Machine Learning

Indah Agustien Siradjuddin

Artificial Neural Networks / Jaringan Syaraf Tiruan

Semester Gasal 2019-2020

Jaringan Syaraf

sumber :

Kelebihan :

1. Pembelajaran
2. Beradaptasi
3. Generalisasi
4. *Pararelism*

Implementasi

Pengenalan Suara

Identifikasi Atribut

Neuron

Abstraksi neuron - NN

Neuron memetakan input dengan dimensi *d* menjadi [0,1] or [-1,1] :

$$f : \mathbb{R}^d \rightarrow [0, 1]$$

atau

$$f : \mathbb{R}^d \rightarrow [-1, 1]$$

Pemrosesan input :

1. Hitung net input

$$net = x_1w_1 + x_2w_2 + \ldots + x_dw_d$$

$$net = \sum_{i=1}^{d} x_i w_i$$

dimana :

$$x_{1..d} = \text{sinyal input/fitur dengan jumlah dimensi } d$$

$$w_{1..d} = \text{bobot setiap sinyal input}$$

2. Hitung nilai aktivasi

$$y = f(net)$$

$$y = f\big(\sum_{i=1}^{d} x_i w_i\big)$$

dimana $f(net)$ adalah fungsi aktivasi

Activation Function

1. Fungsi *Linear*

$$f(netInput) = netInput$$

2. Fungsi *Stepwise*

$$f(netInput) = \begin{cases} \gamma_1, & \text{if } netinput \geq \theta \\ \gamma_2, & \text{if } netinput < \theta \end{cases}$$

3. Fungsi *Sigmoid*

$$f(netInput) = \frac{1}{1 + e^{-\lambda(netInput - \theta)}}$$

Perceptron

Satu *layer*, yaitu *output layer* (terdiri dari sejumlah neuron)

Agar perceptron dapat men-generalisasi sinyal input, maka diperlukan proses pelatihan/pembelajaran:

1. Model (arsitektur jaringan syaraf dan bobotnya) dapat dibentuk setelah proses pembelajaran/pelatihan.
2. bobot di*update* agar mendapatkan error minimal

Algoritma Pembelajaran Perceptron

1. Inisialisasi Bobot
2. Hitung net input

$$y = f\big(\sum_{i=1}^{d} x_i w_i\big)$$

3. Hitung output atau nilai aktivasi dari neuron output berdasarkan fungsi aktivasi, misal: fungsi *stepwise*
4. update bobot :

$$w_i(t+1) = w_i \text{ ; if output}==\text{targetOutput}$$
$$w_i(t+1) = w_i + x_i(t) \text{ ; if output} < \text{targetOutput}$$
$$w_i(t+1) = w_i - x_i(t) \text{ ; if output} > \text{targetOutput}$$

5. Lakukan langkah 2-4 sampai jumlah epoch yang telah ditentukan atau error minimum telah dicapai

Latihan

Buat sebuah model perceptron dengan menggunakan data logical AND, dan asumsi bobot awal adalah $w_1 = 0.1$ and $w_2 = 0.2$, jumlah epoch = 3

X1	X2	Y
1	1	1
1	0	0
0	1	0
0	0	0

Learning on Perceptron

- **Algoritma-1**

  1. Inisialisasi bobot dan bias (w_0)
  2. hitung net input

  $$y = f(\sum_{i=0}^{d} x_i w_i)$$

  3. hitung output ata nilai aktivasi neuron output berdasarkan fungsi aktivasi, misal : fungsi *stepwise*
  4. update bobot :

  $$w_i(t+1) = w_i \text{ ; if output==targetOutput}$$
  $$w_i(t+1) = w_i + x_i(t) \text{ ; jika output} < \text{targetOutput}$$
  $$w_i(t+1) = w_i - x_i(t) \text{ ; jika output} > \text{targetOutput}$$

