Maciej Krzywda,
Inżynieria Obliczeniowa, IMiIP
Podstawy Sztucznej Inteligencji
nr albumu: 293102

Sprawozdanie 1

Tytuł projektu:

Budowa i działanie perceptronu

- **Cel projektu:**

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

Zadanie jakie mieliśmy wykonać było oparte o implementację sztucznego neuronu w środowisku MATLAB, następnie mieliśmy przygotować dla niego dane pozwalające mu się uczyć (odbywało się to ze zmienną liczbą danych uczących). Ostatnim etapem ćwiczeń który mieliśmy przygotować był test pracy perceptronu.

- **Przebieg ćwiczenia**

1. **Implementacja sztucznego neuronu wg algorytmu podanego na wykładzie lub dowolnego innego (z podaniem źródła).**

2. **Wygenerowanie danych uczących i testujących wybranej funkcji logicznej dwóch zmiennych.**

3. **Uczenie perceptronu dla różnej liczby danych uczących, różnych współczynników uczenia.**

4. **Testowanie perceptronu**

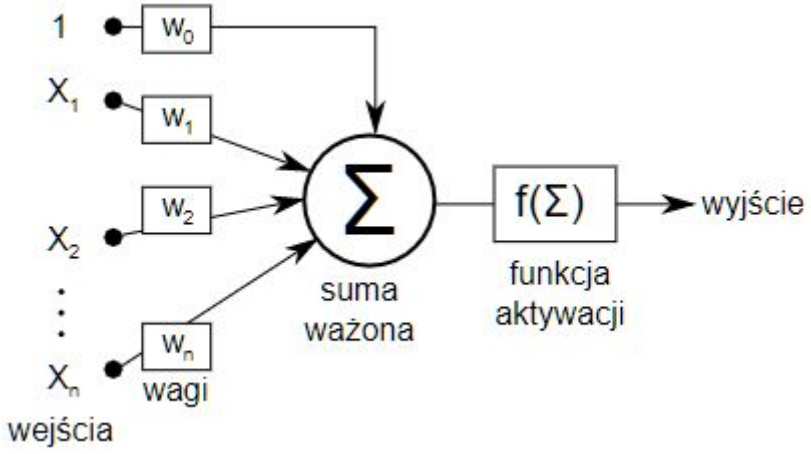
● Część Teoretyczna

Sieć neuronowa – model matematyczny inspirowany budową naturalnych neuronów znajdujących się w mózgu człowieka. Modele takie składają się z neuronów które to realizują podstawowe obliczenia na swoim wejściu i nazywamy je sztucznymi neuronami.

Systemy oparte o sieci neuronowe wykazują zdolności uczenia się, na podstawie przykładów uczymy taką sieć aby następnie ona już samodzielnie analizowała nowe dane przedstawione w innym lecz analogicznym kontekście.

Sieć neuronowa spisuje się świetnie w zastosowaniach przy których klasyczny komputer nie mógłby sobie poradzić tzn. Uogólnienie posiadanej wiedzy na inne przypadki włącznie z wymyśleniem nowego sposobu rozwiązania nieznanego jej dotychczas problemu, odporność na uszkodzenia – niepełne, błędne dane, sprawne działanie nawet gdy część jej elementów nie działa poprawnie lub nie działa w ogóle. Sieci neuronowe nadają się idealnie do rozwiązywania problemów przy których klasyczny język programowania nie daje sobie rady tzn. brakuje odpowiedniego algorytmu.

Neuron McCullocha-Pittsa – jest to jeden z matematycznych modeli neuronu oraz podstawowy budulec wspomnianej wyżej sieci neuronowej (perceptron). Posiada wiele wejść gdzie każde ma swoją wagę (tzw. Wagę wejścia), oraz jedno wyjście którego waga jest obliczona dodatkowo za pomocą wzoru (funkcja aktywacji).



