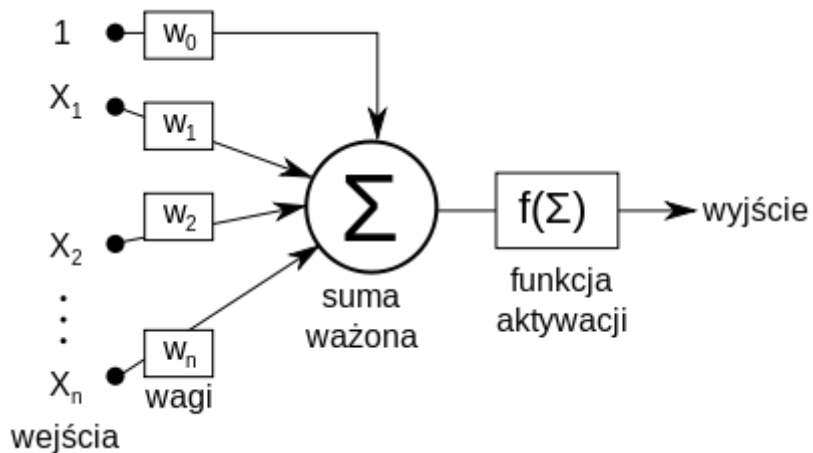


## Sprawozdanie 2 – Budowa i działanie sieci jednowarstwowej

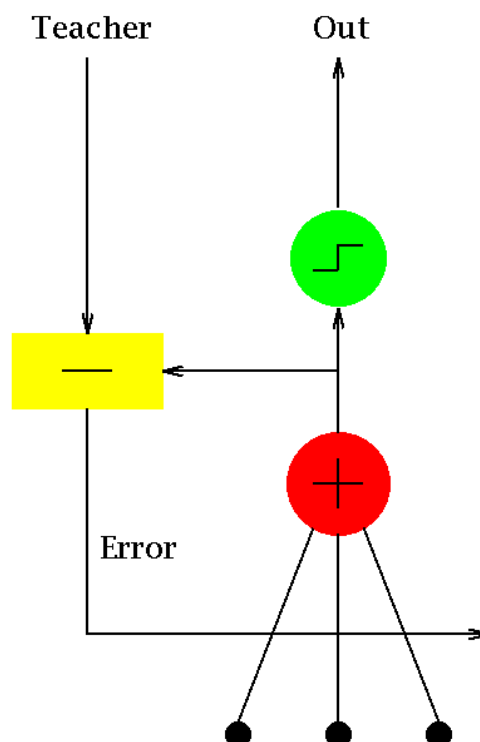
### 1. Wstęp

W ćwiczeniu zostały zaimplementowane sieci jednowarstwowe oparte o model perceptronu McCullocha-Pittsa oraz neuron Adaline.

a) Schemat neuronu McCullocha-Pittsa:



b) Schemat uczenia w neuronie Adaline:



Sieci zbudowane zostały z siedmiu neuronów, gdzie sześć z nich tworzy pierwszą warstwę a siódmy jest neuronem wyjściowym. Warstwa pierwsza dostaje 7 wejść, gdzie pierwsze jest biasem, a następnie jej wyniki zostają przesłane do neuronu wyjściowego. Do nauki

neuronów użyto algorytmu Widrowskiego-Hoffa. Polega on na dobraniu wag początkowych losowo, podaniu wektora uczącego, obliczeniu wartości wyjściowej perceptronu, porównaniu wartości wyjściowej z oczekiwaną a następnie dokonania modyfikacji według zależności danej wzorami:  $w_{i+1} = w_i + (d_i - y_i) * lr * x_i$

## 2. Cel ćwiczenia

Celem ćwiczenia było poznanie budowy i działania jednowarstwowych sieci neuronowych oraz uczenie rozpoznawania wielkości liter.

## 3. Przebieg ćwiczenia

a) wygenerowanie danych uczących i testujących, zawierających 10 dużych i 10 małych liter w postaci dwuwymiarowej tablicy 5x7. Reprezentacja liter została podzielona na 6 obszarów, które określały wejściowy wektor uczący. Jeżeli w danym obszarze występowała część litery, to sygnał przyjmował wartość 1 a jeśli nie – 0.

b) implementacja dwóch różnych jednowarstwowych sieci – Madaline oraz sieć zbudowaną z perceptronów na podstawie modelu McCullocha-Pittsa

