

# Perceptron

Randy Cahya Wihandika, S.ST., M.Kom.

# Perceptron

- Dirancang oleh seorang psikolog bernama **Frank Rosenblatt** pada 1957
- Ditujukan untuk ditanamkan pada sebuah mesin
- Diimplementasikan pertama kali pada komputer IBM 704 kemudian pada sebuah mesin bernama “**Mark I Perceptron**”

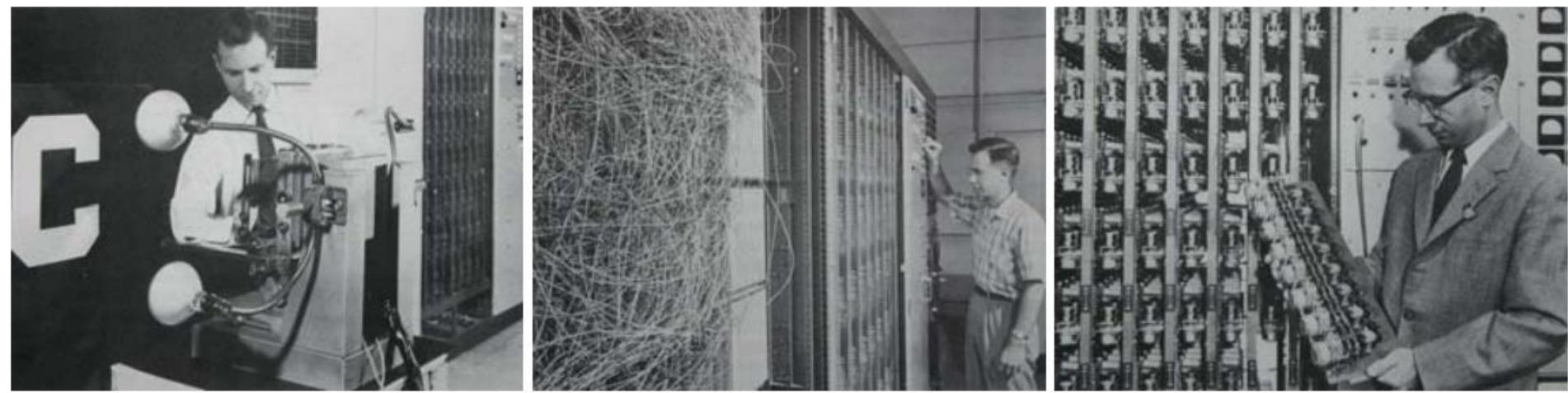


Frank Rosenblatt (1928–1971)



IBM 704

Sumber: <http://www.columbia.edu>



**Figure 4.8** Illustration of the Mark 1 perceptron hardware. The photograph on the left shows how the inputs were obtained using a simple camera system in which an input scene, in this case a printed character, was illuminated by powerful lights, and an image focussed onto a  $20 \times 20$  array of cadmium sulphide photocells, giving a primitive 400 pixel image. The perceptron also had a patch board, shown in the middle photograph, which allowed different configurations of input features to be tried. Often these were wired up at random to demonstrate the ability of the perceptron to learn without the need for precise wiring, in contrast to a modern digital computer. The photograph on the right shows one of the racks of adaptive weights. Each weight was implemented using a rotary variable resistor, also called a potentiometer, driven by an electric motor thereby allowing the value of the weight to be adjusted automatically by the learning algorithm.

Sumber: Bishop (2006)

# NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI)—The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's \$2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of \$100,000.

Dr. Frank Rosenblatt, designer of the Perceptron, conducted the demonstration. He said the machine would be the first device to think as the human brain. As do human be-

ings, Perceptron will make mistakes at first, but will grow wiser as it gains experience, he said.

Dr. Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers.

## Without Human Controls

The Navy said the perceptron would be the first non-living mechanism "capable of receiving, recognizing and identifying its surroundings without any human training or control."

The "brain" is designed to remember images and information it has perceived itself. Ordinary computers remember only what is fed into them on punch cards or magnetic tape.

Later Perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech or writing in another language, it was predicted.

Mr. Rosenblatt said in principle it would be possible to build brains that could reproduce themselves on an assembly line and which would be conscious of their existence.

In today's demonstration, the "704" was fed two cards, one with squares marked on the left side and the other with squares on the right side.

## Learns by Doing

In the first fifty trials, the machine made no distinction between them. It then started registering a "Q" for the left squares and "O" for the right squares.

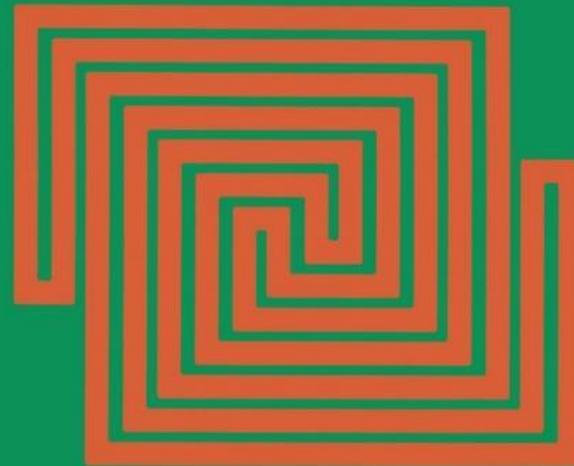
Dr. Rosenblatt said he could explain why the machine learned only in highly technical terms. But he said the computer had undergone a "self-induced change in the wiring diagram."

The first Perceptron will have about 1,000 electronic "association cells" receiving electrical impulses from an eye-like scanning device with 400 photo-cells. The human brain has 10,000,000,000 responsive cells, including 100,000,000 connections with the eyes.

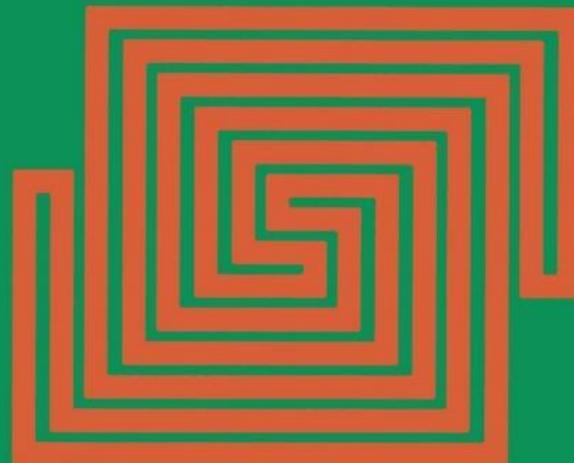
# Perceptron

- Pemberitaan saat itu menimbulkan **kontroversi**
- Pada 1969, Marvin Minsky dan Seymour Papert dalam bukunya yang berjudul “Perceptrons” menunjukkan **keterbatasan** kemampuan Perceptron
- Mereka membuktikan bahwa Perceptron tidak dapat menyelesaikan kasus XOR

Expanded Edition



# Perceptrons



Marvin L. Minsky  
Seymour A. Papert

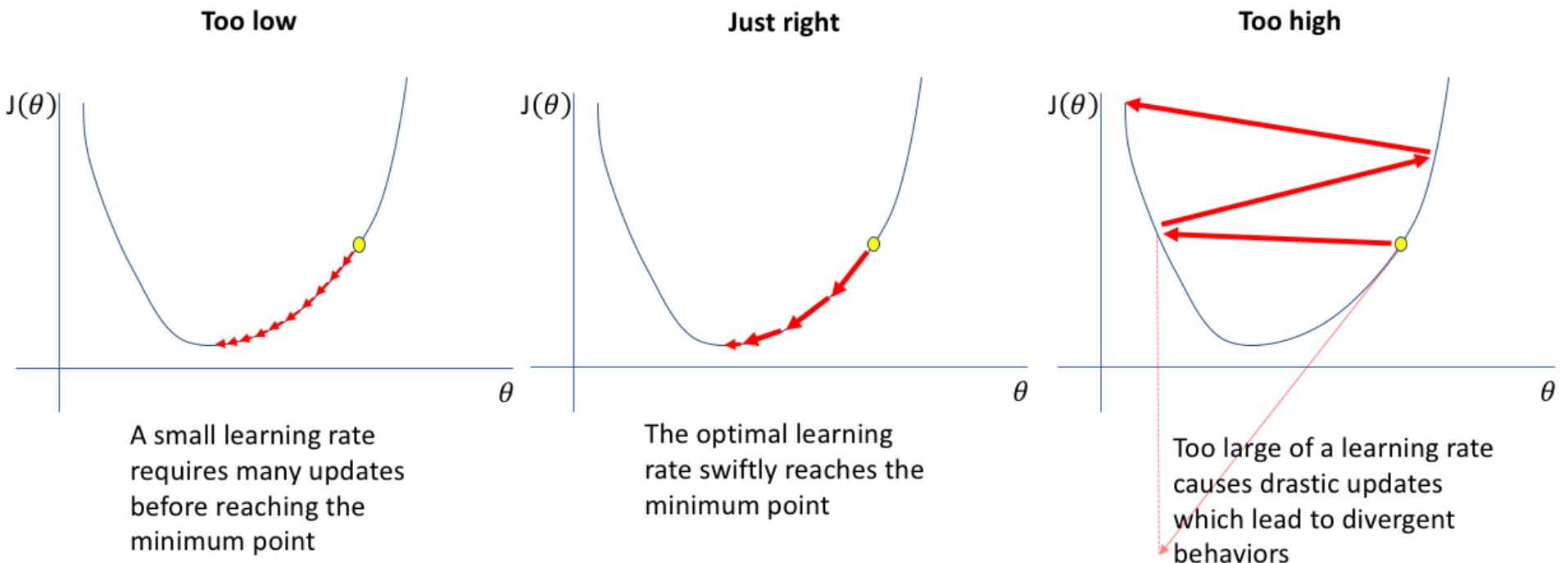
# Perceptron

- Hal tersebut menyebabkan penurunan minat para peneliti untuk mengembangkan JST selama sekitar sepuluh tahun yang disebut dengan *neural network winter*

# Perceptron

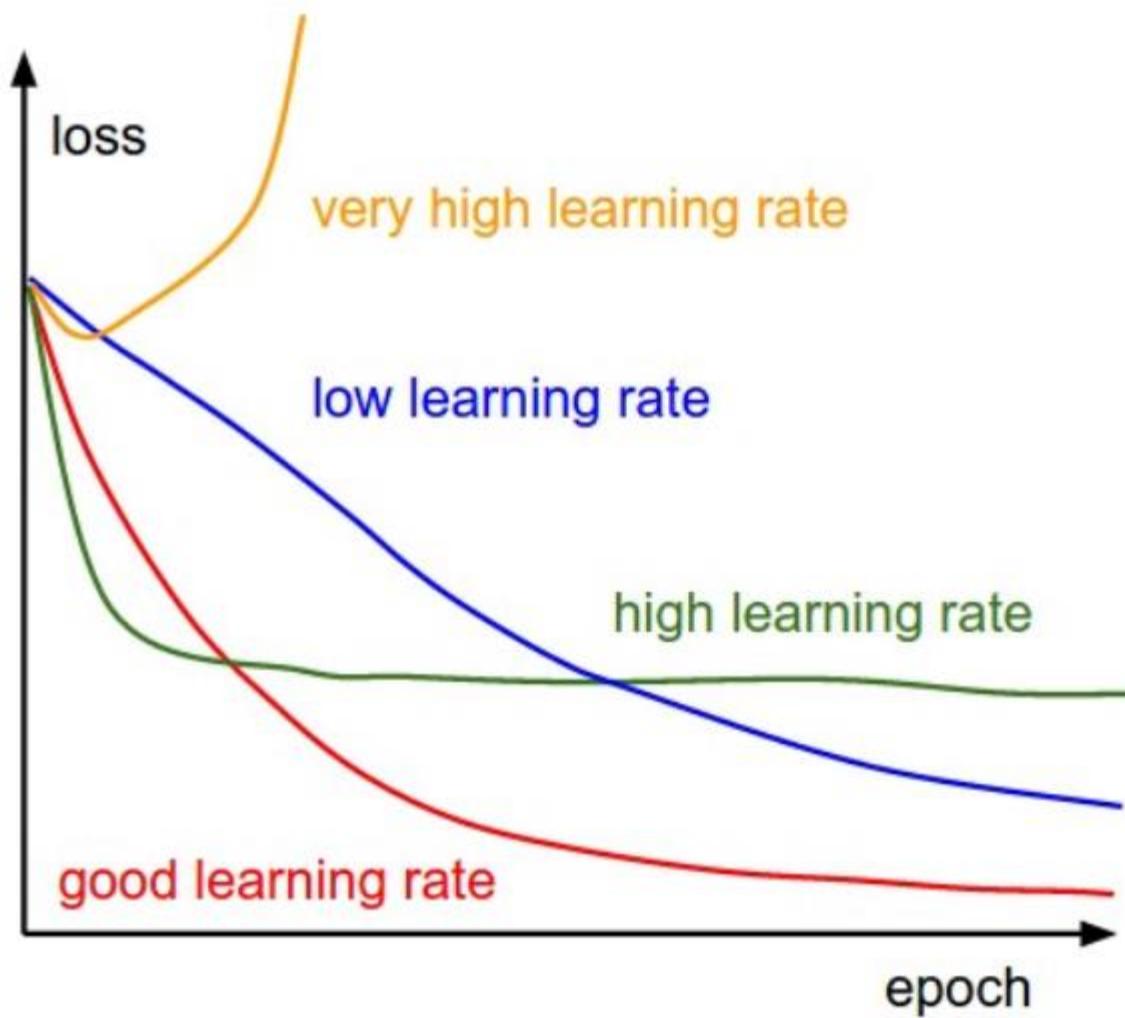
- Lebih baik daripada Hebb rule
- Proses pelatihan iteratif (beberapa *epoch*)
- Menggunakan parameter *learning rate*  $\alpha$  untuk mengatur laju pelatihan
- Parameter  $\alpha$  digunakan pada saat mengubah nilai bobot:

$$w'_i = w_i + \alpha t x_i$$



Sumber: Jordan (2018)

