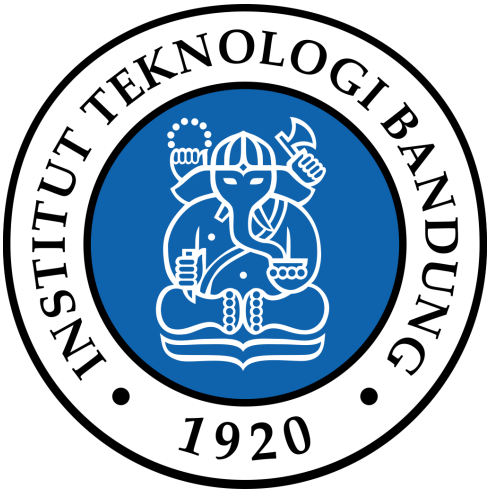
Laporan Tugas 1

IF3270 Pembelajaran Mesin

Implementasi Forward Propagation untuk Feed Forward Neural Network



Disusun oleh

Arjuna Marcelino	13519021
Sharon Bernadetha Marbun	13519092
Epata Tuah	13519120
Giant Andreas Tambunan	13519127

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2022

I. Penjelasan Implementasi

Implementasi Feed Forward Neural Network dilakukan dengan menggunakan Jupyter Notebook. Program akan menerima input file eksternal berekstensi csv. File eksternal tersebut merupakan data uji yang berupa tabel dan memuat bobot neutron-nya. Pada tugas ini, data uji dan model neuron diambil dari slide perkuliahan IF3270 Pembelajaran Mesin pada kasus XOR. Berikut ini adalah data uji dan model neuron yang digunakan.

Data Uji XOR

x0	x1	f
0	0	0
0	1	1
1	0	1
1	1	0

Model XOR Sigmoid

$$W_{xh} = \begin{bmatrix} 20 & 20 \\ -20 & -20 \end{bmatrix}, \quad c_{xh} = \begin{bmatrix} -10 \\ 30 \end{bmatrix}$$

$$W_{hy} = \begin{bmatrix} 20 \\ 20 \end{bmatrix}, \quad b = \begin{bmatrix} -10 \\ 30 \end{bmatrix}$$

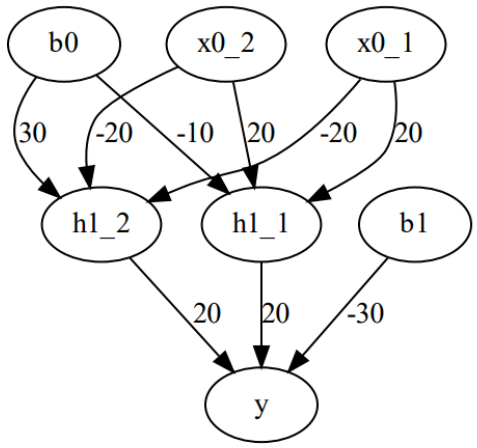
Model XOR ReLU-Linear

$$W_{xh} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad c_{xh} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

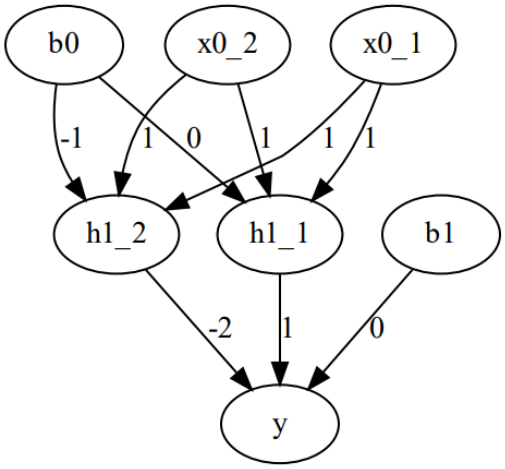
$$W_{hy} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, \quad b = 0$$

Dalam program ini, kita perlu terlebih dahulu menginstal library “graphviz”. Dengan menggunakan library “graphviz”, program dapat menampilkan hasil visualisasi model yang diberikan. Hasil visualisasi model akan diberikan dalam bentuk file berformat PDF. Berikut ini adalah tampilan dari model yang diberikan.

Model XOR Sigmoid



Model XOR ReLU-Linear



Program akan melakukan pembelajaran terhadap data uji yang ada. Data training dan model neuron akan direpresentasi dalam bentuk array dua dimensi. Ada empat buah fungsi aktivasi yang

akan digunakan, yaitu Linear, Sigmoid, ReLU, dan Softmax. Pada setiap layer, kita bisa menggunakan fungsi aktivasi yang berbeda. Fungsi aktivasi bertujuan untuk memberikan prediksi terhadap hasil pembelajaran model tersebut. Hasil yang didapat dari fungsi aktivasi berupa matriks yang berisi nilai keluaran dari setiap neuron pada layer.

