Tugas Besar I IF3270 Pembelajaran Mesin Bagian A

Implementasi *Forward Propagation* untuk *Feed Forward Neural Network*

Disusun oleh:

Mohammad Afif Akromi	**13519110**
Bintang Fajarianto	**13519138**
Muhammad Fadli Gunardi	**13519130**

PROGRAM STUDI SARJANA INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021

Daftar Isi

1. Penjelasan Implementasi

Program diimplementasikan menggunakan bahasa pemrograman Python. Pertama, kami mendefinisikan dan mengimplementasikan keempat fungsi aktivasi yang akan digunakan dalam algoritma FFNN ini. Keempat fungsi aktivasi itu adalah *linear*, *RELU*, *sigmoid*, dan *softmax*. Keempat fungsi ini sama-sama menerima *input* berupa satu angka, lalu mengembalikan nilai bertipe *float*. Keempat fungsi tersebut dijadikan menjadi satu kesatuan fungsi yang dapat dilihat dibawah ini

```
import numpy as np
import json

''' --- Activation Function --- '''
def activation_func(nama, x):
  switcher = {
    "linear" : x,
    "sigmoid" : 1 / (1 + np.exp(-x)),
    "relu" : np.maximum(0, x),
    "softmax" : np.exp(x) / np.sum(np.exp(x)),
  }
  return switcher.get(nama, "")
```

Lalu, model FFNN disimpan dalam file .json yang dapat di-*import* ke dalam program berbahasa Python. Informasi yang disimpan pada file tersebut beruapa array of object yang isi objectnya berupa jumlah neuron, jenis aktivasi fungsi, array bobot, dan array bias

```
''' --- Predict --- '''
def predict(array):

    prediction = []
    for idx in range(len(layers)):
        x = np.dot(array, np.array(layers[idx]["weights"])) + np.array(layers[idx]["biases"]) \
            if idx == 0 else np.dot(layers[idx-1]["activation_value"], np.array(layer["weights"])) + np.array(layer["biases"])

        layers[idx]["activation_value"] = activation_func(layers[idx]["activation_function"], x)

    last_layer_activation_value = layers[-1]["activation_value"]
    prediction = np.copy(last_layer_activation_value)
    prediction = prediction.reshape(prediction.shape[0], 1)

    return prediction
```

Pada implementasi terdapat *method* bernama *predict* yang menerima *input* bertipe *numpy array*. *Input* harus terdapat *bias* yang digunakan untuk *input* (bernilai +1) dan tiap *instance* harus berada pada satu *array*. Sehingga, nantinya *input* memiliki bentuk seperti *matrix*. Lalu, *input* akan digunakan sebagai inisiasi dari *output*. Lalu, program akan menelusuri *weight matrix* pada masing-masing *layer* selain *input layer*. Lalu, dilakukan operasi berikut ini untuk menghasilkan *output*.

$$output = f(xW^T)$$

Dengan f adalah fungsi aktivasi yang digunakan pada *layer* tersebut, x adalah matriks *input*, dan W adalah *weight matrix*. Nantinya, jika belum mencapai *output layer*, *output* yang berupa *matrix* tersebut akan diberikan *row* yang berisikan *bias* (bernilai +1).

Ketika semua *layer* sudah ditelusuri dan sudah mencapai *output*, proses iterasi akan selesai dan *atribut* pada kelas yang bernama *output* akan di-*update*, dan fungsi mengembalikan *matrix* dalam bentuk *numpy array* yang merupakan hasil prediksi FFNN menggunakan *forward propagation*.

Output dari program ini adalah informasi dari file json itu sendiri yang mencakup jumlah neuron, jumlah layer, bobot, bias, dan jenis activation function yang akan dikombinasikan dengan output prediksi dari model tersebut dan nilai activation valuenya.

```
''' --- Info --- '''
def print_info():
    print(f"# Jumlah Layer: {len(layers)}")
    for idx in range(len(layers)):
        print(f"Layer ({idx+1})")
        print(f"- Jumlah Neuron: {layers[idx]['n_neuron']}")
        print(f"- Jenis Fungsi Aktivasi: {layers[idx]['activation_function']}")
        print(f"- Nilai Aktivasi: \n\t{layers[idx]['activation_value']}")
        print(f"- Bobot: \n\t{layers[idx]['weights']}")
        print(f"- Bias: \n\t{layers[idx]['biases']}\n")
    print(f"# Hasil Prediksi: {final_predictions}")
```

2. Hasil Pengujian

a. Model Pertama (Sigmoid)

