Kaja Kwiatkowska
Inżynieria Obliczeniowa
286114

Podstawy sztucznej inteligencji
Sprawozdanie ćwiczenie 1 - Budowa i działanie perceptronu

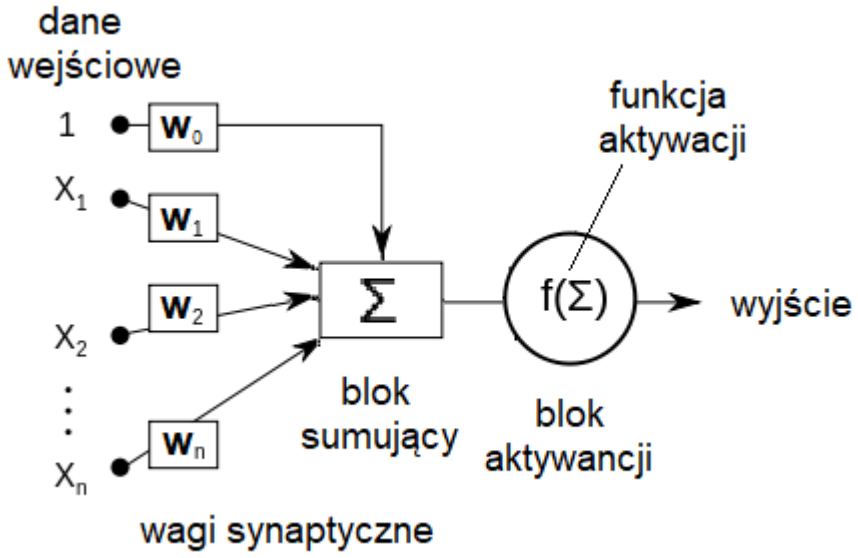
Cel:

Celem ćwiczenia było poznanie budowy i działania perceptronu poprzez implementację oraz uczenie perceptronu realizującego wybraną funkcję logiczną dwóch zmiennych.

Wiedza:

Neuron jest prostym systemem przetwarzającym wartości sygnałów wejściowych na pojedynczą wartość wyjściową. Jest to podstawowy element sieci neuronowej.

Neuron McCullocha – Pittsa jest jednym z matematycznych modeli nauronu. Na wejściu ma wiele sygnałów, a na wyjściu tylko jeden. Każde z wejść ma przyporządkowaną wagę wejścia - liczbę rzeczywistą. Wartość na wyjściu jest wartością funkcji aktywacji z obliczonej sumy iloczynów wartości wejściowej i jej wagi.



Budowa pojedynczego neuronu McCullocha - Pittsa

Perceptron jest najprostszą siecią neuronową składającą się z jednego lub wielu niezależnych neuronów McCullocha – Pittsa, która implementuje algorytm uczenia nadzorowanego (uczenie z nauczycielem) klasyfikatorów binarnych, potrafi określić przynależność parametrów wejściowych do jednej z dwóch klas.
Zasada działania perceptronu polega na klasyfikowaniu danych wejściowych i ustawianiu adekwatnej wartości wyjścia po uprzednim uczeniu na przykładowych danych, modyfikując w odpowiedni sposób wagi dla wejść i połączeń między warstwami neuronów, aby otrzymać pożądane wartości na wyjściu.

Zadania, które wykonałam w ramach ćwiczenia:

- zaimplementowałam sztuczny neuron
- wprowadziłam dane uczące i wygenerowałam losowo dane testujące dla dwóch zmiennych
- przeprowadziłam proces uczenia perceptronu dla różnej liczby danych uczących oraz różnych współczynników uczenia
- przetestowałam perceptron.

Do wykonania ćwiczenia, jako funkcję logiczną wybrałam **koniunkcję** (**bramkę logiczną AND**).

