Machine Learning 2016
系級：電機四
姓名：鍾勝隆
學號：B02901001

HW1 Report – PM2.5 Prediction

1.(1%) Linear regression function by Gradient Descent.

```
#intial coefficient  (import numpy as np)
weight = np.zeros((1, 162))
bias = 0
learning_rate = 0.2
learning_time = 8964
#Regularization
Lambda = 0
#for Adagrad
G_w = np.zeros((1, 162))
G_b = 0

t = 1
while(True): #Start training
  change = ttraining_results - bias - np.sum((training_datas * weight), axis=1)
  b_w = change.sum()
  g_w = np.sum((np.transpose(training_datas) * \
        change), axis=1) - Lambda * weight

  #gradient
  gradient_w = -2 * g_w
  gradient_b = -2 * b_w
  G_w += gradient_w ** 2          #Adagrad
  G_b += gradient_b ** 2
  weight = weight - learning_rate * (1 / (G_w) ** 0.5 ) * gradient_w
  bias = bias - learning_rate * (1 / (G_b) ** 0.5 ) * gradient_b
  t += 1
  if (t % 10 == 0):
    print("The", t, "times")
  if ( t > learning_time):
    print ("Linear Regression training is done.")
    break
```

2. (1%) Describe your method.

首先，在開始 Linear regression 之前，必須先決定要用多少的 feature 去預測第 10 小時的 PM2.5，由於測資是給前連續 9 個小時的 18 項天氣觀測資料，故我以此 162(9*18)資料爲 feature 去進行 training。我將 train.csv 中同月份的連續九小時資料取出爲一筆資料(包含跨日的資料)：

$$((x_1^n, x_{2,\ldots}^n, x_{162}^n), y^n)$$

共可以產生 5652 筆 data，使用 numpy 的二維 array(shape 爲(5652*162))儲存，設定參數一維 array，每個 feature 都有對應的 weight，故我初始化 162 個 0 起始值，由於 linear regresion 的 loss function 沒有 local minimum，初始值不需要特別決定，以及設 bias 也爲 0 。

在 training 中，先計算以 loss function：

$$\sum_{x^n, y^n \in Training set} \left(y^n - \sum_{i=1}^{162} (x_i^n w_i) - b\right)^2 + \sum \lambda (w_i^2)$$

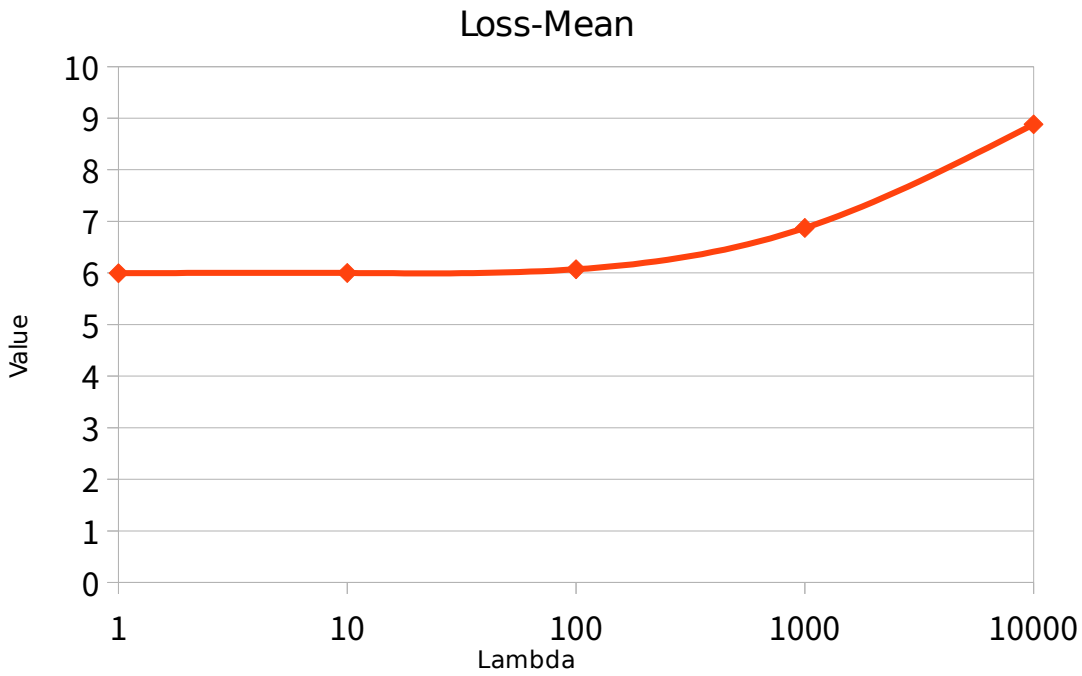
對應各個 feature 的 gradient，因爲我有使用 Adagrad，故將每次計算的 gradient 平方後，累積在 G_w 和 G_b ，以 $1/\sqrt{(G_w)}$ 和 $1/\sqrt{(G_b)}$ 再乘上 learning rate 和 gradient，加上原本的 weight 和 bias，即完成 Adagrad 的 Gradient Descent。

