



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

BAB : HEBB NET
NAMA : ARION SYEMAEL SIAHAAN
NIM : 225150207111060
TANGGAL : 11/09/2024
ASISTEN : ALIFAH KHAIRUNNISA
ANDHIKA IHSAN CENDEKIA



A. Praktikum

1. Buka Google Collaboratory melalui [tautan ini](#).
2. Tulis kode berikut ke dalam setiap *cell* pada *notebook* tersebut.

a. Fungsi Step Bipolar

```
def bipstep(y, th=0):  
    return 1 if y >= th else -1
```

b. Fungsi training Hebb

```
def hebb_fit(train, target, verbose=False, draw=False,  
draw_padding=1):  
    w = np.zeros(len(train[0]) + 1)  
    bias = np.ones((len(train), 1))  
    train = np.hstack((bias, train))  
    for r, row in enumerate(train):  
        w = [w[i] + row[i] * target[r] for i in range(len(row))]  
  
    if verbose:  
        print('Bobot:', w)  
    if draw:  
        plot(line(w, 0), train, target, draw_padding)  
  
    return w
```

c. Fungsi testing Hebb

```
def hebb_predict(X, w):  
    Y = []  
    for x in X:  
        y_in = w[0] + np.dot(x, w[1:])  
        y = bipstep(y_in)  
        Y.append(y)  
    return Y
```

d. Fungsi Hitung Akurasi

```
def calc_accuracy(a, b):  
    s = [1 if a[i] == b[i] else 0 for i in range(len(a))]  
    return sum(s) / len(a)
```

e. Logika AND

```
from sklearn.metrics import accuracy_score

train = (1, 1), (1, -1), (-1, 1), (-1, -1)
target = 1, -1, -1, -1
model = hebb_fit(train, target, verbose=False, draw=False)
output = hebb_predict(train, model)
accuracy = accuracy_score(output, target)

print('Output:', output)
print('Target:', target)
print('Accuracy:', accuracy)
```

f. Logika OR

```
train = (1, 1), (1, -1), (-1, 1), (-1, -1)
target = 1, 1, 1, -1
model = hebb_fit(train, target, verbose=True, draw=True)
output = hebb_predict(train, model)
accuracy = accuracy_score(output, target)

print('Output:', output)
print('Target:', target)
print('Accuracy:', accuracy)
```

g. Logika AND NOT

```
train = (1, 1), (1, -1), (-1, 1), (-1, -1)
target = -1, 1, -1, -1

model = hebb_fit(train, target, verbose=True, draw=True)
output = hebb_predict(train, model)
accuracy = accuracy_score(output, target)

print('Output:', output)
print('Target:', target)
print('Accuracy:', accuracy)
```

h. Logika XOR

```
train = (1, 1), (1, -1), (-1, 1), (-1, -1)
target = -1, 1, 1, -1
model = hebb_fit(train, target, verbose=True, draw=True)
output = hebb_predict(train, model)
accuracy = accuracy_score(output, target)

print('Output:', output)
print('Target:', target)
print('Accuracy:', accuracy)
```

B. Screenshot

a. Fungsi *Step* Bipolar

▼ a) Fungsi *step* Bipolar

Tulis kode ke dalam *cell* di bawah ini:

```
[5] def bipstep(y, th=0):  
    return 1 if y >= th else -1
```

ARION SYEMAEL SIAHAAN
225150207111060

b. Fungsi Training Hebb

▼ b) Fungsi *training* Hebb

Tulis kode ke dalam *cell* di bawah ini:

```
[7] def hebb_fit(train, target, verbose=False, draw=False,  
draw_padding=1):  
    w = np.zeros(len(train[0]) + 1)  
    bias = np.ones((len(train), 1))  
    train = np.hstack((bias, train))  
    for r, row in enumerate(train):  
        w = [w[i] + row[i] * target[r] for i in range(len(row))]  
  
    if verbose:  
        print('Bobot:', w)  
    if draw:  
        plot(line(w, 0), train, target, draw_padding)  
  
    return w
```

ARION SYEMAEL SIAHAAN
225150207111060

c. Fungsi testing Hebb

c) Fungsi *testing* Hebb

Tulis kode ke dalam *cell* di bawah ini:

```
[8] def hebb_predict(X, w):  
    Y = []  
    for x in X:  
        y_in = w[0] + np.dot(x, w[1:])  
        y = bipstep(y_in)  
        Y.append(y)  
    return Y
```

ARION SYEMAEL SIAHAAN
225150207111060

d. Fungsi Hitung Akurasi

d) Fungsi Hitung Akurasi

```
[9] def calc_accuracy(a, b):  
    s = [1 if a[i] == b[i] else 0 for i in range(len(a))]  
    return sum(s) / len(a)
```