  5. Lakukan langkah 2-4 sampai jumlah epoch tertentu atau error minimum telah tercapai

- **Algoritma-2**

  1. update bobot :

  $$w_i(t+1) = w_i \text{ ; if output==targetOutput}$$
  $$w_i(t+1) = w_i + \eta x_i(t) \text{ ; jika output} < \text{targetOutput}$$
  $$w_i(t+1) = w_i - \eta x_i(t) \text{ ; jika output} > \text{targetOutput}$$

- **Algoritma-3**

  1. update bobot :

  $$\Delta = d(t) - y(t)$$
  $$\text{; d(t) adalah data target dan y(t) adalah output sebenarnya}$$
  $$w_i(t+1) = w_i(t) + \eta \Delta x_i(t)$$

Perceptron dengan scikit

In [1]:

```
import numpy as np
data=np.array([[1,1],[1,0],[0,1],[0,0]]) # data
targetAnd=np.array([1,0,0,0])
```

In [2]:

```
from sklearn.linear_model import Perceptron
```

In [3]:

```
clf = Perceptron(tol=1e-3, random_state=0)
clf.fit(data, targetAnd)
```

Out[3]:

```
Perceptron(alpha=0.0001, class_weight=None, early_stopping=False, eta0=1.
0,
           fit_intercept=True, max_iter=1000, n_iter_no_change=5, n_jobs=N
one,
           penalty=None, random_state=0, shuffle=True, tol=0.001,
           validation_fraction=0.1, verbose=0, warm_start=False)
```

In [4]:

```
print(clf.coef_)
print(clf.intercept_)
```

```
[[2. 2.]]
[-2.]
```

In [5]:

```
#testing
clf.predict(data)
```

Out[5]:

```
array([1, 0, 0, 0])
```

Code : perceptron

In [6]:

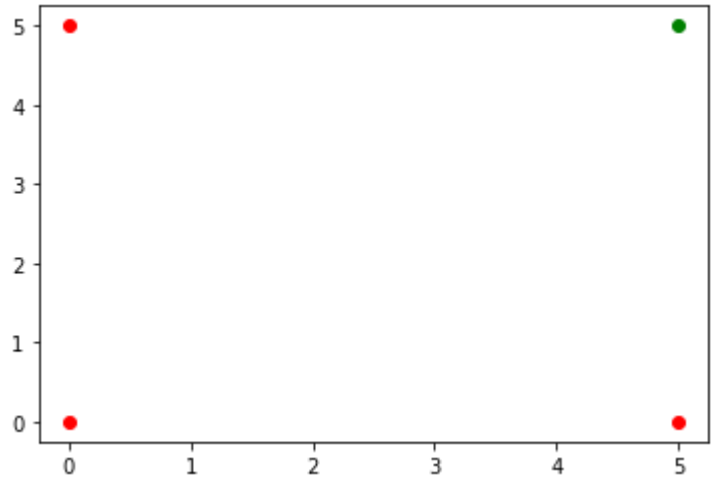
```
import numpy as np
import matplotlib.pyplot as matPlot
#inisialisasi data
data=np.array([[1,1,1],[1,1,0],[1,0,1],[1,0,0]]) # bias dan data
targetAnd=np.array([1,0,0,0])
targetOr=np.array([1,1,1,0])
targetXor=np.array([0,1,1,0])
targetAndNOT=np.array([0,1,0,0])
targetnotPOrQ=np.array([1,0,1,1])
targetPOrnotQ=np.array([1,1,0,1])
```

