Homework 1 Report - PM2.5 Prediction

學號：r06921002 系級：電機所碩一 姓名：張哲誠

1. **(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，比較並討論這兩種模型的 root mean-square error（根據 kaggle 上 public/private score）。**

Learning rate	Regulization λ	Type	Final RMSE	RMSE on kaggle
1	10	All features	6.30494271064	8.00134
		PM2.5 only	12.8046236439	12.08580
10	0.5	All features	6.16350689439	7.81784
		PM2.5 only	12.7997912335	11.95372
10	1	All features	6.17355569431	7.82984
		PM2.5 only	12.7998317579	11.96094

以下為在相同 learning rate, regulization λ 的情況下，RMSE 的分佈圖(only for kaggle)：

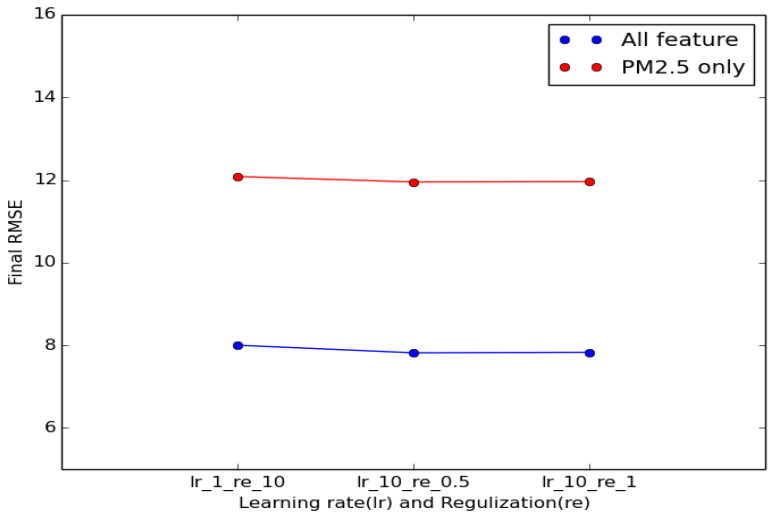

Fig. 1 全部 feature 與 PM2.5 feature 的 kaggle 結果分佈圖

討論：

從在 training data 上最終的 RMSE (Final RMSE 欄位)可以看到，僅用 PM2.5 所得到最後的 RMSE(約位於 12.7~12.9 之間)為用 9 小時後所有 feature 所 training 出的 RMSE 的兩倍(約位於 6.1~6.4 之間)。故從 training data 上來看，我認為 PM2.5 值並非只受前幾個時刻的 PM2.5 影響，而是連其他 feature 如 PM10, NO2, 風向等都要考慮在內，才能有比較好的誤差值。

反觀在 kaggle 上的分數，僅用 PM2.5 所得到的結果跟在 training data 上相差不大，但用所有 feature 的結果卻差至 1 個單位左右。雖然使用所有 feature 的結果還是優於前者，但我覺得可以後續嘗試使用 PM2.5 的二次項來作 training 嘗試，若是在

training data 上能得到跟使用所有 feature 達到一樣的效果，再加上僅使用 PM2.5 在 testing data 上的表現與 training data 也較為接近的話，或許可以得到不錯的結果。

做完這題之後，也讓我重新思考 feature 擷取的方法，我開始使用 training data 上 PM2.5 以外的的 17 種 feature 與 PM2.5 做 correlation，看看彼此資料的相關性，最後決定我第四題的做法。

2. (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。

下表為在六種不同的 learning rate 情況底下，在 training data 所測出的 RMSE 值

Learning rate	Regulization λ	Iteration	Final RMSE	RMSE on kaggle
0.0001	10	20000	12.3773123961	14.59231
0.001			7.79802937966	8.93983
0.01			6.30964794246	8.01134
1			6.30494271064	8.00134
10			6.30512225925	8.00168
100			6.30514048315	8.00171

