說明：請各位使用此template進行Report撰寫，如果想要用其他排版模式也請註明題號以及題目內容（請勿擅自更改題號），最後上傳至cool前，請務必轉成PDF檔，並且命名為report.pdf，否則將不予計分。

備註 :   
 a. NR請皆設為0，其他的非數值(特殊字元)可以自己判斷

b. 所有 advanced 的 gradient descent 技術(如: adam, adagrad 等) 都是可以用的

c. 第1～2題請都以題目給訂的兩種model來回答

d. 同學可以先把model訓練好，kaggle死線之後便可以無限上傳。

----------------------------------------閱讀完以上文字請刪除------------------------------------

學號：R11921A16 系級： 電機碩一 姓名：何秉學

1. (1%) 解釋什麼樣的data preprocessing可以improve你的training/testing accuracy，e.g., 你怎麼挑掉你覺得不適合的data points。請提供數據(例如 kaggle public score RMSE)以佐證你的想法。

2. (1%) 請實作 2nd-order polynomial regression model (不用考慮交互項)。

1. 貼上 polynomial regression 版本的 Gradient descent code 內容

## Code

**def** minibatch**(**x**,** y**,** config**):**

# Randomize the data in minibatch

index **=** np**.**arange**(**x**.**shape**[**0**])**

np**.**random**.**shuffle**(**index**)**

x **=** x**[**index**]**

y **=** y**[**index**]**

# Initialization

batch\_size **=** config**.**batch\_size

lr **=** config**.**lr

lam **=** config**.**lam

epoch **=** config**.**epoch

beta\_1 **=** np**.**full**(**x**[**0**].**shape**,** 0.9**).**reshape**(-**1**,** 1**)**

beta\_2 **=** np**.**full**(**x**[**0**].**shape**,** 0.99**).**reshape**(-**1**,** 1**)**

# Linear regression: only contains two parameters (w, b).

w **=** np**.**full**(**x**[**0**].**shape**,** 0.1**).**reshape**(-**1**,** 1**)**

w2 **=** np**.**full**(**x**[**0**].**shape**,** 0.1**).**reshape**(-**1**,** 1**)** # Implement 2-nd polynomial regression

bias **=** 0.1

m\_t **=** np**.**full**(**x**[**0**].**shape**,** 0**).**reshape**(-**1**,** 1**)**

v\_t **=** np**.**full**(**x**[**0**].**shape**,** 0**).**reshape**(-**1**,** 1**)**

m\_t\_2 **=** np**.**full**(**x**[**0**].**shape**,** 0**).**reshape**(-**1**,** 1**)** # Implement 2-nd polynomial regression

v\_t\_2 **=** np**.**full**(**x**[**0**].**shape**,** 0**).**reshape**(-**1**,** 1**)** # Implement 2-nd polynomial regression

m\_t\_b **=** 0.0

v\_t\_b **=** 0.0

t **=** 0

epsilon **=** 1e-8

# Training loop

**for** num **in** **range(**epoch**):**

**for** b **in** **range(int(**x**.**shape**[**0**]/**batch\_size**)):**

t**+=**1

x\_batch **=** x**[**b **\*** batch\_size**:(**b**+**1**)** **\*** batch\_size**]**

y\_batch **=** y**[**b **\*** batch\_size**:(**b**+**1**)** **\*** batch\_size**].**reshape**(-**1**,**1**)**

# Prediction of linear regression

# pred = np.dot(x\_batch, w) + bias

# Implement 2-nd polynomial regression

pred **=** np**.**dot**(**x\_batch**,** w**)** **+** np**.**dot**(**x\_batch**\*\***2**,** w2**)** **+** bias

# loss

loss **=** y\_batch **-** pred

# Compute w gradient

g\_t **=** np**.**dot**(**x\_batch**.**transpose**(),** loss**)** **\*** **(-**2**)** **+** 2 **\*** lam **\*** np**.sum(**w**)**

m\_t **=** beta\_1 **\*** m\_t **+** **(**1**-**beta\_1**)** **\*** g\_t

v\_t **=** beta\_2 **\*** v\_t **+** **(**1**-**beta\_2**)** **\*** np**.**multiply**(**g\_t**,** g\_t**)**

m\_cap **=** m\_t **/** **(**1**-(**beta\_1**\*\***t**))**

v\_cap **=** v\_t **/** **(**1 **-** **(**beta\_2**\*\***t**))**

# Compute w2 gradient

g\_t\_2 **=** np**.**dot**(**x\_batch**.**transpose**(),** loss**)** **\*** **(-**2**)** **+** 2 **\*** lam **\*** np**.sum(**w2**)**

m\_t\_2 **=** beta\_1 **\*** m\_t\_2 **+** **(**1**-**beta\_1**)** **\*** g\_t\_2

v\_t\_2 **=** beta\_2 **\*** v\_t\_2 **+** **(**1**-**beta\_2**)** **\*** np**.**multiply**(**g\_t\_2**,** g\_t\_2**)**

m\_cap\_2 **=** m\_t\_2 **/** **(**1**-(**beta\_1**\*\***t**))**

v\_cap\_2 = v\_t\_2 / (1 - (beta\_2\*\*t))

# Compute bias gradient

g\_t\_b = loss.sum(axis=0) \* (-2)

m\_t\_b = 0.9 \* m\_t\_b + (1 - 0.9) \* g\_t\_b

v\_t\_b = 0.99 \* v\_t\_b + (1 - 0.99) \* (g\_t\_b \* g\_t\_b)

m\_cap\_b = m\_t\_b / (1 - (0.9\*\*t))

v\_cap\_b = v\_t\_b / (1 - (0.99\*\*t))

w\_0 = np.copy(w)

# Update weight & bias

w -= ((lr \* m\_cap) / (np.sqrt(v\_cap) + epsilon)).reshape(-1, 1)

w2 -= ((lr \* m\_cap\_2) / (np.sqrt(v\_cap\_2) + epsilon)).reshape(-1, 1)

bias -= (lr \* m\_cap\_b) / (math.sqrt(v\_cap\_b) + epsilon)

return w, bias

1. 在只使用 NO 數值作為 feature 的情況下，紀錄該 model 所訓練出的 parameter 數值（w2, w1, b）以及 kaggle public score.

3.(4%) Refer to math problem:

<https://hackmd.io/@lH2AB7kCSAS3NPw2FffsGg/Sk1n8xPWo?fbclid=IwAR0LiCps2fhIZFJT-gYP8kr7KlvLaRvS9-ftLIaPQY5DVgye1AuHM-RW3Yg>