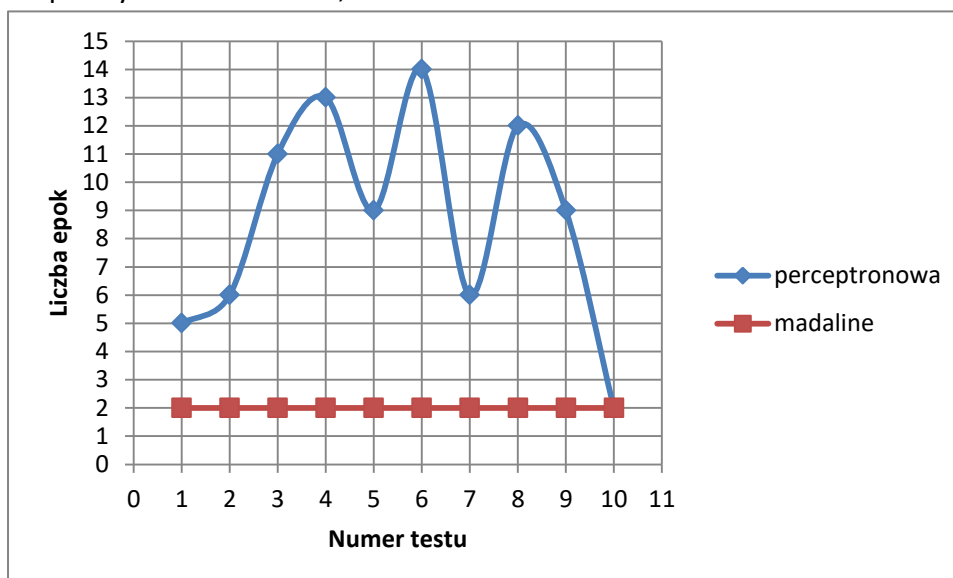
c) uczenie sieci przy różnych współczynnikach uczenia.

d) testowanie sieci

## 4. Przedstawienie i analiza wyników

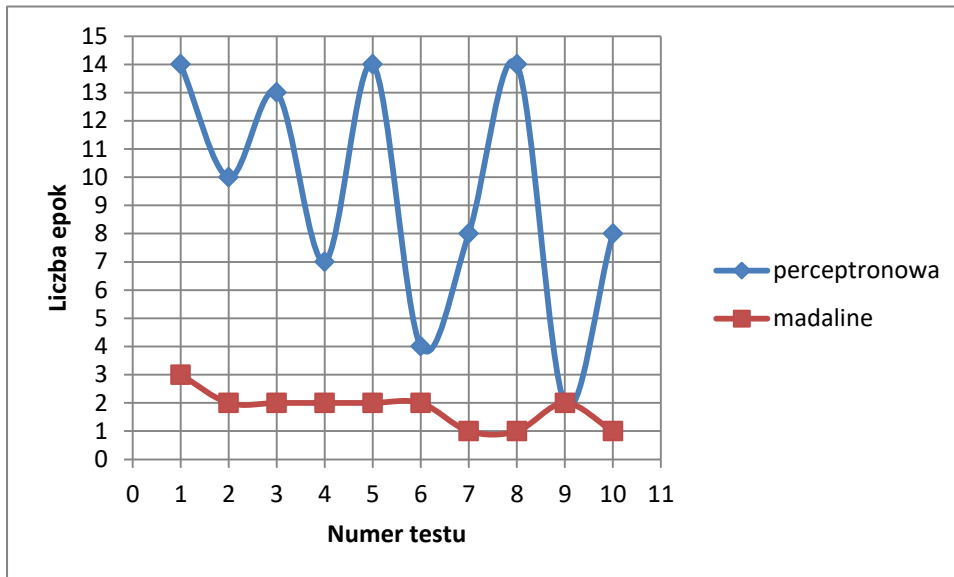
Wykresy zależności liczby epok potrzebnych do nauczenia sieci w kolejnych testach przy losowo generowanych wagach, dla różnych współczynników uczenia:

a) Współczynnik uczenia = 0,1



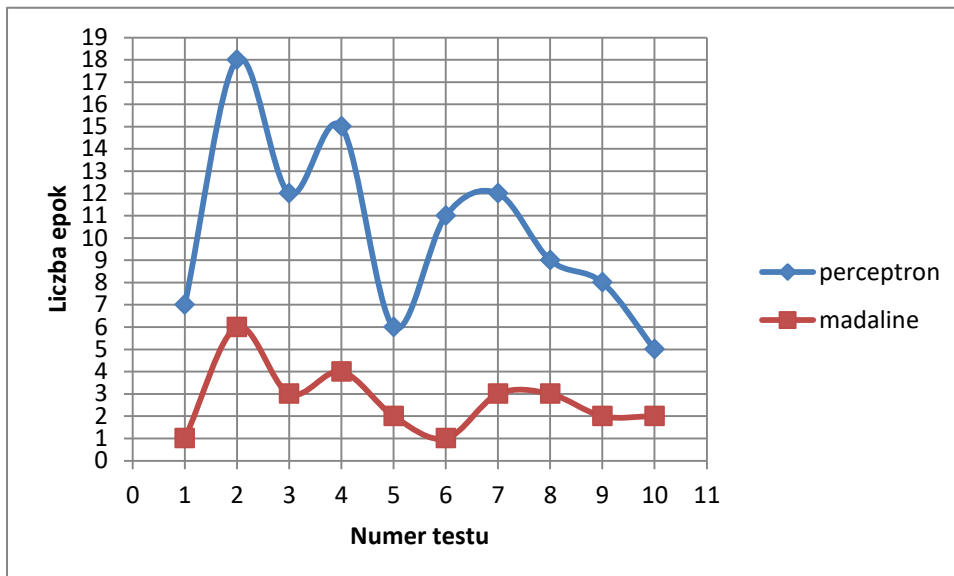
Na powyższym wykresie widzimy, że dla sieci opartej o adaline dla kroku uczenia 0,1 liczba epok jest stała, zaś dla sieci perceptronowej wyniki są zróżnicowane.

