6주차 과제 보고서

과목명: 인공지능 기초

담당 교수: 김용우

소속 학과: 시스템반도체공학과

학번: 201921390

이름: 윤승재

**1. 코드 구현(linear regression)**

**(1) data code**

# 신생아 표준 몸무게 와 키의 관계 (0개월부터 12월까지)  
import numpy as np  
from matplotlib import pyplot as plt  
import csv  
  
f = open('money.csv','r')  
file = csv.reader(f) # csv파일 읽어오기  
value = list(file)  
data = np.array(value,dtype='f') # csv파일 데이터 타입을 float형으로 바꾸었다.  
x = data[:, 0] # x값에 몸무게를 대입  
y = data[:, 1] # y값에 몸무게를 대입  
  
plt.scatter(data[:, 0], data[:, 1])  
plt.title("cm / kg")  
plt.xlabel("baby weight (kg)")  
plt.ylabel("baby height (cm)")  
plt.axis([0,12,0,100])  
plt.show()

**(2) plot code**

import numpy as np  
from matplotlib import pyplot as plt  
import csv  
# 시속 단속  
f = open('input.csv','r')  
file = csv.reader(f) # csv파일 읽어오기  
value = list(file)  
data = np.array(value[1:])  
  
# plt.plot(data[:, 0], data[:, 1])  
plt.scatter(data[:, 0], data[:, 1]) # 점으로 데이터 표현  
plt.title("fine Pay/Pass vs km/h")  
plt.xlabel("km/h")  
plt.ylabel("Pay/Pass")  
plt.grid()  
# plt.axis([0, 420, 0, 50])  
plt.show()

**(3) cost code**

import numpy as np  
import csv  
from matplotlib import pyplot as plt  
  
f = open('money.csv','r')  
file = csv.reader(f) # csv파일 읽어오기  
value = list(file)  
data = np.array(value,dtype='f') # csv파일 데이터 타입을 float형으로 바꾸었다.  
x = data[:, 0] # x값에 몸무게를 대입  
y = data[:, 1] # y값에 몸무게를 대입  
  
x = data[:, 0]  
y = data[:, 1]  
  
w = np.random.uniform(low=1, high=10)  
b = np.random.uniform(low=0, high=10)  
print('w: ', w, 'b: ', b)  
  
y\_hat = w \* x + b  
  
error = (y\_hat - y) \*\* 2  
print('error: ', error)  
print('shape: ', error.shape)  
  
cost = error.mean()  
print('cost: ', cost)

**(4)gradient code**

# 신생아 표준 몸무게 와 키의 관계 (0개월부터 12월까지)  
import numpy as np  
from matplotlib import pyplot as plt  
import csv  
  
f = open('money.csv','r')  
file = csv.reader(f) # csv파일 읽어오기  
value = list(file)  
data = np.array(value,dtype='f') # csv파일 데이터 타입을 float형으로 바꾸었다.  
  
x = data[:, 0]/100  
y = data[:, 1]/100  
  
num\_epoch = 3000000 # w,b를 최대한 정확히 찾기 위해 10만번으로 설정  
  
# 학습율 (learning\_rate)  
learning\_rate = 0.01  
  
costs = []  
  
w = np.random.uniform(low=1, high=10)  
b = np.random.uniform(low=0, high=10)  
print('w: ', w, 'b: ', b)  
  
for epoch in range(num\_epoch):  
 y\_hat = w \* x + b  
  
 error = ((y\_hat - y)\*\*2)  
 cost = error.mean()  
  
 if cost < 0.00002: # 10만번을 돌리면서 error의 값을 최소화 하기 위해  
 break  
  
 w = w - learning\_rate \* ((y\_hat - y) \* x).mean()  
 b = b - learning\_rate \* (y\_hat - y).mean()  
  
 costs.append(cost)  
  
 if epoch % 10 == 0:  
 print("{0:2} w = {1:.5f}, b = {2:.5f} error = {3:.5f}".format(  
 epoch, w, b, cost))  
  
print("----" \* 15)  
print("{0:2} w = {1:.5f}, b = {2:.5f} error = {3:.5f}".format(epoch, w, b, cost))  
  
  
plt.figure(figsize=(10, 7))  
plt.plot(costs)  
plt.xlabel('Epochs')  
plt.ylabel('Costs')  
plt.show()

**(5) prediction code**

# 신생아 표준 몸무게 와 키의 관계 (0개월부터 12월까지)  
import numpy as np  
from matplotlib import pyplot as plt  
import csv  
  
f = open('money.csv','r')  
file = csv.reader(f) # csv파일 읽어오기  
value = list(file)  
print(value)  
data = np.array(value,dtype='f') # csv파일 데이터 타입을 float형으로 바꾸었다.  
  
