



**LABORATORIUM PEMBELAJARAN ILMU KOMPUTER**  
**FAKULTAS ILMU KOMPUTER**  
**UNIVERSITAS BRAWIJAYA**

---

**BAB : PERCEPTRON**

**NAMA : RAIKHAN GEZA ALBURAMA**  
**NIM : 225150207111040**  
**TANGGAL : 17/09/2024**  
**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('Epochs:', epoch)
print('Output:', output)
print('Accuracy:', accuracy)
```

### i. Load dan plot data

```
import seaborn as sns
import matplotlib.pyplot as plt

iris = sns.load_dataset('iris')

sns.pairplot(iris, hue='species')
plt.show()
```

#### j. Menghapus Kelas Virginica

```
iris = iris.loc[iris['species'] != 'virginica']

sns.pairplot(iris, hue='species')
plt.show()
```

#### k. Menghapus ciri sepal width dan petal width

```
iris = iris.drop(['sepal_width', 'petal_width'], axis=1)

sns.pairplot(iris, hue='species')
plt.show()
```

#### l. Proses Training dan Testing

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import minmax_scale

X = iris[['sepal_length', 'petal_length']].to_numpy()
X = minmax_scale(X)

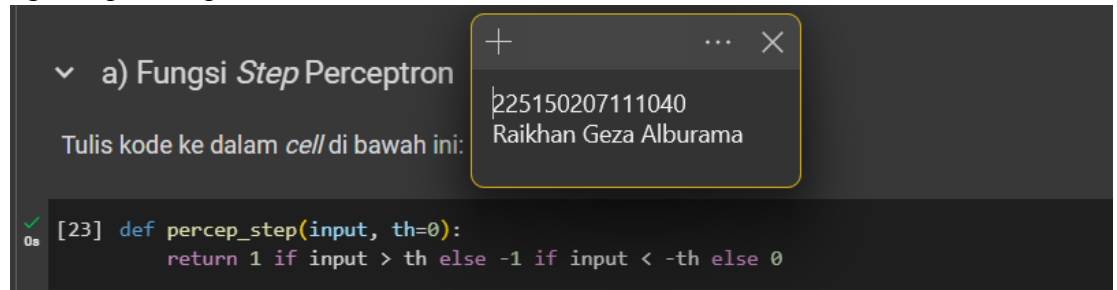
y = iris['species'].to_numpy()
c = {'setosa': -1, 'versicolor': 1}
y = [c[i] for i in y]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=.3)
# Remove the marker argument.
w, epoch = percep_fit(X_train, y_train, verbose=True, draw=True)
out = percep_predict(X_test, w)
accuracy = calc_accuracy(out, y_test)

print('Epochs:', epoch)
print('Accuracy:', accuracy)
```

