3. RNN 문제

1번째 문제는 손실값을 최소화하는 optimizer와 loss 함수를 찾는 문제입니다. Optimizer는 adam이고 loss 함수는 mse인 것 같습니다.

2번째 문제는 손실값을 개선시키는 문제입니다. Optimizer를 adam으로 loss 함수를 mse로 했을 때 학습 시간은 5.84초가 나오고 최종 손실값은 0.0016나왔습니다.

### Keras RNN으로 예약자 수 예측

1. 패키지 수입 및 파라미터 설정

# 수입

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from time import time

from sklearn.preprocessing import MinMaxScaler

from keras.layers import Dense

from keras.layers import LSTM, InputLayer

from keras.models import Sequential

from sklearn.model\_selection import train\_test\_split

# 파라미터

MY\_PAST = 12

MY\_SPLIT = 0.8

MY\_HIDDEN = 300

MY\_SHAPE = (MY\_PAST, 1)

MY\_EPOCH = 300

2. 데이터 처리

# 데이터 읽기

raw = pd.read\_csv('/content/drive/MyDrive/data/airline.csv',

header=None,

usecols=[1])

# 샘플 출력

print('데이터 원본')

print(raw.head(15))

print(raw.describe())

plt.figure(figsize=(10, 5))

plt.plot(raw)

# 데이터 정규화

# scaler = StandardScaler()

scaler = MinMaxScaler()

s\_data = scaler.fit\_transform(raw)

print('데이터 형식:', type(s\_data))

print(s\_data.flatten())

plt.figure(figsize=(10, 5))

plt.plot(s\_data)

# 데이터 분할

data = [ ]

for i in range(len(s\_data) - MY\_PAST):

data.append(s\_data[i: i + MY\_PAST + 1])

print('한 묶음의 크기:', len(data[0]))

print('총 묶음의 수:', len(data))

print('첫번째 묶음 내용:\n', data[0])

# 데이터를 numpy로 전환

print(type(data))

data = np.array(data)

print(type(data))

# 묶음들을 임의로 섞음

np.random.shuffle(data)

print(data.shape)

# 데이터를 입력과 출력으로 분할

X\_data = data[:, 0:MY\_PAST]

Y\_data = data[:, -1]

print('입력 데이터 모양:', X\_data.shape)

print('출력 데이터 모양:', Y\_data.shape)

# 데이터 4분할

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X\_data,

Y\_data,

train\_size=MY\_SPLIT,

shuffle=False)

# 최종 데이터 모양 출력

print('학습용 입력 데이터 모양:', X\_train.shape)

print('학습용 출력 데이터 모양:', Y\_train.shape)

print('평가용 입력 데이터 모양:', X\_test.shape)

print('평가용 출력 데이터 모양:', Y\_test.shape)

3. 인공신경망 구현

# RNN 구현

model = Sequential()

model.add(InputLayer(input\_shape=MY\_SHAPE))

model.add(LSTM(units=MY\_HIDDEN))

model.add(Dense(units=1,

activation='sigmoid'))

print('RNN 요약')

model.summary()

4. 인공신경망 학습

# RNN 학습

model.compile(optimizer='adam',

loss='mse')

begin = time()

model.fit(x=X\_train,

y=Y\_train,

epochs=MY\_EPOCH,

verbose=0)

end = time()

print('학습 시간: {:.2f}'.format(end-begin))

5. 인공신경망 평가

# RNN 평가

score = model.evaluate(x=X\_test,

y=Y\_test,

verbose=0)

print('최종 손실값:', score)

6. 인공신경망 예측

# RNN으로 예측

pred = model.predict(x=X\_test)

# 예측 결과 및 정답 그래프 그리기

plt.figure(figsize=(10, 7))

plt.plot(pred, 'r')

plt.plot(Y\_test, 'g')

plt.legend(['prediction', 'truth'])

plt.show()

print('역전환 전:\n', pred)

# 데이터 역전환

pred = scaler.inverse\_transform(pred)

print('역전환 후:\n', pred)

print(pred.shape)

# squeeze 결과

tmp = np.squeeze(pred).astype(int)

print('squeeze 후:', tmp)

print(tmp.shape)

# 정답 출력

print('역전환 전 정답:\n', Y\_test)

Y\_test = scaler.inverse\_transform(Y\_test)

tmp = np.squeeze(Y\_test).astype(int)

print('역전환 후 정답:\n', tmp)

# 예측 결과 및 정답 그래프 그리기

plt.figure(figsize=(10, 7))

plt.plot(pred, 'r')

plt.plot(Y\_test, 'g')

plt.legend(['prediction', 'truth'])

plt.show()