x = data[:, 0] # x값에 몸무게를 대입  
y = data[:, 1] # y값에 몸무게를 대입  
print(x)  
print(y)  
  
w = np.random.uniform(low=1, high=10)  
b = np.random.uniform(low=0, high=10)  
print('w: ', w, 'b: ', b)  
  
y\_hat = w \* x + b  
  
error = (y\_hat - y) \*\* 2  
print('error: ', error)  
print('shape: ', error.shape)  
  
cost = error.mean()  
print('cost: ', cost)  
  
w = 4.79174  
b = 0.30931  
org\_x = np.linspace(0, 420, 100)  
pred\_x = org\_x / 100 # 정규화  
pred\_y = w \* pred\_x + b  
pred\_y = pred\_y \* 100 # 정규화 반대 과정  
  
  
x = 8.9 # True : 74  
y\_predict = x / 100 \* w + b  
print(y\_predict \* 100)  
  
x = 7.9 # True : 67.6  
y\_predict = x / 100 \* w + b  
print(y\_predict \* 100)  
  
x = 8.5 # True : 71.5  
y\_predict = x / 100 \* w + b  
print(y\_predict \* 100)  
  
x = 11 # True: 82.4  
y\_predict = x / 100 \* w + b  
print(y\_predict \* 100)

**(6)linear\_regression code**

# 신생아 표준 몸무게 와 키의 관계 (0개월부터 12월까지)  
import numpy as np  
from matplotlib import pyplot as plt  
import csv  
  
f = open('money.csv','r')  
file = csv.reader(f) # csv파일 읽어오기  
value = list(file)  
print(value)  
data = np.array(value,dtype='f') # csv파일 데이터 타입을 float형으로 바꾸었다.  
  
x = data[:, 0] # x값에 몸무게를 대입  
y = data[:, 1] # y값에 몸무게를 대입  
print(x)  
print(y)  
  
w = np.random.uniform(low=1, high=10)  
b = np.random.uniform(low=0, high=10)  
print('w: ', w, 'b: ', b)  
  
y\_hat = w \* x + b  
  
error = (y\_hat - y) \*\* 2  
print('error: ', error)  
print('shape: ', error.shape)  
  
cost = error.mean()  
print('cost: ', cost)  
  
w = 4.13194  
b = 0.35780  
org\_x = np.linspace(0, 420, 100)  
pred\_x = org\_x / 100 # 정규화  
pred\_y = w \* pred\_x + b  
pred\_y = pred\_y \* 100 # 정규화 반대 과정  
  
plt.scatter(x, y)  
plt.title("baby's height and weight")  
plt.xlabel("baby weight (kg)")  
plt.ylabel("baby height (cm)")  
plt.plot(org\_x, pred\_y)  
plt.axis([0, 12, 0, 100])  
plt.show()

**-csv 파일 데이터값(신생아 몸무게와 키)**

3.2,49.1  
3.3,49.9  
4.2,53.7  
4.5,54.7  
5.1,57.1  
5.6,58.4  
5.8,59.8  
6.4,61.4  
7.0,63.9  
7.3,65.7  
7.5,65.9  
7.9,67.6  
8.2,70.1  
8.3,69.2  
8.5,71.5  
8.6,70.6  
8.7,72.8  
8.9,74  
9.2,73.3  
9.4,74.5  
9.6,75.7  
9.9,76.9  
10.1,78

**2. 고찰**

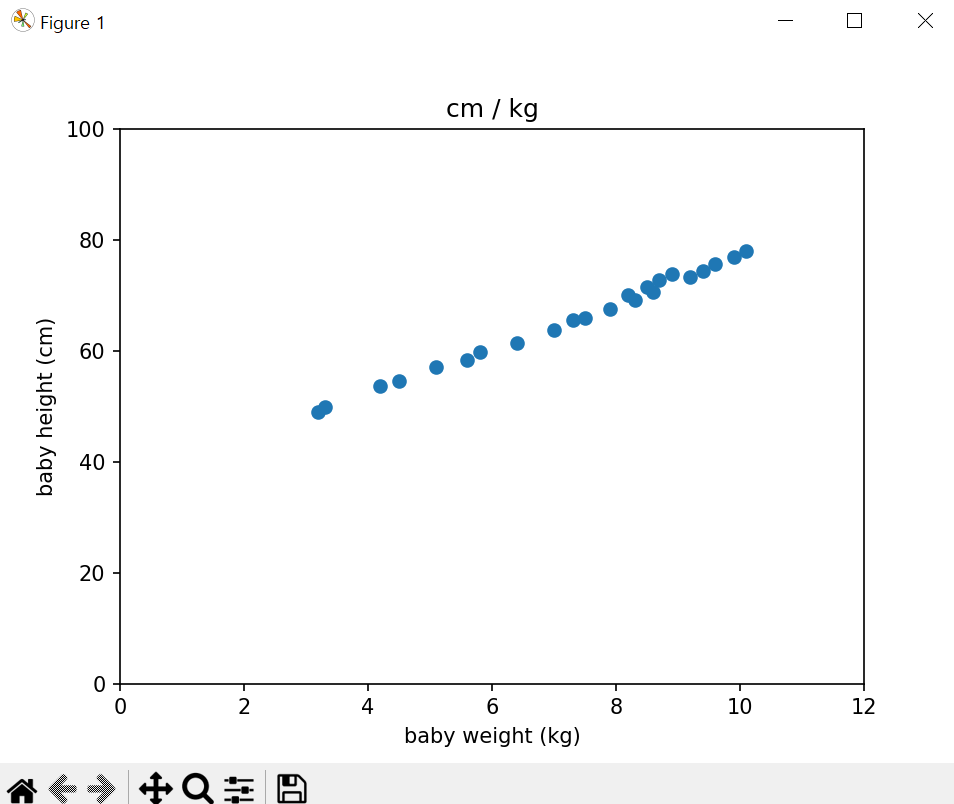
먼저, 강의시간에 실습한 것을 분석하고 나서 이 코드를 짜게 되었다. 먼저, 실습코드와 같이 돌려보았는데 찾은 csv파일의 데이터인 신생아의 몸무게와 키의 값들이 작아서 최적화된 w, b를 찾는데 어려움을 겪었다. 그래서 num\_epoch를 10만번으로 바꾸고 learning rate를 0.01, cost를 0.0002로 바꿔서 실행하였다. 바로 이렇게 한 것이 아니라 처음에는 num\_epoch를 2만번으로 하고 5만번으로 하면서 여러 번의 시도를 하였다. 데이터 값들이 작아 기울기인 w값에 따라 값들의 오차가 심하게 심하게 나는 거 같아서 많은 시도를 한 끝에 제일 오차가 적은 w, b를 구하게

