

Tugas 2
IF4074 Pembelajaran Mesin Lanjutan

Forward Propagation pada *Long Short Term Memory* (LSTM)



Disusun oleh:

Kelompok 10

| | |
|---------------------------|----------|
| M. Rafli Zamzami | 13519067 |
| Aria Bachrul Ulum Berlian | 13519115 |
| Safiq Faray | 13519145 |

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2023

A. Struktur Program

Program yang kami buat adalah implementasi *Long Short Term Memory* (LSTM), yaitu pada proses *forward propagation*-nya saja pada tugas ini. Setiap layer yang kami buat merupakan operasi matriks, dan kami menggunakan library “numpy” untuk memproses hal tersebut.

Proses pembuatan model dan operasi lainnya dilakukan di file “main.py”. Untuk implementasi LSTM, kami letakkan di folder “neuralnetwork” sebagai modul untuk di-import.

Berikut adalah *class* yang kami buat untuk menyusun LSTM.

a. Dense Layer

Class “Dense” yang kami buat menerima parameter jumlah unit, panjang input, dan jenis fungsi aktivasi. Saat instansiasi, objek akan menginisiasi bobot secara acak beserta bias. Saat forward propagation dilakukan, input akan dijadikan array satu dimensi, yang kemudian dilakukan perkalian dot product antara input dan bobot yang ditranspose, kemudian dijumlahkan dengan bias.

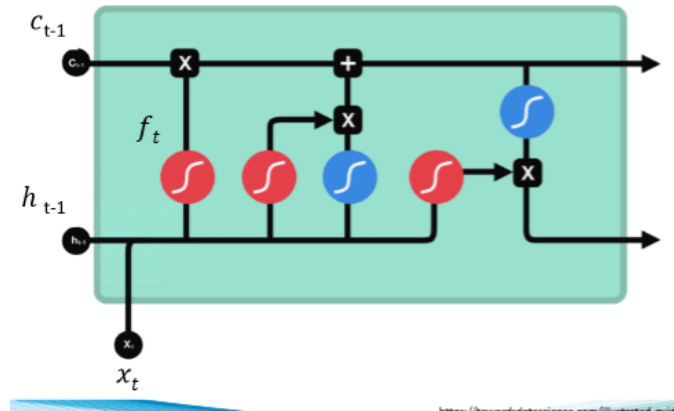
b. Activation

Untuk fungsi aktivasi, kami membuat Softmax, Sigmoid, dan Relu pada masing-masing class berbeda dengan menggunakan satu interface bernama Activation.

c. Model

Model yang kami buat diimplementasikan dengan class bernama “NN”. Class NN menerima parameter bentuk input dan layer-layernya. Objek yang telah diinstansiasi dapat ditambahkan layer baru dengan method add. Model yang dibuat akan melakukan semua proses forward propagate pada masing-masing layer jika dijalankan method forward_propagate() pada modelnya.

d. LSTM



Layer LSTM menerima dua argumen, yaitu `input_units` yang merupakan jumlah feature yang akan diproses, dan `num_units` yang merupakan jumlah hidden layer (LSTM cells). Pada masing-masing gate, akan diinisiasi random weights dan juga random bias. Data input akan di concat dengan h_{t-1} .

Pada forget gate, data yang telah di concat akan dikalikan dengan bobot forget, dijumlahkan dengan bias, lalu diaktivasi dengan sigmoid.

Pada input gate, data yang telah di concat akan dikalikan dengan bobot input, dijumlahkan dengan bias, lalu diaktivasi dengan sigmoid.

Pada output gate, data yang telah di concat akan dikalikan dengan bobot output, dijumlahkan dengan bias, lalu diaktivasi dengan sigmoid.

Pada cell hat, data yang telah di concat akan dikalikan dengan bobot cell hat, dijumlahkan dengan bias, lalu diaktivasi dengan tanh.

Nilai cell baru adalah penjumlahan antara forget gate dikali c_{t-1} dengan input gate dikali cell hat.

Nilai h_t atau logits adalah output gate dikali dengan tanh dari nilai cell baru.

B. Hasil

Berikut adalah hasil implementasi LSTM.

```
import numpy as np
from neuralnet.Layer import Layer
from neuralnet.Activation import *

class LSTM(Layer):
```

```

"""
LSTM

Args:
input_units (int): Number of input units.
num_units (int): Number of LSTM units.

"""

def __init__(self, input_units: int, num_units: int):
    super().__init__()
    self.type = "lstm"
    self.num_units = num_units
    self.input_units = input_units
    self.feature_map_shape = num_units

    # cell weights
    self.forget_weights = np.random.randn(
        self.input_units + self.num_units, self.num_units
    )
    self.forget_biases = np.random.randn(1, self.num_units)

    self.input_weights = np.random.randn(
        self.input_units + self.num_units, self.num_units
    )
    self.input_biases = np.random.randn(1, self.num_units)

    self.output_weights = np.random.randn(
        self.input_units + self.num_units, self.num_units
    )
    self.output_biases = np.random.randn(1, self.num_units)

    self.cell_hat_weights = np.random.randn(
        self.input_units + self.num_units, self.num_units
    )
    self.cell_hat_biases = np.random.randn(1, self.num_units)

    def __iter__(self):
        yield from {
            "type": self.type,
            "num_units": self.num_units,
            "input_units": self.input_units,
            "forget_weights": self.forget_weights.tolist(),
            "input_weights": self.input_weights.tolist(),
            "output_weights": self.output_weights.tolist(),
            "cell_hat_weights": self.cell_hat_weights.tolist(),
            "forget_biases": self.forget_biases.tolist(),
            "input_biases": self.input_biases.tolist(),
            "output_biases": self.output_biases.tolist(),
            "cell_hat_biases": self.cell_hat_biases.tolist(),
        }.items()

