인공지능

과제 01

컴퓨터공학부 컴퓨터공학전공

202001845 홍인혜

[문제]

특정지역(천안, 서울 등)의 최근 1 년치 온도 데이터(월별 또는 일별)를 구하여,
n-차 다항식 회귀(regression) 문제를 텐서플로를 이용하여 구현하고, 예측해보시오.

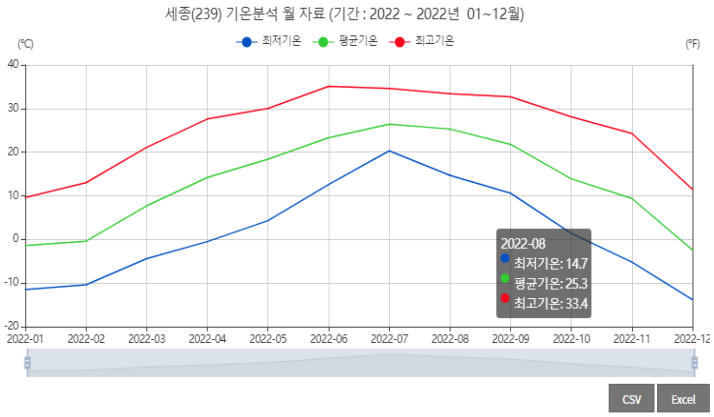
[본문]

[본문 -1 데이터 수집 방법]

우선 특정 지역을 고른 후 그 지역의 최근 1 년치의 일 또는 월별 온도 데이터를 구해야 합니다. 특정 지역의 온도 데이터를 구하기 위하여 기상청의 기상 자료 개방 포털(기후통계분석)에 들어가 데이터를 수집하였습니다.

URL 주소: https://data.kma.go.kr/climate/RankState/selectRankStatisticsDivisionList.do)

제가 현재 거주하고 있는 지역은 천안 밑에 위치한 세종특별자치시입니다. 따라 저는 세종시의 온도 데이터를 구하고, 사용하기로 결정하였습니다. 다만 2023 년이 끝나기에는 아직 몇 달의 시간이 남아있기 때문에 2023 년의 데이터가 아닌 2022 년의 데이터를 수집하였습니다.



지점번호	지점명	일시	평균기온(℃	평균최고기온
239	세종	Jan-22	-1.4	4.2
239	세종	Feb-22	-0.4	5.2
239	세종	Mar-22	7.7	13.9
239	세종	Apr-22	14.2	21.6
239	세종	May-22	18.4	25.3
239	세종	Jun-22	23.3	28.7
239	세종	Jul-22	26.4	30.9
239	세종	Aug-22	25.3	29.2
239	세종	Sep-22	21.8	27.2
239	세종	Oct-22	13.9	20
239	세종	Nov-22	9.4	16.1
239	세종	Dec-22	-2.4	2.6

데이터를 저장하기 위해 왼쪽 그래프 하단의 Excel 을 누르면 오른쪽 그림처럼 세종시의 1 년동안의 월별 지역, 일시, 평균기온, 평균 최고 기온 등의 데이터가 나열되어 있습니다. 지점번호 지점 명 등과 같은 데이터는 필요가 없고 일시와 평균기온만 필요하기 때문에 적당히 데이터를 수정하여 일시와 평균기온만이 있는 엑셀 데이터 sejong_Data.xlsx 를 만들어 주었습니다.

```
import pandas as pd
import os
os.chdir('D:/')
data = pd.read_excel('sejong_Data.xlsx')
data
```

	date	temperature
0	2022-01-01	-1.4
1	2022-02-01	-0.4
2	2022-03-01	7.7
3	2022-04-01	14.2
4	2022-05-01	18.4
5	2022-06-01	23.3
6	2022-07-01	26.4
7	2022-08-01	25.3
8	2022-09-01	21.8
9	2022-10-01	13.9
10	2022-11-01	9.4
11	2022-12-01	-2.4

이제 해당 엑셀 데이터를 파이썬으로 불러와야 합니다. 파이썬에 데이터를 불러오기 위해 이 두 라이브러리를 더 설치하여 불러왔습니다. (import pandas as pd, import os) 정상적으로 데이터를 불러올 수 있는 것을 확인하였기 때문에 이 데이터를 이용해 n 차 다항식 회귀 학습 결과 그래프 문제를 해결하고자 합니다.