ARION SYEMAEL SIAHAAN
225150207111060

e. Logika AND

e) Logika AND

Tulis kode ke dalam *cell* di bawah ini:

```
from sklearn.metrics import accuracy_score  
  
train = (1, 1), (1, -1), (-1, 1), (-1, -1)  
target = 1, -1, -1, -1  
model = hebb_fit(train, target, verbose=False, draw=False)  
output = hebb_predict(train, model)  
accuracy = accuracy_score(output, target)  
  
print('Output:', output)  
print('Target:', target)  
print('Accuracy:', accuracy)
```

```
Output: [1, -1, -1, -1]  
Target: (1, -1, -1, -1)  
Accuracy: 1.0
```

ARION SYEMAEL SIAHAAN
225150207111060

f. Logika OR

▼ f) Logika OR

Tulis kode ke dalam *cell* di bawah ini:

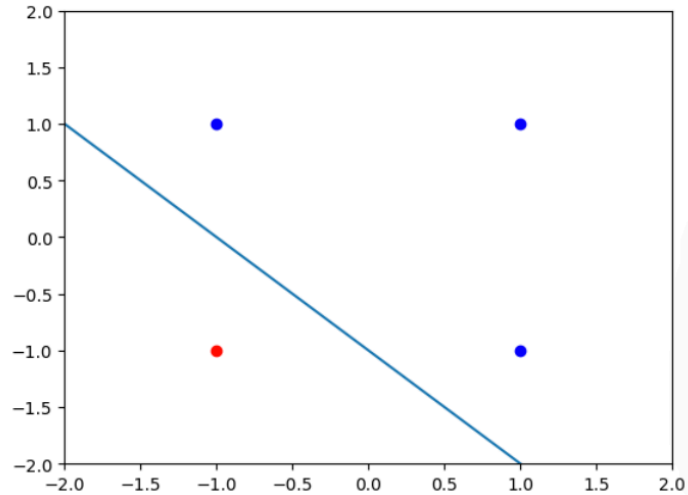
```
train = (1, 1), (1, -1), (-1, 1), (-1, -1)
target = 1, 1, 1, -1
model = hebb_fit(train, target, verbose=True, draw=True)
output = hebb_predict(train, model)
accuracy = accuracy_score(output, target)

print('Output:', output)
print('Target:', target)
print('Accuracy:', accuracy)
```

↻ Bobot: [1.0, 1.0, 1.0]

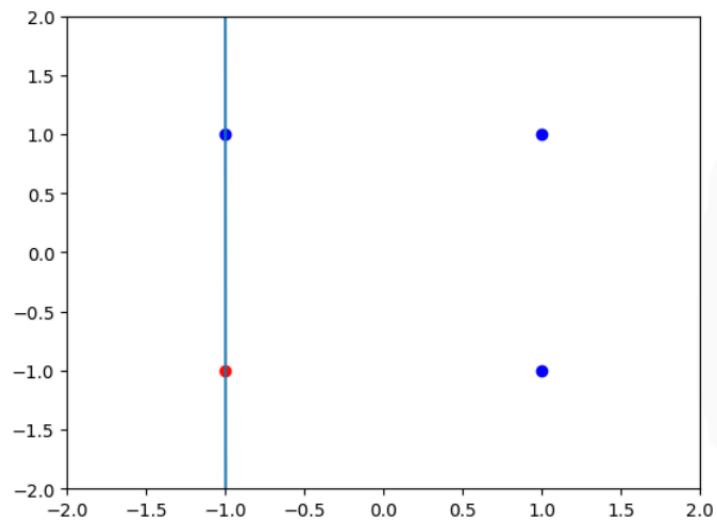
ARION SYEMAEL SIAHAAN
225150207111060

Bobot: [1.0, 1.0, 1.0]