Table. 1 不同 learning rate 的比較圖表

以下列出在不同 iteration 次數底下，所呈現的 RMSE 收斂圖

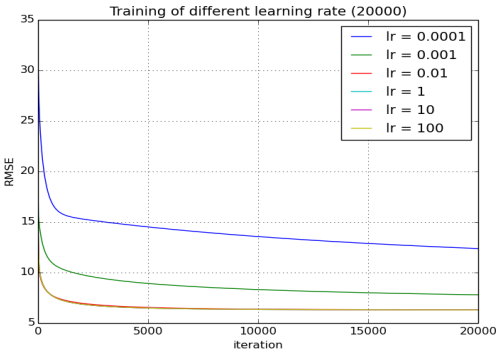

Fig. 2 iteration= 20000

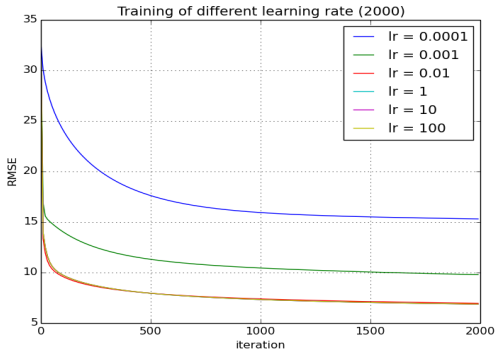

Fig. 3 iteration= 2000

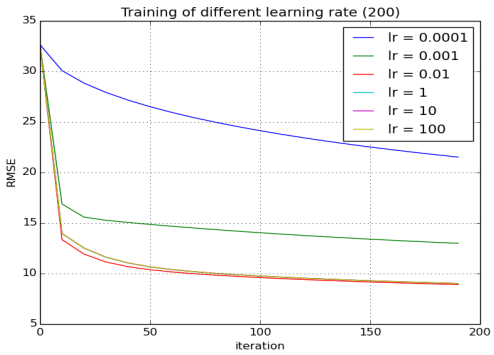

Fig. 4 iteration= 200

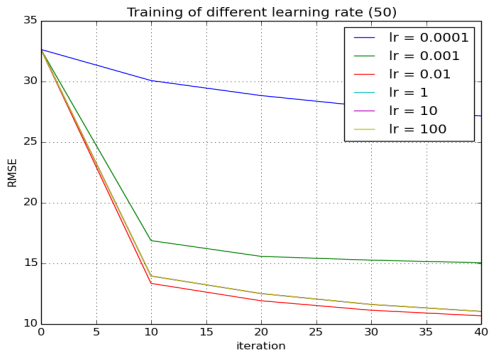

Fig. 5 iteration= 50

以下為在不同的 learning rate 下，最終 RMSE 的變化趨勢圖(包含在 training data 與 testing dara)

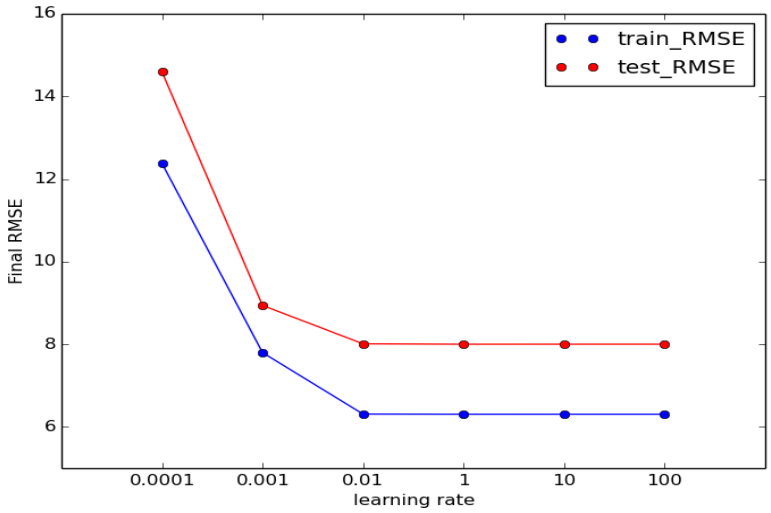

Fig. 6 不同 learning rate 之 RMSE 收斂圖

討論：

從 Table1 可得知，當調整 learning rate 數值到 0.01 以上時，在 training data 上的表現可將 RMSE 收斂至 6.30 左右；而在 kaggle 上得到的成績也都很接近，約為 8.01。這個現象甚至可以從 Fig. 1~Fig. 4 中發現。除了 leanring rate 為 0.0001 和 0.001 以外，其餘四條的下降與收斂速度都是差不多。在 Fig. 1 與 Fig. 2 兩個 iteration 次數比較大的圖來看，甚至分不出四個 learning rate 的差別，直到我將 iteration 的次數縮小到 50 之後，才能略為分辨出四者的不同。

另外，雖然六種不同的 learning rate 數值有不同的下降速度，但從 Fig. 1 中發現，下降的速度雖不同，但整體的 pattern 都是相同的，只差在 learning rate 小的值需要較多的時間，才能真正收斂到 global minimum。

