Podstaw sztucznej inteligencji-projekt 1.

Program został wykonany w języku C++. Jego celem jest uczenie, a następnie próba odgadnięcia wartości funkcji Boolowskiej AND składającej się z dwóch zmiennych.

Próg aktywacji ustawiony jest na wartość 1. Jeśli wynik sumowania iloczynów sygnałów wejściowych i wag przekroczy wartość 1, wtedy następuje aktywacja perceptronu.

Regulacji ulegają: współczynnik uczenia, ilość danych uczących, a co za tym idzie-ilość danych testujących oraz ilość cykli uczenia.

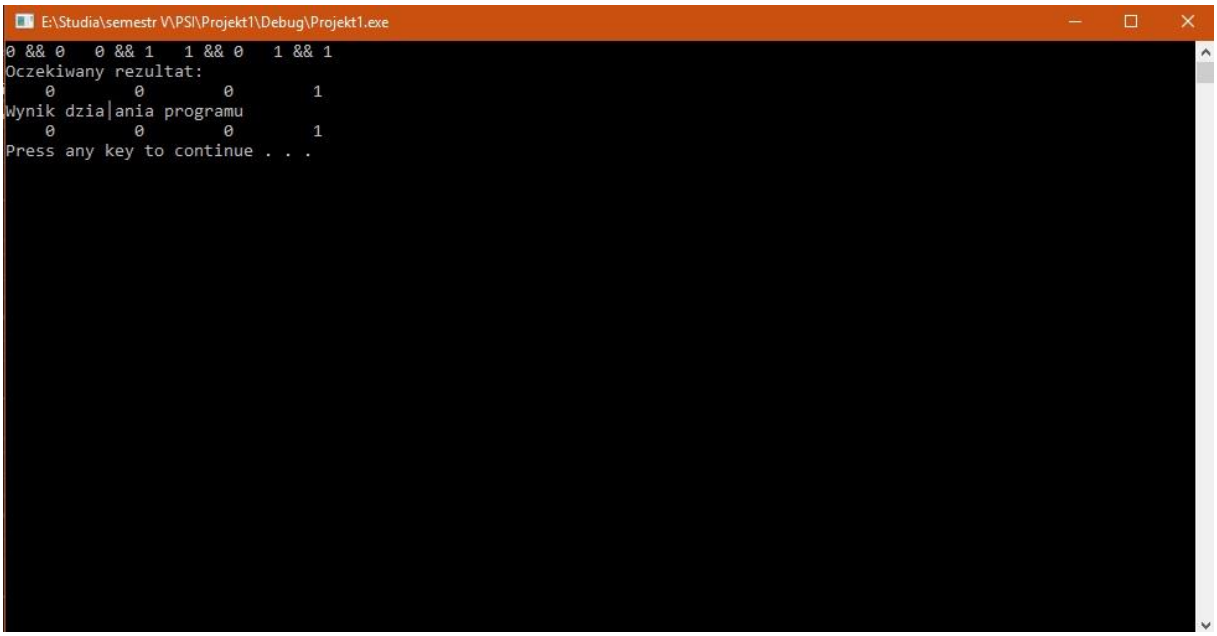
Dla funkcji logicznej dwóch zmiennych możliwe są cztery scenariusze. Dla iloczynu logicznego przedstawia się to następująco:

Wartość 1	Wartość 2	Wynik
1	1	1
1	0	0
0	1	0
0	0	0

Początkową wagę ustaliłem losowo na wartość 0.33. W dalszej części programu zmienia się ona zgodnie z zasadą Widrow-Hoffa.

Najbliższym ideału współczynnikiem uczenia było 0.1. Zmiejszenie, lub zwiększenie go skutkowały gorszymi wynikami, dość proporcjonalnie do różnicy tej wartości. Przy ustawieniu 2 danych jako uczące oraz 3 sesjach nauki trafność programu zawsze wynosiła 100%. Przy ustawieniu 2 danych jako uczące, program miał 50% skuteczności, przy 1 i 4-25%.

Gdy współczynnik uczenia wynosił 1 wyniki pogorszyły się. Program był w stanie podać prawidłową konfigurację tylko, gdy daliśmy mu 3 wiersze do nauki. Przy współczynniku 0.01 i 3 wierszach uczących program nadal potrafił nauczyć się funkcji.



Screen z działania programu.