# *Learning Rate*



# Learning Rate

- Jika terlalu **kecil**, maka perubahan nilai bobot sangat kecil pada setiap iterasi dan proses pelatihan akan berjalan lama
- Jika terlalu **besar**, maka perubahan nilai bobot sangat besar pada setiap iterasi dan kemungkinan tidak akan mencapai konvergensi
- Nilai yang umum digunakan adalah 0,1

# Perceptron

- Pada proses pelatihan, dihitung nilai *error* antara nilai *output* dan nilai target
- Jika tidak ada *error*, maka nilai bobot tidak diubah

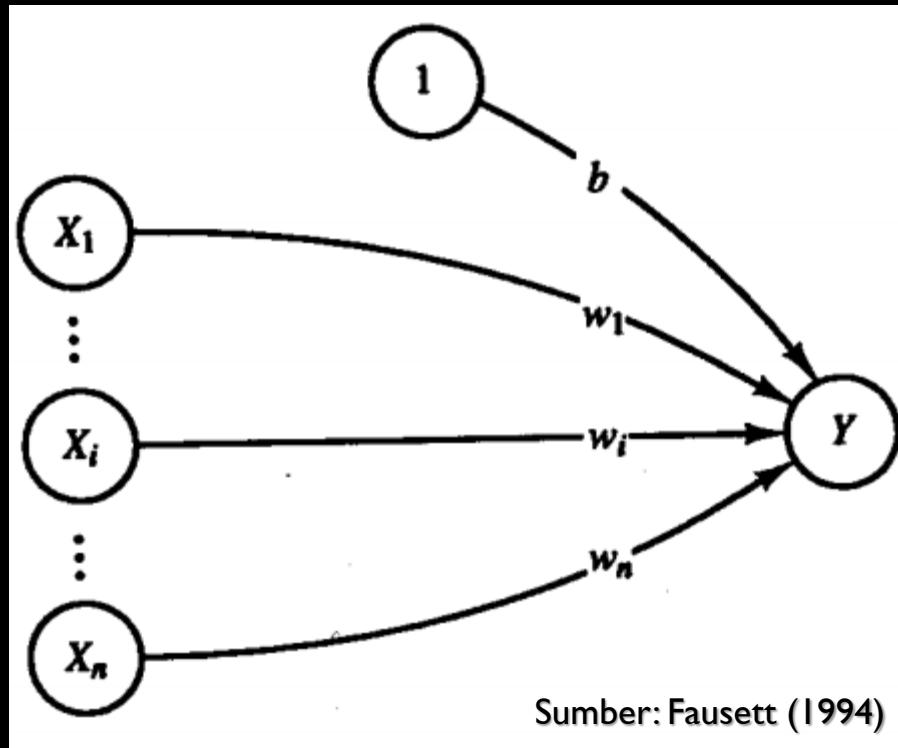
# Perceptron

- Jika data latih *linearly separable*, maka Perceptron akan dapat menemukan nilai bobot yang sesuai
- Jika tidak, maka proses pelatihan Perceptron tidak akan berakhir

# Perceptron

- Fungsi aktivasi:

$$f(y_{in}) = \begin{cases} 1, & \text{if } y_{in} > \theta \\ 0, & \text{if } -\theta \leq y_{in} \leq \theta \\ -1, & \text{if } y_{in} < -\theta \end{cases}$$



# Perceptron

Algoritme:

1. Inisialisasi bobot dan bias  
Set *learning rate*  $\alpha$  ( $0 < \alpha \leq 1$ )
2. Selama kondisi berhenti belum tercapai,  
lakukan langkah 3–7
3. Untuk setiap data latih  $s$  dan target  $t$ ,  
lakukan langkah 4–6

# Perceptron

4. Set nilai aktivasi setiap neuron input:

$$x_i = s_i$$

5. Hitung nilai aktivasi neuron *output*.

$$y_{in} = b + \sum_i x_i w_i$$

$$y_{out} = \begin{cases} 1, & \text{if } y_{in} > \theta \\ 0, & \text{if } -\theta \leq y_{in} \leq \theta \\ -1, & \text{if } y_{in} < -\theta \end{cases}$$

# Perceptron

6. Jika  $y \neq t$ :

$$\begin{aligned}w_i' &= w_i + \alpha tx \\b' &= b + \alpha t\end{aligned}$$

Jika tidak:

$$\begin{aligned}w_i' &= w_i \\b' &= b\end{aligned}$$

# Perceptron

7. Jika tidak ada perubahan nilai bobot, hentikan pelatihan.  
Jika ada, lanjutkan pelatihan.

# Logika AND

- Input biner, target bipolar
- Nilai bobot dan bias awal 0
- Nilai  $\alpha = 1, \theta = 0.2$

$x_1$	$x_2$	$b$	$t$
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

# Logika AND



$x_1$	$x_2$	$b$	$t$
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

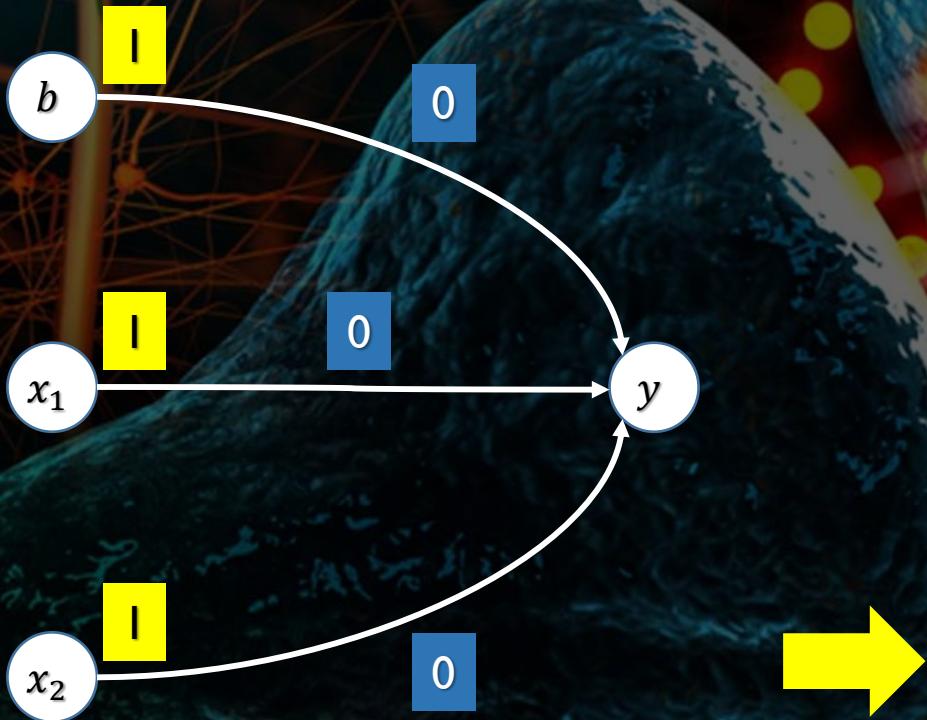
# Logika AND



Inisialisasi semua bobot  
dengan nilai 0

$x_1$	$x_2$	$b$	$t$
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

# Logika AND

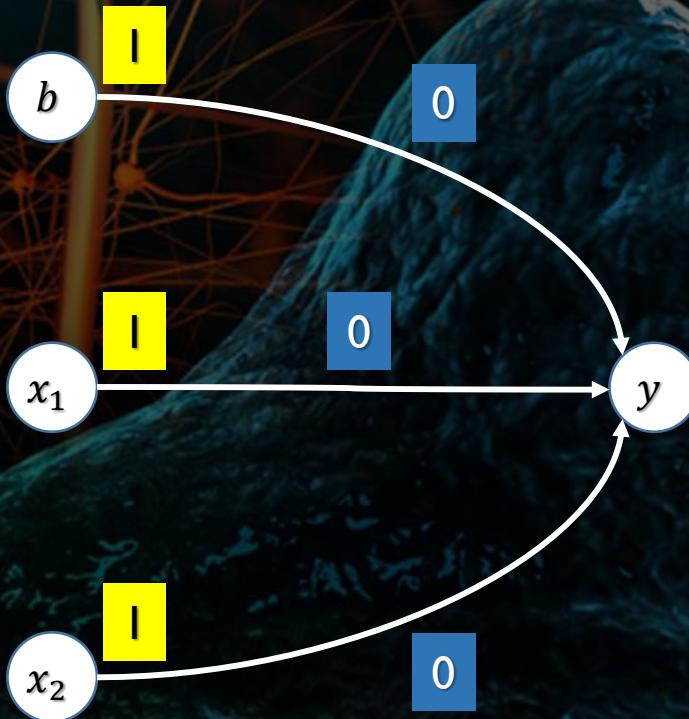


Set nilai aktivasi neuron  
input:

$$x_i = s_i$$

$x_1$	$x_2$	$b$	$t$
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

# Logika AND



Hitung nilai  $y_{in}$ :

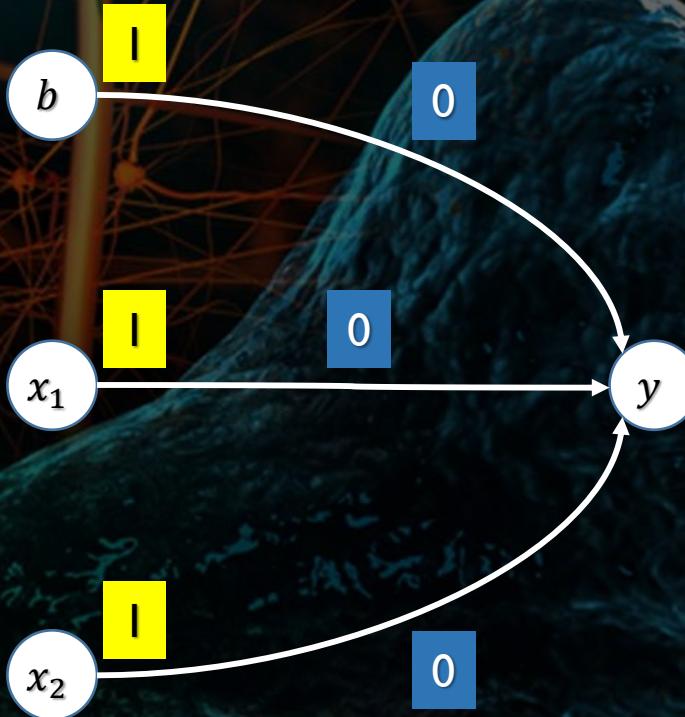
$$y_{in} = b + \sum_i x_i w_i$$

$$y_{in} = 0 + 1.0 + 1.0$$

$$y_{in} = 0$$

$x_1$	$x_2$	$b$	$t$
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

# Logika AND

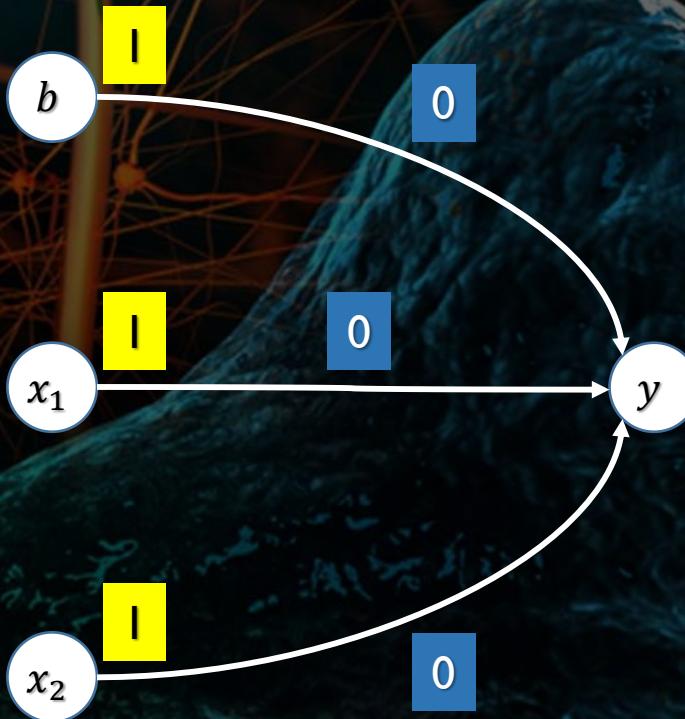


Hitung nilai  $y_{out}$ :

$$y_{out} = \begin{cases} 1, & \text{if } y_{in} > 0.2 \\ 0, & \text{if } -0.2 \leq y_{in} \leq 0.2 \\ -1, & \text{if } y_{in} < -0.2 \end{cases}$$
$$y_{out} = 0$$

