LAPORAN TUGAS

Jaringan Syaraf Tiruan

"Implementasi Algoritma Perceptron, Hebb Net, dan Adaline"

NAMA :

RIDHO NUR ROHMAN WIJAYA

06111840000065

DEPARTEMEN MATEMATIKA

FAKULTAS MATEMATIKA KOMPUTASI DAN SAINS DATA

INSTITUT TEKNOLOGI SEPULUH NOPEMBER

SURABAYA

2021

A. SOAL

Pada tugas ini akan ditampilkan penerapan algoritma dari soal yang ada.

1. Deskripsi Soal

Implementasikan algortima Perceptron, Hebb Net, dan Adaline dengan input dan target bipolar menggunakan fungsi NAND, AND NOT, OR, NOR, XOR.

2. Metode penyelesaian

Penyelesaian pada masalah ini dapat diselesaikan dengan langkah-langkah sebagai berikut:

1) Tentukan terlebih dahulu semua variabel yang diperlukan
2) Tentukan semua target yang ditanyakan
3) Buat function untuk setiap algoritma
4) Hitung semua hasil dari semua algoritma
5) Tampilkan komparasi hasil dari semua algoritma
6) Visualisasikan komparasi hasil dari semua algoritma

Pada laporan ini akan diimplementasikan dengan menggunakan bahasa pemrograman Python.

3. Jawaban Masalah

Percobaan dilakukan secara langsung dengan membangkitkan bilangan acak yang sesuai untuk setiap algoritma yang ada. Sebagai contoh untuk fungsi NAND dengan learning rate 0.15, threshold 0.12, dan tolerasi = 0.05 maka akan dihasilkan hasil sebagai berikut:

Algoritma	w_1	w_2	b
Perceptron	-0.15	-0.15	0.15
Hebb Net	-2	-2	2
Adaline	-0.25	-0.20	0.29

Sehingga akan menghasilkan garis lurus sebagai berikut:

a) Algoritma Perceptron

$$b + \sum_{i=1}^{n} w_i x_i = 0$$

$$0.15 - 0.15x_1 - 0.15x_2 = 0$$

$$x_1 + x_2 = 1$$

b) Algoritma Hebb Net

$$b + \sum_{i=1}^{n} w_i x_i = 0$$

$$2 - 2x_1 - 2x_2 = 0$$

$$x_1 + x_2 = 1$$

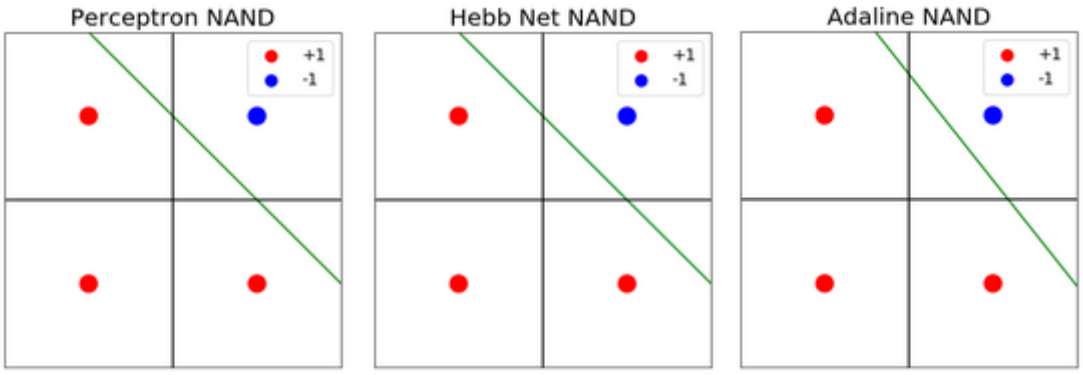
c) Algoritma Adaline

$$b + \sum_{i=1}^{n} w_i x_i = 0$$

$$0.29 - 0.25x_1 - 0.20x_2 = 0$$

$$25x_1 + 20x_2 = 29$$

serta jika di visualisasikan akan menghasilkan grafik sebagai berikut:



Secara keseluruhan, source code program dari setiap algoritma akan ada pada bab Source Code dan hasil training semua algoritma akan ditampilkan pada bab Running Program.

