Tugas Besar A

Implementasi Forward Propagation untuk Feed Forward Neural Network

IF3270 Pembelajaran Mesin

Dibuat oleh:

Daffa Pratama Putra	13518033
Muchammad Ibnu Sidqi	13518072
Muhammad Firas	13518117
Muhammad Daffa Dinaya	13518141

Teknik Informatika

Institut Teknologi Bandung

2021

1. Penjelasan Implementasi

Pada tugas ini, kami membuat implementasi Feed Forward Neural Network menggunakan Jupyter Notebook. Program yang kami buat, akan menerima input berupa file eksternal csv yang berisikan data uji berupa tabel dan bobot neuron-nya. Data uji dan model neuron yang digunakan diambil dari slide perkuliahan IF3270 Pembelajaran Mesin pada kasus XOR. Berikut 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

Neuron Model 1 (Sigmoid)

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

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

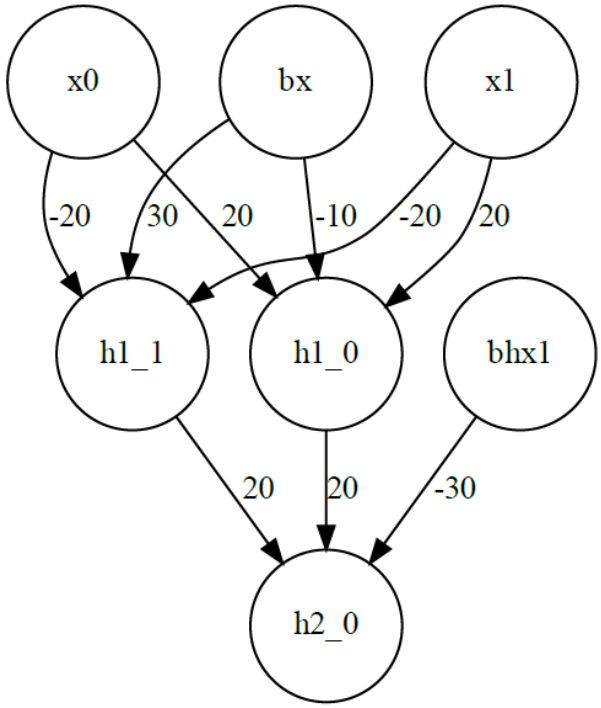
Neuron Model 2 (Relu & Linear)

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

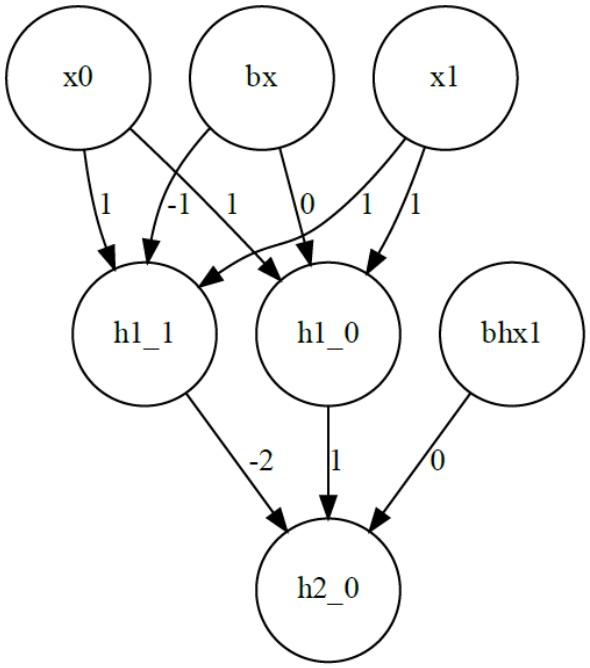
$$W_{hy} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, \quad b = 0$$

Program dapat menampilkan visualisasi dari model yang diberikan. Visualisasi tersebut menggunakan library “graphviz” dari Python, sehingga untuk menampilkan bentuk model diwajibkan melakukan instalasi library “graphviz”. Keluaran dari visualisasi model ditampilkan dalam file dengan format PDF yang dibuat secara otomatis oleh program. Berikut adalah tampilan dari model yang diberikan.