$x_1$	$x_2$	$b$	$t$
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

# Logika AND



Terjadi  $y \neq t$

$$b' = b + \alpha t$$

$$b = 0 + 1 \cdot 1 = 1$$

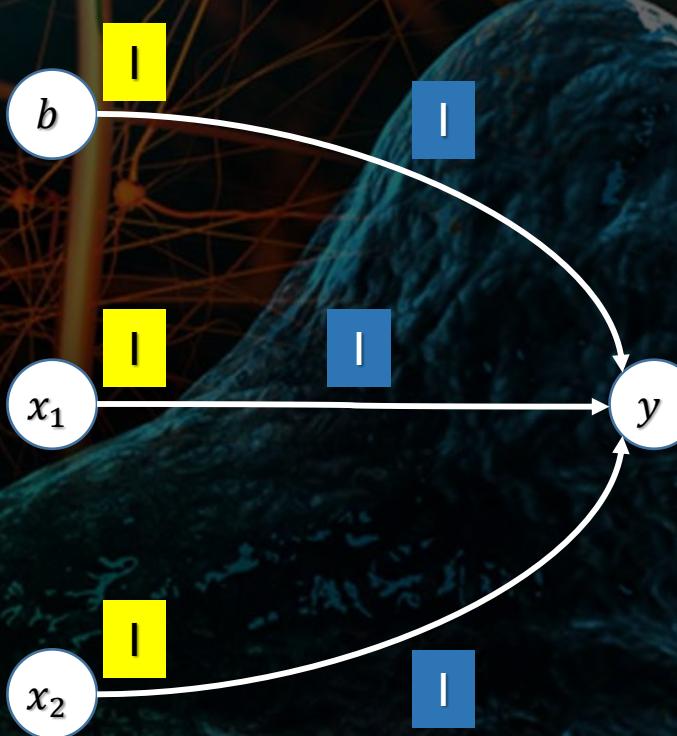
$$w'_i = w_i + \alpha t x$$

$$w_1 = 0 + 1 \cdot 1 \cdot 1 = 1$$

$$w_2 = 0 + 1 \cdot 1 \cdot 1 = 1$$

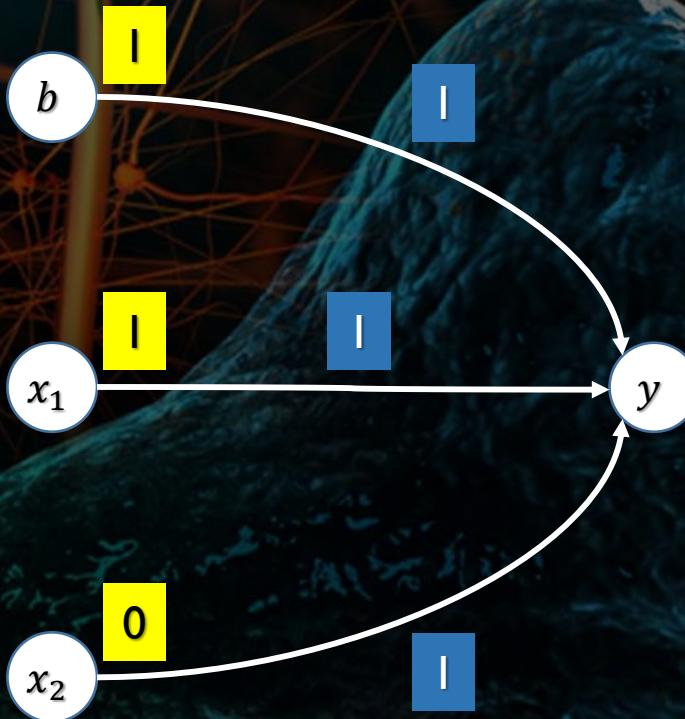
$x_1$	$x_2$	$b$	$t$
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

# Logika AND



$x_1$	$x_2$	$b$	$t$
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

# Logika AND

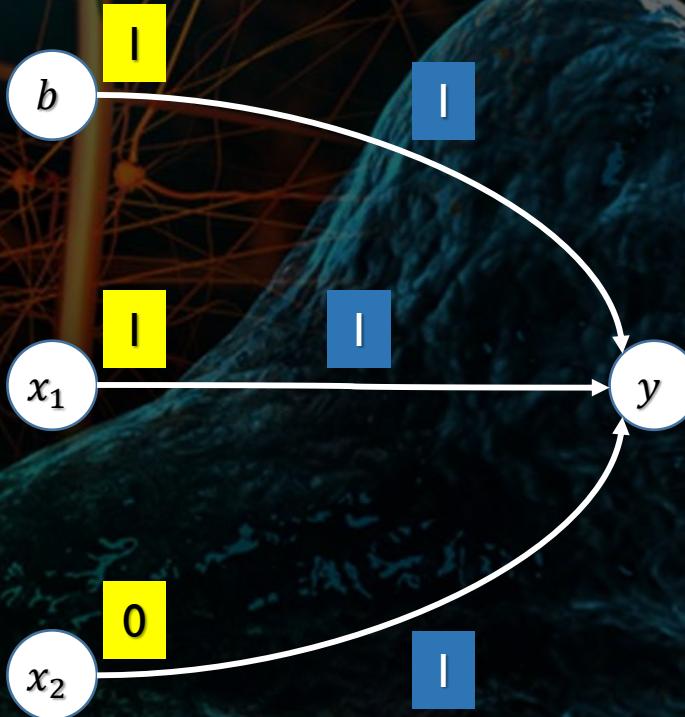


Set nilai aktivasi neuron  
input:

$$x_i = s_i$$

$x_1$	$x_2$	$b$	$t$
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

# Logika AND



Hitung nilai  $y_{in}$ :

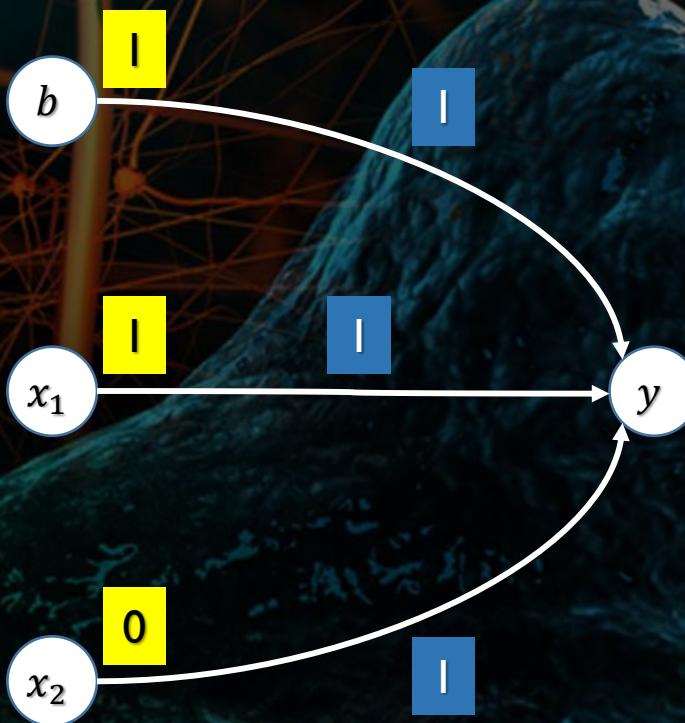
$$y_{in} = b + \sum_i x_i w_i$$

$$y_{in} = 1 + 1 \cdot 1 + 0 \cdot 0$$

$$y_{in} = 2$$

$x_1$	$x_2$	$b$	$t$
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

# Logika AND

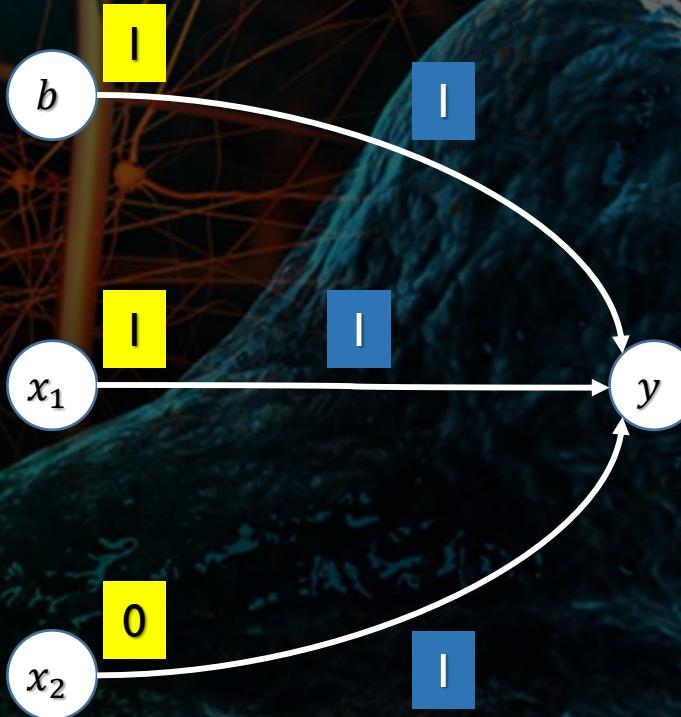


Hitung nilai  $y_{out}$ :

$$y_{out} = \begin{cases} 1, & \text{if } y_{in} > 0.2 \\ 0, & \text{if } -0.2 \leq y_{in} \leq 0.2 \\ -1, & \text{if } y_{in} < -0.2 \end{cases}$$
$$y_{out} = 1$$

$x_1$	$x_2$	$b$	$t$
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

# Logika AND



Terjadi  $y \neq t$

$$b' = b + \alpha t$$

$$b = 1 + 1 \cdot -1 = 0$$

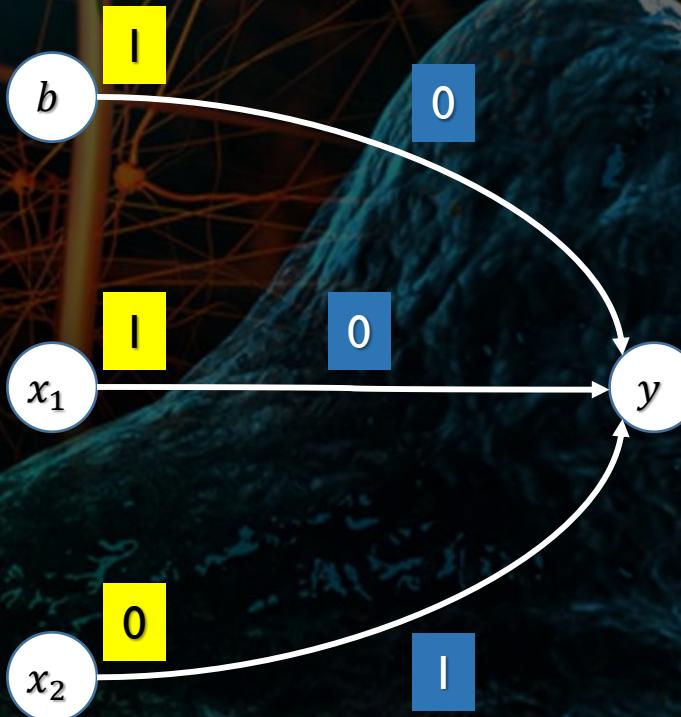
$$w'_i = w_i + \alpha t x$$

$$w_1 = 1 + 1 \cdot -1 \cdot 1 = 0$$

$$w_2 = 1 + 1 \cdot -1 \cdot 0 = 1$$

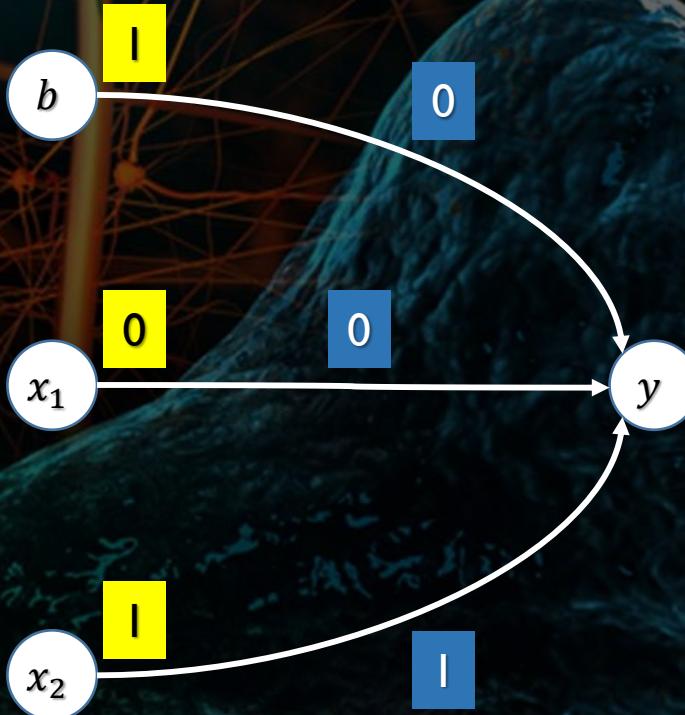
$x_1$	$x_2$	$b$	$t$
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

# Logika AND



$x_1$	$x_2$	$b$	$t$
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

# Logika AND

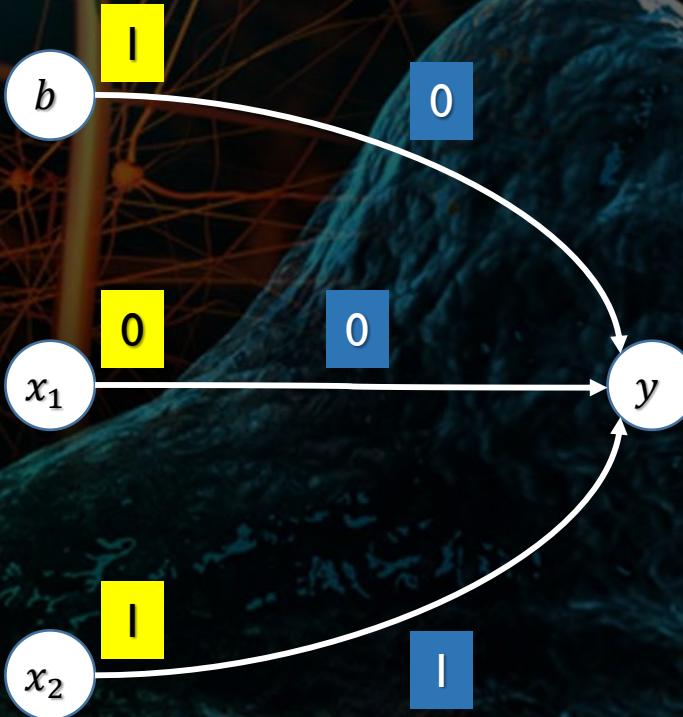


Set nilai aktivasi neuron  
input:

$$x_i = s_i$$

$x_1$	$x_2$	$b$	$t$
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

# Logika AND



Hitung nilai  $y_{in}$ :

