IF4074 PEMBELAJARAN MESIN LANJUT

Tugas 2 Long Short-Term Memory

Disusun oleh

Rozan Fadhil Al Hafidz	13520039
Adzka Ahmadetya Zaidan	13520127
M Syahrul Surya Putra	13520161

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

A. Penjelasan Kode Program

1. Kelas Layer
   Layer adalah sebuah *abstract class* dari layer-layer *neural network*.

2. Kelas LSTM
   LSTM adalah *child class* dari kelas Layer. Kelas ini melakukan pemrosesan data berurutan dengan menggunakan gates untuk melakukan kontrol pada *neural network*.

   ```
   LSTM(units: int)
   ```

   Kelas tersebut menerima argumen units, yaitu seberapa banyak neuron yang dibentuk dalam satu LSTM.

   Kelas ini memiliki beberapa method sebagai berikut.

   2.1. _compile(self, input_shape: int)
   Metode ini menginisialisasi input_shape, output_shape, weight, bias, recurrent_weight, timestep, nilai ct dan ht untuk LSTM. Secara default, nilai weight, bias, dan recurrent_weight diisi dengan nilai random.

   Terdapat 4 jenis bobot untuk melakukan kontrol besar nilai pada gate LSTM (forget, input, cell, dan output). Misalkan n adalah banyaknya unit pada LSTM dan m adalah dimensi input, maka bobot-bobot pada LSTM adalah sebagai berikut
   - Weight (U): berukuran n x m x 4
   - bias (b): berukuran n x 4
   - recurrent_weight: berukuran n x n x 4

   Recurrent_weight menyimpan bobot recurrent yang memengaruhi hubungan antara unit LSTM dari satu timestep ke timestep berikutnya. timestep menandai langkah waktu saat melakukan perhitungan di setiap timestep. ct (cell state) dan ht (hidden state) adalah memori long-term (cell state) dan output (hidden state) LSTM yang diperbarui di setiap timestep untuk memproses data berurutan.

   2.2. forward(self, input_data: np.ndarray)
   Metode ini digunakan untuk melakukan proses forward propagation pada layer LSTM. Forward menerima data input dalam bentuk array dan menghitung output dari layer LSTM. Proses ini melakukan iterasi setiap timestep dalam data input serta menghitung nilai gate dengan aktivasi sigmoid, cell state, dan hidden state berdasarkan operasi-operasi matriks dan aktivasi sigmoid dan tanh.

2.3. get_params_count(self)
Metode ini mengembalikan jumlah total parameter pada layer LSTM dengan menjumlahkan total bobot. Total bobot akan memiliki nilai

2.4. get_params(self) dan set_params(self, params)
get_params digunakan untuk mendapatkan parameter-parameter yang ada pada layer LSTM. Parameter tersebut yaitu weight (W), recurrent_weight (U), dan bias (b) untuk keempat gate (forget, input, cell, dan output). Sedangkan set_params digunakan untuk mengganti parameter model dengan parameter masukan.

3. Kelas Dense
Dense merupakan kelas turunan dari kelas Layer. Kelas ini digunakan untuk lapisan terhubung penuh dalam neural network.

```
Dense(input_size: int, units: int, activation: Literal["sigmoid", "relu"] = "relu")
```

Kelas Dense menerima beberapa argumen sebagai berikut:

- input_size: int
  Ukuran data input (jumlah fitur).

- units: int
  Jumlah neuron dalam lapisan.

- activation: str
  Fungsi aktivasi yang digunakan, dapat berupa "sigmoid" atau "relu" (default adalah "relu").

Setelah objek instansi dari kelas Dense dibuat, bobot dan bias lapisan akan diinisialisasi secara acak. Bobot diinisialisasi sebagai matriks 2D dengan ukuran (input_size, units). Bias diinisialisasi sebagai vektor 1D dengan ukuran (units).

Kelas Dense memiliki beberapa metode:

3.1. forward(input_data: np.ndarray) → np.ndarray
Forward melakukan proses forward pass pada lapisan Dense. Menerima parameter input_data berupa data masukan dalam bentuk array NumPy. Langkah-langkahnya termasuk transformasi linear dan menggunaan fungsi aktivasi. Metode ini mengembalikan output setelah transformasi linear dan aktivasi.

