Akademia Górniczo – Hutnicza 17.10.2017 , Kraków

WIMiIP Inżynieria Obliczeniowa

Podstawy Sztucznej Inteligencji

Budowa i działanie perceptronu

sprawozdanie z projektu 1:

Agnieszka Majkut

nr indeksu 286116

Wprowadzenie:

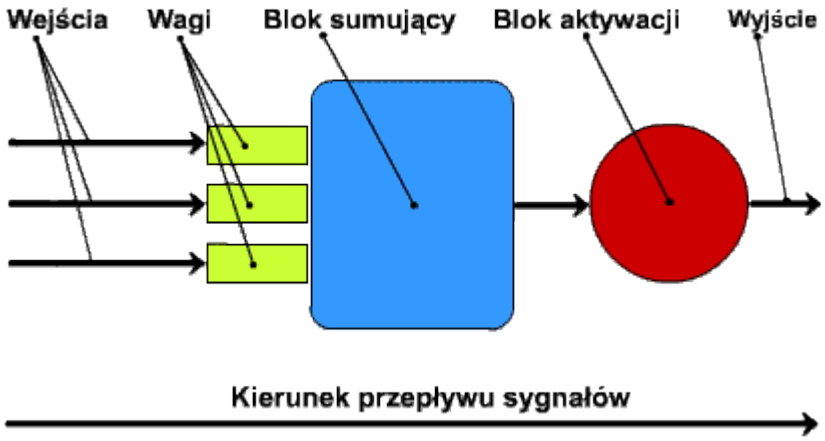
Celem projektu było poznanie budowy i działania perceptronu poprzez implementację oraz uczenie perceptronu realizującego wybraną funkcję logiczną dwoch zmiennych.

Zadania do wykonania:

- implementacja sztucznego neuronu według podanego na wykładzie algorytmu
- wygenerownie danych uczących i testujących wybranej funkcji logicznej dwóch zmiennych
- uczenie preceptronu dla różnej liczby danych uczacych, różnych współczynnikow uczenia
- testowanie perceptronu

Perceptronem nazywamy prosty element obliczeniowy, który sumuje ważone sygnały wejściowe i porównuje tę sumę z progiem aktywacji - w zależności od wyniku perceptron może być albo wzbudzony (wynik 1), albo nie (wynik 0).

Sztuczny neuron - prosty system przetwarzający wartości sygnałów wprowadzanych na jego wejścia w pojedynczą wartość wyjściową, wysyłaną na jego jedynym wyjściu (dokładny sposób funkcjonowania określony jest przez przyjęty model neuronu). Jest to podstawowy element sieci neuronowych, jednej z metod sztucznej inteligencji, pierwowzorem zbudowania sztucznego neuronu był biologiczny neuron.
Schemat sztucznego neuronu:



Algorytm uczenia sieci jest procesem dochodzenia wag do wartości optymalnych – zapewniających odpowiednie reakcje sieci na pokazywane jej sygnały wejściowe, czyli prawidłowe rozwiązanie zadań.

W programie Matlab zaimplementowałam sztuczny neuron realizujący bramkę logiczną OR. Bramka zwraca 1 w przypadku wprowadzenia przynajmniej jednej jedynki.

Funkcje, które wykorzystałam:

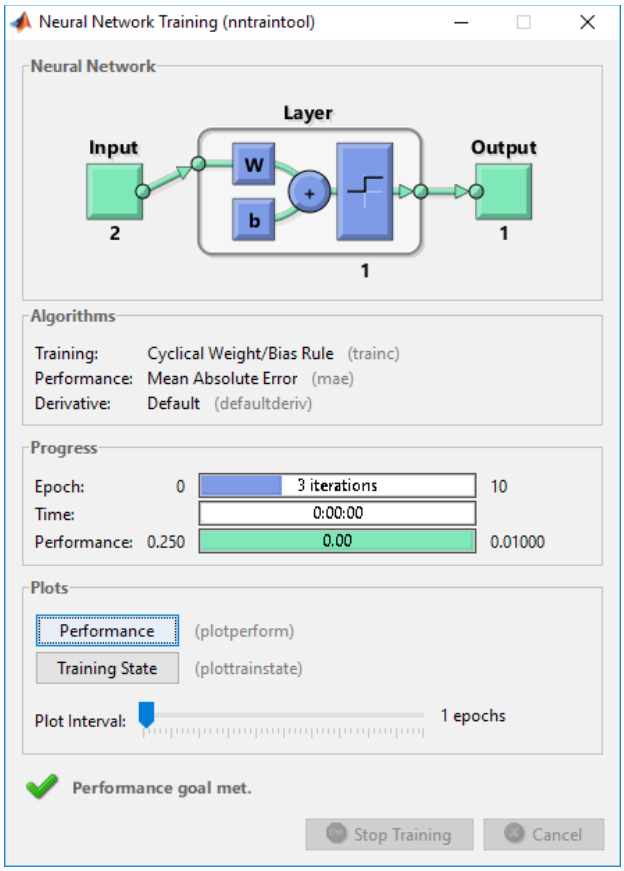
- **newp** – tworzy jednowarstwową sieć neuronową
- **plotpv** – do wyświetlania wektorów
- **sim** – symuluje działanie perceptronu
- **plotpc** – wyświetla granicę decyzyjną dla sieci neuronowej

Net jest to struktura zawierająca opis sieci neuronowej.

