기온 예측 하기

- 실제로 이 방법으로 기온을 예측 하는것은 불가능해요.
- 데이터가 너무 적고, 기온에 영향을 미치는 특성들에 대한 데이터가 하나도 없기 때문이에요.
- 하지만 RNN은 이렇게 돌아간다를 보기위해서 진행하는 실습이니까 진행해주세요.

In [ ]:

```
# 사용할 모듈들 import 해줄게요.
import numpy as np
import pandas as pd
from pprint import pprint

# 기온 그래프 그릴때 사용할거에요.
from matplotlib import pyplot as plt

# 이번에는 이정도만 사용할거에요.
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, SimpleRNN, LSTM
```

In [ ]:

```
# Pandas말고 일반 파이썬으로 데이터 읽어 볼게요.
data_path = '../data/'
file_name = '기온.csv'

# open() 함수는 파이썬 내장함수로 파일 읽고 쓸때 사용하는 함수에요.
# open(file, mode, encoding) 으로 구성되어 있어요.
# file_path는 읽어나 쓸 파일의 위치와 이름이에요.
# mode 는 'r'이 읽기, 'w'가 쓰기에요.
# encoding은 주로 'utf-8'을 사용하고, 가끔 'cp949'를 사용해요.
# mode='r'에서 파일을 읽을때 주로 read(), readlines(), readline()을 사용해요.
# .read()는 한번에 모든 문서를 다 읽는 함수에요.
# .readlines()는 한번에 모든 문서를 다 읽는데 한줄씩 list로 저장되어있어요.
# .readline()은 한번에 한줄을 읽는 함수에요.

# 먼저 모든 라인을 읽고 정상적으로 읽어졌는지 확인할게요.
lines = open(file=data_path+file_name, mode='r', encoding='cp949').readlines()
print(lines[0])
print(lines[-1])
```

In [ ]:

```
# 다음으로 데이터를 정리해줄거에요.

# 사용할 데이터를 저장할 list 생성할게요.
data = []

# lines에는 모든 line이 들어있으니까, 하나씩 읽으면서 진행할게요.
# for line in lines 는 lines안에 있는 요소 하나를 line으로 받아오는거에요.
# for i in range(len(lines)): 와 동일한 결과를 보여줘요.
#     line = lines[i]
# 첫번째 줄에는 날짜, 지점 같이 컬럼명이 있으니까 lines[1:] (slicing)으로 넘어갈게요.
for line in lines[1:]:
    # 하나의 라인은 위 출력한것처럼 ','로 나눠져 있어요.
    # line.strip().split(',') 사용해서 전부 ','로 나눠줄게요.
    # .strip()은 line의 좌우 공백이랑 줄바꿈문자를 지워주는 함수에요.
    date, _, avg_temp, min_temp, max_temp = line.strip().split(',')

    # date는 2019-12-31 의 포맷으로 되어있으니까,
    # .split('-') 사용해서 년월일로 나눠줄게요.
    y, m, d = date.strip().split('-')

    # 정리한 정보를 data list에 넣어줄게요.
    # 중간에 정보가 없는 line이 있어요.
    # try-except로 해당 라인은 넘어가도록 할게요.
    # try-except는 try문에서 에러가 발생했을 때,
    # except문으로 넘어가서 계속 처리를 진행하는 구문이에요.
    try:
        data.append([np.int(y), m, d,
                     np.float(avg_temp), np.float(min_temp), np.float(max_temp)])
    except:
        print([y, m, d,
                     avg_temp, min_temp, max_temp])
```

In [ ]:

```
# 위에서 정리한 데이터로 DataFrame 생성하고 확인할게요.
df = pd.DataFrame(data)
df.columns = ['year', 'month', 'day', 'avg_temp', 'min_temp', 'max_temp']
df
```

In [ ]:

```
# 10년동안 기온이 어떻게 되었는지 pandas 내부 plot으로 그려볼게요.
df[['avg_temp', 'min_temp', 'max_temp']].plot(subplots=True)
```

