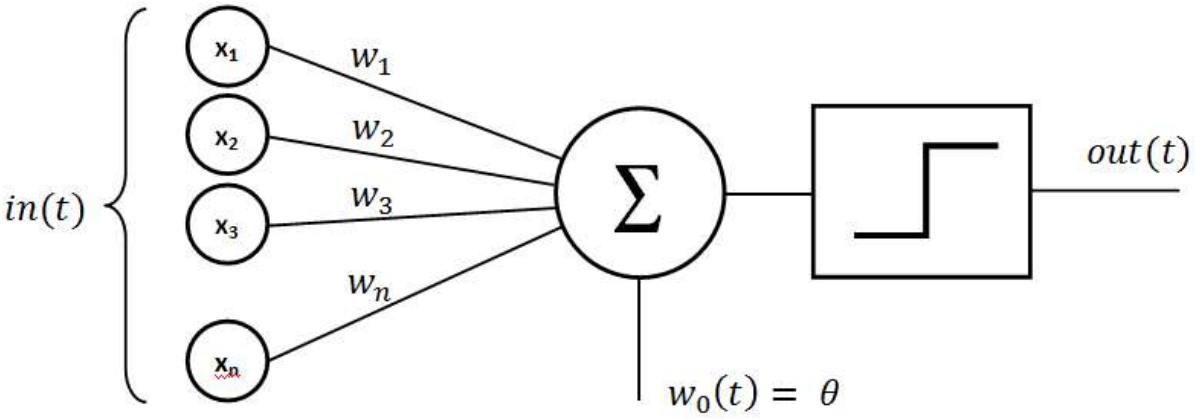
| 18.12.2017 r. | Izabela Musztyfaga | Podstawy Sztucznej Inteligencji |
|---|---|---|
| Scenariusz 1 – budowa i działanie perceptronu | | |

Perceptron jest najprostszą możliwą siecią neuronową, która zbudowana jest z jednego lub wielu niezależnych neuronów.




*Rys. 1. Perceptron (źródło: [https://pl.wikipedia.org/wiki/Perceptron](https://pl.wikipedia.org/wiki/Perceptron))*

## Algorytm uczenia perceptronu

Dane uczące to proste funkcje logiczne AND oraz OR. Dane uczące można potraktować również jako testujące, a sam proces uczenia jako proces testowania *„dynamicznego i przez obserwację”* [1].

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Tab.1 Logiczny AND

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Tab..2 Logiczny OR

Algorytm uczenia perceptronu oparty jest na doborze odpowiednich wag (na Rys. 1 $w_1,w_2,\ldots,w_n$). Początkowo wagi określamy losowo. Następnie dla każdego przypadku funkcji logicznej określana jest odpowiedź perceptronu. Jeżeli jest ona

1 Źródło: http://testerzy.pl/materialy/index.php?file=marek-zukowicz-uczenie-sztucznych+sieci-neuronowych-a-testowanie.pdf

nieprawidłowa, wagi są modyfikowane aż do skutku.

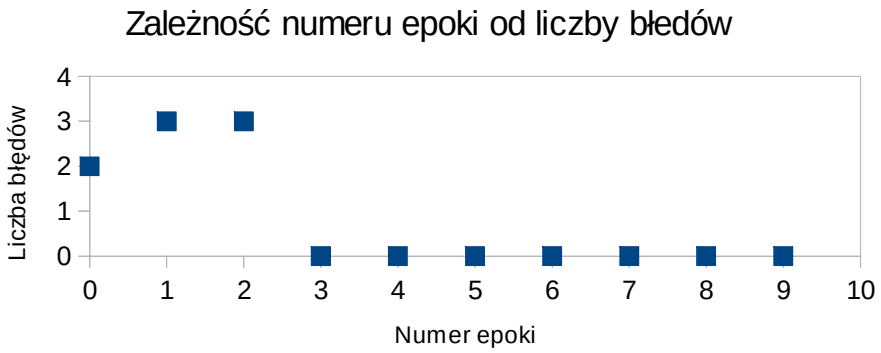
W wykorzystanym algorytmie na wyjściu otrzymujemy kolejne numery epok, wykorzystany współczynnik uczenia, oraz liczbę popełnionych błędów.