b) Współczynnik uczenia = 0,07



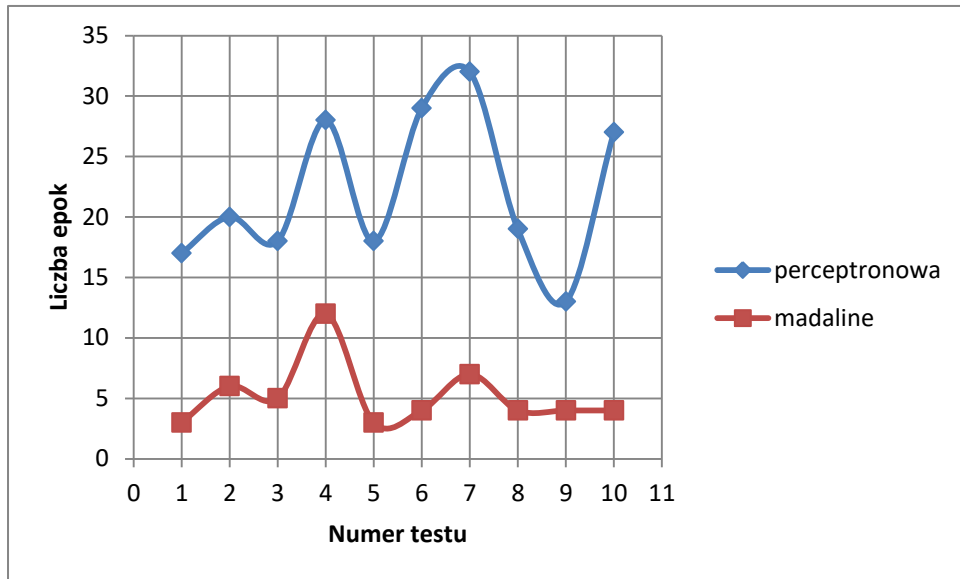
W tym przypadku sieć oparta o adaline osiąga kolejny raz świetne wyniki na poziomie 1 do 3 epok. Sieć perceptronowa zaś kolejny raz osiąga zróżnicowane wyniki.

c) Współczynnik uczenia = 0,03



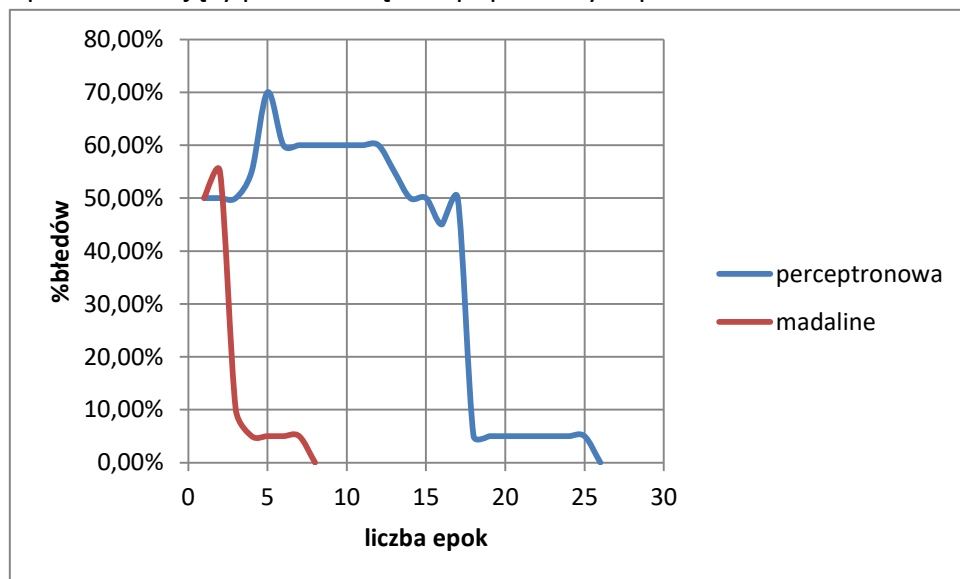
W tym przypadku sieć madaline zaczyna osiągać większy zakres potrzebnych epok. Liczba epok rośnie w obydwu sieciach.

d) Współczynnik uczenia = 0,01



Na tym wykresie dalej obserwujemy wzrost liczby epok zarówno w sieci madaline jak i w sieci perceptronowej.

Wykres przedstawiający procent błędów popełnianych przez sieć w zależności od epoki:

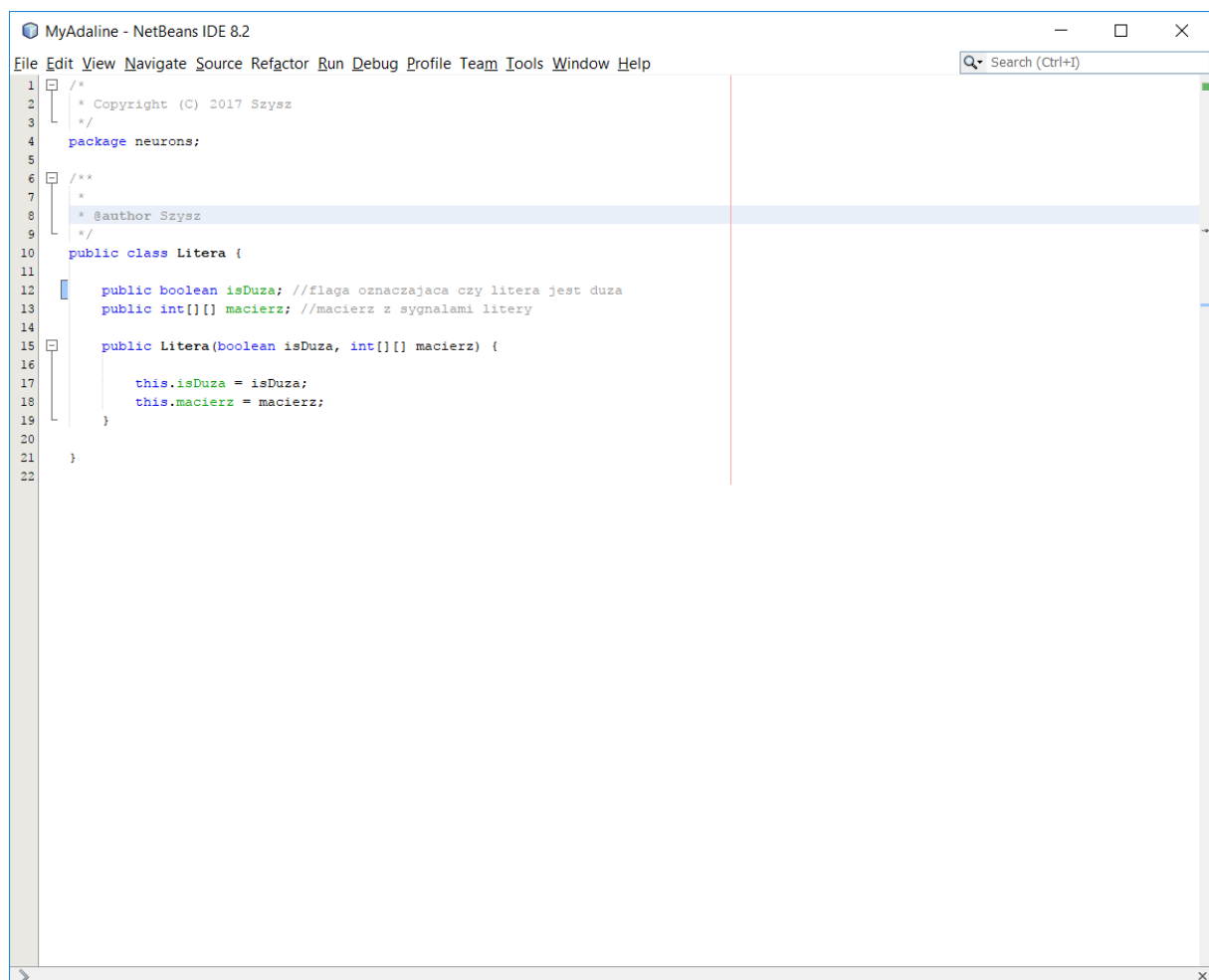


Na powyższym wykresie można zaobserwować, że obydwie sieci na początku popełniają błędy na poziomie 50%, lecz sieć madaline bardzo szybko niweluje je w kolejnych epokach. Sieć perceptronowa z kolei potrzebuje więcej czasu na osiągnięcie bezbłędnych rezultatów.

## 5. Wnioski

Na podstawie otrzymanych wyników można łatwo stwierdzić, że sieć neuronowa oparta o neuron Adaline, działa dużo wydajniej niż sieć perceptronowa. Przy każdej wartości współczynnika uczenia Adaline uzyskiwało lepsze wyniki. Wyjaśnić ten fakt może nam wykres przedstawiający procent popełnionych błędów uczenia. Adaline niwelował błędy bardzo szybko co wpływało na szybkość uczenia. Wywnioskować można zatem, że modyfikowanie wag przed funkcją aktywacji jest lepsze. I rzeczywiście: w przypadku dużego błędzi wagi zostaną znacznie zmodyfikowane, zaś przy małym błędzie wagi również zostaną zmodyfikowane w niewielkim stopniu. Wpływ na dokładność modyfikowania wag, może mieć również fakt, iż w przypadku sieci perceptronowej korzystamy z sygnałów wejściowych 0 lub 1 (wagi mogą być tylko zwiększane), a w przypadku madaline -1 lub 1 (wagi mogą być zwiększane i zmniejszane).