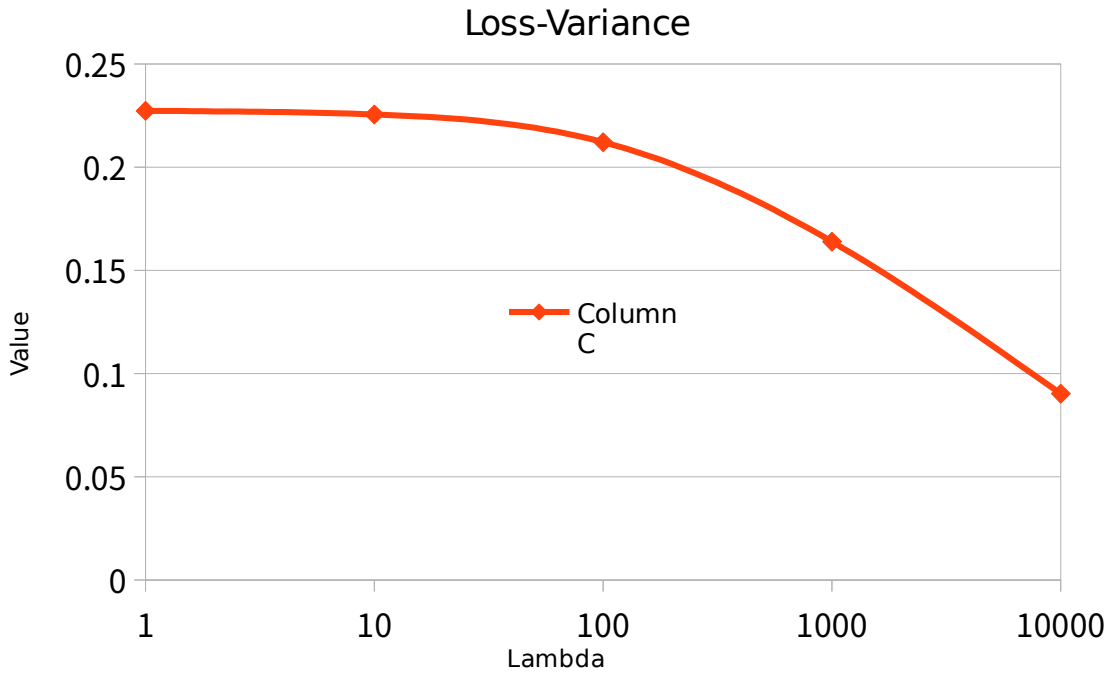
至於要如何設定 learning rate，training time(次數)，以及 regularization 的係數 λ ，我是透過 **K-fold cross-validation(交互驗證)**去評估各個參數的最佳值，在 data_process.py 將全部的資料**分成 10 等分**，其中 1 份當作 Validation set，剩下 9 份爲 Training set，故產生 10 組 Training set 和 Validation set，存在 data_validation 資料夾下，validation.py 中會讀進這些資料，平行 train 10 個獨立的 Training set，使用 Validation set 算 train 出資料的 loss，再由 10 個 loss 的 **mean** 和 **variance** 評估該參數建立的 model 是否較好，調整 learning rate，training times，和 λ ，以達到最低的 mean 和 variance，並在將該三個參數輸入 predictor.py 中，把全部的 5652 筆資料重新 train 一遍，讀入 test_X.csv，計算出答案。

3. (1%) Discussion on regularization.

使用 regularization 以避免 model train 到 overfitting，λ 太小的話就等於沒有作 regularization，但是，如果 λ 太大則無法使 loss 有效下降，變成 underfitting。

以下爲以相同的 learning rate = 0.2，learning time = 2000，在不同λ下，validation set loss 的 mean 和 variance 的實驗。





可由上面兩張圖看出λ變大，loss 的 variance 有效的下降，較不容易 overfitting，但卻使 loss 的 mean 上升，降低預測的準確度。

4. (1%) Discussion on learning rate.

Learning rate 的大小就等於是每次在更新 weight 和 bias 的大小，所以要是 learning rate 太大，無法慢慢接近最佳的參數值，甚至過度成長，反而使 loss 變大至無限。反之，要是太小，則是無論經過多少 learning time，loss 依然沒有改變。使用 Adagrad 後，因爲每次的 learning rate 會被實作 Adagrad 的係數牽制，進而可以大幅減少過大的 learning rate，但是特性是最終都會變得很小，所以 training time 太長會沒有顯著的改進。

5. Reference

NTUEE – ML2016Fall 上課投影片
http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML16.html
交叉驗證 cross-validation
https://cg2010studio.com/2012/10/22/%E4%BA%A4%E5%8F%89%E9%A9%97%E8%AD%89-cross-validation/
Wikipedia K-fold cross validation
https://www.wikiwand.com/zh/%E4%BA%A4%E5%8F%89%E9%A9%97%E8%AD%89
An overview of gradient descent optimization algorithms (Adagrad)
http://sebastianruder.com/optimizing-gradient-descent/

P.S.linear_regression.sh == kaggle_best.sh