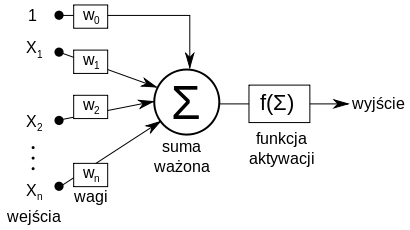
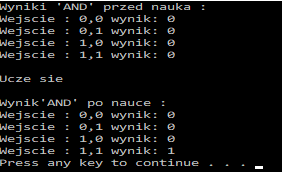
**Mateusz Bożek  
  
Scenariusz 1**  
  
**Temat :** Budowa i działanie perceptronu  
  
**Cel ćwiczenia :** Poznanie budowy i działania perceptronu poprzez implementację oraz uczenie perceptronu realizującego wybraną funkcję logiczną dwóch zmiennych.  
  
**1.** Do zrealizowania zadania wykorzystano perceptron zgodny z modelem McCulocha-Pittsa.W sumatorzeodbywa się sumowanie sygnałów wraz odpowiednimi wagami. Następnie dodawany jest sygnał polaryzacji xo = 1 dla którego waga wynosi wio. By wyliczyć sygnał wyjścia z neuronu należy skorzystać z zależności :  
  
 **2.** Perceptron użyty w podanym zadaniu ma za zadanie nauczyć się funkcji logicznej AND



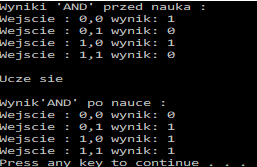
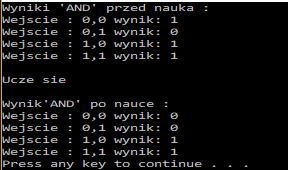
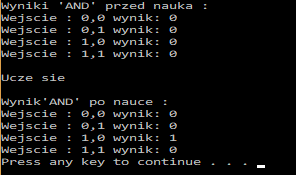
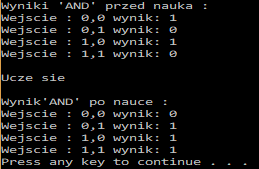
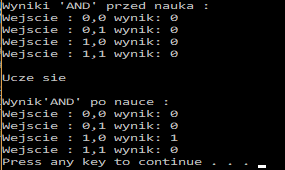
|  |  |  |
| --- | --- | --- |
| **X1** | **X2** | **y** |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 0 | 0 | 1 |

Na początku programu wagi zostały ustawione w wyniku użycia funkcji losującej, natomiast podczas procesu nauki przez program, wagi były odpowiednio modyfikowane. Nauczanie. Podczas testowania nie nastąpiły żadne zmiany. Po przeprowadzeniu 16 epok, program nauczył się prawidłowo. Przedstawia to poniższy zrzut ekranu :

**3.**Przy zmianie wartości epok, program uczy się nieprawidłowo. Dla 8 epok, program uczy się złych wartości co przedstawia niżej zamieszczony obraz



Podobnie jest w przypadku gdy liczba epok wynosi 5.Przy zmianie liczby epok oraz przy zmianie początkowego współczynnika nauki program również nie przyswaja wiedzy prawidłowo. Wyniki nauczania są nieprawidłowe.  
  
  
  
Liczba epok : 6  
współczynnik nauki : 0,5  
  
  
  
Liczba epok : 4  
współczynnik nauki : 0,7  
  
  
  
Liczba epok : 6  
Współczynnik nauki : 0,9  
  
  
  
 **4.**Wnioski:  
  
- Duży wpływ na proces nauki programu ma liczba epok. Oczywiste jest, że wraz z mniejszą ilością epok program nie ma czasu na prawidłowe wyciągnięcie wniosków, czego konsekwencją są złe wyniki.  
  
- Na skuteczność nauki wpływa współczynniki uczenia się. Wraz z podawaniem mniejszej jej wartości program wyciągał złe wnioski, czego konsekwencją są złe wyniki finalne programu. Natomiast gdy jest on wyższy program proces uczy sie lepiej.  
  
- Znaczenie dla prawidłowego uczenia się programu ma metoda uczenia się z nauczycielem. Wynik przy tej metodzie jest poprawny, natomiast w przypadku nauczania bez nauczyciela, program podaje nieprawidłowe wartości.  
  
