Tugas Besar 2

IF4074 Advanced Machine Learning

LSTM - Long Short-term Memory

oleh :

Syihabuddin Yahya Muhammad	/ 13519149
Imam Nurul Hukmi	/ 13519150
Muhammad Furqon	/ 13519184

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

2022

1. Penjelasan Kode

1.1. Kelas LSTM

Kelas LSTM merupakan kelas utama dari implementasi arsitektur LSTM. Kelas ini dikembangkan dari kelas FFNN(atau disebut juga dense) dari tugas sebelumnya. Fitur baru dari kelas ini hanyalah sebuah opsi untuk menggunakan layer LSTM. Pada penggunaan model LSTM, cukup kelas ini saja yang digunakan. Kelas ini hanya dapat melakukan *forward propagation* jika mengandung layer LSTM, namun tetap dapat melakukan backward propagation jika tidak ada layer LSTM.

1.2. Kelas Layer_LSTM

Kelas ini merupakan kelas yang mengimplementasikan layer LSTM dengan menggunakan Cell. pada kelas ini belum(tidak akan) terdapat fungsi untuk melakukan *backward propagation* sehingga jika layer ini digunakan, model tidak akan dapat melakukan *backward propagation*.

1.3. Kelas Cell State dan Gate

Kelas Cell mewakili cell tempat dilakukannya pemrosesan data pada layer LSTM. Pada kelas Cell, telah disimpan nilai U, W, dan bobot yang dibutuhkan dalam LSTM. Selain itu, telah tersedia juga method-method yang dapat digunakan untuk melakukan perhitungan pada LSTM, mulai dari perhitungan setiap gate hingga pemrosesan data input pada Cell untuk mendapatkan nilai h dan C.

1.3.1. Base gate calculation

```python
#Gate sigmoid
def calculate_gate(self,u: list[float], x: list[float], w: float, h: float, b:float):
    return sigmoid(np.dot(u, x) + w * h + b)
```

1.3.2. Forgot gate

```python
#Forgot gate
def calculate_forgot(self,uf: list[float], x: list[float], wf: float, hprev: float, bf:float):
    return self.calculate_gate(uf, x, wf, hprev, bf)
```

1.3.3. Input gate

```python
#Input gate
def calculate_input(self,ui: list[float], x: list[float], wi: float, hprev: float, bi:float):
    return self.calculate_gate(ui, x, wi, hprev, bi)
```

1.3.4. Cell state

```python
#Calculate candidate
def calculate_candidate(self,uc: list[float], x: list[float], wc: float, hprev: float, bc:float):
    return np.tanh(np.dot(uc, x) + wc * hprev + bc)

#Cell state
def calculate_cell(self,ft: float, cprev:float, it: float, candidate:float):
    return ft*cprev + it*candidate
```

1.3.5. Output gate

```python
#Output gate
def calculate_output(self,uo: list[float], x: list[float], wo: float, hprev: float, bo:float):
    return self.calculate_gate(uo,x,wo,hprev,bo)
```

1.3.6. Kalkulasi data timestep

```python
def calculate_timestep(self,x, cprev=0, hprev=0, verbose=False):
    ft = self.calculate_forgot(self.Uf, x, self.Wf, hprev, self.bf)
    it = self.calculate_input(self.Ui, x, self.Wi, hprev, self.bi)
    candidate = self.calculate_candidate(self.Uc, x, self.Wc, hprev, self.bc)
    ct = self.calculate_cell(ft, cprev, it, candidate)
    ot = self.calculate_output(self.Uo, x, self.Wo, hprev, self.bo)
    ht = self.calculate_hidden(ot,ct)

    if(verbose):
        print("ht: ",ht)
        print("ft: ",ft)
        print("it: ",it)
        print("c~: ",candidate)
        print("ct: ",ct)
        print("ot: ",ot)
        print("Timestep ct {} dan ht {}".format(ct,ht))

    # self.cprev = ct
    # self.hprev = ht
    return ct,ht
```

```python
def calculate_hidden(self,ot:float, ct:float):
    return ot * np.tanh(ct)
```

2. Hasil Eksperimen

Kode program dapat diakses di https://github.com/LastAeon/tubes-ML-lanjutan-CNN
Kode program dari eksperimen model sequential dapat diakses di LinkColab https://colab.research.google.com/drive/1XAzagyKsNBDMXaAB-CteGz1qQgPROlvV?usp=sharing

2.1. Input

Input yang digunakan adalah input dengan 32 timesteps. Dilakukan preproses terhadap input dengan menghilangkan fitur tanggal atau *date*.

```
# Import TRAIN data

input_dir = '../dataset_LSTM/'

df_test = pd.read_csv(input_dir + 'ETH-USD-Test.csv')
del df_test['Date']
df_test.head()
```