## B. Screenshot

### a. Fungsi *Step* Perceptron

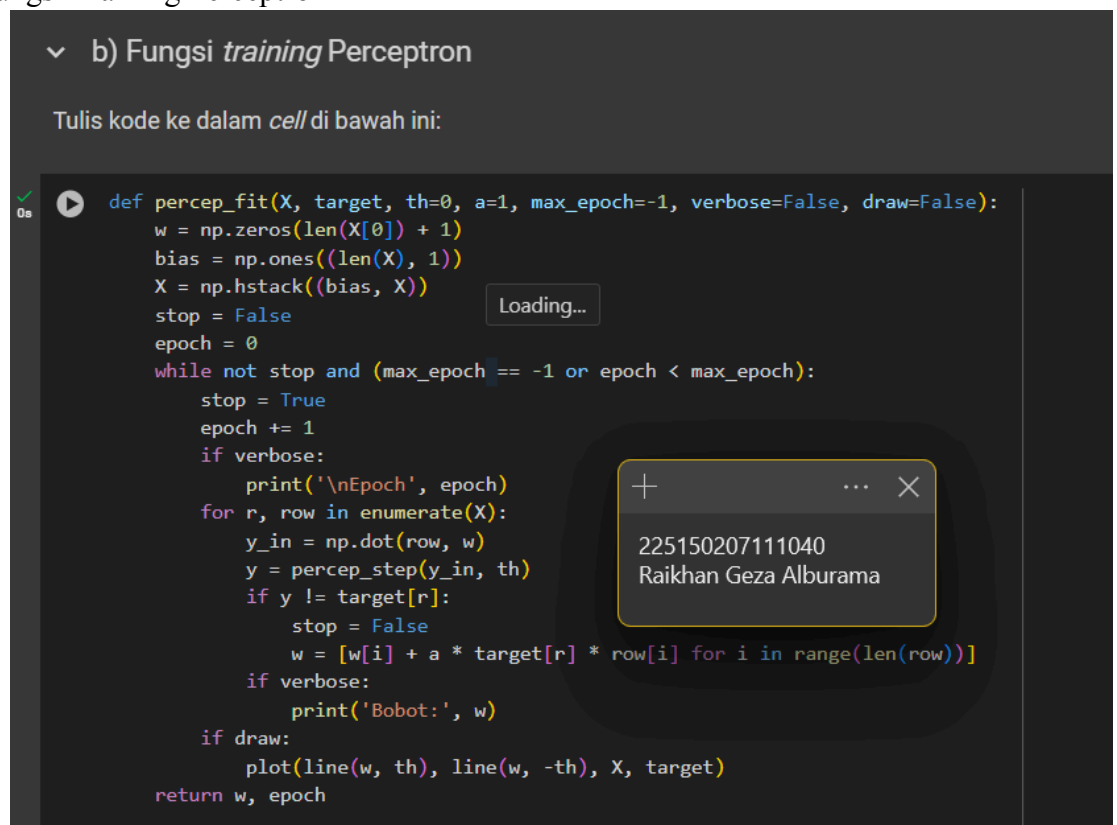


▼ a) Fungsi *Step* Perceptron

Tulis kode ke dalam *cell* di bawah ini:

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

### 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
```

c. Fungsi Testing Perceptron

▼ c) Fungsi *testing* Perceptron

Tulis kode ke dalam *cell* di bawah ini:

```
[25] 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
```

225150207111040  
Raikhan Geza Alburama

d. Fungsi Hitung Akurasi

Fungsi Hitung Akurasi

```
[26] 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)
```

225150207111040  
Raikhan Geza Alburama

e. Logika AND

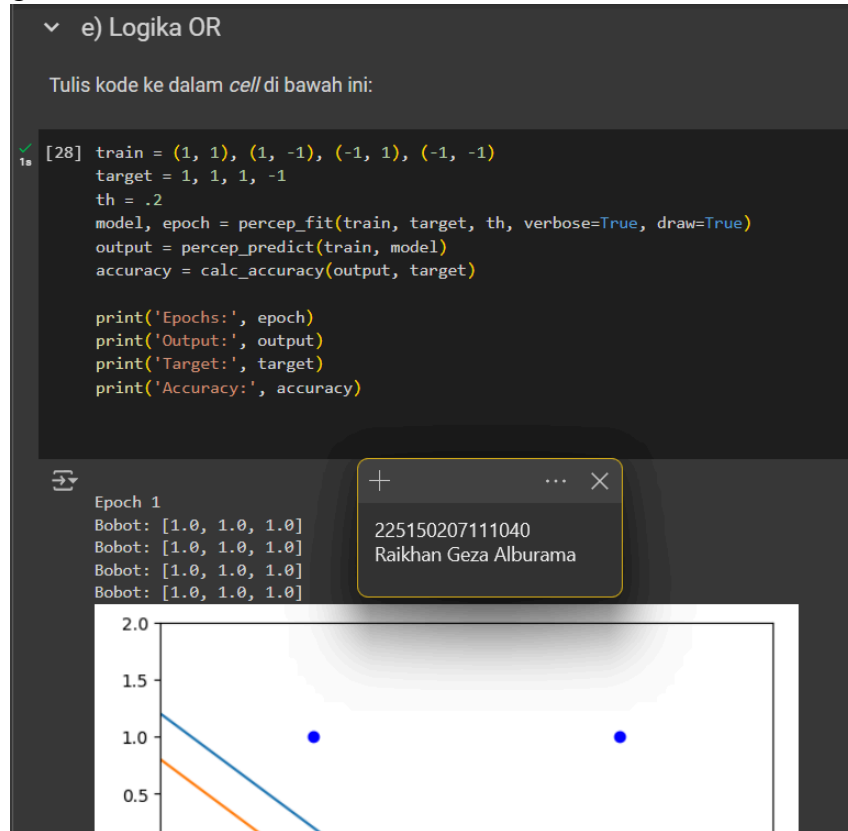
▼ d) Logika AND

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)
```

225150207111040  
Raikhan Geza Alburama

f. Logika OR





g.