In [7]:

```
import perceptron
```

In [8]:

```
perceptron.plotting(data,targetAnd)
```



In [9]:

```
weight=perceptron.perceptronLearning(data,targetAnd)
```

```
data= [[1 1 1]
 [1 1 0]
 [1 0 1]
 [1 0 0]]
target= [1 0 0 0]
bobot= [[0.3]
 [0.2]
 [0.4]]
bobot= [[0.3]
 [0.2]
 [0.4]]
jumlah epoch10
[[0.3 0.3]
 [0.2 0.2]
 [0.4 0.4]]
[[ 0.3  0.3 -0.7]
 [ 0.2  0.2 -0.8]
 [ 0.4  0.4  0.4]]
[[ 0.3  0.3 -0.7 -0.7]
 [ 0.2  0.2 -0.8 -0.8]
 [ 0.4  0.4  0.4  0.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8]
 [ 0.4  0.4  0.4  0.4  0.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
  -1.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
   0.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
   1.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
  -1.7 -1.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
   0.2  0.2]
```

```
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
   1.4  1.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
  -1.7 -1.7 -1.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
   0.2  0.2  0.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
   1.4  1.4  1.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
  -1.7 -1.7 -1.7 -0.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
   0.2  0.2  0.2  1.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
   1.4  1.4  1.4  2.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
  -1.7 -1.7 -1.7 -0.7 -1.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
   0.2  0.2  0.2  1.2  0.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
   1.4  1.4  1.4  2.4  2.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
  -1.7 -1.7 -1.7 -0.7 -1.7 -2.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
   0.2  0.2  0.2  1.2  0.2  0.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
   1.4  1.4  1.4  2.4  2.4  1.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
  -1.7 -1.7 -1.7 -0.7 -1.7 -2.7 -2.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
   0.2  0.2  0.2  1.2  0.2  0.2  0.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
   1.4  1.4  1.4  2.4  2.4  1.4  1.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
  -1.7 -1.7 -1.7 -0.7 -1.7 -2.7 -2.7 -1.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
   0.2  0.2  0.2  1.2  0.2  0.2  0.2  1.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
   1.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
  -1.7 -1.7 -1.7 -0.7 -1.7 -2.7 -2.7 -1.7 -1.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
   0.2  0.2  0.2  1.2  0.2  0.2  0.2  1.2  1.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
   1.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
  -1.7 -1.7 -1.7 -0.7 -1.7 -2.7 -2.7 -1.7 -1.7 -2.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
   0.2  0.2  0.2  1.2  0.2  0.2  0.2  1.2  1.2  1.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
   1.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  1.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
  -1.7 -1.7 -1.7 -0.7 -1.7 -2.7 -2.7 -1.7 -1.7 -2.7 -2.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
   0.2  0.2  0.2  1.2  0.2  0.2  0.2  1.2  1.2  1.2  1.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
   1.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  1.4  1.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
  -1.7 -1.7 -1.7 -0.7 -1.7 -2.7 -2.7 -1.7 -1.7 -2.7 -2.7 -1.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
   0.2  0.2  0.2  1.2  0.2  0.2  0.2  1.2  1.2  1.2  1.2  2.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
```

```
   1.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
  -1.7 -1.7 -1.7 -0.7 -1.7 -2.7 -2.7 -1.7 -1.7 -2.7 -2.7 -1.7 -2.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
   0.2  0.2  0.2  1.2  0.2  0.2  0.2  1.2  1.2  1.2  1.2  2.2  1.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
   1.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
  -1.7 -1.7 -1.7 -0.7 -1.7 -2.7 -2.7 -1.7 -1.7 -2.7 -2.7 -1.7 -2.7 -2.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
   0.2  0.2  0.2  1.2  0.2  0.2  0.2  1.2  1.2  1.2  1.2  2.2  1.2  1.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
   1.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  2.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
  -1.7 -1.7 -1.7 -0.7 -1.7 -2.7 -2.7 -1.7 -1.7 -2.7 -2.7 -1.7 -2.7 -2.7
  -2.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
   0.2  0.2  0.2  1.2  0.2  0.2  0.2  1.2  1.2  1.2  1.2  2.2  1.2  1.2
   1.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
   1.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  2.4
   2.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
  -1.7 -1.7 -1.7 -0.7 -1.7 -2.7 -2.7 -1.7 -1.7 -2.7 -2.7 -1.7 -2.7 -2.7
  -2.7 -2.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
   0.2  0.2  0.2  1.2  0.2  0.2  0.2  1.2  1.2  1.2  1.2  2.2  1.2  1.2
   1.2  1.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
   1.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  2.4
   2.4  2.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
  -1.7 -1.7 -1.7 -0.7 -1.7 -2.7 -2.7 -1.7 -1.7 -2.7 -2.7 -1.7 -2.7 -2.7
