LABORATORIUM PEMBELAJARAN ILMU KOMPUTER

FAKULTAS ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

BAB : PERCEPTRON
NAMA : ARION SYEMAEL SIAHAAN
NIM : 225150207111060
TANGGAL : 18/09/2024
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* Perceptron

```
def percep_step(input, th=0):
  return 1 if input > th else -1 if input < -th else 0
```

b. Fungsi training Perceptron

```
def percep_fit(X, target, th=0, a=1, max_epoch=-1, verbose=False,
draw=False):
    w = np.zeros(len(X[0]) + 1)
    bias = np.ones((len(X), 1))
    X = np.hstack((bias, X))
    stop = False
    epoch = 0

    while not stop and (max_epoch == -1 or epoch < max_epoch):
        stop = True
        epoch += 1

        if verbose:
            print('\nEpoch', epoch)

        for r, row in enumerate(X):
            y_in = np.dot(row, w)
            y = percep_step(y_in, th)

            if y != target[r]:
                stop = False
                      w = [w[i] + a * target[r] * row[i] for i in
range(len(row))]

                if verbose:
                    print('Bobot:', w)

        if draw:
            plot(line(w, th), line(w, -th), X, target)

    return w, epoch
```

c. Fungsi testing Perceptron

```
def percep_predict(X, w, th=0):
    Y = []
    for x in X:
        y_in = w[0] + np.dot(x, w[1:])
        y = percep_step(y_in, th)
        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

```
train = (1, 1), (1, -1), (-1, 1), (-1, -1)
target = 1, -1, -1, -1
th = .2
model, epoch = percep_fit(train, target, th, verbose=True,
draw=True)
output = percep_predict(train, model)
accuracy = calc_accuracy(output, target)

print('Epochs:', epoch)
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
th = .2
model, epoch = percep_fit(train, target, th, verbose=True,
draw=True)
output = percep_predict(train, model)
accuracy = calc_accuracy(output, target)

print('Epochs:', epoch)
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
th = .2
```

```
model, epoch = percep_fit(train, target, th, verbose=True,
draw=True)
output = percep_predict(train, model)
accuracy = calc_accuracy(output, target)

print('Epochs:', epoch)
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
th = .2
model, epoch = percep_fit(train, target, th, max_epoch=50,
verbose=True, draw=False)
output = percep_predict(train, model)
accuracy = calc_accuracy(output, target)

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

B. Screenshot

a. Fungsi *Step* Perceptron

b. Fungsi Training Perceptron

b) Fungsi *training* Perceptron

Tulis kode ke dalam *cell* di bawah ini:

```
def percep_fit(X, target, th=0, a=1, max_epoch=-1, verbose=False, draw=False):
    w = np.zeros(len(X[0]) + 1)
    bias = np.ones((len(X), 1))
    X = np.hstack((bias, X))
    stop = False
    epoch = 0

    while not stop and (max_epoch == -1 or epoch < max_epoch):
        stop = True
        epoch += 1

        if verbose:
            print('\nEpoch', epoch)

        for r, row in enumerate(X):
            y_in = np.dot(row, w)
            y = percep_step(y_in, th)

            if y != target[r]:
                stop = False
                w = [w[i] + a * target[r] * row[i] for i in range(len(row))]

                if verbose:
                    print('Bobot:', w)

        if draw:
            plot(line(w, th), line(w, -th), X, target)

    return w, epoch
```

ARION SYEMAEL SIAHAAN
225150207111060

c. Fungsi testing Perceptron

c) Fungsi *testing* Perceptron

Tulis kode ke dalam *cell* di bawah ini:

