Machine Learning HW#1 Report

Linear regression 電機四 b02901126 張大方

1&2. **Linear regression function by Gradient descent**

```
for i in range(0, 300000, 1):
    Xw = np.dot(train, w_wgt)
    gradient = np.dot(np.transpose(train), Xw - np.transpose(y_ans))
    gradient_n = np.dot(2*Eta[comp]/5652, gradient)
    w_wgt = w_wgt - gradient_n - np.dot( w_wgt, 2*lamda[comp]*Eta[comp])
```

參數解釋:

i. *train*: 就是 Lost function 中的 X，包含了我所有的 training data。

ii. *w_wgt*: weight，是一個(64*1)的矩陣，包含 9hr*7paremeters+1bias。

iii. *y_ans*: 從 training data 中取得的答案，扣除掉每月的第一天前九小時得到的資料。

iv. *Eta[]*: learning rate，為了比較不同 learning rate 的影響，所以用了一個 array 來存

v. L*amda[]*: 同 Eta[]，importance of regularization

作法解釋:

我將每一個月一號的 0:00 到二十號的 14:00 當作每筆資料的起始小時，再將每一個月連接起來。因此我的 training data 共有$(24 * 20 - 9) * 12 = 5652$ 筆資料。其中每筆資料包含了九個小時，每小時我抽取出七種 feature 來訓練，最後再加上 bias。所以我的 training data 是一個(5652*64)的矩陣。

在 Gradient descent 的過程中，由於我的 training data 是一個大矩陣，所以在計算時我用到了矩陣運算的技巧來處理 gradient，並非直接將 Loss function 作微分，其過程如下:

$$\mathrm{L} = \sum (\hat{y} - X \cdot w)^2 = \|(\hat{y} - X \cdot w)\|^2$$

$$= (\hat{y} - X \cdot w)^T \cdot (\hat{y} - \mathrm{X} \cdot w)$$

$$= \hat{y}^T \cdot \hat{y} - 2(Xw)^T \cdot \hat{y} + (Xw)^T \cdot (Xw)$$

$$\nabla \mathrm{L} = -2X^T \cdot \hat{y} + 2X^T(Xw)$$

$$= 2X^T(Xw - \hat{y})$$

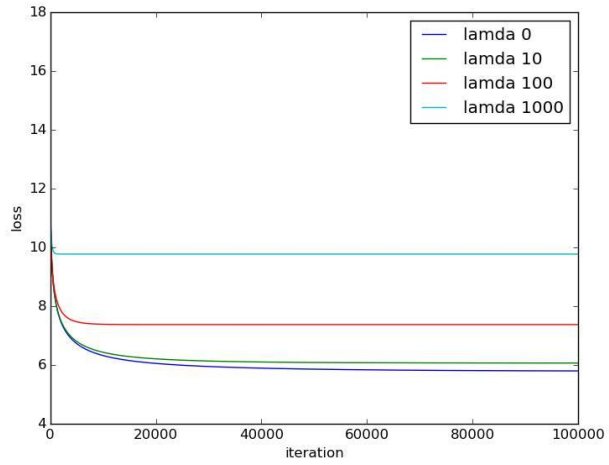
$\nabla \mathrm{L}$ 就是這 5652 筆資料的 gradient，因此在計算新的 weight 時需要將這個 gradient 做平均，乘上 Learning rate 之後便能與 weight 相減得到新的 weight 了。後面 Lamda 項是 regularization 的參數。

*** 所選取的七個參數，分別為:*NO NO2 NOx O3 PM10 PM2.5 SO2*

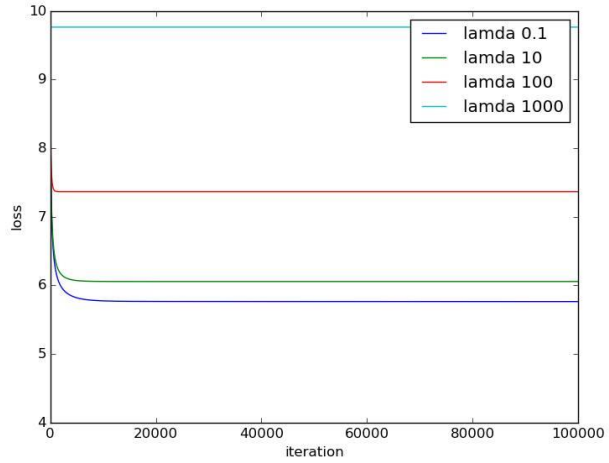
根據 https://goo.gl/fi4qDU ，氮硫氧化物與臭氧跟 PM2.5 相關性高

但是無法考慮到冬天東北季風帶來的影響

3. **Regularization**

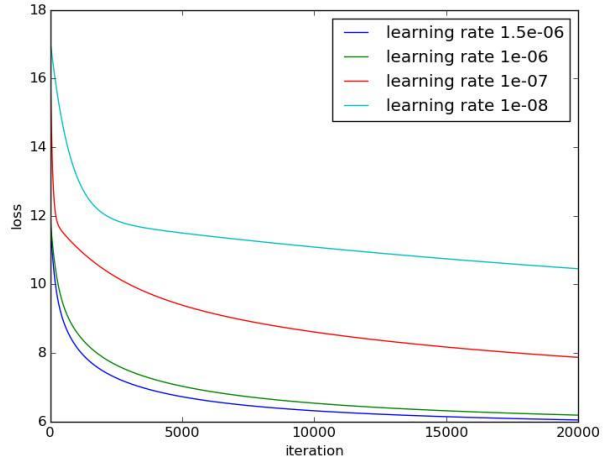


由圖可知，加上一般的 regularization 方法後，並不能使的訓練結果更好。由於這張圖並沒有實作 Adagrad 的方法，可以看出 regularize 在初期迅速降到一個值之後，中後期就收斂的非常緩慢，無法更靠近收斂的解，降低 Loss。



這張圖有利用 Adagrad 的方法，動態調整 learning rate。與上圖比較，可以看出 Loss 有比較小，更早趨近於 local optimum 的值。

4. **Learning rate**



由圖可以明顯看出，較小的 rate 趨近於最小值的速度也比較慢，其中最下面那條線就是我所選擇的 rate 所跑出的圖形，其 Loss 下降較快速。若選擇更大的 learning rate 的話，其結果無法呈現在圖表中，train 到一半就會發現值 nan 掉了。