B. SOURCE CODE

Program penyelesaian masalah tersebut dapat di implementasikan pada source code berikut ini:

Tugas1PerceptronHebbNetAdaline.py

```
1.  # Mengimport library yang diperlukan
2.  import numpy as np
3.  import matplotlib.pyplot as plt
4.
5.  # Inisialisasi input bipolar
6.  xxb_input = np.array([[1,1,1],
7.                       [1,-1,1],
8.                       [-1,1,1],
9.                       [-1,-1,1]])
10.
11. # Inisialisasi learning rate
12. learning_rate = np.random.uniform(0.1,0.2)
13.
14. # Inisialisasi threshold
15. threshold = np.random.uniform(0,0.25)
16.
17. # Inisialisasi toleransi
18. tol = 0.05
19.
20.
21. # Inisialisasi logika NAND dengan target bipolar
22. target_nand = np.array([-1,1,1,1])
23.
24. # Inisialisasi logika AND NOT dengan target bipolar
25. target_andnot = np.array([-1,1,-1,-1])
26.
27. # Inisialisasi logika OR dengan target bipolar
28. target_or = np.array([1,1,1,-1])
29.
30. # Inisialisasi logika NOR dengan target bipolar
31. target_nor = np.array([-1,-1,-1,1])
32.
33. # Inisialisasi logika XOR dengan target bipolar
34. target_xor = np.array([-1,1,1,-1])
35.
36.
37.
38. # Function untuk metode Perceptron
39. def PerceptronMethod(xxb,t,alpha,threshold):
40. #     Inisialisasi weights dan bias serta wadah weight changes
41.     wwb = np.zeros(xxb.shape[1])
42.     weight_changes = np.ones(xxb.shape)
43.     max_iteration = 1
44.
```

```
    while not np.array_equal(weight_changes,np.zeros(xxb.shape))
and max_iteration<100:
        for i in range(len(t)):
            y_in = sum(xxb[i,:]*wwb)
            theta = threshold
            if y_in > theta:
                y = 1
            elif y_in < -theta:
                y = -1
            else:
                y = 0
            temp = wwb
            if y != t[i]:
                wwb = wwb + alpha*t[i]*xxb[i,:]
            weight_changes[i,:] = wwb-temp
        max_iteration += 1
    return wwb


# Function untuk metode Hebb Net
def HebbNetMethod(xxb,y):
#     Inisialisasi weights dan bias
    wwb = np.zeros(xxb.shape[1])

    for i in range(len(y)):
        wwb = wwb + xxb[i,:]*y[i]
    return wwb


# Function untuk metode Adaline
def AdalineMethod(xxb,t,alpha,tol):
#     Inisialisasi weight dan bias
    wwb = np.zeros(xxb.shape[1])
    delta = np.full(xxb.shape[1],tol+1)
    max_iteration = 1

    while max(delta[:-1])>tol and max_iteration<=100:
        for i in range(len(t)):
            y_in = sum(xxb[i,:]*wwb)
            delta = alpha*(t[i]-y_in)*xxb[i,:]
            wwb = wwb + delta
        max_iteration += 1
    if not (np.array_equal(t,TestingAdaline(xxb,wwb))):
        wwb = np.zeros(xxb.shape[1])
    return wwb

# Function untuk testing Adaline
def TestingAdaline(xxb,wwb):
    y = np.zeros(xxb.shape[0])

    for i in range(xxb.shape[0]):
        y_in = sum(xxb[i,:]*wwb)
        if y_in >=0:
            y[i] = 1
```

```
98.             else:
99.                 y[i] = -1
100.     return y
101.
102.
103. # Hasil keluaran dengan metode Perceptron
104. perceptron_result = {
105.     0 : PerceptronMethod(xxb_input,target_nand,learning_rate,threshold), # Perceptron dengan logika AND