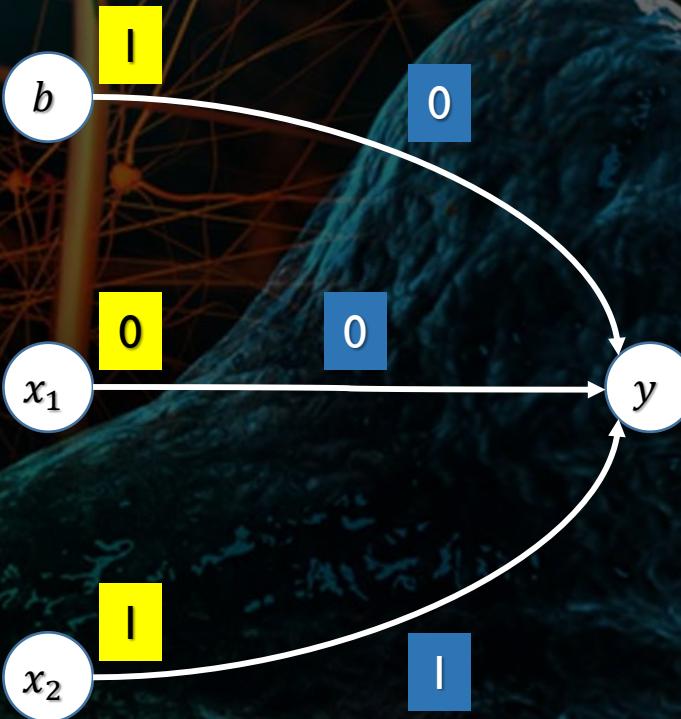
$$y_{in} = b + \sum_i x_i w_i$$

$$y_{in} = 0 + 0.0 + 1.1$$

$$y_{in} = 1$$

$x_1$	$x_2$	$b$	$t$
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

# Logika AND

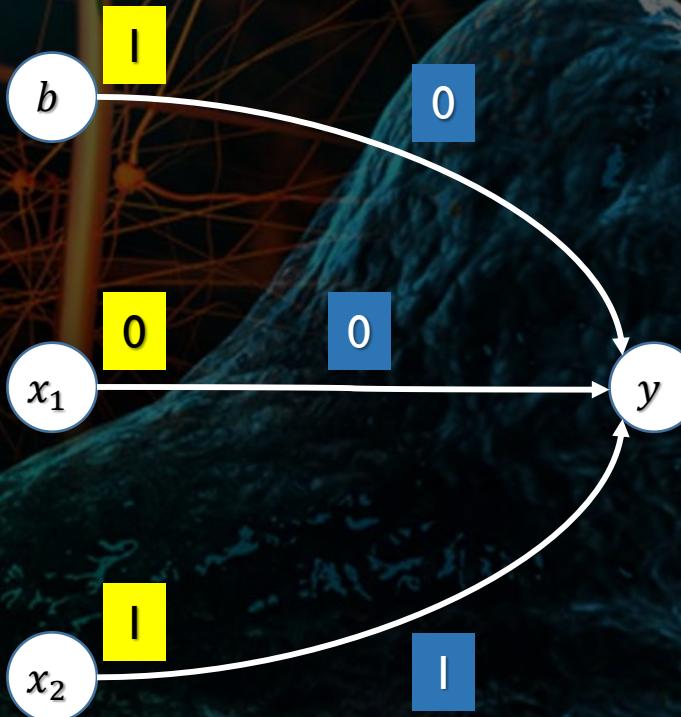


Hitung nilai  $y_{out}$ :

$$y_{out} = \begin{cases} 1, & \text{if } y_{in} > 0.2 \\ 0, & \text{if } -0.2 \leq y_{in} \leq 0.2 \\ -1, & \text{if } y_{in} < -0.2 \end{cases}$$
$$y_{out} = 1$$

$x_1$	$x_2$	$b$	$t$
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

# Logika AND



Terjadi  $y \neq t$

$$b' = b + \alpha t$$

$$b = 0 + 1 \cdot -1 = -1$$

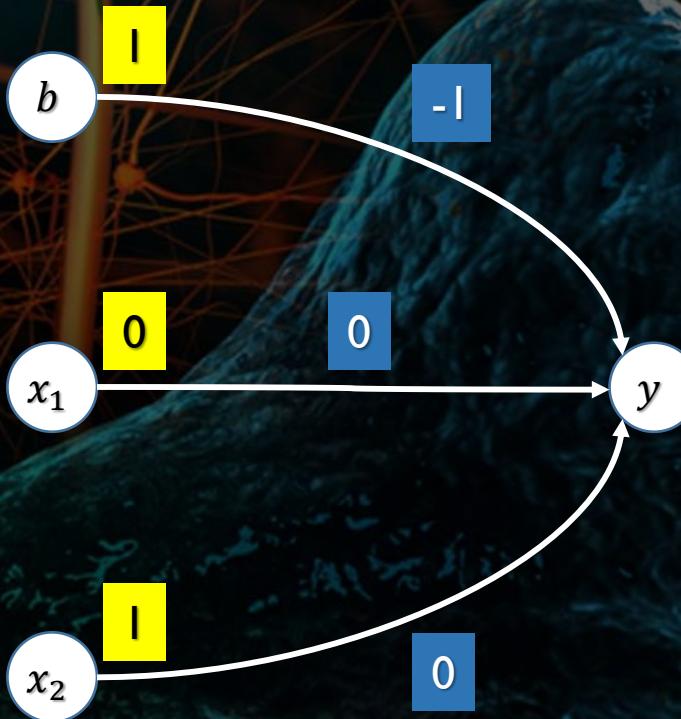
$$w'_i = w_i + \alpha t x$$

$$w_1 = 0 + 1 \cdot -1 \cdot 0 = 0$$

$$w_2 = 1 + 1 \cdot -1 \cdot 1 = 0$$

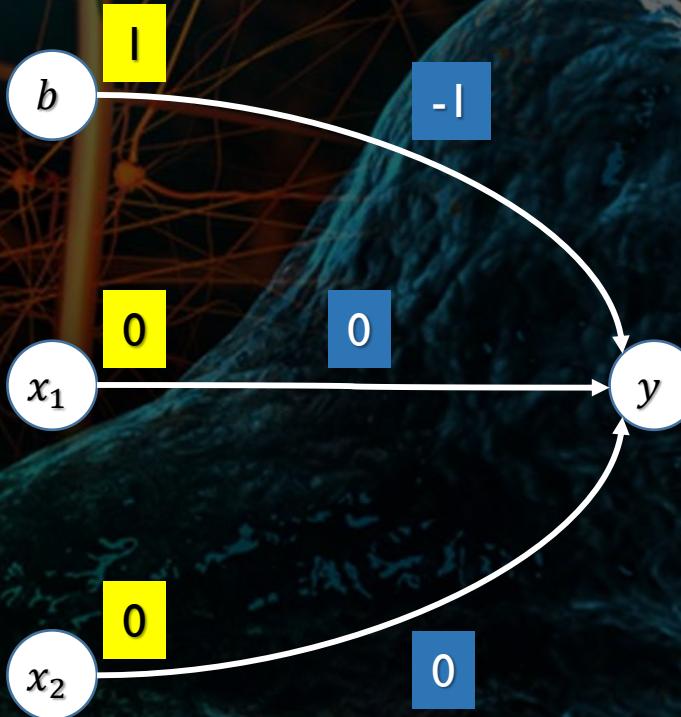
$x_1$	$x_2$	$b$	$t$
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

# Logika AND



$x_1$	$x_2$	$b$	$t$
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

# Logika AND

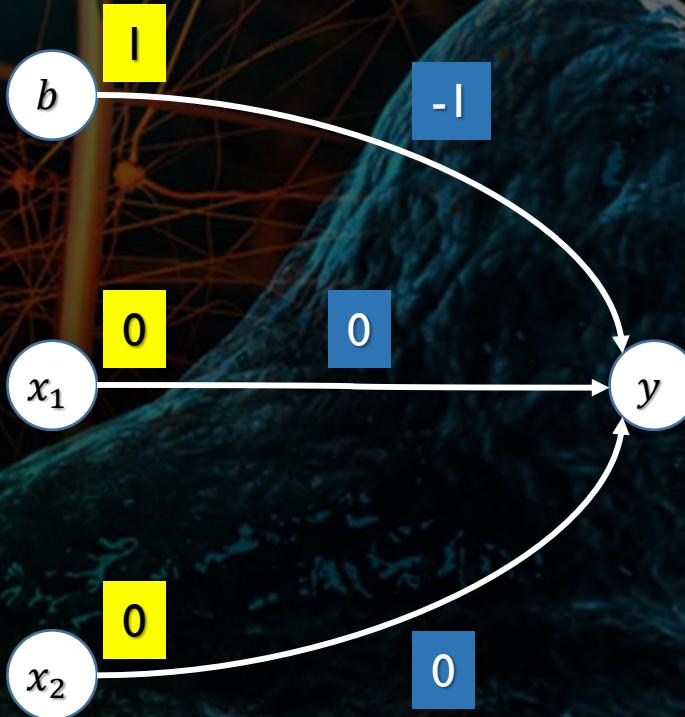


Set nilai aktivasi neuron  
input:

$$x_i = s_i$$

$x_1$	$x_2$	$b$	$t$
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

# Logika AND



Hitung nilai  $y_{in}$ :

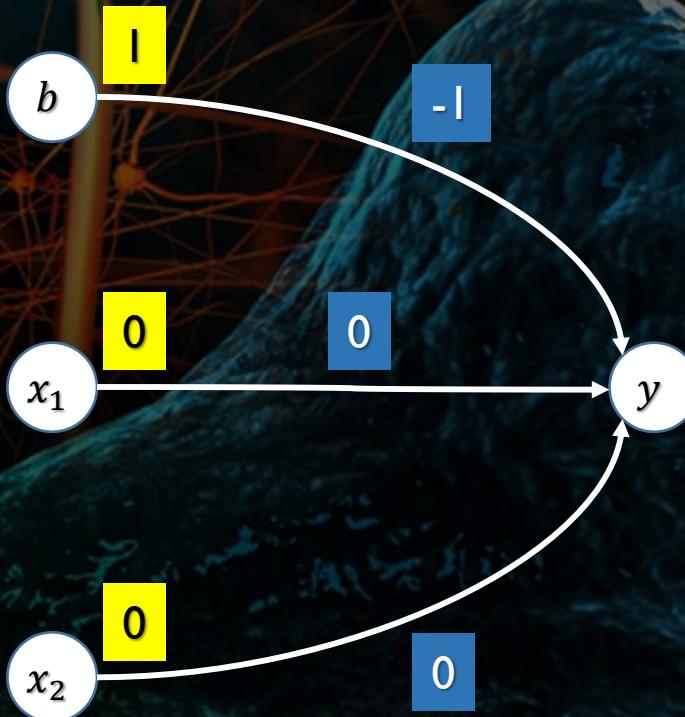
$$y_{in} = b + \sum_i x_i w_i$$

$$y_{in} = -1 + 0.0 + 0.0$$

$$y_{in} = -1$$

$x_1$	$x_2$	$b$	$t$
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

# Logika AND

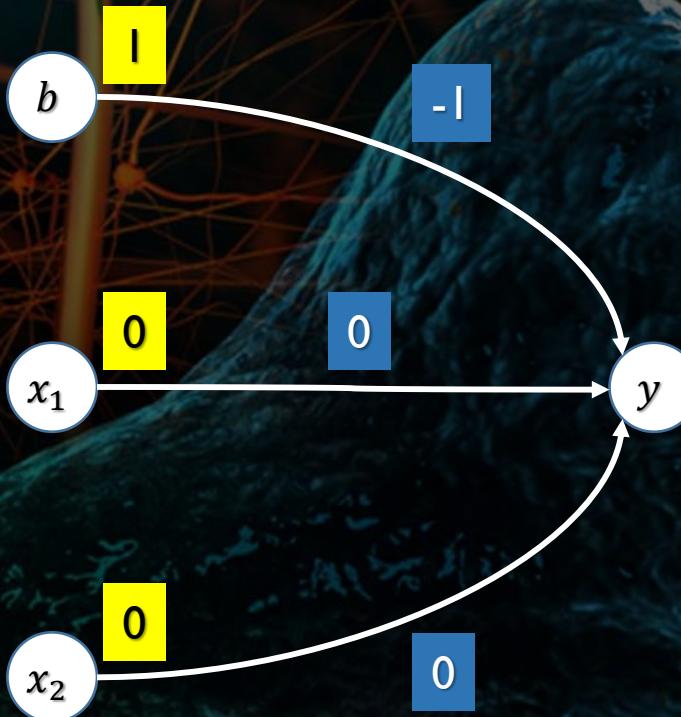


Hitung nilai  $y_{out}$ :

$$y_{out} = \begin{cases} 1, & \text{if } y_{in} > 0.2 \\ 0, & \text{if } -0.2 \leq y_{in} \leq 0.2 \\ -1, & \text{if } y_{in} < -0.2 \end{cases}$$
$$y_{out} = -1$$

$x_1$	$x_2$	$b$	$t$
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

# Logika AND



Tidak terjadi  $y \neq t$

$x_1$	$x_2$	$b$	$t$
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

# Logika AND

- Epoch #1
- Epoch #2

$b$	$w_1$	$w_2$
1	1	1
0	0	1
-1	0	0
-1	0	0

$b$	$w_1$	$w_2$
0	1	1
-1	0	1
-2	0	0
-2	0	0

# Logika AND

- Epoch #3
- Epoch #4

$b$	$w_1$	$w_2$
-1	1	1
-2	0	1
-2	0	1
-2	0	1

$b$	$w_1$	$w_2$
-1	1	2
-2	0	2
-3	0	1
-3	0	1

# Logika AND

- Epoch #5
- Epoch #6

$b$	$w_1$	$w_2$
-2	1	2
-2	1	2
-3	1	1
-3	1	1

$b$	$w_1$	$w_2$
-2	2	2
-3	1	2
-3	1	2
-3	1	2

# Logika AND

- Epoch #7
- Epoch #8

$b$	$w_1$	$w_2$
-2	2	3
-3	1	3
-4	1	2
-4	1	2

$b$	$w_1$	$w_2$
-3	2	3
-3	2	3
-4	2	2
-4	2	2

# Logika AND

- Epoch #9
- Epoch #10

$b$	$w_1$	$w_2$
-3	3	3
-4	2	3
-4	2	3
-4	2	3

$b$	$w_1$	$w_2$
-4	2	3
-4	2	3
-4	2	3
-4	2	3

# Logika AND

- Nilai bobot yang dihasilkan:

$b$	$w_1$	$w_2$
-4	2	3

- Persamaan *decision boundary* pertama:

$$b + x_1 w_1 + x_2 w_2 = \theta$$

$$-4 + 2x_1 + 3x_2 = 0.2$$

$$x_2 = -\frac{2}{3}x_1 + \frac{7}{5}$$

# Logika AND

- Nilai bobot yang dihasilkan:

$b$	$w_1$	$w_2$
-4	2	3

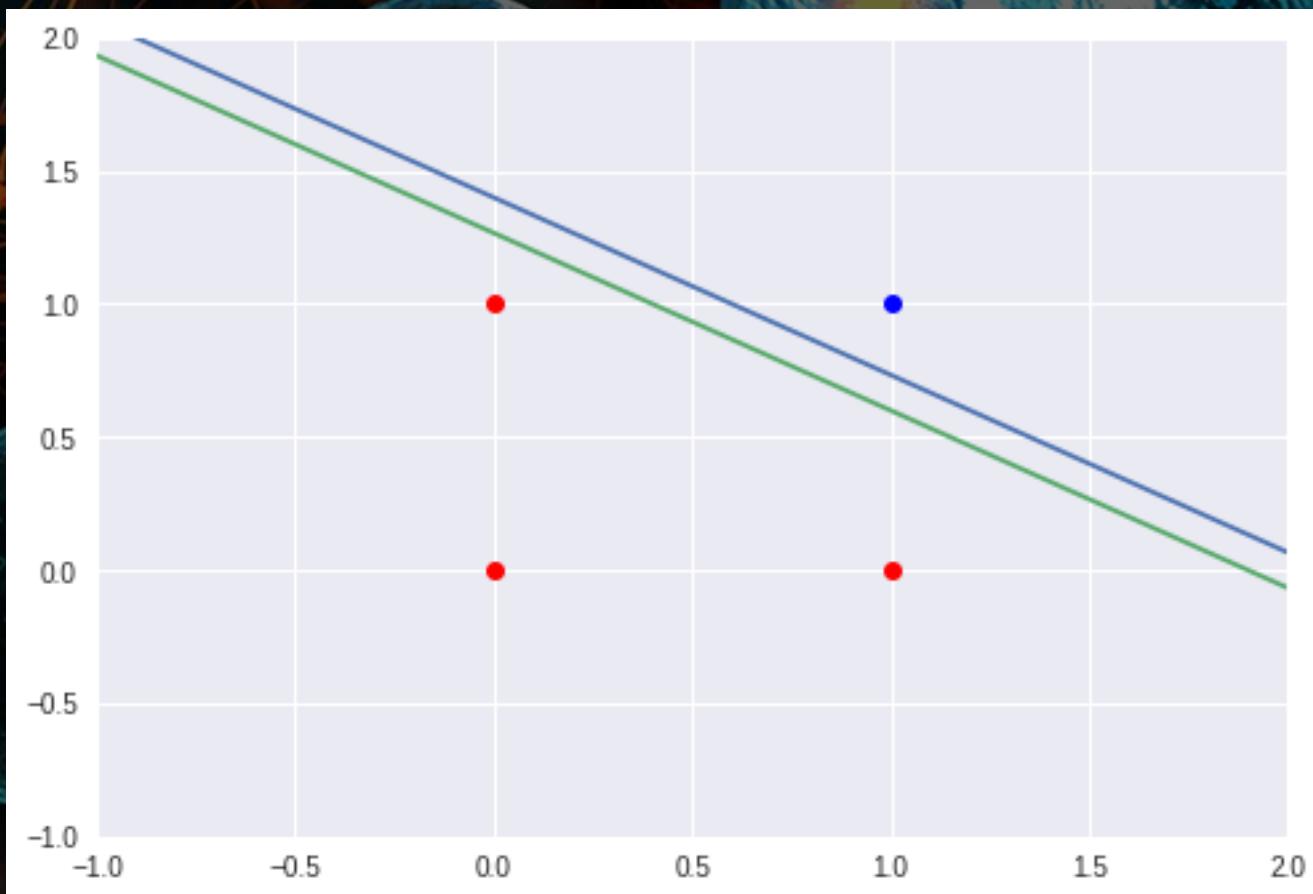
- Persamaan *decision boundary* kedua:

$$b + x_1 w_1 + x_2 w_2 = -\theta$$

$$-4 + 2x_1 + 3x_2 = -0.2$$

$$x_2 = -\frac{2}{3}x_1 + \frac{19}{15}$$

# Logika AND



# Implementasi

```
import numpy as np
import matplotlib.pyplot as plt

# Melakukan thresholding dengan dua nilai threshold,
# yaitu +th dan -th
def percep_step(input, th=0):
    return 1 if input > th else 0 if -th <= input <= th
else -1

# Membuat persamaan garis untuk visualisasi
def line(w, th=0):
    w2 = w[2] + .001 if w[2] == 0 else w[2]

    return lambda x: (th - w[1] * x - w[0]) / w2
```

# Implementasi

```
# Menggambar grafik menggunakan matplotlib
def plot(f1, f2, s, t):
    x = np.arange(-2, 3)
    col = 'ro', 'bo'

    for c, v in enumerate(np.unique(t)):
        p = s[np.where(t == v)]

        plt.plot(p[:,1], p[:,2], col[c])

    plt.axis([-2, 2, -2, 2])
    plt.plot(x, f1(x))
    plt.plot(x, f2(x))
    plt.show()
```

```
def percep_train(s, t, th=0, a=1, draw=False):
    # Inisialisasi bobot
    w = np.zeros(len(s[0]) + 1)

    # Inisialisasi bias
    b = np.ones((len(s), 1))

    # Menggabungkan bias dan data latih menjadi satu layer
    s = np.hstack((b, s))

    # Variabel kondisi berhenti
    stop = False
    epoch = 0

    # Lakukan pelatihan selama kondisi berhenti bernilai False
    while not stop:
        stop = True
        epoch += 1

        print(f'\nEpoch #{epoch}')
```

```
print(f'\nEpoch #{epoch}')

for r, row in enumerate(s):
    # Hitung yin menggunakan dot product
    y_in = np.dot(row, w)

    # Hitung y menggunakan fungsi step
    y = percep_step(y_in, th)

    # Jika output tidak sama dengan target
    if y != t[r]:
        stop = False

    # Ubah nilai bobot
    w = [w[i] + a * t[r] * row[i] for i in
range(len(row))]

    print(f'Bobot: {w}')

    # Buat grafik jika parameter draw bernilai True
    if draw:
        plot(line(w, th), line(w, -th), s, t)

return w
```

# Implementasi

```
# Melakukan pengujian menggunakan nilai bobot (w)
# yang dihasilkan oleh fungsi percep_train()

def percep_test(x, w, th=0):
    y_in = w[0] + np.dot(x, w[1:])
    return percep_step(y_in, th)
```

# Implementasi Logika AND

```
# Logika AND  
# Input biner, output bipolar  
  
train = [[1, 1], [1, 0], [0, 1], [0, 0]]  
target = [1, -1, -1, -1]  
th = .2  
w = percep_train(train, target, th, draw=True)  
  
print(percep_test([1, 1], w, th))
```

# Implementasi Logika AND

```
# Logika AND  
# Input bipolar, output bipolar  
  
train = [[1, 1], [1, -1], [-1, 1], [-1, -1]]  
target = [1, -1, -1, -1]  
w = percep_train(train, target, th, draw=True)  
  
print(percep_test([1, 1], w))
```

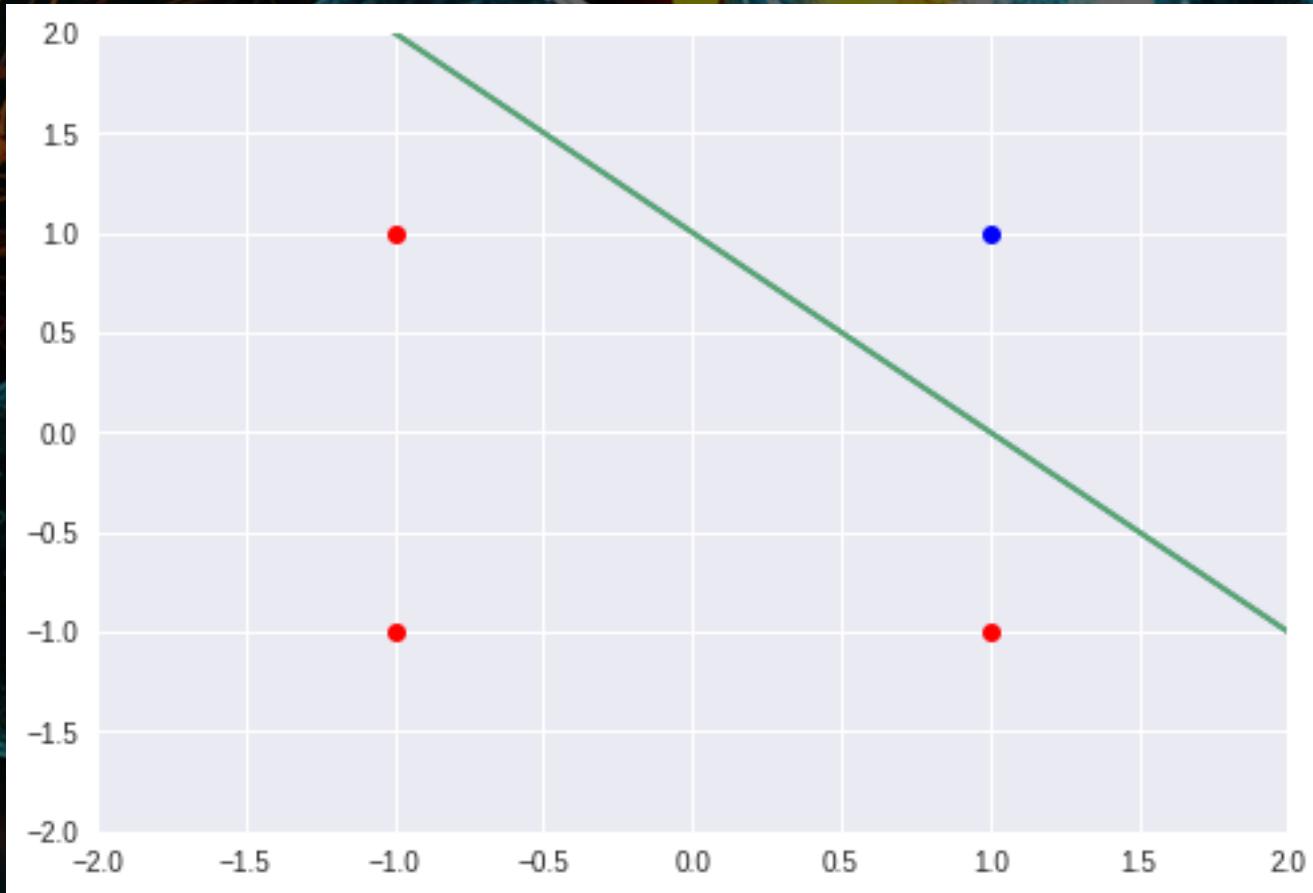
# Logika AND

- Epoch #1
- Epoch #2

$b$	$w_1$	$w_2$
1	1	1
0	0	2
-1	1	1
-1	1	1

$b$	$w_1$	$w_2$
-1	1	1
-1	1	1
-1	1	1
-1	1	1

# Logika AND



# Implementasi Logika OR

```
# Logika OR  
# Input bipolar, output bipolar  
  
train = [[1, 1], [1, -1], [-1, 1], [-1, -1]]  
target = [1, 1, 1, -1]  
w = percep_train(train, target)  
  
print(percep_test([1, 1], w))
```