Perceptron – jest to podstawowa sieć neuronowa składająca się z jednego lub wielu neuronów McCullocha-Pittsa. Perceptron jest w stanie określić przynależność parametrów wejściowy do jednej z dwóch klas lecz jego ograniczeniem jest fakt że może zrobić to wyłącznie dla klas liniowo separowalnych.

Uczenie sieci

Sieci neuronowe mają zdolność do uczenia się, czyli zdolność do samodzielnego dostosowywania współczynników wagowych. czyli inaczej mówiąc uczenie sieci jest to wymuszenie na niej określonego zareagowania na sygnały wejściowe. Dzięki temu mówi się ze mają one właśnie charakter AI, bo potrafią samodzielnie dostosowania sie do zmieniających się do warunków. Celem uczenia jest taki dobór wag w poszczególnych neuronach aby sieci mogła rozwiązywać stawiane przed nią problemy. Uczenie polega na modyfikacji wag.

Na potrzeby projektu wykorzystana została bramka AND której tablice prawdy przy odpowiednich wejściach przedstawiłem poniżej:

Bramka logiczna AND:

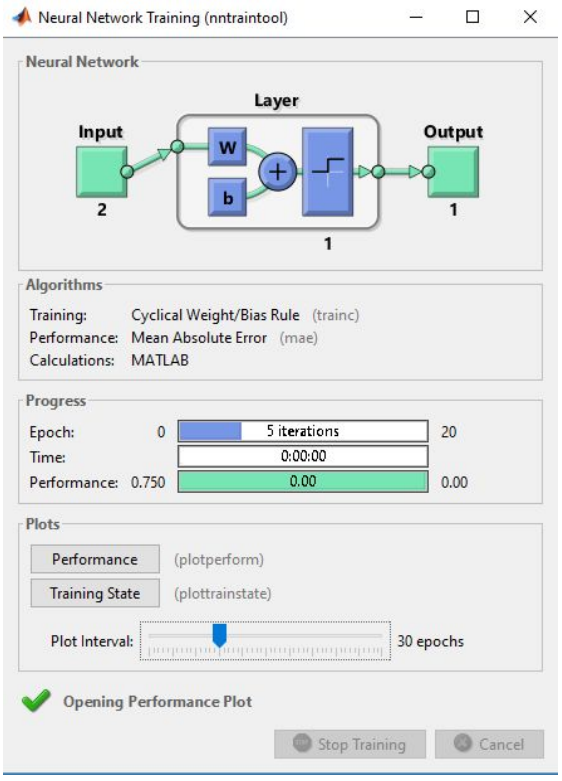
Inputs		Output
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

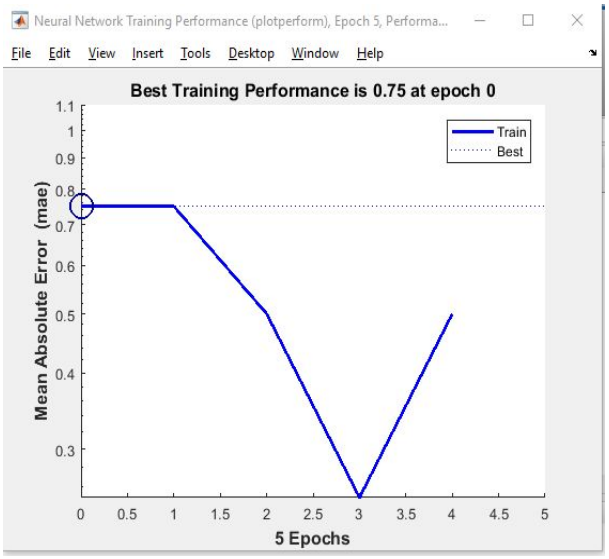
Oczywiście nie stoi nic na przeszkodzie zastosowania innej bramki logicznej, wystarczy później odpowiednio dobrać wektor W i T tak by spełniał odpowiednią tablice prawdy dla danej bramki logicznej.

