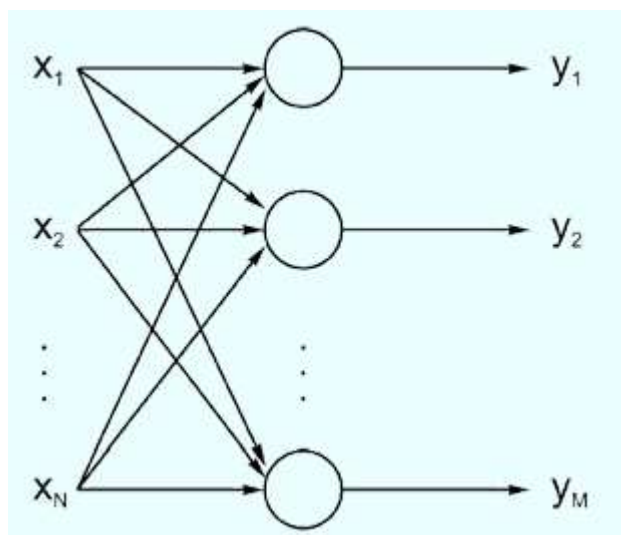


19.12.2017 r.	Izabela Musztyfaga	Podstawy Sztucznej Inteligencji
Scenariusz 2 - Budowa i działanie sieci jednowarstwowej		

Jednowarstwowa sieć neuronowa to sieć zbudowana z jednej warstwy neuronów. Każde z wielu wejść ($x_1 - x_N$) wchodzi do każdego z neuronów, natomiast każdy z neuronów przekazuje jedną wartość na wyjście ($y_1 - y_M$).

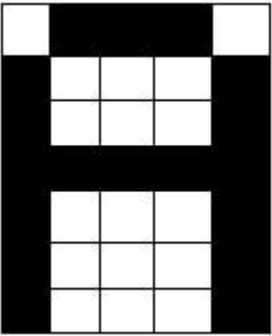
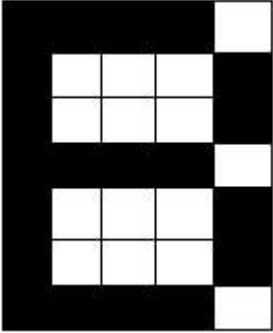
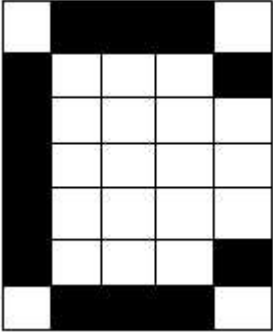


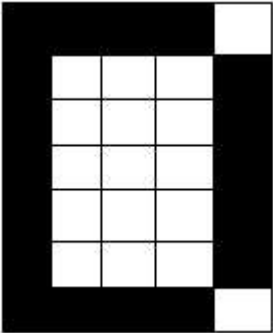
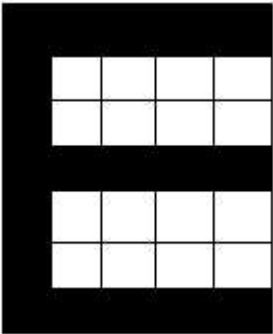
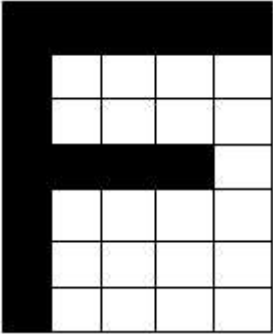
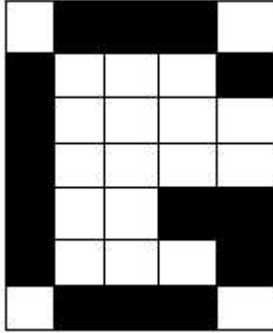
Rys. 1¹ – Jednowarstwowa sieć neuronowa

