ML HW2 Report B02902096 資工四 王浩恩

1. Logistic function:

```
for i in xrange(0,iterations):
    batch_it = i % (num_of_batch) #indicate the batch to train
    probability = sigmoid(np.dot(batch[batch_it],Theta) + bias) # feed the signoid funtion
    difference = -1 * (col_batch[batch_it][57] - probability) # calculate the difference between table and probability
    grad_map = col_batch[batch_it] * difference
    b_grad = np.sum(difference)
    AdaGrad[0] += b_grad**2
    bias = bias - (learning_rate/(AdaGrad[0]**(1/2))) * (b_grad)
    for j in xrange(0,len(grad_map) - 1):
        w_grad = np.sum(grad_map[j])
        AdaGrad[j+1] += w_grad**2
        Theta[j] = Theta[j] - (learning_rate / (AdaGrad[j+1] ** (1/2))) * (w_grad)
```

使用投影片上的公式 + AdaGrad

2. Another method:

我做了一個2 layers 的 NN，learning rate是固定的，最後上傳private board的分數沒有比原本的logistic regression高，可能還是要加上adagrad 或是 adadelta才能下的去最小值。

```
syn0 = 2 * np.random.random((50,64)) - 1
syn1 = 2 * np.random.random((64,1)) - 1
count = 0
for j in xrange(4000):
    l0 = trainingData
    l1 = nonlin(np.dot(l0,syn0))
    l2 = nonlin(np.dot(l1,syn1))
    y = y.reshape(y.shape[0],1)
    l2_error = y - l2
    if (j% 100) == 0:
        print "Error:" + str(np.mean(np.abs(l2_error)))
    l2_delta = l2_error*nonlin(l2,deriv=True)
    l1_error = l2_delta.dot(syn1.T)
    l1_delta = l1_error * nonlin(l1,deriv=True)
    syn1 += 0.0001 * l1.T.dot(l2_delta)
    syn0 += 0.0001 * l0.T.dot(l1_delta)
```

NN的作法參考了此網頁：
http://iamtrask.github.io/2015/07/12/basic-python-network/

3. Other discussion:

logistic regression的部分，一開始上傳public board只有0.91左右。後來做了以下的修正，才讓分數突破baseline。

(a). batch:

我把data切成500一個batch，然後把其中一個batch留作validation。

(b). AdaGrad:

在hw1中我並沒有做AdaGrad，這次加上AdaGrad之後validation的正確率就有上升，然而似乎還是不夠，下次會嚐試使用AdaDelta。

(c). normalization:

一開始把dot完的結果直接丟進sigmoid，就發現它輕易的就overflow，為了解決這問題我把所有data都normalize，並且增設邊界條件，就能夠避免exponential的計算數字過大。