3. **(1%) 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）。**

Learning rate	Regulization λ	Iteration	Final RMSE	RSME on kaggle
10	0.5	20000	6.16350689439	7.81784
	1		6.17355569431	7.82984
	10		6.30512225925	8.00168
	100		7.22624218993	8.99216
	1000		9.45287162168	11.48793

以下列出在不同 iteration 次數底下，所呈現的 RMSE 收斂圖

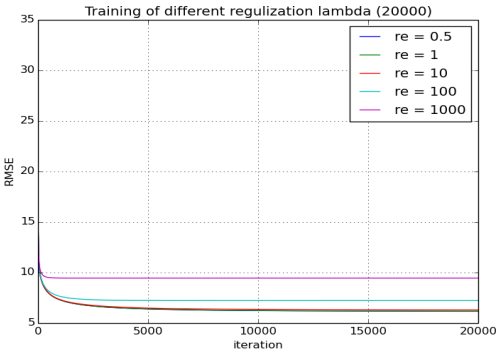

Fig. 7 iteration= 20000

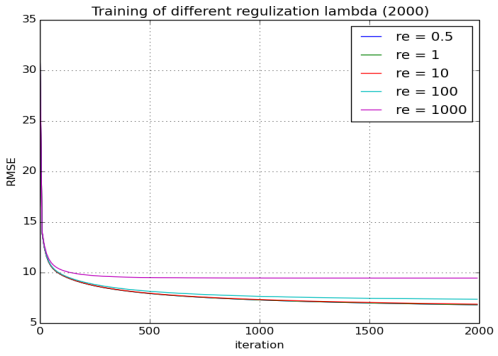

Fig. 8 iteration= 2000

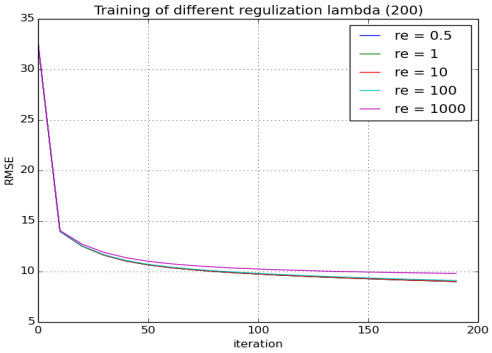

Fig. 9 iteration= 200

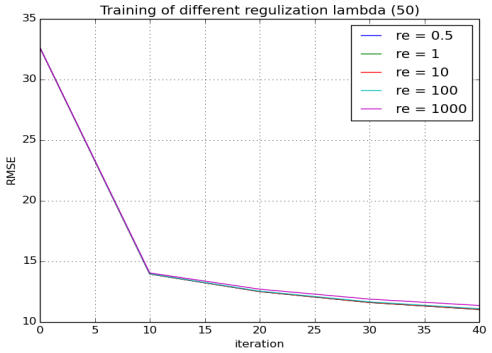

Fig. 10 iteration= 50

以下為在不同 regulation 的 λ，最終 RMSE 的變化趨勢圖(包含在 training data 與 testing dara)

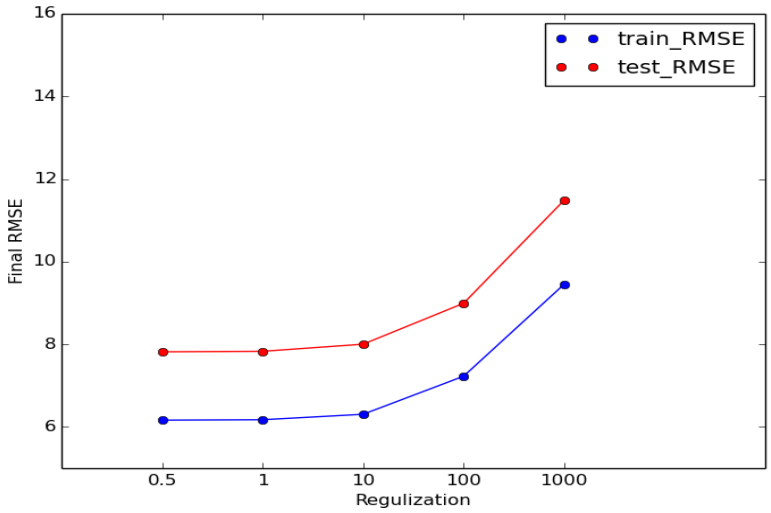

Fig. 11 不同 regulization 值下的最終 RMSE 收斂比較

討論：

以我這次所做的實驗來看，在 learning rate 為 10 的情況下，regulization 的 λ 位於 0.5, 1, 10 這之間對於 testing data 的結果沒有太大的差別，但當 regulization 到 100 甚至是 1000 時，我發現 RMSE 已經慢慢的降不下來。此現象應證上課時老師曾經提過，會有 regulization 是為了 training 的結果更平滑，在 testing 上的表現會較穩定。然而，也就是當 λ 太大時，會導致所 train 出來的結果過度平滑，發生"矯枉過正"的結果，而因為考慮 error 較少，所以才會導致 RMSE 逐漸上升。

