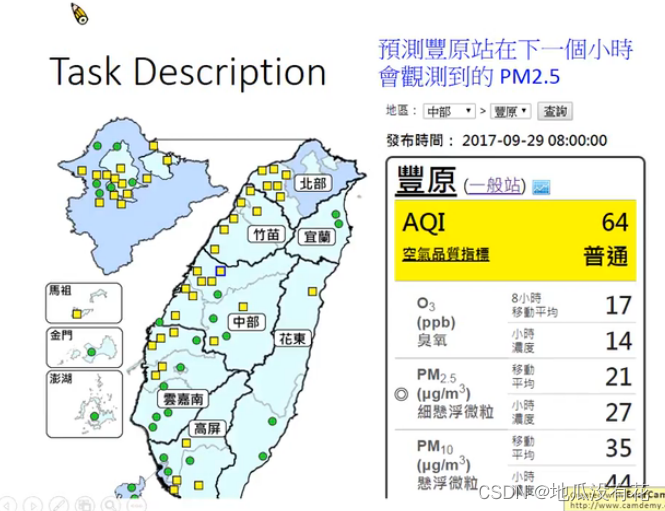
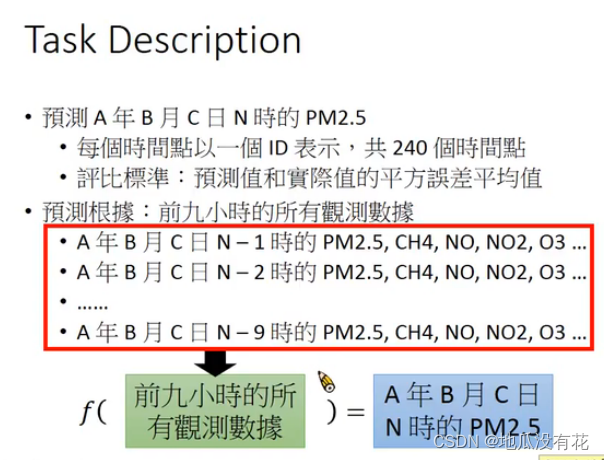
預測PM2.5的值

1. 作業描述

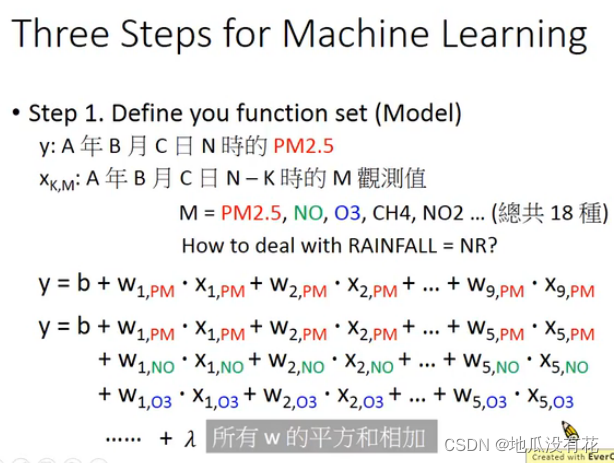
* 視頻：李宏毅《機器學習》- - PM2.5預測https://www.bilibili.com/video/BV1Ht411g7Ef/?p=9
* 作業描述

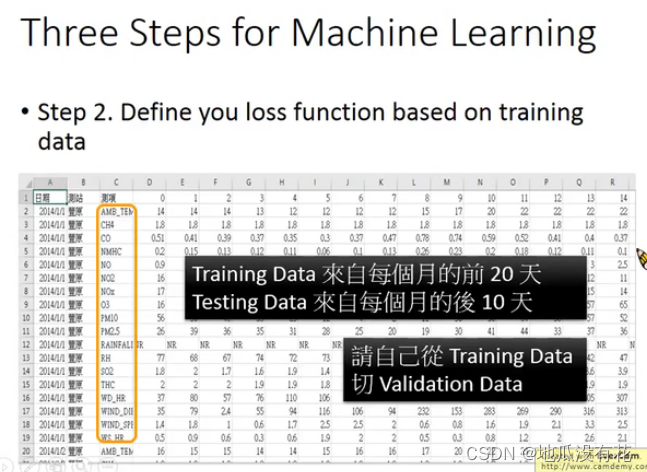
讓機器預測**豐原站**在下一個小時會觀測到的PM2.5，舉例來說，如現在是2017-09-29 08：00：00，那麼要預測2017-09-29 09：00：00豐原站的PM2.5值會是多少。

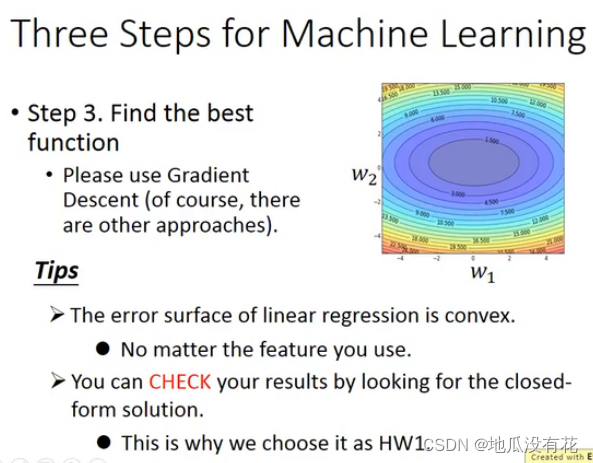




解析：







**任務**

* 任務要求：**預測PM2.5的值**，我們將用**梯度下降法** (**Gradient Descent**) **預測 PM2.5** 的值 (**Regression** 回歸問題)
* 環境要求：
  + 要求 **python3.5+**
  + 搭配應用模組
    - numpy
    - scipy
    - pandas
  + 方法：**梯度下降+線性回歸**
* 數据介紹： 本次作業使用豐原站的觀測記錄，分成 **train set** 跟 **test set**，train set 是豐原站每個月的前20天所有數据，test set則是從豐原站剩下的數据中取樣出來。
  + **train.csv**:每個月前20天每個小時的氣象數据(每小時有18種測資)共12個月。
  + **test**.csv:從剩下的數据當中取樣出連續的10小時為一筆，前九小時的所有觀測數據當作feature，第十小時的PM2.5當作answer。一共取出240筆不重複的 test data，請根據feature預測這240筆的PM2.5。

