Praktikum 5
Madaline

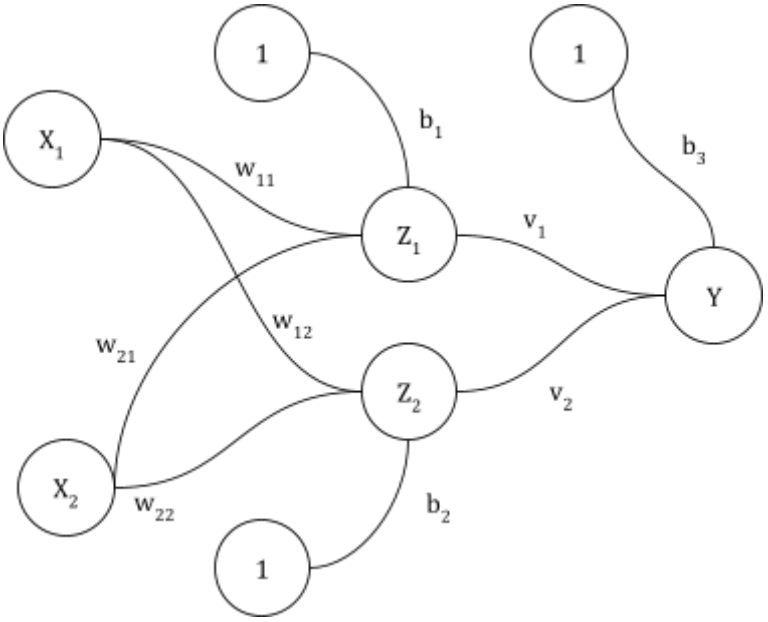
Tujuan

1. Praktikan mampu memahami algoritma Madaline
2. Praktikan mampu mengaplikasikan algoritma Madaline untuk melakukan klasifikasi biner sederhana

Dasar Teori

Madaline

Jaringan Madaline (MAny ADAptive Linear NEuron) merupakan sebuah arsitektur jaringan saraf tiruan yang tersusun dari sekumpulan Adaline. Berbeda dengan Adaline, Madaline merupakan sebuah jaringan multilayer yang terdiri dari input, hidden, dan output layer. Gambar 5.1 menunjukkan sebuah contoh arsitektur Madaline yang mampu menyelesaikan permasalahan logika biner.



Gambar 5.1 Madaline dengan dua *input neuron,* dua *hidden neuron*, dan satu output neuron.

Algoritma Pelatihan MRI

Jaringan Madaline lebih rumit karena merupakan jaringan *multilayer*. Oleh karena itu, algoritma pelatihannya juga lebih rumit. Terdapat berbagai variasi algoritma pelatihan Madaline dan algoritma versi pertama disebut dengan algoritma MRI (Madaline Rule 1). Algoritma MRI hanya mengubah bobot dan bias pada *hidden layer* saja, sedangkan bobot dan bias pada *output layer* dibuat konstan.

Pada arsitektur yang digambarkan di Gambar 5.1, nilai v_1 , v_2 , b_3 diset secara manual sehingga output dari Y bernilai -1 jika output dari z_1 dan z_2 bernilai -1 dan output dari Y bernilai 1 jika output salah satu (atau kedua) z_1 dan z_2 bernilai 1. Salah satu kombinasi yang sesuai adalah v_1 =0,5 , v_2 =0,5 , dan b_3 =0,5. Selanjutnya, bobot dan bias pada hidden layer ditentukan melalui proses pelatihan sebagai berikut:

1. Inisialisasi bobot dan bias pada hidden layer menggunakan nilai acak antara 0 dan 1. Nilai α berada pada rentang 0 dan 1. Fungsi aktivasi yang digunakan adalah fungsi step biner.
2. Selama kondisi berhenti bernilai salah, lakukan langkah 2-8
3. Untuk setiap pasangan input dan target (x:t) pada data latih, lakukan langkah 4-7
4. Hitung input pada setiap hidden neuron

$$z_{in1} = b_1 + x_1 w_{11} + x_2 w_{21}$$
$$z_{in2} = b_2 + x_1 w_{12} + x_2 w_{22}$$