另外，從 Fig.5~8 可以看到，iteration 次數越少，不同的 λ 所顯現的差異性就越小。例如 Fig. 8，在只有 iteration=50 的情況下，可以看到五個 λ 沒有對圖形造成很大的改變。然而當時間越長(iteration 次數越高)，彼此的差異性可以從收斂值看出。

總得而言，我認為在 hw1_py 這個檔案中，learning rate 約在 1~10 左右較佳，而搭配 λ = 0.5~1 之間，可以在 training data 和 testing data 都得到較好的結果。

4. (1%) 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？）

(1) Feature: 在 best_hw1 我僅拿 **CO、NO_2、PM10 與 PM2.5** 的一次項都做我的特徵來做訓練。由於在做訓練前，我分別看了每一種 feature 與 PM2.5 的 correlation 值，我發現 PM2.5 與 PM10 的相關性為全部特徵內最高(corr = 0.49)，其餘

CO、NO_2 分別位區其次的值。我認為，若是擷取出相關性較高的特徵會在 training 上有較好的結果，因此選取此四特徵。

(2) Data Preprocessing: 這一次 best_hw1 中，我分別在 training data 與 testing data 上做了錢處理，分別是:

I. Training data: 在每一筆的訓練資料中，若是有小於等於 0 的感測資料，則就刪除這筆 data，因為我認為在 training data 中，可能包含很多這種感測器臨時出問題，而擷取到的錯誤資料，會導致 training 的結果不好。再者，若是有 PM2.5 的值大於 300 的測資，無論是在 X 的部分還是在$\hat{y}$，我也會將這筆 data 給刪除。理由在於我認為這麼大的 PM2.5 值也是錯誤的感測值，會降低 training 結果的正確性

II. Testing data: 在 testing data 中比較有問題的是，有時候感測值會在正常的情況下，突然出現 0 值，例如在 id = 178 的 PM2.5 值，可以看到在最後幾筆資料出現 58-0-69 的值，從直觀的角度我認為這也是資料有誤的部分，因此當遇到這種情況，我會執行以下作業：

A. 當測資為第一筆時，用後一筆來取代。

B. 當測資為最後一筆時，用前一筆來取代。

C. 當測資為非第一筆且非最後一筆時，我會用前後兩項的平均來取代。

以上是我對 training data 與 testing data 所做的前處理，我認為有做前處理可以讓 RMSE 大幅下降，**因此前處理為本次作業的核心所在**

(3) Learning rate 與 regulization 的選擇：在 learning rate 的部分，因為我有做 adagrad，加上前面的分析，只有 learning rate 大於 0.01，基本上結果不會差太多，也不用擔心選太大的 learning rate 會有震盪的問題，因此我在這裡的 **learning rate 選擇為 10**。另外，在上傳多次 kaggle 的結果後，我發現 regulization 的 λ **選為 10** 可以較有效降低 testing 與 training 之間的距離。

以下是我在 hw1_best.py 執行出在 training error 的變化趨勢圖，而最終的 error 為 5.26，在 kaggle 上的成績為 7.39200。

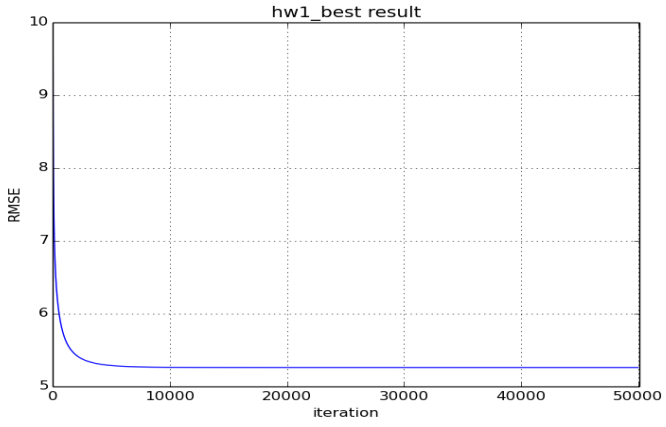

Fig. 12 Final result of hw1_best.py in training