Berikut ini adalah implementasi dari **fungsi aktivasi** yang digunakan.

```
import math
import numpy as np

def linear(x, kwargs=None):
    return x

def sigmoid(x):
    value = float(1 / (1 + math.exp(x * -1)))
    threshold = 0.1
    if value < threshold:
        return 0
    else:
        return 1

def relu(x):
    alpha = 0.0
    max_value = 1.0
    threshold = 0.0
    if x < threshold:
        if x > x*alpha:
            return x
        else:
            return x*alpha
    else:
        return min(x, max_value)

def softmax(arr):
    arr_exp = np.exp(arr)
    return arr_exp / arr_exp.sum()
```

Setelah melakukan aktivasi, program akan melakukan pembelajaran pada setiap data training menggunakan model neuron yang telah didefinisikan. Proses ini menggunakan Feed Forward Neural Network di mana pada setiap layer akan dilakukan perhitungan menggunakan fungsi aktivasi yang dipilih.

```
print("# # # # # # # # # # # # # # # # # #")
print("# Feed Forward Neural Network : XOR #")
print("# # # # # # # # # # # # # # # # # #\n")

bias, weigth , activation = readModel("modelsigmoid.txt")
data, target = readData()
fnnn = FNNN(bias, weigth, activation, data)
result = fnnn.get_results()

# print result etc
```

```
fnnn.resolve()
print()
print(f"TARGET CLASS\t\t: {target}")
print(f"RESULT CLASS\t\t: {result}")
print("============================")
if(result == target):
    print("Result: GOOD PREDICT")
    print("GOOD")
else:
    print("Result: BAD PREDICT")
    print("NO GOOD")


### VISUALIZATION
from graphviz import Digraph

dGraph = Digraph("FNNN: XOR", filename="model sigmoid.gv")

max = len(fnnn.layers)
for layer in fnnn.layers:
    idxL = fnnn.layers.index(layer)
    edges = layer.getDiGraph(idxL, max)
    for ed in edges:
        dGraph.edge(ed[0], ed[1], str(ed[2]))

dGraph.view()
```

Setelah itu, akan dijalankan program untuk setiap instance data training yang ada. Data training dapat berupa instance tunggal ataupun kumpulan berbentuk batch. Setiap instance pada data training disimpan ke array yang menjadi atribut dari kelas struktur model. Neural Network kemudian melakukan proses penghitungan pada setiap layer menggunakan fungsi *get_result()*. Fungsi *get_result()* akan memanggil fungsi *get_result()* dari semua layer yang disimpan oleh kelas model, yang kemudian untuk result setiap instance akan disimpan dalam bentuk array of result. Hal tersebut akan diulang-ulang hingga semua instance sudah di proses. Fungsi kemudian mengembalikan sebuah array of result.

```
def get_results(self):
        result = []
        for x in self.input:
            input = x
            for layer in self.layers:
                input = layer.get_result(input)

            result.append(input[0])

        return result

def get_result(self, input:list[int]):
       sigma = self.get_sigma(input)
       result = []
       for i in range(len(sigma)):
```

```
            result.append(self.activate_function(sigma[i]))
        return result
```

Pada fungsi *get_result()* yang digunakan untuk menghitung keluaran dari setiap layer. Sumber data input dari layer selain layer input adalah hasil perhitungan dari layer sebelumnya. Pada kelas FNNN, setiap layer akan disimpan secara berururt sedemikian rupa sehingga keluaran yang dihasilkan dengan menggunakan fungsi *get_sigma()* yang akan menghitung sigma dari layer yang kemudian dilakukan aktivasi fungsi (*activate*) yang mengembalikan result yang akan dipakai oleh layer selanjutnya.

```
def get_sigma(self, input:list[int]) -> list[int]:
        if(len(self.weight) == len(bias)):
            result = []
            for i in range(len(self.bias)):
                res = 1*self.bias[i]
                for j in range(len(input)):
                    res = res + input[j]*self.weight[j][i]

                result.append(res)

            return result

def get_result(self, input:list[int]):
        sigma = self.get_sigma(input)
        result = []
        for i in range(len(sigma)):
            result.append(self.activate_function(sigma[i]))
        return result
```

Perhitungan fungsi sigma dilakukan dengan perkalian matriks input dari hasil layer sebelumnya dengan matriks weight yang dimilikinya dengan menggunakan looping. Kemudian result dari layer akan dihitung dengan masukan output dari *get_sigma()* dan diaktivasi menggunakan fungsi *activate_function()*. Fungsi *activate_function()* akan mengaktivasi sigma dengan menggunakan fungsi aktivasi yang sesuai dengan model layer.