되었다.

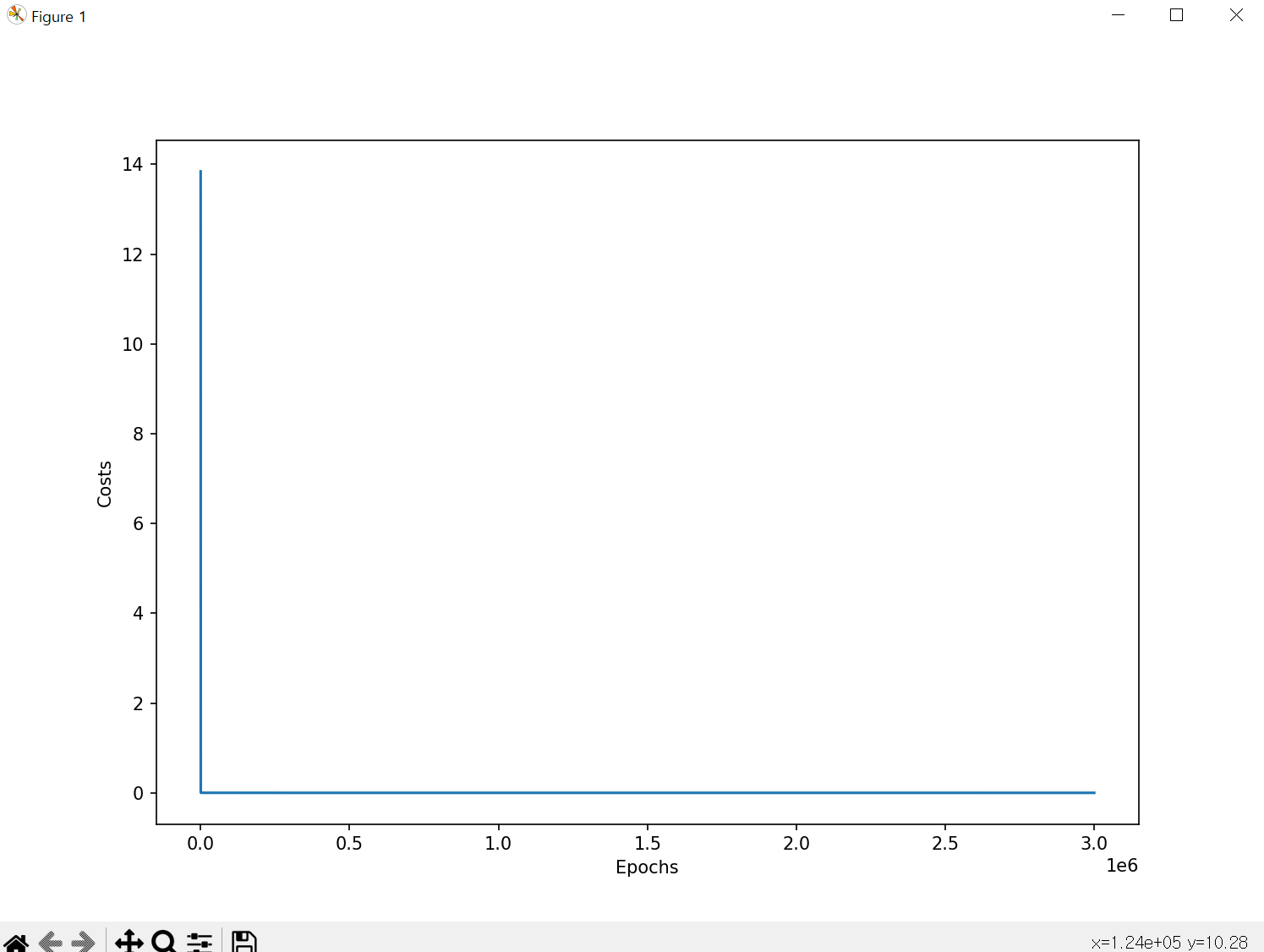
또한, csv파일을 찾기 어려워서 찾은 데이터를 엑셀파일에 직접 옮긴뒤 csv로 저장하여 실습하였다. 밑의 구현 결과를 보면, 전체적인 값들이 구한 linear\_regression의 그래프와 거의 일치하였다. 아쉬운 점은 데이터 값들끼리의 차이가 매우 근소하여, 코드를 구현하는데 많은 어려움을 겪어서 다음에는 데이터 값들의 차이가 좀 나는 데이터를 찾아서 하면 좀 더 좋은 선명한 결과를 얻을 수 있을 것 같다.