2. 代碼實現

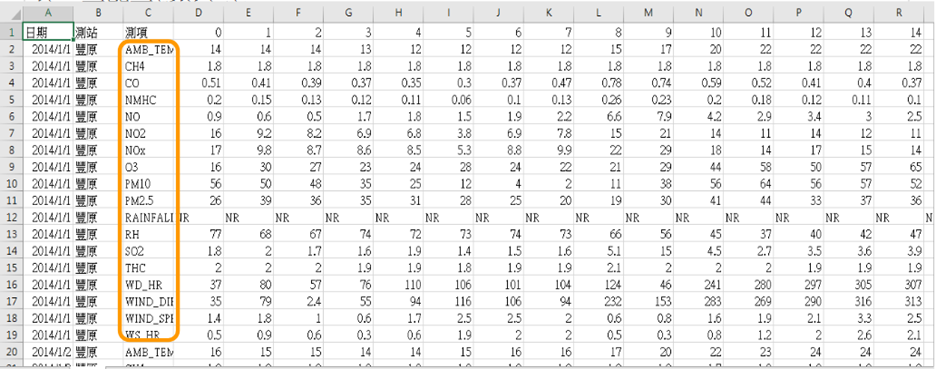
* 要求：根據前9小時的數据，用線性回歸來預測第10個小時的PM2.5的數值。
* 數据說明：

本次作業使用了某個檢測站一年的觀測數据，數据中每個小時有18個觀測指標，將其作為特徵。將數據分為train.csv和test.csv，其中**train.csv**是該檢測站每個月前20天的所有數据，**test.csv**是從該檢測站剩餘數据中取樣出的部分數据。

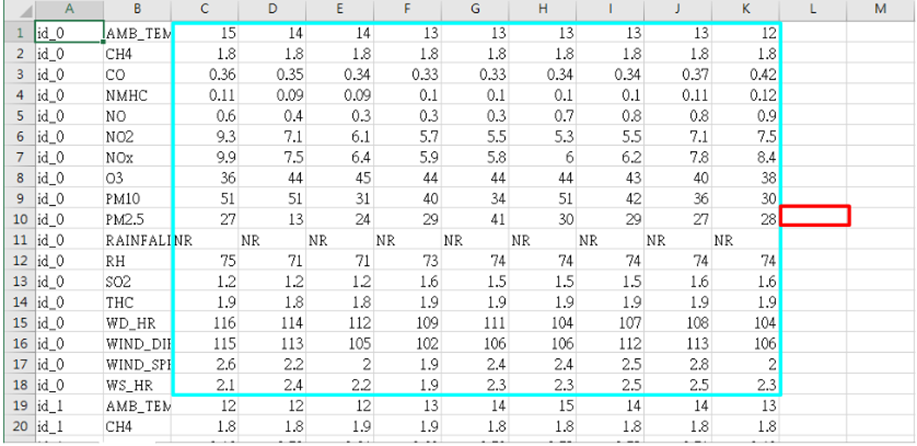
* **train.csv**：每個月前20天的完整數据
* **test.csv**：從剩下的數据中取樣連續的10小時為一組，前9小時所有觀測數据當做feature，第10小時的PM2.5當做answer。一共取出240組不重複的test data，請根據feature預測這240組的PM2.5.

所有的數据含有18個觀測數据（特徵）：AMB\_TEMP, CH4, CO, NHMC, NO, NO2, NOx, O3, PM10, PM2.5, RAINFALL, RH, SO2, THC, WD\_HR, WIND\_DIREC, WIND\_SPEED, WS\_HR。

**train.csv：**



**test.csv：**



作業概述：

* 輸入：9個小時的數据，共18項特徵（AMB\_TEMP, CH4, CO, NHMC, NO, NO2, NOx, O3, PM10, PM2.5, RAINFALL, RH, SO2, THC, WD\_HR, WIND\_DIREC, WIND\_SPEED, WS\_HR）
* 輸出：第10小時的PM2.5數值
* 模型：線性回歸

原始代碼（baseline）

**\*請先將train.csv及test.csv上傳至Colab平臺（此部份看教師操作）**

from google.colab import files

files.upload()

載入train.csv

import sys

import pandas as pd

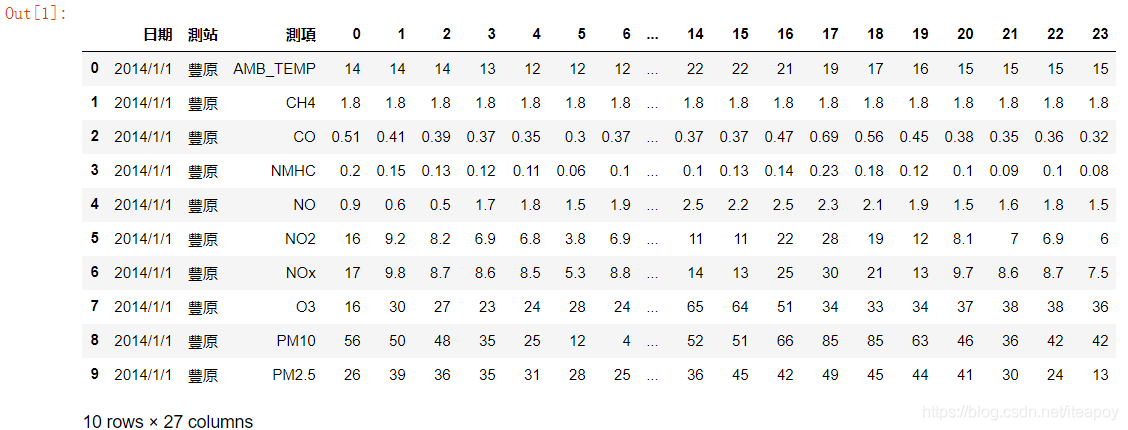
import numpy as np

# 讀入train.csv，繁體字以big5編碼

data = pd.read\_csv('./train.csv', encoding = 'big5')

# 顯示前10行

data.head(10)



第1列是日期，第2列是觀測站所在地，第3列是觀測指標，第4列-第27列是0-23共24小時。

data.shape



數据規格為：4320行，27列

**預處理**

可以看到降雨（rainfall）都是字元“NR”，將它變成數值0；從第3列開始是數值數据，提取出這些數值。

說明：下述代碼中的.to\_numpy()函數需要pandas版本>=0.24.0，否則會報錯

# 丟棄前兩列，需要的是從第三列開始的數值

data = data.iloc[:, 3:]