[본문 – 2 구현설명]

```
import tensorflow as tf
import pandas as pd
import os
import numpy as np
import matplotlib.pyplot as plt
```

우선 문제 해결에 필요한 4 가지 라이브러리를 불러옵니다. 딥러닝 및 기계 학습 프레임워크 라이브러리 TensorFlow, 데이터 조작과 분석을 위한 라이브러리 Pandas, 수치 계산을 위한 라이브러리 Numpy, 데이터 시각화를 위한 라이브러리 Matplotlib 입니다.

```
MSE =
tf.keras.losses.MeanSquaredError()
```

선형 대수 평균 제곱 오차를 나타내는 손실 함수를 정의합니다.

```
def mse_loss():
    y = tf.zeros_like(x,
dtype=tf.float64)
    for i in range(W.shape[0]):
        y += W[i] * (x ** (i + 1))
    y += b  # bias
    return MSE(y, t)
```

모델의 손실을 계산하는 함수인 mes_loss()를 정의합니다. W 는 모델의 가중치 벡터, b 는 편향입니다. 모델은 x 와 W 를 이용하여 다항식 함수를 표현하고 b 를 추가하여 최종 예측 값을 계산합니다. 실세 값 t 와 모델의 예측 값 y 를 MSE(y, t)를 이용해 계산 후 반환합니다.

```
os.chdir('D:/')
data =
pd.read_excel('sejong_Data.xlsx')
x = np.arange(len(data))
t = data["temperature"].values
```

데이터를 불러옵니다.

```
n = 1  # n 차 다항식 회귀
W = tf.Variable(tf.random.normal([n],
dtype=tf.float64))
b = tf.Variable(tf.random.normal([],
dtype=tf.float64))
```

n 은 다항식의 차수를 나타냅니다. 현재는 1 로 설정되어 있으나 2, 3 등으로도 바꿀 수 있습니다. N 차 다항식의 가중치 W 와 편향 b 의 훈련변수를 생성합니다.

```
opt =
tf.keras.optimizers.Adam(learning_rat
e=0.01)
```

최적화 객체 opt 를 생성합니다.

```
loss_list = []
for epoch in range(EPOCH):
    opt.minimize(mse_loss,
var_list=[W, b])

    loss = mse_loss().numpy()
    loss_list.append(loss)

    if not epoch % 100:
        print("epoch={}:
loss={}".format(epoch, loss))
```

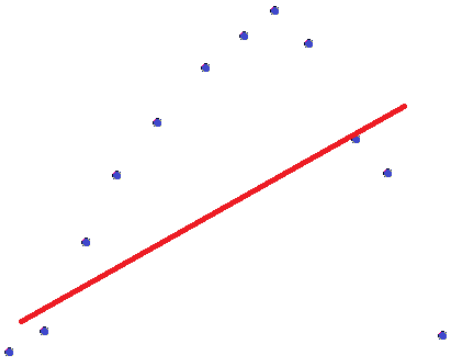
opt_minimize 로 손실함수 mse_loss 에서 변수 리스트 [W, b]에 대해 최적화합니다. 손실 함수 mess_loss 는 인수가 없어야 합니다.

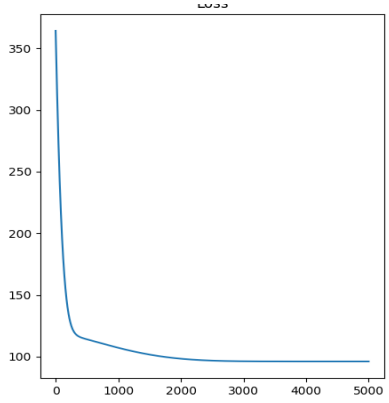
```
plt.figure(figsize=(11, 6))
plt.subplot(1, 2, 1)
plt.plot(loss_list)
plt.title('Loss')
plt.subplot(1, 2, 2)
plt.scatter(x, t, label='Original
Data')
t_pred = tf.zeros_like(x,
dtype=tf.float64)
for i in range(W.shape[0]):
    t_pred += W[i] * (x ** (i + 1))
t_pred += b  # bias
plt.plot(x, t_pred, 'red',
label='Predicted Curve')
plt.legend()
plt.show()
```