**-코드구현 결과**

**- data.py**

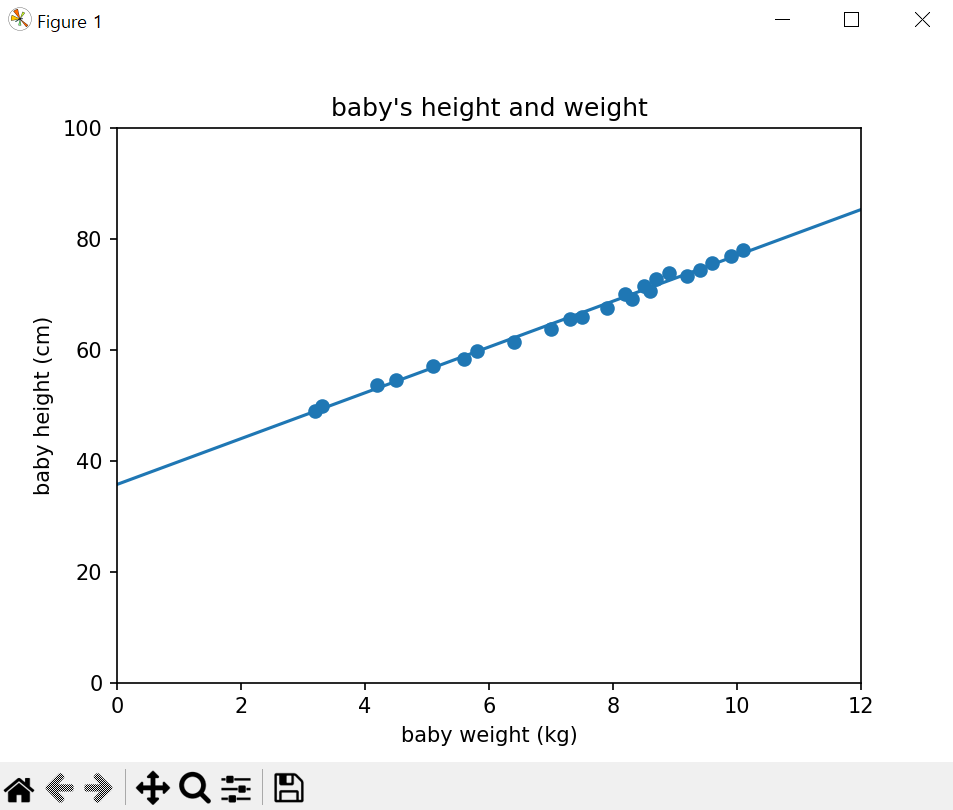
****

**- cost와 num\_epoch 그래프**

****

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**-linear regression**

**1. 코드 구현(logistic regression)**

**(1) data**

import numpy as np  
import csv  
from matplotlib import pyplot as plt  
  
# 시속 단속  
f = open('input.csv','r')  
file = csv.reader(f) # csv파일 읽어오기  
value = list(file)  
data = np.array(value[1:])  
print(data)  
  
# plt.plot(data[:, 0], data[:, 1])  
plt.scatter(data[:, 0], data[:, 1]) # 점으로 데이터 표현  
plt.title("fine Pay/Pass vs km/h")  
plt.xlabel("km/h")  
plt.ylabel("Pay/Pass")  
plt.grid()  
# plt.axis([0, 420, 0, 50])  
plt.show()

**(2)plot**

import numpy as np  
from matplotlib import pyplot as plt  
import csv  
# 시속 단속  
f = open('input.csv','r')  
file = csv.reader(f) # csv파일 읽어오기  
value = list(file)  
data = np.array(value[1:])  
  
# plt.plot(data[:, 0], data[:, 1])  
plt.scatter(data[:, 0], data[:, 1]) # 점으로 데이터 표현  
plt.title("fine Pay/Pass vs km/h")  
plt.xlabel("km/h")  
plt.ylabel("Pay/Pass")  
plt.grid()  
# plt.axis([0, 420, 0, 50])  
plt.show()

**(3)hypothesis code**

import numpy as np  
from matplotlib import pyplot as plt  
import csv  
# 시속 단속  
f = open('input.csv','r')  
file = csv.reader(f) # csv파일 읽어오기  
value = list(file)  
data = np.array(value[1:],dtype='f')  
  
  
x = data[:, 0]  
y = data[:, 1]  
  
import math  
  
# array (numpy)  
def sigmoid(x): # 시그모이드 함수 정의  
 return 1/(1+np.exp(-x))  
  
# x is scalar 1 EA  
# def sigmoid(x): # 시그모이드 함수 정의  
# return 1/(1+math.exp(-x))  
x = np.arange(-10.0, 10.0, 0.1)  
print(x.shape, x.ndim, x.size)  
print(type(x))  
# print(x)  
y = sigmoid(x)  
# plt.scatter(data[:, 0], data[:, 1])  
plt.plot(x, y, 'g')  
plt.plot([0, 0], [1.0, 0.0], ':') # 가운데 점선 추가  
plt.title('Sigmoid Function')  
plt.show()  
  