## 6. Listing kodu



```
1  /*
2   * Copyright (C) 2017 Szysz
3   */
4  package neurons;
5
6  /**
7   *
8   * @author Szysz
9   */
10 public class Litera {
11
12     public boolean isDuza; //flaga oznaczajaca czy litera jest duza
13     public int[][] macierz; //macierz z sygnalami litery
14
15     public Litera(boolean isDuza, int[][] macierz) {
16
17         this.isDuza = isDuza;
18         this.macierz = macierz;
19     }
20
21 }
22
```

```
MyAdaline - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)

1  /**
2   * Copyright (C) 2017 Szysz
3   */
4   package neurons;
5
6   import java.util.Arrays;
7
8   /**
9   *
10  * @author Szysz
11  */
12  public class MojeLiterary {
13
14      private final Litera[] literary;
15
16      public MojeLiterary() { //wektory wejsciowe liter
17
18          literary = new Litera[20];
19
20          //wielkie literary
21          literary[0] = new Litera(true, new int[][] { //A
22              {0, 0, 1, 0, 0}, {0, 1, 0, 1, 0}, {1, 0, 0, 0, 1}, {1, 1, 1, 1, 1}, {1, 0, 0, 0, 1}, {1, 0, 0, 0, 1}, {1, 0, 0, 0, 1}
23          });
24
25          literary[1] = new Litera(true, new int[][] { //B
26              {1, 1, 1, 1, 0}, {1, 0, 0, 0, 1}, {1, 0, 0, 0, 1}, {1, 1, 1, 1, 0}, {1, 0, 0, 0, 1}, {1, 0, 0, 0, 1}, {1, 1, 1, 1, 0}
27          });
28
29          literary[2] = new Litera(true, new int[][] { //C
30              {0, 1, 1, 1, 1}, {1, 0, 0, 0, 0}, {1, 0, 0, 0, 0}, {1, 0, 0, 0, 0}, {1, 0, 0, 0, 0}, {1, 0, 0, 0, 0}, {0, 1, 1, 1, 1}
31          });
32
33          literary[3] = new Litera(true, new int[][] { //D
34              {1, 1, 1, 1, 0}, {1, 0, 0, 0, 1}, {1, 0, 0, 0, 1}, {1, 0, 0, 0, 1}, {1, 0, 0, 0, 1}, {1, 0, 0, 0, 1}, {1, 1, 1, 1, 0}
35          });
36
37          literary[4] = new Litera(true, new int[][] { //E
38              {1, 1, 1, 1, 1}, {1, 0, 0, 0, 0}, {1, 0, 0, 0, 0}, {1, 1, 1, 1, 1}, {1, 0, 0, 0, 0}, {1, 0, 0, 0, 0}, {1, 1, 1, 1, 1}
39          });
40
41          literary[5] = new Litera(true, new int[][] { //F
42              {1, 1, 1, 1, 1}, {1, 0, 0, 0, 0}, {1, 0, 0, 0, 0}, {1, 1, 1, 1, 1}, {1, 0, 0, 0, 0}, {1, 0, 0, 0, 0}, {1, 0, 0, 0, 0}
43          });
44
45          literary[6] = new Litera(true, new int[][] { //G
46              {0, 1, 1, 1, 1}, {1, 0, 0, 0, 0}, {1, 0, 0, 0, 0}, {1, 0, 0, 1, 1}, {1, 0, 0, 0, 1}, {1, 0, 0, 0, 1}, {0, 1, 1, 1, 1}
47          });
48      }
49  }
```

```
MyAdaline - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)

49      literary[7] = new Litera(true, new int[][] { //H
50          {1, 0, 0, 0, 1}, {1, 0, 0, 0, 1}, {1, 0, 0, 0, 1}, {1, 1, 1, 1, 1}, {1, 0, 0, 0, 1}, {1, 0, 0, 0, 1}, {1, 0, 0, 0, 1}
51      });
52
53      literary[8] = new Litera(true, new int[][] { //I
54          {0, 0, 1, 0, 1}, {0, 0, 1, 0, 0}, {0, 0, 1, 0, 0}, {0, 0, 1, 0, 0}, {0, 0, 1, 0, 0}, {0, 0, 1, 0, 0}, {0, 0, 1, 0, 0}
55      });
56
57      literary[9] = new Litera(true, new int[][] { //J
58          {0, 1, 1, 1, 0}, {0, 0, 0, 1, 0}, {0, 0, 0, 1, 0}, {0, 0, 0, 1, 1}, {0, 0, 1, 1, 0}, {0, 1, 0, 1, 0}, {0, 0, 1, 0, 0}
59      });
60
61      //male literary
62      literary[10] = new Litera(false, new int[][] { //a
63          {0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}, {0, 1, 1, 1, 0}, {1, 0, 0, 1, 0}, {1, 0, 0, 1, 0}, {0, 1, 1, 1, 1}
64      });
65
66      literary[11] = new Litera(false, new int[][] { //b
67          {1, 0, 0, 0, 0}, {1, 0, 0, 0, 0}, {1, 0, 0, 0, 0}, {1, 1, 1, 0, 0}, {1, 0, 0, 1, 0}, {1, 0, 0, 1, 0}, {1, 1, 1, 0, 0}
68      });
69
70      literary[12] = new Litera(false, new int[][] { //c
71          {0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}, {0, 1, 1, 1, 0}, {1, 0, 0, 0, 0}, {1, 0, 0, 0, 0}, {0, 1, 1, 1, 0}
72      });
73
74      literary[13] = new Litera(false, new int[][] { //d
75          {0, 0, 0, 1, 0}, {0, 0, 0, 1, 0}, {0, 0, 0, 1, 0}, {0, 1, 1, 1, 0}, {1, 0, 0, 1, 0}, {1, 0, 0, 1, 0}, {0, 1, 1, 1, 0}
76      });
77
78      literary[14] = new Litera(false, new int[][] { //e
79          {0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}, {0, 1, 1, 0, 0}, {1, 0, 1, 0, 0}, {1, 1, 0, 0, 0}, {0, 1, 1, 1, 0}
80      });
81
82      literary[15] = new Litera(false, new int[][] { //f
83          {0, 0, 0, 0, 0}, {0, 1, 1, 0, 0}, {1, 0, 0, 0, 0}, {1, 1, 1, 0, 0}, {1, 0, 0, 0, 0}, {1, 0, 0, 0, 0}, {1, 0, 0, 0, 0}
84      });
85
86      literary[16] = new Litera(false, new int[][] { //g
87          {0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}, {0, 0, 1, 0, 0}, {0, 1, 0, 1, 0}, {0, 0, 1, 1, 0}, {0, 0, 0, 1, 0}, {0, 1, 1, 0, 0}
88      });
89
90      literary[17] = new Litera(false, new int[][] { //h
91          {1, 0, 0, 0, 0}, {1, 0, 0, 0, 0}, {1, 0, 0, 0, 0}, {1, 1, 1, 0, 0}, {1, 0, 1, 0, 0}, {1, 0, 1, 0, 0}, {1, 0, 1, 0, 0}
92      });
93
94      literary[18] = new Litera(false, new int[][] { //i
95          {0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}, {0, 0, 1, 0, 0}, {0, 0, 0, 0, 0}, {0, 0, 1, 0, 0}, {0, 0, 1, 0, 0}
96      });
97  }
```