Tulis kode ke dalam *cell* di bawah ini:

✓  
0s



h. Logika XOR

g) Logika XOR

Tulis kode ke dalam *cell* di bawah ini:

```
[30] 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('Epochs:', epoch)
      print('Output:', output)
      print('Accuracy:', accuracy)
```

Bobot: [0.0, 0.0, 0.0]

Epoch 42  
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 43  
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 44  
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 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]

225150207111040  
Raikhan Geza Alburama

i. Load dan plot data

h) Load dan plot data

```
[31] import seaborn as sns
      import matplotlib.pyplot as plt

      iris = sns.load_dataset('iris')

      sns.pairplot(iris, hue='species')
      plt.show()
```

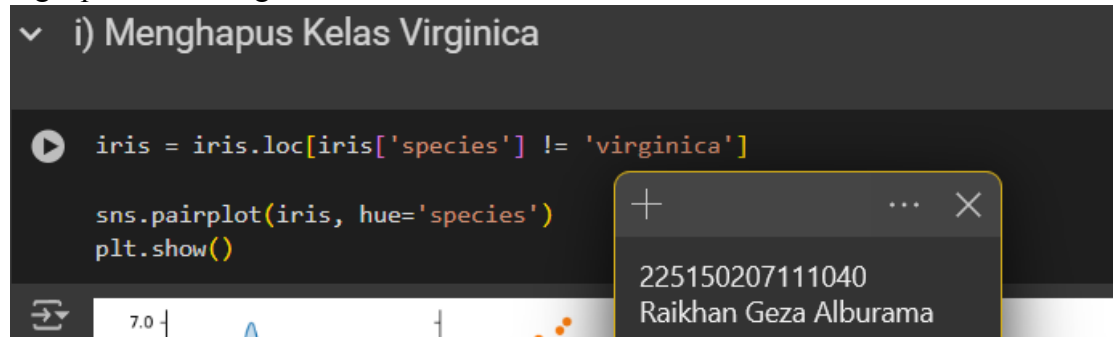
225150207111040  
Raikhan Geza Alburama

j. Menghapus Kelas Virginica

▼ i) Menghapus Kelas Virginica

```
iris = iris.loc[iris['species'] != 'virginica']

sns.pairplot(iris, hue='species')
plt.show()
```

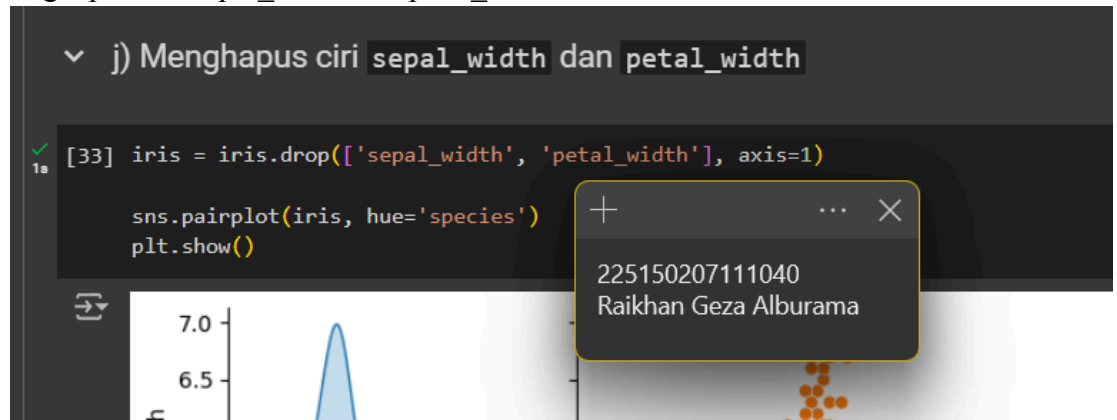


k. Menghapus ciri sepal\_width dan petal\_width

▼ j) Menghapus ciri sepal\_width dan petal\_width

```
[33] iris = iris.drop(['sepal_width', 'petal_width'], axis=1)

sns.pairplot(iris, hue='species')
plt.show()
```