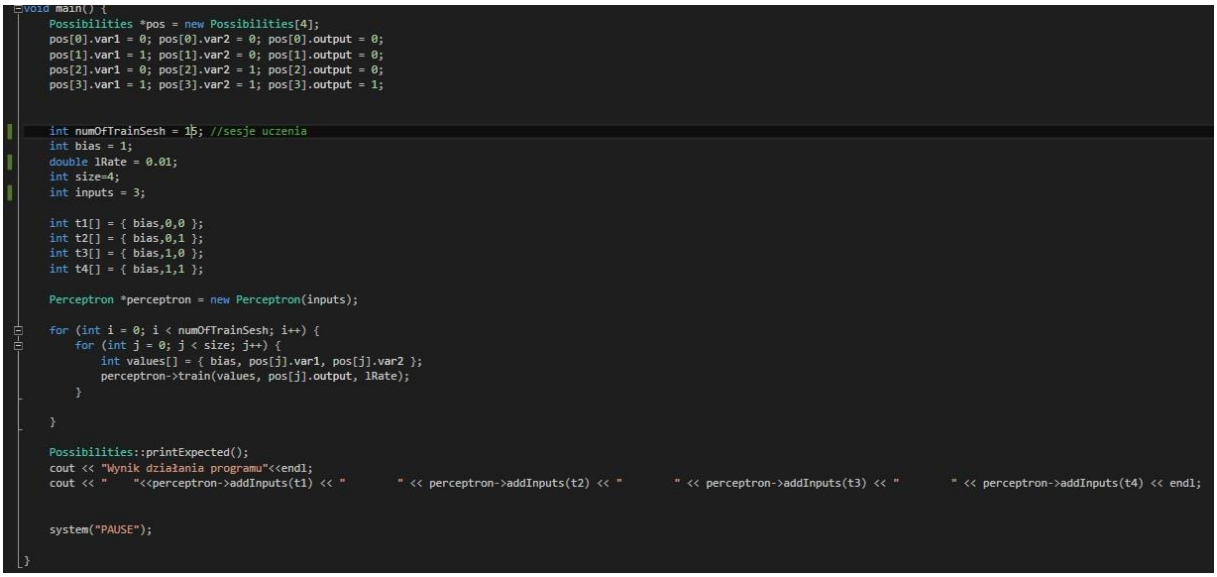
```
struct Perceptron {
    int inputs;
    int bias;
    double *weights;
    bool active(double sum) {
        if (sum > 0)
            return 1;
        else
            return 0;

    };
    double addInputs(int tab[]) {
        double sum = 0;
        for (int i = 0; i < this->inputs; i++) {
            sum += tab[i] * this->weights[i];
        }
        return active(sum);
    };

    void train(int tab[], double y, double lRate) {
        double ys = this->addInputs(tab);

        for (int i = 0; i < this->inputs; i++) {
            this->weights[i] += (y - ys) * lRate * tab[i];
        }
    }
    Perceptron(int _inputs) {
        this->inputs = _inputs;
        this->weights = new double[inputs];

        for (int i = 0; i < inputs; i++) {
            this->weights[i] = 0.33;
        }
    };
};
```



```
void main() {
    Possibilities *pos = new Possibilities[4];
    pos[0].var1 = 0; pos[0].var2 = 0; pos[0].output = 0;
    pos[1].var1 = 1; pos[1].var2 = 0; pos[1].output = 0;
    pos[2].var1 = 0; pos[2].var2 = 1; pos[2].output = 0;
    pos[3].var1 = 1; pos[3].var2 = 1; pos[3].output = 1;


    int numOfTrainSesh = 15; //sesje uczenia
    int bias = 1;
    double lRate = 0.01;
    int size=4;
    int inputs = 3;

    int t1[] = { bias,0,0 };
    int t2[] = { bias,0,1 };
    int t3[] = { bias,1,0 };
    int t4[] = { bias,1,1 };

    Perceptron *perceptron = new Perceptron(inputs);

    for (int i = 0; i < numOfTrainSesh; i++) {
        for (int j = 0; j < size; j++) {
            int values[] = { bias, pos[j].var1, pos[j].var2 };
            perceptron->train(values, pos[j].output, lRate);
        }

    }

    Possibilities::printExpected();
    cout << "Wynik działania programu"<<endl;
    cout << "    "<<perceptron->addInputs(t1) << "        " << perceptron->addInputs(t2) << "        " << perceptron->addInputs(t3) << "        " << perceptron->addInputs(t4) << endl;


    system("PAUSE");

}
```

Wnioski:

Wbrew możliwemu początkowemu przykonaniu, wyższy współczynnik uczenia nie jest proporcjonalny do ilości prawidłowych odpowiedzi. Jednoznacznie zaprzecza temu test, który wykonałem dla wsp 0.1, kiedy to procent prawidłowych odpowiedzi wynosił 100%, oraz dla wsp 1, kiedy procent prawidłowych odpowiedzi wynosił 75%.