neurons.MojeLiterary

```
MyAdaline - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)

97     litery[19] = new Litera(false, new int[][]{
98         {0, 0, 0, 0, 0}, {0, 0, 1, 0, 0}, {0, 0, 0, 0, 0}, {0, 0, 1, 0, 0}, {0, 0, 1, 0, 0}, {0, 0, 1, 0, 0}, {0, 1, 1, 0, 0}
99     });
100
101 }
102
103 public Litera[] getLitery() {
104     return litery;
105 }
106
107 //funkcja zmieniajaca litere na wektor uczacy i zwracajaca go
108 public int[] getLearningVektorOfLitera(int index) {
109
110     int[] wektor = new int[7];
111     Arrays.fill(wektor, 1, 6, 0);
112     wektor[0] = 1; //bias
113
114     //reprezentacja litery jest dzielona na 6 obszarow
115     //obszar 1
116     for (int i = 0; i < 3; i++) {
117         for (int j = 0; j < 2; j++) {
118             if (litery[index].macierz[i][j] == 1) {
119                 wektor[1] = 1;
120                 //break;
121             }
122         }
123     }
124     //obszar 2
125     for (int i = 0; i < 3; i++) {
126         for (int j = 2; j < 3; j++) {
127             if (litery[index].macierz[i][j] == 1) {
128                 wektor[2] = 1;
129                 //break;
130             }
131         }
132     }
133     //obszar 3
134     for (int i = 0; i < 3; i++) {
135         for (int j = 3; j < 5; j++) {
136             if (litery[index].macierz[i][j] == 1) {
137                 wektor[3] = 1;
138                 //break;
139             }
140         }
141     }
142     //obszar 4
143     for (int i = 3; i < 7; i++) {
144         for (int j = 0; j < 2; j++) {
```

```
MyAdaline - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)

142     //obszar 4
143     for (int i = 3; i < 7; i++) {
144         for (int j = 0; j < 2; j++) {
145             if (litery[index].macierz[i][j] == 1) {
146                 wektor[4] = 1;
147                 //break;
148             }
149         }
150     }
151     //obszar 5
152     for (int i = 3; i < 7; i++) {
153         for (int j = 2; j < 3; j++) {
154             if (litery[index].macierz[i][j] == 1) {
155                 wektor[5] = 1;
156                 //break;
157             }
158         }
159     }
160     //obszar 6
161     for (int i = 3; i < 7; i++) {
162         for (int j = 3; j < 5; j++) {
163             if (litery[index].macierz[i][j] == 1) {
164                 wektor[6] = 1;
165                 //break;
166             }
167         }
168     }
169     return wektor;
170 }
171 }
172
173 }
```

```
MyAdaline - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)

12 public class Perceptron {
13
14     private final int rozmiar; //liczba wejsc
15     private final double wage[]; //wagi
16
17     public Perceptron(int n) {
18         rozmiar = n;
19         wage = new double[n];
20         Random rand = new Random();
21         for (int i = 0; i < n; i++) {
22             wage[i] = rand.nextDouble();
23             // wage[i] = 0.1;
24         }
25     }
26
27
28     private int activate(double y) { //funkcja aktywacji
29         if (y < 0) {
30             return 0;
31         } else {
32             return 1;
33         }
34     }
35
36     public int process(int[] x) { //funkcja sumujaca
37         double y = 0;
38         for (int i = 0; i < rozmiar; i++) {
39             y += x[i] * wage[i];
40         }
41
42         return activate(y);
43     }
44
45     public double learn(int xl[], double d, double learnRate) { //funkcja uczaca
46         double y = process(xl);
47         double error = 0;
48         for (int i = 0; i < rozmiar; i++) {
49             wage[i] += (d - y) * learnRate * xl[i];
50             error += Math.abs(d - y); //blad == |wartosc oczekiwana - wartosc otrzymana|
51         }
52         return error;
53     }
54 }
```

```
MyAdaline - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)

12 public class Adaline {
13
14     private final int liczbaWejsc; //liczba wejsc
15     private final double w[]; //wagi
16
17     public Adaline(int liczbaWejsc) {
18         this.liczbaWejsc = liczbaWejsc;
19         w = new double[this.liczbaWejsc];
20
21         for (int i = 0; i < this.liczbaWejsc; i++) {
22             w[i] = new Random().nextDouble(); //wagi początkowe sa losowane
23         }
24     }
25
26     //funkcja aktywacji
27     public int f(double s) {
28         if (s > 0) {
