學號：r07921001　　系級：電機所碩二　姓名：李尚倫

備註：

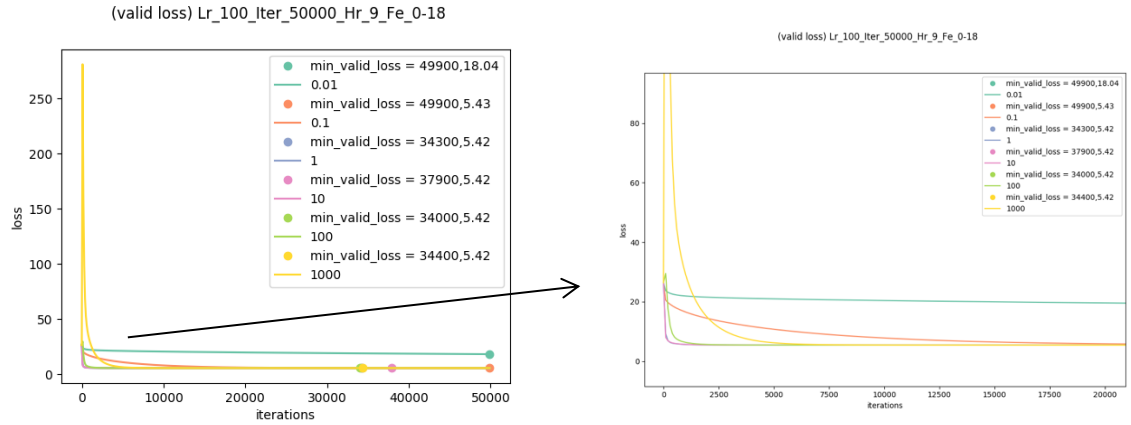
a. 1~3 題的回答中，NR 請皆設為 0，其他的數值不要做任何更動。

b. 可以使用所有 advanced 的 gradient descent 技術（如 Adam、Adagrad）。

c. 1~3 題請用 linear regression 的方法進行討論作答。

1. (2%) 使用四種不同的 learning rate 進行 training (其他參數需一致)，作圖並討論其收斂過程（橫軸為 iteration 次數，縱軸為 loss 的大小，四種 learning rate 的收斂線請以不同顏色呈現在一張圖裡做比較）。

如下圖是比較不同 learning rate 在前 80%的 data 上 train 後 20%的 data 上 validate 的收斂結果(即在多少 iteration 會出現 loss 最小)，min_valid_loss= iteration, loss：



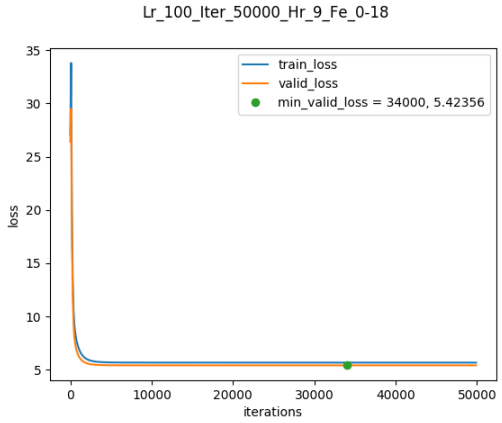
由上圖觀察可知，太大的 learning rate 會導致一個 step 太大，在趨向 local minimal 時可能會超過，導致一開始的 loss 極大，然後才收斂，而太小的 learning rate 則會下降得太慢，雖然也有在朝 local minimal 走，但收斂的 iteration 數要非常多。因此適當的的 learning rate 可以幫助收斂的比較快，同時也能獲得更小的 loss 值(些微差別，見最後面的第一題補充)，而此次比較中 learning rate = 100 在 baseline model 中的效果最好。

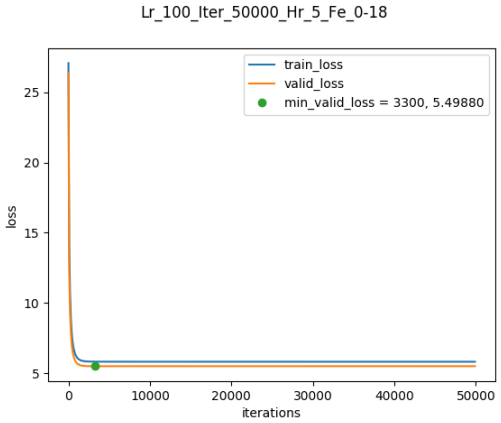
2. (1%) 比較取前 5 hrs 和前 9 hrs 的資料（5*18 + 1 v.s 9*18 + 1）在 validation set 上預測的結果，並說明造成的可能原因（1. 因為 testing set 預測結果要上傳 Kaggle 後才能得知，所以在報告中並不要求同學們呈現 testing set 的結果，至於什麼是 validation set 請參考：https://youtu.be/D_S6y0Jm6dQ?t=1949 2. 9hr:取前 9 小時預測第 10 小時的 PM2.5；5hr:在前面的那些 features 中，以 5~9hr 預測第 10 小時的 PM2.5。這樣兩者在相同的 validation set 比例下，會有一樣筆數的資料）。

如下圖是比較只取前 5 hrs 的資料(右)和取所有前 9 hrs 的資料 (左)在前 80%的 data 上 train 後 20%的 data 上 validate 的結果：

只取前 5 小時的 validation loss 明顯不比取前 9 小時的資料來的低，5.49880 > 5.42356，不僅如此，如果今天題目改為用前 19 小時的資料來預測第二十小時的結果的話，validation loss 則會大幅下降到 5.2 多，這是因為在不改動 feature 和一次 model 的前提下，5 小時和 9 小時的 model 分別為 90 元一次式和 162 元一次式，包含 constant 項在內，有 91 個和 162 個 weight 可以做 regression，其實就代表了這個

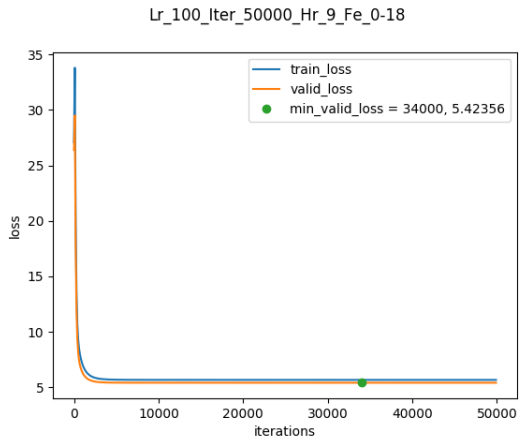
function 的複雜程度，而如果能用 19 小時，就會有 342 個 weight，會更複雜，而對大部分的問題來說，越複雜的 model 越有可能 fit 所有的 data，得出比較小的 loss 值，當然也有可能因為太複雜而 overfit。但前 9 小時的 model 比前 5 小時的 model 確實多了許多有用的資訊，並不是增加不相關的參數，因此較複雜而較趨近真實的函式，且無 overfitting 的現象，故 baseline model 中仍採用取前 9 小時的 data 來做預測的方式。

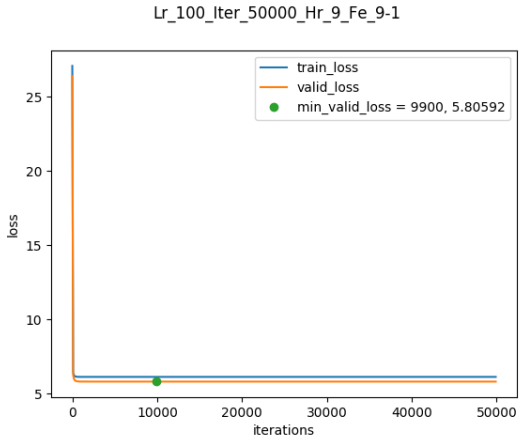




3. (1%) 比較只取前 9 hrs 的 PM2.5 和取所有前 9 hrs 的 features（9*1 + 1 vs. 9*18 + 1）在 validation set 上預測的結果，並說明造成的可能原因。

如下圖是比較只取前 9 hrs 的 PM2.5 (右)和取所有前 9 hrs 的 features(左)在前 80%的 data 上 train 後 20%的 data 上 validate 的結果:

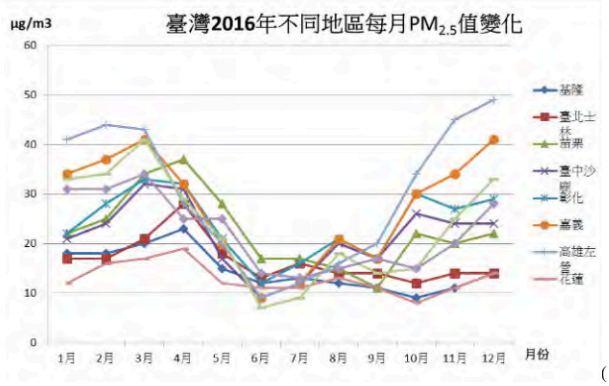




如 2.裡所討論的，拿到 feature 直接的影響也是 model 變簡單了，只有取 pm2.5 的方式會只剩 10 個 weight，變成比第二題取前五小時還要更簡單的 model，因此無法更好的 fit 真實的函數，導致 loss 比原本的還要大很多。但其實也不是說拿掉 feature 減少維度就一定會變差，像是只拿掉 rainfall 的話 loss 就會變小，因此必須在降低 model 複雜度和濾掉不相關的 feature 之間做小心的取捨，做了幾次嘗試，目前 dataset 中的 feature 都多少能對預測做出一點幫助，並不真的有嚴重沒關係(來搞破壞)的 feature，除了 rainfall 和 wind direction 之外，而這兩個 feature 的調整會在下一題中再做說明。

4. (2%) 請說明你超越 baseline 的 model(最後選擇在 Kaggle 上提交的) 是如何實作的（例如：怎麼進行 feature selection, 有沒有做 pre-processing、learning rate 的調整、advanced gradient descent 技術、不同的 model 等等）。

- 1. Feature selection:
  本次 dataset 共有 18 個 feature 包含 pm2.5 本身，網路上可以查到許多論文或研究報告指出相關指數對空氣中成分的影響，我參考了幾篇做了相關的調整，把沒相關的拿掉，基本上成果都不會獲得更好的改善，feature 還是越多，使 model 越複雜，預測出來的效果比較好。但其中 rainfall 的參數因為大部分都是 NR(=0)，僅少部分有值，讓訓練資料太不平衡，即使許多論文指出 pm2.5 和降雨有關，還是應該講此項拿除，可獲得些許改善，另外就是是風向，dataset 中表示得為角度，並不具備直接的物理意義，所以應轉為 $\sin\theta^2$和 $\cos\theta^2$ 使其代表為 x 方向的風量大小和 y 方向的風量大小，較具物理意義，可獲得些許改善。因此最後 strong baseline 的 model 即是拿掉 rainfall 和 wind direct 加入$\sin\theta^2$和$\cos\theta^2$的結果，總計 feature 數還是 18 個。時間的取用則同 2. 的結論，仍取前 9 小時。
- 2. Pre-processing:
  由於這次的 linear regression 除了有做 normalization，我還加入了 filter 去濾掉大於特定標準差的值，高過 2 個標準差者則 = 兩個標準差，來避免過大的值出現，結果有顯著的改善，由此可推斷，這個 dataset 中可能存在著不少 noise 在影響我們的結果。而最後 strong baseline 的結果即為基於此現象上再做調整，針對正的超出太多和負的超出太多，做不同的 threshold 限制，filter 出結果。
- 3. Learning rate:
  Learing rate 的調整如同前面 1. 的結論，100 的收斂效果仍然不錯，但以最終的結果來說 200 的結果有些許的改善，最終 model 是使用 200
- 4. Initial weight:
  原設計是 initial weight 都給 0，但根據知識，這個 model 預測第十小時的 pm2.5 值必定跟前面幾小時的 pm2.5 很有關，因此我們把過去九小時的 pm2.5 weight 設為 1，其他保持為 0，然後再做 gradient decent，結果確實在下表(iteration)的實驗中原本其實沒在 50000 內收斂的都收斂了，確實能幫助結果比原先的還快收斂，但最後拿去實測，並沒有比 initial weight 是 0 的來得好。
- 5. Iteration:
  為了避免 overfitting 我們的 training data，我透過不同的切法把 training dataset 切成 20% validation, 80%training 的方式做 Cross Validation，但由於 12 個月份的空氣情況不同，如下圖，因此做出來的 validation loss 也差距很大。



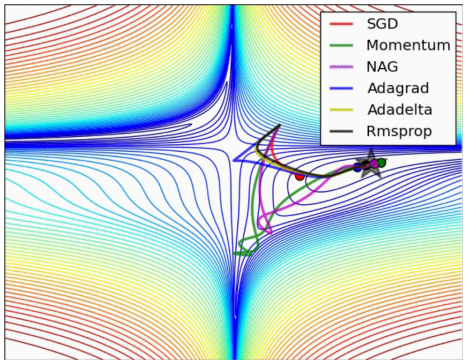
(https://activity.ntsec.gov.tw/activity/race-1/57/pdf/030505.pdf)

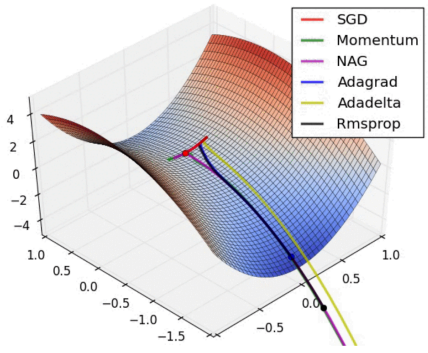
因此除了做了五個區段的 80%20%拆分，還做了對 12 個月每個月取 20 筆出來當 validation

的狀況，如下表，記錄了跑 50000 個 iteration 時 minimum validation loss 發生的位置和大小，而最後 strong baseline 的 model 即是取全部 training data 下去 train 了 25100 個 iteration 的結果，於 kaggle public 測資上的 score 為 5.36054，leaderboard(57/312)

Valid/Train separation	Iteration	Training loss	Validation loss	Testing score
0.0-0.2 / rest	5900	5.5631	6.3107	-
0.2-0.4 / rest	13200	5.5326	6.4194	-
0.4-0.6 / rest	50000	5.4808	6.5356	-
0.6-0.8 / rest	29200	5.9180	4.7081	-
0.8-1.0 / rest	10800	5.7344	5.5596	-
Every month 20 / rest	26100	5.6519	6.0967	-
Final result (init 1)	26100 (set as above)	5.6684	-	5.38883
Final result (init 0)	25100 (set as above)	5.6714	-	5.36054

- 6. Future work:
  現有的結果還有許多可以做的空間，例如追加關聯性高的 data 的次方項(pm2.5 的次方或三方)、交叉項(pm2.5 和 O3, SO2 等相乘)，來增加 model 的複雜程度，可能有機會能更好的 fit 我們想求得 function。或是更換 gradient decent 的演算法，如下圖，Adadelta 的收斂效果平均來說應該都會比 Adagrad 來得好，有可能能增加收斂的速度，用比較少的 iteration 達到結果。例如更換 model 從一次至二次或三次等可能有機會變好，更 fit 想求得的 function。





(https://twitter.com/alecrad)

*第一題補充：

#0.01 iters: 49900 , train_loss: 18.729618155494936, valid_loss: 18.043221717255232 min!
#0.1 iters: 49900 , train_loss: 5.685879811554643, valid_loss: 5.432788535479192 min!
#1 iters: 34300 , train_loss: 5.679868125269123, valid_loss: 5.4237776520366126 min!
#10 iters: 37900 , train_loss: 5.679887534278326, valid_loss: 5.423738930812674 min!
#100 iters: 34000 , train_loss: 5.6800454206538085, valid_loss: 5.42356064369309 min!
#1000 iters: 34400 , train_loss: 5.6809708945496675, valid_loss: 5.42247826974859 min!