Homework 1 Report – PM2.5Prediction

學號：b04502139 系級：電機三 姓名：戴瑋辰

1. 請分別使用每筆 data9 小時內所有 feature 的一次項（含 bias 項）以及每筆 data9 小時內 PM2.5 的一次項（含 bias 項）進行 training，比較並討論兩種模型的 root-mean-square error（根據 kaggle 上的 public/private score）。

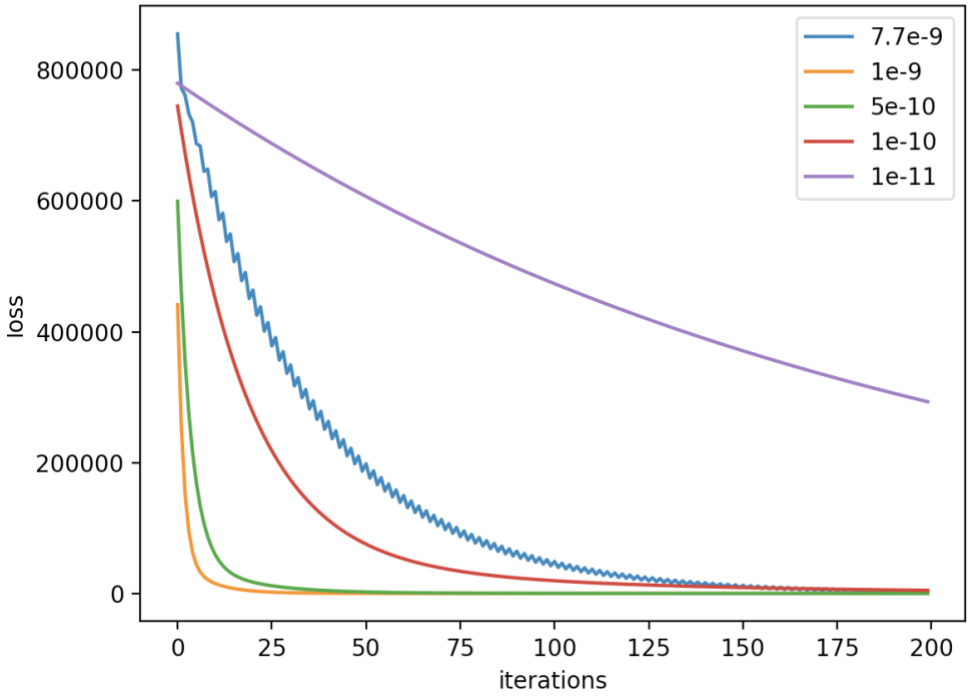
   我以一個月為單位來區分資料，因為需要 9 個小時內的 features，所以一個內可以搜集 471 筆 data。我隨機挑選六個月的資料作為 training data，為了比較兩種模型的好壞，我固定 training data，只用 features 的不同區分兩中模型。在上傳 kaggle 之後，我發現只用 PM2.5 來預測所產生的 error 會比用所有 features 來預測所產生的 error 小一些，就算將 data 數目增加，結果依然如此。最終在 kaggle 上的分數 9 個 features 的拿了 10.86733；而所有 features 的卻是 12.02832。
   雖然在資料量多時，照理來說應該要更準確，但實際非如此。我推測是因為有一些觀測數據與 PM2.5 的數值沒有太大的關係，將他們考慮進去後反而傷害了預測結果。

	Public score	Private score
All features	12.02832	9.52502
PM2.5 only	10.86733	11.06171

   爲了判斷哪一些事跟 PM2.5 有關的資料，我在 excel 做了有關的分析，將在第四點說明。

2. 請分別使用至少四種不同數值的 learning rate 進行 training（其他參數需一致），作圖並且討論其收斂過程。

為了測試 learning rate（以下用 lr 簡寫）對 training 的影響，我選了每筆 data 中 5 個 feature 來 train，再經過多次測試之後我挑選了五個不同的 lr 做比較，結果如下圖：



$> 7.7*10^{-9}$	$7.7*10^{-9}$	10^{-9}	$5*10^{-10}$	10^{-10}	10^{-11}
過大 不收斂	臨界值	最佳解 收斂快	收斂速度稍慢	收斂速度比左邊更慢	太小 效果非常差

從圖中可以看出在 lr = 10^{-9} 時（橘線），其效果最好，參數更新不到 25 次就收斂；隨著 lr 漸漸減少，其所需要的更新次數漸漸增加（綠線、紅線），但是當 lr 太小就會發生更新速度太慢的問題（紫線），可以看到經過 200 次更新，其距離收斂的值還有一段差距。

比較特別的是藍色那條線，他有非常多鋸齒的形狀出現，此時 lr = $7.7*10^{-9}$，比我找出的最佳值大。會出現這樣的情況是因為 lr 一次跳太多，無法有效率的更新，但是以總體效果來看，他仍然收斂，所以也算是一個可行的 lr。