29             return 1;
30         } else {
31             return -1;
32         }
33     }
34
35     //funkcja sumujaca
36     public double process(int[] x) {
37         double y = 0;
38         for (int i = 0; i < liczbaWejsc; i++) {
39             y += x[i] * w[i];
40         }
41
42         return y;
43     }
44
45     //funkcja uczaca
46     public void learn(int[] x, double d, double learnRate) {
47         double y = process(x);
48         for (int i = 0; i < liczbaWejsc; i++) {
49             w[i] += (d - y) * learnRate * x[i]; //modyfikacja wag
50         }
51     }
52
53     //funkcja testujaca
54     public int test(int[] x) {
55         double result = process(x);
56         return f(result);
57     }
58
59 }
60
61
62 }
```



```
MyAdaline - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)

14 public class PerceptronMain {
15
16     public static void main(String[] args) {
17
18         int liczbaWejsc = 7;           //ilość wejść
19         int liczbaLiter = 20;
20         int epochCounter = 0;         //licznik ilości epok uczenia się
21         double learnRate = 0.01;      //współczynnik uczenia
22
23         Perceptron[] perceptrons = new Perceptron[liczbaWejsc];
24         for (int i = 0; i < liczbaWejsc; i++) {
25             perceptrons[i] = new Perceptron(liczbaWejsc);
26         }
27
28         int[] y = new int[liczbaLiter];
29         Arrays.fill(y, 0, liczbaLiter / 2, 0); //przygotowanie tablicy wartosci oczekiwanej
30         Arrays.fill(y, liczbaLiter / 2, liczbaLiter, 1); // 0 duża litera, 1 mała litera
31
32         int[] result = new int[liczbaLiter]; //tablica przechowująca wyniki testowania perceptronow
33         Arrays.fill(result, 0, liczbaLiter, 0);
34
35         while (!Arrays.equals(y, result)) { //dopoki wynik jest rozny od wartosci oczekiwanej ucz siec
36
37             for (int i = 0; i < liczbaLiter; i++) {
38                 learn(perceptrons, liczbaWejsc, learnRate, i);
39             }
40
41             result = test(perceptrons, liczbaLiter, liczbaWejsc);
42             epochCounter++;
43         }
44
45         System.out.println("Ilość kroków do nauczzenia się = " + epochCounter);
46     }
47 }
```

```
MyAdaline - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)

47
48     public static double learn(Perceptron[] perceptrons, int liczbaWejsc, double learnRate, int i) { //zwraca ilosc bladów
49
50         MojeLitery mojeLitery = new MojeLitery();
51         int[] wektor; //tablica przechowująca wektor sygnałów wejściowych do uczenia pierwszej warstwy sieci
52         wektor = mojeLitery.getLearningWektorOfLitera(i);
53
54         int[] outputWektor = new int[liczbaWejsc]; //tablica przechowująca wektor sygnałów wyjściowych pierwszej warstwy sieci
55         outputWektor[0] = 1; //bias
56
57         int isDuza; //zmienna przechowująca wartosc oczekiwana
58         if (mojeLitery.getLitera()[i].isDuza) {
59             isDuza = 0;
60         } else {
61             isDuza = 1;
62         }
63
64         for (int k = 0; k < liczbaWejsc - 1; k++) { //uczenie pierwszej warstwy
65             perceptrons[k].learn(wektor, isDuza, learnRate);
66             outputWektor[k + 1] = perceptrons[k].process(wektor); //pobranie sygnału wyjściowego
67         }
68         double blad = perceptrons[liczbaWejsc - 1].learn(outputWektor, isDuza, learnRate); //uczenie neuronu wynikowego na podstawie
69         return blad;
70     }
71
72     private static int[] test(Perceptron[] adaline, int liczbaLiter, int liczbaWejsc) {
73
74         MojeLitery mojeLitery = new MojeLitery();
75         int[] result = new int[liczbaLiter];
76         int[] wektor; //tablica przechowująca wektor sygnałów wejściowych do testowania pierwszej warstwy sieci
77         int[] wektorWyjscowy = new int[liczbaWejsc]; //tablica przechowująca wektor sygnałów wyjściowych pierwszej warstwy sieci
78         wektorWyjscowy[0] = 1; //bias
79
80         for (int i = 0; i < liczbaLiter; i++) { //testowanie, celem upewnienia się, czy sieć już nauczona
81
82             wektor = mojeLitery.getLearningWektorOfLitera(i); //pobierz wektor uczący tej litery
83
84             for (int k = 0; k < liczbaWejsc - 1; k++) {
85                 wektorWyjscowy[k + 1] = adaline[k].process(wektor);
86             }
87