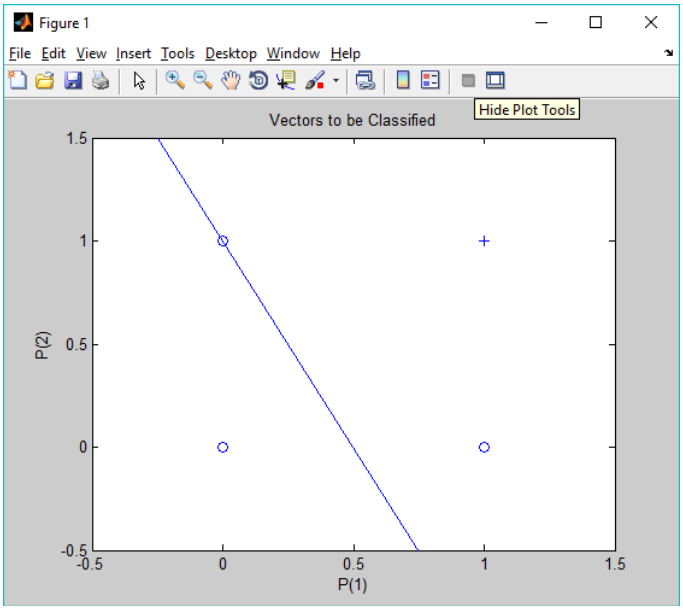
p	q	$p \wedge q$
1	1	1
1	0	0
0	1	0
0	0	0

Poprawne wyniki koniunkcji

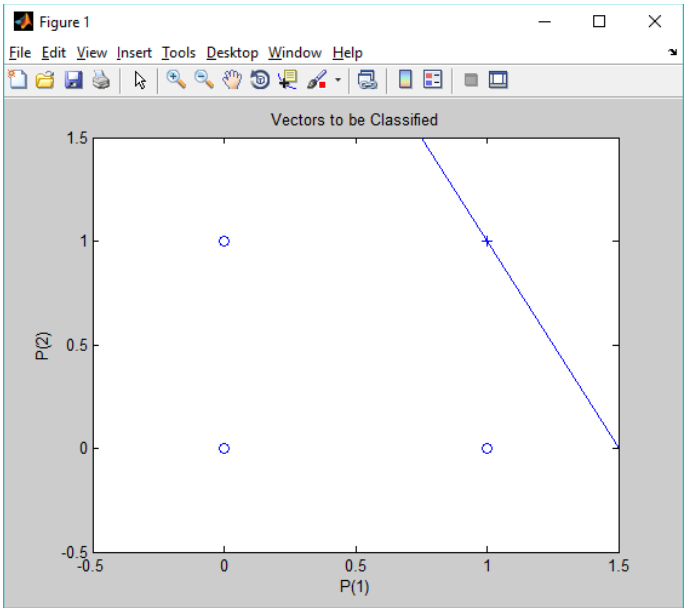
Jako funkcji uczącej użyłam funkcji **train()**, która wyróżnia się szybkością działania oraz zmianą wagi synaptycznej przy każdej iteracji.

W celu osiągnięcia prawidłowych wyników dobrałam parametr treningu - maksymalną ilością epok. Przy 5 epokach i więcej perceptron podawał poprawne wyniki.

Liczba epok w treningu	2	4	5	6	8	10	12
1 test	test = 0 0 0 1 1 0 1 0 efekt = **1 0 1 1**	test = 0 0 0 0 0 1 1 1 efekt = 0 **1 1 1**	test = 0 1 0 0 0 1 efekt = 0 0 0	test = 0 0 0 0 1 1 efekt = 0 0 0	test = 0 0 1 1 1 0 1 0 efekt = 0 0 1 0	test = 0 0 1 1 1 1 efekt = 0 0 1	test = 1 1 0 0 efekt = 0 0
2 test	test = 0 1 0 1 1 1 efekt = **1 1 1**	test = 1 1 1 1 efekt = 1 1	test = 1 0 0 1 0 1 efekt = 1 0 0	test = 1 0 0 0 efekt = 0 0	test = 0 0 1 0 0 0 efekt = 0 0 0	test = 0 0 0 1 0 1 1 1 efekt = 0 0 0 1	test = 0 1 1 1 0 0 efekt = 0 0 0



Wykres z wynikami działania perceptronu dla bramki logicznej AND dla 2 epok - NIEPOPRAWNE