## Dla funkcji logicznej AND

**Współczynnik uczenia: 0.1**

| Numer epoki | Liczba błędów |
|---|---|
| 0 | 2 |
| 1 | 3 |
| 2 | 3 |
| 3 | 0 |
| 4 | 0 |
| 5 | 0 |
| 6 | 0 |
| 7 | 0 |
| 8 | 0 |
| 9 | 0 |

*Tab. 3 - Wyniki dla współczynnika 0.1*




*Wykres 1 - Dane z tab 3*

**Współczynnik uczenia: 0.4**

| Numer epoki | Liczba błędów |
|---|---|
| 0 | 2 |
| 1 | 3 |
| 2 | 3 |
| 3 | 0 |
| 4 | 0 |
| 5 | 0 |
| 6 | 0 |
| 7 | 0 |
| 8 | 0 |
| 9 | 0 |

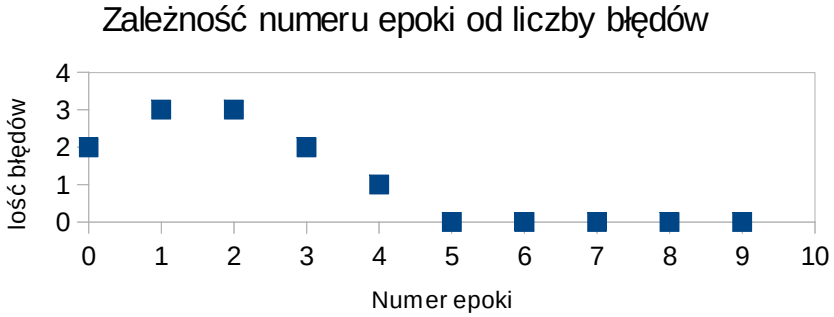
*Tab. 4 - Wyniki dla współczynnika 0.4*


*Wykres 2 - Dane z tab 4*

**Współczynnik uczenia: 0.7**

| Numer epoki | Liczba błędów |
|---|---|
| 0 | 2 |
| 1 | 3 |
| 2 | 3 |
| 3 | 2 |
| 4 | 1 |
| 5 | 0 |
| 6 | 0 |
| 7 | 0 |
| 8 | 0 |
| 9 | 0 |

*Tab. 5 - Wyniki dla współczynnika 0.7*




*Wykres 3 - Dane z tab 5*

Dla współczynników uczenia 0,1 oraz 0,4 proces przebiega w dokładnie ten sam sposób – ilość

błędów w każdej z 10 epok jest taka sama. Zmiana następuje w chwili, gdy współczynnik uczenia wynosi 0,7. O ile w poprzednich przypadkach od 3 epoki liczba ta wynosiła 0, tak w tym przypadku potrzeba do tego 5 epok. Oczywiście od pewnego momentu popełniane błędy są coraz mniejsze.

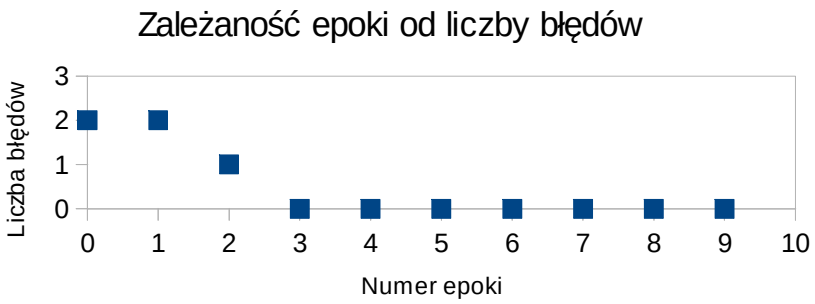
Przyglądając się wykresom można wywnioskować, że sam układ punktów w pewnym momencie przypomina funkcję trygonometryczną.

## Dla funkcji logicznej OR

**Współczynnik uczenia: 0.1**

| Numer epoki | Liczba błędów |
|---|---|
| 0 | 2 |
| 1 | 2 |
| 2 | 1 |
| 3 | 0 |
| 4 | 0 |
| 5 | 0 |
| 6 | 0 |
| 7 | 0 |
| 8 | 0 |
| 9 | 0 |

*Tab. 6 - Wyniki dla współczynnika 0.1*

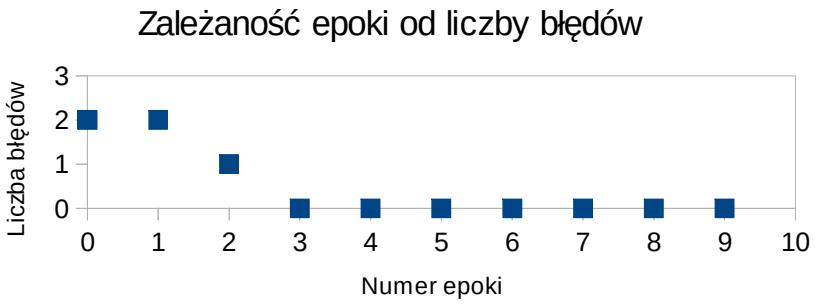



*Wykres 4 - Dane z tab.6*

**Współczynnik uczenia: 0.4**

| Numer epoki | Liczba błędów |
|---|---|
| 0 | 2 |
| 1 | 2 |
| 2 | 1 |
| 3 | 0 |
| 4 | 0 |
| 5 | 0 |
| 6 | 0 |
| 7 | 0 |
| 8 | 0 |
| 9 | 0 |