  
# W 값에 따른 경사도 변화  
x = np.arange(-5.0, 5.0, 0.1)  
y1 = sigmoid(0.5\*x)  
y2 = sigmoid(x)  
y3 = sigmoid(2\*x)  
  
plt.plot(x, y1, 'r', linestyle='--') # W의 값이 0.5일때  
plt.plot(x, y2, 'g') # W의 값이 1일때  
plt.plot(x, y3, 'b', linestyle='--') # W의 값이 2일때  
plt.plot([0, 0], [1.0, 0.0], ':') # 가운데 점선 추가  
plt.title('Sigmoid Function')  
plt.show()  
  
# b 값에 따른 좌, 우 이동  
x = np.arange(-5.0, 5.0, 0.1)  
y1 = sigmoid(x+0.5)  
y2 = sigmoid(x+1)  
y3 = sigmoid(x+1.5)  
  
plt.plot(x, y1, 'r', linestyle='--') # x + 0.5  
plt.plot(x, y2, 'g') # x + 1  
plt.plot(x, y3, 'b', linestyle='--') # x + 1.5  
plt.plot([0, 0], [1.0, 0.0], ':') # 가운데 점선 추가  
plt.title('Sigmoid Function')  
plt.show()

**(4)cost\_code**

import numpy as np  
from matplotlib import pyplot as plt  
import csv  
# 시속 단속  
f = open('input.csv','r')  
file = csv.reader(f) # csv파일 읽어오기  
value = list(file)  
data = np.array(value[1:],dtype='f')  
  
x = data[:, 0] / 80 # /100은 정규화 미분할때 미분값이 크면 잘 학습이 안되서 일부러 줄임  
# x = data[:, 0]  
y = data[:, 1]  
  
  
def sigmoid(x): # 시그모이드 함수 정의  
 return 1/(1+np.exp(-x))  
  
w = np.random.uniform(low=0, high=20)  
b = np.random.uniform(low=-20, high=10)  
print('w: ', w, 'b: ', b)  
  
num\_epoch = 500000 # 50만 돌리기  
learning\_rate = 10 #괴장히 빠르게 학습을 할려면 키워줘야 된다.  
costs = []  
eps = 1e-5  
  
for epoch in range(num\_epoch):  
 hypothesis = sigmoid(w \* x + b)  
  
 cost = y \* np.log(hypothesis + eps) + (1 - y) \* np.log(1 - hypothesis + eps)  
 # eps값을 넣은 것은 log(0)의 값을 방지하기 위해 조금이라도 작은 값을 넣어준 것  
 cost = -1 \* cost  
 cost = cost.mean()  
  
 if cost < 0.00005:  
 break  
  
 w = w - learning\_rate \* ((hypothesis - y) \* x).mean()  
 b = b - learning\_rate \* (hypothesis - y).mean()  
  
 costs.append(cost)  
  
 if epoch % 50 == 0:  
 print("{0:2} w = {1:.5f}, b = {2:.5f} error = {3:.5f}".format(  
 epoch, w, b, cost))  
  
print("----" \* 15)  
print("{0:2} w = {1:.5f}, b = {2:.5f} error = {3:.5f}".format(epoch, w, b, cost))  
  
plt.figure(figsize=(10, 7))  
plt.plot(costs)  
plt.xlabel('Epochs')  
plt.ylabel('Costs')  
plt.show()  
  