3.2. __activate(X: np.ndarray, activation: Literal["sigmoid", "relu"] = "relu") → np.ndarray
Activate digunakan untuk menerapkan fungsi aktivasi pada data masukan X. Parameter activationnya adalah jenis fungsi aktivasi yang digunakan ("sigmoid" atau "relu").

3.3. get_params dan set_params
get_params digunakan untuk mendapatkan parameter-parameter yang ada pada layer Dense. Parameter tersebut yaitu units, activation, weights, dan bias. Sedangkan set_params digunakan untuk mengganti parameter model dengan parameter masukan.

4. Kelas Sequential

Sequential adalah kelas yang digunakan untuk membangun dan melatih *neural network* dengan urutan berbagai layers.

```
class Sequential:
    def __init__(self, layers: list[object] = []):
        self.layers = layers
```

Kelas ini memiliki atribut layers yang merupakan daftar lapisan (layers) yang akan disusun dalam urutan tertentu.

4.1. add(layer: Layer) → None
Metode add digunakan untuk menambahkan lapisan baru (layer) ke dalam model.

4.2. forward(inp: np.ndarray) → np.ndarray
Metode forward menghitung hasil keluaran (output) dari model dengan meneruskan data input melalui semua lapisan (layers) dalam urutan yang ditentukan.

4.3. predict(img: np.ndarray) → np.ndarray
Metode predict digunakan untuk memprediksi hasil output dari model berdasarkan data gambar yang diberikan.

4.4. compile
Metode compile digunakan untuk menginisialisasi dan mengkonfigurasi model. Compile mengatur bentuk input dan output dari setiap lapisan (layers) dalam model.

4.5. summary
Metode summary digunakan untuk menampilkan ringkasan (summary) dari model, termasuk informasi tentang nama lapisan (layers), bentuk output, dan jumlah parameter yang digunakan dalam model.

4.6. fit(training_data, epochs: int, learning_rate: float) → None
Metode fit digunakan untuk melatih model dengan data pelatihan (training_data) sebanyak sejumlah epoch yang ditentukan. Fit menggunakan tingkat pembelajaran (learning rate) yang telah ditentukan.

4.7. save_model(filename)
Metode save_model digunakan untuk menyimpan model ke dalam sebuah file dengan format tertentu. save_model mencatat parameter dan jenis lapisan (layers) yang digunakan dalam model.

4.8. load_model(filename)
Metode load_model digunakan untuk memuat model dari file yang telah disimpan sebelumnya. load_model membaca parameter dan jenis lapisan (layers) dari file dan mengembalikan model yang sudah disimpan.

B. Contoh Hasil Prediksi

Sebelum dilakukan prediksi, dilakukan terlebih dahulu preprocessing data dengan mensplit data sesuai dengan timestep yang ada. Dan tidak lupa, kami menentukan suatu seed untuk np.random agar memudahkan melakukan perbandingan. Adapun implementasinya adalah sebagai berikut:

```
# split for getting next day result
def split_sequence(sequence, n_steps):
    X, y = list(), list()
    for i in range(len(sequence)):
        # find the end of this pattern
        end_ix = i + n_steps
        # check if we are beyond the sequence
        if end_ix > len(sequence)-1:
            break
        # gather input and output parts of the pattern
        seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
        X.append(seq_x)
        y.append(seq_y)
return np.array(X), np.array(y)
```

```
raw_train = pd.read_csv("TubesRNN/Train_stock_market.csv")
test = pd.read_csv("TubesRNN/Test_stock_market.csv")

train_date = raw_train["Date"]
train_date = pd.to_datetime(train_date)

train = raw_train.drop("Date", axis=1)

train['Date'] = train_date
test['Date'] = pd.to_datetime(test['Date'])

# print(test.head())
# print(train.head())

# Take relevant columns
train = train[['Date', 'Open', 'High', 'Low', 'Close']]
test = test[['Date', 'Open', 'High', 'Low', 'Close']]
```