- **Listing programu z opisem poszczególnych kroków.**

```
%(Tworzenie pojedynczego neuronu o podanych przedziałach wartości.W moim
przypadku są to dwa wejścia: pierwsze [0,1] oraz drugie [-2,2] %)
            net=newp([0 1; -2 2],1);

%(Tworzenie wektorów wynikowych W (opisuje wartości na wejściach) oraz T
(opisuje wartości wynikowe). Otrzymujemy zestaw trzech wektorów które opisuje
działanie bramki logicznej, której tablicę prawdy przedstawiłem już wcześniej.
%)
            W=[0 0 1 1; 0 1 0 1];
            T=[0 0 0 1];

% Inicjujemy sieć perceptronowa w której wartość wag i progów jest losowa.
            net=init(net);

%Dokonujemy symulacji przed treningiem
            INPUT=sim(net,W)

% Określamy liczbę epok dla której należy wykonać w treningu sieci.
            net.trainParam.epochs=20;

% Trenujemy perceptron.
            net=train(net,W,T);

% Symulujemy sieć neuronowa dla wartości wcześniej wytrenowanych.
            OUTPUT=sim(net,W)
```

```
INPUT =

     1     1     1     1

OUTPUT =

     0     0     0     1
```





MAE - absolutny błąd to ilość błędów w pomiarach

Wybierając opcję Plot > Performance uzyskałam wykres „uczenia się” sieci, widzimy że w epoce zerowej MAE wynosi 0.75 jest ona stabilna do epoki pierwszej, później błąd znacznie maleje i w epoce trzeciej następuje nauczenie sieci.

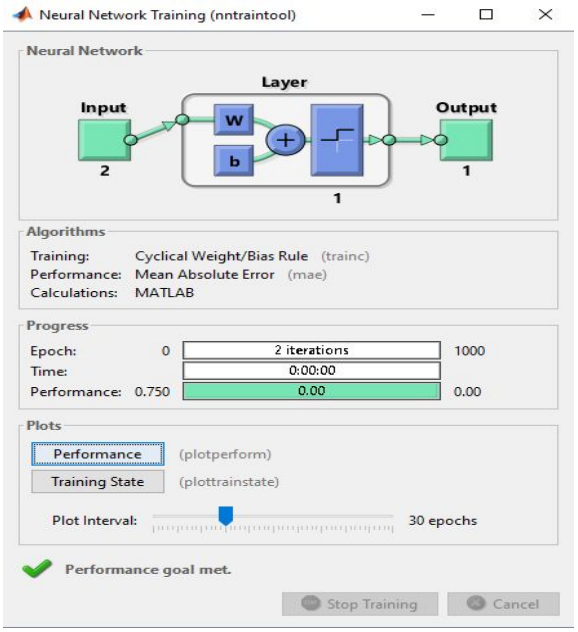
Widzimy jednak, że od epoki trzeciej rośnie nam MAE, oznacza to jednak, że nastąpiło „przeuczenie” sieci, jest to zjawisko polegające na nadmiernym dopasowaniu się sieci do punktów uczących, któremu towarzyszy błędne działanie sieci dla danych nie prezentowanych w trakcie uczenia. Przeuczenie pojawia się w przypadku zbyt długiego uczenia (działania algorytmu uczącego) lub wówczas, gdy zastosowana sieć jest zbyt złożona w porównaniu ze złożonością problemu lub liczbą dostępnych danych uczących.