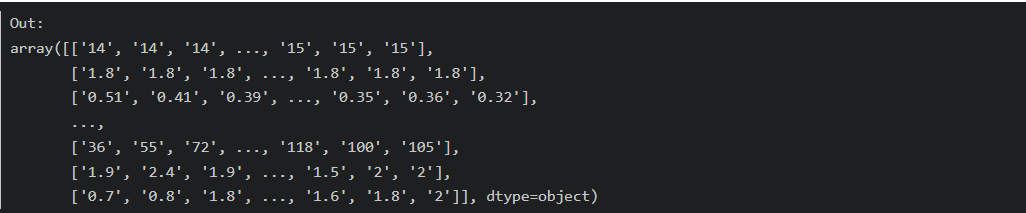
# 把降雨的NR字元變成數值0

data[data == 'NR'] = 0

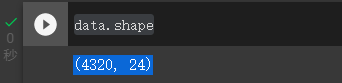
# 把dataframe轉換成numpy的陣列

raw\_data = data.to\_numpy()

raw\_data

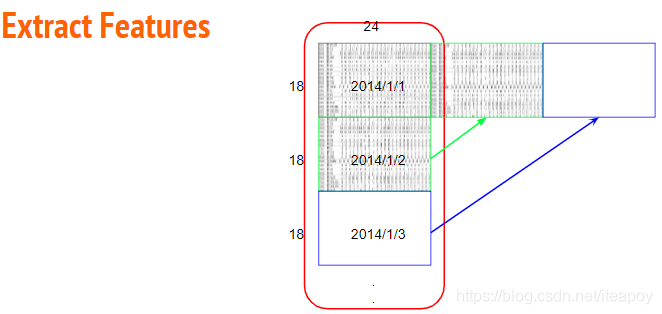


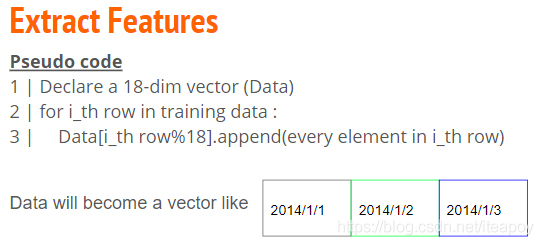
data.shape



此時，數據變成（4320行，24列）

**提取特徵**





4320行中，每18行（18個觀測指標）是一天的數据，將18行作為一天，4320/18=240天（一年12個月，每個月20天），根據每個月將4320行×24列的數据分成12 組18 行(features) × 480 列(hours) 的數據：

month\_data = {}

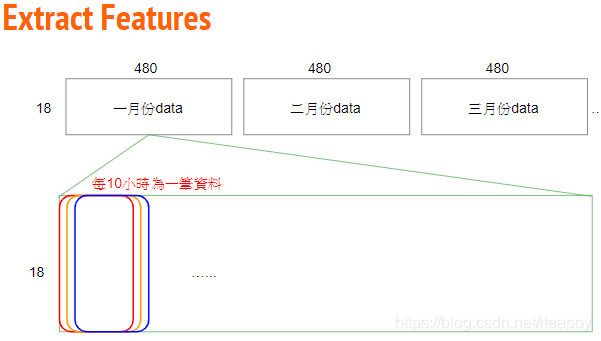
for month in range(12):

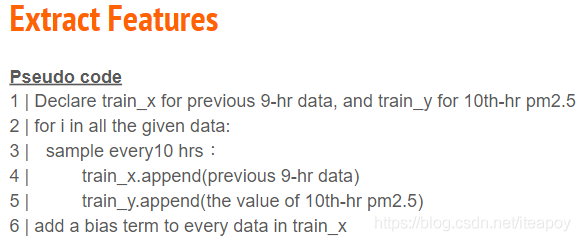
sample = np.empty([18, 480])

for day in range(20):

sample[:, day \* 24 : (day + 1) \* 24] = raw\_data[18 \* (20 \* month + day) : 18 \* (20 \* month + day + 1), :]

month\_data[month] = sample





分成了12個月，每個月有18行×480列的數据。

對於每個月，每10個小時分成一組，由前9個小時的數据來預測第10個小時的PM2.5，把前9小時的數据放入x，把第10個小時的數据放入y。視窗的大小為10，從第1個小時開始向右滑動，每次滑動1小時。因此，每個月都有471組這樣的數据。

接著把一組18×9的數据平鋪成一行向量，然後放入x的一行中，每個月有471組，共有12×471組向量，因此x有12×471行，18×9列。

將預測值放入y中，y有12（月）×471（組）行，1列。

x = np.empty([12 \* 471, 18 \* 9], dtype = float)

y = np.empty([12 \* 471, 1], dtype = float)

for month in range(12):

for day in range(20):

for hour in range(24):

if day == 19 and hour > 14:

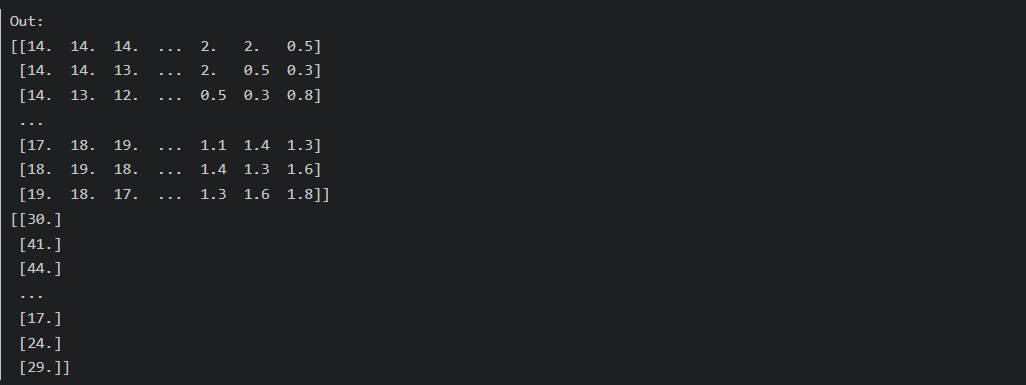
continue

x[month \* 471 + day \* 24 + hour, :] = month\_data[month][:,day \* 24 + hour : day \* 24 + hour + 9].reshape(1, -1) #vector dim:18\*9 (9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9)