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

ARION SYEMAEL SIAHAAN
225150207111060

d. Fungsi Hitung Akurasi

d) Fungsi Hitung Akurasi

Tulis kode ke dalam *cell* di bawah ini:

```
[15] 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

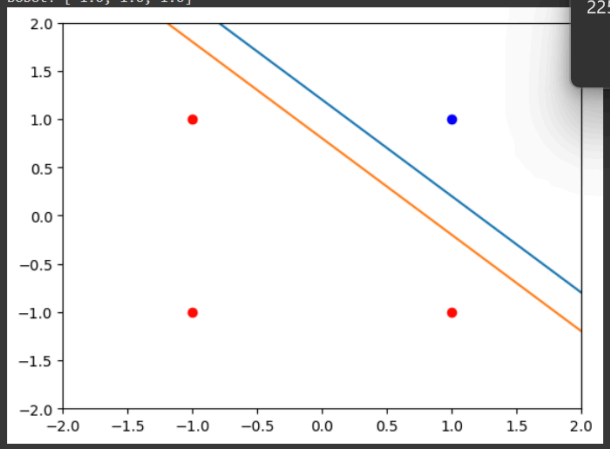
e. Logika AND

e) Logika AND

Tulis kode ke dalam *cell* di bawah ini:

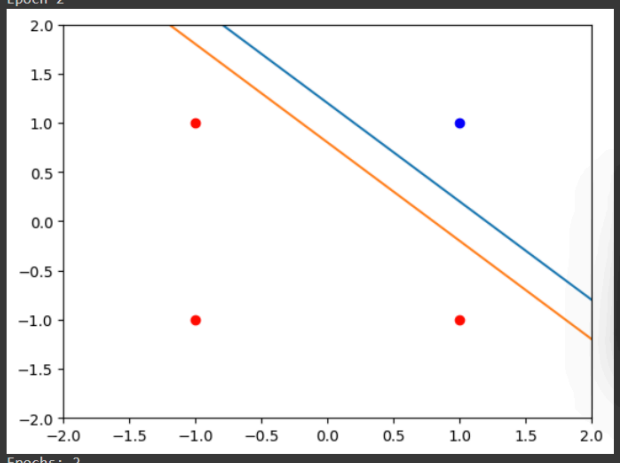
```
train = (1, 1), (1, -1), (-1, 1), (-1, -1)
target = 1, -1, -1, -1
th = .2
model, epoch = percep_fit(train, target, th, verbose=True,
draw=True)
output = percep_predict(train, model)
accuracy = calc_accuracy(output, target)

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

```
Bobot: [1.0, 1.0, 1.0]
Bobot: [0.0, 0.0, 2.0]
Bobot: [-1.0, 1.0, 1.0]
```



ARION SYEMAEL SIAHAAN
225150207111060

Epoch 2




```
Epochs: 2
Output: [1, -1, -1, -1]
Target: (1, -1, -1, -1)
Accuracy: 1.0
```

ARION SYEMAEL SIAHAAN
225150207111060

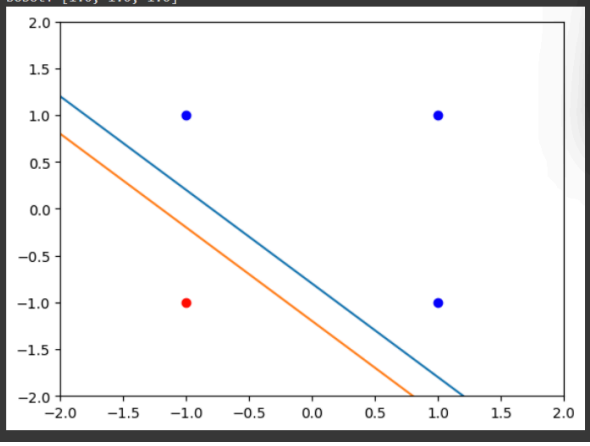
f. Logika OR

f) Logika OR

Tulis kode ke dalam *cell* di bawah ini:

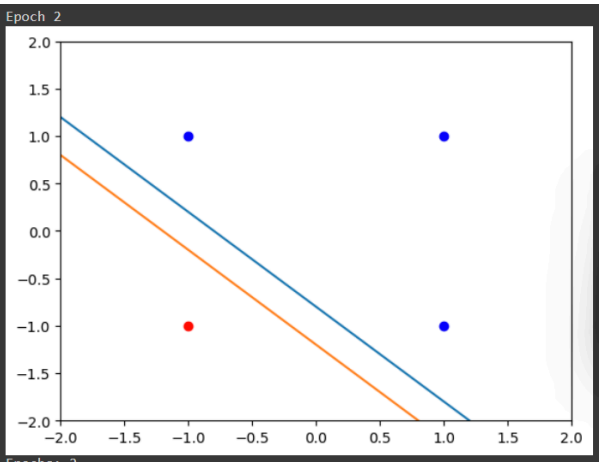
```
train = (1, 1), (1, -1), (-1, 1), (-1, -1)
target = 1, 1, 1, -1
th = .2
model, epoch = percep_fit(train, target, th, verbose=True,
draw=True)
output = percep_predict(train, model)
accuracy = calc_accuracy(output, target)

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

