Laporan Tugas 2 IF4074 Pembelajaran Mesin Lanjut

Muhammad Iqbal Sigid - 13519152

Penjelasan Program

Implementasi feed forward LSTM dibuat pada python menggunakan library numpy untuk pemrosesan matrix. Program terdiri dari dua file, LSTM.py dan main.py. File LSTM.py berisi kelas LSTMLayer, InputLayer, DenseLayer, dan Model. File main.py berisi penggunaan model LSTM untuk menjalankan feed forward. Kelas LSTM hanya akan mengembalikan hasil pada timestep terakhir.

Kelas-Kelas

1. #### Kelas Input

   Kelas Input hanya berguna untuk mengetahui ukuran dari data yang diperlukan oleh layer berikutnya, yaitu layer LSTM. Kelas Input tidak memiliki fungsi khusus. Input untuk kelas ini sendiri hanya berupa input_shape.

2. #### Kelas Flatten

   Kelas ini akan mengubah matrix menjadi array satu dimensi. Kelas ini menerima parameter prev_layer sebagai referensi layer sebelumnya dan fungsi flatten yang menerima matrix input.

3. #### Kelas DenseLayer

   Kelas ini menerima parameter
   - n_neurons, berupa jumlah neuron pada dense layer
   - prev_layer, merujuk kepada layer sebelumnya
   - activation, berupa ‘sigmoid’ atau ‘relu’ untuk menentukan fungsi aktivasi yang akan digunakan

   Fungsi pada kelas ini adalah dense yang menerima input matrix. Fungsi ini akan melakukan perkalian matrix dengan bobot yang diinisialisasi secara random.

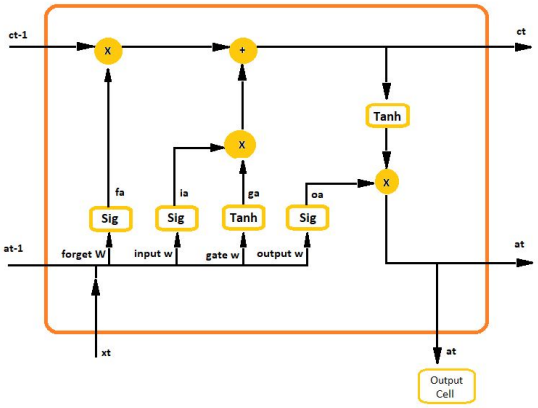
4. #### Kelas Model

   Kelas Model akan menyimpan layer-layer yang ditambahkan pada array network_layers. Kelas ini memiliki tiga fungsi. Fungsi add_layer yang menerima parameter layer, yang kemudian akan disimpan pada array. Fungsi summary yang akan mengeluarkan output

layer yang terdapat pada model. Serta fungsi feed_forward yang menerima input matrix gambar. Fungsi ini akan loop layer yang ada pada array dan melakukan feed forward matrix gambar terhadap layer yang ada pada model.

5. Kelas LSTMLayer

LSTMLayer menerima parameter hidden_unit berupa jumlah unit pada LSTM. LSTMLayer hanya menerima input layer sebagai layer sebelum. LSTMLayer akan diinisialisasi dengan menyimpan weight untuk forget gate, input gate, cell gate, dan output gate secara random. Weight untuk input (U) dan sel sebelumnya (W) disimpan pada satu array.



LSTMLayer memiliki fungsi lstm_cell yang akan melakukan operasi perkalian matriks antara input dengan weight yang ada pada layer. Pertama input akan di-concat dengan hasil aktivasi sel sebelumnya (at-1) menjadi concat_data.

- Pada forget gate, concat_data dikalikan dengan forget gate weight (fgw), kemudian digunakan aktivasi sigmoid.
- Pada input gate, concat_data dikalikan dengan input gate weight (igw) dan digunakan sigmoid, serta cocat_data dikalikan dengan cell gate weight (cgw) dan digunakan tanh. Kedua hasil kemudian dikalikan dengan perkalian matriks.
- Hasil dari input gate dijumlahkan dengan hasil perkalian forget gate dan hasil sel sebelumnya (ct-1)
- Pada output gate, concat_data dikalikan dengan output gate weight (ogw) dan digunakan sigmoid. Hasil ini dikalikan dengan cell state yang telah digunakan fungsi aktivasi tanh.

Contoh Hasil

main.py

```
import pandas as pd
import numpy as np
import LSTM

df = pd.read_csv('ETH-USD-Train.csv')

X_train = np.array(df[['Open','High','Low','Close','Volume']])
y_train = np.array(df[['Open','High','Low','Close','Volume']])

def prep_data(X_datain, y_datain, time_step):
    y_indices = np.arange(start=time_step, stop=len(y_datain))
    y_tmp = y_datain[y_indices]

    rows_X = len(y_tmp)
    X_tmp = np.zeros((rows_X, time_step, X_datain.shape[1]))
    for i in range(X_tmp.shape[0]) :
        X_tmp[i] = X_datain[i:i+time_step]
    return X_tmp, y_tmp

timestep = 32
X_train, y_train = prep_data(X_train, y_train, timestep)

input_layer = LSTM.Input(input_shape=X_train.shape)
LSTM_layer = LSTM.LSTMLayer(prev_layer=input_layer, hidden_units=8)
flatten_layer = LSTM.Flatten(prev_layer=LSTM_layer)
dense_layer = LSTM.DenseLayer(n_neurons=5, prev_layer=flatten_layer,
activation="linear")

model = LSTM.Model()
model.add_layer(LSTM_layer)
model.add_layer(flatten_layer)
model.add_layer(dense_layer)

res = model.feed_forward(X_train[0])
```

```
print(res)
```

Kode main.py di atas meng-input data csv menjadi pandas dataframe kemudian membaginya menjadi 32 timestep dengan fungsi prep_data. Untuk model, terdiri dari input layer, LSTM layer, flatten, dan dense dengan output 5 dimensi. Dari dataset yang telah dibagi, digunakan batch pertama dengan ukuran (32,5) atau 32 timestep dengan 5 dimensi per timestep.

Output

```
C:\Users\sigid\Documents\_Scripts\Python\LSTM>py main.py
C:\Users\sigid\Documents\_Scripts\Python\LSTM\LSTM.py:8: RuntimeWarning: overflow encountere
d in exp
  return 1.0 / (1 + np.exp(-1 * sop))
[-0.03745535  0.08728004  0.00905874  0.06236542 -0.03407359]
```

Hasilnya sebagai di atas. Hasil prediksi tentu saja tidak baik karena bobot pada model random. Namun, hal yang saya kurang yakin ada pada fungsi aktivasi dan perkalian bobot karena terlihat pada output terdapat overflow karena terdapat nilai yang besar pada pembagian.

Links

Google Colab:
https://colab.research.google.com/drive/1Wbc2-x3e3-sk3FFanT2Qppj0TV6ko9fX?usp=sharing
GitHub:
https://github.com/PlumStream24/LSTM
Video:
https://youtu.be/xKUbGdmfP3o