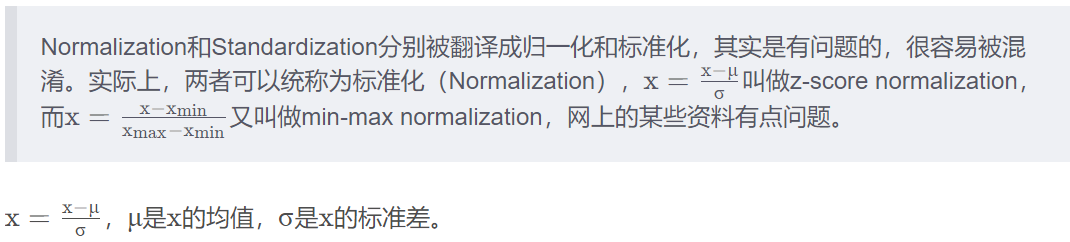
y[month \* 471 + day \* 24 + hour, 0] = month\_data[month][9, day \* 24 + hour + 9] #value

print(x)

print(y)



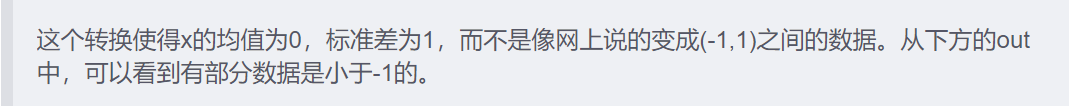
標準化（Normalization）



通過標準化，可以：

將有量綱的運算式，經過變換，化為無量綱的運算式，成為標量

使得數据更加符合獨立同分佈條件



這裡每一列是一個觀測指標，按列進行標準化。

mean\_x = np.mean(x, axis = 0) #18 \* 9

std\_x = np.std(x, axis = 0) #18 \* 9

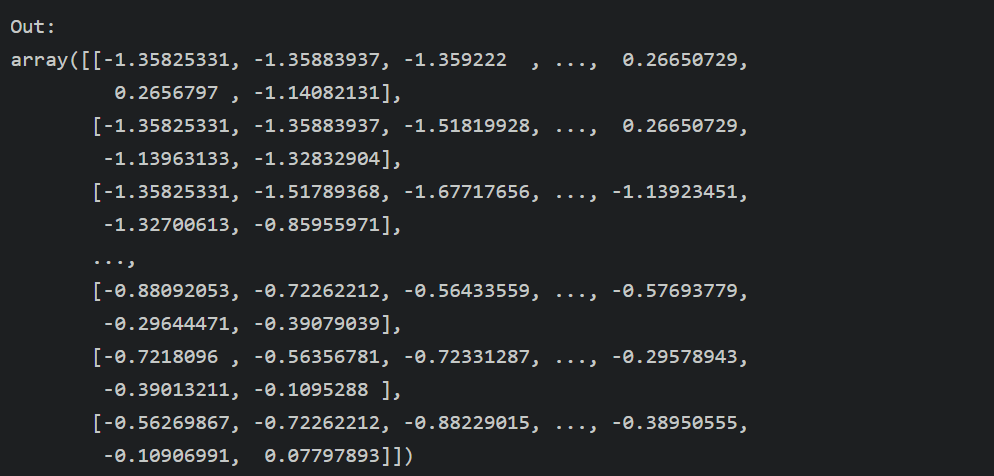
for i in range(len(x)): #12 \* 471

for j in range(len(x[0])): #18 \* 9

if std\_x[j] != 0:

x[i][j] = (x[i][j] - mean\_x[j]) / std\_x[j]

x



把訓練數据分成訓練集train\_set和驗證集validation，其中train\_set用於訓練，而validation不會參與訓練，僅用於驗證。（在baseline中並沒有用）

import math

x\_train\_set = x[: math.floor(len(x) \* 0.8), :]

y\_train\_set = y[: math.floor(len(y) \* 0.8), :]

x\_validation = x[math.floor(len(x) \* 0.8): , :]

y\_validation = y[math.floor(len(y) \* 0.8): , :]

print(x\_train\_set)

print(y\_train\_set)

print(x\_validation)

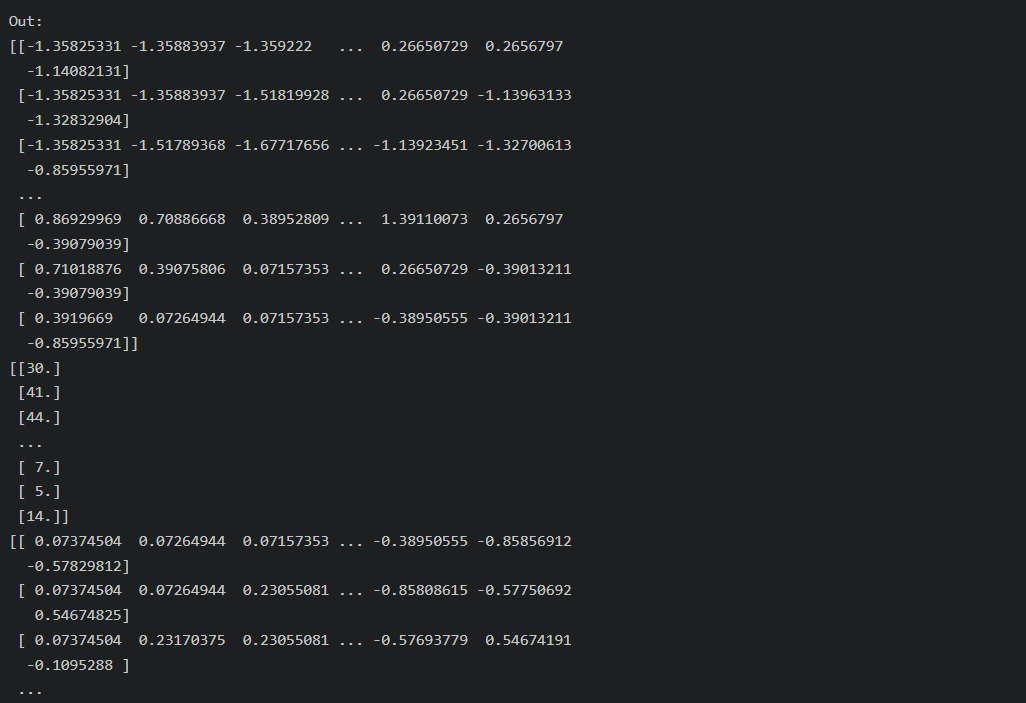
print(y\_validation)

print(len(x\_train\_set))