Neuron Model 1 (Sigmoid)



Neuron Model 2 (Relu & Linear)



Setelah data training dan model neuron sudah tersedia, program akan melakukan pembelajaran terhadap data uji tersebut. Representasi data training dan model neuron menggunakan array 2 dimensi. Terdapat 4 fungsi aktivasi yang dapat digunakan, yakni Linear, Sigmoid, ReLU, dan Softmax. Setiap layer bisa menggunakan fungsi aktivasi yang berbeda dengan layer lainnya. Fungsi aktivasi ini digunakan untuk memprediksi hasil dari pembelajaran model tersebut. Keluaran dari fungsi aktivasi adalah matriks yang berisi nilai keluaran dari setiap neuron pada suatu layer. Berikut adalah implementasi dari fungsi aktivasi yang digunakan.

Fungsi Aktivasi

```
import math
import numpy as np

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

def sigmoid(x, kwargs=None):
    value = float(1 / (1 + math.exp(x * -1)))
    threshold = kwargs.get("threshold", None)
    if threshold == None:
        return value
    else:
        if value < threshold:
            return 0
        else:
```

```
            return 1

def relu(x, kwargs):
    alpha = kwargs.get("alpha", 0.0)
    max_value = kwargs.get("max_value", 1.0)
    threshold = kwargs.get("threshold", 0.0)
    if x < threshold:
        return max(x, x * alpha)
    else:
        if max_value == None:
            return x
        else:
            return min(x, max_value)

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

Selanjutnya program akan melakukan pembelajaran untuk setiap data training yang diberikan menggunakan model neuron yang sudah didefinisikan di atas. Pada proses ini menggunakan Feed Forward Neural Network. Pada setiap layer akan dilakukan perhitungan untuk menggunakan fungsi aktivasi yang dipilih dengan masukan berupa data training dan model neuron pada layer tersebut.

Fungsi Main

```
for data in data_training:
        layer.insert(0, InputLayer(data))
        neural_network.base_layer = layer
        neural_network.solve()
        for x in neural_network.current_layer:
            print("result")
            print(x.result)
        result.append(neural_network.current_layer[-1].result)
        neural_network.deque_layer()
for x in result:
        print(result)
```

Program akan menjalankan Feed Forward Neural Network dengan menggunakan struktur neural network dari model yang sudah ada. Kemudian Neural Network akan menjalankan program untuk setiap instance data training yang sudah diberikan. Data training dapat berupa instance tunggal maupun dalam kumpulan berbentuk batch. Setiap instance akan data training akan dimasukkan ke dalam urutan pertama dari struktur model. Kemudian Neural Network akan melakukan proses penghitungan pada setiap layer menggunakan fungsi *solve*. Setelah sebuah

instance data training selesai digunakan untuk training. Maka hasil dari output akan disimpan di list *result*. Kemudian data training yang sudah digunakan tersebut akan "dilepas" dari layer pertama Neural Network dan digantikan dengan data instance berikutnya jika data training berbentuk batch.

Fungsi Solve

```
def solve(self):
        self.current_layer = self.base_layer.copy()
        for idx in range(len(self.current_layer)):
            if idx != 0:
                self.current_layer[idx].input_value =
self.current_layer[idx-1].result
            self.current_layer[idx].compute()
```

Pada fungsi *solve* yang digunakan untuk menghitung output dari setiap layer, kumpulan layer dari model akan disalin ke current layer terlebih dahulu agar model selalu memiliki struktur asli dari pembacaan data text. Sumber data input dari suatu layer selain layer input adalah hasil perhitungan dari layer sebelumnya. Kemudian setiap layer selain layer input akan dihitung output yang dihasilkan. dengan menggunakan fungsi *compute* dimana fungsi compute akan melakukan perhitungan sigma(*sigma*) dari layer dan melakukan aktivasi fungsi(*activate*)

Fungsi Sigma dan Activate

