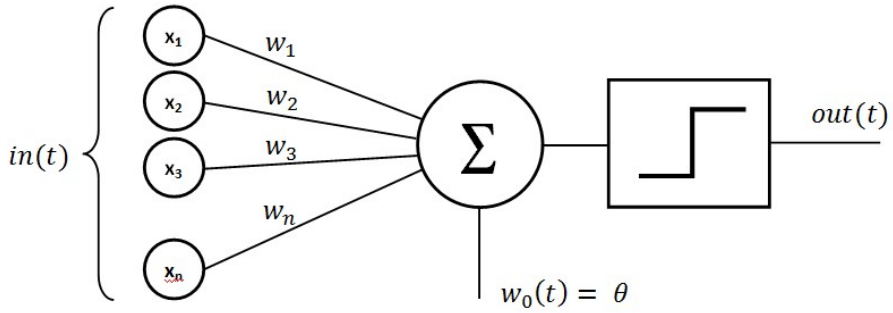
Magdalena Górka
Podstawy Sztucznej Inteligencji
Sprawozdanie nr 2

Cel ćwiczenia:

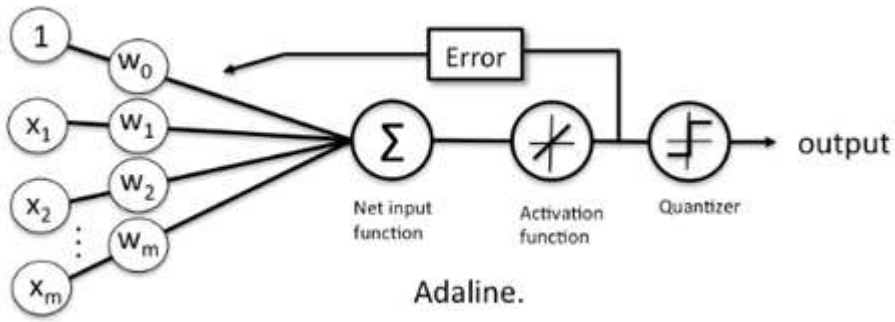
Celem ćwiczenia jest poznanie budowy i działania jednowarstwowych sieci neuronowych

oraz uczenie rozpoznawania wielkości liter.

Neuron McCullocha-Pittsa (ozn. MP). Neuron MP, będący pierwszym formalnie zdefiniowanym modelem neuronu, pomimo zastosowania bardzo dużych uproszczeń w stosunku do modelu neuronu biologicznego (a po części właśnie z tego powodu) znalazł zastosowanie w wielu modelach sieci neuronowych. Schemat neuronu MP przedstawiony jest na rysunku ponizej.



Model Adaline (ang. AdaptiveLinear Neuron) -Schemat neuronu Adaline przedstawiono na ponizszym rysunku. Budowa tego neuronu jest bardzo podobna do modelu perceptronu, a jedyna rożnica dotyczy algorytmu uczenia. Sygnał wyznacza się w ten sam sposób, co w przypadku uczenia perceptronu. Jednak w przypadku neuronu typu Adaline porównuje się sygnał wzorcowy d z sygnałem s, na wyjściu części liniowej neuronu (nie uwzględnia się funkcji aktywacji).



Adaline.