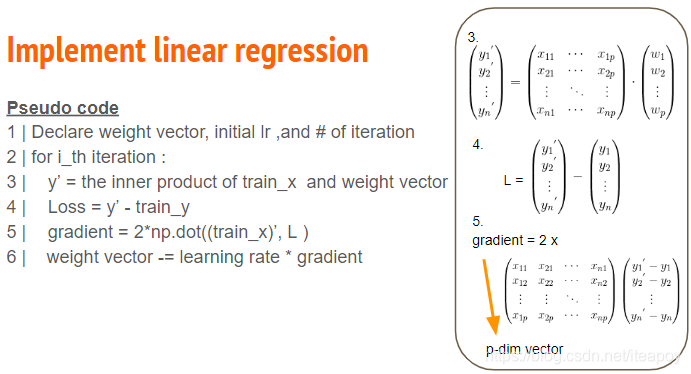
print(len(y\_train\_set))

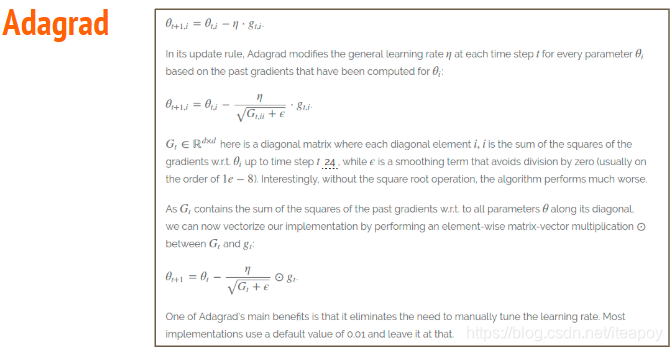
print(len(x\_validation))

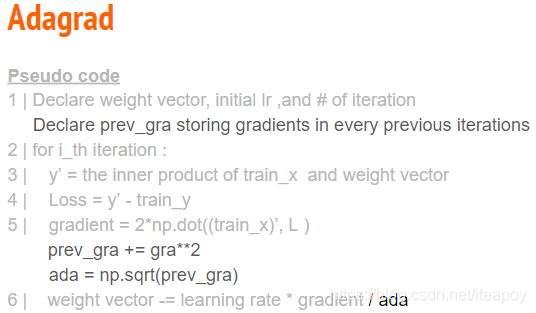
print(len(y\_validation))



訓練







和上圖不同處: 下面Loss的代碼用到的是 Root Mean Square Error

因為存在常數項b，所以維度（dim）需要多加一列，即原來是 y = wx + b ，可以統一成 y = [w b] [x 1]；eps項是極小值，避免adagrad的分母為0.

每一個維度（dim）會對應到各自的gradient和權重w，通過一次次的反覆運算（iter\_time）學習。最終，將訓練得到的模型（權重w）存儲為.npy格式的檔。

dim = 18 \* 9 + 1

w = np.zeros([dim, 1])

x = np.concatenate((np.ones([12 \* 471, 1]), x), axis = 1).astype(float)

learning\_rate = 100

iter\_time = 1000

adagrad = np.zeros([dim, 1])

eps = 0.0000000001

for t in range(iter\_time):

loss = np.sqrt(np.sum(np.power(np.dot(x, w) - y, 2))/471/12)#rmse

if(t%100==0):

print(str(t) + ":" + str(loss))

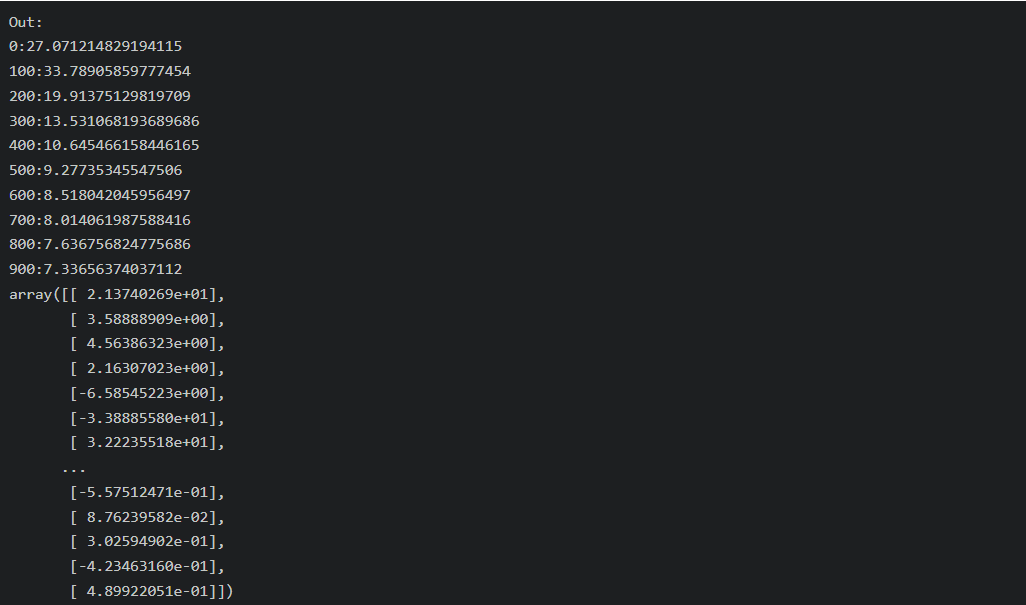
gradient = 2 \* np.dot(x.transpose(), np.dot(x, w) - y) #dim\*1

adagrad += gradient \*\* 2

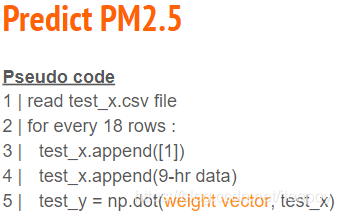
w = w - learning\_rate \* gradient / np.sqrt(adagrad + eps)

np.save('weight.npy', w)

w



測試



# 讀入測試數据test.csv

testdata = pd.read\_csv('./test.csv', header = None, encoding = 'big5')

# 丟棄前兩列，需要的是從第3列開始的數据

test\_data = testdata.iloc[:, 2:]

# 把降雨為NR字元變成數位0

test\_data[test\_data == 'NR'] = 0

# 將dataframe變成numpy陣列

test\_data = test\_data.to\_numpy()

# 將test數據也變成 240 個維度為 18 \* 9 + 1 的數據。

test\_x = np.empty([240, 18\*9], dtype = float)

for i in range(240):

test\_x[i, :] = test\_data[18 \* i: 18\* (i + 1), :].reshape(1, -1)

for i in range(len(test\_x)):