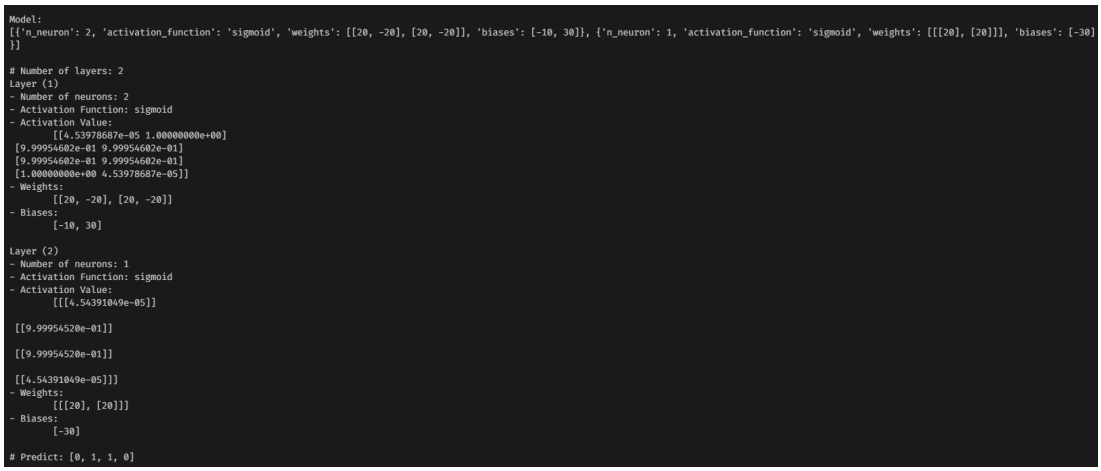
Pengujian dilakukan dengan menggunakan kedua model FFNN XOR pada *slide* kuliah (Sigmoid dan RELU-Linear). Berikut ini adalah konfigurasi input file yang sesuai dengan model pertama pada *slide* kuliah (Sigmoid).

File model.json (Sigmoid)

```
[
    {
        "n_neuron": "2",
        "activation_function": "sigmoid",
        "weights": [
            [20, -20],
            [20, -20]
        ],
        "biases": [-10, 30]
    },
    {
        "n_neuron": "1",
        "activation_function": "sigmoid",
        "weights": [
            [
                [20],
                [20]
```

```
                ]
            ],
            "biases": [-30]
        }
]
```

Lalu, model ini diujikan untuk tiap *instance* yang ada pada *slide* (model XOR). Berikut hasil pengujian pada tiap *instance*-nya.



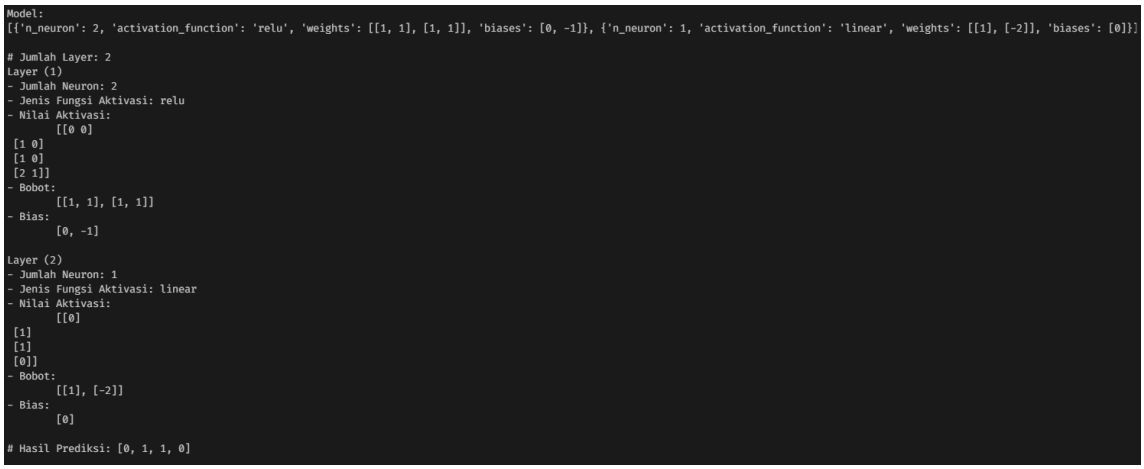
b. Model Kedua (RELU-Linear)

Sementara itu, berikut ini konfigurasi *input file* sehingga bisa sesuai dengan model kedua yang dimaksudkan pada *slide* kuliah (RELU-Linear).

```
[
  {
    "n_neuron": 2,
    "activation_function": "relu",
    "weights": [
      [1, 1],
      [1, 1]
    ],
    "biases": [0, -1]
```

```
  },
  {
    "n_neuron": 1,
    "activation_function": "linear",
    "weights": [[1], [-2]],
    "biases": [0]
  }
]
```