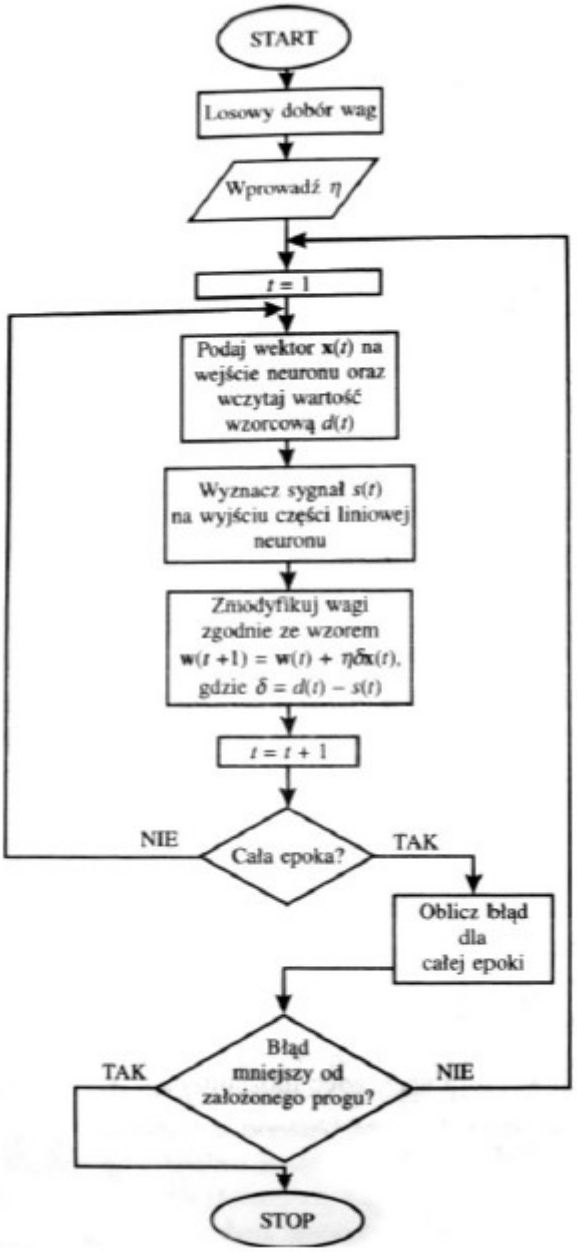
Magdalena Górka
Podstawy Sztucznej Inteligencji
Sprawozdanie nr 2

Modele te różnią się algorytmem uczenia. Różnicę między sygnałem wzorcowym, a sygnałem s nazywamy błędem ε = d - s. Uczenie neuronu, czyli dobór wag, sprowadza się do minimalizacji funkcji określonej w sposób następujący:

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

Miarę błędu określa się mianem błędu średniego kwadratowego.

Algorytm działania modelu Adaline:



Oznaczenia:

- i-numer wagi neuronu,
- t-numer iteracji w epoce,

Magdalena Górka
Podstawy Sztucznej Inteligencji
Sprawozdanie nr 2
• d-sygnał wzorcowy,

• y-sygnał wyjściowy neuronu,

• s-sygnał wyjściowy sumatora neuronu,

• x-wartość wejściowa neuronu,

• η - współczynnik uczenia (0,1).

Przeprowadzenie ćwiczenia:

Litery zostały wprowadzone z pliku. Tworzył y je 0 oraz 1 ułożone w tablicy o wymiarach 8x9 w taki sposób aby '1' tworzyła dana literę pisaną dużą jak i mała.

Przyładowy wygląd jednej litery :

000010000

000101000

001000100

010000010

011111110

010000010

010000010

000000000 -> Duża litera A

000000000

000110000

001001000

010000100

011111100

010000100

010000100

000000000-> Mała litera a

Magdalena Górka
Podstawy Sztucznej Inteligencji
Sprawozdanie nr 2
Otrzymane Wyniki:

Współczynik Nauczania	**Liczba Epok**	**Liczba Błędnych**	**Liczba Poprawnych**
0.001	**325**	**5**	**15**
0.01	**36**	**2**	**18**
0.1	**14**	**5**	**15**
0.2	**26**	**10**	**10**
0.5	**6**	**10**	**10**

4. Wnioski:

Patrząc na powyższe wyniki stwierdzić, że algorytm przy współczynniku uczenia 0.001 dalej w miarę zadowalające wyniki lecz jeżeli spojrzymy na wyniki tego samego algorytmu przy współczynniku uczenia równym 0.01 widać iż nie tylko jego precyzja była lepsza ale szybkość została bardzo zwiększona. Warto zauważyć również iż algorytm przy współczynniku uczenia się równym 0.1 dał podobny wyniki co przy współczynniku 0.001 lecz jego szybkość było jeszcze większa niż przypadku algorytmu o współczynniku 0.01. Można więc stwierdzić że szybkość nauki a przy tym szybkość algortymu nie idzie w parze z „efektywna nauka”.

