Tugas 4

nama kelompok :

- Wafa bila syaefurokhman
- Rizky Chayono Putra

CPMK yang dicapai:

- CPMK 1: Mahasiswa mampu memahami metode yang digunakan dalam Kecerdasan Buatan
- CPMK 2: Mahasiswa mampu mengimplementasikan metode-metode dalam Kecerdasan Buatan dengan bahasa pemrograman tertentu

Deskripsi Tugas:

Tabel 1

TB	BB	JK
170	80	L
172	81	L
160	64	P
165	59	P
167	74	L
156	57	P

Kelompok (maksimal 3 orang) diminta untuk membuat program klasifikasi jenis kelamin berdasarkan data pada Tabel 1 menggunakan neural network sederhana. Program harus dapat menentukan bobot dan bias optimal menggunakan metode **gradient descent**.

Spesifikasi Tugas:

1. **Struktur Neural Network:**
    - **Input layer:** 2 node (Height, Weight).
    - **Hidden layer:** 2 node.
    - **Output layer:** 1 node (Gender: 1 untuk Male (M), 0 untuk Female (F)).
2. **Langkah-langkah Implementasi:**
    - Inisialisasi bobot (w1,w2,w3,w4,w5,w6) dan bias (b1,b2,b3) secara random.
    - Implementasikan fungsi aktivasi **sigmoid** untuk hidden dan output layer.

- Bangun fungsi feedforward untuk menghitung nilai output dari neural network.
- Implementasikan **backpropagation** untuk menghitung gradien bobot dan bias.
- Terapkan **gradient descent** untuk memperbarui bobot dan bias hingga mencapai konvergensi (error minimal).
- Tampilkan hasil bobot (w1,w2,w3,w4,w5,w6) dan bias (b1,b2,b3) setelah proses selesai.

3. Tidak boleh menggunakan library apapun kecuali numpy
4. **Pelaporan:**
   - Buat laporan yang menjelaskan:
     - Alur kerja program.
     - Implementasi feedforward dan backpropagation.
     - Hasil akhir nilai bobot, bias, dan prediksi model.
   - Sertakan kode program yang rapi dan terdokumentasi.

Output yang Harus Ditampilkan:

1. Nilai akhir dari w1,w2,w3,w4,w5,w6,b1,b2,b3.
2. Prediksi akhir model untuk setiap data pada tabel.
3. Visualisasi nilai loss selama iterasi.

Pengerjaan :

1. import library

[1]:
```
import numpy as np
```

Mengimpor library numpy yang digunakan untuk operasi matematika dan manipulasi array. Kita memberi alias np supaya lebih singkat saat menggunakan fungsi-fungsi dari numpy.

2. Implementasi Fungsi Aktivasi dan Loss Function

[2]:
```python
# Fungsi aktivasi sigmoid
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

# Derivatif dari fungsi sigmoid untuk backpropagation
def sigmoid_derivative(x):
    return x * (1 - x)

# Mean Squared Error Loss Function
def mse_loss(y_true, y_pred):
    return ((y_true - y_pred) ** 2).mean()
```

- **2.1**: **Fungsi sigmoid(x)**
  Digunakan untuk mengaktifkan output neuron menjadi nilai antara 0 dan 1, sangat cocok untuk klasifikasi biner. Formula:

$$\sigma(x) = \frac{1}{1 + e - x1}$$

- **2.2**: **Turunan dari fungsi sigmoid (sigmoid_derivative(x))**
  Digunakan dalam **backpropagation** untuk menghitung gradien error. Fungsi ini menghitung seberapa besar perubahan output terhadap inputnya. Formula:

$$\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x))$$

- **2.3**: **Fungsi Mean Squared Error (mse_loss(y_true, y_pred))**
  Fungsi ini digunakan untuk menghitung kesalahan antara nilai yang diprediksi dengan nilai yang sebenarnya. Semakin kecil nilai MSE, semakin baik prediksi model. Formula:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(ytrue, i - ypred, i)^2$$