II. Hasil Pengujian

Ada dua pengujian yang dilakukan pada implementasi *forward propagation* untuk FFNN, yakni:

a. Memprediksi output untuk input 1 *instance*

- Model XOR Sigmoid

  Input:

```
#### INPUT 1 ####

====LAYER 0====
= = = = = = = = =
Input           : [1, 1]
weight          : [[20, -20], [20, -20]]
Activation      : sigmoid
Sigma           : [30, -10]

Result          : [1, 0]
= = = = = = = = =

====LAYER 1====
= = = = = = = = =
Input           : [1, 0]
weight          : [[20], [20]]
Activation      : sigmoid
Sigma           : [-10]

Result          : [0]
= = = = = = = = =
```

  Pada pengujian model XOR Sigmoid dengan menggunakan input *data training* 1 *instance*, didapatkan hasil sebagai berikut.

```
RESULT ==> [0]

TARGET CLASS            : [0]
RESULT CLASS            : [0]
==========================
Result: GOOD PREDICT
GOOD
'model sigmoid.gv.pdf'
```

- Model XOR ReLU-Linear

  Input:

```
#### INPUT 1 ####

====LAYER 0====
= = = = = = = = =
Input           : [1, 1]
weight          : [[1, 1], [1, 1]]
Activation      : relu
Sigma           : [2, 1]

Result          : [1.0, 1]
= = = = = = = = =

====LAYER 1====
= = = = = = = = =
Input           : [1.0, 1]
weight          : [[1], [-2]]
Activation      : linear
Sigma           : [-1.0]

Result          : [-1.0]
= = = = = = = = =
```

  Pada pengujian model XOR Relu-Linear dengan menggunakan input *data training batch*, didapatkan hasil sebagai berikut.

```
RESULT ==> [-1.0]

TARGET CLASS            : [0]
RESULT CLASS            : [-1.0]
==========================
Result: BAD PREDICT
NO GOOD
'model relu-linier.gv.pdf'
```

b. Memprediksi output untuk input *batch* sejumlah *instances*.

- Model XOR Sigmoid

  Input:

```
#### INPUT 1 ####

====LAYER 0====
= = = = = = = = = =
Input           : [0, 0]
weight          : [[20, -20], [20, -20]]
Activation      : sigmoid
Sigma           : [-10, 30]

Result          : [0, 1]
= = = = = = = = = =

====LAYER 1====
= = = = = = = = = =
Input           : [0, 1]
weight          : [[20], [20]]
Activation      : sigmoid
Sigma           : [-10]

Result          : [0]
= = = = = = = = = =

#### INPUT 2 ####

====LAYER 0====
= = = = = = = = = =
Input           : [0, 1]
weight          : [[20, -20], [20, -20]]
Activation      : sigmoid
Sigma           : [10, 10]

Result          : [1, 1]
= = = = = = = = = =

====LAYER 1====
= = = = = = = = = =
Input           : [1, 1]
weight          : [[20], [20]]
Activation      : sigmoid
Sigma           : [10]

Result          : [1]
= = = = = = = = = =
```

```
#### INPUT 3 ####

====LAYER 0====
= = = = = = = = = =
Input           : [1, 0]
weight          : [[20, -20], [20, -20]]
Activation      : sigmoid
Sigma           : [10, 10]

Result          : [1, 1]
= = = = = = = = = =

====LAYER 1====
= = = = = = = = = =
Input           : [1, 1]
weight          : [[20], [20]]
Activation      : sigmoid
Sigma           : [10]

Result          : [1]
= = = = = = = = = =

#### INPUT 4 ####

====LAYER 0====
= = = = = = = = = =
Input           : [1, 1]
weight          : [[20, -20], [20, -20]]
Activation      : sigmoid
Sigma           : [30, -10]

Result          : [1, 0]
= = = = = = = = = =

====LAYER 1====
= = = = = = = = = =
Input           : [1, 0]
weight          : [[20], [20]]
Activation      : sigmoid
Sigma           : [-10]

Result          : [0]
= = = = = = = = = =
```