```
def activate(self):
        self.result = self.activation_function(self.result, self.kwargs)

def sigma(self):
        # case 1 Dimension
        if(len(self.weight) == 1):
            self.result = np.matmul(
                self.input_value, self.weight.flatten()) + self.bias
        else:
            self.result = np.matmul(np.transpose(
                self.input_value), self.weight) + self.bias
```

Perhitungan fungsi sigma menggunakan perkalian matriks input dari hasil keluaran layer sebelumnya dengan matriks weight yang dimiliki oleh layer tersebut. Penggunaan transpose

ditujukan agar matriks input memiliki dimensi yang bersesuaian untuk perkalian matriks weight. Untuk kasus satu dimensi juga digunakan fungsi flatten juga ditujukan untuk tujuan yang sama.

2. Hasil Pengujian

Berikut adalah hasil pengujian dari implementasi Forward Propagation untuk Feed Forward Neural Network yang kami buat.

- Model Sigmoid
    - 1 Instance

```
if __name__ == "__main__":
    main()
```

```
Feed Forward Neural Network : XOR
==================================
Activation      : sigmoid sigmoid
Target Class    :  [0]
Predict Class   :  [array([0])]
==================================
Result : Good Predict
```

Pada pengujian kasus XOR menggunakan model Sigmoid dengan data training untuk 1 instance (x1=0, x2=0), didapatkan prediksi kelasnya **sesuai** dengan kelas yang diharapkan pada target class.

- Batch

```
if __name__ == "__main__":
    main()
```

```
Feed Forward Neural Network : XOR
==================================
Activation      : sigmoid sigmoid
Target Class    :  [0, 1, 1, 0]
Predict Class   :  [array([0]), array([1]), array([1]), array([0])]
==================================
Result : Good Predict
```

Pada pengujian kasus XOR menggunakan model Sigmoid dengan data training batch, didapatkan prediksi kelasnya **sesuai** dengan kelas yang diharapkan pada target class.

- Model ReLU + Linear

```
if __name__ == "__main__":
    main()
```

```
Feed Forward Neural Network : XOR
==================================
Activation      : relu linear
Target Class    :  [1]
Predict Class   :  [array([1])]
==================================
Result : Good Predict
```

- 1 Instance

Pada pengujian kasus XOR menggunakan model Relu+Linear dengan data training untuk 1 instance (x1=1, x2=0), didapatkan prediksi kelasnya **sesuai** dengan kelas yang diharapkan pada target class.

- Batch

```
if __name__ == "__main__":
    main()
```

```
Feed Forward Neural Network : XOR
==================================
Activation      : relu linear
Target Class    :  [0]
Predict Class   :  [array([0])]
==================================
Result : Good Predict
```

Pada pengujian kasus XOR menggunakan model ReLU + linear, didapatkan prediksi kelasnya **sesuai** dengan kelas yang diharapkan pada target class.

3. Perbandingan Perhitungan

Berikut adalah beberapa perbandingan perhitungan menggunakan program dan manual.

1. Neuron Model Sigmoid

- Perhitungan manual

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

- Perhitungan dengan implementasi yang kami buat

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

Berikut adalah keluaran dari kode program

```
Feed Forward Neural Network : XOR
==================================
Activation      : sigmoid sigmoid

LAYER === 0
Input layer     : [0 0]

LAYER === 1
Input   :  [0 0]
Sigma   :  [-10  30]
Weight  :  [[ 20 -20]
 [ 20 -20]]
Result  :  [0 1]

LAYER === 2
Input   :  [0 1]
Sigma   :  [-10]
Weight  :  [[20]
 [20]]
Result  :  [0]

LAYER === 0
Input layer     : [0 1]

LAYER === 1
Input   :  [0 1]
Sigma   :  [10 10]
Weight  :  [[ 20 -20]
 [ 20 -20]]
Result  :  [1 1]

LAYER === 2
Input   :  [1 1]
Sigma   :  [10]
Weight  :  [[20]
 [20]]
Result  :  [1]
```