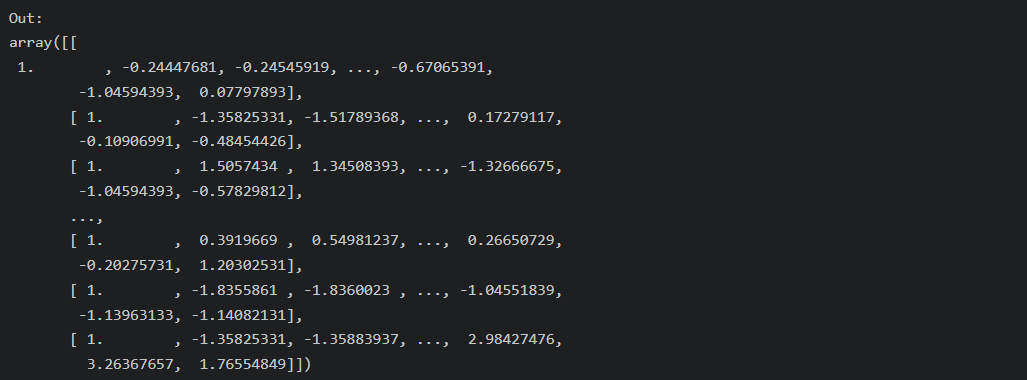
for j in range(len(test\_x[0])):

if std\_x[j] != 0:

test\_x[i][j] = (test\_x[i][j] - mean\_x[j]) / std\_x[j]

test\_x = np.concatenate((np.ones([240, 1]), test\_x), axis = 1).astype(float)

test\_x

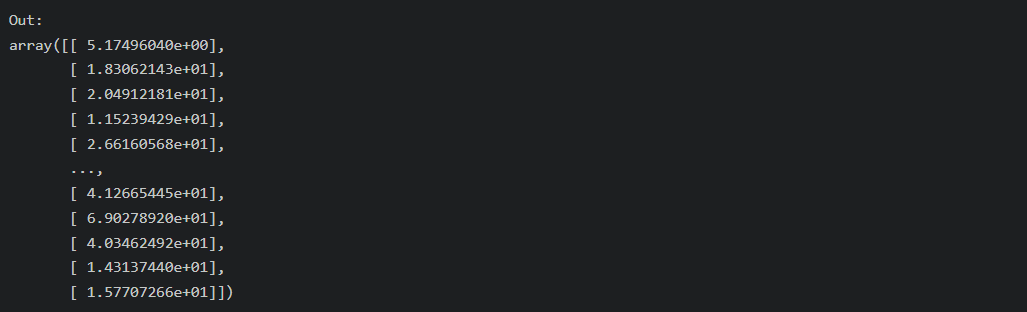


載入模型即可對test數据進行預測，得到預測值ans\_y。

w = np.load('weight.npy')

ans\_y = np.dot(test\_x, w)

ans\_y



把預測值保存為CSV檔

import csv

with open('submit.csv', mode='w', newline='') as submit\_file:

csv\_writer = csv.writer(submit\_file)

header = ['id', 'value']

print(header)

csv\_writer.writerow(header)

for i in range(240):

row = ['id\_' + str(i), ans\_y[i][0]]

csv\_writer.writerow(row)

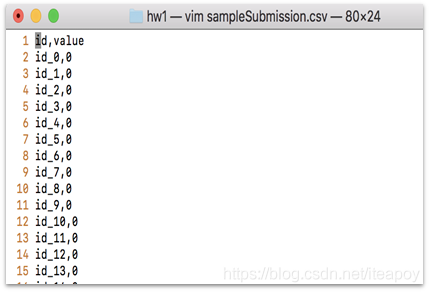
print(row)



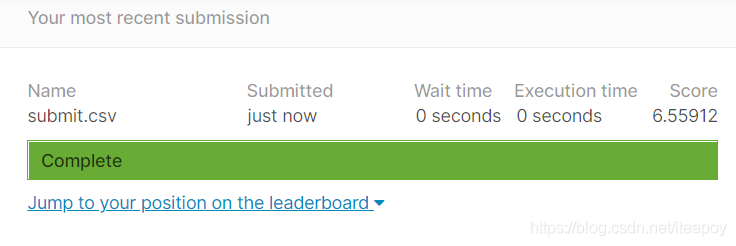
結果評測

將240組testing data 中的 PM2.5 值存為submit.csv文件。將submit.csv提交至kaggle平臺進行測試，submit.csv內的格式為：

