Nama = Muhammad Syahroni
Nim = 691220142
gol = A

Neural Network

1. diketahui :
   $W_1 = 0$
   $W_2 = 0$
   b(bias) = 0
   learning rate = a = 1
   epoch = 3

data training

X_1	X_2	target
0	0	-1
0	1	-1
1	0	-1
1	1	1

Epoch 1

1. Input 1 (0,0) Target -1
   hitung $y_{in} = W_1 \cdot X_1 + W_2 \cdot X_2 + b$
   persamaan = 0.0+0.0+0 = 0
   Output y = 0 (karena $y_{in} = 0 \leq \theta$) tidak sesuai dengan target -1
   update bobot dan bias :
   $w_1 = w_1 + a \cdot t \cdot x_1 = 0+1 \cdot (-1) \cdot 0 = 0$
   $w_2 = w_2 + a \cdot t \cdot x_2 = 0+1 \cdot (-1) \cdot 0 = 0$
   $b = b + a \cdot t = 0+1 \cdot -1 = -1$
   bobot baru $W_1=0\ W_2=0\ b=-1$

Input 2 (0,1) target -1
hitung $y_{in} = W_1 \cdot X_1 + W_2 \cdot X_2 + b$
persamaan = 0.0+0.1+-1 = -1
Output y = -1 (karena $y_{in} < 0$) Tidak ada perubahan karena sesuai target
bobot tetap $W_1=0\ W_2=0\ b=-1$

Input 3 (1,0) target -1
hitung $y_{in} = W_1 \cdot X_1 + W_2 \cdot X_2 + b$ =

Persamaan $y_{in} = 0 \cdot 1 + 0 \cdot 0 + -1 = -1$
Output $y = -1$ (karena y kurang dari 0) sesuai target
bobot tetap : $W_1 = 0$ $W_2 = 0$ $b = -1$

Input 4 (1,1) Target 1
$y_{in} = W_1 \cdot X_1 + W_2 \cdot X_2 + b$
Persamaan $= 0 \cdot 1 + 0 \cdot 1 + -1 = -1$
Output $y = -1$ (karena y kurang dari 0) tidak sesuai target karena target 1

Update bobot dan bias
$W_1 = W_1 + a \cdot t \cdot X_1 = 0 + 1 \cdot 1 \cdot 1 = 1$
$W_2 = W_2 + a \cdot t \cdot X_2 = 0 + 1 \cdot 1 \cdot 1 = 1$
$b = b + a \cdot t = -1 + 1 \cdot 1 = 0$
bobo baru $= W_1 = 1$, $W_2 = 1$, $b = 0$

Epoch 2
1. Input 1 (0,0) Target -1
$y_{in} = W_1 \cdot X_1 + W_2 \cdot X_2 + b$
persamaan $= 1 \cdot 0 + 1 \cdot 0 + 0 = 0$
Output $y_{in} = 0$ tidak sesuai target -1
Update bobot dan bias
$W_1 = W_1 + a \cdot t \cdot X_1 = 1 + 1 \cdot -1 \cdot 0 = 1$
$W_2 = W_2 + a \cdot t \cdot X_2 = 1 + 1 \cdot -1 \cdot 0 = 1$
$b = b + a \cdot t = 0 + 1 \cdot -1 = -1$

bobot baru $W_1 = 1$ $W_2 = 1$ $b = -1$

2. Input 2 (0,1) Target -1
$y_{in} = W_1 \cdot X_1 + W_2 \cdot X_2 + b$
persamaan $= 1 \cdot 0 + 1 \cdot 1 + -1 = 0$
$y_{in} = 0$ tidak sesuai, Target -1
Update bobot dan bias
$W_1 = W_1 + a \cdot t \cdot X_1 = 1 + 1 \cdot -1 \cdot 0 = 1$
$W_2 = W_2 + a \cdot t \cdot X_2 = 1 + 1 \cdot -1 \cdot 1 = 0$
$b = b + a \cdot t = -1 + 1 \cdot -1 = -2$
bobot baru $W_1 = 1$ $W_2 = 0$ $b = -2$

sa

3. Input 3 (1,0) Target -1
yin = $W_1 \cdot x_1 + W_2 \cdot x_2 + b$
Persamaan = $1 \cdot 1 + 0 \cdot 0 + (-2) = 1 - 2 = -1$
Output yin = -1 tidak ada perubahan karena sesuai target.