106.     1 : PerceptronMethod(xxb_input,target_andnot,learning_rate,threshold), # Perceptron dengan logika AND NOT
107.     2 : PerceptronMethod(xxb_input,target_or,learning_rate,threshold), # Perceptron dengan logika OR
108.     3 : PerceptronMethod(xxb_input,target_nor,learning_rate,threshold), # Perceptron dengan logika NOR
109.     4 : PerceptronMethod(xxb_input,target_xor,learning_rate,threshold) # Perceptron dengan logika XOR
110. }
111.
112. # Hasil keluaran dengan metode Hebb Nett
113. hebbnet_result = {
114.     0 : HebbNetMethod(xxb_input,target_nand), # Hebb Net dengan logika AND
115.     1 : HebbNetMethod(xxb_input,target_andnot), # Hebb Net dengan logika AND NOT
116.     2 : HebbNetMethod(xxb_input,target_or), # Hebb Net dengan logika OR
117.     3 : HebbNetMethod(xxb_input,target_nor), # Hebb Net dengan logika NOR
118.     4 : HebbNetMethod(xxb_input,target_xor) # Hebb Net dengan logika XOR
119. }
120.
121. # Hasil keluaran dengan metode Adaline
122. adaline_result = {
123.     0 : AdalineMethod(xxb_input,target_nand,learning_rate,tol), # Adaline dengan logika AND
124.     1 : AdalineMethod(xxb_input,target_andnot,learning_rate,tol), # Adaline dengan logika AND NOT
125.     2 : AdalineMethod(xxb_input,target_or,learning_rate,tol), # Adaline dengan logika OR
126.     3 : AdalineMethod(xxb_input,target_nor,learning_rate,tol), # Adaline dengan logika NOR
127.     4 : AdalineMethod(xxb_input,target_xor,learning_rate,tol), # Adaline dengan logika XOR
128. }
129.
130.
131. # Komparasi hasil keluaran dengan metode Perceptron, Hebb Net, dan Adaline
```

```
132.
133. # Membuat header tabel
134. print("Metode Perceptron".center(40),end="")
135. print("Metode Hebb Net".center(40),end="")
136. print("Metode Adaline".center(40))
137.
138. print("".center(120,"-"))
139. for k in range(3):
140.     print("Logika".center(10) + "w1".center(10) + "w2".center(10)+
"b".center(10),end="")
141. print("".center(120,"-"))
142.
143. # Membuat nama logika yang digunakan
144. logic_name = {
145.     0 : "NAND",
146.     1 : "AND NOT",
147.     2 : "OR",
148.     3 : "NOR",
149.     4 : "XOR"
150. }
151.
152. for i in range(5):
153.     logic = logic_name.get(i)
154.     result = {
155.         0 : perceptron_result[i],
156.         1 : hebbnet_result[i],
157.         2 : adaline_result[i]
158.     }
159.
160. #     Menampilkan hasil training
161.     for j in range(3):
162.         print(logic.center(10) +
163.               "{:.2f}".format(result[j][0]).center(10) +
164.               "{:.2f}".format(result[j][1]).center(10) +
165.               "{:.2f}".format(result[j][2]).center(10), end="")
166.     print()
167.
168. print("".center(120,"-"))
169.
170.
171. # Visualisasi hasil
172.
173. target = {
174.     0 : target_nand,
175.     1 : target_andnot,
176.     2 : target_or,
177.     3 : target_nor,
178.     4 : target_xor,
179. }
180. title = {
181.     0 : ["Perceptron NAND","Perceptron AND NOT","Perceptron
OR","Perceptron NOR","Perceptron XOR"],
182.     1 : ["Hebb Net NAND","Hebb Net AND NOT","Hebb Net OR","Hebb
Net NOR","Hebb Net XOR"],
```