## 1. Proses Training dan Testing

```
✓ 0s k) Proses Training dan Testing

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import minmax_scale

X = iris[['sepal_length', 'petal_length']].to_numpy()
X = minmax_scale(X)

y = iris['species'].to_numpy()
c = {'setosa': -1, 'versicolor': 1}
y = [c[i] for i in y]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.3)
# Remove the marker array
w, epoch = perceptron_fit(X_train, y_train, verbose=True, draw=True)
out = perceptron_predict(X_test, w)
accuracy = calc_accuracy(out, y_test)

print('Epochs:', epoch)
print('Accuracy:', accuracy)
```

ndarray: X\_train  
ndarray with shape (70, 2) X, y, test\_size=.3

225150207111040  
Raikhan Geza Alburama

Epoch 1  
Bobot: [1.0, 0.8888888888888891, 0.902439024390244]  
Bobot: [0.0, 0.5555555555555556, 0.804878048780488]  
Bobot: [0.0, 0.5555555555555556, 0.804878048780488]  
Bobot: [0.0, 0.5555555555555556, 0.804878048780488]  
Bobot: [0.0, 0.5555555555555556, 0.804878048780488]  
Bobot: [0.0, 0.5555555555555556, 0.804878048780488]  
Bobot: [-1.0, 0.2592592592592593, 0.6585365853658538]  
Bobot: [-1.0, 0.2592592592592593, 0.6585365853658538]  
Bobot: [0.0, 1.0, 1.609756097560976]  
Bobot: [-1.0, 0.4814814814814812, 1.4390243902439028]  
Bobot: [-1.0, 0.4814814814814812, 1.4390243902439028]  
Bobot: [-1.0, 0.4814814814814812, 1.4390243902439028]

### C. Analisis

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

Jawab : Karena perceptron menggunakan decision boundary yang berupa garis linear. Pemisahan dari logika XOR tidak dapat dilakukan sehingga hal tersebut yang akan menyebabkan model gagal melakukan proses training.

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

Jawab: Peningkatan nilai learning rate akan mengurangi jumlah epoch yang dibutuhkan untuk mencapai konvergensi. Namun, jika learning rate ditetapkan terlalu tinggi, hal ini dapat menyebabkan ketidakstabilan pada model, sehingga membuatnya gagal dalam mencapai konvergensi dengan baik. Sebaliknya, penggunaan learning rate yang terlalu rendah dapat memperlambat proses pelatihan secara signifikan, mengakibatkan waktu yang lebih lama untuk mencapai hasil yang diinginkan. Oleh karena itu, sangat penting untuk memilih nilai learning rate yang tepat agar proses pelatihan dapat berlangsung efisien dan efektif.

### D. Kesimpulan

1. Jelaskan perbedaan antara Perceptron dengan Hebb net?
2. Mengapa learning rate dibutuhkan?
3. Menurut Anda, kasus seperti apa yang bisa diselesaikan dan tidak bisa diselesaikan oleh Perceptron?

Algoritma Perceptron dirancang untuk menyelesaikan masalah yang bersifat linearly separable, atau dalam istilah lain, sebagai linear classifier. Meskipun demikian, Perceptron memiliki keunggulan dibandingkan Hebb Net, karena Perceptron dapat melakukan proses pelatihan secara iteratif dan menjalani banyak epoch. Sementara itu, Hebb Net hanya melatih data satu kali dalam satu epoch, sehingga setiap data latih hanya mendapatkan proses pelatihan sekali. Selain itu, Perceptron menggunakan learning rate, yang berfungsi untuk mengatur kecepatan pelatihan.

Fungsi aktivasi pada algoritma Perceptron memiliki dua ambang nilai, yaitu  $\theta$  dan  $-\theta$ , yang memungkinkan model menghasilkan tiga output: 1, 0, dan -1. Learning rate dalam Perceptron adalah parameter penting yang mengontrol seberapa besar langkah pergeseran decision boundary selama proses pelatihan. Nilai learning rate umumnya berkisar antara 0 hingga 1, dan pemilihan nilai ini sangat berpengaruh terhadap proses dan hasil pelatihan. Jika learning rate terlalu rendah, perubahan bobot dan pergeseran decision boundary akan berlangsung sangat lambat pada setiap iterasi, sehingga proses pelatihan menjadi berkepanjangan. Sebaliknya, jika

learning rate terlalu tinggi, pelatihan dapat berlangsung terlalu cepat dan menyebabkan ketidakstabilan.

Perceptron terbatas pada penyelesaian masalah yang memiliki sifat linearly separable karena hanya memiliki satu neuron output. Contoh masalah yang dapat diselesaikan oleh Perceptron termasuk logika AND dan logika OR. Namun, Perceptron tidak mampu menyelesaikan masalah nonlinearly separable, seperti logika XOR. Untuk mengatasi masalah nonlinearly separable, arsitektur Perceptron dapat dikembangkan menjadi multilayer, yang dikenal sebagai multilayer Perceptron (MLP). Meskipun MLP dapat menyelesaikan masalah tersebut, arsitekturnya lebih kompleks dibandingkan dengan Perceptron tunggal.