4. Input 4 (1,1) Target 1
yin = $W_1 \cdot x_1 + W_2 \cdot x_2 + b$
Persamaan, $1 \cdot 1 + 0 \cdot 1 + -2 = 1 - 2 = -1$ perubahan karena tidak sesuai target yaitu 1
Update bobot dan bias
$W_1 = W_1 + a \cdot t \cdot x_1 = 1 + 1 \cdot 1 \cdot 1 = 1 + 1 = 2$
$W_2 = W_2 + a \cdot t \cdot x_2 = 0 + 1 \cdot 1 \cdot 1 = 1$
$b = b + a \cdot t = -2 + 1 \cdot 1 = -1$
bobot baru $W_1 = 2$ $W_2 = 1$ $b = -1$

Epoch 3

1. Input 1 (0,0) Target -1
yin = $W_1 \cdot x_1 + W_2 \cdot x_2 + b$
persamaan = $2 \cdot 0 + 1 \cdot 0 + -1 = -1$
Output yin = -1 tidak ada perubahan karena sesuai target

2. Input 2 (0,1) Target -1
hitung yin = $W_1 \cdot x_1 + W_2 \cdot x_2 + b$
Pmaan = $2 \cdot 0 + 1 \cdot 1 + -1 = 0$
output yin = 0 tidak sesuai target -1. Update bobot dan bias
$W_1 = W_1 + a \cdot t \cdot x_1 = 2 + 1 \cdot -1 \cdot 0 = 2$
$W_2 = W_2 + a \cdot t \cdot x_2 = 1 + 1 \cdot -1 \cdot 1 = 0$
$b = b + a \cdot t = -1 + 1 \cdot -1 = -2$
bobot baru $W_1 = 2$ $W_2 = 0$ $b = -2$

3. Input 3 (1,0) Target = -1
hitung yin = $W_1 \cdot x_1 + W_2 \cdot x_2 + b$
persamaan = $2 \cdot 1 + 0 \cdot 0 + -2 = 0$
output = 0 tidak sesuai target -1
Update bobot dan bias
$W_1 = W_1 + a \cdot t \cdot x_1 = 2 + 1 \cdot -1 \cdot 1 = 1$
$W_2 = W_2 + a \cdot t \cdot x_2 = 0 + 1 \cdot -1 \cdot 0 = 0$
$b = b + a \cdot t = -2 + 1 \cdot -1 = -3$
bobot baru $W_1 = 1$ $W_2 = 0$ $b = -3$

1. Input 1 (1,1) Target 1
masukan : $W_1 \cdot X_1 + W_2 \cdot X_2 + b$
Persamaan : $1 \cdot 1 + 0 \cdot 1 + -3 = 1 - 3 = -2$
Output yin = -1 (karena yin < 0) : tidak sesuai target = -1
Update bobot dan bias
$W_1 = W_1 + a \cdot t \cdot x_1 = 1 + 1 \cdot 1 \cdot 1 = 2$
$W_2 = W_2 + a \cdot t \cdot x_2 = 0 + 1 \cdot 1 \cdot 1 = 1$
$b = b + a \cdot t = -3 + 1 \cdot 1 = -2$
bobot baru $W_1 = 2$ $W_2 = 1$ $b = -2$ = jadi
bobot akhir $W_1 = 2$ $W_2 = 1$ $b = -2$

2. Data Uji

X_1	X_2	target
0	1	-1
1	0	-1
1	1	1
0	0	-1

1. Input 1 (0,1) Target -1
$W_1 \cdot X_1 + W_2 \cdot X_2 + b = 2 \cdot 0 + 1 \cdot 1 + -2 = -1$ ~~(Benar)~~ (benar)

2. Input 2 (1,0) Target -1
$2 \cdot 1 + 1 \cdot 0 + -2 = 0$ (salah seharusnya -1)

3. Input 3 (1,1) Target 1
$2 \cdot 1 + 1 \cdot 1 + -2 = 1$ (benar)

4. Input 4 (0,0) Target -1
$2 \cdot 0 + 1 \cdot 0 + -2 = -2$. output -1 (benar)

prediksi benar = 3
Total data = 4
$\frac{3}{4} \times 100 = 75\%$ Jadi akurasi pada data uji 75%

(SIDU)