  -2.7 -2.7 -2.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
   0.2  0.2  0.2  1.2  0.2  0.2  0.2  1.2  1.2  1.2  1.2  2.2  1.2  1.2
   1.2  1.2  1.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
   1.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  2.4
   2.4  2.4  2.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
  -1.7 -1.7 -1.7 -0.7 -1.7 -2.7 -2.7 -1.7 -1.7 -2.7 -2.7 -1.7 -2.7 -2.7
  -2.7 -2.7 -2.7 -2.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
   0.2  0.2  0.2  1.2  0.2  0.2  0.2  1.2  1.2  1.2  1.2  2.2  1.2  1.2
   1.2  1.2  1.2  1.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
   1.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  2.4
   2.4  2.4  2.4  2.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
  -1.7 -1.7 -1.7 -0.7 -1.7 -2.7 -2.7 -1.7 -1.7 -2.7 -2.7 -1.7 -2.7 -2.7
  -2.7 -2.7 -2.7 -2.7 -2.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
   0.2  0.2  0.2  1.2  0.2  0.2  0.2  1.2  1.2  1.2  1.2  2.2  1.2  1.2
   1.2  1.2  1.2  1.2  1.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
   1.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  2.4
   2.4  2.4  2.4  2.4  2.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
  -1.7 -1.7 -1.7 -0.7 -1.7 -2.7 -2.7 -1.7 -1.7 -2.7 -2.7 -1.7 -2.7 -2.7
  -2.7 -2.7 -2.7 -2.7 -2.7 -2.7]
```

```
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
   0.2  0.2  0.2  1.2  0.2  0.2  0.2  1.2  1.2  1.2  1.2  2.2  1.2  1.2
   1.2  1.2  1.2  1.2  1.2  1.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
   1.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  2.4
   2.4  2.4  2.4  2.4  2.4  2.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
  -1.7 -1.7 -1.7 -0.7 -1.7 -2.7 -2.7 -1.7 -1.7 -2.7 -2.7 -1.7 -2.7 -2.7
  -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
   0.2  0.2  0.2  1.2  0.2  0.2  0.2  1.2  1.2  1.2  1.2  2.2  1.2  1.2
   1.2  1.2  1.2  1.2  1.2  1.2  1.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
   1.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  2.4
   2.4  2.4  2.4  2.4  2.4  2.4  2.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
  -1.7 -1.7 -1.7 -0.7 -1.7 -2.7 -2.7 -1.7 -1.7 -2.7 -2.7 -1.7 -2.7 -2.7
  -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
   0.2  0.2  0.2  1.2  0.2  0.2  0.2  1.2  1.2  1.2  1.2  2.2  1.2  1.2
   1.2  1.2  1.2  1.2  1.2  1.2  1.2  1.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
   1.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  2.4
   2.4  2.4  2.4  2.4  2.4  2.4  2.4  2.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
  -1.7 -1.7 -1.7 -0.7 -1.7 -2.7 -2.7 -1.7 -1.7 -2.7 -2.7 -1.7 -2.7 -2.7
  -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
   0.2  0.2  0.2  1.2  0.2  0.2  0.2  1.2  1.2  1.2  1.2  2.2  1.2  1.2
   1.2  1.2  1.2  1.2  1.2  1.2  1.2  1.2  1.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
   1.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  2.4
   2.4  2.4  2.4  2.4  2.4  2.4  2.4  2.4  2.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
  -1.7 -1.7 -1.7 -0.7 -1.7 -2.7 -2.7 -1.7 -1.7 -2.7 -2.7 -1.7 -2.7 -2.7
  -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
   0.2  0.2  0.2  1.2  0.2  0.2  0.2  1.2  1.2  1.2  1.2  2.2  1.2  1.2
   1.2  1.2  1.2  1.2  1.2  1.2  1.2  1.2  1.2  1.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
   1.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  2.4
   2.4  2.4  2.4  2.4  2.4  2.4  2.4  2.4  2.4  2.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
  -1.7 -1.7 -1.7 -0.7 -1.7 -2.7 -2.7 -1.7 -1.7 -2.7 -2.7 -1.7 -2.7 -2.7
  -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
   0.2  0.2  0.2  1.2  0.2  0.2  0.2  1.2  1.2  1.2  1.2  2.2  1.2  1.2
   1.2  1.2  1.2  1.2  1.2  1.2  1.2  1.2  1.2  1.2  1.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
   1.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  2.4
   2.4  2.4  2.4  2.4  2.4  2.4  2.4  2.4  2.4  2.4  2.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
  -1.7 -1.7 -1.7 -0.7 -1.7 -2.7 -2.7 -1.7 -1.7 -2.7 -2.7 -1.7 -2.7 -2.7
  -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7]
 [ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
   0.2  0.2  0.2  1.2  0.2  0.2  0.2  1.2  1.2  1.2  1.2  2.2  1.2  1.2
   1.2  1.2  1.2  1.2  1.2  1.2  1.2  1.2  1.2  1.2  1.2  1.2]
 [ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
   1.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  2.4
   2.4  2.4  2.4  2.4  2.4  2.4  2.4  2.4  2.4  2.4  2.4  2.4]]
[[ 0.3  0.3 -0.7 -0.7 -0.7  0.3 -0.7 -1.7 -1.7 -0.7 -0.7 -1.7 -1.7 -0.7
```

```
 -1.7 -1.7 -1.7 -0.7 -1.7 -2.7 -2.7 -1.7 -1.7 -2.7 -2.7 -1.7 -2.7 -2.7
 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7 -2.7]
[ 0.2  0.2 -0.8 -0.8 -0.8  0.2 -0.8 -0.8 -0.8  0.2  0.2  0.2  0.2  1.2
  0.2  0.2  0.2  1.2  0.2  0.2  0.2  1.2  1.2  1.2  1.2  2.2  1.2  1.2
  1.2  1.2  1.2  1.2  1.2  1.2  1.2  1.2  1.2  1.2  1.2  1.2  1.2]
[ 0.4  0.4  0.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4  1.4  0.4  0.4  1.4
  1.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  1.4  1.4  2.4  2.4  2.4
  2.4  2.4  2.4  2.4  2.4  2.4  2.4  2.4  2.4  2.4  2.4  2.4  2.4]]
```

In [10]:

```
print(weight[:,40])
```

```
[-2.7  1.2  2.4]
```

In [11]:

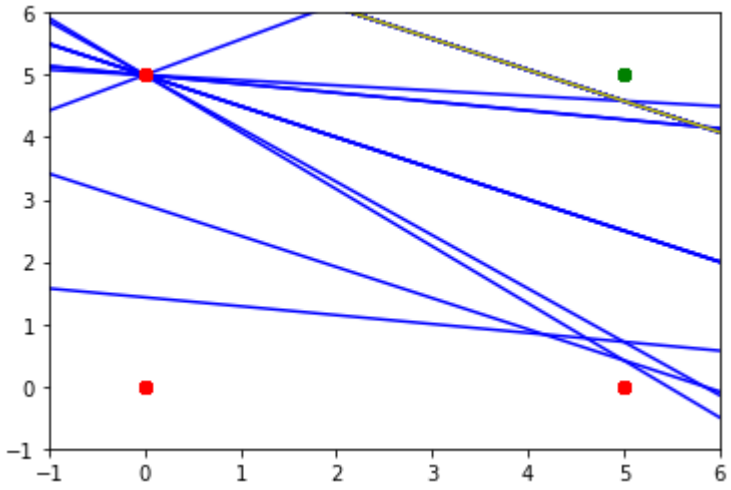
```
#testing

for i in range(4):
    nilai=perceptron.activationValue(data[i,:],weight[:,40])
    print(nilai)
```

```
1
0
0
0
```

In [12]:

```
#visualisation
perceptron.detFunction(weight,'b')
```

```
41
```



Exercise

build a model using perceptron of logical data XOR

X1	X2	Y
1	1	0
1	0	1
0	1	1
0	0	0

Referensi