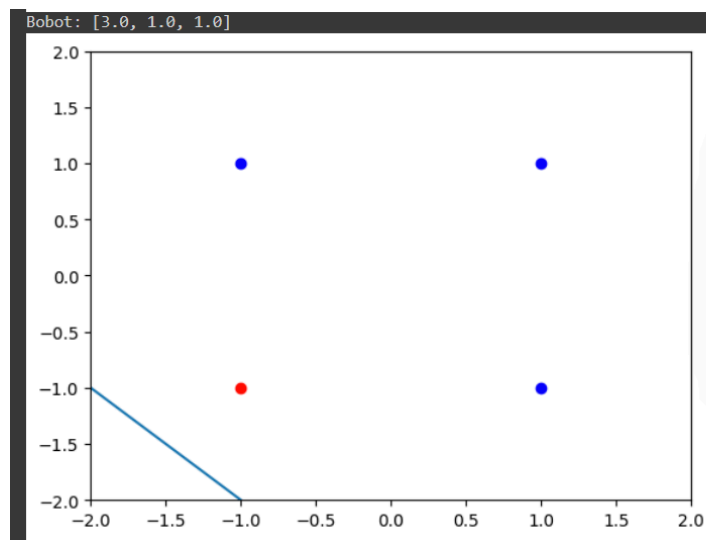


ARION SYEMAEL SIAHAAN
225150207111060

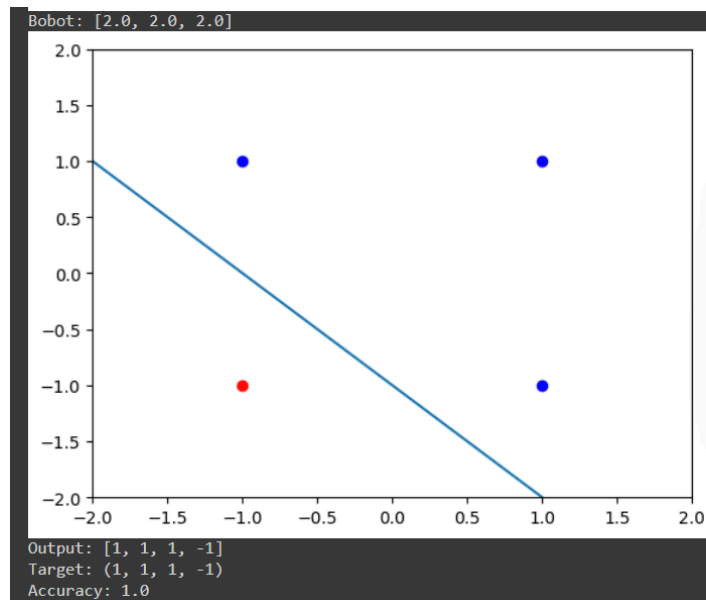
Bobot: [2.0, 2.0, 0.0]



ARION SYEMAEL SIAHAAN
225150207111060



ARION SYEMAEL SIAHAAN
225150207111060



ARION SYEMAEL SIAHAAN
225150207111060

g. Logika AND NOT

g) Logika AND NOT

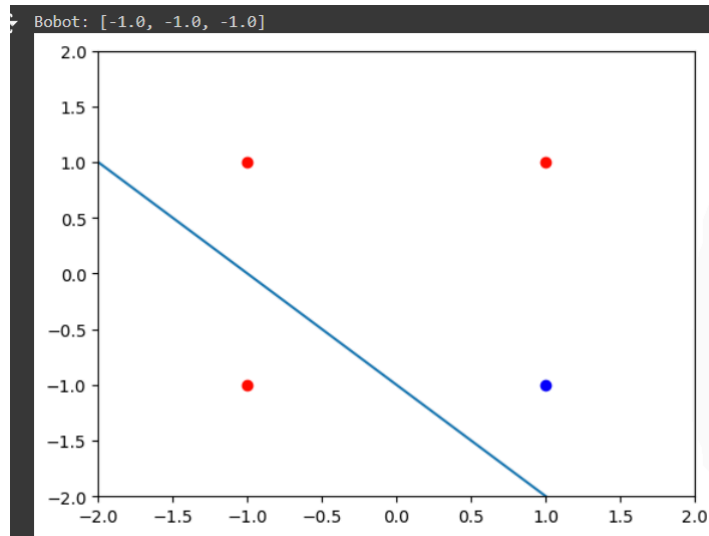
Tulis kode ke dalam *cell* di bawah ini:

```
train = (1, 1), (1, -1), (-1, 1), (-1, -1)
target = -1, 1, -1, -1

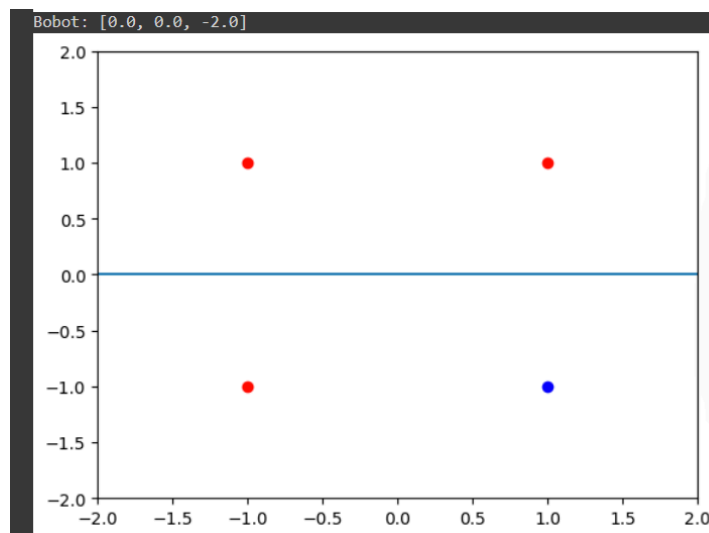
model = hebb_fit(train, target, verbose=True, draw=True)
output = hebb_predict(train, model)
accuracy = accuracy_score(output, target)

print('Output:', output)
print('Target:', target)
print('Accuracy:', accuracy)
```

