Homework 1 Report - PM2.5 Prediction

學號：b05902008　系級：資工二　姓名：王行健

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上的public/private score) 。**

   只將資料接起來，沒有額外預處理

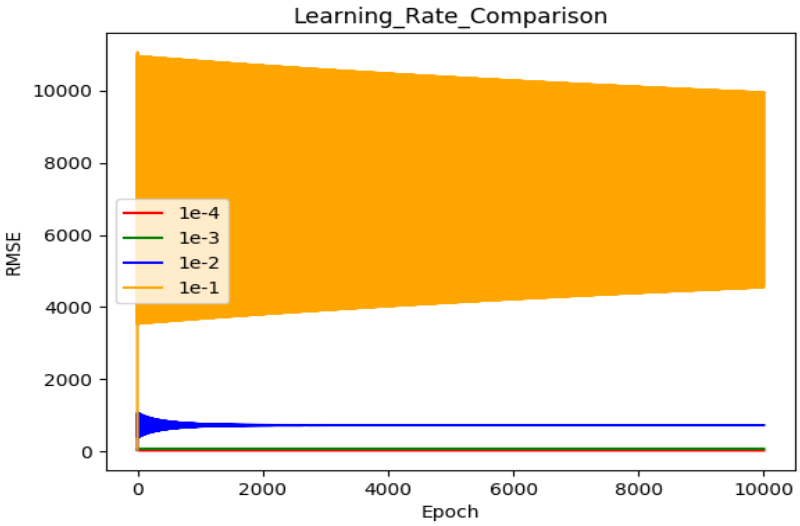
   PM2.5 : trainloss[23.19370]　public[9.54174]　private[9.67831]

   ALL　: trainloss[22.50634]　public[9.36233]　private[9.03040]

   從以上的數據可以發現幾件事。首先，trainloss已經收斂卻還是遠大於private/public，這意味著traindata絕對含有雜訊，而且要不是偏差嚴重就是數量龐大。再進一步比較，不難看出PM2.5的loss略高於ALL。這可以從不只一個方面解讀，第一種解讀方法是ALL含有一些額外的資訊，因此可以稍微達到比較好的效果；第二種則是ALL雖然不包含額外訊息，但因為造稱誤差的雜訊大部分都在PM2.5中，增加變數數量剛好減弱了雜訊的影響。至於實際狀況如何，還需要進一步實驗。

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。**

   只用接起來的PM2.5，沒有額外預處理，w初始為零向量



   在learning rate<=1e-3的時候，都快速收斂。這顯示w*其實相當接近零向量。而learning rate達到1e-2時，loss開始出現小波動，在1e-1更是大幅度的波動則是因為learning rate太大使w有機會跳出loss的local minima。然而即使在1e-1時，loss仍然逐漸收斂，而沒有趨於發散。若在實驗更大的learning rate，或許有機會看到發散的過程。

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) 。**

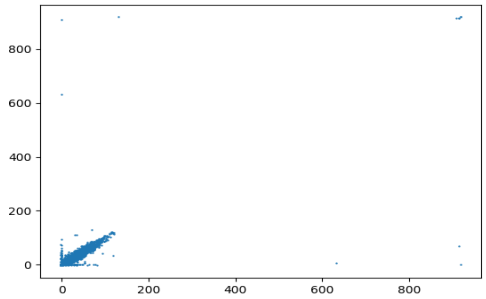
只用接起來的PM2.5，沒有額外預處理，w初始為零向量

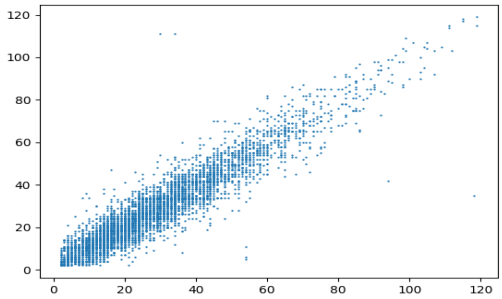
lambda=1 : trainloss[23.19370] public[9.54175] private[9.67831]
lambda=1e3 : trainloss[23.21121] public[9.57786] private[9.69949]
lambda=1e6 : trainloss[23.81822] public[10.50656] private[10.58211]
lambda=1e10 : trainloss[43.59710] public[39.29938] private[38.72167]

正如上面所提到的，檢視沒有L2regularlization時的w，可以發現其長度本來就相當短(所有維度絕對值皆<2)。再加上model相當簡單，因此regularlization防止overfit的效果並不會太好。實際實驗的結果也顯示，由於w長度很短，在lambda<=1e6之內，loss都沒有顯著影響(也沒有overfit)。而當lambda=1e10時，則由於過度限制model，造成相當嚴重的underfit。

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?)**

由上面幾個實驗的結果可以簡單的看出首要任務就是清理dataset。由於個人傾向第二種假設(PM2.5包含了大部分必要資料)，引此一開始便先單獨抽出PM2.5並已連續兩時段的值作圖。





從圖中可以發現 <2 和 >120 的時候值並不像其他部分一樣有明顯正相關。因此，大膽的先切兩刀，把包含這兩個職以外的traindata都刪掉。做完這一步public loss大約可降到 7~8。

接下來檢視testdata，發現裡面也有極端值，但礙於不能直接刪除，只能以其他數值替代。我選擇的方式是以後一時段的資料取代，而如果最後一個值是0，則假設最後三個時段的值都是壞掉的，因此已倒數四筆資料替代。如此做完，public loss可以再降到 6.5 左右。

接下來，由於相信越偏向極端的數值越少出現，再將testdata裡面>=110的數值都替換成110，<=2的數值都替換成2。這是一步保險性的動作，由於不知道實際train完狀況會如何，保守式的削到較大或較小的資料應該可以促使總錯誤降低。到這裡public loss已經降到6.1。

最後，由於traindata原本就不多，因此作了一個決定性的修正。放棄資料量轉而追求資料相似性。計算每筆traindata平均並切成12分，切分方法及數量是以多次的cross validation決定的。切完之後各組分別train，且每筆testdata各自依其平均分組已進行預測。 public loss再次下降到5.8。
此時有一個較為麻煩的問題，就是validation loss和public loss之間存在差距，而public loss明顯較低。這是我懷疑public loss overfit，因此，我再檢視各組的validation loss並發現其中有兩組(average:50~53,55~57)特別高，經過幾次實驗，我又發現以average-1取代50~53，prediction*2-average取代55~57其validation loss結果似乎最好，因次就做了這個嘗試，結果使 public loss 和 validation loss皆落在 5.8~6。於是就以此為最終版本上傳。