# 機器學習(HW2)　姓名：袁培傑　學號：B03901134

## Logistic regression function by Gradient Descent

```
lr = 0.01
iter = 10001
G = np.zeros(57)
B = 0
for times in range(iter) :
    z = np.dot(mat, par) + bias
    sig = 1 / (1 + np.exp(-z))
    cross = -(    ans   * np.log(np.maximum(sig, 1e-15)) +
              (1-ans) * np.log(np.maximum(1-sig, 1e-15)) )

    g = -1 * np.dot( (ans - sig), mat)
    G = G + np.square(g)
    b = -1 * 2 * np.sum(ans - sig)
    B = B + b * b

    par = par - lr * g / np.sqrt(G)
    bias = bias - lr * b / np.sqrt(B)

    loss = np.sum(cross)
    print(loss)
```

Logistic regression

Gradient decent using adagrad

## Second Method: Deep Neural Network

### (1) Forward Pass

以我表現比較好的 Model 為例: input = 57, # of layer = 2, nodes = 40, output = 1

Random initialize 兩個 weight matrix，分別為 57*40、40*1，Data 有 57-dimension，將 57 個值當作 input，進行第一層的 logistic regression，得出 40 個 nodes 的 z 值，將 40 個 nodes 帶到 sigmoid function，以此作為第二層的 input，再進行 logistic regression，最後得出 1 個 output 解。

### (2) Back Propagation

計算 Loss function(Cross entropy)對於每個 weight 的偏微分，根據 Chain rule:

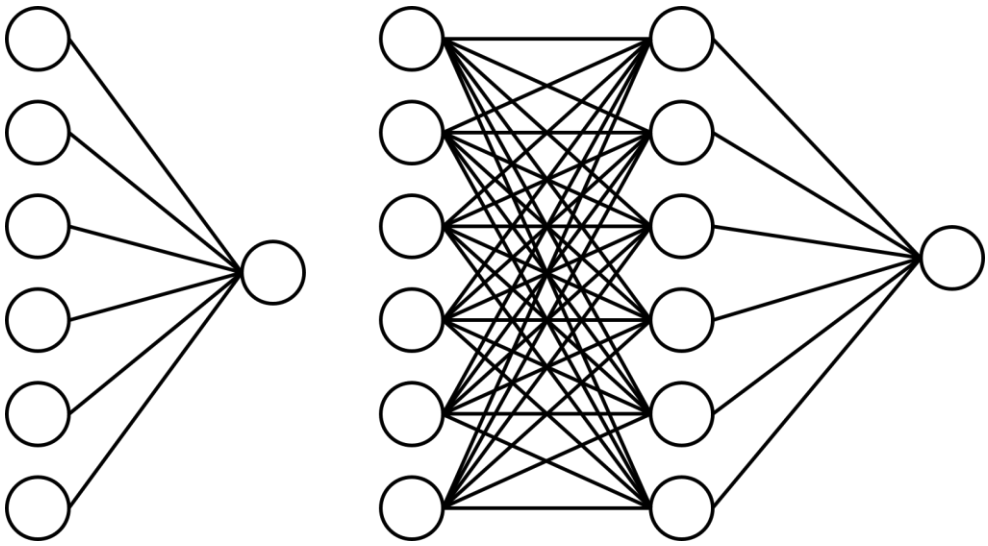
$\frac{\partial C}{\partial w} = \frac{\partial C}{\partial y}\frac{\partial y}{\partial z}\frac{\partial z}{\partial w}$，$\frac{\partial y}{\partial z}$ = sigmoid function 之微分，$\frac{\partial z}{\partial w}$ = 對於 w 的 input 值

對於每一層的 node，其 $\delta^l = M^T * \delta^{l-1}$，也就是說原本在 Pass Forward，此 node 對於下一層每個 node 貢獻多少 weight，那下一次每個 node 的$\delta$就貢獻多少 weight 給此 node 的$\delta$。

對於最後一層的單個 node，其

$$\delta = \frac{\partial C}{\partial y}\frac{\partial y}{\partial z} = \left(\frac{1-ans}{1-myans} - \frac{ans}{myans}\right) * (myans)(1-myans) = \text{myans} - \text{ans}$$

## Comparison



上左圖為 logistic regression，上右圖為 DNN 的結構圖。

從上圖可以看出，DNN 的空間複雜度遠超過單純的 logistic regression，同理時間複雜度也是差很多。Logistic 要 train 的是 57 個 weight 加上 1 個 bias，而我的 DNN model 要 train 的是 57*40 加上 40*1 個 weight 再加上 2 個 bias，同樣 Train 10000 epochs，logistic 約莫 5 分鐘，而 DNN 大約需要 15 分鐘以上。而前者的正確率在 public 上有 0.94，DNN 約 0.96，正確率只有上升 0.02，大約 6 題而已。

## Improvement

礙於時間因素，有些許技巧、方式，未加入我的 DNN Model 內，在此列舉一些可以優化 DNN 的方法。

### (1) Validation

此次的 Training Data 有 4001 筆，是屬於缺乏 Data 的情況，再切 Validation 可能會造成 Data 數更缺乏，但是 Data 數這麼少的情況，很有可能導致 Model overfit，使得 testing 的正確率下降。為了避免此情況，run 一次 program 的同時進行十次 training，十次皆獨立且隨機切資料出去做 Validation Set，做完十次後再將十組 Model 丟進 Validation Set 看 Loss，也必須再丟入全部的 Training Data 看 Loss，挑最少的進行 Test，以此降低 overfit 也增進準確率。

### (2) Dropout

由於 Data 過少，容易發生 overfit，在此情況下我們可以透過 Dropout，在每一層隨機選取 node 並將之移除，進行 training，因為 node 減少了，其他 node 相

對 weight 提高，必須再最後將 Model 取平均。在 layer 數不少的 Model 下，會有很大的功用，因為它等於是對這個 Model 取很多 Subset 進行 Training，會讓這個 Model 對於各種情況有比較高的適應力、準確率。但是這個方法在我一層的 DNN，預計只會有很小效用，因為他只是找一層的 subset，等於減少 node 數，所以在我相對比較簡單的 Deep Neural Network 來說比較無用武之地。