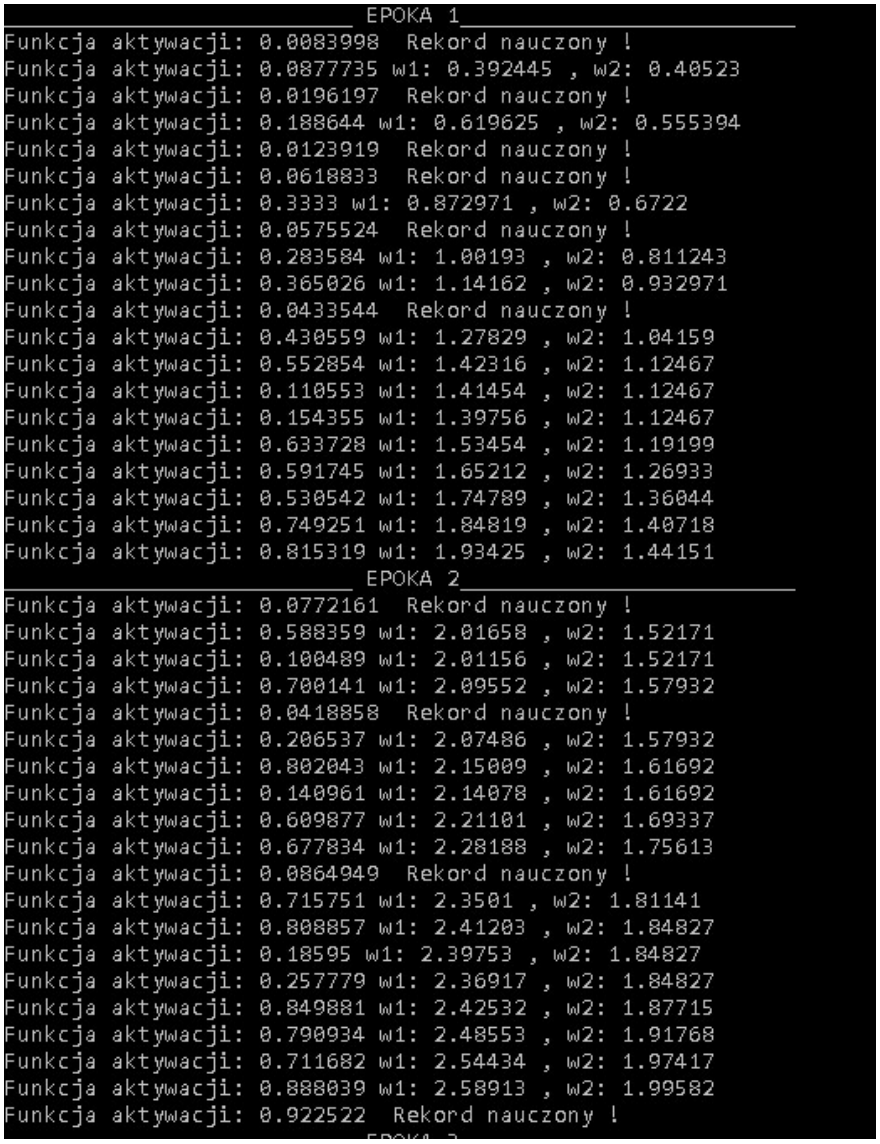
Stała uczenia = 0.2

Liczba rekordów uczących z pliku= 20 – zwiekszono liczbe

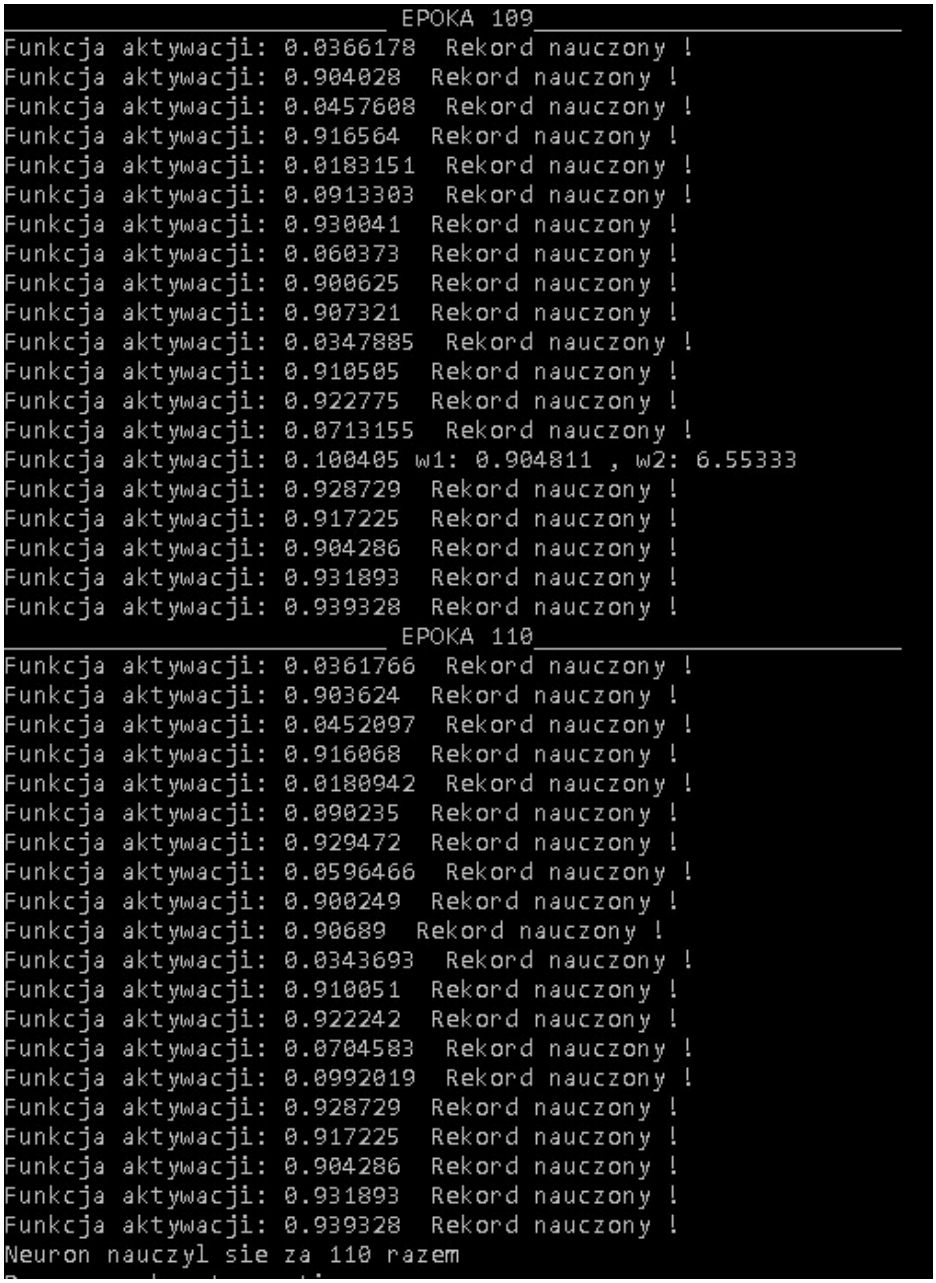
Ilość epok, po których widoczny jest postęp nauczania (przybliżając wartości do 0.1 dla 0 oraz 0.99 dla 1), dla których możemy stwierdzić, że neuron się nauczył zależy od wygenerowanych wag.

Przykładowy program:




EPOKA 1
Funkcja aktywacji: 0.0083998 Rekord nauczony !
Funkcja aktywacji: 0.0877735 w1: 0.392445 , w2: 0.40523
Funkcja aktywacji: 0.0196197 Rekord nauczony !
Funkcja aktywacji: 0.188644 w1: 0.619625 , w2: 0.555394
Funkcja aktywacji: 0.0123919 Rekord nauczony !
Funkcja aktywacji: 0.0618833 Rekord nauczony !
Funkcja aktywacji: 0.3333 w1: 0.872971 , w2: 0.6722
Funkcja aktywacji: 0.0575524 Rekord nauczony !
Funkcja aktywacji: 0.283584 w1: 1.00193 , w2: 0.811243
Funkcja aktywacji: 0.365026 w1: 1.14162 , w2: 0.932971
Funkcja aktywacji: 0.0433544 Rekord nauczony !
Funkcja aktywacji: 0.430559 w1: 1.27829 , w2: 1.04159
Funkcja aktywacji: 0.552854 w1: 1.42316 , w2: 1.12467
Funkcja aktywacji: 0.110553 w1: 1.41454 , w2: 1.12467
Funkcja aktywacji: 0.154355 w1: 1.39756 , w2: 1.12467
Funkcja aktywacji: 0.633728 w1: 1.53454 , w2: 1.19199
Funkcja aktywacji: 0.591745 w1: 1.65212 , w2: 1.26933
Funkcja aktywacji: 0.530542 w1: 1.74789 , w2: 1.36044
Funkcja aktywacji: 0.749251 w1: 1.84819 , w2: 1.40718
Funkcja aktywacji: 0.815319 w1: 1.93425 , w2: 1.44151
EPOKA 2
Funkcja aktywacji: 0.0772161 Rekord nauczony !
Funkcja aktywacji: 0.588359 w1: 2.01658 , w2: 1.52171
Funkcja aktywacji: 0.100489 w1: 2.01156 , w2: 1.52171
Funkcja aktywacji: 0.700141 w1: 2.09552 , w2: 1.57932
Funkcja aktywacji: 0.0418858 Rekord nauczony !
Funkcja aktywacji: 0.206537 w1: 2.07486 , w2: 1.57932
Funkcja aktywacji: 0.802043 w1: 2.15009 , w2: 1.61692
Funkcja aktywacji: 0.140961 w1: 2.14078 , w2: 1.61692
Funkcja aktywacji: 0.609877 w1: 2.21101 , w2: 1.69337
Funkcja aktywacji: 0.677834 w1: 2.28188 , w2: 1.75613
Funkcja aktywacji: 0.0864949 Rekord nauczony !
Funkcja aktywacji: 0.715751 w1: 2.3501 , w2: 1.81141
Funkcja aktywacji: 0.808857 w1: 2.41203 , w2: 1.84827
Funkcja aktywacji: 0.18595 w1: 2.39753 , w2: 1.84827
Funkcja aktywacji: 0.257779 w1: 2.36917 , w2: 1.84827
Funkcja aktywacji: 0.849881 w1: 2.42532 , w2: 1.87715
Funkcja aktywacji: 0.790934 w1: 2.48553 , w2: 1.91768
Funkcja aktywacji: 0.711682 w1: 2.54434 , w2: 1.97417
Funkcja aktywacji: 0.888039 w1: 2.58913 , w2: 1.99582
Funkcja aktywacji: 0.922522 Rekord nauczony !

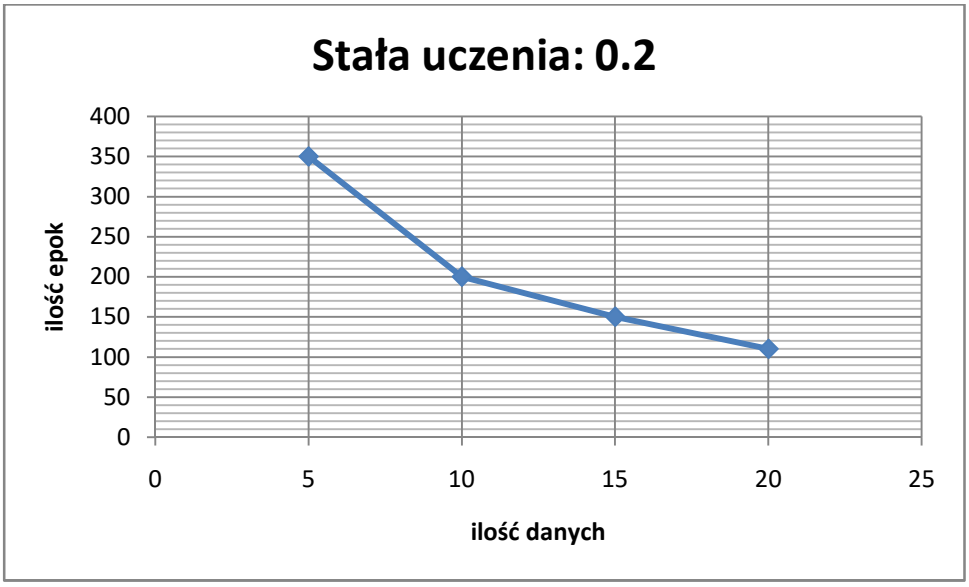
.....



Obserwacje:

Ilość iteracji utrzymywała się w zakresie 100-120.

W przeprowadzonym teście nie napotkano żadnych błędów.



Wersja III:

Stała uczenia = 0.5 – zwiększono stałą

Liczba par danych uczących = 4

Przykładowy program :

```
Podaj ilosc danych do 20: 4

WAGI poczatkowe: 0.26 0.44
Dane z pliku:
0.2  0  0
1  1  1
0.25  0  0
1.4  1  1
                          EPOKA 1
Funkcja aktywacji: 0.0259941  Rekord nauczony !
Funkcja aktywacji: 0.336376 w1: 0.591812 , w2: 0.702733
Funkcja aktywacji: 0.0738419  Rekord nauczony !
Funkcja aktywacji: 0.644384 w1: 0.840743 , w2: 0.835148
                          EPOKA 2
Funkcja aktywacji: 0.0838768  Rekord nauczony !
Funkcja aktywacji: 0.684719 w1: 0.998384 , w2: 0.972966
Funkcja aktywacji: 0.124154 w1: 0.982864 , w2: 0.972966
Funkcja aktywacji: 0.825706 w1: 1.10487 , w2: 1.04745
                          EPOKA 3
Funkcja aktywacji: 0.11004 w1: 1.09387 , w2: 1.04745
Funkcja aktywacji: 0.789709 w1: 1.19901 , w2: 1.1431
Funkcja aktywacji: 0.148764 w1: 1.18042 , w2: 1.1431
Funkcja aktywacji: 0.884885 w1: 1.261 , w2: 1.19484
                          EPOKA 4
Funkcja aktywacji: 0.125436 w1: 1.24845 , w2: 1.19484
Funkcja aktywacji: 0.840139 w1: 1.32838 , w2: 1.26908
Funkcja aktywacji: 0.164539 w1: 1.30782 , w2: 1.26908
Funkcja aktywacji: 0.913788  Rekord nauczony !
                          EPOKA 5
```

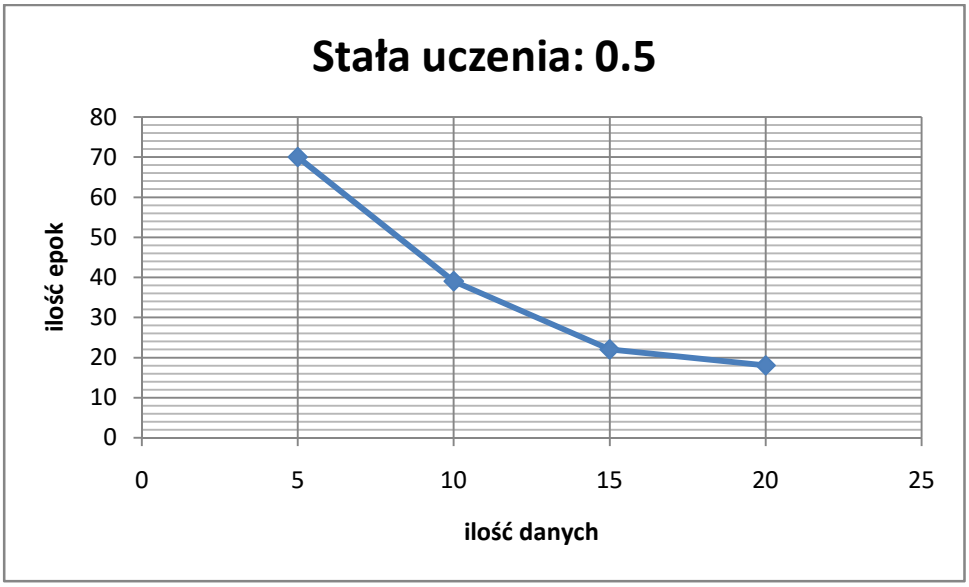
……

```
                        EPOKA 64
Funkcja aktywacji: 0.0840887  Rekord nauczony !
Funkcja aktywacji: 0.907257  Rekord nauczony !
Funkcja aktywacji: 0.104972 w1: 0.829756 , w2: 2.18071
Funkcja aktywacji: 0.931708  Rekord nauczony !
                        EPOKA 65
Funkcja aktywacji: 0.0827857  Rekord nauczony !
Funkcja aktywacji: 0.906089  Rekord nauczony !
Funkcja aktywacji: 0.103349 w1: 0.816838 , w2: 2.18071
Funkcja aktywacji: 0.930505  Rekord nauczony !
                        EPOKA 66
Funkcja aktywacji: 0.0815026  Rekord nauczony !
Funkcja aktywacji: 0.904926  Rekord nauczony !
Funkcja aktywacji: 0.101751 w1: 0.804119 , w2: 2.18071
Funkcja aktywacji: 0.9293  Rekord nauczony !
                        EPOKA 67
Funkcja aktywacji: 0.080239  Rekord nauczony !
Funkcja aktywacji: 0.903768  Rekord nauczony !
Funkcja aktywacji: 0.100178 w1: 0.791596 , w2: 2.18071
Funkcja aktywacji: 0.928095  Rekord nauczony !
                        EPOKA 68
Funkcja aktywacji: 0.0789947  Rekord nauczony !
Funkcja aktywacji: 0.902614  Rekord nauczony !
Funkcja aktywacji: 0.0986279  Rekord nauczony !
Funkcja aktywacji: 0.928095  Rekord nauczony !
Neuron nauczyl sie za 68 razem
```

Obserwacje:

Ilość iteracji utrzymywała się w zakresie 50-80.

W przeprowadzonym teście nie napotkano żadnych błędów.