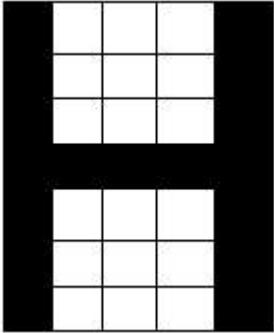
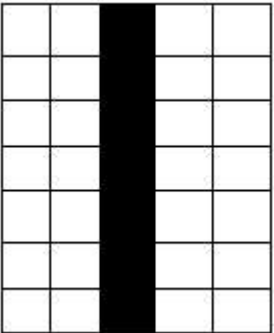
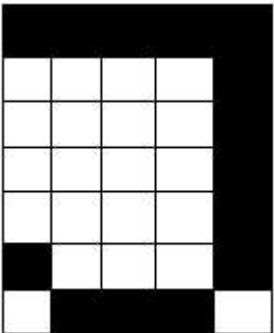
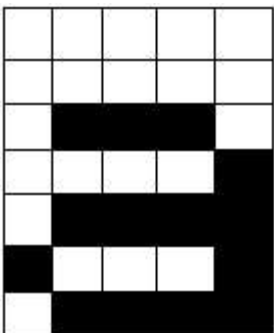
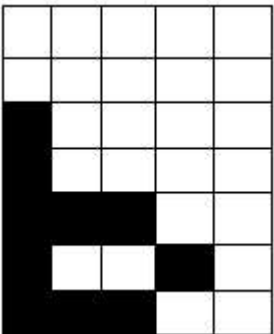
W tym ćwiczeniu należało nauczyć jednowarstwową sieć rozpoznawania wielkości liter. W tym celu zostały wykorzystano litery (10 wielkich oraz 10 małych) przedstawione graficznie na matrycach 5x7, które następnie zamieniono na 0 i 1, z czego finalnie powstały wektory. Każda wartość liczbową odpowiada jednemu pikselowi – jeżeli pole w matrycy jest puste przypisujemy mu wartość 0, jeżeli jest 'pełne' przypisujemy mu wartość 1.

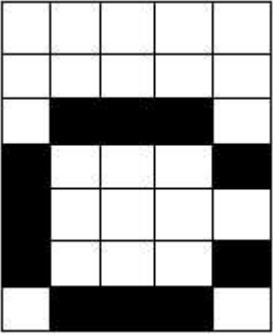
Każdy wektor jest unikalny, tzn. żaden się nie powtarza.

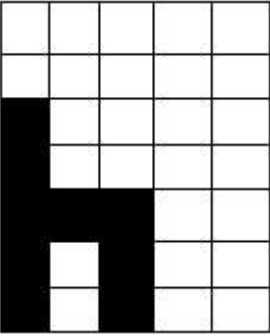
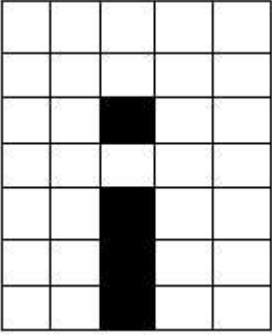
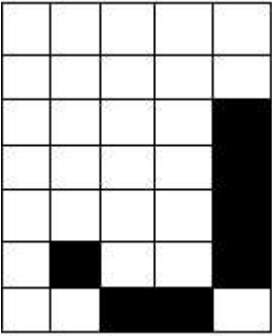
1 Źródło obrazu: http://galaxy.agh.edu.pl/~vlsi/AI/koho_t/siec.jpg

Litera przedstawiona graficznie	Litera po konwersji do 0 i 1	Wektor
	0 1 1 1 0 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	0 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 1 1 1 1 0 0 0 1 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 1 0 0 0 1 1 0 0 0 1 1 1 1 1 0	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
	0 1 1 1 0 1 0 0 0 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 1 1 1 0	0 1 1 1 0 1 0 0 0 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 1 1 1 0

	<pre>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 0 0 0 1 1 1 1 1 0</pre>	<pre>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</pre>
	<pre>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</pre>	<pre>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</pre>
	<pre>1 1 1 1 1 1 0 0 0 0 1 0 0 0 0 1 1 1 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0</pre>	<pre>1 1 1 1 1 1 0 0 0 0 1 0 0 0 0 1 1 1 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0</pre>
	<pre>0 1 1 1 0 1 0 0 0 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 1 1 1 0</pre>	<pre>0 1 1 1 0 1 0 0 0 1 1 0 0 0 0 1 0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 1 1 1 0</pre>

	<div>10001 10001 10001 11111 10001 10001 10001</div>	<div>10001100011000111111100011000110001</div>
	<div>00100 00100 00100 00100 00100 00100 00100 00100</div>	<div>0010000100001000010000100001000010000100</div>
	<div>11111 00001 00001 00001 00001 00001 10001 01110</div>	<div>1111100001000010000100001000011000101110</div>
	<div>00000 00000 01110 00001 01111 10001 01111</div>	<div>000000000000111000001011111000101111</div>
	<div>00000 00000 10000 10000 11100 10010 11100</div>	<div>000000000001000010000111001001011100</div>

	<div>00000</div> <div>00000</div> <div>01110</div> <div>10001</div> <div>10000</div> <div>10001</div> <div>01110</div>
---	--