# learning rate는 경험치로 하는 것

**(5)prediction**

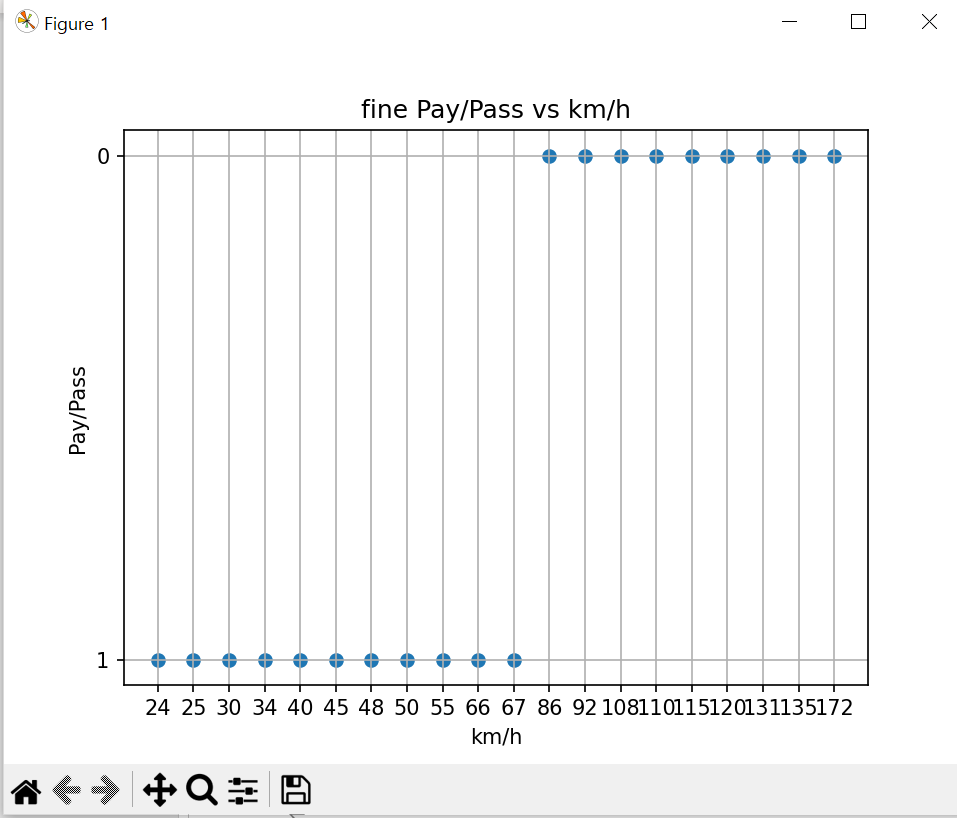
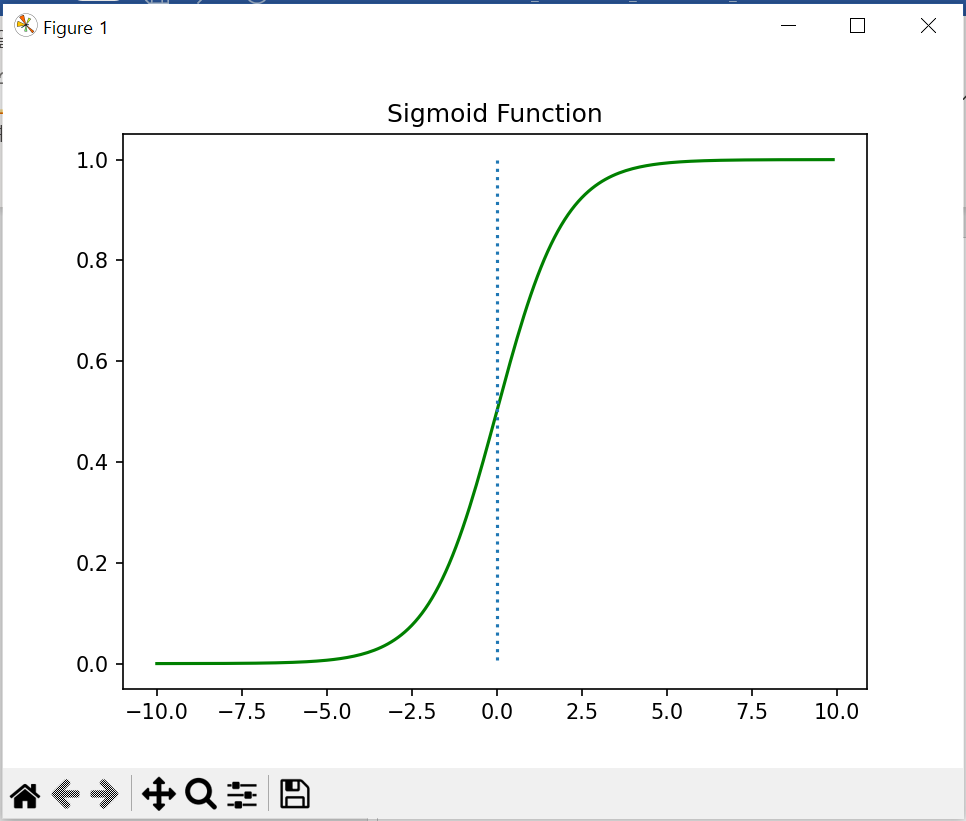
import numpy as np  
from matplotlib import pyplot as plt  
import csv  
# 시속 단속  
f = open('input.csv','r')  
file = csv.reader(f) # csv파일 읽어오기  
value = list(file)  
data = np.array(value[1:],dtype='f')  
  
# x w = -79.26862, b = 60.77626  
# x = data[:, 0] / 100  
x = data[:, 0]  
y = data[:, 1]  
  
def sigmoid(x): # 시그모이드 함수 정의  
 return 1/(1+np.exp(-x))  
  
# 예측  
w = -64.06770  
b = 61.38567  
x = 108 # True : 0  
x = x / 80  
pred\_y = sigmoid(w \* x + b)  
print('x = ', x\*80, 'pred: ', pred\_y)  
  
x = 50 # True : 1  
x = x / 80  
pred\_y = sigmoid(w \* x + b)  
print('x = ', x\*80, 'pred: ', pred\_y)  
  
x = 67 # True : 1  
x = x / 80  
pred\_y = sigmoid(w \* x + b)  
print('x = ', x\*80, 'pred: ', pred\_y)  
  
  
x = 172 # True : 0  
x = x / 80  
pred\_y = sigmoid(w \* x + b)  
print('x = ', x\*80, 'pred: ', pred\_y)  
  
x = 120 # True : 0  
x = x / 80  
pred\_y = sigmoid(w \* x + b)  
print('x = ', x\*80, 'pred: ', pred\_y)  
  
x = 66 # True : 1  
x = x / 80  
pred\_y = sigmoid(w \* x + b)  
print('x = ', x\*80, 'pred: ', pred\_y)  
  
x = 145 # True : 0  
x = x / 80  
pred\_y = sigmoid(w \* x + b)  
print('x = ', x\*80, 'pred: ', pred\_y)

