ML HW4

Q4.1：

首先讀取iris.data的資料，將此150筆資料前面四個attributes轉成float型態並append到x，而target(在這使用t)部分用label判斷，使用for-loop跑150圈，在前50圈讓t.append([1, 0, 0])，第51~100圈讓t.append([0, 1, 0])，第101~150圈讓t.append([0, 1, 0])。

接著我define了四個function：S、F、Weight、MSE。

F(nid)：

這個function用來計算兩層hidden-layer neurons及output-layer neurons的outputs，nid表示為第幾筆資料 (x)。

計算output的方法是利用雙層for-loop去實作sigmoid function（如下圖），內層計算attributes*weight的總和，外層則是每次（hidden-layers四次，output-layer三次）將總和*(-1)後做exponential，再用1除以將此結果+1，三層做法相似，不同的地方在於最下層（第三層）的attributes為training data的attributes，第二層的attributes為第三層做完的結果，最上層的attributes為第二層做完的結果，因此在此function三個for-loop的順序不可顛倒。

（以下簡稱第三層的output為h，第二層的output為g，第一層的output為y）

$$f(\Sigma) = \frac{1}{1 + e^{-\Sigma}}$$

S(nid)：

這個function用來計算兩層hidden-layer neurons及output-layer neurons的responsibility，nid表示為第幾筆資料 (x)。

最上層的是responsibility公式如下左圖，進行實作，responsibility會等於y*(i-y)*(t-y)。

第二層與第三層layer是代下圖右的公式，由於多了有後面sigma的部分，所以要多用一層回圈先算出sigma的部分（算第二層時responsibility代第一層的，算第三層時responsibility代第二層的），再乘上h(1-h)或g(1-g)，將直存入所屬陣列（s1、s2、s3）就完成了，直得注意的是，由於再算第二、三層時都需用到前一層的responsibility，因此三個for-loop的順序不可顛倒。

$$\delta_i^{(1)} = y_i(1 - y_i)(t_i - y_i)$$

$$\delta_j^{(2)} = h_j(1 - h_j) \sum_i \delta_i^{(1)} w_{ji}^{(1)}$$

Weight (lr, nid)：

這個function用來更新三層layer的weight（weight1、weight2、weight3），lr表示learning rate，nid表第幾筆資料 (x)。

更新weight的公式如下圖所示，但由於我的第二層的output neurons為g，第一層的為h，所以在實作方面，最上層必須代lr*s1*g (原本為lr*s1*h)，第二層必須代lr*s1*h，並用雙層for-loop去更改三層裡所有weight的值，就完成更新的部分了。

$$\text{'output-layer neurons: } w_{ji}^{(1)} := w_{ji}^{(1)} + \eta\delta_i^{(1)}h_j$$
$$\text{hidden-layer neurons: } w_{kj}^{(2)} := w_{kj}^{(2)} + \eta\delta_j^{(2)}x_k$$

Mse ()：

這個function用來算epoch的mean square error。

由於每個epoch的MSE是用最後一筆資料跑完的weight再重新計算每筆資料output-layer的output後做mean square error（公式如下圖）加總取平均，於是我用雙層for-loop來做，最外層為150圈表示做150筆資料，內圈是做公式裡sigma的部分，做完後return加總取平均的值。

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(t_i - y_i)^2$$

最後實作learning rate = 0.1的部分，先利用for-loop及random.uniform(-0.1,0.1)創造出weight1、weight2、weight3，h、g、y、s1、s2、s3也利用for-loop來初始化，設一個變數為converge用來判斷程式的終止點（這題converge的條件為：(當次epoch的MSE-前次epoch的MSE)/前次epoch的MSE 取絕對值小於0.0001時），接著用while-loop去跑（判斷S式為是否converge），裡面包一個150次的for-loop（因為資料有150筆），在裡面依序呼叫F(nid)、S(nid)、Weight(lr, nid)，接著用一個nsum累加每筆資料的的MSE（呼叫Mse (nid)），在做完後nsum即為當次epoch的MSE，除了第一個epoch外，其餘的epoch接需確認是否converge（條件如前述），重複做此while-loop直到converge即完成。

下圖為此題結果

```
learning-rate : 0.1
epoch :  1 The absolute fraction of change in MSE :  X
epoch :  2 The absolute fraction of change in MSE :  0.0075179005464747935
epoch :  3 The absolute fraction of change in MSE :  0.00015895789303902393
epoch :  4 The absolute fraction of change in MSE :  0.0006181389655207842
epoch :  5 The absolute fraction of change in MSE :  0.0003439880828420782
epoch :  6 The absolute fraction of change in MSE :  0.00012711333990968014
epoch :  7 The absolute fraction of change in MSE :  7.800540798624108e-06
Total epoch :  7
```

Q4.2：

在此題資料讀取以及四個function：S、F、Weight、MSE的define均與第一題相同，唯一不同的地方是這題需apply 5種不同的learning rate，於是我在while-loop的外層再包一個for-loop去apply 0.1~0.5的learning rate，其餘做法及converge部分也與第一題相同。

輸出結果在資料夾中的“HW4.2_result.txt”。

將epoch數量及learning rate做成圖表後可看出：找到適當的learning-rate非常重要！

重複做了好幾次的結果都有一兩筆破千個epoch，因此我認為learning-rate的影響非常大。