  
  
**5.Kod programu :**  
  
source.c



#include <stdio.h>

#include "perceptron.h"

const int iteration = 16, numberOfImputs = 2;

const double learnRate = 0.2;

int main() {

const double ZERO\_ZERO[] = {0, 0};

const double ZERO\_ONE[] = {0, 1};

const double ONE\_ZERO[] = {1, 0};

const double ONE\_ONE[] = {1, 1};

int i;

Perceptron \*pOR = Perceptron\_new(numberOfImputs, learnRate);

printf("Wyniki 'AND' przed nauka :\n");

printf("Wejscie : 0,0 wynik: %d\n", uzyskajWynik(pOR, ZERO\_ZERO));

printf("Wejscie : 0,1 wynik: %d\n", uzyskajWynik(pOR, ZERO\_ONE));

printf("Wejscie : 1,0 wynik: %d\n", uzyskajWynik(pOR, ONE\_ZERO));

printf("Wejscie : 1,1 wynik: %d\n", uzyskajWynik(pOR, ONE\_ONE));

printf("\nUcze sie \n\n");

for (i = 0 ; i < iteration; i++) {

ucz(pOR, ZERO\_ZERO, 0);

ucz(pOR, ZERO\_ONE, 0);

ucz(pOR, ONE\_ZERO, 0);

ucz(pOR, ONE\_ONE, 1);

}

printf("Wynik 'AND' po nauce :\n");

printf("Wejscie : 0,0 wynik: %d\n", uzyskajWynik(pOR, ZERO\_ZERO));

printf("Wejscie : 0,1 wynik: %d\n", uzyskajWynik(pOR, ZERO\_ONE));

printf("Wejscie : 1,0 wynik: %d\n", uzyskajWynik(pOR, ONE\_ZERO));

printf("Wejscie : 1,1 wynik: %d\n", uzyskajWynik(pOR, ONE\_ONE));

system("pause");

return 0;

}  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
perceptron.h   
  
#ifndef PERCEPTRON\_H

#define PERCEPTRON\_H

#ifdef \_\_cplusplus

extern "C" {

#endif

typedef struct Perceptron {

unsigned numInputs\_;

double \*weights\_;

double threshold\_, trainingRate\_;

} Perceptron;

Perceptron \*Perceptron\_new(unsigned numOfInputs, double trainingRate);

void ucz(Perceptron \*perceptron, const double inputs[], int expectedResult);

int uzyskajWynik(const Perceptron \*perceptron, const double inputs[])

double uzyskajWartosc(const Perceptron \*perceptron, const double inputs[]);

void ustawWage(Perceptron \*perceptron, const double \*weights);

const double \*uzyskajWage(const Perceptron \*perceptron);

double losowaWartosc();

void zmianaWagi(Perceptron \*perceptron, int actualResult, int desiredResult, const double inputs[]);

#ifdef \_\_cplusplus

}

#endif

#endif  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
perceptron.c  
  
#include <stdlib.h>

#include <time.h>

#include "perceptron.h"

#ifdef \_\_cplusplus

extern "C" {

#endif

Perceptron \*Perceptron\_new(unsigned numOfInputs, double trainingRate) {

unsigned i;

static int initializedRandomization\_ = 0;

Perceptron \*perc = (Perceptron\*)malloc(sizeof(Perceptron));

perc->numInputs\_ = numOfInputs;

if (perc->numInputs\_ <= 0) {

perc->numInputs\_ = 1;

}

perc->trainingRate\_ = trainingRate;

perc->weights\_ = (double\*)malloc(perc->numInputs\_ \* sizeof(double));

if (! initializedRandomization\_) {

srand(time(NULL));

initializedRandomization\_ = 1;//

}

for (i = 0 ; i < perc->numInputs\_ ; i++) {

perc->weights\_[i] = losowaWartosc();

}

perc->threshold\_ = losowaWartosc();

return perc;

}

void ucz(Perceptron \*perceptron, const double inputs[], int expectedResult) {

int result = uzyskajWynik(perceptron, inputs);

if (result == expectedResult) {

return;

}

zmianaWagi(perceptron, result, expectedResult, inputs);

}

void zmianaWagi(Perceptron \*perceptron, int actualResult, int desiredResult, const double inputs[]) {

unsigned i;

for (i = 0; i < perceptron->numInputs\_; i++) {

perceptron->weights\_[i] += perceptron->trainingRate\_ \* (desiredResult - actualResult) \* inputs[i];

}

perceptron->threshold\_ -= perceptron->trainingRate\_ \* (desiredResult - actualResult);

}

double uzyskajWartosc(const Perceptron \*perceptron, const double inputs[]) {

unsigned i;

double ans = 0;

for (i = 0 ; i < perceptron->numInputs\_ ; i++) {

ans += perceptron->weights\_[i] \* inputs[i];

}

return ans;

}

void ustawWage(Perceptron \*perceptron, const double \*weights) {

unsigned i;

for (i = 0; i < perceptron->numInputs\_; i++) {

perceptron->weights\_[i] = weights[i];

}

}

int uzyskajWynik(const Perceptron \*perceptron, const double inputs[]) {

return (uzyskajWartosc(perceptron, inputs) >= perceptron->threshold\_);

}

double losowaWartosc() {

double randValue = ((double)rand() / (double)RAND\_MAX);

double negativeRand = ((double)rand() / (double)RAND\_MAX);

if (negativeRand < 0.5) {

randValue \*= -1.0;

}

return randValue;

}

const double \*uzyskajWage(const Perceptron \*perceptron) {

return perceptron->weights\_;

}

#ifdef \_\_cplusplus

}

#endif