**(6) logistic\_regression**

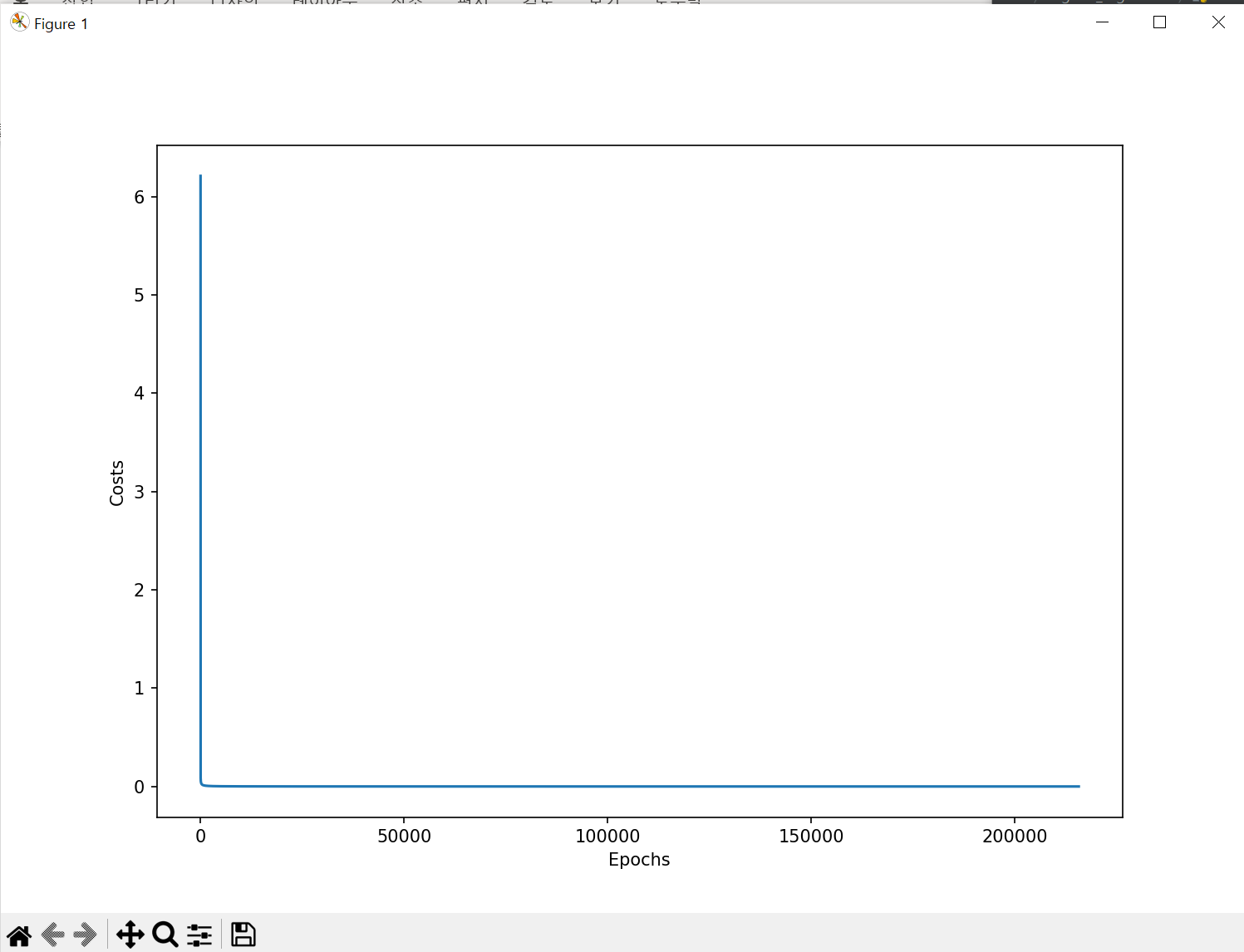
import numpy as np  
from matplotlib import pyplot as plt  
import csv  
# 시속 단속  
f = open('input.csv','r')  
file = csv.reader(f) # csv파일 읽어오기  
value = list(file)  
data = np.array(value[1:],dtype='f')  
# text file input/output  
  
# x = data[:, 0] / 80  
x = data[:, 0]  
y = data[:, 1]  
  
def sigmoid(x): # 시그모이드 함수 정의  
 return 1/(1+np.exp(-x))  
  
w = np.random.uniform(low=0, high=20)  
b = np.random.uniform(low=-20, high=10)  
print('w: ', w, 'b: ', b)  
  
num\_epoch = 100000  
  
learning\_rate = 100  
  
costs = []  
  
eps = 1e-5  
  
for epoch in range(num\_epoch):  
 hypothesis = sigmoid(w \* x + b)  
  
 cost = y \* np.log(hypothesis + eps) + (1 - y) \* np.log(1 - hypothesis + eps)  
 cost = -1 \* cost  
 cost = cost.mean()  
  
 if cost < 0.0005:  
 break  
 w = w - learning\_rate \* ((hypothesis - y) \* x).mean()  
 b = b - learning\_rate \* (hypothesis - y).mean()  
  
 costs.append(cost)  
  
 if epoch % 5000 == 0:  
 print("{0:2} w = {1:.5f}, b = {2:.5f} error = {3:.5f}".format(  
 epoch, w, b, cost))  
  
print("----" \* 15)  
print("{0:2} w = {1:.5f}, b = {2:.5f} error = {3:.5f}".format(epoch, w, b, cost))  
  