Prediksi pada data latihan

1. Input 1 (0,0) target -1
   2.0 + 1.0 + -2 = -1 (benar)

2. Input 2 (0,1) Target -1
   2.0 + 1.1 + -2 = -1 (benar)

3 Input 3 (1,0) Target -1
   2.1 + 1.0 + -2 = 0 (salah seharusnya -1)

4. Input 4 (1,1) Target 1
   2.1 + 1.1 + -2 = 1 (benar)

Prediksi benar 3
Total data = 4

$\frac{3}{4} \times 100 = 75\ \%$

3- a. Akurasi pada data training = 75 %
   b. Akurasi pada data uji = 75 %
   c berhasil, dikarenakan ~~data training dan data uji~~ model mengklasifikan nilai yang sama ~~dengan~~ antara data uji dengan data training

Code

```
import matplotlib.pyplot as plt

def activation_function(y_in):
    """Fungsi Activasi tiga nilai."""
    if y_in > 0:
        return 1
    elif 0 <= y_in <= 1:
        return 0
    else:
        return -1

def train_perceptron(data, learning_rate, epochs):
    """Melatih perceptron dengan data yang diberikan"""
    weights = [0, 0]  # w1, w2
    bias = 0
    for epoch in range(epochs):
        print(f"Epoch {epoch + 1}")
        for x1, x2, target in data:
            # Hitung Input net
            y_in = bias + (weights[0] * x1 + weights[1] * x2)
            # Hitung Output
            output = activation_function(y_in)
            # Update bobot dan bias jika terjadi error
            if output != target:
                weights[0] += learning_rate * target * x1
                weights[1] += learning_rate * target * x2
                bias += learning_rate * target
                print(f"Updated weights: {weights}, bias: {bias}")
            else:
                print("No updates")
    return weights, bias

def calculate_accuracy(data, weights, bias):
    """Menghitung akurasi model."""
    correct_prediction = 0
    total_predictions = len(data)
```