最後我還有測試 lr = 10^{-8}，然而此項數據結果卻無法在圖上呈現（在更新 10 次後 loss 趨近於無限大），因爲這樣的選擇會讓 loss 發散，也就是說 lr 一次更新太多，使得 gradient descent 失去效用。

從以上分析可以看出 lr 的選擇很重要，過猶不及！

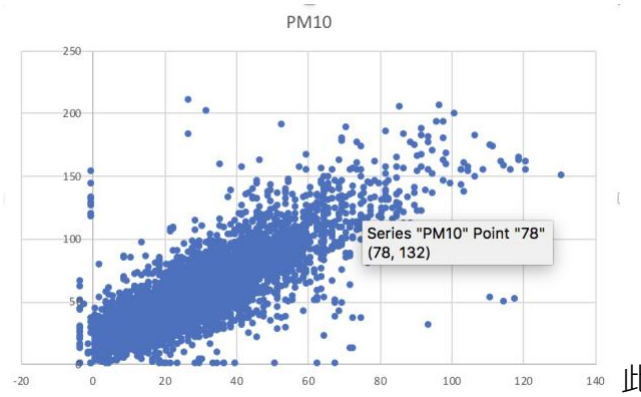
備註：圖中的 loss 為 squared error 的 sum，非平均值

3. 請分別使用至少四種不同數值的 regulization parameter λ 進行 training（其他參數需一至），討論其 root mean-square error（根據 kaggle 上的 public/private score）。

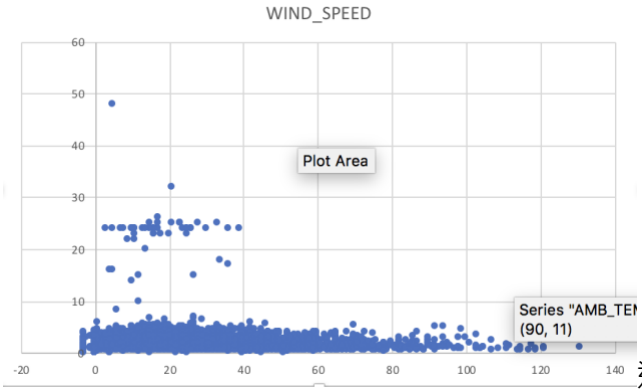
   測試 λ 時，我將整個 model 提高到 2 次，分別測試 λ = 0、λ = 1、λ = 10000、λ = 0.001。由於我選用的 features 非常多，在 λ = 0 時，馬上就看出 overfitting 的狀況，在 kaggle 上的分數 150 多，完全爆掉了。
   但是當把 λ 慢慢調高之後，我發現 error 變小不少，但是其表現卻不如做 linear regression 時來的好，分數最好的時候大約 14 左右。

4. 請這次作業你的 best_hw1.sh 是如何實作的？（e.g. 有無對 Data 做任何 Preprocessing？Features 的選用有無任何考量？訓練相關參數的選用有無任何依據？）

   在經過多次測試以後，我發現即便是把 features 取得非常多，loss 的改變並不大，有時候甚至會愈來愈高，因此，我第一個解決方案是先取一個 feature 就好（PM2.5），並用這個 input train 到二次項。但是經過 cross validation 後其效益也不佳，我換了不同的 feature、重複一樣的動作，不管如何改變我的 model，都不見效。
   想了很久，開始懷疑是不是 training data 有 garbage 在裡面。在我用肉眼看了一下 training data 後，我發現它裡面有很多爛掉的資料（都是 0）！！這很明顯是觀測數據出問題，於是我把這些 0 的資料全部濾掉再重新 train 了一次，效果非常好。
   因此，我重新判斷選用的 features。



此為 PM2.5 對 PM10 作圖所得到的結果，可以看出資料的分佈情形其相關係數偏高



這個則是 Windspeed 對 PM2.5 座圖得到的結果，明顯看出這是個沒用的 feature，相關係數幾乎是零。

在把每個資料都做相同的判斷後，我發現除了 PM2.5 跟 PM10 以外的數據都與我想要的結果沒有很大的關聯性。因此，我最後決定使用 PM2.5、PM10 的一次項做 training，並用 sklearn 來建立 linear regression 的模型。

最後在 kaggle 上的成績是 7.16793，可見 training 前對 data 先做一些分析會對整個結果有很大的幫助！

綜合以上，hw1.sh 以及 best_hw1.sh 都使用了 linear regression，選用的 features 也都相同。最大的差異在於 hw1.sh 中，gradient decent 使用 Adagrad，training 過程中 iteration 自己手動調整；best_hw.sh 則使用了套件（sklearn），比起我直接手刻，正確性提高了一點點、運算速度提高了很多，不過兩者所預測出的結果大致相同。