<u>Wersja IV</u>:

Stała uczenia = 1 – zwiększono stałą

Liczba rekordów uczących z pliku= 4

stwierdzić, że neuron się nauczył zależy od wygenerowanych wag.

Przykładowy program:

```
Podaj ilosc danych do 20: 4

WAGI poczatkowe: 0.79 0.63
Dane z pliku:
0.2  0  0
1  1  1
0.25  0  0
1.4  1  1
_____________________ EPOKA 1_____________________
Funkcja aktywacji: 0.156698 w1: 0.75866 , w2: 0.63
Funkcja aktywacji: 0.882876 w1: 0.875785 , w2: 0.723812
Funkcja aktywacji: 0.215513 w1: 0.821906 , w2: 0.723812
Funkcja aktywacji: 0.953999  Rekord nauczony !
_____________________ EPOKA 2_____________________
Funkcja aktywacji: 0.162916 w1: 0.789323 , w2: 0.723812
Funkcja aktywacji: 0.907494  Rekord nauczony !
Funkcja aktywacji: 0.194809 w1: 0.740621 , w2: 0.723812
Funkcja aktywacji: 0.942579  Rekord nauczony !
_____________________ EPOKA 3_____________________
Funkcja aktywacji: 0.14705 w1: 0.711211 , w2: 0.723812
Funkcja aktywacji: 0.892691 w1: 0.818519 , w2: 0.811301
Funkcja aktywacji: 0.201821 w1: 0.768064 , w2: 0.811301
Funkcja aktywacji: 0.955075  Rekord nauczony !
_____________________ EPOKA 4_____________________
Funkcja aktywacji: 0.152416 w1: 0.737581 , w2: 0.811301
Funkcja aktywacji: 0.913601  Rekord nauczony !
Funkcja aktywacji: 0.182333 w1: 0.691998 , w2: 0.811301
Funkcja aktywacji: 0.944706  Rekord nauczony !
_____________________ EPOKA 5_____________________
Funkcja aktywacji: 0.137523 w1: 0.664493 , w2: 0.811301
Funkcja aktywacji: 0.900677  Rekord nauczony !
Funkcja aktywacji: 0.164612 w1: 0.62334 , w2: 0.811301
Funkcja aktywacji: 0.933376  Rekord nauczony !
_____________________ EPOKA 6_____________________
Funkcja aktywacji: 0.124026 w1: 0.598535 , w2: 0.811301
Funkcja aktywacji: 0.887459 w1: 0.711076 , w2: 0.90219
Funkcja aktywacji: 0.17592 w1: 0.667096 , w2: 0.90219
Funkcja aktywacji: 0.950422  Rekord nauczony !
_____________________ EPOKA 7_____________________
Funkcja aktywacji: 0.132633 w1: 0.640569 , w2: 0.90219
Funkcja aktywacji: 0.912583  Rekord nauczony !
Funkcja aktywacji: 0.158787 w1: 0.600872 , w2: 0.90219
Funkcja aktywacji: 0.940621  Rekord nauczony !
_____________________ EPOKA 8
```