```
Epoch 1
Bobot: [1.0, 1.0, 1.0]
```



ARION SYEMAEL SIAHAAN
225150207111060

Epoch 2



ARION SYEMAEL SIAHAAN
225150207111060

```
Epochs: 2
Output: [1, 1, 1, -1]
Target: (1, 1, 1, -1)
Accuracy: 1.0
```

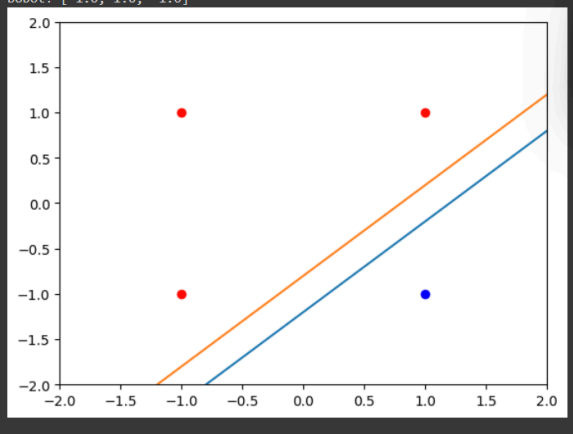
g. Logika AND NOT

g) Logika AND NOT

Tulis kode ke dalam *cell* di bawah ini:

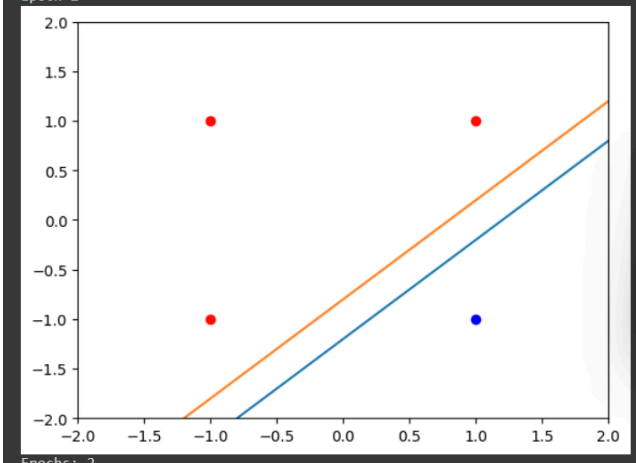
```
[8] train = (1, 1), (1, -1), (-1, 1), (-1, -1)
    target = -1, 1, -1, -1
    th = .2
    model, epoch = percep_fit(train, target, th, verbose=True,
    draw=True)
    output = percep_predict(train, model)
    accuracy = calc_accuracy(output, target)

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

```
Epoch 1
Bobot: [-1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, -2.0]
Bobot: [-1.0, 1.0, -1.0]
```



ARION SYEMAEL SIAHAAN
225150207111060

Epoch 2



ARION SYEMAEL SIAHAAN
225150207111060

```
Epochs: 2
Output: [-1, 1, -1, -1]
Target: (-1, 1, -1, -1)
Accuracy: 1.0
```

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
th = .2
model, epoch = percep_fit(train, target, th, max_epoch=50,
verbose=True, draw=False)
output = percep_predict(train, model)
accuracy = calc_accuracy(output, target)

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

```
Epoch 1
Bobot: [-1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, -2.0]
Bobot: [1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, 0.0]

Epoch 2
Bobot: [-1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, -2.0]
Bobot: [1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, 0.0]

Epoch 3
Bobot: [-1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, -2.0]
Bobot: [1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, 0.0]

Epoch 4
Bobot: [-1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, -2.0]
Bobot: [1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, 0.0]

Epoch 5
Bobot: [-1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, -2.0]
Bobot: [1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, 0.0]
```