In [ ]:

```
# DataFrame을 년도별로 잘라서 list에 저장하도록 할게요.
# min()함수는 python 내장함수로, ()안의 값 중에 최소값을 반환해줘요.
# max()함수도 python 내장함수로, ()안의 값 중에 최대값을 반환해줘요.
years = []
for year in range(min(df['year']), max(df['year'])+1):
    years.append(df[df['year']==year])
print(years)
```

In [ ]:

```
# 평균 기온의 변화를 plot으로 그려볼게요.
for year in years:
    year['avg_temp'].plot()
```

In [ ]:

```
# 최저 기온의 변화를 plot으로 그려볼게요.
for year in years:
    year['min_temp'].plot()
```

In [ ]:

```
# 최고 기온의 변화를 plot으로 그려볼게요.
for year in years:
    year['max_temp'].plot()
```

In [ ]:

```
# 데이터를 학습데이터랑 평가데이터로 나눌게요.
# 학습데이터는 2010년~2018년의 데이터이고,
# 평가데이터는 2019년의 데이터에요.
# dataFrame.values[:, 3:]에서 앞의 : 은 전체 열을 의미하고,
# 뒤에 3: 은 각 열에서 3번째 행부터 끝가지를 가져오겠다는 의미에요.
train_dataset = df[df['year'] < 2019].dropna().values[:, 3:]
test_dataset = df[df['year'] == 2019].dropna().values[:, 3:]

# 학습데이터랑 평가데이터로 나눈 데이터를
# 다시 평균, 최소, 최대 기온 3개로 나눌게요.
avg_data_train_set = train_dataset[:, 0]
min_data_train_set = train_dataset[:, 1]
max_data_train_set = train_dataset[:, 2]

avg_data_test_set = test_dataset[:, 0]
min_data_test_set = test_dataset[:, 1]
max_data_test_set = test_dataset[:, 2]
```

In [ ]:

```
# 평균 기온의 학습데이터를 matplotlib의 scatter로 그려볼게요.
x = range(len(avg_data_train_set))
plt.scatter(x, avg_data_train_set)
plt.show()
```

In [ ]:

```
# 평균 기온의 평가데이터를 matplotlib의 scatter로 그려볼게요.
x = range(len(avg_data_test_set))
plt.scatter(x, avg_data_test_set)
plt.show()
```

In [ ]:

```
# 데이터를 입력이랑 타겟으로 나누는 함수를 작성할게요.
# data는 전체 데이터를 뜻하고
# seek_step은 입력으로 몇개의 데이터를 사용할거냐를 뜻해요.
# features는 하나의 데이터가 몇개의 수치로 이루어져있나를 뜻해요.
def div_xy(data, seek_step, features):
    x, y = [], []
    for i in range(len(data)-seek_step-1):
        x.append(data[i:i+seek_step])
        y.append(data[i+seek_step])
    return np.array(x, dtype=np.float).reshape(-1, seek_step, features), np.array(y, dtype=np.f
loat).reshape(-1, features)
```

In [ ]:

```
# 이번 실습에서 사용할 seek_step과 features 설정이에요.
# 이번 실습에서 seek_step은 예를들어 내일의 기온을 예측하기위해서
# 몇 일전까지의 기온을 볼것일가를 뜻해요. 총 5일의 데이터를 사용하겠다는 뜻이에요.
# features는 예를들어, 하루의 평균 기온 데이터는 몇개인가를 뜻해요.
# 하루 평균 기온은 하나뿐이에요.
seek_step = 5
features = 1
```

In [ ]:

```
# 작성한 함수를 사용해서 데이터를 입력이랑 타겟으로 나눠주고, 확인할게요.
x_train, y_train = div_xy(avg_data_train_set, seek_step, features)
x_test, y_test = div_xy(avg_data_test_set, seek_step, features)
# shape 한번 봐주세요.
print(x_train.shape)
print(y_train.shape)
print('''--------------------------------------''')

# 즉, [-4.8 0.4 -0.4 -0.4 -2] 다음에 -4.2가 나왔었다는 의미에요.
# range()안 숫자 조금 늘려서 확인해 보실게요.
for i in range(1):
    print(x_train[i], end=' : ')
    print(y_train[i])
```