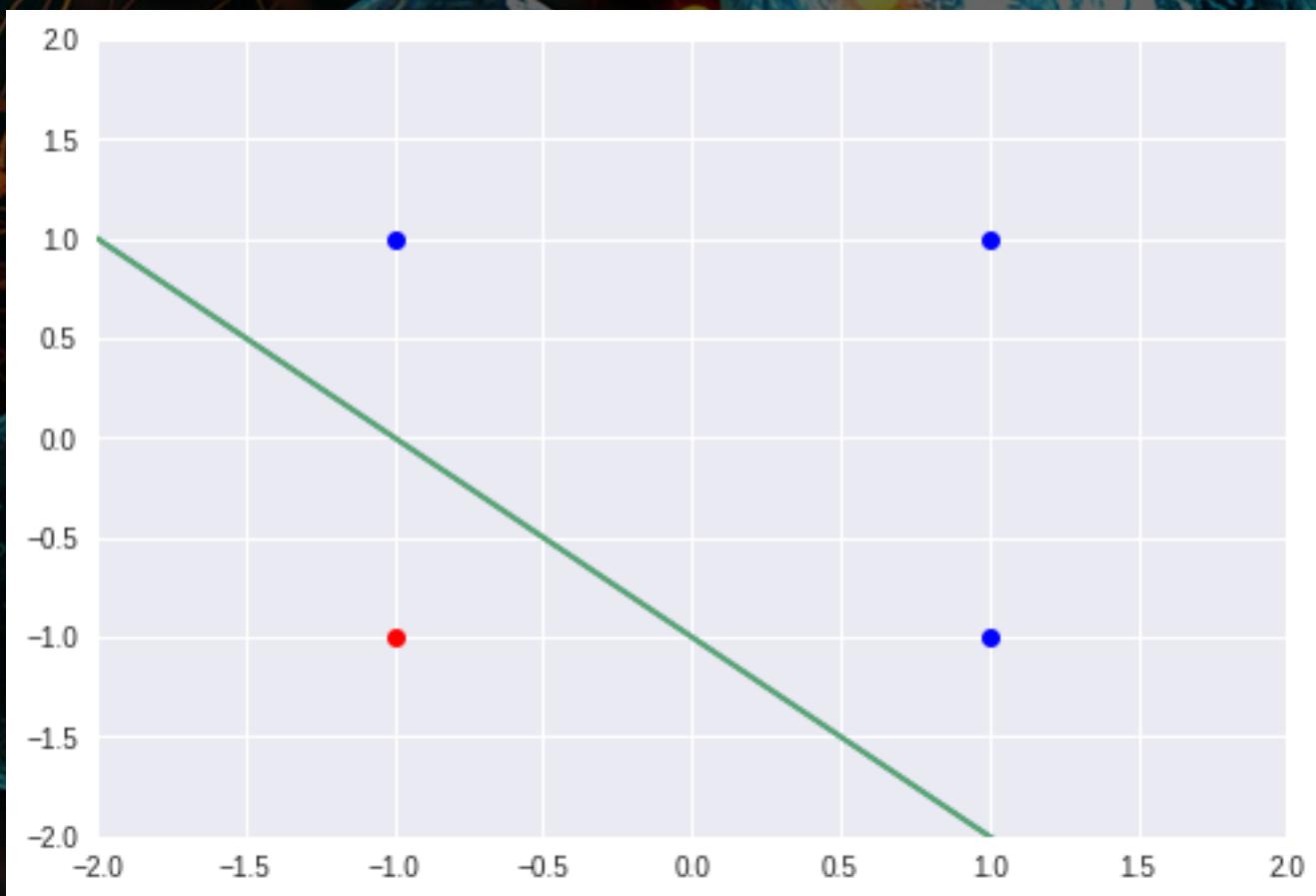
# Logika OR

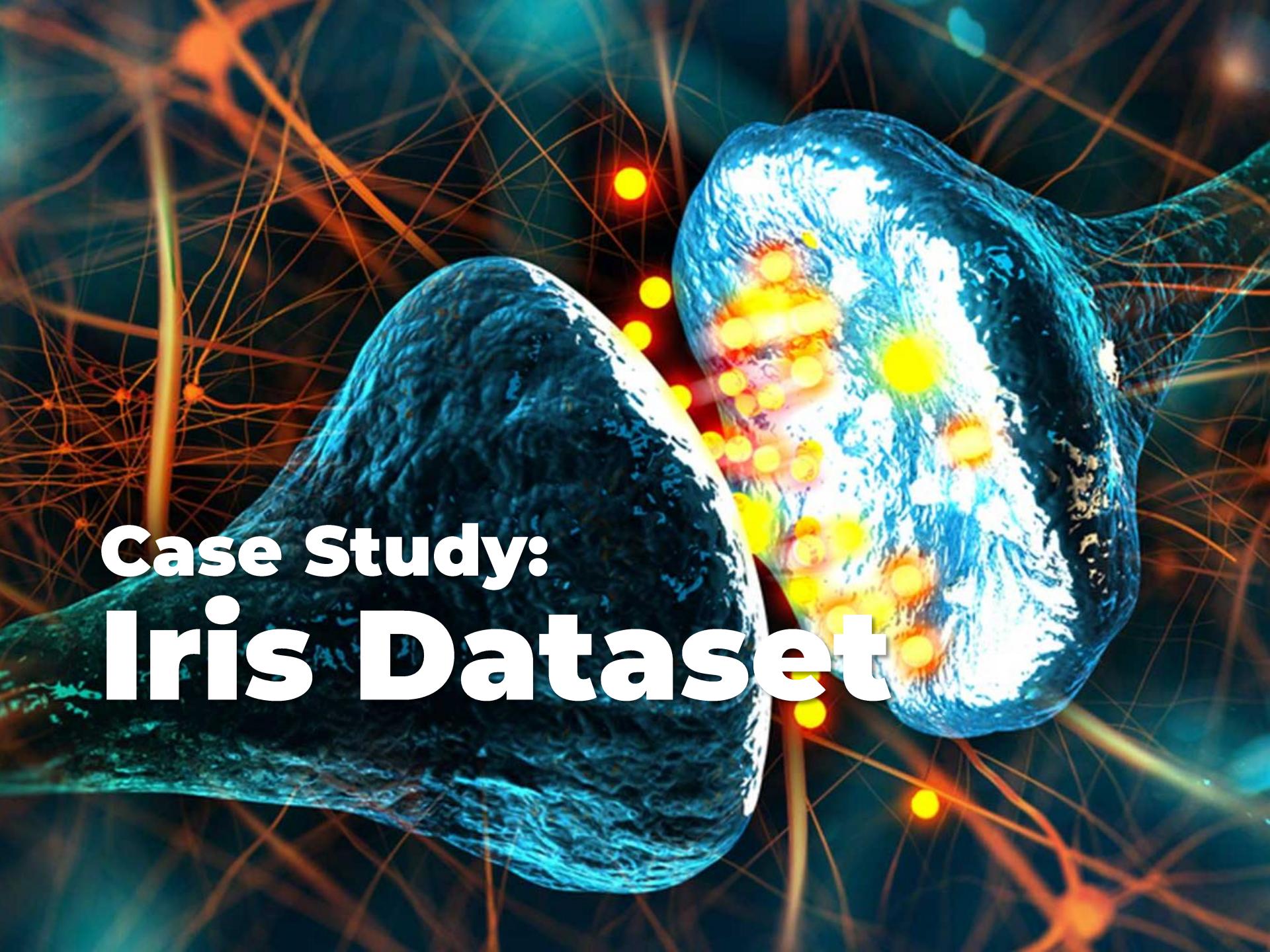
- Epoch #1
- Epoch #2

$b$	$w_1$	$w_2$
1	1	1
1	1	1
1	1	1
1	1	1

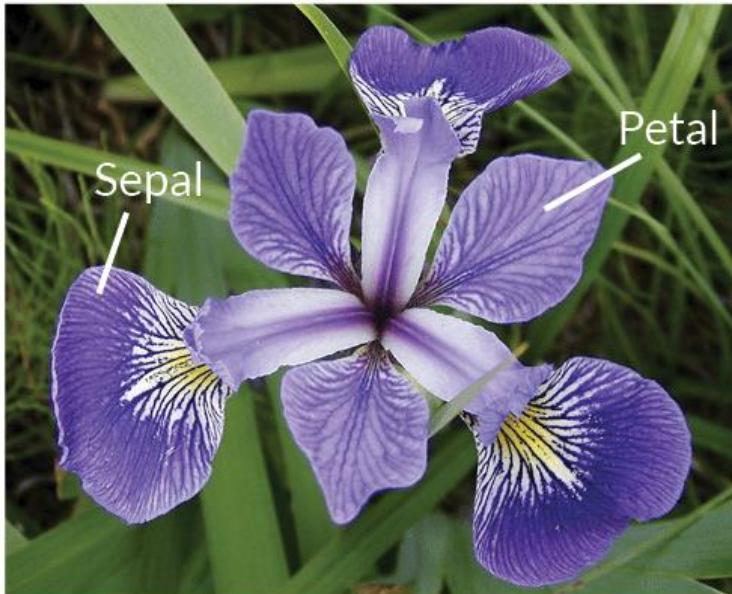
$b$	$w_1$	$w_2$
1	1	1
1	1	1
1	1	1
1	1	1

# Logika OR



A background image showing two large, blue, textured neurons against a dark background. The neurons have numerous thin, orange-red lines extending from them, representing synapses. Small, glowing yellow spheres are scattered throughout the scene, some near the neurons and others in the background.

# Case Study: Iris Dataset



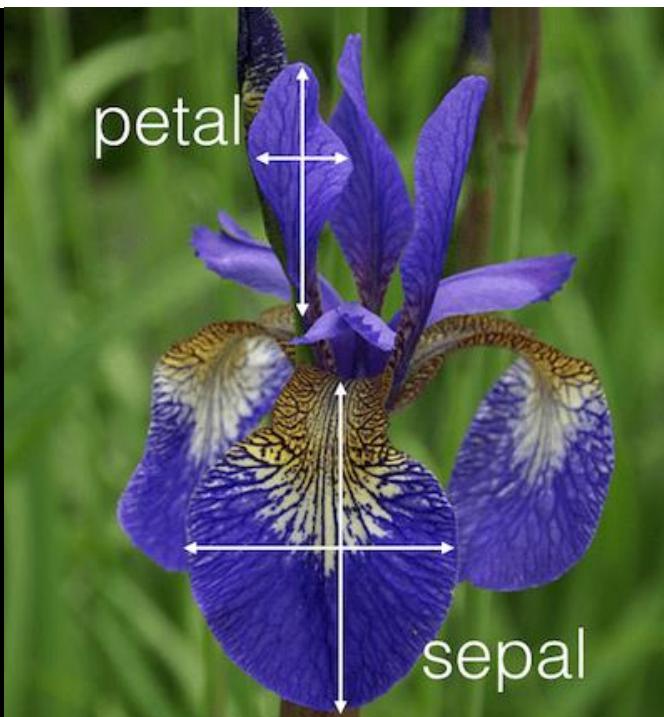
**Iris Versicolor**



**Iris Setosa**

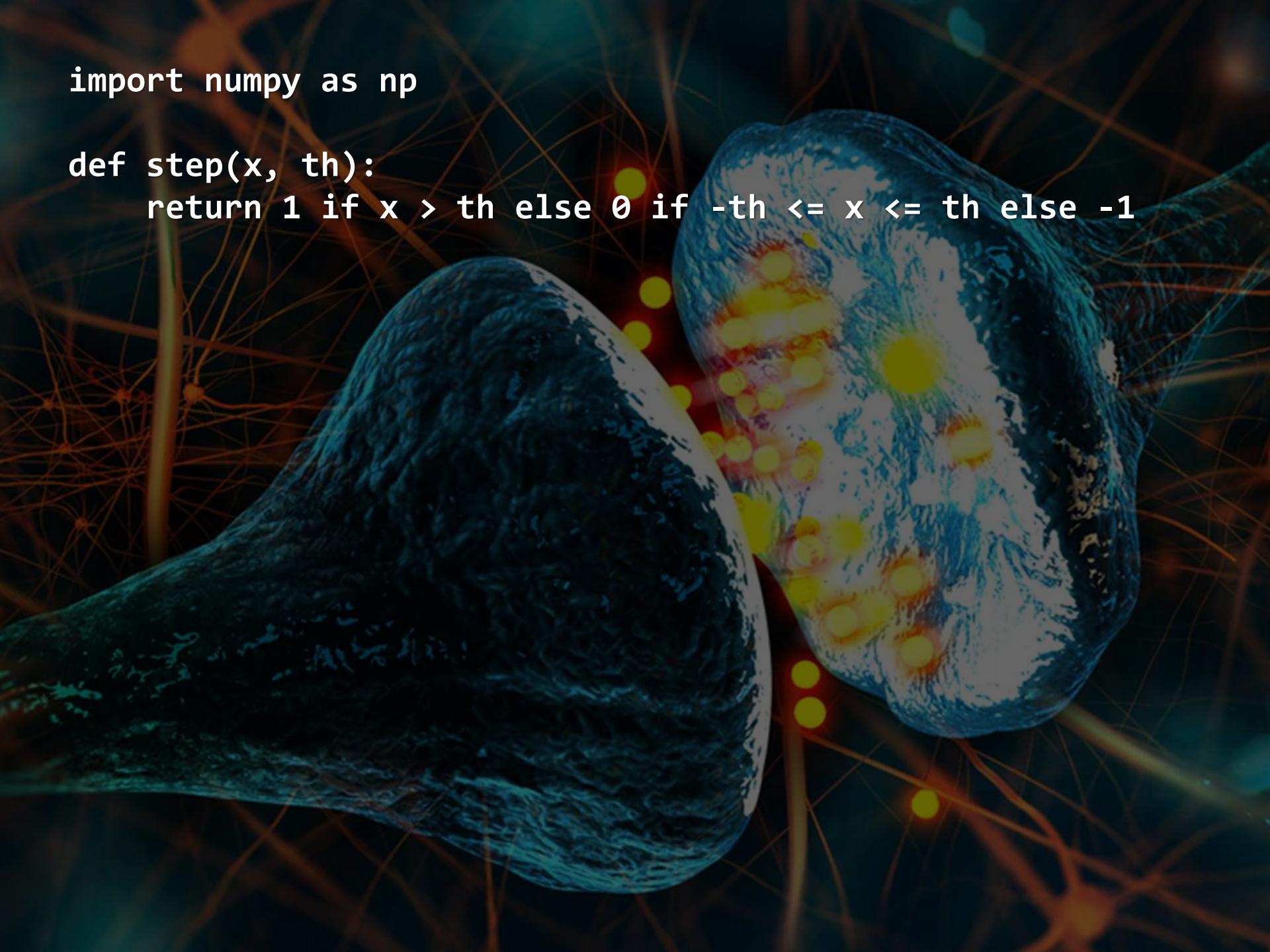


**Iris Virginica**



```
import numpy as np

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



```
def percep_fit(s, t, th=0., a=1):
    # Jumlah neuron input
    nx = len(s[0])