* 第一行必須是 id,value
* 第二行開始，每行分別為 id 值及預測 PM2.5 的數值，以逗號隔開。



直接將生成的submit.csv提交至kaggle測試結果如下(\*參考：此是此博主參加kaggle競賽要上傳結果供系統作評量之用)：

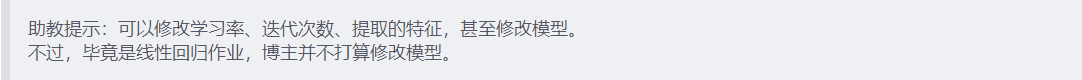




Private Score為8.73773，Public Score為6.55912

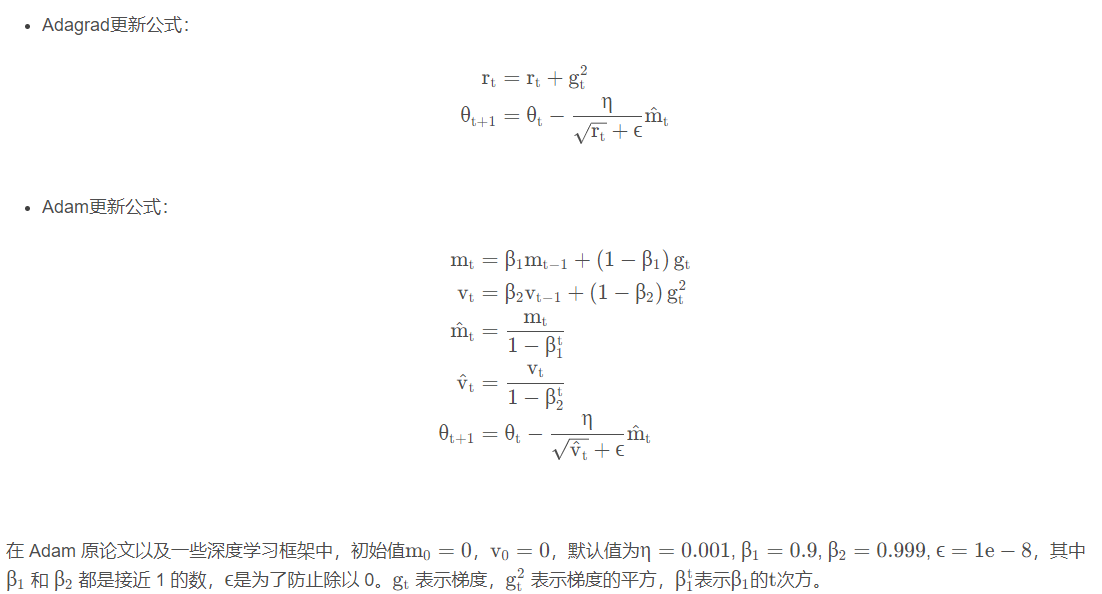
接下來，需要在此baseline的基礎上優化模型，目標是降低Score。

3 修改代碼



版本1

博主嘗試了修改成adam，然而效果並不好，最後還是改回了Adagrad。



最後，博主調整學習率為2，反覆運算6000次（多次煉丹的結果）。

除此之外，觀察PM2.5的值，可以發現沒有負數，都是整數，因此對預測值進行了微調，小於0的數都歸為0，而對所有的浮點數四捨五入為整數。

完整代碼如下：

# 導入相關庫

import sys

import pandas as pd

import numpy as np

# 讀入數據

data = pd.read\_csv('./train.csv', encoding = 'big5')

# 數據預處理

data = data.iloc[:, 3:]

data[data == 'NR'] = 0

raw\_data = data.to\_numpy()

# 按月分割數據

month\_data = {}

for month in range(12):

sample = np.empty([18, 480])

for day in range(20):

sample[:, day \* 24 : (day + 1) \* 24] = raw\_data[18 \* (20 \* month + day) : 18 \* (20 \* month + day + 1), :]

month\_data[month] = sample

# 分割x和y

x = np.empty([12 \* 471, 18 \* 9], dtype = float)

y = np.empty([12 \* 471, 1], dtype = float)

for month in range(12):

for day in range(20):

for hour in range(24):

if day == 19 and hour > 14:

continue

x[month \* 471 + day \* 24 + hour, :] = month\_data[month][:,day \* 24 + hour : day \* 24 + hour + 9].reshape(1, -1) #vector dim:18\*9 (9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9)

y[month \* 471 + day \* 24 + hour, 0] = month\_data[month][9, day \* 24 + hour + 9] #value

print(x)

print(y)

# 對x標準化

mean\_x = np.mean(x, axis = 0) #18 \* 9

std\_x = np.std(x, axis = 0) #18 \* 9

for i in range(len(x)): #12 \* 471

for j in range(len(x[0])): #18 \* 9

if std\_x[j] != 0:

x[i][j] = (x[i][j] - mean\_x[j]) / std\_x[j]

# 訓練模型並保存權重

dim = 18 \* 9 + 1

w = np.zeros([dim, 1])

x2 = np.concatenate((np.ones([12 \* 471, 1]), x), axis = 1).astype(float)

learning\_rate = 2

iter\_time = 10000

adagrad = np.zeros([dim, 1])

eps = 1e-7

for t in range(iter\_time):

loss = np.sqrt(np.sum(np.power(np.dot(x2, w) - y, 2))/471/12)#rmse

if(t%100==0):

print(str(t) + ":" + str(loss))

gradient = 2 \* np.dot(x2.transpose(), np.dot(x2, w) - y) #dim\*1

adagrad += gradient \*\* 2

w = w - learning\_rate \* gradient / (np.sqrt(adagrad) + eps)

np.save('weight.npy', w)

# 導入測試數据test.csv

testdata = pd.read\_csv('./test.csv', header = None, encoding = 'big5')

test\_data = testdata.iloc[:, 2:]

test\_data[test\_data == 'NR'] = 0

test\_data = test\_data.to\_numpy()

test\_x = np.empty([240, 18\*9], dtype = float)

for i in range(240):

test\_x[i, :] = test\_data[18 \* i: 18\* (i + 1), :].reshape(1, -1)

for i in range(len(test\_x)):

for j in range(len(test\_x[0])):

if std\_x[j] != 0:

test\_x[i][j] = (test\_x[i][j] - mean\_x[j]) / std\_x[j]

test\_x = np.concatenate((np.ones([240, 1]), test\_x), axis = 1).astype(float)

# 對test的x進行預測，得到預測值ans\_y

w = np.load('weight.npy')

ans\_y = np.dot(test\_x, w)

# 加一個預處理<0的都變成0

for i in range(240):

if(ans\_y[i][0]<0):

ans\_y[i][0]=0

else:

ans\_y[i][0]=np.round(ans\_y[i][0])

# 保存為csv文件，並提交到kaggle：https://www.kaggle.com/c/ml2020spring-hw1/submissions

import csv

with open('submit.csv', mode='w', newline='') as submit\_file:

csv\_writer = csv.writer(submit\_file)

header = ['id', 'value']

print(header)

csv\_writer.writerow(header)

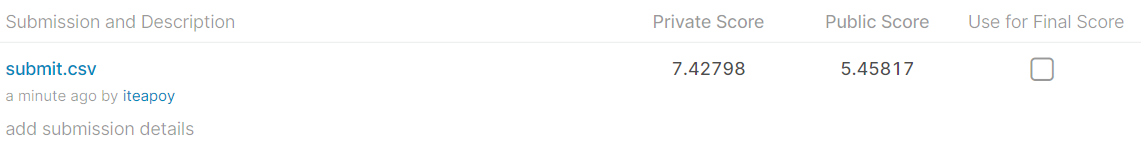
for i in range(240):

row = ['id\_' + str(i), ans\_y[i][0]]

csv\_writer.writerow(row)

print(row)

最終，博主的測試結果（100名開外）：



版本2

之後，博主又嘗試了加入x的平方項，private score的分數下降了很多，不過public score的分數上升了。

主要的修改部分為：

# 訓練集

for month in range(12):

for day in range(20):

for hour in range(24):

if day == 19 and hour > 14:

continue

x1 = month\_data[month][:, day \* 24 + hour: day \* 24 + hour + 9].reshape(1,-1) # vector dim:18\*9 (9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9)

x[month \* 471 + day \* 24 + hour, :18 \* 9] = x1

# 在這裡加入了x的二次項

x[month \* 471 + day \* 24 + hour, 18 \* 9: 18 \* 9 \* 2] = np.power(x1, 2)

y[month \* 471 + day \* 24 + hour, 0] = month\_data[month][9, day \* 24 + hour + 9] # value

