

REPORT

Kongju National University



2022년 천안 평균 온도 n차 다항식 회귀 예측

학과명 : 컴퓨터공학과

교과명 : 인공지능

분 반 : 01분반

학 번 : 201901822

이 름 : 이한결

제출일 : 2023-10-07



공주대학교
KONGJU NATIONAL UNIVERSITY

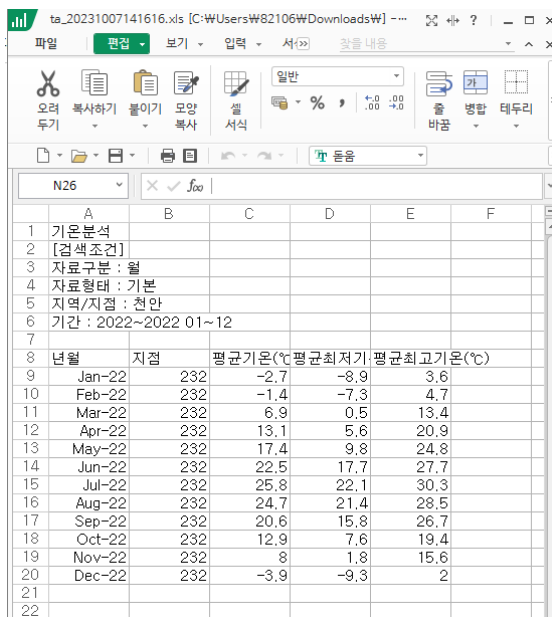
1. 문제 설명

천안의 최근 1년치 온도 데이터(월별, 2022년)를 데이터를 구하여, n차 다항식 회귀 문제를 텐서플로로 이용하여 구현하고, 예측하기

2. 본문

- 데이터 수집 방법

천안의 2022년 평균 온도 데이터를 수집하기 위해 기상청 기상청 기상 자료 개방 포털에서 통계 분석 중 기온 분석(<https://data.kma.go.kr/stcs/grnd/grndTaList.do?pgmNo=70>)에서 데이터를 수집하였다.



년월	지점	평균기온(℃)	평균최저기	평균최고기온(℃)
Jan-22	232	-2.7	-8.9	3.6
Feb-22	232	-1.4	-7.3	4.7
Mar-22	232	6.9	0.5	13.4
Apr-22	232	13.1	5.6	20.9
May-22	232	17.4	9.8	24.8
Jun-22	232	22.5	17.7	27.7
Jul-22	232	25.8	22.1	30.3
Aug-22	232	24.7	21.4	28.5
Sep-22	232	20.6	15.8	26.7
Oct-22	232	12.9	7.6	19.4
Nov-22	232	8	1.8	15.6
Dec-22	232	-3.9	-9.3	2

> 천안 2022년 온도 excel 파일 캡처본

-구현 설명

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

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

1. 필요한 모듈을 import 한다 (tensorflow, numpy, matplotlib)

2. 평균 제곱 오차 함수를 정의한다

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

3. n차식 mse_loss()함수를 정의한다.

```
temp=np.array([
    [1,-2.7],
    [2,-1.4],
    [3,6.9],
    [4,13.1],
    [5,17.4],
    [6,22.5],
    [7,25.8],
    [8,24.7],
    [9,20.6],
    [10,12.9],
    [11,8],
    [12,-3.9]],dtype=np.float32)
```

4. temp(온도) 배열에 1월부터 12월까지 평균 온도를 입력한다

```
x=temp[:, :-1]
t=temp[:, -1:]
```

5. 배열 슬라이스를 이용하여 회귀를 위한 x배열에 월, t배열에 평균 온도를 슬라이스한다.

```
W=tf.Variable(tf.random.normal([n]))
b=tf.Variable(tf.random.normal([1]))
```

6. n차 다항식의 가중치 $W = \text{tf.Variable}(\text{tf.random.normal}([n]))$ 와 바이어스 $b=\text{tf.Variable}(\text{tf.random.normal}([1]))$ 의 훈련 변수를 생성한다

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

7. 객체 opt를 생성하고, 학습률을 0.01로 설정한다. 방법은 Adam를 사용했다.