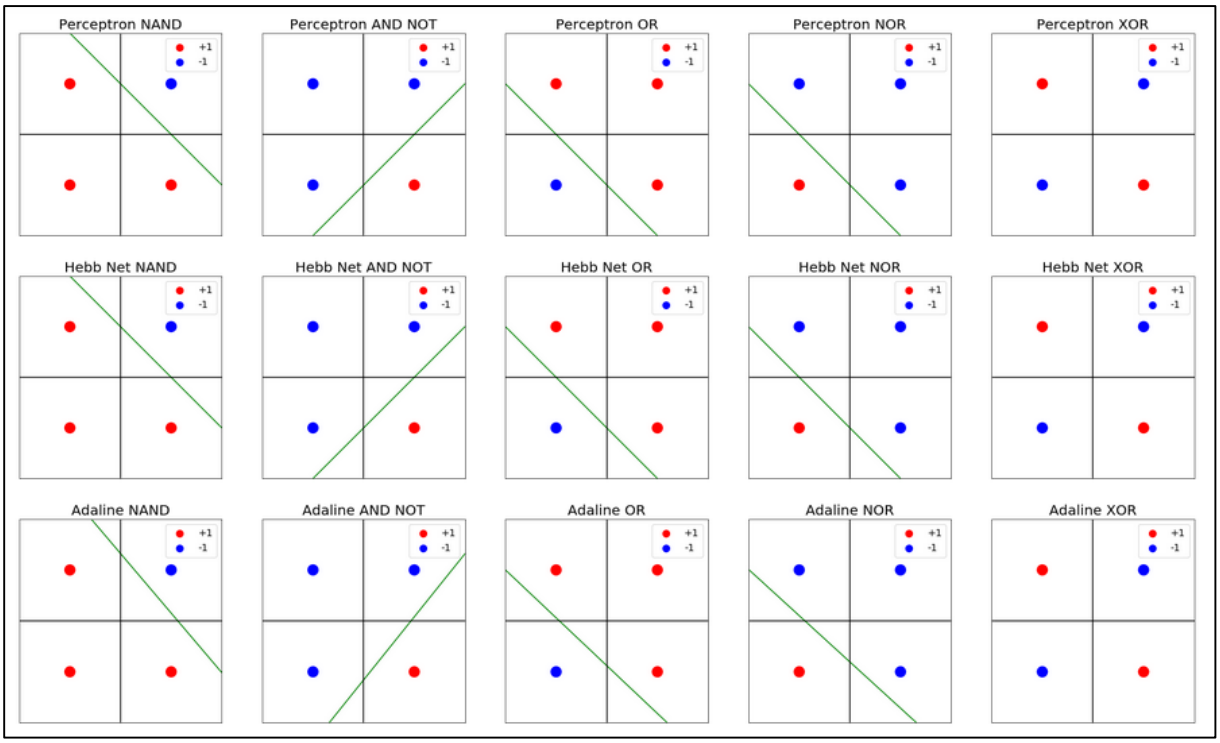
```
    2 : ["Adaline NAND","Adaline AND NOT","Adaline OR","Adaline NOR","Adaline XOR"]
}
result = {
    0 : perceptron_result,
    1 : hebbnet_result,
    2 : adaline_result
}

# Function untuk memvisualisasikan hasil
def plot_result(k):
    plt.figure(figsize=(30,30))
    for i in range(5):
#         Warna marker
        color = ["r" if c==1 else "b" for c in target[i]]

#         Transformasikan hasil ke dalam garis linier
        x = np.linspace(-2,2,2)
        if result.get(k)[i][1]==0:
            m=0
            n=3
        else:
            m = -result.get(k)[i][0]/result.get(k)[i][1]
            n = -result.get(k)[i][2]/result.get(k)[i][1]
        y = m*x+n

#         Visualisasikan hasil
        plt.subplot(1,5,i+1)
        plt.plot(x,y,"g-")
#         Visualisasikan marker penanda target
plt.scatter(xxb_input[:,0],xxb_input[:,1],c=color,linewidths=10)
        plt.scatter(3,3,c="r",linewidths=5,label="+1")
        plt.scatter(3,3,c="b",linewidths=5,label="-1")

#         Buat garis sumbu-x dan sumbu-y
        plt.plot([-2, 2],[0, 0],"k-")
        plt.plot([0, 0],[-2, 2],"k-")

#         Buat keterangan grafik
        plt.title(title.get(k)[i],fontsize = 20)
        plt.legend(fontsize = "x-large",loc = "best")
        plt.axis("square")

#         Batasi sumbu-x dan sumbu-y
        plt.xlim(-2,2)
        plt.ylim(-2,2)
        plt.xticks([])
        plt.yticks([])
    plt.show()
# Komparasikan visualisasi hasil
for k in range (3):
    plot_result(k)
```