*Tab. 7 - Wyniki dla współczynnika 0.4*

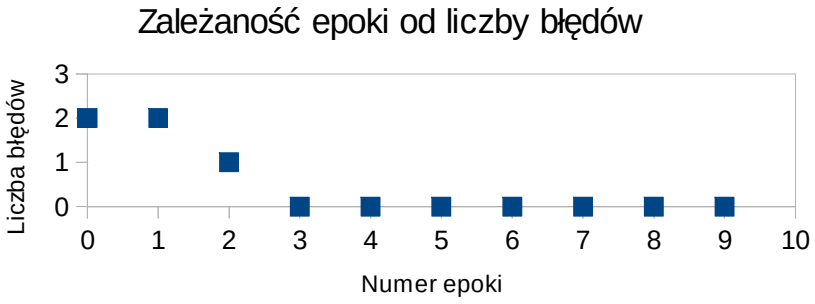



*Wykres 5 - Dane z tab.7*

**Współczynnik uczenia: 0.7**

| Numer epoki | Liczba błędów |
|---|---|
| 0 | 2 |
| 1 | 2 |
| 2 | 1 |
| 3 | 0 |
| 4 | 0 |
| 5 | 0 |
| 6 | 0 |
| 7 | 0 |
| 8 | 0 |
| 9 | 0 |

*Tab. 8 - Wyniki dla współczynnika 0.7*




*Wykres 6 - Dane z tab.8*

W tym przypadku można zaobserwować, że proces uczenia i testowania przebiegł dokładnie w ten sam sposób dla każdego z trzech współczynników uczenia: 0,1 , 0,4 i 0,7 – ilość błędów jest dokładnie taka sama w kolejnych epokach uczenia perceptronu.

Tak samo jak w przypadku poprzedniej funkcji – od pewnego momentu liczba błędów w każdej kolejnej epoce spada, aby w końcu dotrzeć do 0. Wtedy możemy postawić stwierdzenie, że perceptron nauczył się funkcji.

## Wnioski

W procesie uczenia perceptronu bardzo dużą rolę odgrywa współczynnik uczenia. Można zauważyć, że sam wynik zależy tylko od niego. Jest on (przynajmniej na potrzeby wykonywanego ćwiczenia) dobierany eksperymentalnie.

Gdy liczba błędów w kolejnych epokach będzie wynosiła 0 możemy uznać, że perceptron został poprawnie nauczony. Dla małej ilości danych użytych do wykonania ćwiczenia perceptron uczy się szybko, bo średnio po 3-5 epokach.

## Listing kodu[2]

```
def predict(row, weights):
    activation = weights[0]
    for i in range(len(row) - 1):                #obliczanie funkcji aktywacji
        activation += weights[i + 1] * row[i]
    return 1.0 if activation >= 0.0 else 0.0

# Obliczanie/trenowanie wag funkcji
def train_weights(train, l_rate, n_epoch):
    weights = [0.0 for i in range(len(train[0]))]
    for epoch in range(n_epoch):
        sum_error = 0.0
        for row in train:
            prediction = predict(row, weights)
            error = row[-1] - prediction
            sum_error += error ** 2
            weights[0] = weights[0] + l_rate * error
            for i in range(len(row) - 1):
                weights[i + 1] = weights[i + 1] + l_rate * error * row[i]
        print('>epoch=%d, lrate=%.3f, error=%.3f' % (epoch, l_rate, sum_error))
    return weights
# Dane
#funkcja której chcemy nauczyć
dataset = [[0, 0, 0],
           [0, 1, 1],
           [1, 0, 1],
           [1, 1, 1]
         ]
l_rate = 0.1            #współczynnik uczenia się
n_epoch = 10            #liczba epok

#drukowanie wyniku
weights = train_weights(dataset, l_rate, n_epoch)
```

2 Źródło kodu: https://machinelearningmastery.com/implement-perceptron-algorithm-scratch-python/