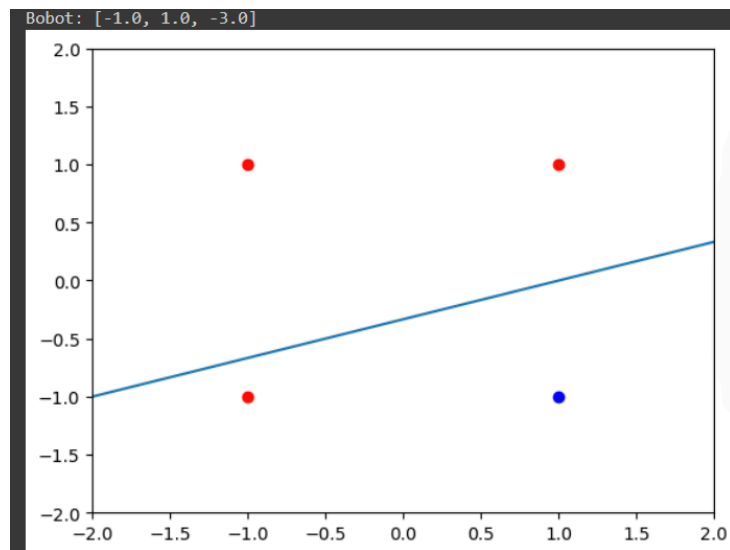
ARION SYEMAEL SIAHAAN
225150207111060



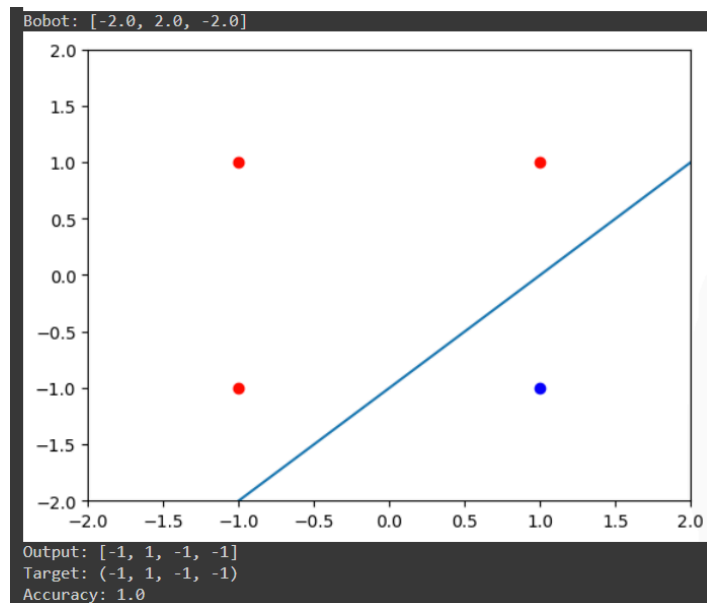
ARION SYEMAEI SIAHAAN
225150207111060



ARION SYEMAEI SIAHAAN
225150207111060



ARION SYEMAEL SIAHAAN
225150207111060



ARION SYEMAEL SIAHAAN
225150207111060

h. Logika XOR

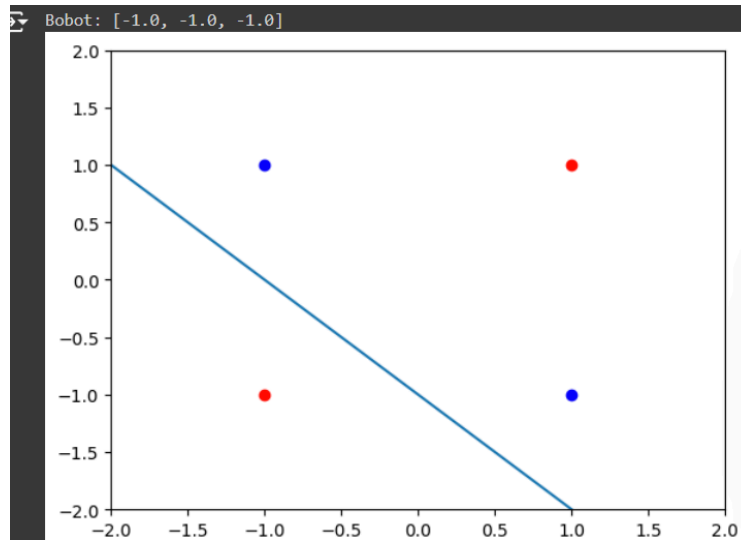
h) Logika XOR

Tulis kode ke dalam *cell* di bawah ini:

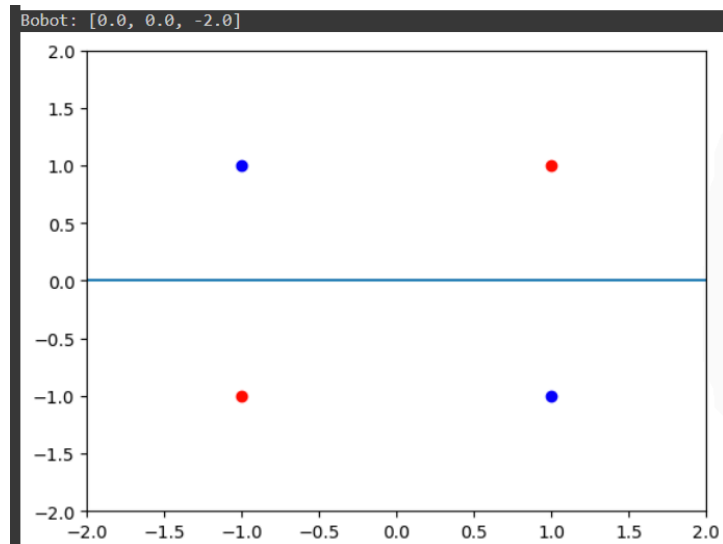
```
train = (1, 1), (1, -1), (-1, 1), (-1, -1)
target = -1, 1, 1, -1
model = hebb_fit(train, target, verbose=True, draw=True)
output = hebb_predict(train, model)
accuracy = accuracy_score(output, target)

print('Output:', output)
print('Target:', target)
print('Accuracy:', accuracy)
```