88             result[i] = adaline[liczbaWejsc - 1].process(wektorWyjscowy); //wpisz otrzymany wynik do tablicy wyniku
89         }
90         return result;
91     }
92 }
93 }
```

```
MyAdaline - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)

14 public class AdalineMain {
15
16     public static void main(String[] args) {
17         int liczbaWejsc = 7; //ilość wejść
18         int liczbaLiter = 20; //liczba liter
19         int epochCounter = 0; //licznik ilości epok uczenia się
20         double learnRate = 0.01; //współczynnik uczenia
21
22         Adaline[] adaline = new Adaline[liczbaWejsc];
23         for (int i = 0; i < liczbaWejsc; i++) {
24             adaline[i] = new Adaline(liczbaWejsc);
25         }
26
27         int[] y = new int[liczbaLiter]; //przygotowanie tablicy wartosci oczekiwanej
28         Arrays.fill(y, 0, liczbaLiter / 2, -1); // -1 duza litera, 1 mala litera
29         Arrays.fill(y, liczbaLiter / 2, liczbaLiter, 1);
30
31         int[] result = new int[liczbaLiter]; //tablica przechowująca wyniki testowania adaline
32         Arrays.fill(result, 0, liczbaLiter, 0);
33
34         while (!Arrays.equals(y, result)) { //dopoki wynik jest rozny od wartosci oczekiwanej ucz siec
35
36             for (int i = 0; i < liczbaLiter; i++) {
37                 learn(adaline, liczbaWejsc, learnRate, i);
38             }
39
40             result = test(adaline, liczbaLiter, liczbaWejsc);
41             epochCounter++;
42         }
43
44         System.out.println("Ilość kroków do nauczania się = " + epochCounter);
45     }
46 }

MyAdaline - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)

47 private static void learn(Adaline[] adaline, int liczbaWejsc, double learnRate, int i) {
48
49     MojeLittery mojeLittery = new MojeLittery();
50     int[] wektor; //tablica przechowująca wektor sygnałów wejściowych do uczenia pierwszej warstwy sieci
51     wektor = mojeLittery.getLearningVektorOfLittera(i); //pobierz wektor uczący i-tej litery
52     format(wektor); //zamien w wektorze 0 na -1
53
54     int[] outputWektor = new int[liczbaWejsc]; //tablica przechowująca wektor sygnałów wyjściowych pierwszej warstwy sieci
55     outputWektor[0] = 1; //bias
56
57     int isDuza; //zmienna przechowująca wartosc oczekiwana
58     if (mojeLittery.getLittery()[i].isDuza) {
59         isDuza = -1;
60     } else {
61         isDuza = 1;
62     }
63
64     for (int k = 0; k < liczbaWejsc - 1; k++) { //uczenie pierwszej warstwy
65         adaline[k].learn(wektor, isDuza, learnRate);
66         outputWektor[k + 1] = adaline[k].test(wektor); //pobranie sygnału wyjściowego
67     }
68     adaline[liczbaWejsc - 1].learn(outputWektor, isDuza, learnRate); //uczenie neuronu wynikowego na podstawie sygnałów wyjściowych pierwszej warstwy
69 }
70
71 private static int[] test(Adaline[] adaline, int liczbaLiter, int liczbaWejsc) {
72
73     MojeLittery mojeLittery = new MojeLittery();
74     int[] result = new int[liczbaLiter];
75     int[] wektor; //tablica przechowująca wektor sygnałów wejściowych do testowania pierwszej warstwy sieci
76     int[] wektorWyjscowy = new int[liczbaWejsc]; //tablica przechowująca wektor sygnałów wyjściowych pierwszej warstwy sieci
77     wektorWyjscowy[0] = 1; //bias
78
79     for (int i = 0; i < liczbaLiter; i++) { //testowanie, celem upewnienia się, czy siec już nauczona
80
81         wektor = mojeLittery.getLearningVektorOfLittera(i);
82         format(wektor);
83
84         for (int k = 0; k < liczbaWejsc - 1; k++) {
85             wektorWyjscowy[k + 1] = adaline[k].test(wektor);
86         }
87
88         result[i] = adaline[liczbaWejsc - 1].test(wektorWyjscowy);
89     }
90
91     return result;
92 }
93
94 //w przypadku adaline sygnały wejściowe = 0 muszą być zamienione na sygnały -1
95 private static void format(int[] wektor) {
96     for (int k = 0; k < wektor.length; k++) {
97         if (wektor[k] == 0) {
98             wektor[k] = -1;
99         }
100     }
101 }

main.AdalineMain learn
```

## 7. Bibliografia

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

[https://pl.wikipedia.org/wiki/Neuron\\_McCullocha-Pittsa](https://pl.wikipedia.org/wiki/Neuron_McCullocha-Pittsa)

[http://zsi.tech.us.edu.pl/~nowak/si/SI\\_w4.pdf](http://zsi.tech.us.edu.pl/~nowak/si/SI_w4.pdf)

<https://pl.wikipedia.org/wiki/Perceptron>