作業 1 報告，R04922081 江宗臻

Linear Regression 和 Gradient Descent

假設有 n 筆資料，有 m 筆 features，則第 i 筆資料的回歸式如下：

$$y_i = w_0 + w_1 x_{i,1} + w_2 x_{i,2} + \cdots + w_n x_{i,m}$$

我用 MSE (Mean Squared Error) 和 L2-regularization 計算 error：

$$e = \frac{1}{n}\sum_{i=1}^{n}(l_i - y_i)^2 + \alpha \sum_{j=1}^{m} {w_j}^2$$

把 e 對每個參數 w_j 偏微分，即得 w_j 的更新法則：

$$\frac{\partial e}{\partial w_j} = \frac{2}{n}\sum_{i=1}^{n}(y_i - l_i)x_{i,j} + 2\alpha w_j$$

$$w_j^{t+1} = w_j^t - \frac{\eta^t}{\sigma^t}\frac{\partial e}{\partial w_j}$$

其中 η 是 learning rate，除以 σ 則是 adaptive gradient descent。

方法

機器學習重點是選好的 model，例如要用 linear regression 或 SVM 或 neural network? 要選擇哪些 features? 如何設定 regularization 參數? 在不要 overfitting 前提下，選一個 model 讓 error function 值為最小。

也就是，我排列組合過不同方法，看看 error 有沒有變小。例如我試著刪除 p-value 大的 features，若 error 變小，之後就也刪除它們；我也試過 neural network（自己實作！），可惜 error 變大，也就沒保留了。

但 error 變小可能是 overfitting 所造成。為減緩 overfitting，我把 training 資料進一步切成 training 和 validation。我改變 model 時，利用 n-fold 方法觀察 training 和 validation 的 error 變化。詳細流程如下：

1. 修改方法（例如改成 neural network、刪除 feature）
2. 使用 n-fold 交叉驗證：
    a. 把資料 shuffle，分成 training 和 validation
    b. 計算 training 和 validation 的 error
    c. 回到 a 再做一次，共重複 n 次
    d. 把 n 次的 error 做平均

3. 如果平均的 training 和 validation error 同時下降，則保留此方法；反之若 training error 下降，但 validation error 上升，則很有可能 overfitting，因此就不保留此方法。

Regularization

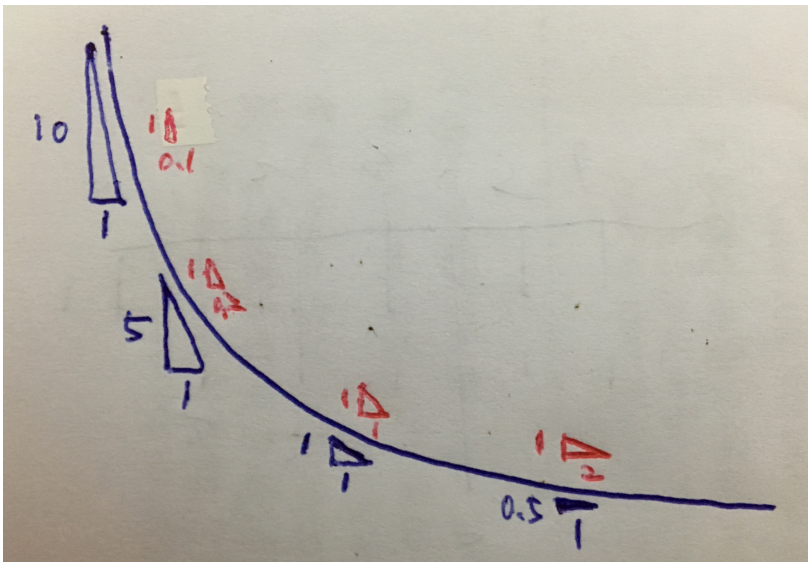
不論 linear regression 或是 neural network 我都是使用 L2-regularization，最前面的 error 公式後面那項即是。

因為 neural network 因為能 fit 任何函式，故極容易 overfitting，我嘗試 regularization 針對 NN 效果不錯，可惜 NN 似乎不是用這次作業。然而 linear regression 我發現效果並不明顯，當 $\alpha > 1.0$ 訓練和驗證 error 都很大，設得很小則和沒有差不多，因此 linear regression 我後乾脆不用 regularization 或是設 $\alpha = 0.01$ 意思一下。

但 regularization 另外的好處是，收斂較快！因為參數（權重）被縮在比較小的範圍。

Learning Rate

我用兩種 adaptive gradient descent 來調整 learning rate，如下圖藍色和紅色。



藍色是固定的 step size，紅色是固定的 error 下降，一般的 AdaGrad 比較接近藍色的。紅色的方法概念是，在比較陡的坡，走小步一點，在平滑地方則走比較快。我嘗試過兩者速度沒差太多，但藍色的後期會比較穩定！