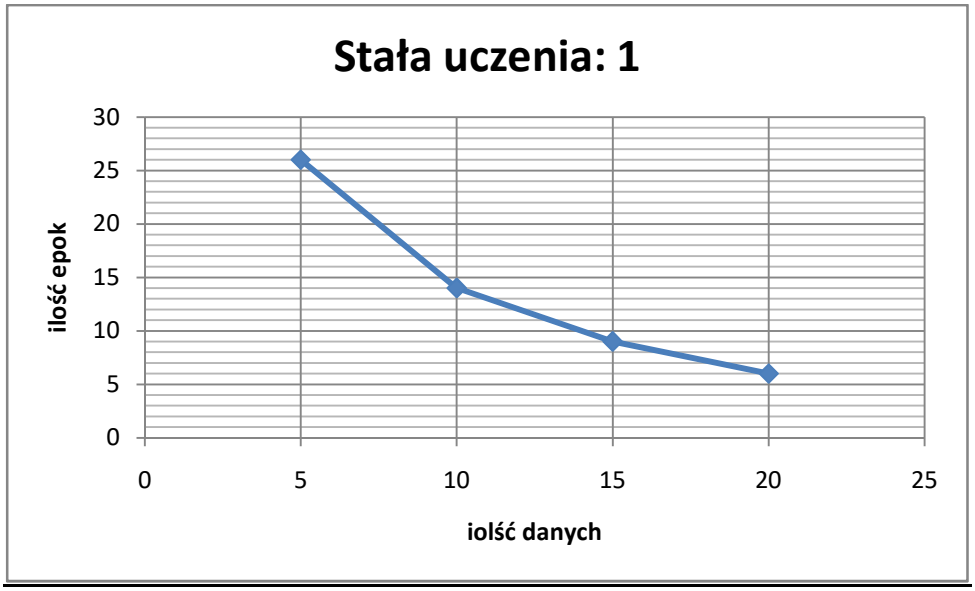
```
_____________________ EPOKA 8_____________________
Funkcja aktywacji: 0.119599 w1: 0.576953 , w2: 0.90219
Funkcja aktywacji: 0.901307  Rekord nauczony !
Funkcja aktywacji: 0.143246 w1: 0.541141 , w2: 0.90219
Funkcja aktywacji: 0.930189  Rekord nauczony !
_____________________ EPOKA 9_____________________
Funkcja aktywacji: 0.107808 w1: 0.51958 , w2: 0.90219
Funkcja aktywacji: 0.889967 w1: 0.629612 , w2: 0.991457
Funkcja aktywacji: 0.156116 w1: 0.590583 , w2: 0.991457
Funkcja aktywacji: 0.948666  Rekord nauczony !
_____________________ EPOKA 10_____________________
Funkcja aktywacji: 0.11757 w1: 0.567069 , w2: 0.991457
Funkcja aktywacji: 0.915181  Rekord nauczony !
Funkcja aktywacji: 0.140825 w1: 0.531863 , w2: 0.991457
Funkcja aktywacji: 0.939769  Rekord nauczony !
_____________________ EPOKA 11_____________________
Funkcja aktywacji: 0.105973 w1: 0.510668 , w2: 0.991457
Funkcja aktywacji: 0.905532  Rekord nauczony !
Funkcja aktywacji: 0.126978 w1: 0.478924 , w2: 0.991457
Funkcja aktywacji: 0.930479  Rekord nauczony !
_____________________ EPOKA 12_____________________
Funkcja aktywacji: 0.0954929  Rekord nauczony !
Funkcja aktywacji: 0.89965 w1: 0.579274 , w2: 1.07432
Funkcja aktywacji: 0.143814 w1: 0.54332 , w2: 1.07432
Funkcja aktywacji: 0.950309  Rekord nauczony !
_____________________ EPOKA 13_____________________
Funkcja aktywacji: 0.108238 w1: 0.521672 , w2: 1.07432
Funkcja aktywacji: 0.921062  Rekord nauczony !
Funkcja aktywacji: 0.129684 w1: 0.489251 , w2: 1.07432
Funkcja aktywacji: 0.942421  Rekord nauczony !
_____________________ EPOKA 14_____________________
Funkcja aktywacji: 0.0975392  Rekord nauczony !
Funkcja aktywacji: 0.915996  Rekord nauczony !
Funkcja aktywacji: 0.121707 w1: 0.458825 , w2: 1.07432
Funkcja aktywacji: 0.937461  Rekord nauczony !
_____________________ EPOKA 15_____________________
Funkcja aktywacji: 0.0915082  Rekord nauczony !
Funkcja aktywacji: 0.91096  Rekord nauczony !
Funkcja aktywacji: 0.114206 w1: 0.430273 , w2: 1.07432
Funkcja aktywacji: 0.932432  Rekord nauczony !
```

```
Funkcja aktywacji: 0.952432  Rekord nauczony !
                          EPOKA 16
Funkcja aktywacji: 0.0858429  Rekord nauczony !
Funkcja aktywacji: 0.905974  Rekord nauczony !
Funkcja aktywacji: 0.107155 w1: 0.403484 , w2: 1.07432
Funkcja aktywacji: 0.92736  Rekord nauczony !
                          EPOKA 17
Funkcja aktywacji: 0.0805222  Rekord nauczony !
Funkcja aktywacji: 0.901055  Rekord nauczony !
Funkcja aktywacji: 0.10053 w1: 0.378352 , w2: 1.07432
Funkcja aktywacji: 0.92227  Rekord nauczony !
                          EPOKA 18
Funkcja aktywacji: 0.0755263  Rekord nauczony !
Funkcja aktywacji: 0.896219 w1: 0.482133 , w2: 1.15947
Funkcja aktywacji: 0.119953 w1: 0.452145 , w2: 1.15947
Funkcja aktywacji: 0.946021  Rekord nauczony !
                          EPOKA 19
Funkcja aktywacji: 0.0901832  Rekord nauczony !
Funkcja aktywacji: 0.923399  Rekord nauczony !
Funkcja aktywacji: 0.112557 w1: 0.424005 , w2: 1.15947
Funkcja aktywacji: 0.941725  Rekord nauczony !
                          EPOKA 20
Funkcja aktywacji: 0.0845984  Rekord nauczony !
Funkcja aktywacji: 0.919143  Rekord nauczony !
Funkcja aktywacji: 0.105606 w1: 0.397604 , w2: 1.15947
Funkcja aktywacji: 0.937394  Rekord nauczony !
                          EPOKA 21
Funkcja aktywacji: 0.0793536  Rekord nauczony !
Funkcja aktywacji: 0.914945  Rekord nauczony !
Funkcja aktywacji: 0.0990749  Rekord nauczony !
Funkcja aktywacji: 0.937394  Rekord nauczony !
Neuron nauczyl sie za 21 razem
```

Obserwacje:

Ilość iteracji utrzymywała się w zakresie 15-30.

