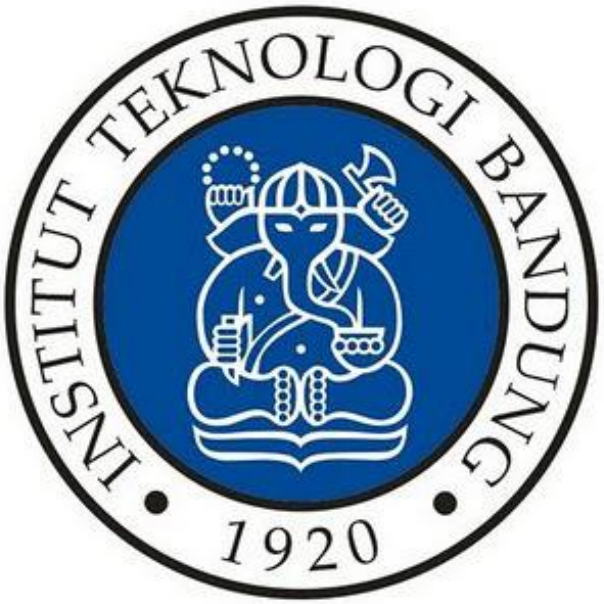
IF3270 Pembelajaran Mesin

Tugas Besar Bagian A: Implementasi Forward Propagation untuk Feed Forward Neural Network



Disusun oleh:

Daffa Ananda Pratama Resyaly	13519107
Raihan Astrada Fathurrahman	13519113
M. Ibnu Syah Hafizh	13519177
Muhammad Rayhan Ravianda	13519201

PROGRAM STUDI INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2019

I. Penjelasan Implementasi

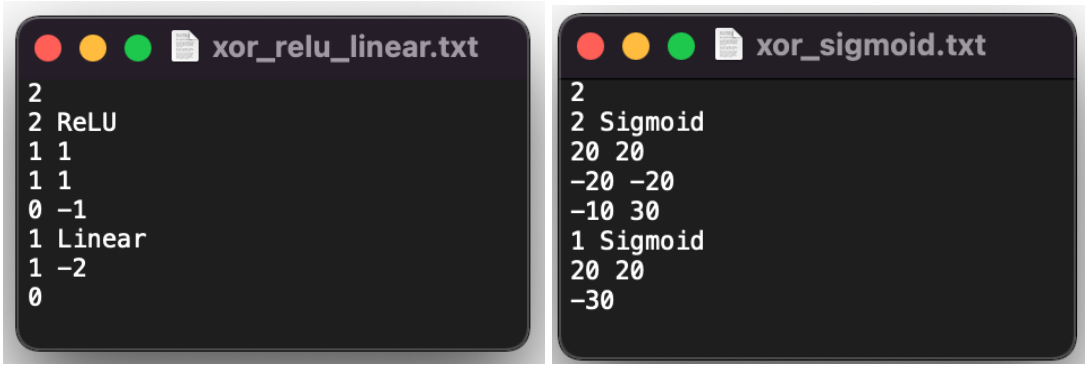
Implementasi forward propagation untuk FFNN dalam tugas besar bagian A ini menggunakan program dalam bahasa python. Program yang dibangun menggunakan library pandas dan numpy. Pada program ini dibuat 2 buah kelas yaitu kelas **Layer** yang merepresentasikan sebuah Layer dalam Neural Network, serta kelas **NeuralNetwork** yang merepresentasikan sebuah Neural Network.

Kelas **Layer** merupakan sebuah layer yang memiliki atribut ***weight*** yang berarti bobot untuk setiap neuron, bias yang merupakan nilai ***bias*** setiap neuron, dan ***activation*** yang berarti fungsi aktivasi pada layer tersebut. Pada kelas **Layer**, terdapat method atau fungsi forward propagation dengan nama fungsi ***forward_propagation(input)***. Method ini menerima input berupa data masukkan array (dapat berupa matriks, array n dimensi) dan akan mengembalikan nilai dari perhitungan fungsi aktivasi pada suatu layer yang sudah disesuaikan dengan fungsi aktivasi yang digunakan (dalam program ini yang dikenali adalah ***Linear***, ***Sigmoid***, ***ReLU***, dan ***Softmax***) dengan input fungsi aktivasi adalah hasil perhitungan perkalian dot antara weight atau matriks bobot dengan input awal pada fungsi forward propagation. Setiap fungsi aktivasi mengembalikan nilai menggunakan perhitungan pada setiap fungsi aktivasi yang valid.

Kelas **NeuralNetwork** memiliki atribut ***layers*** yang merupakan array of layer yang digunakan untuk menyimpan layer-layer yang terdapat pada file eksternal. Pada kelas **NeuralNetwork**, method yang terdapat dalam kelas ini yang pertama adalah ***loadfile(filename)*** yang digunakan untuk membaca dan memproses file eksternal menjadi layer-layer dan menambahkan layer-layer tersebut ke dalam atribut layers dengan menggunakan operasi ***add(layer)***. Method kedua adalah ***summary()***, yang digunakan untuk menampilkan model struktur dan koefisien dari file eksternal (beban) yang diinput dengan format yang telah ditentukan. Method selanjutnya yaitu method ***predict(input)*** yang digunakan untuk memprediksi output dari nilai yang diinput berdasarkan layers yang terdapat pada kelas.

II. Hasil Pengujian

Untuk menguji program yang kami buat, digunakan file txt eksternal yaitu "xor_sigmoid.txt" dan "xor_relu_linear.txt" yang masing-masing isinya adalah sebagai berikut:



Dengan menggunakan kedua file diatas, untuk masing-masing file dilakukan pengujian. Berikut adalah hasil pengujian yang diperoleh dengan menggunakan program yang dibangun:

A. Membaca model FFNN

```
# XOR Sigmoid Model dan XOR Relu-Linear Model
model1 = NeuralNetwork()
model2 = NeuralNetwork()
filename1 = 'xor_sigmoid.txt'
filename2 = 'xor_relu_linear.txt'

model1.load_file(filename1)
model2.load_file(filename2)
```

[4] ✓ 0.1s Python

```
File loaded. Model detected
File loaded. Model detected
```

B. Menampilkan model berupa struktur dan koefisiennya

```
model1.summary()
```

[5] ✓ 0.1s Python

```
Jumlah layer: 2
============================================================
Layer 1 (Activation: "Sigmoid", Units: 2)
Weight:
[[ 20.  20.]
 [-20. -20.]]
Bias:
[-10.  30.]
============================================================
Layer 2 (Activation: "Sigmoid", Units: 1)
Weight:
[[20. 20.]]
Bias:
[-30.]
============================================================
```

```
model2.summary()
```
[6] ✓ 0.2s Python

```
Jumlah layer:  2
============================================================
Layer 1 (Activation: "ReLU", Units: 2)
Weight:
[[1. 1.]
 [1. 1.]]
Bias:
[ 0. -1.]
============================================================
Layer 2 (Activation: "Linear", Units: 1)
Weight:
[[ 1. -2.]]
Bias:
[0.]
============================================================
```

C. Memprediksi output untuk input satu instance

```
data = [[0, 0],
        [0, 1],
        [1, 0],
        [1, 1]]
```
[7] ✓ 0.7s Python

```
model1.predict(data[0])
```
[8] ✓ 0.9s Python

```
[4.543910487654591e-05]
```

```
model2.predict(data[0])
```
[10] ✓ 0.2s Python

```
[0.0]
```

D. Memprediksi output untuk input batch sejumlah instances

```
model1.predict(data)
```
[9] ✓ 0.1s Python

```
[[4.543910487654591e-05],
 [0.999954519621495],
 [0.999954519621495],
 [4.543910487654591e-05]]
```

```
model2.predict(data)
```
[11] ✓ 0.2s Python

```
[[0.0], [1.0], [1.0], [0.0]]
```

III. Perbandingan dengan Hasil Perhitungan Manual

Selanjutnya adalah perbandingan nilai antara hasil perhitungan manual dengan hasil perhitungan program. Pertama, Pengujian dilakukan dengan struktur model sigmoid dan input x1 dan x2 = [[0, 0], [0, 1], [1, 0], [1, 1]].

Berikut adalah hasil perhitungan manual,

x0	x1	x2	f	Σh1	Σh2	Σy	h1	h2	y
1	0	0	0	-10	30	-30	0.00004539786	1	0.00004543910
1	0	1	1	10	10	-10	0.9999546021	0.9999546021	0.9999545196
1	1	0	1	10	10	-10	0.9999546021	0.9999546021	0.9999545196
1	1	1	0	30	-10	10	1	0.00004539786	0.00004543910

Layer 1 Nodes 1	Layer 1 Nodes 2	Output Layer Nodes 1	Weight
-10	30	-30	b0
20	-20	20	x1
20	-20	20	x2

Berikut adalah hasil perhitungan dengan menggunakan program,

```
In [4]: # XOR Sigmoid Model dan XOR Relu-Linear Model
        model1 = NeuralNetwork()
        model2 = NeuralNetwork()
        filename1 = 'xor_sigmoid.txt'
        filename2 = 'xor_relu_linear.txt'

        model1.load_file(filename1)
        model2.load_file(filename2)

        File loaded. Model detected
        File loaded. Model detected

In [5]: model1.summary()

        Jumlah layer:  2
        ============================================================
        Layer 1 (Activation: "Sigmoid", Units: 2)
        Weight:
        [[ 20.  20.]
         [-20. -20.]]
        Bias:
        [-10.  30.]
        ============================================================
        Layer 2 (Activation: "Sigmoid", Units: 1)
        Weight:
        [[20. 20.]]
        Bias:
        [-30.]
        ============================================================

In [7]: data = [[0, 0],
                [0, 1],
                [1, 0],
                [1, 1]]

In [8]: model1.predict(data[0])

Out[8]: [4.543910487654591e-05]

In [9]: model1.predict(data)

Out[9]: [[4.543910487654591e-05],
         [0.999954519621495],
         [0.999954519621495],
         [4.543910487654591e-05]]
```

Kemudian, pengujian kedua merupakan pengujian yang dilakukan dengan struktur model ReLU dan Linear dengan input x1 dan x2 = [[0, 0], [0, 1], [1, 0], [1, 1]].

Berikut adalah hasil perhitungan manual,

	A	B	C	D	E	F	G	H	I	J
1	Layer 1 Nodes 1	Layer 1 Nodes 2	Output Layer Nodes 1							
2	**Weight**	**Weight**	**Weight**							
3	0	-1	0	**b0**		**Layer**	**Activation**	**Nodes**	**Nodes**	
4	1	1	1	**x1**		Layer 1	ReLU	2	2	
5	1	1	-2	**x2**		Layer 2	Linear	1	1	
6										
7										
8	Layer 1									
9	**Xb1**	**X1**	**X2**	**Σh1**	**Σh2**	**Σy**	**ReLU(h1)**	**Linear(h2)**		
10	1	0	0	0	-1	0	0.00E+00	0		
11	1	0	1	1	0	-2	1.00E+00	0		
12	1	1	0	1	0	1	1.00E+00	0		
13	1	1	1	2	1	-1	2.00E+00	1		
14										
15										
16	Output Layer									
17	**Xb1**	**X1**	**X2**	**Σh1**	**y**					
18	1	0.00E+00	0.00E+00	0.00E+00	0.00E+00					
19	1	1.00E+00	0.00E+00	1.00E+00	1.00E+00					
20	1	1.00E+00	0.00E+00	1.00E+00	1.00E+00					
21	1	2.00E+00	1.00E+00	0.00E+00	0.00E+00					
22										

Berikut adalah hasil perhitungan dengan menggunakan program,

In [4]:
```
# XOR Sigmoid Model dan XOR Relu-Linear Model
model1 = NeuralNetwork()
model2 = NeuralNetwork()
filename1 = 'xor_sigmoid.txt'
filename2 = 'xor_relu_linear.txt'

model1.load_file(filename1)
model2.load_file(filename2)
```

```
File loaded. Model detected
File loaded. Model detected
```

In [6]:
```
model2.summary()
```

```
Jumlah layer:  2
============================================================
Layer 1 (Activation: "ReLU", Units: 2)
Weight:
[[1. 1.]
 [1. 1.]]
Bias:
[ 0. -1.]
============================================================
Layer 2 (Activation: "Linear", Units: 1)
Weight:
[[ 1. -2.]]
Bias:
[0.]
============================================================
```

In [7]:
```
data = [[0, 0],
        [0, 1],
        [1, 0],
        [1, 1]]
```

In [10]:
```
model2.predict(data[0])
```

Out[10]:
```
[0.0]
```

In [11]:
```
model2.predict(data)
```

Out[11]:
```
[[0.0], [1.0], [1.0], [0.0]]
```

Berdasarkan hasil perhitungan dengan menggunakan metode manual dan dengan menggunakan program pada kedua struktur model, dapat dilihat bahwa nilai perhitungan yang dihasilkan oleh program sama dengan nilai perhitungan yang dihasilkan oleh perhitungan manual sehingga dapat disimpulkan bahwa program yang dibangun sudah tepat mengimplementasikan forward propagation untuk FFNN.

IV. Pembagian Tugas

Nama	NIM	Tugas
Daffa Ananda Pratama Resyaly	13519107	Debugging, Laporan, Analisis Hasil
Raihan Astrada Fathurrahman	13519113	Pemodelan Neural Network, Laporan, Analisis Hasil
M. Ibnu Syah Hafizh	13519177	Prediksi, Laporan, Analisis Hasil
Muhammad Rayhan Ravianda	13519201	Laporan, Analisis Hasil

V. Lampiran

Github:
https://github.com/slarkdarr/Tubes-A-ML.git

Spreadsheet perhitungan manual:
https://docs.google.com/spreadsheets/d/1rUt1DHFe92PbKfIeEGWkuxZlOEvCfUv7aph5_n2f7W4/edit#gid=1795621192