3. Implementasi Neural Network Sederhana (SimpleNN)

[3]:
```python
class SimpleNN:
    def __init__(self, input_size, hidden_size, output_size):
        # Inisialisasi bobot dan bias secara random
        self.w1 = np.random.randn(input_size, hidden_size)  # Bobot input ke hidden
        self.b1 = np.random.randn(hidden_size)  # Bias hidden
        self.w2 = np.random.randn(hidden_size, output_size)  # Bobot hidden ke output
        self.b2 = np.random.randn(output_size)  # Bias output

    def feedforward(self, X):
        # Perhitungan hidden layer
        self.hidden_input = np.dot(X, self.w1) + self.b1
        self.hidden_output = sigmoid(self.hidden_input)

        # Perhitungan output layer
        self.output_input = np.dot(self.hidden_output, self.w2) + self.b2
        self.output = sigmoid(self.output_input)

        return self.output

    def backpropagate(self, X, y, learning_rate):
        # Menghitung error
        output_error = self.output - y
        output_delta = output_error * sigmoid_derivative(self.output)

        # Menghitung error di hidden layer
        hidden_error = output_delta.dot(self.w2.T)
        hidden_delta = hidden_error * sigmoid_derivative(self.hidden_output)

        # Update bobot dan bias
        self.w2 -= self.hidden_output.T.dot(output_delta) * learning_rate
        self.b2 -= np.sum(output_delta, axis=0) * learning_rate
        self.w1 -= X.T.dot(hidden_delta) * learning_rate
        self.b1 -= np.sum(hidden_delta, axis=0) * learning_rate

    def train(self, X, y, epochs, learning_rate):
        for epoch in range(epochs):
            # Feedforward
            self.feedforward(X)

            # Backpropagation
            self.backpropagate(X, y, learning_rate)

            # MSE Loss (untuk melihat progres training)
            if epoch % 100 == 0:
                loss = mse_loss(y, self.output)
                print(f"Epoch {epoch}, Loss: {loss}")
```

Pada tahap ini, kita membuat **class SimpleNN** untuk membangun dan melatih neural network sederhana dengan dua lapisan (hidden layer dan output layer). Berikut penjelasan masing-masing bagian:

3.1. Inisialisasi Bobot dan Bias (Constructor __init__)

Di dalam konstruktor, bobot dan bias untuk jaringan inisialisasi secara acak:

w1: Bobot yang menghubungkan input layer ke hidden layer.

b1: Bias untuk hidden layer.

w2: Bobot yang menghubungkan hidden layer ke output layer.

b2: Bias untuk output layer.

3.2. Fungsi Feedforward (Method feedforward)

Fungsi ini digunakan untuk menghitung output dari jaringan berdasarkan input yang diberikan:

- Hidden layer: Menghitung input dan output hidden layer menggunakan bobot dan bias yang ada, lalu melalui fungsi sigmoid.
- Output layer: Menghitung input dan output untuk output layer, lalu melalui fungsi sigmoid juga untuk mendapatkan prediksi akhir.

3.3. Fungsi Backpropagation (Method backpropagate)

Fungsi ini menghitung dan memperbarui bobot dan bias menggunakan metode backpropagation:

- Error di output: Menghitung error pada output dan mengalikan dengan turunan dari fungsi sigmoid untuk mendapatkan delta output.
- Error di hidden layer: Error pada hidden layer dihitung dengan propagasi mundur dari delta output dan bobot.
- Update bobot dan bias: Bobot dan bias diperbarui dengan cara mengurangi nilai gradien yang dihitung, dikalikan dengan learning rate.

3.4. Fungsi Training (Method train)

Fungsi ini digunakan untuk melatih model menggunakan data input X dan target output y selama sejumlah epoch:

- Di setiap epoch, dilakukan feedforward untuk menghitung output dan backpropagation untuk memperbarui bobot dan bias.
- MSE Loss: Di setiap 100 epoch, fungsi ini menampilkan Mean Squared Error (MSE) sebagai indikator progres training.

4. Menyiapkan Data dan Melatih Model

[4]:
```
# Data input sesuai Tabel 1
X = np.array([[170, 80], [172, 81], [160, 64], [155, 59], [167, 74], [156, 57]])  # Height, Weight
y = np.array([[1], [1], [0], [0], [1], [0]])  # 1 untuk Male (L), 0 untuk Female (P)

# Membuat instance neural network
nn = SimpleNN(input_size=2, hidden_size=2, output_size=1)

# Training model
nn.train(X, y, epochs=1000, learning_rate=0.1)
```

```
Epoch 0, Loss: 0.30697579238293593
Epoch 100, Loss: 0.2500000000045637
Epoch 200, Loss: 0.24999999999999997
Epoch 300, Loss: 0.24999999999999992
Epoch 400, Loss: 0.24999999999999992
Epoch 500, Loss: 0.24999999999999992
Epoch 600, Loss: 0.24999999999999992
Epoch 700, Loss: 0.24999999999999992
Epoch 800, Loss: 0.24999999999999992
Epoch 900, Loss: 0.24999999999999992
```

Pada tahap ini, kita menyiapkan data input sesuai dengan Tabel 1 dan melatih model neural network menggunakan data tersebut. Berikut penjelasan untuk setiap langkah:

4.1. Menyiapkan Data Input (X dan y)

Data input X adalah informasi tinggi badan (Height) dan berat badan (Weight) yang diambil dari Tabel 1. Data target y adalah label jenis kelamin (1 untuk Male (L), 0 untuk Female (P)).