5. Tentukan nilai output pada setiap *hidden neuron*

$$z_1 = f(z_{in1})$$
$$z_2 = f(z_{in2})$$

6. Hitung output dari Madaline

$$y_{in} = b_3 + z_1 v_1 + z_2 v_2$$
$$y = f(y_{in})$$

7. Hitung error dan update bobot. Update bobot tidak dilakukan jika $t = y$
   - Jika t=1, update bobot pada z_j yaitu *hidden neuron* dengan input yang paling mendekati 0

$$b_j = b_j + \alpha(1 - z_{in\,j})$$
$$w_{ij} = w_{ij} + \alpha(1 - z_{in\,j})\, x_i$$

- Jika t=-1, update bobot pada z_k, yaitu semua hidden neuron yang mempunyai input positif

$$b_k = b_k + \alpha(-1 - z_{in\,k})$$
$$w_{ik} = w_{ik} + \alpha(-1 - z_{in\,k})\, x_i$$

8. Setiap 1 epoch, test kondisi berhenti. Jika perubahan bobot telah berhenti (atau lebih kecil dari suatu ambang batas), atau jika iterasi telah melampaui jumlah epoch tertentu, iterasi dihentikan

Praktikum

1. Buka Google Colaboratory melalui tautan ini.
2. Tulis kode berikut ke setiap cell pada notebook tersebut.
   a. Import modul

```
import numpy as np
```

   b. Fungsi aktivasi

```
def aktivasi(x):
  if x < 0:
      return -1
  else:
      return 1
```

   c. Fungsi *training* Madaline

```
def train(train_data,train_target,alpha=0.1,max_epoch=10):
        w = np.random.random((2,2))
        v = np.array([0.5,0.5])
        b = np.random.random(2)
        b = np.append(b,0.5)
        epoch = 0
        v_aktivasi = np.vectorize(aktivasi)
        weight_updated = True
        while weight_updated == True and epoch < max_epoch:
        weight_updated = False
```

```
        for data,target in zip(train_data,train_target):
        z_in = np.dot(data,w)
        z_in = z_in + b[:-1]
        z = v_aktivasi(z_in)
        y_in = np.dot(z,v) + b[-1]
        y = v_aktivasi(y_in)
        if y!= target:
        weight_updated = True
        if target == 1:
                index = np.argmin(np.abs(z_in))
                b[index] = b[index] + alpha * (1 - z_in[index])
                w[:, index] = w[:, index] + alpha * (1 - z_in[index])*data
        elif target == -1:
                index = np.where(z_in>0)[0]
                if len(index)==1:
                index = index[0]
                b[index] = b[index] + alpha * (-1 - z_in[index])
                w[:, index] = w[:, index] + alpha * (-1 - z_in[index]) * data
        epoch = epoch +1
        return (w,v,b)
```

d. Fungsi *testing* Madaline

```
def test(w,v,b,test_data):
        v_aktivasi = np.vectorize(aktivasi)
        z_in = np.dot(test_data, w)
        z_in = z_in + b[:-1]
        z = v_aktivasi(z_in)
        y_in = np.dot(z, v) + b[-1]
        y = v_aktivasi(y_in)
        return y
```

e. 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)
```

f. Logika AND

```
target = np.array([1,-1,-1,-1])
(w,v,b) = train(data,target,alpha=0.8,max_epoch=10)
output = test(w,v,b,data)
accuracy = calc_accuracy(output, target)


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

g. Logika OR

```
data = np.array([[1,1],[1,-1],[-1,1],[-1,-1]])
target = np.array([1,1,1,-1])
(w,v,b) = train(data,target,alpha=0.2,max_epoch=10)
output = test(w,v,b,data)
accuracy = calc_accuracy(output, target)


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

Analisis

1. Berdasarkan source code yang ada, kapan proses training pada Madaline akan berhenti?
2. Cobalah mengganti nilai alpha pada logika AND dengan rentang nilai 0,1 - 1.0. Apakah ada pengaruh alpha terhadap akurasi?
3. Apakah jaringan Madaline dapat mengenali logika XOR dengan tepat? Mengapa demikian?
4. Ubahlah data dan target pada Logika OR menggunakan bilangan biner (bukan bipolar). Jangan lupa mengubah pula fungsi aktivasi agar menghasilkan bilangan biner. Apakah Madaline dapat mengenali Logika OR dengan data dan target biner? Mengapa demikian?