In [ ]:

```
# loss 확인용 클래스 작성할게요.
class LossHistory(keras.callbacks.Callback):
    def init(self):
        self.losses = []

    def on_epoch_end(self, batch, logs={}):
        self.losses.append(logs.get('loss'))
```

In [ ]:

```
# 모델 생성하고 컴파일 진행할게요.
history = LossHistory()
history.init()

model = Sequential()
model.add(SimpleRNN(units=256))#, batch_input_shape=(5,seek_step,features)))
model.add(Dense(features))
model.compile(loss='mse', optimizer='RMSprop', metrics=['accuracy'])
```

In [ ]:

```
# 평균 기온 데이터 모델에 학습할게요.
model.fit(x_train, y_train, epochs=10, batch_size=5, callbacks=[history])
```

In [ ]:

```
# Loss를 plot으로 한번 확인해 볼게요.
loss = history.losses
epochs = range(len(loss))
plt.plot(epochs, loss)
plt.show()
```

In [ ]:

```
# 모델에 학습시킨 데이터로 모델을 한번 확인해 볼게요.
y_hat = []
for trained_x in x_train:
    trained_x = trained_x.reshape(1, seek_step, features)
    y_hat.append(model.predict(trained_x))

length = range(len(y_hat))
plt.plot(length, y_train, np.squeeze(y_hat))
plt.show()
```

In [ ]:

```
# 테스트 데이터로도 모델 확인 해볼게요.
y_hat = []
for test_x in x_test:
    test_x = test_x.reshape(1, seek_step, features)
    y_hat.append(model.predict(test_x))
length = range(len(y_hat))
plt.plot(length, y_test, np.squeeze(y_hat))
plt.show()

# 마음에 안드니까 학습을 한번 더해요. model.fit()이 작성된 셀부터
# 현재 셀까지 한번더 실행시켜 주세요.
```

In [ ]:

```

```

In [ ]:

```
# 모델 SVG나 summary로 구조 구성 확인해보세요.
# 마음에 안드시면 모델 수정하셔도 되요.
```

In [ ]:

```
# 나머지는 평균에서 최저, 최고 기온으로 바꾸고 실험하는거랑
# 한번에 평균, 최저, 최고를 전부 사용하는 실험이에요.
# 한번 진행해보세요.
```

In [ ]:

```

```

In [ ]:

```
# 최저 기온 데이터 -> 입력, 출력 으로 나누기
x_train, y_train = div_xy(min_data_train_set, seek_step, features)
x_test, y_test = div_xy(min_data_test_set, seek_step, features)
```

In [ ]:

```
# 새로운 모델 생성 및 컴파일

history = LossHistory()
history.init()

model = Sequential()
model.add(SimpleRNN(units=256, batch_input_shape=(1,seek_step,features)))
model.add(Dense(features))
model.compile(loss='mse', optimizer='RMSprop', metrics=['accuracy'])
```

In [ ]:

```
# 최저 기온 데이터 모델에 학습
model.fit(x_train, y_train, epochs=10, batch_size=1, callbacks=[history])
```

In [ ]:

```
# Loss 확인
loss = history.losses
epochs = range(len(loss))
plt.plot(epochs, loss)
plt.show()
```

In [ ]:

```
# 학습한 데이터로 모델 확인
y_hat = []
for trained_x in x_train:
    trained_x = trained_x.reshape(1, seek_step, features)
    y_hat.append(model.predict(trained_x))
length = range(len(y_hat))
plt.plot(length, y_train, np.squeeze(y_hat))
plt.show()
```

In [ ]:

```
# 테스트 데이터로 모델 확인
y_hat = []
for test_x in x_test:
    test_x = test_x.reshape(1, seek_step, features)
    y_hat.append(model.predict(test_x))
length = range(len(y_hat))
plt.plot(length, y_test, np.squeeze(y_hat))
plt.show()
```