	Open	High	Low	Close	Volume
0	1657.336548	1718.183228	1656.856079	1696.457031	14818795695
1	1696.324585	1698.561035	1498.771240	1507.782837	26713710143
2	1508.156982	1517.150024	1454.282959	1491.395020	18120831899
3	1491.206787	1505.791992	1430.547363	1430.547363	12823572918
4	1430.439453	1556.309570	1427.728394	1553.037354	17965837488

```
[ ] x_list = list(df_test.values[:,0:4])

data_x = []
data_y = []

timesteps = 32

for i in range(timesteps, len(x_list)):
    data_x.append(x_list[i - timesteps:i])
    data_y.append(x_list[i])

X_data, y_data = np.array(data_x), np.array(data_y)
X_data
```

Dilakukan prediksi dengan data test (X_data)

```
from LSTM import LSTM

rnn_test = LSTM("LSTM_Architecture_For_Test.txt")
rnn_test.printModel()
print("LSTM result:", rnn_test.predict(X_data[0]))
```

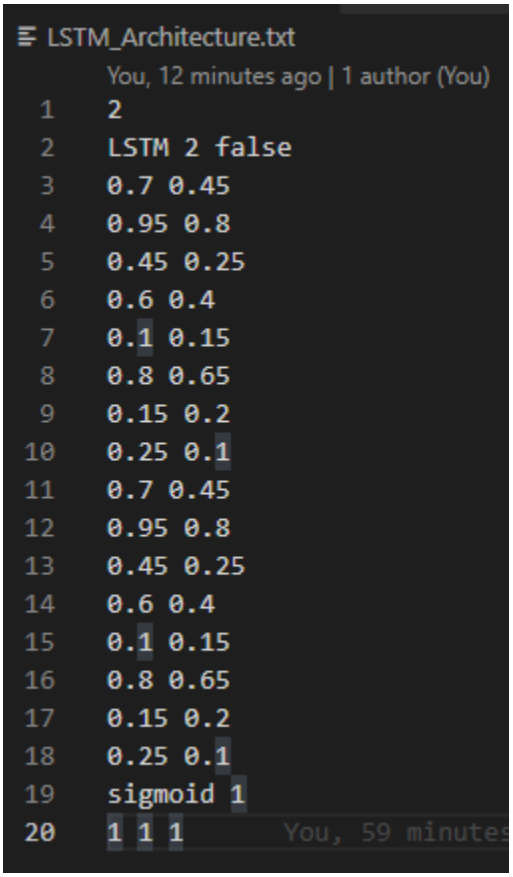
2.2. Architecture

Dikarenakan LSTM ini dikembangkan dari FFNN tugas sebelumnya, spesifikasi arsitekturnya juga sama. Yang membedakan hanyalah ketika muncul layer LSTM. Berikut arsitektur yang digunakan untuk melakukan testing.

LSTM_Architecture_For_Test.txt

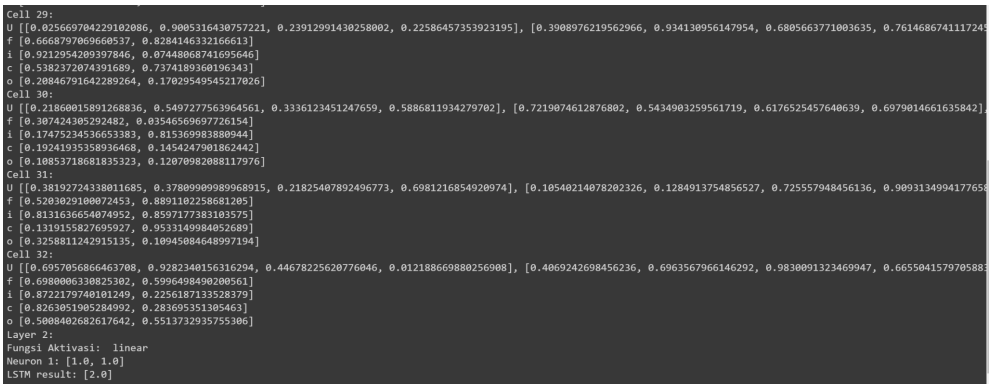
```
2
LSTM 32 true 4
linear 1
1 1
```

Gambar di atas merupakan arsitektur LSTM yang digunakan pada test. Arsitektur LSTM dijabarkan pada baris 2, dimana variable pertama merupakan jenis layernya(LSTM), variabel kedua merupakan banyak cell/timesteps dari layer, variabel ketiga merupakan isRandom layer, dimana jika bernilai true maka semua bobot layer akan diinisialisasi otomatis secara random, variabel terakhir adalah banyak fitur.



Gambar diatas merupakan alternatif arsitektur LSTM jika nilai isRandom false, sehingga baris inisialisasi LSTM akan diikuti oleh 8*jumlah cell baris yang merupakan nilai bobot pada layernya. Untuk setiap cellnya 4 baris pertama akan menjadi bobot U, yang diikuti oleh 1 baris f, 1 baris i, 1 baris c, dan 1 baris o.

2.3. Output



Keluaran yang didapat adalah keluaran dari LSTM selanjutnya dengan FFNN dengan fungsi aktivasi linear.

3. Pembagian Tugas

NIM	Nama	Tugas yang Dikerjakan
13519149	Syihabuddin Yahya Muhammad	LSTM Main, LSTM Layer
13519150	Imam Nurul Hukmi	RNN Gate, Cell
13519184	Muhammad Furqon	RNN Gate, Initialization Cell