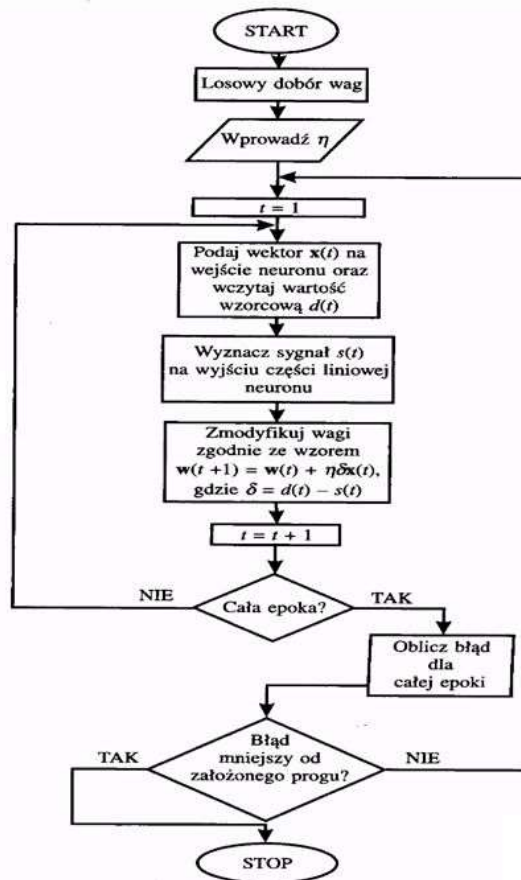
	00000 00000 00100 00000 00100 00100 00100	000000000001000010000111001010010100
	00000 00000 00100 00000 00100 00100 00100	000000000000010000000001000010000100
	00000 00000 00001 00001 00001 00001 01001 00110	0000000000000000100001000010100100110

Tab. 1 – Litery przedstawione graficznie, przy użyciu 0 i 1 oraz jako wektor

Algorytmy uczenia sieci

Adaline

Schemat blokowy
algorytmu uczenia
neuronu typu *adaline*.



Legenda:

- i-numer wagi neuronu,
- t-numer iteracji w epoce,
- d-sygnał wzorcowy,
- y-sygnał wyjściowy neuronu,
- s-sygnał wyjściowy sumatora neuronu,
- x-wartość wejściowa neuronu,
- η - współczynnik uczenia (0,1).