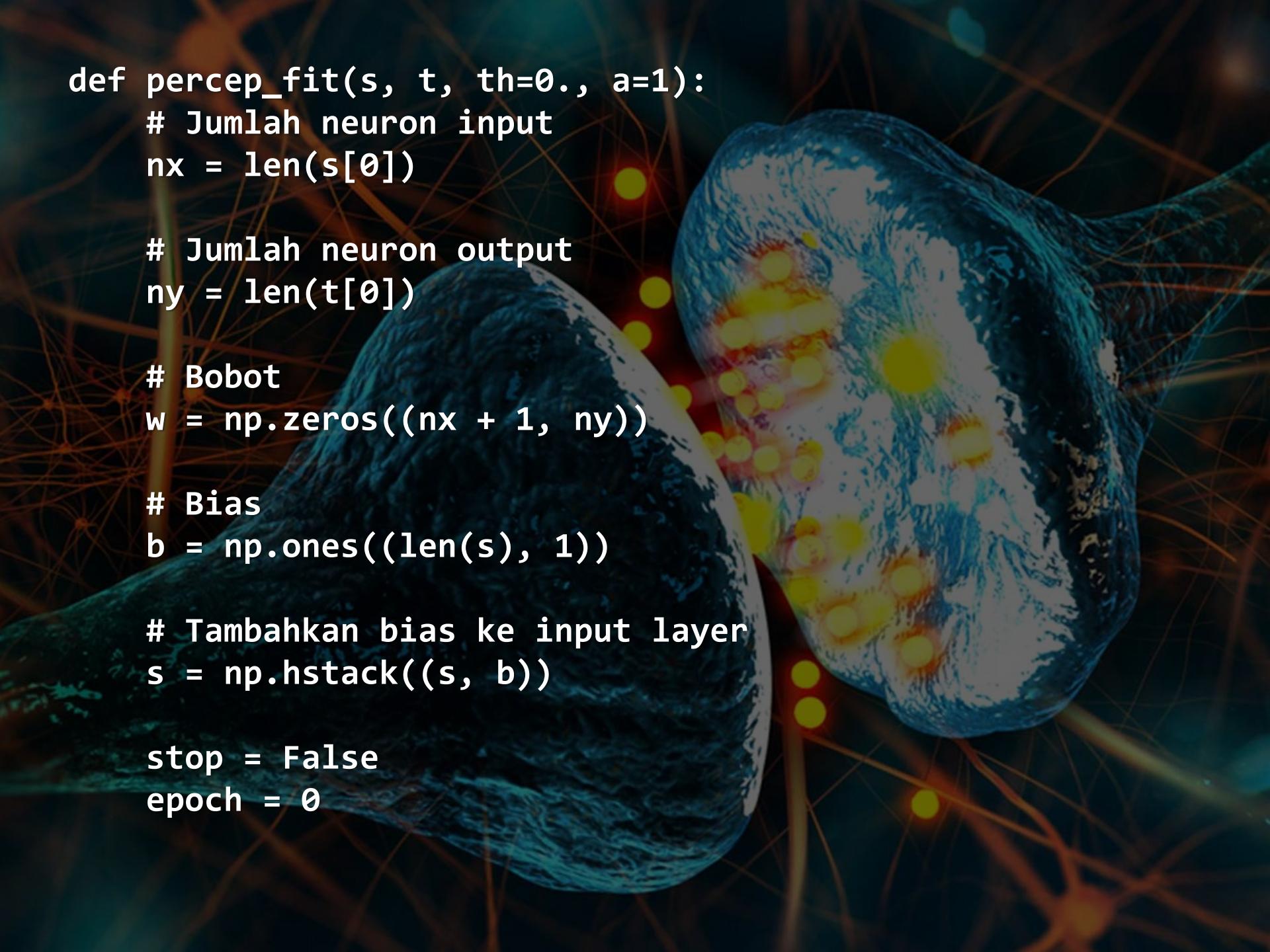
    # Jumlah neuron output
    ny = len(t[0])

    # Bobot
    w = np.zeros((nx + 1, ny))

    # Bias
    b = np.ones((len(s), 1))

    # Tambahkan bias ke input layer
    s = np.hstack((s, b))

    stop = False
    epoch = 0
```



```
while not stop:  
    stop = True  
    epoch += 1  
  
    print(f'Epoch: {epoch}')  
  
    for i, s_ in enumerate(s):  
        for j, t_ in enumerate(t[i]):  
  
            # Hitung yin menggunakan dot product  
            yin = np.dot(s_, w[:, j:j + 1])[0]  
  
            # Hitung nilai y  
            y = step(yin, th)  
  
            # Jika output tidak sama dengan target  
            if y != t_:  
                stop = False  
  
                # Ubah nilai bobot  
                dw = a * t_ * s_  
                w[:, j:j + 1] += dw.reshape(nx + 1, -1)  
  
return w, epoch
```

```
def percep_predict(X, w, th=0):
    # Loop setiap data
    for x in X:

        # Hitung nilai input dan nilai aktivasi
        y_in = np.dot([*x, 1], w)
        y = [step(i, th) for i in y_in]

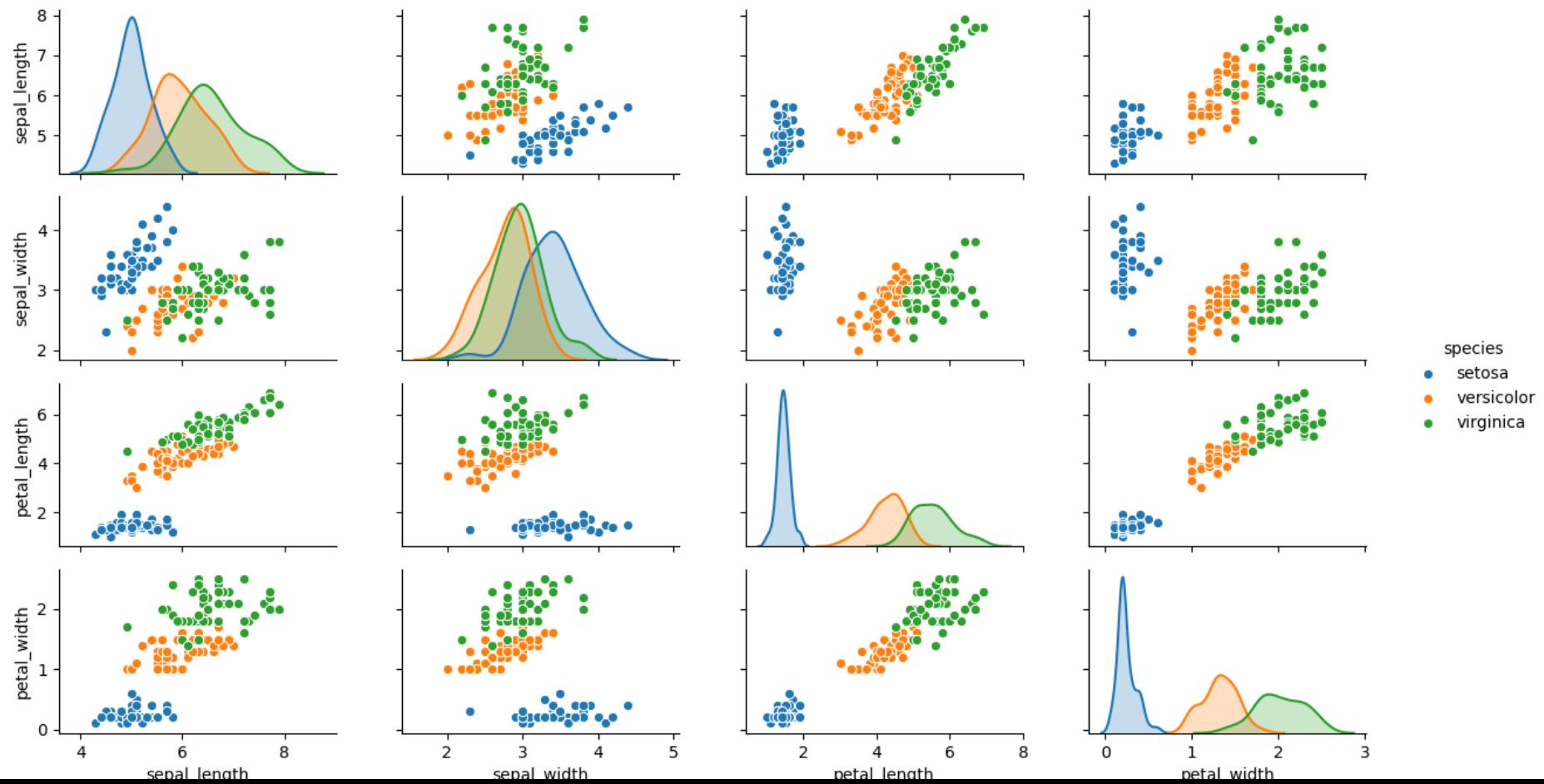
    yield y
```

# Load dan Plot Menggunakan Seaborn dan Pandas

```
# Instalasi: pip install seaborn
import seaborn as sns
import matplotlib.pyplot as plt

# Load dataset
iris = sns.load_dataset('iris')

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



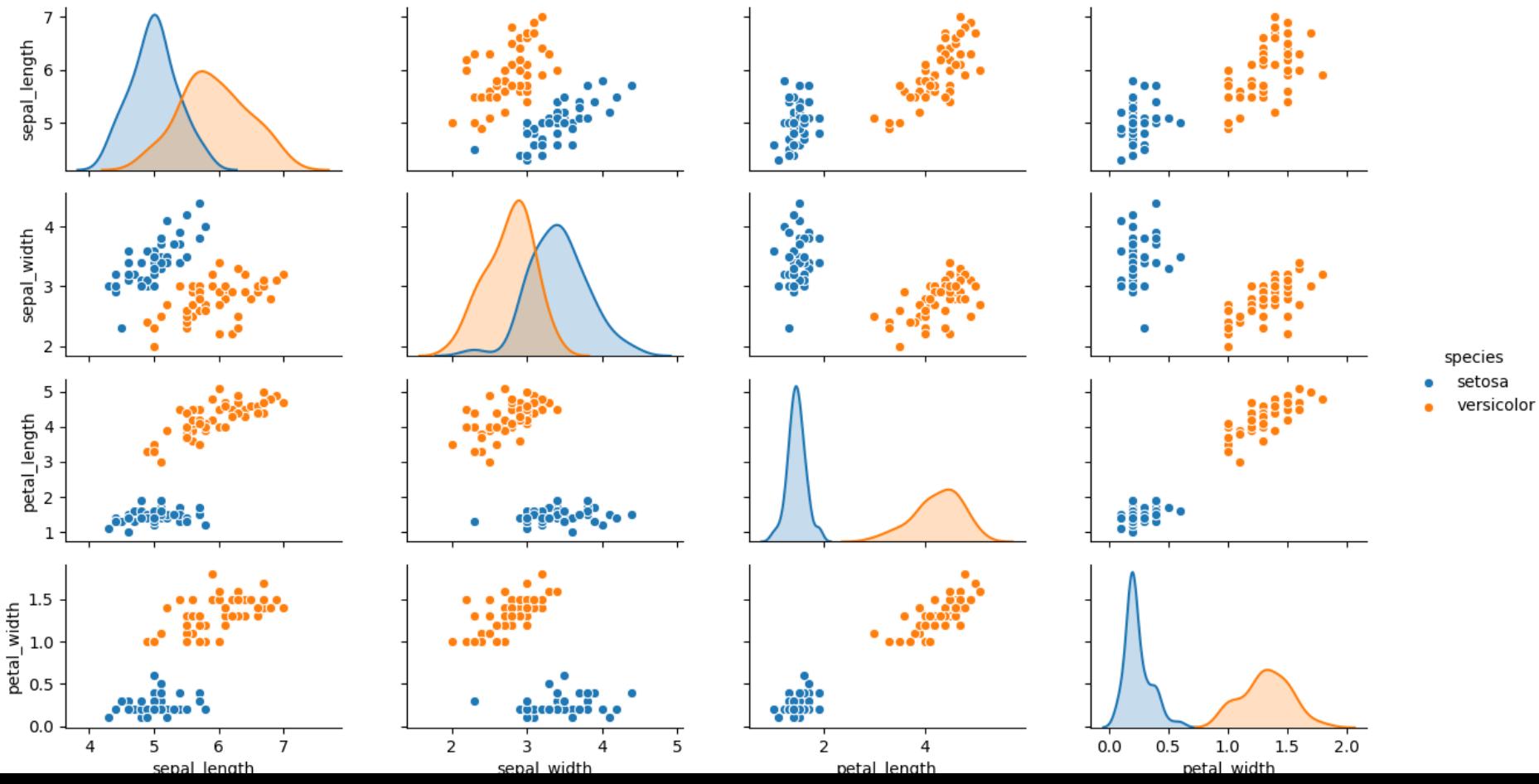
# Hapus Kelas Virginica

```
# Instalasi: pip install seaborn
import seaborn as sns
import matplotlib.pyplot as plt

# Load dataset
iris = sns.load_dataset('iris')

# Hapus kelas virginica
iris = iris.loc[iris['species'] != 'virginica']

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



# Hapus Ciri *Sepal Width* dan *Petal Width*

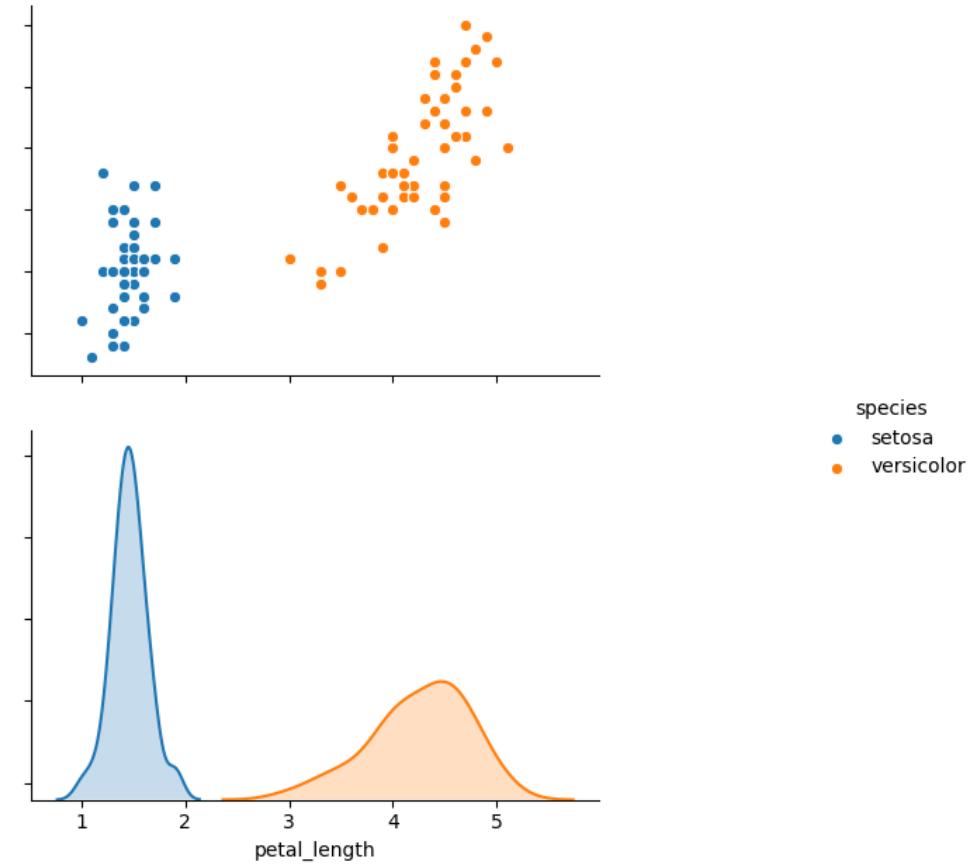
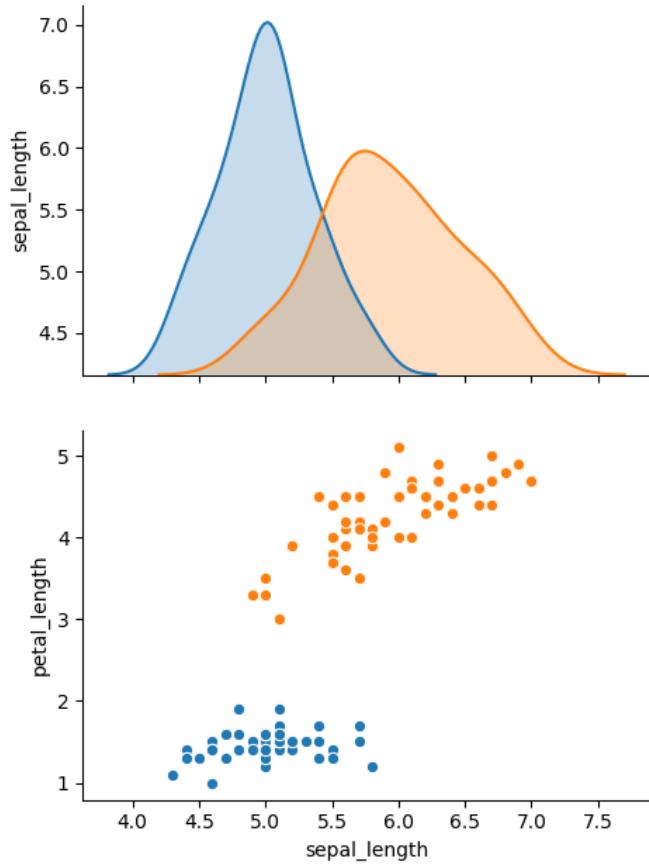
```
# Instalasi: pip install seaborn
import seaborn as sns
import matplotlib.pyplot as plt