```
LAYER === 0
Input layer     : [1 0]

LAYER === 1
Input   :  [1 0]
Sigma   :  [10 10]
Weight  :  [[ 20 -20]
 [ 20 -20]]
Result  :  [1 1]

LAYER === 2
Input   :  [1 1]
Sigma   :  [10]
Weight  :  [[20]
 [20]]
Result  :  [1]

LAYER === 0
Input layer     : [1 1]

LAYER === 1
Input   :  [1 1]
Sigma   :  [ 30 -10]
Weight  :  [[ 20 -20]
 [ 20 -20]]
Result  :  [1 0]

LAYER === 2
Input   :  [1 0]
Sigma   :  [-10]
Weight  :  [[20]
 [20]]
Result  :  [0]
Target Class    :  [0, 1, 1, 0]
Predict Class   :  [array([0]), array([1]), array([1]), array([0])]
==================================
Result : Good Predict
```

Dari perbandingan perhitungan menggunakan program dan manual terdapat beberapa perbedaan nilai, yakni pada h1, h2, dan y. Hal ini terjadi karena pada perhitungan manual tidak menggunakan pembulatan ke bilangan bulat terdekat, sedangkan pada kode program yang kami buat menggunakan pembulatan. Tetapi jika dibulatkan, keduanya akan menghasilkan nilai yang **sama**.

2. Neuron Model Relu + Linear

- Perhitungan manual

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

- Perhitungan dengan implementasi yang kami buat

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

Berikut adalah keluaran dari program

```
Feed Forward Neural Network : XOR
==================================
Activation      : relu linear

LAYER === 0
Input layer     : [0 0]

LAYER === 1
Input   :  [0 0]
Sigma   :  [ 0 -1]
Weight  :  [[1 1]
 [1 1]]
Result  :  [0 0]

LAYER === 2
Input   :  [0 0]
Sigma   :  [0]
Weight  :  [[ 1]
 [-2]]
Result  :  [0]

LAYER === 0
Input layer     : [0 1]

LAYER === 1
Input   :  [0 1]
Sigma   :  [1 0]
Weight  :  [[1 1]
 [1 1]]
Result  :  [1 0]

LAYER === 2
Input   :  [1 0]
Sigma   :  [1]
Weight  :  [[ 1]
 [-2]]
Result  :  [1]

LAYER === 0
Input layer     : [1 0]

LAYER === 1
Input   :  [1 0]
Sigma   :  [1 0]
Weight  :  [[1 1]
 [1 1]]
Result  :  [1 0]

LAYER === 2
Input   :  [1 0]
Sigma   :  [1]
Weight  :  [[ 1]
 [-2]]
Result  :  [1]

LAYER === 0
Input layer     : [1 1]

LAYER === 1
Input   :  [1 1]
Sigma   :  [2 1]
Weight  :  [[1 1]
 [1 1]]
Result  :  [2 1]

LAYER === 2
Input   :  [2 1]
Sigma   :  [0]
Weight  :  [[ 1]
 [-2]]
Result  :  [0]
Target Class    :  [0, 1, 1, 0]
Predict Class   :  [array([0]), array([1]), array([1]), array([0])]
==================================
Result : Good Predict
```

Dari perbandingan perhitungan manual dengan program, keduanya menghasilkan keluaran yang **sama**.

4. Pembagian Kerja

NIM	Nama	Kontribusi
13518033	Daffa Pratama Putra	- Membuat laporan - Membuat data uji - Membuat reader data uji dan model - Melakukan pengujian pada data uji dan model neuron
13518072	Muchammad Ibnu Sidqi	- Membuat laporan - Membuat fungsi aktivasi - Melakukan testing pada fungsi aktivasi
13518117	Muhammad Firas	- Membuat laporan - Melakukan perhitungan manual terhadap data uji
13518141	Muhammad Daffa Dinaya	- Membuat laporan - Membuat struktur neural network dan layer - Integrasi layer dengan fungsi aktivasi - Membuat visualisasi model