Pada pengujian model XOR Sigmoid dengan menggunakan input *data training* 1 *instance*, didapatkan hasil sebagai berikut.

```
RESULT ==> [0, 1, 1, 0]

TARGET CLASS            : [0, 1, 1, 0]
RESULT CLASS            : [0, 1, 1, 0]
==========================
Result: GOOD PREDICT
GOOD
'model sigmoid.gv.pdf'
```

- Model XOR ReLU-Linear

  Input:

```
#### INPUT 1 ####

====LAYER 0====
= = = = = = = = =
Input           : [0, 0]
weight          : [[1, 1], [1, 1]]
Activation      : relu
Sigma           : [0, -1]

Result          : [0, -0.0]
= = = = = = = = =

====LAYER 1====
= = = = = = = = =
Input           : [0, -0.0]
weight          : [[1], [-2]]
Activation      : linear
Sigma           : [0.0]

Result          : [0.0]
= = = = = = = = =

#### INPUT 2 ####

====LAYER 0====
= = = = = = = = =
Input           : [0, 1]
weight          : [[1, 1], [1, 1]]
Activation      : relu
Sigma           : [1, 0]

Result          : [1, 0]
= = = = = = = = =

====LAYER 1====
= = = = = = = = =
Input           : [1, 0]
weight          : [[1], [-2]]
Activation      : linear
Sigma           : [1]

Result          : [1]
= = = = = = = = =
```

```
#### INPUT 3 ####

====LAYER 0====
= = = = = = = = =
Input           : [1, 0]
weight          : [[1, 1], [1, 1]]
Activation      : relu
Sigma           : [1, 0]

Result          : [1, 0]
= = = = = = = = =

====LAYER 1====
= = = = = = = = =
Input           : [1, 0]
weight          : [[1], [-2]]
Activation      : linear
Sigma           : [1]

Result          : [1]
= = = = = = = = =

#### INPUT 4 ####

====LAYER 0====
= = = = = = = = =
Input           : [1, 1]
weight          : [[1, 1], [1, 1]]
Activation      : relu
Sigma           : [2, 1]

Result          : [1.0, 1]
= = = = = = = = =

====LAYER 1====
= = = = = = = = =
Input           : [1.0, 1]
weight          : [[1], [-2]]
Activation      : linear
Sigma           : [-1.0]

Result          : [-1.0]
= = = = = = = = =
```

Pada pengujian model XOR Relu-Linear dengan menggunakan input *data training batch*, didapatkan hasil sebagai berikut.

```
RESULT ==> [0.0, 1, 1, -1.0]

TARGET CLASS            : [0, 1, 1, 0]
RESULT CLASS            : [0.0, 1, 1, -1.0]
============================
Result: BAD PREDICT
NO GOOD
'model relu-linier.gv.pdf'
```

III. Perbandingan dengan Hasil Perhitungan Manual

1. Model XOR Sigmoid

Hasil perhitungan oleh program adalah sebagai berikut.

```
#### INPUT 1 ####

====LAYER 0====
- - - - - - - - - -
Input           : [0, 0]
weight          : [[20, -20], [20, -20]]
Activation      : sigmoid
Sigma           : [-10, 30]

Result          : [0, 1]
- - - - - - - - - -

====LAYER 1====
- - - - - - - - - -
Input           : [0, 1]
weight          : [[20], [20]]
Activation      : sigmoid
Sigma           : [-10]

Result          : [0]
- - - - - - - - - -

#### INPUT 2 ####

====LAYER 0====
- - - - - - - - - -
Input           : [0, 1]
weight          : [[20, -20], [20, -20]]
Activation      : sigmoid
Sigma           : [10, 10]

Result          : [1, 1]
- - - - - - - - - -

====LAYER 1====
- - - - - - - - - -
Input           : [1, 1]
weight          : [[20], [20]]
Activation      : sigmoid
Sigma           : [10]

Result          : [1]
- - - - - - - - - -

#### INPUT 3 ####

====LAYER 0====
- - - - - - - - - -
Input           : [1, 0]
weight          : [[20, -20], [20, -20]]
Activation      : sigmoid
Sigma           : [10, 10]

Result          : [1, 1]
- - - - - - - - - -

====LAYER 1====
- - - - - - - - - -
Input           : [1, 1]
weight          : [[20], [20]]
Activation      : sigmoid
Sigma           : [10]

Result          : [1]
- - - - - - - - - -

#### INPUT 4 ####

====LAYER 0====
- - - - - - - - - -
Input           : [1, 1]
weight          : [[20, -20], [20, -20]]
Activation      : sigmoid
Sigma           : [30, -10]

Result          : [1, 0]
- - - - - - - - - -

====LAYER 1====
- - - - - - - - - -
Input           : [1, 0]
weight          : [[20], [20]]
Activation      : sigmoid
Sigma           : [-10]

Result          : [0]
- - - - - - - - - -

RESULT ==> [0, 1, 1, 0]

TARGET CLASS            : [0, 1, 1, 0]
RESULT CLASS            : [0, 1, 1, 0]
============================
Result: GOOD PREDICT
GOOD
'model sigmoid.gv.pdf'
```