Magdalena Górka
Podstawy Sztucznej Inteligencji
Sprawozdanie nr 2
Kod Programu:

LAB2_Perceptron.h

```
#include <cstdio>
#include <cstdlib>
#include <cmath>
#include <ctime>
#include <iostream>
#include <fstream>
#include <iomanip>



using namespace std;

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

public:
	Perceptron();

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



	~Perceptron();
};
```

LAB2_Perceptron.cpp

```
#include "LAB2_Perceptron.h"
#include <cstdlib>
#include <iostream>
	using namespace std;


Perceptron::Perceptron()
{
	this->ilosc_wejsc = 15;
	this->wsp_nauki = 0.5;

	this->Litery = new int*[20];
	for (int i = 0; i < 20; i++)
		Litery[i] = new int[72];

	this->wagi = new float[72];
	for (int i = 0; i < 72; 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 < 72; 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 < 72; j++)
	{
		suma += this->Litery[i][j] * this->wagi[j];
	}
	return suma;
}

void Perceptron::ucze()
{
	float local_err;
	float global_err = 0;
	int ID_litery;
	float nauczony = 1.55;
	int numer_epoki = 0;
	do
	{
		global_err = 0;
		for (ID_litery = 0; ID_litery < this->ilosc_wejsc; ID_litery++)
		{
			local_err = this->wyniki[ID_litery] - f_sumowania(ID_litery);
			for (int i = 0; i < 72; i++)
			{
				this->wagi[i] += this->wsp_nauki * local_err * this->Litery[ID_litery][i];
			}

			global_err += (local_err*local_err);
		}
```

```
			numer_epoki++;
		} while (global_err > nauczony);
		cout << "Ilosc sesji nauczania = " << numer_epoki << endl;
}


void Perceptron::sprawdzam(int tab[])
{

		float suma = 0;
		float local_err = 0;
		float global_err = 0;
		float nauczony_aktywacji = 1.5;
		for (int i = 0; i < 72; 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 litera!" << endl << endl;
		else
			cout << "duza litera!" << endl << endl;
}


Perceptron::~Perceptron()
{
}
```

LAB2_Source.cpp

```
#include "LAB2_Perceptron.h"
#include <iostream>

using namespace std;

void main()
{

		Perceptron perc;
		perc.ucze();
		int A[72] = { 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1,
1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
		cout << "A = ";
		perc.sprawdzam(A);

		int a[72] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1,
1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
		cout << "a = ";
		perc.sprawdzam(a);
```

Magdalena Górka
Podstawy Sztucznej Inteligencji
Sprawozdanie nr 2

```
        int B[72] = { 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0,
0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
        cout << "B = ";
        perc.sprawdzam(B);

        int b[72] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0,
0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
        cout << "b = ";
        perc.sprawdzam(b);

        int C[72] = { 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
        cout << "C = ";
        perc.sprawdzam(C);

        int c[72] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
        cout << "c = ";
        perc.sprawdzam(c);

        int D[72] = { 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
        cout << "D = ";
        perc.sprawdzam(D);


        int d[72] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
        cout << "d = ";
        perc.sprawdzam(d);

        int E[72] = { 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
        cout << "E = ";
        perc.sprawdzam(E);

        int e[72] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
        cout << "e = ";
        perc.sprawdzam(e);

        int F[72] = { 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
        cout << "F = ";
        perc.sprawdzam(F);

        int f[72] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
        cout << "f = ";
        perc.sprawdzam(f);

        int G[72] = { 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0,
0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
        cout << "G = ";
        perc.sprawdzam(G);

        int g[72] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
        cout << "g = ";
```

Magdalena Górka
Podstawy Sztucznej Inteligencji
Sprawozdanie nr 2

```
        perc.sprawdzam(g);

        int H[72] = { 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0,
0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
        cout << "H = ";
        perc.sprawdzam(H);

        int h[72] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0,
0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
        cout << "h = ";
        perc.sprawdzam(h);

        int K[72] = { 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0,
0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
        cout << "K = ";
        perc.sprawdzam(K);

        int k[72] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0,
1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
        cout << "k = ";
        perc.sprawdzam(k);

        int L[72] = { 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
        cout << "L = ";
        perc.sprawdzam(L);

        int l[72] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
        cout << "l = ";
        perc.sprawdzam(l);

        system("PAUSE");
}
```