In [ ]:

```

```

In [ ]:

```
# 최고 기온 데이터 -> 입력, 출력으로 나누기
x_train, y_train = div_xy(max_data_train_set, seek_step, features)
x_test, y_test = div_xy(max_data_test_set, seek_step, features)
```

In [ ]:

```
# 새로운 모델 생성 및 컴파일
history = LossHistory()
history.init()

model = Sequential()
model.add(SimpleRNN(units=256, batch_input_shape=(1,seek_step,features)))
model.add(Dense(features))
model.compile(loss='mse', optimizer='RMSprop', metrics=['accuracy'])
```

In [ ]:

```
# 최고 기온 데이터 모델에 학습
model.fit(x_train, y_train, epochs=10, batch_size=1, callbacks=[history])
```

In [ ]:

```
# Loss 확인
loss = history.losses
epochs = range(len(loss))
plt.plot(epochs, loss)
plt.show()
```

In [ ]:

```
# 학습한 데이터로 모델 확인
y_hat = []
for trained_x in x_train:
    trained_x = trained_x.reshape(1, seek_step, features)
    y_hat.append(model.predict(trained_x))
length = range(len(y_hat))
plt.plot(length, y_train, np.squeeze(y_hat))
plt.show()
```

In [ ]:

```
# 테스트 데이터로 모델 확인
y_hat = []
for test_x in x_test:
    test_x = test_x.reshape(1, seek_step, features)
    y_hat.append(model.predict(test_x))
length = range(len(y_hat))
plt.plot(length, y_test, np.squeeze(y_hat))
plt.show()
```

In [ ]:

```

```

In [ ]:

```
# 평균, 최저, 최고 기온 한번에 사용하기
dataset = df.dropna().values[:, 3:]
train_dataset = df[df['year'] < 2019].dropna().values[:, 3:]
test_dataset = df[df['year'] == 2019].dropna().values[:, 3:]
print(train_dataset[:5])
print(test_dataset[:5])
```

In [ ]:

```
# Sequence 길이와 feature 개수 설정 및 데이터 나누기
seek_step = 5
features = 3

x_train, y_train = div_xy(train_dataset, seek_step, features)
x_test, y_test = div_xy(test_dataset, seek_step, features)
print(x_train.shape)
print(y_train.shape)
for i in range(5):
    print(x_train[i], end=' : ')
    print(y_train[i])
```

In [ ]:

```
# 새로운 모델 생성 및 컴파일
history = LossHistory()
history.init()

model = Sequential()
model.add(SimpleRNN(units=256, batch_input_shape=(1,seek_step,features)))
model.add(Dense(features))
model.compile(loss='mse', optimizer='RMSprop', metrics=['accuracy'])
```

In [ ]:

```
# (평균, 최저, 최고) 기온 모델에 학습
model.fit(x_train, y_train, epochs=10, batch_size=1, callbacks=[history])
```

In [ ]:

```
# 학습한 데이터로 모델 확인
y_hat = []
for trained_x in x_train:
    trained_x = trained_x.reshape(1, seek_step, features)
    y_hat.append(model.predict(trained_x))
length = range(len(y_hat))
plt.plot(length, y_train, np.squeeze(y_hat))
plt.show()
```

In [ ]:

```
# 테스트 데이터로 모델 확인
y_hat = []
for test_x in x_test:
    test_x = test_x.reshape(1, seek_step, features)
    y_hat.append(model.predict(test_x))
length = range(len(y_hat))
plt.plot(length, y_test, np.squeeze(y_hat))
plt.show()
```

In [ ]:

```
# 한번에 학습한 모델이 예측한 평균기온 확인
plt.plot(length, np.squeeze(y_test)[:, 0], np.squeeze(y_hat)[:,0])
plt.show()
```

In [ ]:

```
# 한번에 학습한 모델이 예측한 최저기온 확인
plt.plot(length, np.squeeze(y_test)[:, 1], np.squeeze(y_hat)[:,1])
plt.show()
```

In [ ]:

```
# 한번에 학습한 모델이 예측한 최고기온 확인
plt.plot(length, np.squeeze(y_test)[:, 2], np.squeeze(y_hat)[:,2])
plt.show()
```