機器學習(HW1)　姓名：袁培傑　學號：B03901134

Linear regression function by Gradient Descent

```
for times in range(iter) :
    data = mat[:, :, 0:8]
    loss = 0
    for i in range(9, 479) : # first train 0-8, last train 470-478
        ans = mat[1, :, i+1]
        data = np.insert(data, 8, mat[:, :, i], axis = 2)
        myans = [0]
        for index in range(num) :
            out = np.dot(data[index], np.transpose(par[index]))
            myans = myans + out
        myans = myans + bias
        dif = ans - myans
        # Gradient for parameters with regularization
        g = np.dot(dif, data[:, :, 0:9]) * -1 + lamda * 2 * par
        G = G + np.square(g)
        # Gradient for bias with regularization
        b = np.sum(2 * dif * -1) + lamda * 2 * bias
        B = B + b * b
        # Linear regression with adagrad
        par[2:num] = par[2:num] - lr * 2 * g[2:num] / np.sqrt(G)[2:num]
        bias = bias - lr * b / math.sqrt(B)

        # Set these parameters to zero
        par[0, 0] = par[0, 1] = par[0, 2] = par[0, 3] = par[0, 4] = par[0, 5]=0
        par[2, 0] = par[2, 1] = par[2, 2] = par[2, 3] = 0
        par[3, 0] = par[3, 1] = par[3, 2] = par[3, 3] = 0
        par[rain, 0] = par[rain, 1] = par[rain, 2] = par[rain, 3] = 0
        par[rain, 4] = par[rain, 5] = par[rain, 6]= 0
        for i in range (4, rain) :
            par[i, 0] = par[i, 1] = par[i, 2] = par[i, 3] = par[i, 4] = 0
        data = np.delete(data, 0, axis = 2)
        loss = loss + np.sum(dif * dif)
```

Linear regression using adgrad

Training Method

(1) Data

先將 Data 處理成 12 個月連接在一起，這樣可以 Train 更多相連的小時，因為大氣環境的變化是一個連續的，我臆測這樣準確率會更高。因為 testing data 是九個小時預測第十個小時，所以我參數設定為九個。Loss 的算法是將整年度預測的值，與第十個小時的值相減然後平方和，最後將所有 Loss 加起來。

(2) Feature

因為大氣環境是一個連續性的變動，所以我首先對 PM2.5 進行 training，而且我猜測預測值與實際值的差值，可以透過其他參數來彌補。所以我一開始先對 PM2.5 train 一次式，然後再 train 二次式，我將這兩組參數記錄下來加上別的

參數，發現加上別的參數後的 Loss 增加了，於是我將這兩組固定再重複 train，發現 Loss 比原本的還少。

在選 Feature 的時候我重複測試了很多組 Feature，進行一些排列組合，去找會產生最低 Loss 的組合，發現使用 PM2.5、O3、NOx、NO2、NO 時，會讓 Loss 最低。

(3) Domain Knowledge

根據相關研究顯示，PM2.5 懸浮微粒其產生通常是本地汽機車、工廠之廢氣所排放，而這些廢氣是起源於化石燃料，會產生 PM2.5、PM10、O3、NOx、NO2、NO、CO，與我的測試有些許相關性，於是我加入 PM10 以及 O3 這兩個參數，並減少九個小時中前幾個小時，測試出來的 Loss 確實變少許多。

下雨會讓 PM2.5 降低，是我從幾篇新聞跟我自己做出來的結果看出來的，懸浮微粒的濃度會被雨水沖刷而降低，在真實世界中，聽起來是很有道理的一件事。

還看到幾篇研究，顯示 NOx/NMHC 之比例與 O3 成正相關，所以我將 NMHC 取倒數加入 Model 裡面，Loss 也變少了。

Regulariztion

我調整 lamda 在 0.0001 到 100 之間，結果發現 lamda 比較高的時候，Loss 會比較高，一直找不到最好的結果。

再加上 Adagrad 之後，lamda 大小的影響更顯得微乎其微，所以我在 Regulariztion 上得到的修正，並沒有對我的結果造成明顯的進步。

Learning Rate

LR 不宜設定過大，容易造成參數數值暴增，數值會直接跑到無限大，對於不同 Model 有不同的 LR，我最適宜的 lr 為 0.000001，過低會 Train 太慢，過高會造成數值暴增。LR 根據不同的 Model，要做適當的調整，因為 Loss 的值根據算法不同，值的大小也有差，在用 Gradient Decent 的方法時，必須適時調整。

我觀察 Loss 發現在最後降低很慢，以為將 LR 隨著時間增加變大，會加速他到最低點的時間，結果發現 Loss 變得很浮動甚至越來越大。隨後我加上了 Adagrad 的方法，並將初始 LR 調高，隨著時間降低，發現 train 的速度有顯著的提升。

Other Discussion

在做 ML 時，Domain Knowledge 顯得格外重要，因為當 Data 很多的時候，選取 feature 的方法有很多種，如何選到正確的 feature 很重要。在這次作業中，我是很晚才找到一些相關學術資料，再去做測試才發現結果變更好，而且有些許研究上的結果，跟我之前在嘗試的 feature 前後呼應，如果今天有一大筆 Data 在眼前，我能夠去找到一些特徵，回推到 Data 本質的理論，可以是新的一種實驗研究方法。