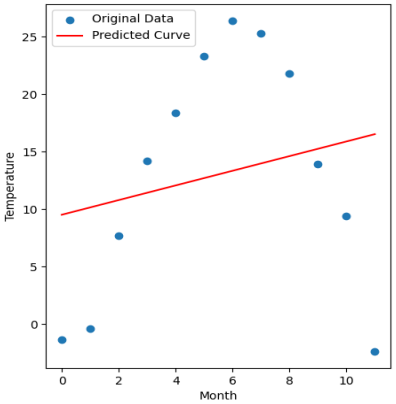
손실 그래프 출력

[실험 - 1 차, 2 차, 3 차 등의 다항식 모델 예측과 화면 덤프를 이용한 구현 결과 설명]

1) 1 차 다항식 모델예측과 결과

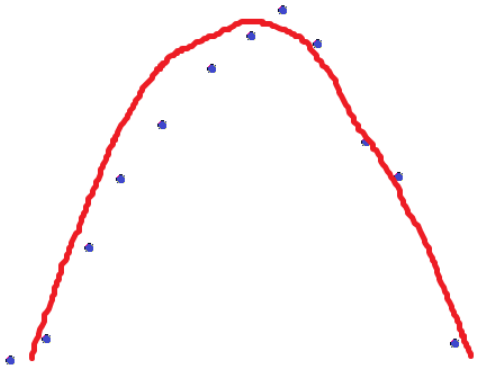


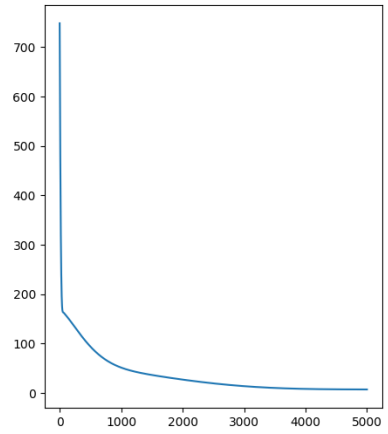


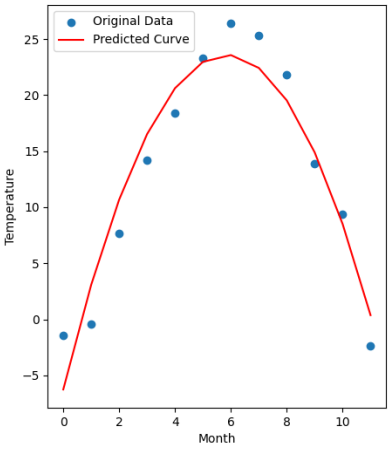


왼쪽 그림과 같은 상승 직선 그래프를 예측하였습니다. 결과 또한 일차함수 상승 그래프입니다.

2) 2 차 다항식 모델예측과 결과

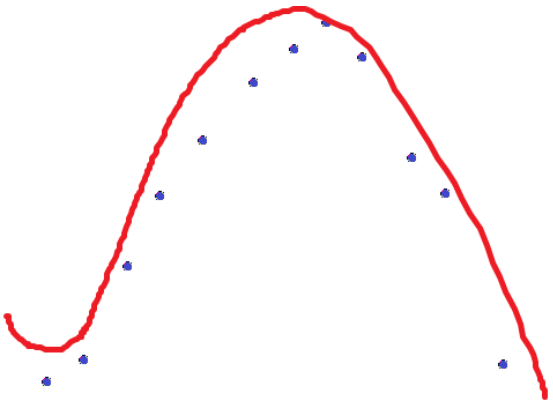


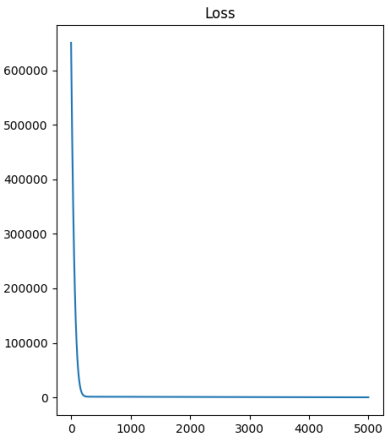


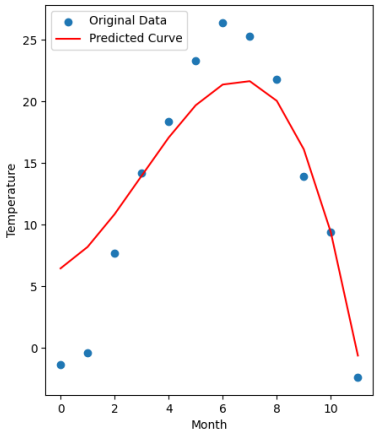


왼쪽 그림과 같은 위로 볼록한 이차함수를 예측하였습니다. 결과 또한 8 월 즈음에서 극대 값을 가지는 위로 볼록한 이차 함수입니다.

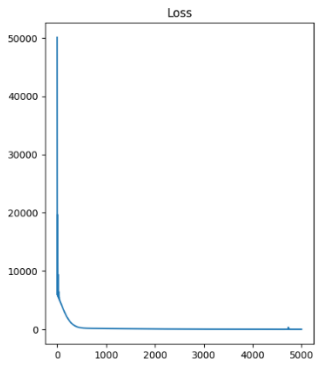
3) 3 차 다항식 모델 예측과 결과

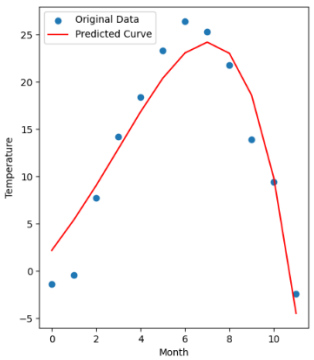






4) 4 차 다항식 모델 결과





1 차, 2 차, 3 차, 4 차 다항식과 같이 차수를 높일수록 모델은 데이터에 대한 더 복잡한 곡선을 학습하게 됩니다. 간단한 1 차 다항식은 직선 형태, 2 차 다항식은 오목, 볼록의 곡선 형태, 3 차 두 번 굴곡이 있는 형태의 곡선, 4 차 이상은 이 앞의 형태보다 더 복잡한 곡선을 모델링 하며, 데이터와 더 유사한 형태로 맞추어지는 것을 볼 수 있습니다. 다만 너무 많은 차수의 증가는 훈련 데이터의 잡음까지 학습할 수 있으므로 조심하자

[참고문헌: 김동근, "텐서플로 딥러닝 프로그래밍"]

[코드]

```
import tensorflow as tf
import pandas as pd
import os
import numpy as np
import matplotlib.pyplot as plt

MSE = tf.keras.losses.MeanSquaredError()

def mse_loss():
    y = tf.zeros_like(x, dtype=tf.float64)
    for i in range(W.shape[0]):
        y += W[i] * (x ** (i + 1))
    y += b  # bias
    return MSE(y, t)
EPOCH = 5000

os.chdir('D:/')
data = pd.read_excel('sejong_Data.xlsx')
x = np.arange(len(data))
t = data["temperature"].values

n = 3  # n 차 다항식 회귀
W = tf.Variable(tf.random.normal([n], dtype=tf.float64))
b = tf.Variable(tf.random.normal([], dtype=tf.float64))

opt = tf.keras.optimizers.Adam(learning_rate=0.01)

loss_list = []
for epoch in range(EPOCH):
    opt.minimize(mse_loss, var_list=[W, b])
    loss = mse_loss().numpy()
    loss_list.append(loss)
    if not epoch % 100:
        print("epoch={}: loss={}".format(epoch, loss))
print("W={}, b={}, loss={}".format(W.numpy(), b.numpy(), loss))

plt.figure(figsize=(11, 6))
plt.subplot(1, 2, 1)
plt.plot(loss_list)
plt.title('Loss')
plt.subplot(1, 2, 2)
plt.scatter(x, t, label='Original Data')
plt.xlabel('Month')
plt.ylabel('Temperature')
t_pred = tf.zeros_like(x, dtype=tf.float64)
for i in range(W.shape[0]):
    t_pred += W[i] * (x ** (i + 1))
t_pred += b  # bias
plt.plot(x, t_pred, 'red', label='Predicted Curve')
plt.legend()
plt.show()
```