  
# 예측  
w = -64.08008  
b = 61.39749  
x = 45 # True : 0  
x = x / 80  
pred\_y = sigmoid(w \* x + b)  
print(pred\_y)  
  
x = 60 # True : 1  
x = x / 80  
pred\_y = sigmoid(w \* x + b)  
print(pred\_y)  
  
x = data[:, 0]  
y = data[:, 1]  
  
org\_x = np.linspace(0, 200, 10)  
pred\_y = sigmoid(w \* (org\_x / 80) + b)  
  
plt.scatter(x, y)   
plt.title("Pay/Fail vs km/h")  
plt.xlabel("km/h")  
plt.ylabel("Pay/Fail")  
plt.plot(org\_x, pred\_y, 'r')  
# plt.axis([0, 420, 0, 50])  
plt.show()

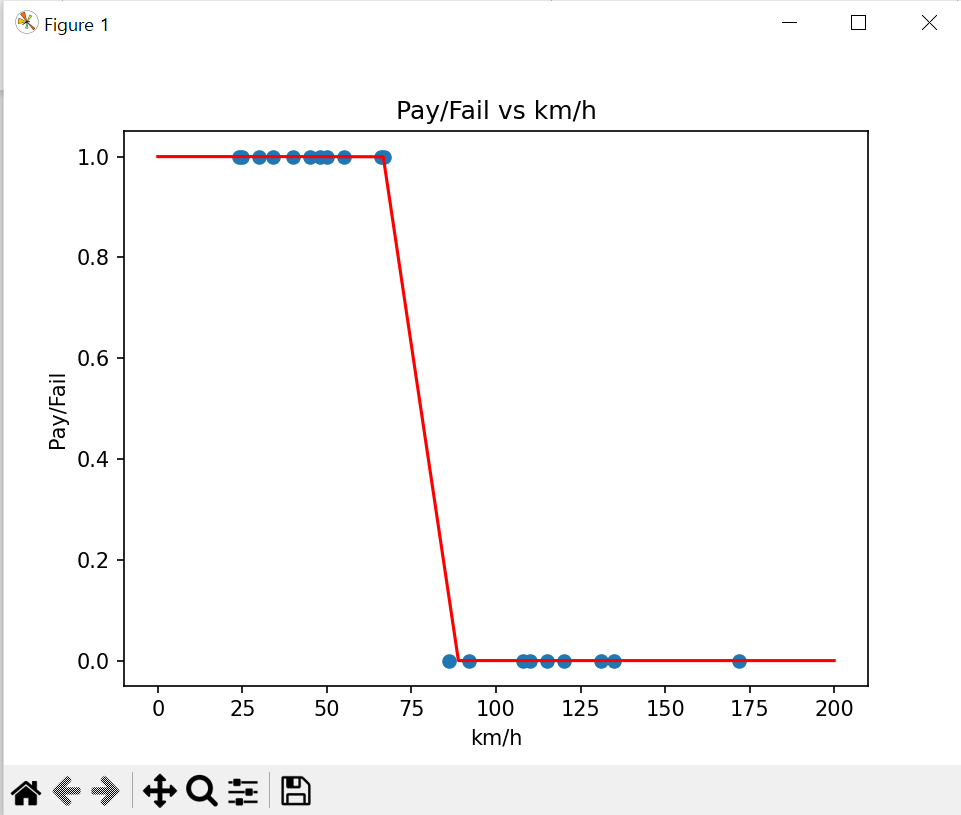
* **csv 파일**

|  |  |
| --- | --- |
| 시속 | 벌금 |
| 24 | 1 |
| 25 | 1 |
| 30 | 1 |
| 34 | 1 |
| 40 | 1 |
| 45 | 1 |
| 48 | 1 |
| 50 | 1 |
| 55 | 1 |
| 66 | 1 |
| 67 | 1 |
| 86 | 0 |
| 92 | 0 |
| 108 | 0 |
| 110 | 0 |
| 115 | 0 |
| 120 | 0 |
| 131 | 0 |
| 135 | 0 |
| 172 | 0 |

**- 코드 실행 결과**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

****

**- 고찰**

**가설을 세우는 부분에서 문제가 발생하였다. muliply라는 에러가 계속 떠서 생각을 해보다가 계산된 값이 0이 되지 않게 하기 위해 eps = 1e-5의 값을 넣어줘서 하니 문제없이 잘 실행하였다.**

**cost함수를 계산하여 나온 그래프가 잘 나오지 않아서 num epoch와 learning rate(학습률) 등을 고쳐서 구하니 error의 값이0.00005가 나왔고, cost함수의 그래프가 0에 최대한 근접하게 잘 나왔다. 마지막 logistic regression은 sigmoid 함수의 그래프 개형과 같이 잘 나왔다.**