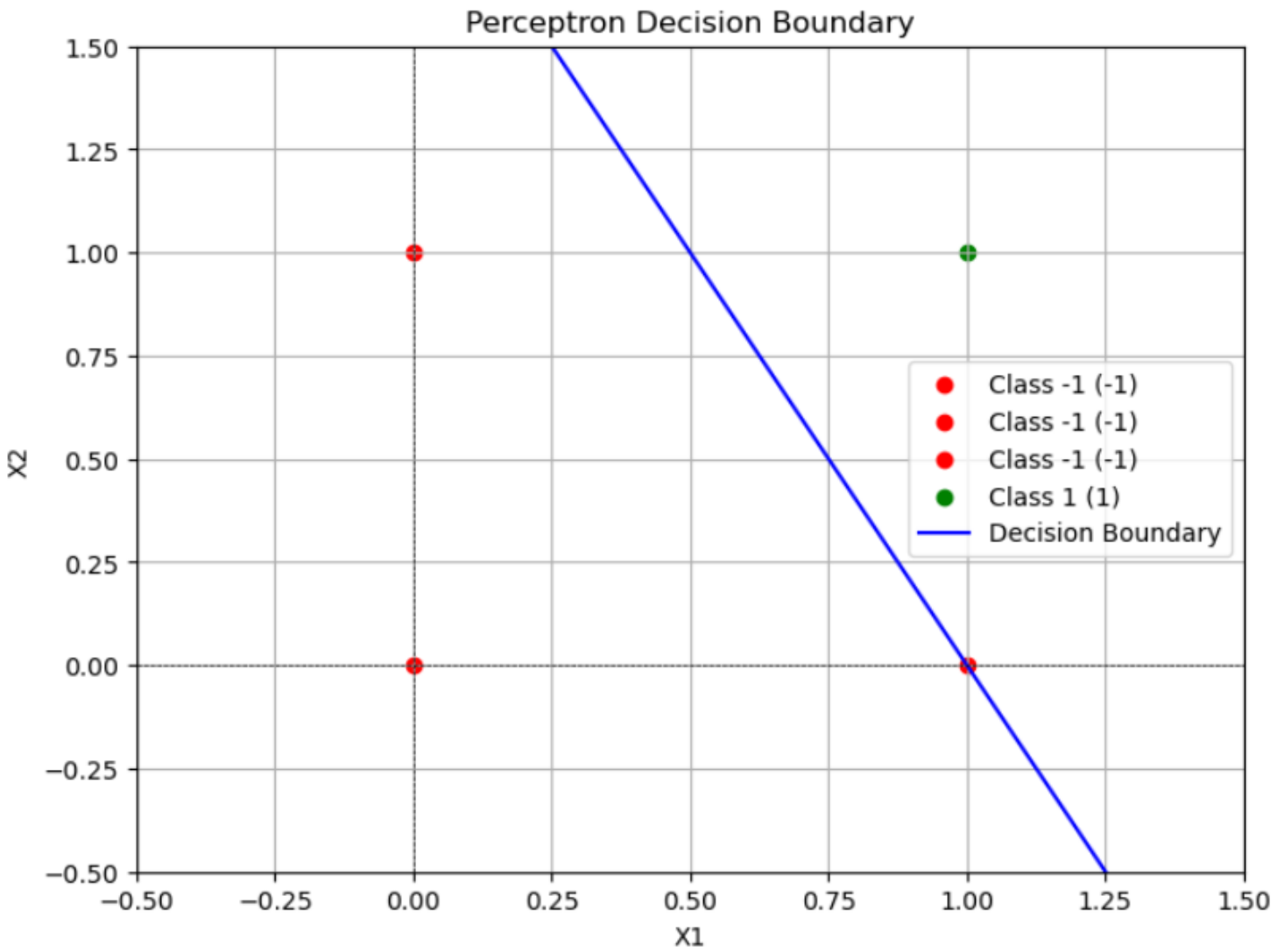
```
    for x1, x2, target in data:
        y_in = bias + (weights[0] * x1 + weights[1] * x2)
        output = activation_function(y_in)
        if output == target:
            correct_prediction += 1

    accuracy = (correct_prediction / total_predictions) * 100
    return accuracy

def predict(weights, bias, test_data):
    """Melakukan prediksi data uji"""
    predictions = []
    for x1, x2 in test_data:
        y_in = bias + (weights[0] * x1 + weights[1] * x2)
        output = activation_function(y_in)
        predictions.append(output)
    return predictions

def plot_decision_boundary(data, weights, bias):
    """Memvisualisasi data dan batas keputusan"""
    plt.figure(figsize=(8, 6))
    for x1, x2, target in data:
        if target == 1:
            plt.scatter(x1, x2, color='green', label='Class 1 (1)' if 'Class 1' not in plt.gca().get_legend_handles_labels()[1] else "")
        elif target == -1:
            plt.scatter(x1, x2, color='red', label='Class -1 (-1)' if 'Class -1' not in plt.gca().get_legend_handles_labels()[1] else "")

    # Garis pemisah
    x = [-0.5, 1.5]
    y = [-(weights[0] * xi + bias) / weights[1] if weights[1] != 0 else None for xi in x]
    plt.plot(x, y, label='Decision Boundary', color='blue')

    # Setting plot
    plt.xlim(-0.5, 1.5)
    plt.ylim(-0.5, 1.5)
    plt.xlabel('X1')
    plt.ylabel('X2')
    plt.title('Perceptron Decision Boundary')
    plt.axhline(0, color='black', linewidth=0.5, ls='--')
    plt.axvline(0, color='black', linewidth=0.5, ls='--')
    plt.grid()
```

```
    plt.legend()
    plt.show()


# Data input
data = [
    [0, 0, -1],
    [0, 1, -1],
    [1, 0, -1],
    [1, 1, 1]
]

# Parameter pelatihan
learning_rate = 1
epochs = 3

# Melatih perceptron
final_weights, final_bias = train_perceptron(data, learning_rate, epochs)

# Menampilkan bobot dan bias akhir
print(f"Final weights: {final_weights}, Final bias: {final_bias}")

# Menghitung akurasi
accuracy = calculate_accuracy(data, final_weights, final_bias)
print(f"Akurasi: {accuracy:.2f}%")

# Memvisualisasikan hasil
plot_decision_boundary(data, final_weights, final_bias)

# Data Uji
test_data = [
    [0, 1],
    [1, 0],
    [1, 1],
    [0, 0],
    [0.5, 0.5]  # Contoh data baru
]

# Melakukan prediksi
predictions = predict(final_weights, final_bias, test_data)
for i, (x1, x2) in enumerate(test_data):
    print(f"Prediksi untuk input ({x1}, {x2}): {predictions[i]}")
```

