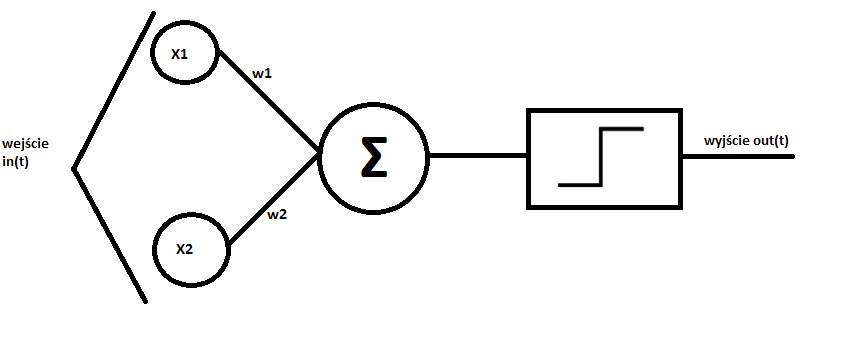
Dariusz Nowak, 278679

**Scenariusz nr 1**

**Budowa i działanie perceptronu**



Rysunek 1. neuron McCullocha-Pittsa

Perceptron jest to podstawowa sieć neuronowa składająca się z jednego lub wielu neuronów. Ogólna zasada działania sprowadza się do odpowiedniego dobrania wag w(w1,w2) oraz połączeń pomiędzy warstwami neuronów w taki sposób, aby uzyskać oczekiwany rezultat końcowy – wartość wyjściową.

Algorytmu uczenia perceptronu - tzn. automatycznego doboru wag na podstawie napływających przykładów:

• Inicjujemy wagi losowo.

• Dla każdego przykładu uczącego obliczamy odpowiedź perceptronu.

• Jeśli odpowiedź perceptronu jest nieprawidłowa, to modyfikujemy wagi:

w1 += n \* (d-y) \* x1

w2 += n \* (d-y) \* x2

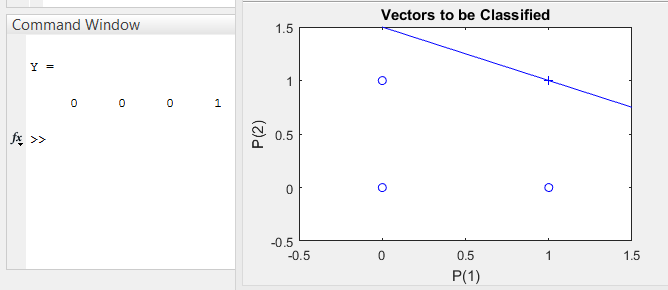
b += n \* (d-y)

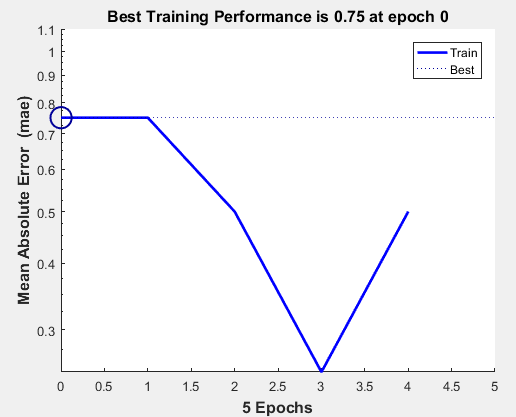
gdzie n jest niewielkim współczynnikiem uczenia (n > 0), d - oczekiwana odpowiedź a y - odpowiedź neuronu.

Implementacja bramki logicznej AND z wykorzystaniem wtyczki neural network pakietu matlab:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | close all**;** clear all**;** clc**;**  net **=** newp**([2** **3;** **-1** **2],** **1);**  P1 **=** **[0** **1** **0** **1;** **0** **0** **1** **1];**  T1 **=** **[0** **0** **0** **1];**  net **=** init**(**net**);**  net **=** train**(**net**,**P1**,**T1**);**  %Y **=** sim**(**net**,**P1**)**  %P2 **=** **[1** **0** **1** **1;** **1** **1** **0** **1];**  Y **=** sim**(**net**,**P1**)**  plotpv**(**P1**,**T1**);**  plotpc**(**net**.**IW**{1},**net**.**b**{1});** |

Zwrot:

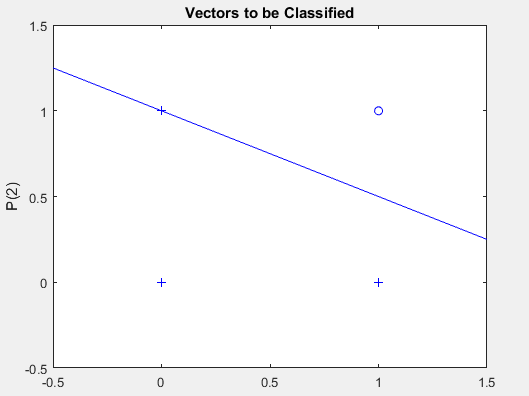


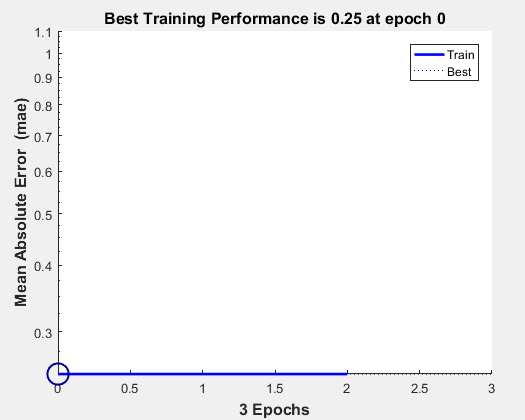


Rysunek wykres błędu bramki AND

Implementacja bramki logicznej OR z wykorzystaniem wtyczki neural network pakietu matlab:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | close all**;** clear all**;** clc**;**  net **=** newp**([2** **3;** **-1** **2],** **1);**  P1 **=** **[0** **1** **0** **1;** **0** **0** **1** **1];**  T1 **=** **[0** **1** **1** **1];**  net **=** init**(**net**);**  net **=** train**(**net**,**P1**,**T1**);**  %Y **=** sim**(**net**,**P1**)**  %P2 **=** **[1** **0** **1** **1;** **1** **1** **0** **1];**  Y **=** sim**(**net**,**P1**)**  plotpv**(**P1**,**T1**);**  plotpc**(**net**.**IW**{1},**net**.**b**{1});** |

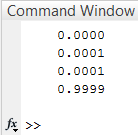




Rysunek wykres błędu bramki OR

Prymitywna implementacja neuronu typu sigmoidalnego w pakiecie matlab realizującego bramkę AND:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | close all**;** clear all**;** clc**;**  input **=** **[0** **0;** **0** **1;** **1** **0;** **1** **1];**  numIn **=** **4;**  desired\_out **=** **[0;0;0;1];**  bias **=** **-1;**  coeff **=** **0.7;**  rand**(**'state'**,**sum**(100\***clock**));**  weights **=** **-1\*2.\***rand**(3,1);**  iterations **=** **100000;**  **for** i **=** **1:**iterations  out **=** zeros**(4,1);**  **for** j **=** **1:**numIn  y **=** bias**\***weights**(1,1)+...**  input**(**j**,1)\***weights**(2,1)+**input**(**j**,2)\***weights**(3,1);**  out**(**j**)** **=** **1/(1+**exp**(-**y**));**  delta **=** desired\_out**(**j**)-**out**(**j**);**  weights**(1,1)** **=** weights**(1,1)+**coeff**\***bias**\***delta**;**  weights**(2,1)** **=** weights**(2,1)+**coeff**\***input**(**j**,1)\***delta**;**  weights**(3,1)** **=** weights**(3,1)+**coeff**\***input**(**j**,2)\***delta**;**  end  end |

Zwrot: 

**Wnioski**

Na podstawie wyników implementacji kodu można stwierdzić, że na „skuteczność” neuronu wpływa ilość epok, czyli przejść pętli w których korygowane są początkowo losowo wybrane wagi. Jednak w przypadku bramki logicznej operującej jedynie na dwóch liczbach binarnych duża ilość przejść pętli nie jest wymagana. Na wykresie ilustrującym wielkość błędu dla bramki logicznej AND(rysunek2) zaobserwować można, że błąd spadał do 3ciego przejścia pętli, a następnie zaczął rosnąć. Natomiast w bramce OR(rysunek3), błąd od pierwszego przejścia pętli wynosił 0. Pętle kończą się w momencie, w którym błąd zaczyna rosnąć, ponieważ następuje przeuczenie neuronu.