Hasil perhitungan manualnya adalah sebagai berikut.

x0	x1	x2	f	Σh1	h1	Σh2	h2	Σy	y
1	0	0	0	-10	0.000045	30	1	-9.99909	0.000045

1	0	1	1	10	0.999954	10	0.999954	9.99818	0.999954
1	1	0	1	10	0.999954	10	0.999954	9.99818	0.999954
1	1	1	0	30	1	-10	0.000045	-9.99909	0.000045

Dengan melihat kedua hasil perhitungan di atas, kita bisa menyimpulkan bahwa keduanya hampir sama. Terdapat perbedaan minor pada nilai h1, h2, dan Σy, karena pada perhitungan oleh program diterapkan pembulatan ke bilangan bulat terdekat.

2. Model XOR ReLU + Linear

   Hasil perhitungan oleh program adalah sebagai berikut.

```
#### INPUT 1 ####

====LAYER 0====
= = = = = = = = =
Input           : [0, 0]
weight          : [[1, 1], [1, 1]]
Activation      : relu
Sigma           : [0, -1]

Result          : [0, -0.0]
= = = = = = = = =

====LAYER 1====
= = = = = = = = =
Input           : [0, -0.0]
weight          : [[1], [-2]]
Activation      : linear
Sigma           : [0.0]

Result          : [0.0]
= = = = = = = = =

#### INPUT 2 ####

====LAYER 0====
= = = = = = = = =
Input           : [0, 1]
weight          : [[1, 1], [1, 1]]
Activation      : relu
Sigma           : [1, 0]

Result          : [1, 0]
= = = = = = = = =

====LAYER 1====
= = = = = = = = =
Input           : [1, 0]
weight          : [[1], [-2]]
Activation      : linear
Sigma           : [1]

Result          : [1]
= = = = = = = = =
```

```
#### INPUT 3 ####

====LAYER 0====
= = = = = = = = =
Input           : [1, 0]
weight          : [[1, 1], [1, 1]]
Activation      : relu
Sigma           : [1, 0]

Result          : [1, 0]
= = = = = = = = =

====LAYER 1====
= = = = = = = = =
Input           : [1, 0]
weight          : [[1], [-2]]
Activation      : linear
Sigma           : [1]

Result          : [1]
= = = = = = = = =

#### INPUT 4 ####

====LAYER 0====
= = = = = = = = =
Input           : [1, 1]
weight          : [[1, 1], [1, 1]]
Activation      : relu
Sigma           : [2, 1]

Result          : [1.0, 1]
= = = = = = = = =

====LAYER 1====
= = = = = = = = =
Input           : [1.0, 1]
weight          : [[1], [-2]]
Activation      : linear
Sigma           : [-1.0]

Result          : [-1.0]
= = = = = = = = =
```

```
RESULT ==> [0.0, 1, 1, -1.0]

TARGET CLASS            : [0, 1, 1, 0]
RESULT CLASS            : [0.0, 1, 1, -1.0]
==========================
Result: BAD PREDICT
NO GOOD
'model relu-linier.gv.pdf'
```

Hasil perhitungan manualnya adalah sebagai berikut.

x0	x1	x2	f	Σh1	h1	Σh2	h2	Σy	y
1	0	0	0	0	0	-1	0	0	0
1	0	1	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1	1
1	1	1	0	2	2	1	1	0	0

Dengan melihat kedua hasil perhitungan di atas, kita bisa menyimpulkan bahwa hasilnya berbeda. Kelas target dari instance ke-4 dari pada perhitungan program adalah -1, sedangkan pada perhitungan manual adalah 0.

IV. Kontribusi Individual

NIM	Nama	Kontribusi
13519021	Arjuna Marcelino	Membuat laporan, implementasi main
13519092	Sharon Bernadetha Marbun	Membuat laporan, membuat fungsi aktivasi, membaca data
13519120	Epata Tuah	Membuat laporan, implementasi main
13519127	Giant Andreas Tambunan	Membuat laporan, membuat kelas layer dan kelas FFNN, implementasi main