C. RUNNING PROGRAM

Beberapa hasil outputan dari program tersebut adalah:

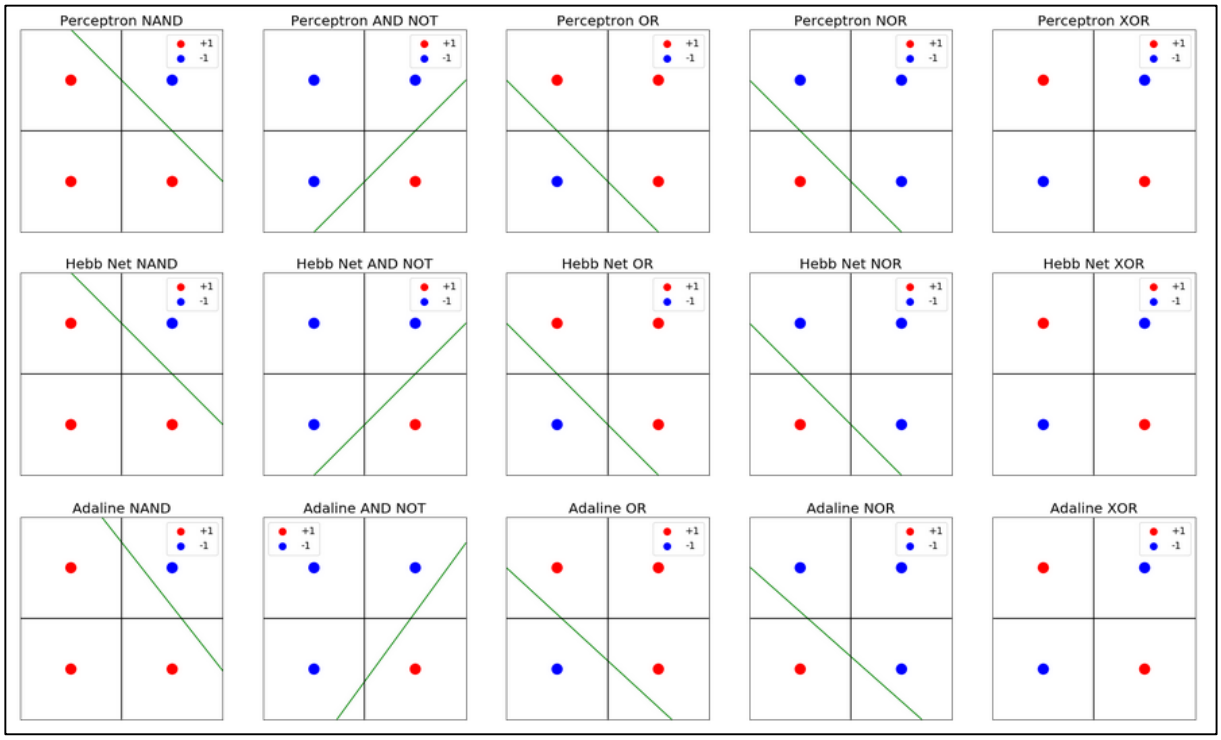
1) Tampilan dengan learning rate = 0.11, threshold = 0.12, dan toleransi = 0.05

Metode Perceptron				Metode Hebb Net				Metode Adaline			
Logika	w1	w2	b	Logika	w1	w2	b	Logika	w1	w2	b
NAND	-0.23	-0.23	0.23	NAND	-2.00	-2.00	2.00	NAND	-0.20	-0.17	0.22
AND NOT	0.23	-0.23	-0.23	AND NOT	2.00	-2.00	-2.00	AND NOT	0.53	-0.43	-0.50
OR	0.23	0.23	0.23	OR	2.00	2.00	2.00	OR	0.53	0.57	0.50
NOR	-0.23	-0.23	-0.23	NOR	-2.00	-2.00	-2.00	NOR	-0.23	-0.25	-0.20
XOR	0.00	0.00	0.00	XOR	0.00	0.00	0.00	XOR	0.00	0.00	0.00



2) Tampilan dengan learning rate = 0.15, threshold = 0.14, dan toleransi = 0.05

Metode Perceptron				Metode Hebb Net				Metode Adaline			
Logika	w1	w2	b	Logika	w1	w2	b	Logika	w1	w2	b
NAND	-0.15	-0.15	0.15	NAND	-2.00	-2.00	2.00	NAND	-0.25	-0.20	0.30
AND NOT	0.15	-0.15	-0.15	AND NOT	2.00	-2.00	-2.00	AND NOT	0.55	-0.40	-0.50
OR	0.15	0.15	0.15	OR	2.00	2.00	2.00	OR	0.55	0.60	0.50
NOR	-0.15	-0.15	-0.15	NOR	-2.00	-2.00	-2.00	NOR	-0.30	-0.34	-0.26
XOR	0.00	0.00	0.00	XOR	0.00	0.00	0.00	XOR	0.00	0.00	0.00



3) Tampilan dengan learning rate = 0.11, threshold = 0.07, dan toleransi = 0.05

Metode Perceptron				Metode Hebb Net				Metode Adaline			
Logika	w1	w2	b	Logika	w1	w2	b	Logika	w1	w2	b
NAND	-0.11	-0.11	0.11	NAND	-2.00	-2.00	2.00	NAND	-0.19	-0.16	0.21
AND NOT	0.11	-0.11	-0.11	AND NOT	2.00	-2.00	-2.00	AND NOT	0.53	-0.44	-0.50
OR	0.11	0.11	0.11	OR	2.00	2.00	2.00	OR	0.53	0.56	0.50
NOR	-0.11	-0.11	-0.11	NOR	-2.00	-2.00	-2.00	NOR	-0.22	-0.24	-0.19
XOR	0.00	0.00	0.00	XOR	0.00	0.00	0.00	XOR	0.00	0.00	0.00