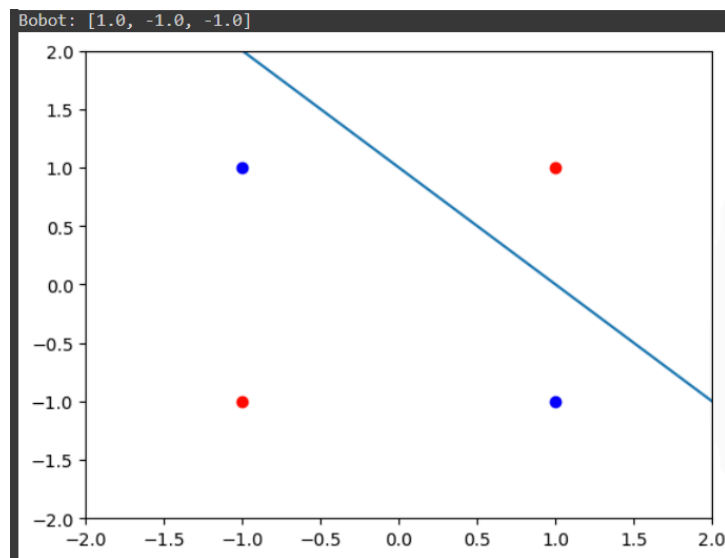
ARION SYEMAEL SIAHAAN
225150207111060



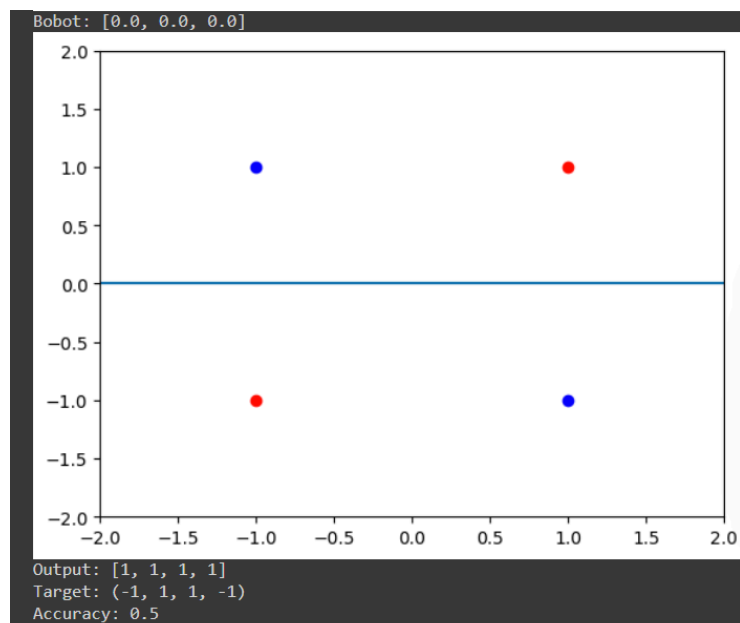
ARION SYEMAEL SIAHAAN
225150207111060



ARION SYEMAEL SIAHAAN
225150207111060



ARION SYEMAEL SIAHAAN
225150207111060



ARION SYEMAEL SIAHAAN
225150207111060

C. Analisis

1. Pada klasifikasi menggunakan logika XOR, mengapa akurasi yang didapatkan tidak mencapai 1 (100%)?

Akurasi pada logika XOR tidak bisa mencapai 100% karena masalah XOR tidak dapat dipisahkan secara linear, maksudnya kita gak bisa menarik garis lurus yang memisahkan kelas-kelasnya secara sempurna. XOR punya pola di mana dua input yang mirip bisa menghasilkan output yang berbeda (contoh: (1,1) menghasilkan -1 dan (1,-1) menghasilkan 1).

Model sederhana seperti perceptron hanya bisa memisahkan data dengan garis lurus, sehingga tidak bisa menangani pola XOR yang lebih kompleks. Untuk memecahkan masalah XOR dengan akurasi 100%, diperlukan model yang lebih canggih, seperti jaringan saraf multilayer yang bisa membuat pemisahan non-linear.