W przeprowadzonym teście nie napotkano żadnych błędów.



Analiza:

Zwiększenie stałej uczącej przyśpieszyło działanie programu. Wynika, że im większy współczynnik uczenia, tym szybciej postępuje uczenie. Dla dużych wartości współczynnika uczenia wystarczy tylko kilka zmian wag, aby można poprawnie nauczyć perceptron. Zwiększenie stałej uczącej spowodowało, że zmniejszyła się dokładność zmiany wagi.

Zauważono wpływ losowych wag na działanie programu, w szczególności na liczbę iteracji. Możliwe było wylosowanie przez program wag zbliżonych do odpowiednich co skutkowało tym, że w jednej próbie potrzeba było mniej iteracji niż w innej by nauczyć perceptron.

W innych próbach nie podanych w sprawozdaniu zauważono, że źle wylosowane wagi zwiększały też ilość potrzebnych iteracji. Pokazuje to znaczenie wag w nauczaniu perceptronu jest kluczowe.

Podczas każdego wykonanego testowania, nie wykryto błędu w obliczeniach perceptronu. Może być to spowodowane faktem, iż funkcje logiczne są jednymi z najprostszych operacji i prawdopodobieństwo otrzymania dobrego wyniku jest równe 50%.

Wnioski:

Przeprowadzone zadanie pokazało jak bardzo działanie perceptronu jest uzależnione od odpowiednich wag, ilości danych uczących oraz stałej uczącej, dlatego bardzo ważne jest prawidłowe dobranie tych współczynników. Nieodpowiednie dobranie może skutkować nie tylko wydłużeniem czasu uczenia perceptronu, ale też błędnymi wynikami.

Listing kodu:

perceptron.h

```
class Neuron{
private:
	double x1;
	double x2;
public:
	double w1;
	double w2;
public:

	double fun_act(double );
	void losuj_wagi();
	void set_w1( double );
	void set_w2( double );
	void set_x1( double );
	void set_x2( double );
	double get_w1(  );
	double get_w2(  );
	double get_x1(  );
	double get_x2(  );
	double suma();
};
```

perceptron.cpp

```
#include <iostream>
#include <time.h>
#include "perceptron.h"
using namespace std;

void Neuron::set_w1( double n1 )
    {
        w1 = n1;
    }
void Neuron::set_w2( double n2 )
    {
        w2 = n2;
    }
void Neuron::set_x1( double a1 )
    {
        x1 = a1;
    }
void Neuron::set_x2( double a2 )
    {
        x2 = a2;
    }
double Neuron::get_w1(){
              return w1;
       }
       double Neuron::get_w2(){
              return w2;
       }
       double Neuron::get_x1(){
              return x1;
       }
       double Neuron::get_x2(){
              return x2;
       }
double Neuron::fun_act(double suma){
       return (2/(1+exp(-2*1*suma)) - 1); //B - wspolczynnik uczenia (0.1 lub 0.5), tangens
hiperboliczny - zwraca wartosci od -1 do 1
}
void Neuron::losuj_wagi(){
              srand(time(NULL));
                     double w_1=(rand() % 100);//przedzial wag dla uproszczenia od -2 do 2
                     double w_2=(rand() % 100);
                     set_w1(w_1/100);
                     set_w2(w_2/100);
       }
double Neuron::suma(){
       double sum;
       sum=(get_x1()*get_w1())+(get_x2()*get_w2());
       return sum;
}
```