Lalu, model ini diujikan untuk tiap *instance* yang ada pada *slide* (model XOR). Pengujian dilakukan terhadap empat *instance* secara terpisah. Didapatkan hasil sebagai berikut.



```
Model:
[{'n_neuron': 2, 'activation_function': 'relu', 'weights': [[1, 1], [1, 1]], 'biases': [0, -1]}, {'n_neuron': 1, 'activation_function': 'linear', 'weights': [[1], [-2]], 'biases': [0]}]

# Jumlah Layer: 2
Layer (1)
- Jumlah Neuron: 2
- Jenis Fungsi Aktivasi: relu
- Nilai Aktivasi:
        [[0 0]
 [1 0]
 [1 0]
 [2 1]]
- Bobot:
        [[1, 1], [1, 1]]
- Bias:
        [0, -1]

Layer (2)
- Jumlah Neuron: 1
- Jenis Fungsi Aktivasi: linear
- Nilai Aktivasi:
        [[0]
 [1]
 [1]
 [0]]
- Bobot:
        [[1], [-2]]
- Bias:
        [0]

# Hasil Prediksi: [0, 1, 1, 0]
```

3. Perbandingan dengan Hasil Perhitungan Manual

Berikut ini adalah hasil perhitungan manual dengan bantuan *Google Sheets* dengan model FFNN XOR pertama (Sigmoid).

Epoch	Bias	X1	X2	W01	W02	W11	W12	W21	W22	Sum H1	Sum H2	Output H1 (Activation - Sigmoid)	Output H2 (Activation - Sigmoid)	Bias2	Wb2	Wh1	Wh2	Sum Output	Output O (Activation - Sigmoid)
1	1	0	0	-10	30	20	-20	20	-20	-10	30	4.54E-05	1.00E+00	1	-30	20	20	-1.00E+01	4.54E-05
	1	0	1	-10	30	20	-20	20	-20	10	10	1.00E+00	1.00E+00	1	-30	20	20	1.00E+01	1.00E+00
	1	1	0	-10	30	20	-20	20	-20	10	10	1.00E+00	1.00E+00	1	-30	20	20	1.00E+01	1.00E+00
	1	1	1	-10	30	20	-20	20	-20	30	-10	1.00E+00	4.54E-05	1	-30	20	20	-1.00E+01	4.54E-05

Didapatkan bahwa seluruh output untuk *instances* yang di-*predict* dengan perhitungan manual bernilai sama dengan yang di-predict dengan program yang dijalankan

di program python yakni **0, 1, 1, dan 0**. Sementara itu, berikut ini hasil perhitungan manual dengan model FFNN XOR kedua (RELU-Linear).

Epoch	Bias	X1	X2	W01	W02	W11	W12	W21	W22	Sum H1	Sum H2	Output H1 (Activation - ReLU)	Output H2 (Activation - ReLU)	Bias2	Wb2	Wh1	Wh2	Sum Output	Output O (Activation - Linear)
1	1	0	0	0	-1	1	1	1	1	0	-1	0.00E+00	0.00E+00	1	0	1	-2	0.00E+00	0.00E+00
	1	0	1	0	-1	1	1	1	1	1	0	1.00E+00	0.00E+00	1	0	1	-2	1.00E+00	1.00E+00
	1	1	0	0	-1	1	1	1	1	1	0	1.00E+00	0.00E+00	1	0	1	-2	1.00E+00	1.00E+00
	1	1	1	0	-1	1	1	1	1	2	1	2.00E+00	1.00E+00	1	0	1	-2	0.00E+00	0.00E+00

Didapatkan bahwa seluruh output untuk *instances* yang di-*predict* dengan perhitungan manual bernilai sama dengan yang di-predict dengan program yang dijalankan di dalam program python yakni **0, 1, 1, dan 0**.

4. Pembagian Tugas setiap Anggota Kelompok

Nama	NIM	Tugas
M Afif Akromi	13519110	- Pembuatan struktur model - Membaca file dari model.json - Membuat fungsi aktivasi - Membuat konversi dari json ke model - Membuat prediksi hasil output dari model - Laporan
M Fadli Gunardi	13519130	- Pembuatan struktur model - Membaca file dari model.json - Membuat fungsi aktivasi - Membuat konversi dari json ke model - Membuat prediksi hasil output dari model - Laporan
Bintang Fajarianto	13519138	- Pembuatan struktur model - Membaca file dari model.json - Membuat fungsi aktivasi - Membuat konversi dari json ke model - Membuat prediksi hasil

		output dari model - Laporan