**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

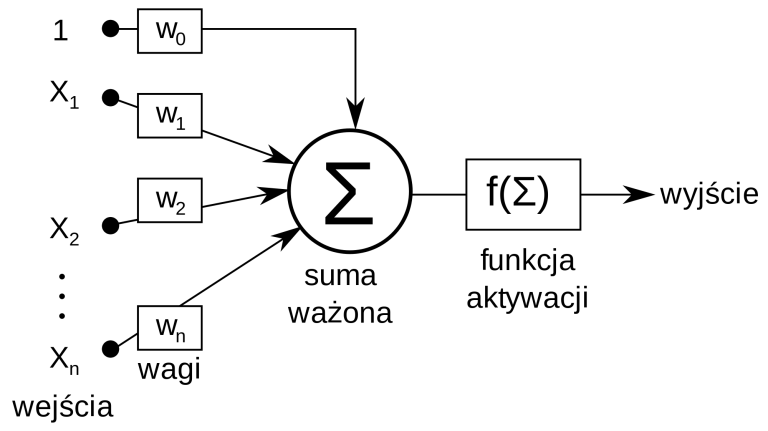
Synapsy – wektor wag

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*.



W zadaniu rozpatruje się neuron o dwóch wejściach.

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 wyświetlenie komunikatu oraz 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)\*x2

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.

**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:

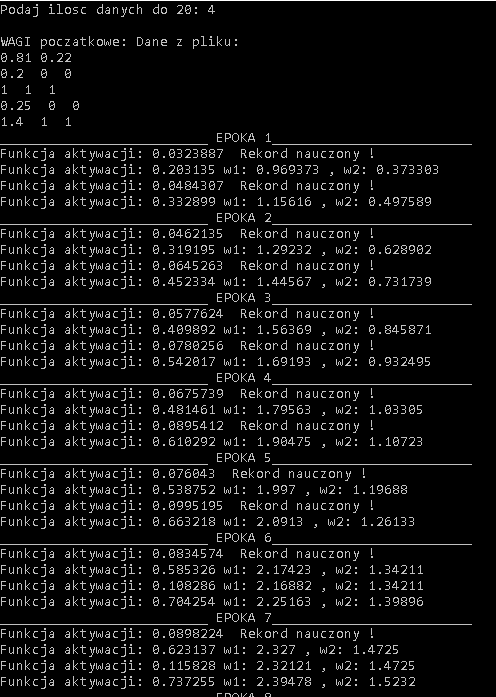
Wersja wyjściowa, do której będą porównywane pozostałe wersje.

Stała uczenia = 0.2

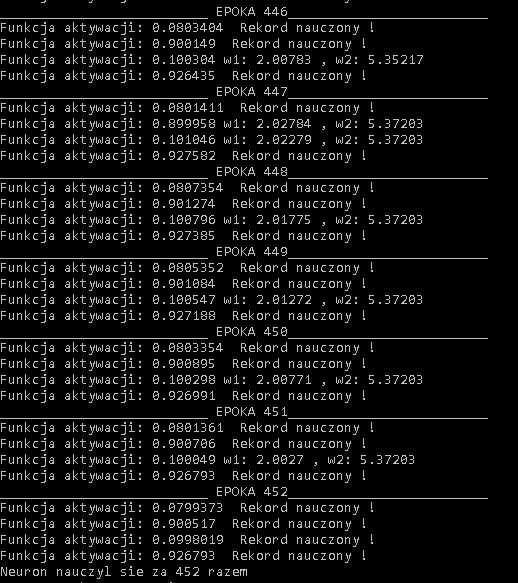
Liczba rekordów uczących z pliku = 4

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:



…………….



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.

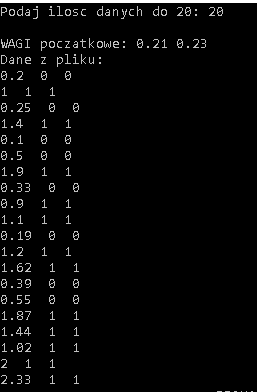
Wersja II:

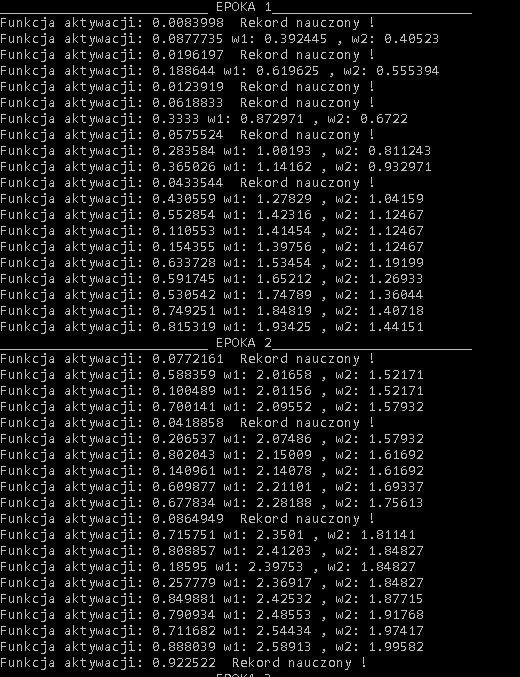
Stała uczenia = 0.2

Liczba rekordów uczących z pliku= 20 – zwiekszono liczbe

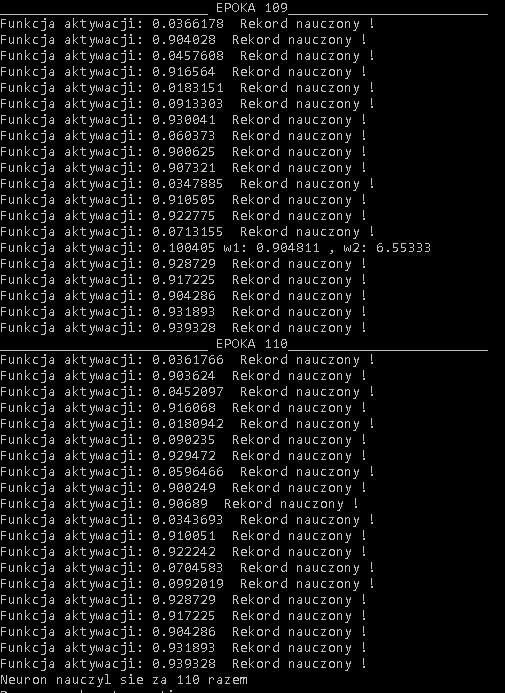
Ilość epok, po których widoczny jest postęp nauczania (przybliżając wartości do 0.1 dla 0 oraz 0.99 dla 1), dla których możemy stwierdzić, że neuron się nauczył zależy od wygenerowanych wag.

Przykładowy program:





……….



Obserwacje:

Ilość iteracji utrzymywała się w zakresie 100-120.

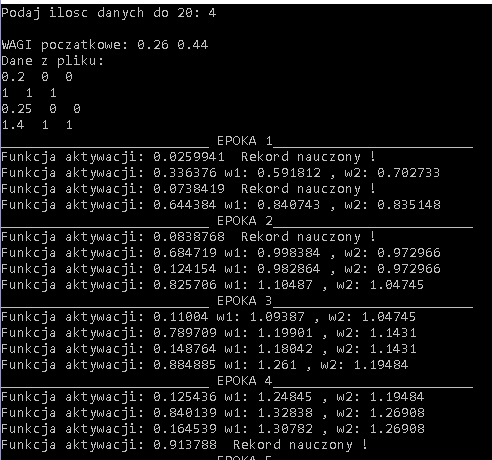
W przeprowadzonym teście nie napotkano żadnych błędów.

Wersja III:

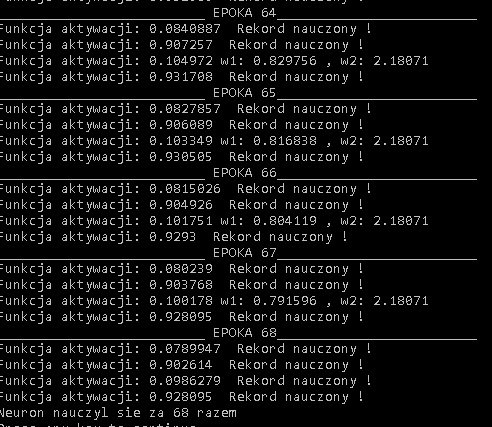
Stała uczenia = 0.5 – zwiększono stałą

Liczba par danych uczących = 4

Przykładowy program :



……



Obserwacje:

Ilość iteracji utrzymywała się w zakresie 50-80.

W przeprowadzonym teście nie napotkano żadnych błędów.