Untuk timestepnya sendiri, kamu melakukan variasi menggunakan 3 timestep, yaitu 10, 35, dan 20. Adapun untuk model yang digunakan dan cara melakukan prediksi adalah sebagai berikut

```
# Create model
model = Sequential([])
model.add(LSTM(units=64))
model.add(Dense(units=32, activation="relu"))
model.add(Dense(units=4)) # 4 output

input_shape = (X_test.shape[1], X_test.shape[2]) # (n_steps, number of features)

# Compile
model.compile(input_shape, from_load=False)
```

```
predictions = []
rmses = []
for i in range(len(X_test)):
    print()
    # print("INPUT: ", X_test[i])
    prediction = model.forward(X_test[i])
    print("PREDICTION: ", prediction)
    print("ACTUAL: ", y_test[i])

    mse = (y_test[i] - prediction) ** 2
    rmse = np.sqrt(np.mean(mse))
    predictions.append(prediction)
    rmses.append(rmse)

    print("Experiment - Average Root Mean Squared Error (RMSE):", np.mean(rmses))
```

Dilakukanlah prediksi berdasarkan data yang telah dipreprocess sesuai dengan masing-masing time step menggunakan model yang telah kami buat. Hasilnya dapat dilihat sebagai berikut:

- Timestep 10

```
PREDICTION: [41.2806934 10.01098199 0. 3.78076974] # Prediction terakhir
ACTUAL: [2.66 2.73 2.66 2.7 ] # Actual terakhir
Experiment 1 - Average Root Mean Squared Error (RMSE): 11.737717138952545
```

- Timestep 35

```
PREDICTION: [16.38445439 6.59553641 0. 0. ] # Prediction terakhir
ACTUAL: [2.66 2.73 2.66 2.7 ] # Actual terakhir
Experiment 2 - Average Root Mean Squared Error (RMSE): 12.465485066918605
```

- Timestep 25

```
PREDICTION: [41.98397333 8.98276555 0. 2.8505238 ] # Prediction terakhir
ACTUAL: [2.66 2.73 2.66 2.7 ] # Actual terakhir
Experiment 3 - Average Root Mean Squared Error (RMSE): 10.402818866487136
```

C. Pembagian Tugas dalam Kelompok

NIM	Nama	Tugas
13520039	Rozan Fadhil Al Hafidz	LSTM. Save and Load. Eksperimen dengan dataset. Penjelasan pada laporan
13520127	Adzka Ahmadetya Zaidan	LSTM. Save and Load. Eksperimen dengan dataset. Penjelasan pada laporan
13520161	M Syahrul Surya Putra	Dense. LSTM. Save and Load. Eksperimen dengan dataset. Penjelasan pada laporan

D. Kesimpulan

Telah dilakukan implementasi secara manual untuk layer LSTM dan Dense dengan menggunakan kelas Sequential untuk melakukan pelatihan *neural network* secara berurutan. Implementasi yang dilakukan hanya pada forward passing tanpa backward. Selain itu, dilakukan juga proses untuk save dan load model berdasarkan layer dan parameternya. Implementasi untuk summary model juga dilakukan untuk menunjukkan layer-layer yang terdiri pada model, bentuk outputnya, dan jumlah parameter.

Dilakukan juga 3 eksperimen pada data test dengan mengganti-ganti jumlah timestepnya. Evaluasi eksperimen dilakukan dengan menghitung besar rata-rata root mean squared error. Berdasarkan hasil eksperimen, eksperimen 3 dengan timestep 25 mendapatkan nilai RMSE paling kecil, yaitu 10.4. Dapat dilihat juga bahwa nilai prediction yang didapat cukup jauh dari test seharusnya karena model tidak dilakukan training tanpa adanya implementasi back-propagation dan weight-weight yang di inisiasi secara random.

E. Lampiran

Link Google Colab

1. CNN: https://colab.research.google.com/drive/1OgyLnbu4RXHkx484JLmitOvEwOiVj6D2#scrollTo=adUIAo6OqXI5
2. RNN (LSTM): https://colab.research.google.com/drive/1Bh7R9dxntX1uujwmtgTRK1gq3qbdFbHF?usp=sharing

Link Repository Github:
https://github.com/rozanfa/IF4074-13520039-13520127-13520161

Link Video Demo:
https://youtu.be/6UTUJIIAsmA