Kod programu:

```
close all; clear all; clc;
% Schemat budowy i działania perceptronu o dwóch wejściach.

net = newp([0 1; -2 2], 1, 'hardlim');

% Wejście 1; Wejście 2
wejscie = [0 0 1 1; 0 1 0 1];

% Wyjście tj. warosci wyniku bramki OR
wyjscie = [0 1 1 1];
plotpv(wejscie, wyjscie);

net.name = 'Bramka OR';
net = init(net);
Y = sim(net, wejscie)

% Ustawienia domyślne uczenia perceptronu:
net.trainParam.epochs = 10;
net.trainParam.goal = 0.01;
net.trainParam.mu = 0.001;
net.adaptParam.passes = 10;
net.trainParam.show = 15;

net = train(net, wejscie, wyjscie);
% wagi:
plotpc(net.iw{1,1}, net.b{1})
Y1 = sim(net, wejscie)

% testowanie perceptronu:
test = [0 0 1 1; 0 1 0 1]
efekt = sim(net, test)

disp(net);
```

Otrzymane wyniki:



rys1.

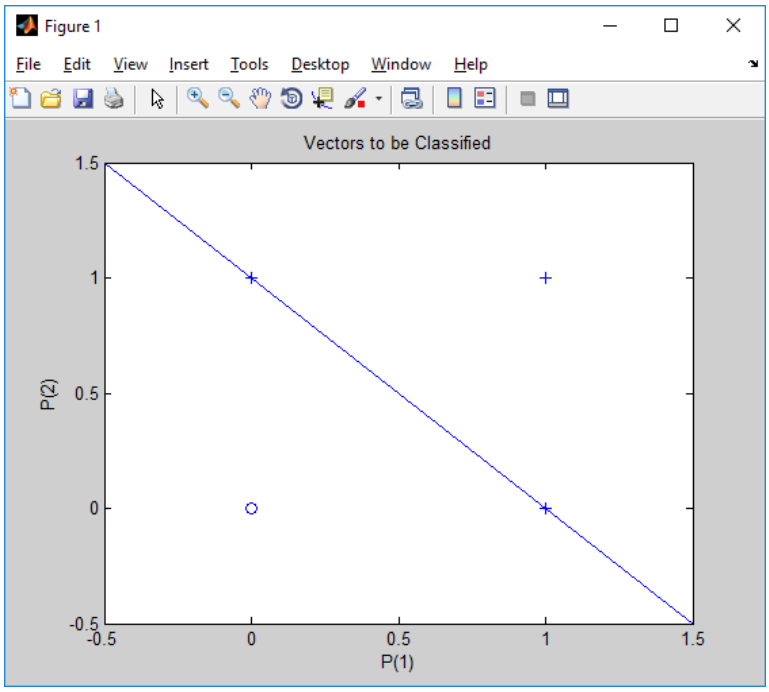
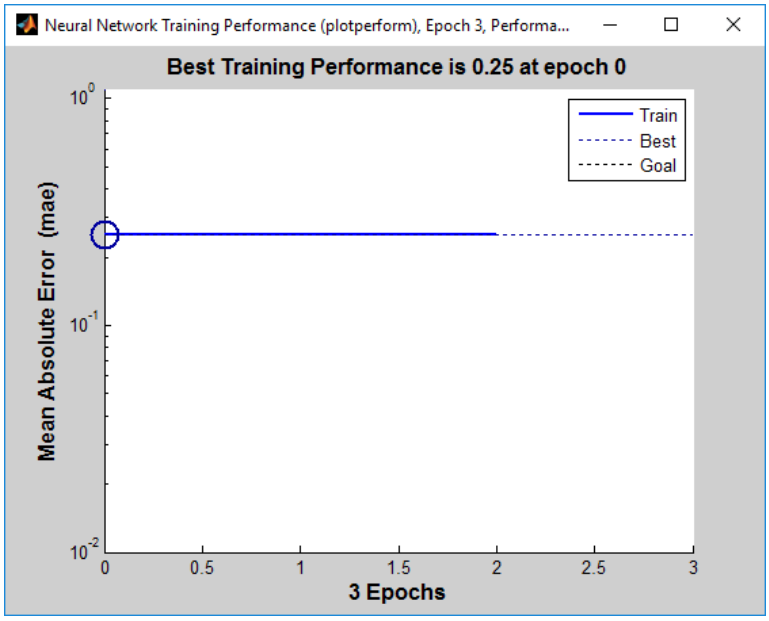


Figure 1
File Edit View Insert Tools Desktop Window Help
Vectors to be Classified
1.5
1
0.5
0
-0.5
P(2)
-0.5
0
0.5
1
1.5
P(1)

rys2.


Neural Network Training Performance (plotperform), Epoch 3, Performa...
Best Training Performance is 0.25 at epoch 0
Train
Best
Goal
Mean Absolute Error (mae)
10^0
10^-1
10^-2
0
0.5
1
1.5
2
2.5
3
3 Epochs

rys3.

Wnioski:

Z rys1 dowiadujemy się, że sieć wykorzystująca bramkę OR nauczyła się w ciągu 3 iteracji. Po sprawdzeniu okna Performance możemy zauważyć, że błąd średni wynosi 0,25 i w rzeczywistości nauka zakończyła się po dwóch iteracjach. Z rys2 możemy odczytać granice decyzyjną. Punkty leżące nad nią są 1, a pozostałe to 0.

Po wykonaniu projektu dowiedziałam się jak działa perceptron i algorytm uczenia. Po przetestowaniu działania programu zauważyłam, że zmiana wag wpływa na długość trwania algorytmu ucznia sieci. Wysokie wagi powodują, że czas nauki wydłuża się nawet kilkukrotnie.