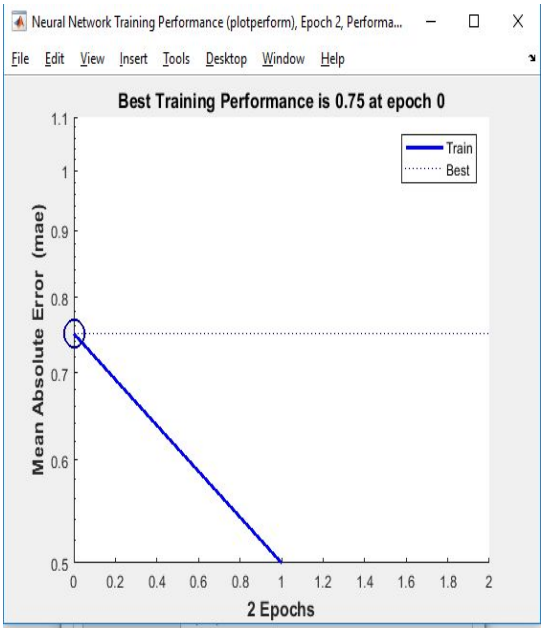
Funkcja TRAIN wygenerowała zamierzony efekt, sieć została nauczona, zamierzony cel chciałam uzyskać w 30 epokach (iteracjach) w tym przypadku sieć dokonała tego do 3 epoki, później jednak wystąpiło przeuczenie.

Możemy również dokonać uczenia dla zmiennych współczynników dodając odpowiednie linie zmieniające wagi:

Całość kodu będzie wyglądać bardzo podobnie do poniższego (w każdej zmianie wag będziemy dokonywać modyfikacji lini kodu które zostały pogrubione)

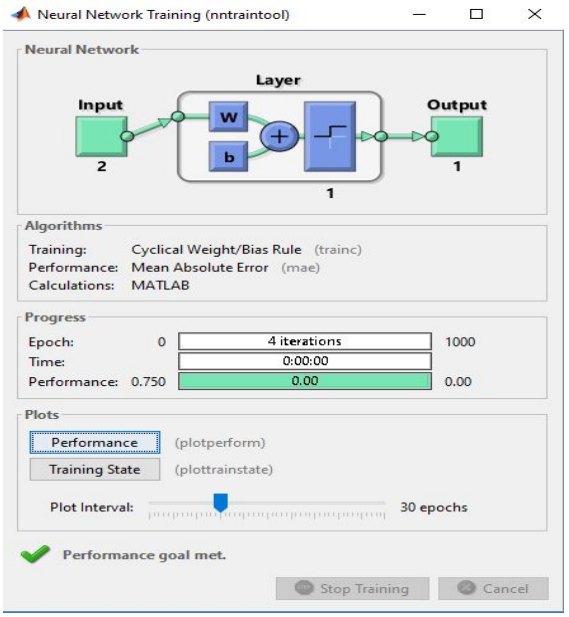
```
net = newp([0 1; -2 2], 1);
W=[0 0 1 1; 0 1 0 1];
T = [0 0 0 1];
% Uczenie dla zmiennych współczynników
% Pierwsza zmiana wag oraz odchylenia:
net.IW{1,1} = [4 4];
net.b{1} = [2]
INPUT = sim(net,W)
net = train(net,W,T)
OUTPUT = sim(net,W)
```

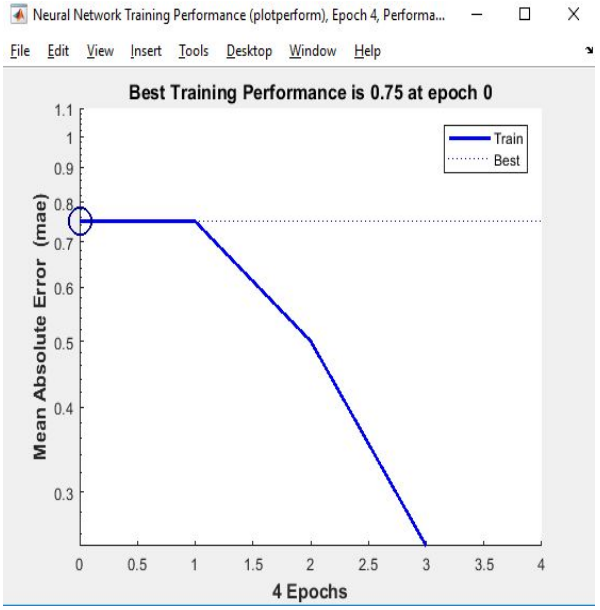




Druga zmiana wag:

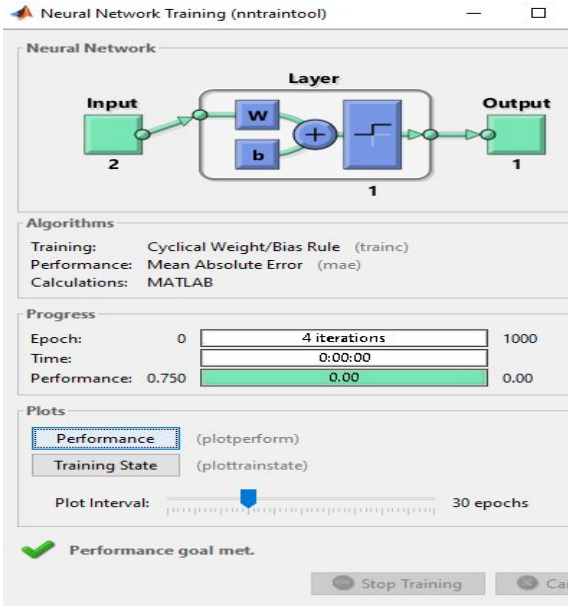
```
net.IW{1,1} = [5 7];
net.b{1} = [5]
```

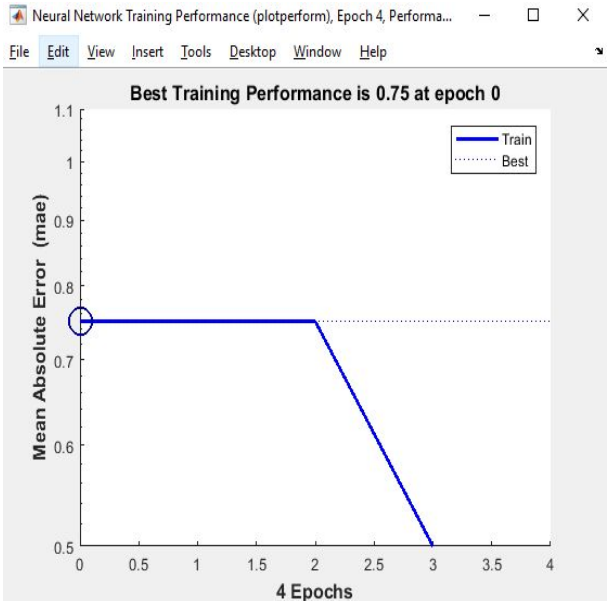




Trzecia zmiana wag:

```
net.IW{1,1} = [8 8];
net.b{1} = [6]
```





Wnioski

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

- Obserwując otrzymane wykresy możemy stwierdzić, że proces nauki został wykonany efektywnie, gdyż wynik otrzymywaliśmy już po kilku iteracjach.

- Jak widzimy na powyższych przykładach, aby zmienić bramkę logiczną wystarczy zmienić wartość W oraz T.

- Podane niżej wywołanie funkcji tworzy sieć zawierającą pojedynczy neuron o dwóch wejściach. Zakres wartości pierwszego wejścia to [0, 1] drugiego [-2, 2].

- Możemy zaobserwować że zmieniając współczynniki uczenia, powodujemy znaczne różnice w zachodzeniu tego procesu. W zależności od zastosowanej bramki logicznej, zmiany tych współczynników skutkują tym, że uczenie zachodzi po mniejszej lub większej ilości iteracji oraz zmienia się efektywność samego uczenia. Na zamieszczonych wykresach wydajności widzimy wydajność uczenia się w zależności od ilości iteracji.

- Funkcje AND daje zbiór separowalny liniowo, a więc możliwe do realizacji perceptronem.