```

```

    def __str__(self):
        return json.dumps(dict(self), cls=MyEncoder,
ensure_ascii=False)

    def __repr__(self):
        return self.__str__()

    def forward_propagate(self, input: np.ndarray):
        """
        Forward propagates the input through the LSTM layer.

        """
        # print(input)

        n_feat = input.shape[1]

        # prev output and prev cell
        h_prev = np.zeros([1, self.num_units])
        c_prev = np.zeros([1, self.num_units])

        for i in range(len(input)):
            data = input[i].reshape(1, n_feat)
            input_and_prev_h = np.concatenate((data, h_prev),
axis=1)

            # forget
            fg = Sigmoid().calculate(
                np.matmul(input_and_prev_h,
self.forget_weights) + self.forget_biases
            )
            # input
            ig = Sigmoid().calculate(
                np.matmul(input_and_prev_h,
self.input_weights) + self.input_biases
            )
            # output
            og = Sigmoid().calculate(
                np.matmul(input_and_prev_h,
self.output_weights) + self.output_biases
            )
            # cell hat
            ch = Tanh().calculate(
                np.matmul(input_and_prev_h,
self.cell_hat_weights)
                + self.cell_hat_biases
            )
            # new cell
            cell = fg * c_prev + ig * ch

            logits = og * Tanh().calculate(cell)

```

```

        h_prev = logits
        c_prev = cell

    return logits

    def backpropagate(self, out: np.ndarray, learn_rate:
float):
        """
        Backpropagates the output through the LSTM layer.
        pass, not implemented

        """
        return super().backpropagate(out, learn_rate)

```

Berikut adalah susunan kode sebagai contoh pembuatan model dan hasil inferensi pada timestep 10.

File main.py

```

import cv2
from neuralnet import NN, ConvLayer
from neuralnet import Pooling
from neuralnet import Flatten
from neuralnet import Dense
from neuralnet import LSTM
from neuralnet import load_model
import numpy as np
from neuralnet import Trainer
from neuralnet import Preprocess
import pandas as pd

df = pd.read_csv("../data/lstm/Train_stock_market.csv")

data = np.array(df[["Low", "Open", "Volume", "High", "Close",
"Adjusted Close"]])

def create_sequences(data, seq_length):
    sequences = []
    targets = []
    data_len = len(data)
    for i in range(data_len - seq_length):
        seq_end = i + seq_length
        seq_x = data[i:seq_end]
        seq_y = data[seq_end]
        sequences.append(seq_x)
        targets.append(seq_y)
    return np.array(sequences), np.array(targets)

```

```

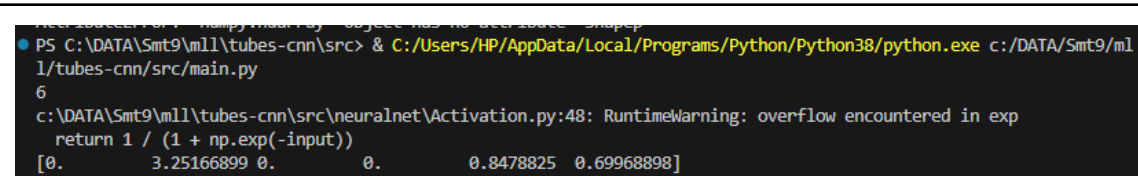
timestep = 10
lstm_cells = 15
X_train, y_train = create_sequences(data, timestep)

print(X_train.shape[2])

model = NN(X_train.shape)
model.add(LSTM(X_train.shape[2], lstm_cells))
flat_shape = model.layers[0].feature_map_shape
model.add(Dense(6, flat_shape, "relu"))
model.save_model("lstm.json", 4)
print(model.forward_propagate(X_train[0]))

```

Hasil inferensi adalah sebagai berikut.



```

PS C:\DATA\Smt9\ml1\tubes-cnn\src> & C:/Users/HP/AppData/Local/Programs/Python/Python38/python.exe c:/DATA/Smt9/ml1/tubes-cnn/src/main.py
6
c:\DATA\Smt9\ml1\tubes-cnn\src\neuralnet\Activation.py:48: RuntimeWarning: overflow encountered in exp
  return 1 / (1 + np.exp(-input))
[0. 3.25166899 0. 0. 0.8478825 0.69968898]

```

Proses forward propagation telah berhasil dilakukan. Namun, hasil inferensi masih salah. Hal ini dikarenakan belum diimplementasi proses training dengan backward propagation. Eksperimen tidak dilakukan karena tidak ada kesimpulan yang bisa ditarik, karena bobot masih random dan tidak ada proses pembelajaran.

C. Pembagian Tugas

| Anggota Kelompok | Pembagian Tugas |
|--------------------------------------|------------------------------|
| M, Rafli Zamzami (13519067) | Implementasi LSTM, save load |
| Aria Bachrul Ulum Berlian (13519115) | Implementasi LSTM, save load |
| Safiq Faray (13519145) | Implementasi LSTM, save load |

D. Lampiran

Github : <https://github.com/Haruray/tubes-cnn-lstm>

Google Colab :

- CNN :

https://colab.research.google.com/drive/1AlfQ78k9crFT_SJ3GAT-RGJAnl1T7cxO?usp=sharing

- LSTM :

<https://colab.research.google.com/drive/12bZXDpePYGasJOWWmCZLGvacw7G6gRtq?usp=sharing>

Youtube : <https://youtu.be/mngXs9dn2bs>