2. Lakukan proses training dan testing menggunakan data berikut.

Training: (-1, .5), (.5, .3), (1, 1.5), (3, 1.9)

Target: (-1, -1, 1, 1)

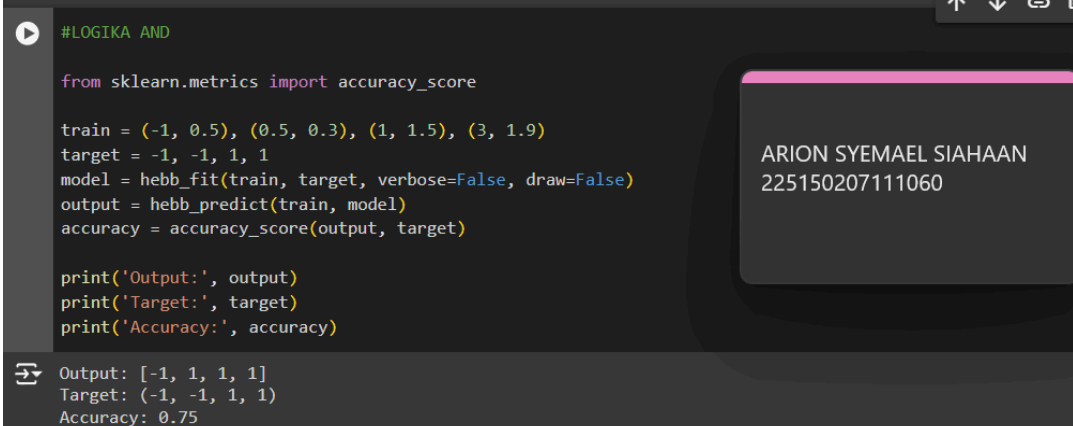
Berapakah akurasi yang didapatkan? Mengapa tidak dapat mencapai akurasi 1 (100%)?

```
#LOGIKA AND

from sklearn.metrics import accuracy_score

train = (-1, 0.5), (0.5, 0.3), (1, 1.5), (3, 1.9)
target = -1, -1, 1, 1
model = hebb_fit(train, target, verbose=False, draw=False)
output = hebb_predict(train, model)
accuracy = accuracy_score(output, target)

print('Output:', output)
print('Target:', target)
print('Accuracy:', accuracy)
```



```
#LOGIKA AND

from sklearn.metrics import accuracy_score

train = (-1, 0.5), (0.5, 0.3), (1, 1.5), (3, 1.9)
target = -1, -1, 1, 1
model = hebb_fit(train, target, verbose=False, draw=False)
output = hebb_predict(train, model)
accuracy = accuracy_score(output, target)

print('Output:', output)
print('Target:', target)
print('Accuracy:', accuracy)
```

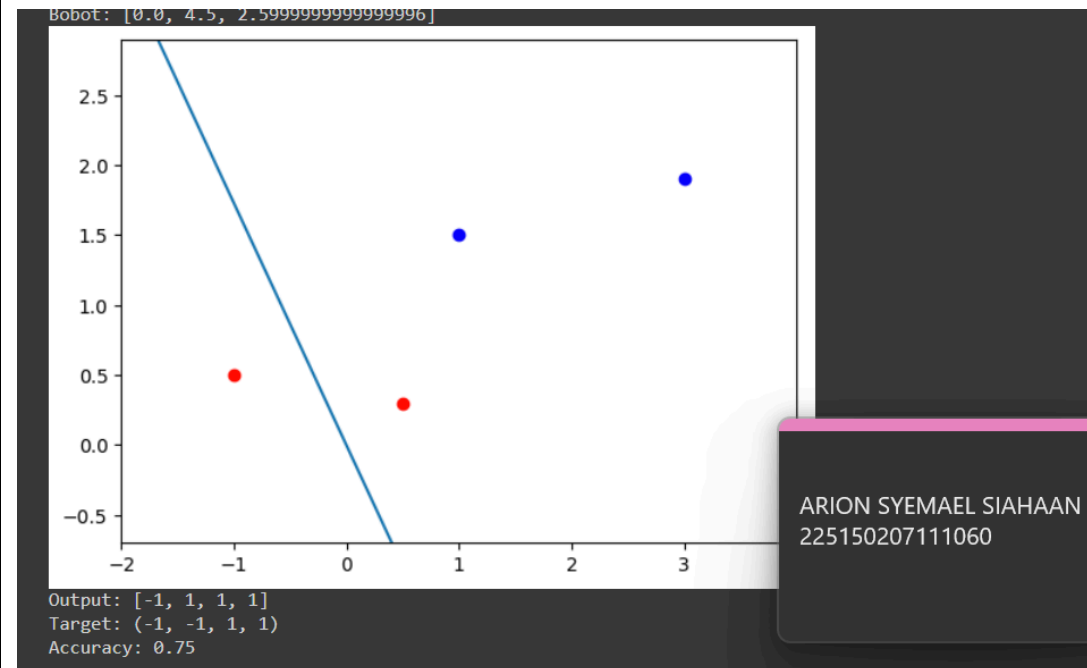
Output: [-1, 1, 1, 1]
Target: (-1, -1, 1, 1)
Accuracy: 0.75