ARION SYEMAEL SIAHAAN
225150207111060

```
Epoch 45
Bobot: [-1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, -2.0]
Bobot: [1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, 0.0]

Epoch 46
Bobot: [-1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, -2.0]
Bobot: [1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, 0.0]

Epoch 47
Bobot: [-1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, -2.0]
Bobot: [1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, 0.0]

Epoch 48
Bobot: [-1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, -2.0]
Bobot: [1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, 0.0]

Epoch 49
Bobot: [-1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, -2.0]
Bobot: [1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, 0.0]

Epoch 50
Bobot: [-1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, -2.0]
Bobot: [1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, 0.0]
Output: [0, 0, 0, 0]
Accuracy: 0.0
```

ARION SYEMAEL SIAHAAN
225150207111060

C. Analisis

1. Mengapa Perceptron gagal dalam melakukan proses training menggunakan data logika XOR? Jelaskan.

Perceptron hanya bisa menyelesaikan masalah yang bisa dipisahkan dengan garis lurus (linear). Data XOR adalah contoh masalah yang tidak bisa dipisahkan dengan garis lurus karena titik-titiknya tidak bisa dikelompokkan dengan satu garis lurus. Jadi, Perceptron gagal memisahkan data ini dengan benar. Untuk mengatasi masalah seperti XOR, dibutuhkan model yang lebih kompleks, seperti jaringan saraf dengan lebih dari satu lapisan (multilayer perceptron).

2. Lakukan pelatihan data logika AND dengan learning rate yang berbeda-beda. Amati jumlah epoch yang dilakukan. Bagaimanakah efeknya pada proses pelatihan?

Learning rate menentukan seberapa besar perubahan bobot pada setiap langkah pelatihan. Jika learning rate kecil, perubahan bobot menjadi lambat, sehingga butuh banyak epoch (iterasi) untuk menyelesaikan pelatihan. Jika learning rate terlalu besar, model akan bergerak terlalu cepat, dan bisa melompati solusi yang benar, menyebabkan pelatihan tidak stabil.

D. Kesimpulan

1. Perceptron dan Hebb Net adalah dua jenis model jaringan saraf yang memiliki pendekatan berbeda dalam pembelajaran. Perceptron bekerja dengan memperbarui bobot berdasarkan kesalahan prediksi, mengikuti prinsip **error correction**. Jika prediksi tidak sesuai dengan hasil yang diharapkan, bobot akan disesuaikan untuk memperbaiki kesalahan tersebut. Sebaliknya, Hebb Net menggunakan aturan **Hebbian learning**, yang menyatakan bahwa koneksi antara neuron akan diperkuat jika mereka sering aktif bersamaan, tanpa memperhitungkan kesalahan. Dengan demikian, Hebb Net memperkuat koneksi berdasarkan keterhubungan aktivitas neuron, sementara Perceptron berfokus pada memperbaiki kesalahan klasifikasi.

2. Learning rate dalam pembelajaran mesin sangat penting karena mengontrol seberapa besar perubahan bobot dilakukan dalam setiap iterasi. Jika learning rate terlalu besar, model mungkin kesulitan mencapai konvergensi karena

loncatan perubahan yang terlalu besar, menyebabkan model melewatkan solusi optimal.
Sebaliknya, jika learning rate terlalu kecil, proses pembelajaran akan berjalan lambat dan bisa terjebak pada solusi sub-optimal. Oleh karena itu, learning rate yang tepat membantu model untuk belajar secara efisien dan stabil menuju hasil yang diinginkan.

3. Perceptron memiliki kemampuan untuk menyelesaikan masalah yang **linier separable**, yaitu masalah di mana data dari dua kelas dapat dipisahkan oleh garis lurus atau hyperplane. Contohnya, Perceptron bisa menangani masalah logika sederhana seperti **AND gate** dan **OR gate**.
   Namun, Perceptron tidak mampu menyelesaikan masalah yang **tidak linier separable**, seperti **XOR gate**, di mana dua kelas tidak bisa dipisahkan hanya dengan satu garis lurus. Untuk kasus-kasus semacam itu, diperlukan model yang lebih kompleks seperti Multi-Layer Perceptron (MLP), yang melibatkan lapisan tersembunyi untuk menangani pola yang lebih rumit.