Machine Learning Report

HW2 Logistic Regression

姓名：許義宏 學號：B02901053

1. Logistic regresssion function :

```
prediction =  estimate(weight,bias,inputdata)
loss = cross_entropy(prediction, dataAns)
```

（圖一）

承接上次作業所學到的方法，我們知道能夠用weight與bias去拿來和inputdata做線性組合進行資料的預測；同樣的在Logistic Regression裡，我們依然透過類似的技巧去進行估測（圖一 行1），接着我們透過一個新學會的函式-cross entropy去進行loss的估測（圖一 行2），如此我們可以去評斷這樣的weight與bias的好壞。

不同的點是這次我們的估計除了線性組合外，我們會將預測出來的值丟進一個sigmoid函式

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

計算爲圖二 func sigmoid(x),
因此func estimate即爲計算這樣整個predict過程的函式。
而cross entropy 定義則爲

$$-\sum[\hat{y}lnf(x)+(1-\hat{y})ln(1-f(x))]$$

可以透過圖二func cross_entropy進行計算。

```
little_num = math.exp(-30)
def sigmoid(x):
    return 1.0/(1.0+np.exp(-x))
def estimate(wei, bi, feature):
    temp = np.dot(wei, feature)
    temp = temp + bi
    temp = sigmoid(temp)
    return temp
def cross_entropy(x,y):
    loss = 0
    for ii in range(y.size):
        if (y[0,ii] == 0):
            loss = loss + math.log(1-x[0,ii]+little_num)
        if (y[0,ii] == 1):
            loss = loss + math.log(x[0,ii]+little_num)
    return (-1)*loss
```

（圖二)

進行估測完後，每個iteration都要進行參數的更新：

```
diff_unit = dataAns - prediction
diff_w = (-1)*np.dot( diff_unit, np.transpose(inputdata))
diff_b = (-1)*np.sum(diff_unit)
```

```
weight = weight - learning_rate*diff_w
bias = bias - learning_rate_b*diff_b
```

（圖三）

更新的方法與linear regression相同，都是由Loss Funtion對參數w, b 進行Gradient Decent, 因此唯一不同的是計算Gradient Decent的值，經由上課的推導可知 $\frac{\partial C}{\partial w} = -\sum(\hat{y} - \sigma(x)) \cdot X^T$，此即爲diff_w算式所代表的，而diff_b 同理可知 $\frac{\partial C}{\partial b} = -\sum(\hat{y} - \sigma(x))$，如此即完成了核心的Logistic Rregression計算。

2. Describe your another method, and which one is best

在第二個方法裡，我用Neural Network的方法來完成練習，概念上是Logistic Regression的延伸，只不過是再多加一層的參數在其中。也就是說，在Hidden Layer（中間的層）裡的neural是input data利用weight. bias . sigmoid(activation function)所構成一群新的Feature(資料)，再將Feature用Logistic Regression的方法拿去估測output。

所以在大架構上與Logistic Regression類似，只是weight 會有兩組，bias亦同(如圖四)，且在做估測時要先計算Hidden Layer的值才能計算Prediction（如圖五）。而在更新時，我們需要計算Gradient Decent對於各個參數的Gradient則需利用BacK Propogation來進行計算（細節參考：http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/DNN%20backprop.pdf 第7-18頁），比較大的困難在於要一步一步的利用連鎖率推移計算（我的方法只需算兩次，因爲中間只有一層)(如圖六）。

```
num_hidden = 20
weight1 = np.random.uniform(-1,1,57*num_hidden).reshape(num_hidden,57)
bias1 = np.random.uniform(-1,1,num_hidden).reshape(num_hidden,1)
weight2 = np.random.uniform(-1,1,num_hidden).reshape(1,num_hidden)
bias2 = np.random.uniform(-1,1,1).reshape(1,1)
```

（圖四）

```
layer1 = estimate_sigmoid(weight1, bias1, inputdata)
prediction = estimate_sigmoid(weight2, bias2, layer1)
```

（圖五）

```
delta2 = dataAns - prediction
diff_w2 = (-1)*np.dot( delta2 , np.transpose(layer1)) #1*10
diff_b2 = (-1)*(np.sum( delta2 )).reshape(1,1)

#hidden layer
#relu_diff1 = diff_relu(np.dot(weight1,inputdata)+bias1)
sigmoid_diff1 = layer1 - np.square( layer1 )
pratial_delta = np.dot(np.transpose(weight2), delta2)#10*3601
delta1 = sigmoid_diff1*pratial_delta #num_hidden*3601
diff_w1 = (-1)*np.dot( delta1, np.transpose(inputdata) )
diff_b1 = (-1)*(np.sum( delta1,1)).reshape((num_hidden,1))
```

（圖六）

在結果的比較上，由於Neural Network等同於使用了一個對資料Feature前處理的手續去讓整體的Loss下降，而可以讓Output出來的預測值不管是training data loss 或是 validation data loss都來的更小。綜合來說，NN的表現比Logisitic Function表現的更好

3. Other discussions and details

以下將Logistic Regression與Method2分開討論

- ➔ Logistic Regressin:
  - ◆ 使用 Adagrad來進行learning rate的調整，比較在同樣的切validation的資料下，比較將accurancy train到0.90所需要的iteration數

Adagrad Rate:	0.001	0.01	0.1
Iteration 數	28300	270	145/但不穩定

所以考量到0.1初始的不穩定性，最終衡量選擇rate設爲0.02

- ◆ 使用Regularization來做測試(rate 0.02: iteration:10000，以validation set 測試的accuracy爲比較標準)

Lamda：	0	2	10	100
Acc	0.9375	0.94	0.9325	0.8875

在有一點點regularization的調整下，準確度有些微的上升，在Public Set的測試上也有類似的結果

➔ DNN（Only One Hidden Layer):

- ◆ Hidden Layer Neuron 數的調整：(iteration：Generally 18000, 若neuron數上升會上升部分，adagrad rate: 0.01，以Public Set測試結果爲參考)

Neuron數：	10	15	20	25	30
Acc	0.94333	0.94667	0.96000	0.92667	0.92667

- ◆ 考量到DNN的Model比較複雜，我的intial point採用random的方式，透過選擇seed去調整看看怎樣的seed做出來效果最好：
  測試的seed : { 1,2,3,4,5,6,7}
  用其他相同的參數下（Neuron20, Iteration18000, adagrad rate0.01），在Public Set上顯示最好的是seed 3

- ◆ 我還有嘗試過將hidden layer的activation function改用relu，但效果不佳，幾乎train不起來，猜測可能是對於負值的inputdata沒有反應的結果