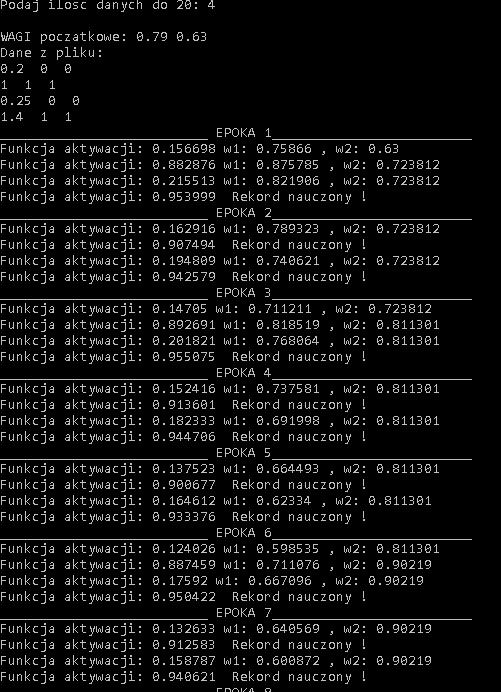
Wersja IV:

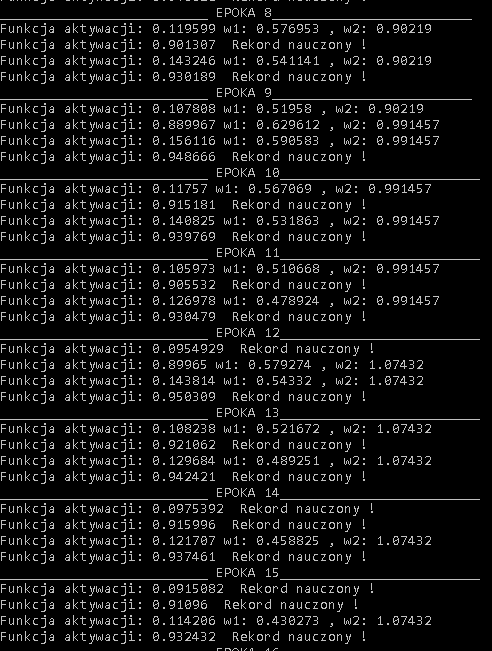
Stała uczenia = 1 – zwiększono stałą

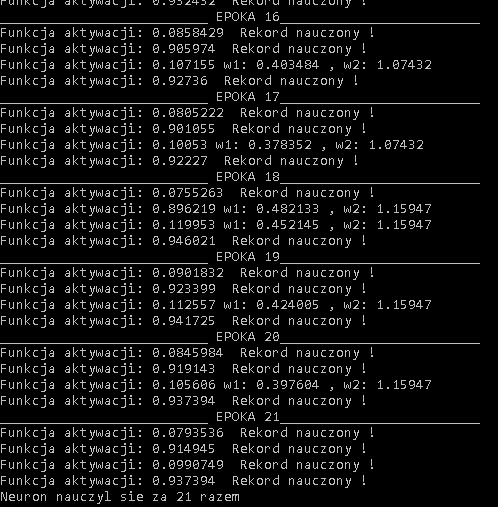
Liczba rekordów uczących z pliku= 4

stwierdzić, że neuron się nauczył zależy od wygenerowanych wag.

Przykładowy program:







Obserwacje:

Ilość iteracji utrzymywała się w zakresie 15-30.

W przeprowadzonym teście nie napotkano żadnych błędów.

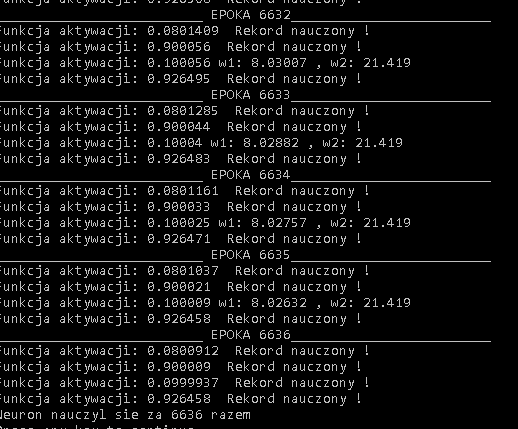
Wersja V:

Stała uczenia = 0.05 – zmniejszono stałą

Liczba rekordów uczących z pliku= 4

stwierdzić, że neuron się nauczył zależy od wygenerowanych wag.

Przykładowy program:



Obserwacje:

Program wykonuje się bardzo dużą ilość razy. Ilość iteracji utrzymywała się w zakresie 6500 w górę.

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;

}