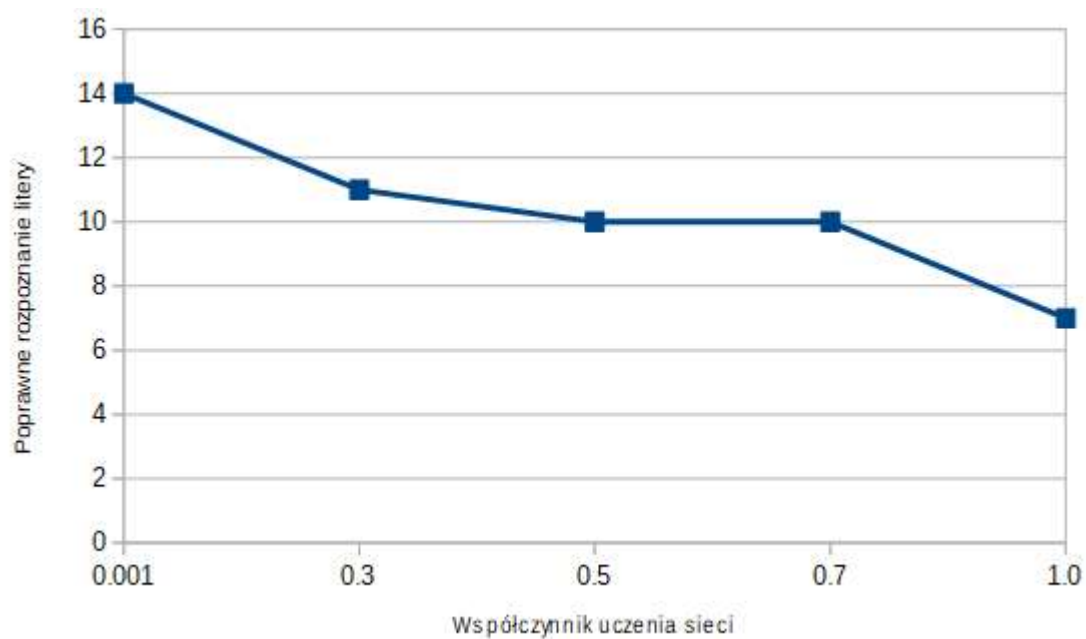
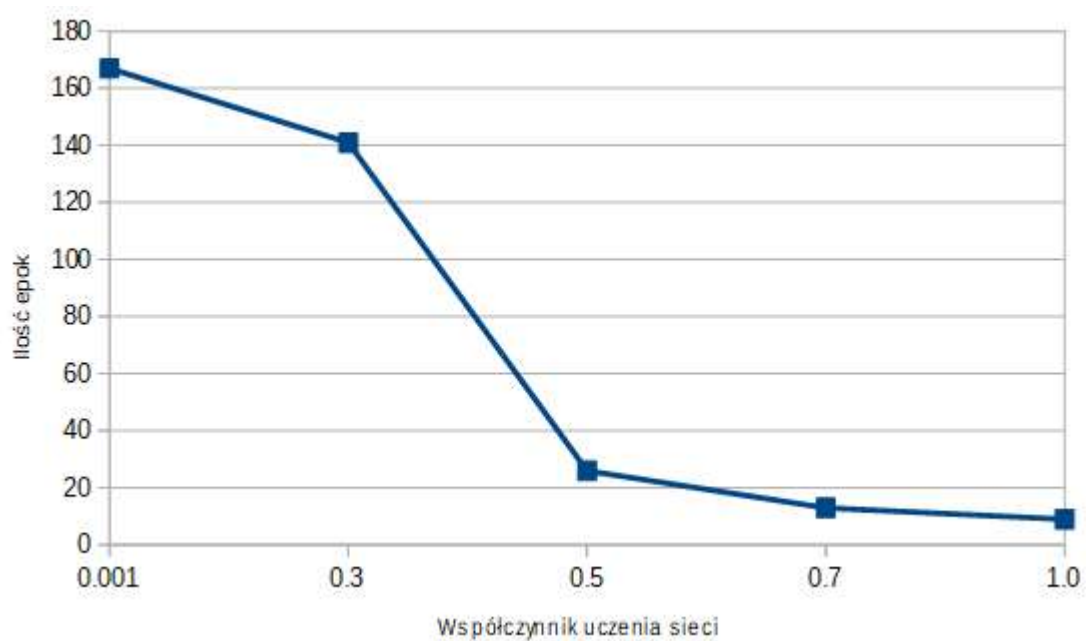
Sam dobór wag następuje na podstawie wzoru:

$$Q(w) = \frac{1}{2} \varepsilon^2 = \frac{1}{2} \left[d - \left(\sum_{i=0}^n w_i x_i \right) \right]^2$$

Otrzymywany błąd jest błędem średniokwadratowym.

Zestawienie otrzymanych wyników

Lp.	Współczynnik uczenia	Ilość epok	Poprawne rozpoznanie	Błędne rozpoznanie
1	0.001	167	14	6
2	0.3	141	11	9
3	0.5	26	10	10
4	0.7	13	10	10
5	1.0	9	7	13



\Wnioski

Czynnikiem mającym największy wpływ na proces uczenia się jednowarstwowej sieci neuronowej jest współczynnik uczenia. W tym ćwiczeniu dobierany jest on w sposób eksperymentalny.

Im większa wartość współczynnika uczenia się sieci, tym cały proces jest bardziej efektywny (mniejsza ilość epok potrzebnych do nauczenia sieci).

Warto zauważyć, że im mniejszy współczynnik uczenia się, tym więcej znaków jest rozpoznawanych poprawnie. Wynika z tego, że sama szybkość uczenia (ilość epok) nie jest wprost proporcjonalna do efektywności uczenia.