source.cpp

```
#include <iostream>

#include "perceptron.h"
#include <fstream>
#include <cstdlib>
#include <string>
#include <iomanip>

using namespace std;

void wczytaj_dane(Neuron n1,double **tw, int n);
double korekcja_w1(double p,Neuron n1);
double korekcja_w2(double p,Neuron n1);


int main(){
	Neuron n1;
	n1.losuj_wagi();
	int n;
	cout<<"Podaj ilosc danych do 20: ";
	cin>>n;
	cout<<endl;
	double **tab_wejscia = new double *[n];
	for(int i=0; i<n;i++){
		tab_wejscia[i]=new double [3];
	}
	cout<<"WAGI poczatkowe: ";
	cout<<n1.get_w1()<<" "<<n1.get_w2()<<endl;
	cout<<"Dane z pliku:\n";
	wczytaj_dane(n1,tab_wejscia,n);
	bool warunek = true;
	int i=1;
	while(warunek){
		cout<<"_____ EPOKA
"<<i<<"_____"<<endl;
		int licz=0;
		for(int j=0;j<n;j++)
		{
			//cout<<" X1 przed : "<<n1.get_x1()<<endl;
			//cout<<" X2 przed : "<<n1.get_x2()<<endl;
			n1.set_x1(tab_wejscia[j][0]);
			n1.set_x2(tab_wejscia[j][1]);
			//cout<<" X1 po : "<<n1.get_x1()<<endl;
			//cout<<" X2 po : "<<n1.get_x2()<<endl;

			if((fabs(n1.fun_act(n1.suma())-
tab_wejscia[j][2])>0.99)||(fabs(tab_wejscia[j][2]-n1.fun_act(n1.suma()))<0.1))
			{
				licz++;
				cout<<"Funkcja aktywacji: "<<n1.fun_act(n1.suma())<<"  Rekord
nauczony !"<<endl;
				if(licz==n){
					cout<<"Neuron nauczyl sie za "<<i<<" razem"<<endl;
					warunek=false;}
			}
			else
			{	cout<<"Funkcja aktywacji: "<<n1.fun_act(n1.suma());
				korekcja_w1(tab_wejscia[j][2]-n1.fun_act(n1.suma()),n1);
				korekcja_w1(tab_wejscia[j][2]-n1.fun_act(n1.suma()),n1);
```

```
                        n1.set_w1(n1.get_w1()+korekcja_w1(tab_wejscia[j][2]-
n1.fun_act(n1.suma()),n1));
                        n1.set_w2(n1.get_w2()+korekcja_w2(tab_wejscia[j][2]-
n1.fun_act(n1.suma()),n1));
                        cout<<" w1: "<<n1.get_w1()<<" , w2: "<<n1.get_w2()<<endl;


                }
            }
            i++;
        }
system("PAUSE");
}

void wczytaj_dane(Neuron n1,double **tw, int n){

        int z=0,j=0,nr_lini=0;
        string linia,pom;
        fstream plik;

        plik.open("LAB1_dane.txt", ios::in);

        if(plik.good()==false)
        {
                cout<<"Blad wczytywania pliku !!!"<<endl;
                exit(0);
        }

        while(getline(plik,linia))
        {
                for(unsigned int i=0; i < linia.length();i++)
                        {
                                if((linia[i]==' ') && (j==0))
                                {
                                        pom.insert(0,linia,z,i);
                                        tw[nr_lini][0]=(atof(pom.c_str()));
                                        j++;
                                        z=i;
                                        pom.clear();
                                }
                                else
                                        if((linia[i]==' ') && (j==1))
                                {
                                        pom.insert(0,linia,z,i-z);
                                        tw[nr_lini][1]=(atof(pom.c_str()));
                                        j++;
                                        z=i;
                                        pom.clear();
                                }
                                else
                                if(i==linia.length()-1 && j==2)
                                {
                                        pom.insert(0,linia,z,i-z+1);
                                        tw[nr_lini][2]=atoi(pom.c_str());
                                        j=0;
                                        z=0;
                                        pom.clear();
                                }
                        }
                nr_lini++;
                if(nr_lini==n){break;}
                }
```

```
		for(int i=0;i<n;i++){
			for(int j=0;j<3;j++)
				cout<< tw[i][j]<<"  ";
		cout<<endl;
		}
}
double korekcja_w1(double r,Neuron n1){
	//cout<<"Funkcja aktywacji: "<<n1.fun_act(n1.suma());
	double c=1;//stala uczenia
	double dw1=c*r*n1.get_x1();//o ile sie zmieni waga1
	//cout<<"  W1 przed: "<<n1.w1;
	n1.w1=n1.w1+dw1;
	//cout<<"  W1 po: "<<n1.w1;
		return  dw1;
}
double korekcja_w2(double r,Neuron n1){
	//cout<<"Funkcja aktywacji: "<<n1.fun_act(n1.suma());
	double c=1;//stala uczenia
	double dw2=c*r*n1.get_x2();//o ile sie zmieni waga2
	//cout<<"  W2 przed: "<<n1.w2;
	n1.w2=n1.w2+dw2;
	//cout<<"  W2 po: "<<n1.w2<<endl;
		return  dw2;
}
```