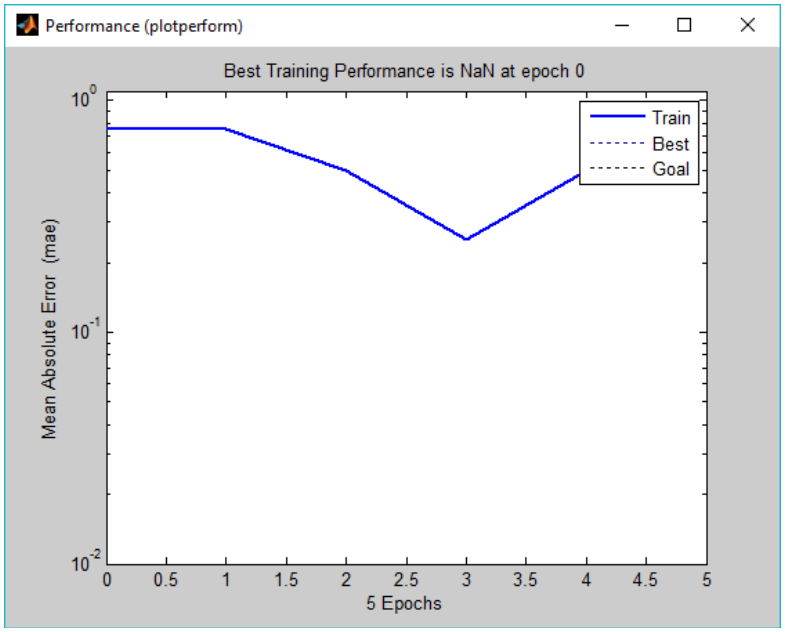


Wykres z wynikami działania perceptronu dla bramki logicznej AND dla 5 epok - POPRAWNE

Wnioski:

Powyższe wykresy potwierdzają poprawność wyników dla 5 epok. Pokazuje, to, że poprawność zależy od liczby epok w treningu (miejsce na wykresie oznaczone „+" – funkcja logiczna przyjmuje wartość 1, dla „-" – funkcja logiczna przyjmuje wartość 0).

Dwa pozostałe parametry uczące również dobrałam na zasadzie testów i sprawdzania poprawności wyników. Stąd wartość błędu średniokwadratowego ustawiłam na 0.001, natomiast parametr osiągnięcia celu na 0.01.



Wykres wydajności uczenia sieci

Wnioski:

Na podstawie powyższego wykresu widać, że dla pierwszej epoki sieć ma największą wydajność, gdzie później następuje spadek wydajności i osiągnięcie najniższej wartości dla 3 epok. Świadczyć to może o przeuczeniu się sieci, co wpływa na błędy w wynikach.

Domyślne wartości wag w funkcji train() działają najbardziej optymalnie, pokazują najlepsze wyniki podczas uczenia. Prawdopodobnie z powodu ich niskich wartości.
Do poprawnego działania perceptron potrzebuje pełnej tablicy prawdy, inaczej trening nie daje oczekiwanych efektów. Jest to spowodowane tym, że budowa sieci neuronowej jest zbyt prosta i nie potrafi samodzielnie przewidzieć poprawnych wyników bez kompletnych danych uczących.

Listing kodu programu:

```
close all; clear all; clc;

net = newp([0 1; 0 1], 1,'hardlim');          %tworzenie sieci o jednym neuronie
dane_wejsciowe = [0 0 1 1;                    %wartości wejściowe pierwszej zmiennej
                  0 1 0 1];                   %wartości wejściowe drugiej zmiennej
wyjscie = [0 0 0 1];                          %wartości wyjściowe bramki logicznej
plotpv(dane_wejsciowe, wyjscie);              %wykres z wynikami

%parametry uczenia - współczynniki uczenia
net.trainParam.epochs = 6;                    %maksymalna ilość epok
net.trainParam.goal = 0.01;                   %osiągnięcie celu
net.trainParam.mu = 0.001;                    %błąd średniokwadratowy

net = train(net, dane_wejsciowe, wyjscie);    %uczenie
plotpc(net.iw{1, 1}, net.b{1})

Y = sim(net, dane_wejsciowe);                 %symulacja danych
test = randi([0 1], 2, randi(5));             %test na wylosowanych danych
efekt = sim(net, test);                       %efekt uczenia

%wypisanie wartości losowych danych wejściowych i efektu uczenia
test
efekt
```