Listing kodu wraz z komentarzami

main.cpp

```
#include "Perceptron.h"
#include <iostream>

using namespace std;

int main()
{
    Perceptron perc;
    perc.ucze();
    int A[35] = { 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0,
1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1 };
    cout << "A = ";
    perc.sprawdzam(A);

    int a[35] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1,
1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1 };
    cout << "a = ";
    perc.sprawdzam(a);

    int B[35] = { 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 };
    cout << "B = ";
    perc.sprawdzam(B);

    int b[35] = { 0, 0, 0, 0, 0, 0, 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, 1, 1, 0, 0 };
    cout << "b = ";
    perc.sprawdzam(b);

    int C[35] = { 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0 }; cout << "C = "; perc.sprawdzam(C);
    int c[35] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0,
0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0 }; cout << "c = "; perc.sprawdzam(c);

    int D[35] = { 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 }; cout << "D = "; perc.sprawdzam(D);

    int d[35] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1,
1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1 }; cout << "d = "; perc.sprawdzam(d);

    int E[35] = { 1, 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 }; cout << "E = "; perc.sprawdzam(E);

    int e[35] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1,
0, 1, 0, 0, 0, 0, 1, 1, 1, 0 }; cout << "e = "; perc.sprawdzam(e);
    int F[35] = { 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0,
0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0 }; cout << "F = "; perc.sprawdzam(F);

    int f[35] = { 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0,
0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0 }; cout << "f = "; perc.sprawdzam(f);

    int G[35] = { 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1,
1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0 }; cout << "G = "; perc.sprawdzam(G);
    int g[35] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0,
1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0 }; cout << "g = "; perc.sprawdzam(g);

    int H[35] = { 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 }; cout << "H = "; perc.sprawdzam(H);

    int h[35] = { 0, 0, 0, 0, 0, 0, 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 }; cout << "h = "; perc.sprawdzam(h);

    int I[35] = { 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, 0, 0, 1, 0, 0 }; cout << "I = "; perc.sprawdzam(I);

    int i[35] = { 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, 0, 0, 1, 0, 0 }; cout << "i = "; perc.sprawdzam(i);

    int J[35] = { 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, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0 }; cout << "J = "; perc.sprawdzam(J);

    int j[35] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0 }; cout << "j = "; perc.sprawdzam(j);

    system("PAUSE");
}

```

Perceptron.h

```

#include <stdio>
#include <stdlib>
#include <cmath>
#include <ctime>
#include <iostream>
#include <fstream>

using namespace std;

class Perceptron {

    int **Litery;
    float *wagi;
    int wyniki[20];
    int ilosc_wejsc;
    bool wynik;
    float wsp_nauki;

public:

```

```

    Perceptron();
    ~Perceptron();
    void wczytywanie_danych_z_pliku();
    float f_sumowania(int i);
    void ucze();
    void sprawdzam(int tab[]);
};

```

Perceptron.cpp

```

#include "Perceptron.h"

using namespace std;

Perceptron::Perceptron()
{
    this->ilosc_wejsc = 20;
    this->wsp_nauki = 0.001;

    this->Litery = new int*[20];
    for (int i = 0; i < 20; i++)
    {
        Litery[i] = new int[35];
    }

    this->wagi = new float[35];
    for (int i = 0; i < 35; i++)
    {
        this->wagi[i] = (float)rand() / (float)RAND_MAX;
    }

    wczytywanie_danych_z_pliku();
}

void Perceptron::wczytywanie_danych_z_pliku()
{
    fstream plik;
    plik.open("litery.txt");
    if (!plik.good())
    {
        cout << "Blad otwarcia" << endl;    system("PAUSE");
    }
    while (!plik.eof())
    {
        for (int i = 0; i < ilosc_wejsc; i++)
        {
            for (int j = 0; j < 35; j++)
            {
                plik >> this->Litery[i][j];
            }
            plik >> this->wyniki[i];
        }
    }
    plik.close();
}

float Perceptron::f_sumowania(int i)

```

```

{
    float suma = 0;
    for (int j = 0; j < 35; j++)
    {
        suma += this->Litory[i][j] * this->wagi[j];
    }
    return suma;
}

void Perceptron::ucze()
{
    float local_err;
    float global_err = 0;
    int ID_litory;
    float nauczony = 1.00;
    int numer_epoki = 0;

    do {
        global_err = 0;
        for (ID_litory = 0; ID_litory < this->ilosc_wejsc; ID_litory++)
        {
            local_err = this->wyniki[ID_litory] - f_sumowania(ID_litory);
            for (int i = 0; i < 35; i++)
            {
                this->wagi[i] += this->wsp_nauki * local_err * this->Litory[ID_litory][i];
            }

            global_err += (local_err*local_err);
        }
        numer_epoki++;
    } while (global_err > nauczony);
    cout << "Ilosc epok - " << numer_epoki << endl;
}

void Perceptron::sprawdzam(int tab[])
{
    float suma = 0;
    float local_err = 0;
    float global_err = 0;
    float nauczony_aktywacji = 1.0;

    for (int i = 0; i < 35; i++)
    {
        suma += tab[i] * this->wagi[i];
    }

    local_err = 1 - suma;
    global_err = local_err*local_err;

    if (global_err < nauczony_aktywacji)
        this->wynik = true;
    else
        this->wynik = false;

    if (this->wynik == true)
        cout << "MALA" << endl << endl;
    else
        cout << "DUZA" << endl << endl;
}

Perceptron::~Perceptron()

```

{

}