4주차 실습과제 코드

2020142001 곽종근

실습과제 1.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

fold_dir = "C:\\Users\\user\\OneDrive - 한국공학대학교\\바탕 화면\\3학년
1학기\\머신러닝실습\\Machine-Learning\\5주차\\lin_regression_data_01.csv"
temp_data = pd.read_csv(fold_dir, header=None)
temp_data = temp_data.to_numpy()  #위에서 읽은 temp_data는 dataframe형태이므로 이를 numpy배열로
변환하여 slice이용한다.

Wei = np.zeros([50])  # x축 값인 Wei(무게)를 zeros 50으로 선언해준다.
Len = np.zeros([50])  # y축 값인 Len(길이)를 zeros 50으로 선언해준다.


for j in range(50):  # 파일을 열어보면 총 개수가 50개이므로 50번 반복하도록 반복문 실행
    Wei[j] = temp_data[j, 0]  # 위에서 선언한 데이터는 50by2가 되는데 여기서 0번째 열의 데이터만 Wei에 저장
    Len[j] = temp_data[j, 1]  # 1번째 열의 데이터만 Len에 저장해준다.

Wei_Len = sum(Wei * Len)   #Wo과 Wi를 구하기 위해 Wei_Len이라는 것을 따로 선언하여 이를 위에 선언한
Wei와 Len배열의 곱한 값의 합으로 선언
Wei_squared = sum(Wei**2)   #Wo과 Wi를 구하기 위해 Wei_squared를 따로 선언해 이를 Wei의 제곱의 합으로
선언

plt.scatter(Wei, Len)  # 점으로 Wei와 Len을 선언한다.
plt.legend(['RealData']) #위에서 선언한 점의 이름을 화면에 띄운다.


Wei_Sum=sum(Wei)  #Wei의 총 합을 Wei_Sum으로 선언
Len_Sum=sum(Len)    #Len의 총 합을 Len_Sum으로 선언


Wo=((Wei_Len/50)-((Wei_Sum/50)*(Len_Sum/50)))/((Wei_squared/50)-(Wei_Sum/50)**2)  #위에서 선언해준
것들을 이용해 Wo의 식 정리
Wi=(Len_Sum-Wo*Wei_Sum)/50  #Wi의 식을 정리한다.



# 그래프 그리기
plt.scatter(Wei, Len, label='Real Data')  # 실제 데이터 산점도
plt.xlabel('Weight[g]')  # x축을 무게 Weight[g]으로 설정
plt.ylabel('Length[cm]')  # y축을 길이 Length[cm]로 설정
```

```
plt.title('Project 1')  # 그래프 제목 설정
plt.grid(True)  # 격자를 표시해준다.


x_values = np.arange(Wei.min() - 1, Wei.max() + 1, (Wei.max() + 1 - (Wei.min() - 1)) / 1000)  # x 값 범위 설정
#x의 범위 설정에서 일단 min을 이용해 Wei의 최소값에서 -1을 한 값을 시작으로 하고, max를 이용해 Wei의 최대값에 +1을 이용해 마지막 값을 선언한다
#그 후 간격을 설정하는데 그 간격을 임의로 1000개로 선언하기 위해 처음값과 끝값을 더해 1000으로 나누면 그 간격이 된다.
y_values = Wo * x_values + Wi  # y^(예측값)의 식을 써주면 y의 범위가 정해진다.
plt.plot(x_values, y_values, color='red', label='Regression Line')  # 회귀선 그리기

fore_value = Wo * Wei + Wi  #위에서 예측값을 위한 Wo과 Wi를 구했으므로 y=Wo x+ Wi 식을 이용해 식을 작성해준다.
mse = (sum((fore_value-Len)**2))/50  #mse의 식을 위에서 정해준 값들로 식을 작성해준다.
print("MSE:", mse) #mse의 값을 표현

