Machine Learning HW1 Report

B02901031 陳緯哲

I. Gradient Descent Code

```
while (count < 300000):
    y = np.dot(X ,w)
    a = np.dot(X.transpose(),np.subtract(y,y_head))   # a = X · w − ŷ
    deltaF = np.multiply(2.0, a.astype(np.float))/5751   # deltaF = gradient
    new_w = np.subtract(w,np.multiply(learning_rate,deltaF)) # update weight
    w = new_w
    count += 1
```

II. Method

How I choose training batch?

首先，我取 data 的方式是取所有的 data，因此一筆 data 會有 18*9 = 162 個數據。而我取的 batch 是取所有的 data ，匯集起來就是一個 5751x162 的矩陣，再加上 bias，就是 5751x163 的矩陣。這樣的方法雖然慢，但是較穩定，也較準。

How do I calculate gradient?

為了計算更快速，我將微分直接用在矩陣上，經過一連串的運算，可以得到 loss gradient 就是 $2X^{\mathrm{T}}(X \cdot \mathrm{w} - \hat{y})$，再將此 gradient 乘上 learning rate 來得到新的 weight。

其中，X 是我的 5751x163 data matrix，w 是我的 weight ，$\hat{y}$ 是 data 的答案。

How do I choose epoch number?

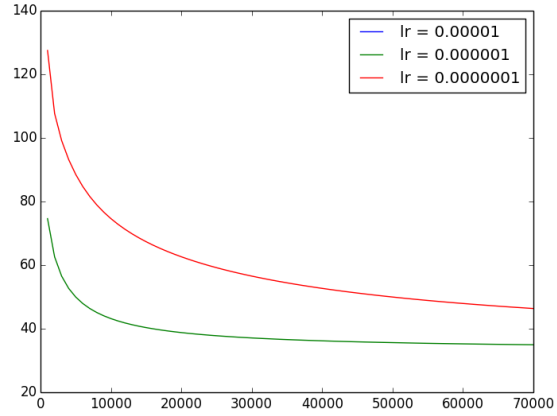
在測試了 10000,50000,100000,300000,1000000 的 epoch 並上傳到 kaggle，發現 300000 epoch 的結果會是最好的，而且 error 似乎也呈現患慢下降的趨勢，推測他的 loss function 是一個平坦的高原。但是 1000000 似乎出現了 overfitting，所以選 300000 epoch, learning rate = 0.0000015 作為 kaggle best。

III. Discussion on regulation

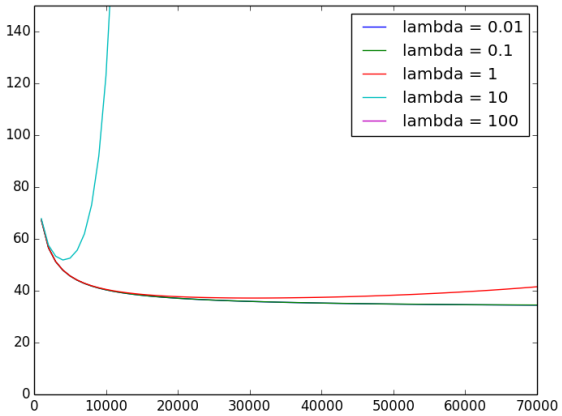
在 apply regulation 後，似乎沒有改善準確度，從 0.01,.0.1,1,10,100 都沒有看到起色。

IV. Discussion on learning rate

最佳的 learning rate 似乎在 0.000001~0.0000016 之間，大於這個數字，error 就會跳太快，而小於這個數字，approach 的太慢。



圖（一）是不同 learning rate 收斂的比較。藍色的線沒有出現是因為 error 太大已經超出範圍。橫軸為 epoch 數，縱軸為 error



圖（二）是不同 lambda 對照 error 的結果。圖中沒有顯示的線則為數據太大超出範圍。橫軸為 epoch 數，縱軸為 error