- **X**: Data input yang terdiri dari tinggi badan dan berat badan.

```
X = np.array([[170, 80], [172, 81], [160, 64], [155, 59], [167, 74], [156, 57]])  # Height, Weight
```

- **y**: Data target untuk jenis kelamin, dengan 1 untuk Male (L) dan 0 untuk Female (P).

```
y = np.array([[1], [1], [0], [0], [1], [0]])  # 1 untuk Male (L), 0 untuk Female (P)
```

4.2. Membuat Instance Neural Network (nn)

Membuat objek dari class SimpleNN dengan jumlah input layer 2 (Height, Weight), hidden layer 2, dan output layer 1 (Male/Female).

```
# Membuat instance neural network
nn = SimpleNN(input_size=2, hidden_size=2, output_size=1)
```

4.3. Melatih Model (Training)

Model dilatih dengan menggunakan data input X, target y, jumlah **epoch** (iterasi pelatihan) sebanyak 1000, dan **learning rate** sebesar 0.1.
Proses training ini akan dilakukan melalui fungsi train, yang memanggil feedforward dan backpropagation untuk memperbarui bobot dan bias di setiap epoch.

```
# Training model
nn.train(X, y, epochs=1000, learning_rate=0.1)
```

5. Menampilkan Hasil Akhir Bobot, Bias, dan Prediksi Model.

[5]:
```
# Menampilkan hasil akhir bobot dan bias
print("\nFinal Weights and Biases:")
print("w1:", nn.w1)
print("b1:", nn.b1)
print("w2:", nn.w2)
print("b2:", nn.b2)

# Menampilkan prediksi akhir model untuk setiap data
predictions = nn.feedforward(X)
print("\nPredictions for the input data:")
print(predictions)
```

```
Final Weights and Biases:
w1: [[ 0.42361946  2.16495572]
 [-0.44771423  0.93812562]]
b1: [-1.15077244  0.08832731]
w2: [[-0.41425858]
 [ 0.65836486]]
b2: [-0.24410629]

Predictions for the input data:
[[0.5]
 [0.5]
 [0.5]
 [0.5]
 [0.5]
 [0.5]]
```

Pada tahap terakhir ini, kita menampilkan hasil akhir dari bobot dan bias yang telah dioptimalkan selama proses pelatihan, serta memeriksa hasil prediksi model terhadap data input yang telah diberikan. Berikut penjelasannya:

5.1. Menampilkan Hasil Akhir Bobot dan Bias

Setelah model dilatih, kita dapat melihat nilai akhir dari bobot dan bias untuk setiap lapisan. Bobot dan bias ini adalah hasil dari pembaruan selama proses backpropagation dengan menggunakan **gradient descent**.

```
# Menampilkan hasil akhir bobot dan bias
print("\nFinal Weights and Biases:")
print("w1:", nn.w1)
print("b1:", nn.b1)
print("w2:", nn.w2)
print("b2:", nn.b2)
```

Dengan menampilkan nilai-nilai ini, kita dapat memeriksa bagaimana bobot dan bias telah disesuaikan untuk meminimalkan error.

5.2. Menampilkan Prediksi Akhir Model untuk Setiap Data

Setelah model selesai dilatih, kita menggunakan fungsi feedforward untuk mendapatkan prediksi akhir berdasarkan data input X. Prediksi ini berupa nilai antara 0 dan 1, yang akan diklasifikasikan sebagai **Male (1)** atau **Female (0)**.

```
# Menampilkan prediksi akhir model untuk setiap data
predictions = nn.feedforward(X)
print("\nPredictions for the input data:")
print(predictions)
```

Dengan memeriksa hasil prediksi ini, kita dapat mengevaluasi sejauh mana model dapat memprediksi jenis kelamin berdasarkan tinggi badan dan berat badan.

Link kaggle :

https://www.kaggle.com/code/loliwibu/tugas-4