ARION SYMAEL SIAHAAN
225150207111060

```
#LOGIKA OR
```

```
train = (-1, 0.5), (0.5, 0.3), (1, 1.5), (3, 1.9)
target = -1, -1, 1, 1
model = hebb_fit(train, target, verbose=True, draw=True)
output = hebb_predict(train, model)
accuracy = accuracy_score(output, target)
```

```
print('Output:', output)
print('Target:', target)
print('Accuracy:', accuracy)
```

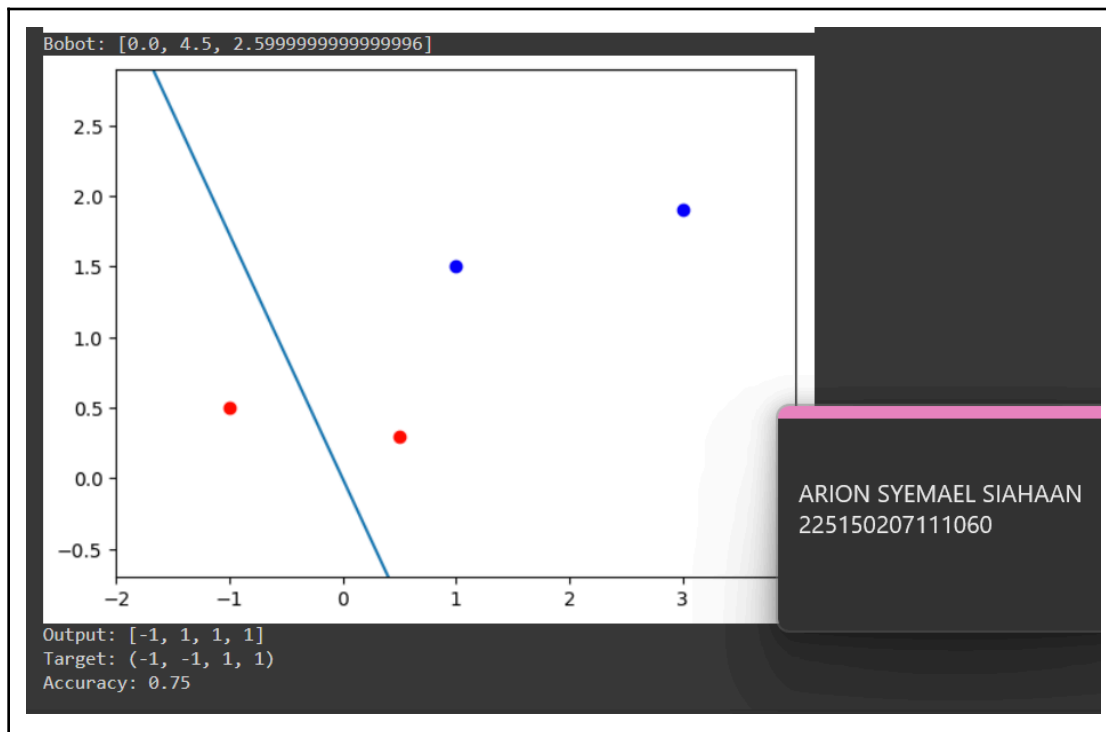


```
#LOGIKA AND NOT
```

```
train = (-1, 0.5), (0.5, 0.3), (1, 1.5), (3, 1.9)
target = -1, -1, 1, 1
```

```
model = hebb_fit(train, target, verbose=True, draw=True)
output = hebb_predict(train, model)
accuracy = accuracy_score(output, target)
```

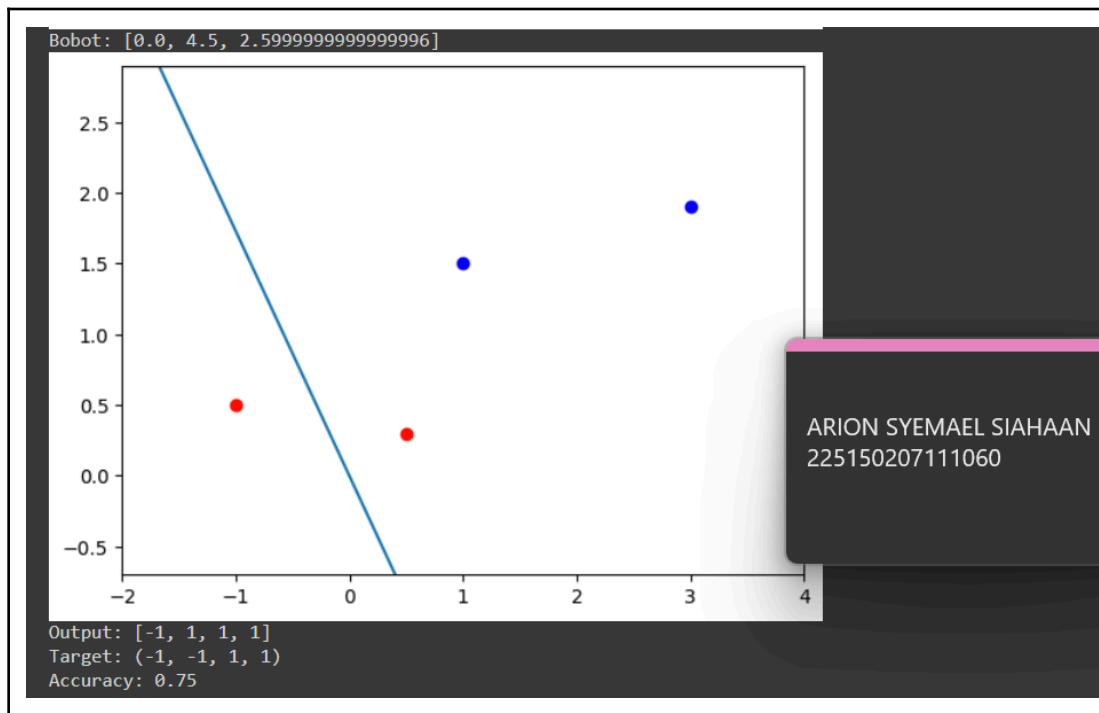
```
print('Output:', output)
print('Target:', target)
print('Accuracy:', accuracy)
```



```
#LOGIKA XOR
```

```
train = (-1, 0.5), (0.5, 0.3), (1, 1.5), (3, 1.9)
target = -1, -1, 1, 1
model = hebb_fit(train, target, verbose=True, draw=True)
output = hebb_predict(train, model)
accuracy = accuracy_score(output, target)

print('Output:', output)
print('Target:', target)
print('Accuracy:', accuracy)
```



Akurasi yang tidak mencapai 100% pada logika AND, AND NOT, OR, dan XOR disebabkan oleh beberapa faktor yang bergantung pada kompleksitas setiap fungsi logika dan bagaimana data dipisahkan.

- Logika AND: Memerlukan semua input untuk bernilai true (1) agar hasilnya true. Jika bobot atau bias model tidak tepat, data yang seharusnya diklasifikasikan sebagai true bisa salah klasifikasi sebagai false.
- Logika AND NOT: Ini lebih kompleks karena melibatkan kombinasi input yang harus bernilai true dan yang harus bernilai false secara bersamaan. Jika pemisahan data tidak presisi, hasil prediksi bisa meleset.
- Logika OR: Meskipun lebih mudah dari AND, model sederhana bisa saja salah memprediksi karena OR membutuhkan setidaknya satu input true untuk menghasilkan true. Jika bobot tidak optimal, ada kemungkinan nilai yang seharusnya true diklasifikasikan sebagai false.
- Logika XOR: Ini yang paling sulit di antara yang lain karena masalah XOR tidak dapat diselesaikan oleh pemisahan linear, yang membuat model seperti Hebb Net atau perceptron tidak mampu memisahkan data dengan benar. Oleh karena itu, akurasi tidak bisa mencapai 100%.

Jadi, secara keseluruhan, untuk mencapai akurasi 100%, diperlukan model yang lebih kompleks dengan kemampuan memisahkan data secara non-linear.

D. Kesimpulan

1. Hebb Net adalah jaringan saraf tiruan yang menggunakan aturan pembelajaran Hebb (Hebb's Rule), yang didasarkan pada prinsip bahwa jika dua neuron sering aktif secara bersamaan, koneksi antara mereka akan diperkuat. Hebb Net belajar dengan memperbarui bobot setiap kali input dan output cocok sesuai dengan pola target.
Perbedaan utama antara Hebb Net dan McCulloch-Pitts Neuron adalah pada algoritma pembelajarannya. McCulloch-Pitts Neuron adalah model yang lebih sederhana dan tidak belajar dari data—semua bobotnya diatur secara manual. Kelebihan Hebb Net adalah kemampuannya belajar dari data, sementara McCulloch-Pitts Neuron lebih cepat tetapi tidak bisa menyesuaikan diri dengan data baru. Kekurangan Hebb Net adalah kesederhanaan algoritmanya yang menyebabkan sulit menangani masalah yang kompleks dan tidak linier, seperti XOR.
2. Bias adalah nilai tambahan yang digunakan dalam jaringan saraf untuk menyesuaikan hasil perhitungan aktivasi neuron. Bias berfungsi sebagai nilai penyesuaian agar jaringan saraf dapat menggeser decision boundary (garis keputusan) sehingga model lebih fleksibel dalam memisahkan data. Tanpa bias, semua garis keputusan akan selalu melewati titik asal (0,0), yang membatasi kemampuan model untuk mengklasifikasikan data dengan baik.
3. Epoch dalam machine learning merujuk pada satu kali pemrosesan seluruh dataset selama pelatihan model. Satu epoch berarti semua contoh data latih telah digunakan untuk memperbarui bobot sekali. Biasanya, pelatihan model dilakukan melalui beberapa epoch agar model dapat terus memperbaiki performanya berdasarkan error yang dihitung setelah setiap epoch. Semakin banyak epoch, semakin baik model dapat belajar dari data, tetapi terlalu banyak epoch juga bisa menyebabkan model overfitting, di mana model belajar terlalu detail pada data latih dan kurang generalisasi pada data baru.