# Load dataset
iris = sns.load_dataset('iris')

# Hapus kelas virginica
iris = iris.loc[iris['species'] != 'virginica']

# Hapus ciri sepal_width dan petal_width
iris = iris.drop(['sepal_width', 'petal_width'],
axis=1)

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



```
import seaborn as sns

# Instalasi: pip install scikit-learn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import minmax_scale
from sklearn.metrics import accuracy_score

iris = sns.load_dataset('iris')
iris = iris.loc[iris['species'] != 'virginica']
iris = iris.drop(['sepal_width', 'petal_width'], axis=1)

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

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

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=.3)
```

# *Training dan Testing*

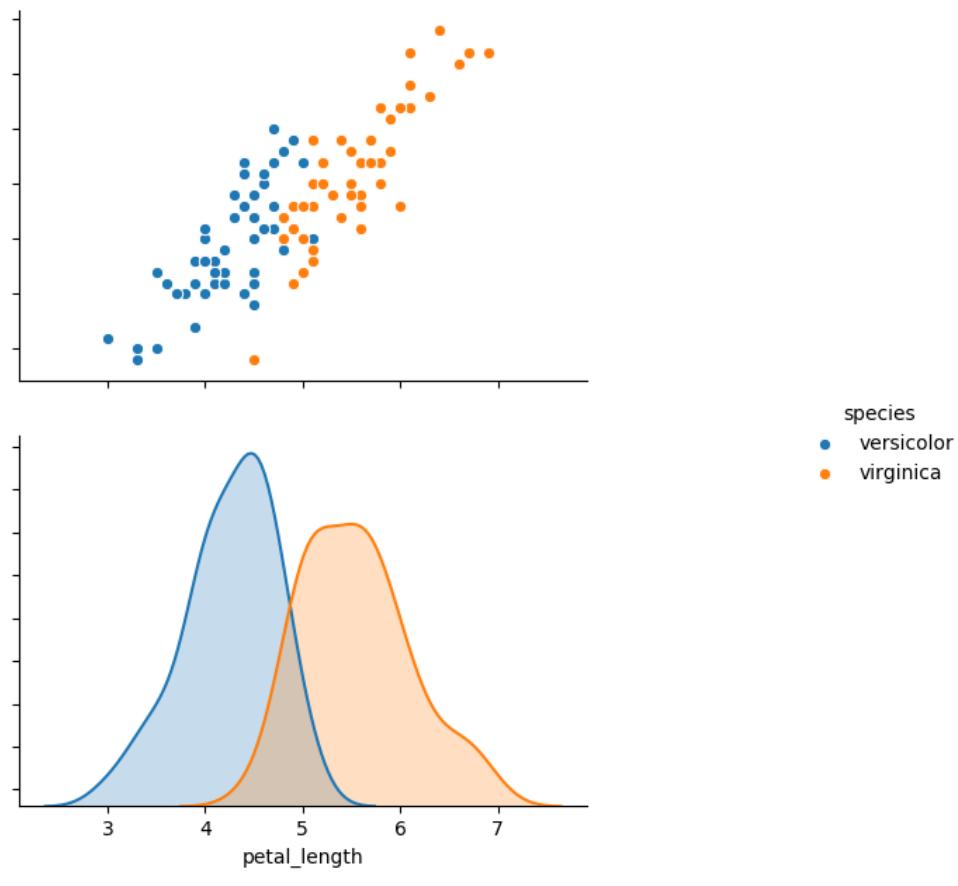
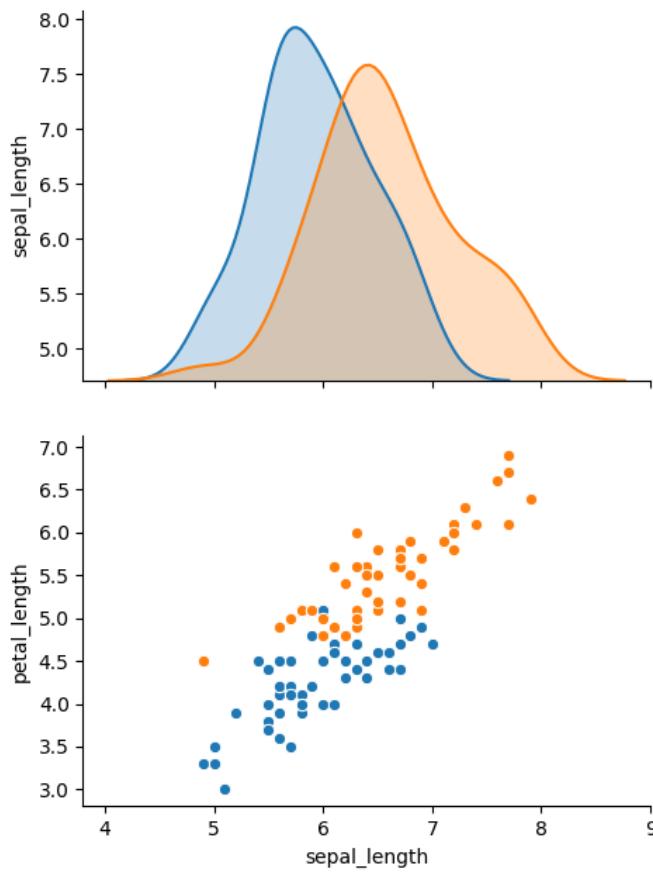
```
# Lakukan training  
w, epoch = percep_fit(X_train, y_train)  
  
# Fungsi untuk mengonversi menjadi urutan kelas (0, 1, 2)  
to_class = lambda x: [0 if i == [-1, -1] else 1 if i ==  
[-1, 1] else 2 for i in x]  
  
# Lakukan testing  
out = list(percep_predict(X_test, w))  
  
# Konversi menjadi urutan kelas (0, 1, 2)  
out = to_class(out)  
y_test = to_class(y_test)  
  
# Hitung akurasi  
acc = accuracy_score(out, y_test)  
  
print(f'Epoch: {epoch}')  
print(f'Accuracy: {acc}')
```

# Hapus Kelas Virginica

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

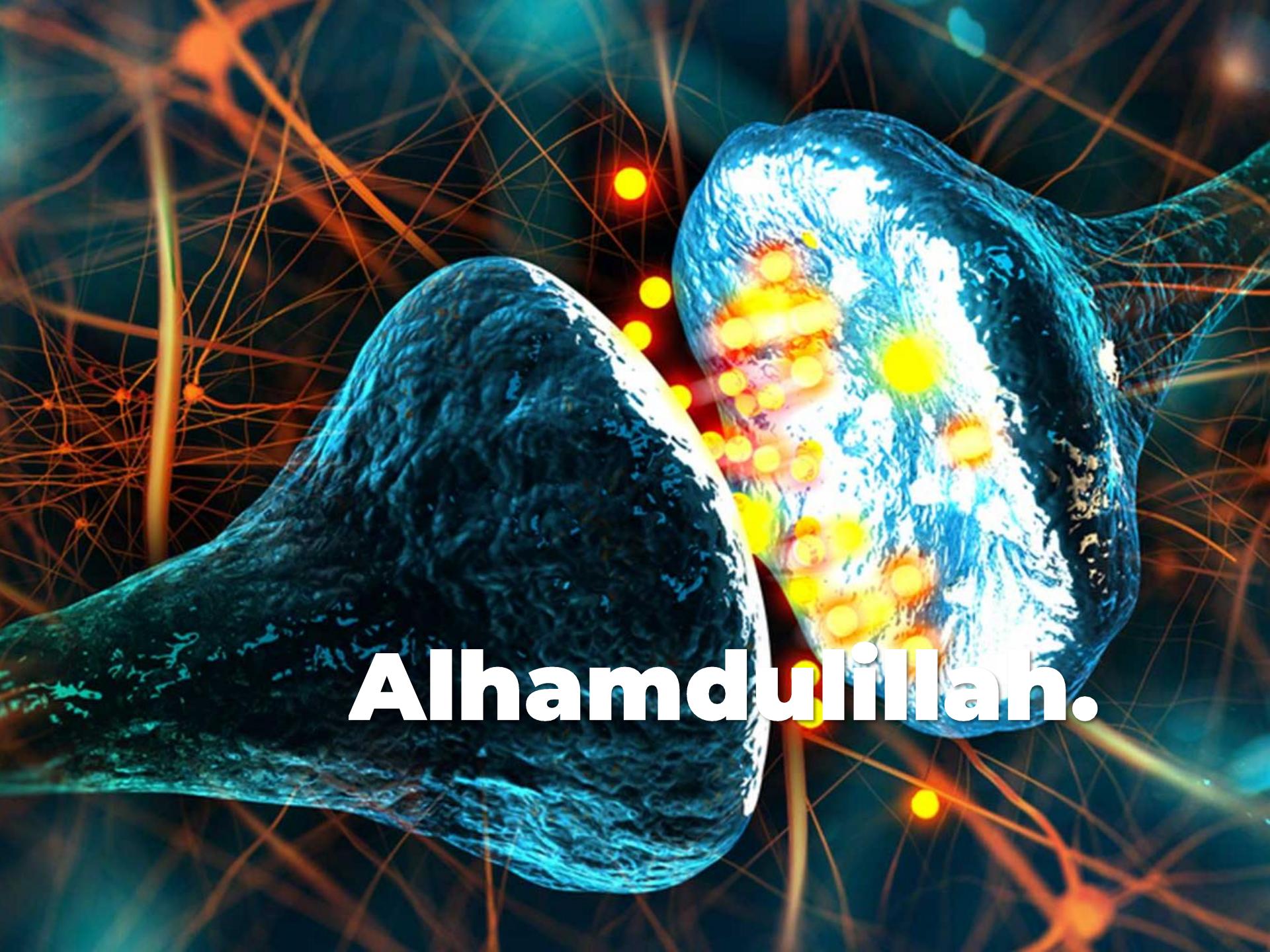
iris = sns.load_dataset('iris')
iris = iris.loc[iris['species'] != 'setosa']

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



# Referensi

- Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning*. Springer.
- Jordan, Jeremy. (2018). *Setting the learning rate of your neural network*.  
<https://www.jeremyjordan.me/nn-learning-rate>



**Alhamdulillah.**