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

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

8. opt . minimize (mse_loss, var_list = [W,b])로 손실함수 mse_loss에서 변수 리스트 [W, b]에 대해 최적화 한다. 그리고 반복이 100번될 때마다 반복 횟수와 loss를 출력한다.

```
print("W={}, b={}, loss={}".format(W.numpy(),b.numpy(), loss))
```

9. 반복문이 끝나고 학습결과 (W, b 값과 loss값) 출력

```
plt.scatter(x,t)
```

10. x와 t를 점 찍어준다.

```
t_pred=tf.zeros_like(x)
for i in range(W.shape[0]):
    t_pred += W[i]*(x**(i+1))
t_pred+=b
```

11. 다항식 회귀 결과 t_pred를 이용하여 x에서의 예측 값 그래프 생성한다

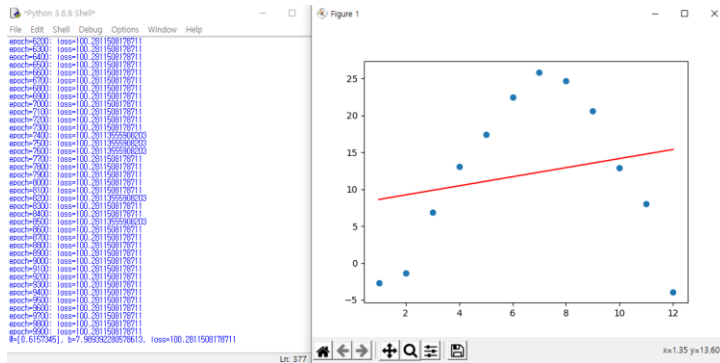
```
plt.plot(x, t_pred, 'red')
plt.show()
```

12. 만든 예측 값 그래프 출력

3. 실험

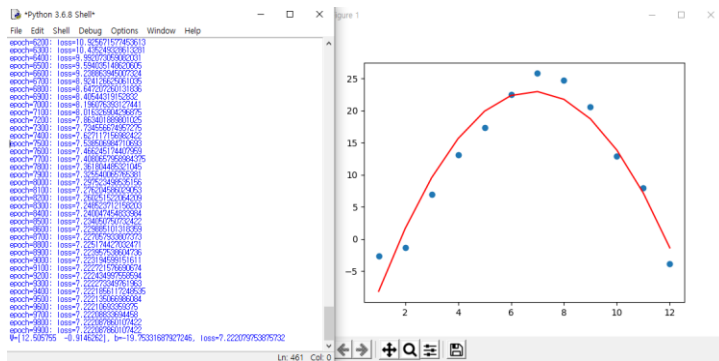
EPOCH=10000, 1차 다항식 모델

>> W=[0.6157345], b=7.989392280578613, loss=100.2811508178711



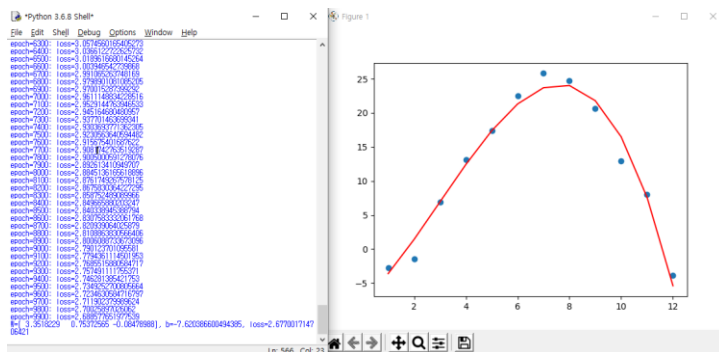
EPOCH=10000, 2차 다항식 모델

>> W=[12.505755 -0.9146262], b=-19.75331687927246, loss=7.222079753875732



EPOCH=10000, 3차 다항식 모델

>> W=[3.3518229 0.75372565 -0.08478988], b=-7.620386600494385, loss=2.677001714706421



4. 참고문헌

김동근, 「텐서플로 딥러닝 프로그래밍」, 가메출판사, 2021, p.73

5. 소스코드

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

MES=tf.keras.losses.MeanSquaredError()
def mse_loss():
    y=tf.zeros_like(x)
    for i in range(W.shape[0]):
        y+=W[i]*(x**(i+1))
    y+=b
    return MES(y,t)

EPOCH=10000
temp=np.array([
    [1,-2.7],[2,-1.4],[3,6.9],[4,13.1],[5,17.4],[6,22.5],
    [7,25.8],[8,24.7],[9,20.6],[10,12.9],[11,8],[12,-3.9]],dtype=np.float32)
x=temp[:,:-1]
t=temp[:, -1:]
n=3
W=tf.Variable(tf.random.normal([n]))
b=tf.Variable(tf.random.normal([]))

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

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

plt.scatter(x,t)
t_pred=tf.zeros_like(x)
for i in range(W.shape[0]):
    t_pred += W[i]*(x**(i+1))
t_pred+=b
plt.plot(x, t_pred, 'red')
plt.show()
```