plt.legend()  # 범례 표시
plt.show  # 그래프 출력
```

실습과제 2.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 데이터 불러오기
fold_dir = "C:\\Users\\user\\OneDrive - 한국공학대학교\\바탕 화면\\3학년 1학기\\머신러닝실습\\Machine-Learning\\5주차\\lin_regression_data_01.csv"
temp_data = pd.read_csv(fold_dir, header=None)
temp_data = temp_data.to_numpy()     #data를 numpy로 불러와서 슬라이싱등 할 수 있도록 선언한다.

# 데이터 분리
Wei = temp_data[:, 0]  # 무게 데이터를 Wei저장
Len = temp_data[:, 1]  # 길이 데이터를 Len에 자ㅣ장

# 초기화와 함께 값을 설정해주고, 값의 변화를 주고자 하면 여기서 변화를 주면 된다
w0 = 2
w1 = 4
Learn_R = 0.001
rp = 10000

# 경사하강법 사용자 지정 함수를 만들어준다.
def GDM(x, y, Learn_R, rp):    # x는 w0, y는 w1, Learn_R은 알파, rp는 반복횟수
```

```
    Wei_new = []    #미분하여 변화하는 Wei값들은 저장하는 배열을 만들어준다.
    Len_new = []    #미분하여 변화하는 Len값들은 저장하는 배열을 만들어준다.
    MSE_new = []    #변화하는 MSE값들은 저장하는 배열을 만들어준다.

    for i in range(rp):          #설정한 반복횟수 rp만큼
        new_y = x * Wei + y      #new_y는 예측값y^을 의미한다.
        error = new_y - Len      #아래의 미분식과 mse를 편하게 계산하기 위해 error로 예측값 - 실제값을 선언
        dif_W0 = np.mean(error * Wei) #W0미분식
        dif_W1 = np.mean(error)      #W1미분식

        x = x - (Learn_R * dif_W0) # x는 계속해서 새로 변화하는 w0값을 저장해주고 최종적으로 마지막 x에는
최종w0값이 저장됨
        y = y - (Learn_R * dif_W1)   # y는 계속해서 새로 변화하는 w1값을 저장해주고 최종적으로 마지막
x에는 최종w0값이 저장됨
        mse = np.mean(error ** 2)  # 평균 제곱 오차 계산식

        Wei_new.append(x)   #위에서 구해지는 변화하는 w0,w1,mse값들을 위에서 만들어준 빈 배열에 계속해서
넣어준다.
        Len_new.append(y)

        MSE_new.append(mse)

    return x, y, Wei_new, Len_new, MSE_new

# 경사 하강법 수행하는데 x,y는 여기서 결과적으로 최종w0,w1값을 의미하게 된다.
x, y, Wei_new, Len_new, MSE_new = GDM(w0, w1, Learn_R, rp)

# 그래프 그리기

# w0, w1 변화 그래프
plt.subplot(1, 3, 1)   #1행 3열 1번에 선언한다는 의미이다
plt.plot(Wei_new, label='W0 change')  # w0의 변화 그래프
plt.plot(Len_new, label='W1 change')  # w1의 변화 그래프
plt.xlabel('Epoch') #x축 이름을 Epoch로 설정
plt.ylabel('W0,W1 Value') #y축 이름을 w0,w1 Value로 설정
plt.title('Optimal Solution') #이 그래프의 제목을 Optimal Solution으로 설정
plt.grid()
plt.legend()

# MSE 변화 그래프
plt.subplot(1, 3, 2)   #1행 3열 2번에 선언
plt.plot(MSE_new, label='MSE')
plt.xlabel('Epoch') #x축 이름을 Epoch로 설정
plt.ylabel('MSE') #y축 이름을 mse로 설정
plt.title('Mean Squared Error') #이 그래프의 제목을 Mean Squared Error으로 설정
plt.grid()
```

```
plt.legend()

# 예측 결과 그래프
plt.subplot(1, 3, 3)
plt.scatter(Wei, Len, label='Real Data')
x_values = np.arange(Wei.min() - 1, Wei.max() + 1, (Wei.max() + 1 - (Wei.min() - 1)) / 1000)
#x의 범위 설정에서 일단 min을 이용해 Wei의 최소값에서 -1을 한 값을 시작으로 하고, max를 이용해 Wei의 최대값에 +1을 이용해 마지막 값을 선언한다
#그 후 간격을 설정하는데 그 간격을 임의로 1000개로 선언하기 위해 처음값과 끝값을 더해 1000으로 나누면 그 간격이 된다.
yy = x * x_values + y  # 회귀선
# y^(예측값)의 식을 써주면 y의 범위가 정해진다.
plt.plot(x_values, yy, c='r', label='Regression Line')  # 회귀선 그래프
plt.xlabel('Weight[g]')
plt.ylabel('Length[cm]')
plt.title('DATA SET')
plt.grid()
plt.legend(loc='upper left')


plt.show()
```