# 測試集

testdata = pd.read\_csv('./test.csv', header = None, encoding = 'big5')

test\_data = testdata.iloc[:, 2:]

test\_data[test\_data == 'NR'] = 0

test\_data = test\_data.to\_numpy()

test\_x1 = np.empty([240, 18\*9], dtype = float)

test\_x = np.empty([240, 18\*9\*2], dtype = float)

for i in range(240):

test\_x1 = test\_data[18 \* i: 18 \* (i + 1), :].reshape(1, -1).astype(float)

# 同樣在這裡加入test x的二次項

test\_x[i, : 18 \* 9] = test\_x1

test\_x[i, 18 \* 9:] = np.power(test\_x1 , 2)

for i in range(len(test\_x)):

for j in range(len(test\_x[0])):

if std\_x[j] != 0:

test\_x[i][j] = (test\_x[i][j] - mean\_x[j]) / std\_x[j]

test\_x = np.concatenate((np.ones([240, 1]), test\_x), axis = 1).astype(float)

完整代碼如下：

import sys

import pandas as pd

import numpy as np

# 讀入數據

data = pd.read\_csv('./train.csv', encoding='big5')

# 數據預處理

data = data.iloc[:, 3:]

data[data == 'NR'] = 0

raw\_data = data.to\_numpy()

# 按月分割數據

month\_data = {}

for month in range(12):

sample = np.empty([18, 480])

for day in range(20):

sample[:, day \* 24: (day + 1) \* 24] = raw\_data[18 \* (20 \* month + day): 18 \* (20 \* month + day + 1), :]

month\_data[month] = sample

# 分割x和y

x = np.empty([12 \* 471, 18 \* 9 \* 2], dtype=float)

y = np.empty([12 \* 471, 1], dtype=float)

for month in range(12):

for day in range(20):

for hour in range(24):

if day == 19 and hour > 14:

continue

x1 = month\_data[month][:, day \* 24 + hour: day \* 24 + hour + 9].reshape(1,-1) # vector dim:18\*9 (9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9)

x[month \* 471 + day \* 24 + hour, :18 \* 9] = x1

# 在這裡加入了x的二次項

x[month \* 471 + day \* 24 + hour, 18 \* 9: 18 \* 9 \* 2] = np.power(x1, 2)

y[month \* 471 + day \* 24 + hour, 0] = month\_data[month][9, day \* 24 + hour + 9] # value

# 對x標準化

mean\_x = np.mean(x, axis=0) # 18 \* 9 \* 2

std\_x = np.std(x, axis=0) # 18 \* 9 \* 2

for i in range(len(x)): # 12 \* 471

for j in range(len(x[0])): # 18 \* 9 \* 2

if std\_x[j] != 0:

x[i][j] = (x[i][j] - mean\_x[j]) / std\_x[j]

# 隨機打散X和Y

def \_shuffle(X, Y):

randomize = np.arange(len(X))

np.random.shuffle(randomize)

return (X[randomize], Y[randomize])

# 訓練模型並保存權重

dim = 18 \* 9 \* 2 + 1

w = np.ones([dim, 1])

learning\_rate = 2

iter\_time = 5000

adagrad = np.zeros([dim, 1])

eps = 1e-7

for t in range(iter\_time):

x, y = \_shuffle(x, y)

x2 = np.concatenate((np.ones([len(x), 1]), x), axis=1).astype(float)

gradient = 2 \* np.dot(x2.transpose(), np.dot(x2, w) - y) # dim\*1

adagrad += gradient \*\* 2

w = w - learning\_rate \* gradient / (np.sqrt(adagrad) + eps)

loss = np.sqrt(np.sum(np.power(np.dot(x2, w) - y, 2)) / len(x)) # rmse

if (t % 100 == 0):

print(str(t) + ":" + str(loss))

np.save('weight.npy', w)

# 導入測試數据test.csv

testdata = pd.read\_csv('./test.csv', header = None, encoding = 'big5')

test\_data = testdata.iloc[:, 2:]

test\_data[test\_data == 'NR'] = 0

test\_data = test\_data.to\_numpy()

test\_x1 = np.empty([240, 18\*9], dtype = float)

test\_x = np.empty([240, 18\*9\*2], dtype = float)

for i in range(240):

test\_x1 = test\_data[18 \* i: 18 \* (i + 1), :].reshape(1, -1).astype(float)

# 同樣在這裡加入test x的二次項

test\_x[i, : 18 \* 9] = test\_x1

test\_x[i, 18 \* 9:] = np.power(test\_x1 , 2)

for i in range(len(test\_x)):

for j in range(len(test\_x[0])):

if std\_x[j] != 0:

test\_x[i][j] = (test\_x[i][j] - mean\_x[j]) / std\_x[j]

test\_x = np.concatenate((np.ones([240, 1]), test\_x), axis = 1).astype(float)

# 對test的x進行預測，得到預測值ans\_y

w = np.load('weight.npy')

ans\_y = np.dot(test\_x, w)

# 加一個預處理<0的都變成0

for i in range(240):

if(ans\_y[i][0]<0):

ans\_y[i][0]=0

else:

ans\_y[i][0]=np.round(ans\_y[i][0])

# 保存為csv文件，並提交到kaggle：https://www.kaggle.com/c/ml2020spring-hw1/submissions

import csv

with open('submit.csv', mode='w', newline='') as submit\_file:

csv\_writer = csv.writer(submit\_file)

header = ['id', 'value']

print(header)

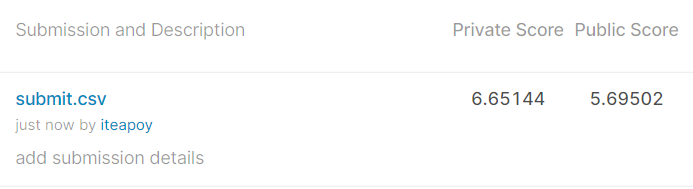
csv\_writer.writerow(header)

for i in range(240):

row = ['id\_' + str(i), ans\_y[i][0]]

csv\_writer.writerow(row)

結果如下：



Private Score：6.65144

Public Score：5.69502

————————————————

原文連結：https://blog.csdn.net/iteapoy/article/details/105431738