Hasil

```
Epoch 1
Updated weights: [0, 0], bias: -1
No updates
No updates
Updated weights: [1, 1], bias: 0
Epoch 2
Updated weights: [1, 1], bias: -1
Updated weights: [1, 0], bias: -2
No updates
Updated weights: [2, 1], bias: -1
Epoch 3
No updates
Updated weights: [2, 0], bias: -2
Updated weights: [1, 0], bias: -3
Updated weights: [2, 1], bias: -2
Final weights: [2, 1], Final bias: -2
Akurasi: 75.00%
```

```
Akurasi: 75.00%
```




```
Prediksi untuk input (0, 1): -1
Prediksi untuk input (1, 0): 0
Prediksi untuk input (1, 1): 1
Prediksi untuk input (0, 0): -1
Prediksi untuk input (0.5, 0.5): -1
```

Code

```
import numpy as np

# Fungsi aktivasi (step function)
def step_function(x):
    return np.where(x >= 0, 1, 0)

# Perceptron Training Function
def perceptron_train(X, y, lr, epochs):
    # Inisialisasi bobot dengan nilai random kecil
    weights = np.random.rand(X.shape[1])
    bias = np.random.rand()

    # Training loop
    for epoch in range(epochs):
        print(f"Epoch {epoch + 1}/{epochs}")
        for i in range(len(X)):
            # Hitung output
            linear_output = np.dot(X[i], weights) + bias
            prediction = step_function(linear_output)

            # Update bobot jika terjadi kesalahan
            error = y[i] - prediction
            weights += lr * error * X[i]
            bias += lr * error

        # Cetak bobot dan bias di setiap epoch
        print(f"Weights: {weights}, Bias: {bias}\n")

    return weights, bias

# Perceptron Prediction Function
def perceptron_predict(X, weights, bias):
    linear_output = np.dot(X, weights) + bias
    return step_function(linear_output)

# Input dari keyboard
lr = float(input("Masukkan nilai learning rate: "))
epochs = int(input("Masukkan jumlah epoch: "))

# Contoh data (X1, X2, X3)
X = np.array([
```

```
    [0, 0, 0],
    [0, 0, 1],
    [0, 1, 0],
    [0, 1, 1],
    [1, 0, 0],
    [1, 0, 1],
    [1, 1, 1]
])

# Target output (label dua kelas)
y = np.array([0, 0, 0, 1, 0, 1, 1])

# Training Perceptron
weights, bias = perceptron_train(X, y, lr, epochs)

# Prediksi menggunakan bobot yang sudah dilatih
print("\nPrediksi setelah training:")
for i in range(len(X)):
    prediction = perceptron_predict(X[i], weights, bias)
    print(f"Input: {X[i]} -> Kelas Data: {y[i]} -> Prediksi: {prediction}")

# Input untuk Data test baru dari keyboard
X_test1 = int(input("Masukkan nilai Xtest1: "))
X_test2 = int(input("Masukkan nilai Xtest2: "))
X_test3 = int(input("Masukkan nilai Xtest3: "))

# Gabungkan input test ke dalam satu array
X_test = np.array([X_test1, X_test2, X_test3])

# Prediksi untuk data tes
test_prediction = perceptron_predict(X_test, weights, bias)
print(f"\nPrediksi untuk input tes [{X_test1}, {X_test2}, {X_test3}] adalah: {test_prediction}")
```

Output

```
Masukkan nilai learning rate:  1
Masukkan jumlah epoch:  3
Epoch 1/3
Weights: [-0.55316793  1.2085659   1.56477491], Bias: -0.9998231154189561

Epoch 2/3
Weights: [-0.55316793  2.2085659   1.56477491], Bias: -0.9998231154189561

Epoch 3/3
Weights: [0.44683207 2.2085659  2.56477491], Bias: -0.9998231154189563


Prediksi setelah training:
Input: [0 0 0] -> Kelas Data: 0 -> Prediksi: 0
Input: [0 0 1] -> Kelas Data: 0 -> Prediksi: 1
Input: [0 1 0] -> Kelas Data: 0 -> Prediksi: 1
Input: [0 1 1] -> Kelas Data: 1 -> Prediksi: 1
Input: [1 0 0] -> Kelas Data: 0 -> Prediksi: 0
Input: [1 0 1] -> Kelas Data: 1 -> Prediksi: 1
Input: [1 1 1] -> Kelas Data: 1 -> Prediksi: 1
Masukkan nilai Xtest1:  1
Masukkan nilai Xtest2:  2
Masukkan nilai Xtest3:  3

Prediksi untuk input tes [1, 2, 3] adalah: 1
```

]: