# Web Retrieval and Mining Programming HW2

B02902029 楊子由

## 簡介

實作 Learning To Rank 各個演算法，效能評估是NDCG：

$$DCG = \sum_{i=1}^{10} \frac{2^r - 1}{\log_2(i+1)}$$

## Q1(Task 1)

Task 1使用 Gradient Descend，要先求出 Loss Function 的偏微分：

$$\begin{aligned}
\frac{\partial L}{\partial w} &= \sum_{r(i)>r(j)} \frac{\partial \log(1 + e^{\sigma(x_j^T w) - \sigma(x_i^T w)})}{\partial w} \\
&= \sum_{r(i)>r(j)} \left(\frac{e^{\sigma(x_j^T w) - \sigma(x_i^T w)}}{1 + e^{\sigma(x_j^T w) - \sigma(x_i^T w)}}\right)\left(\frac{\partial \sigma(x_j^T w)}{\partial w} - \frac{\partial \sigma(x_i^T w)}{\partial w}\right) \\
&= \sum a_{ji}(x_j^T \sigma'(x_j^T w) - x_i^T \sigma'(x_i^T w))
\end{aligned}$$

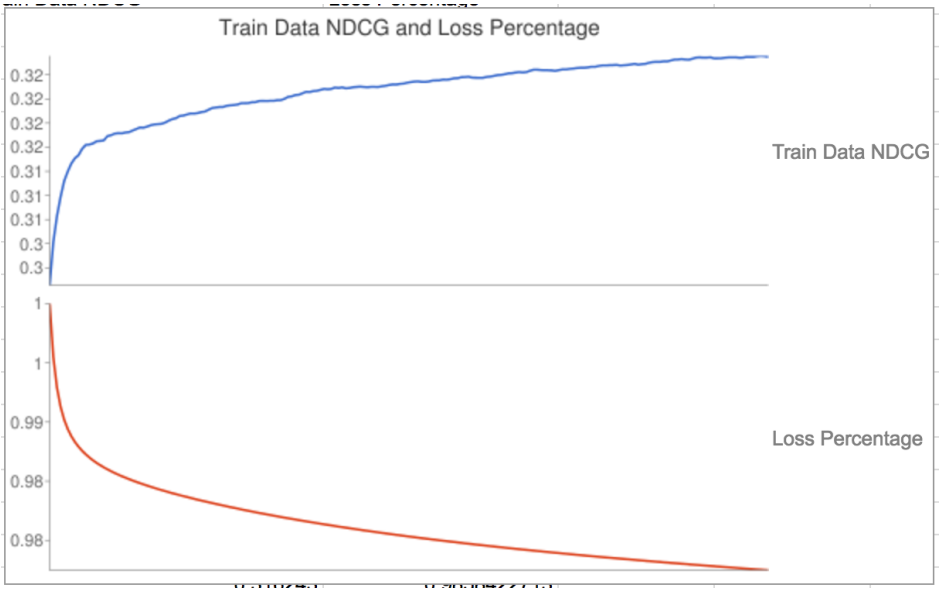
其中

$$a_{ji} = \frac{e^{\sigma(x_j^T w) - \sigma(x_i^T w)}}{1 + e^{\sigma(x_j^T w) - \sigma(x_i^T w)}}$$

$$\sigma'(x) = \frac{e^{-x}}{(1 + e^{-x})^2}$$

實作上， (i, j) 隨機取 |D| 個，留下 r(i) > r(j) 的 pair。
我有使用 Stochastic Gradient Descent ，切塊，Regularize。

| Train Data NDCG : 0.324 | Test Data NDCG : 0.341 |
|---|---|




# Q2(Task 2)

Task 2 的 Loss Function 比較特別，是有 Closed-Form 解的。

$$w = \arg\min_{w'} \sum (y - w'^T x)^2 = (X^T X + \lambda I)^{-1} X^T Y$$

偏微分：

$$\begin{aligned} \frac{\partial L}{\partial x} &= 2\sum -x^T(y - x^T w) \\ &= (-2)\sum x^T y + 2\sum x^T x w \\ &= 2X^T X w - 2X^T Y \end{aligned}$$

其中

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} \quad Y = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{bmatrix}$$

λ是一個很小的數字，用來避免不可逆矩陣。

我用了 Close-form Solution 和 Gradient Descend ，並且進行比較。

## Close-form

這並不需要任何 Iteration ，直接把式子帶進去，得到：

| | |
|---|---|
| Train Data NDCG : 0.37 | Test Data NDCG : 0.38 |

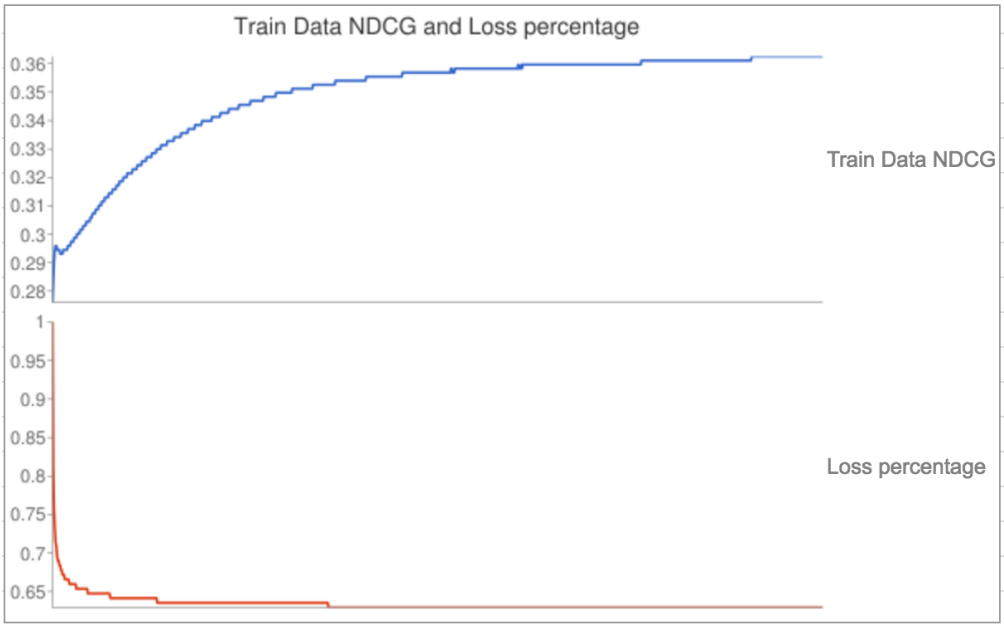
## Gradient Descend

和 Task1 比起來，Task2 的 iteration 可以更多，因為每次迭代純粹是137x137的矩陣運算，不過也需要注意到數字大小不控制好很容易溢位，所以我在實作上有追加 Regularization。

Pass Test Baseline 的設置：learning_rate = $1/(10^4|Q|)$、lambda = $1/10^6$、iteration = $10^5$。

| | |
|---|---|
| Train Data NDCG : 0.366 | Test Data NDCG : 0.386 |

把一開始的 Loss 當作 100% ，每100個 iteration 紀錄一次 Loss 和 NDCG 的結果圖：




# Q3(Task 3)

在 Task3 ，我沿用 Task2 的 Loss function，只是把 Loss function 改成：

$$L = \sum_{query} \frac{1}{|D|} \sum_{d \in D} (1.5^y - wx)^2$$

由於可以改寫成：

$$L = \sum_{query} \sum_{d \in D} (\frac{1.5^y}{\sqrt{|D|}} - w\frac{x}{\sqrt{|D|}})^2$$

所以仍然可以直接用矩陣運算算出來。

| Train Data NDCG : 0.380 | Test Data NDCG : 0.394 |
|---|---|

# Q4(diff Task2 Task3)

在 Task3 裡面，我把 relevance 變成 指數級成長的，因為 DCG 也是用指數來算。而把每個 Query 平均起來則是為了避免過多的 Document 影響到 Loss 。

無論是 Train Data 還是 Test Data ， Task3 的結果都比 Task2 高。

# 參考

- Python Package : numpy, multiprocessing, pickle