Machine Learning Homework#2

--spam classification

b02901120 羅志軒

實驗流程

本次作業以助教給的4001筆57維feature加上1維bias為training data，搭配每一筆資料的label為testing data，訓練出一個能夠辨識垃圾郵件的binary classification model，這次我分別應用logistic regression和deep neural network兩種方式以做比較。

Logistic Regression

logistic regression 是一種常被應用於binary classification的迴歸方式，概念相當於一群資料中切出一條界線，將資料群分成兩類，並用cross-entropy cost function計算現在兩組資料預測結果的cost，回傳並更新參數，重複以上步驟使cos持續t降低，以下為公式推導及應用於程式上的方式：

classification function：$f(x) = \sigma\ (\Sigma w_i x_i + b)$

cross entropy function：$L(x) = -[y^n lnf(x^n) + (1-y^n) * (1 - lnf(x^n))]$

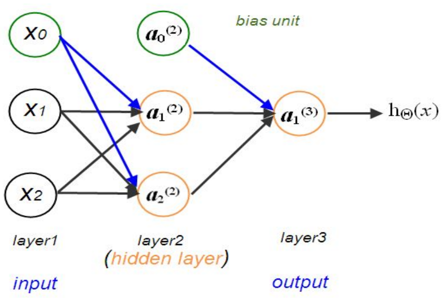
```
f = (np.array(1+ np.exp(-np.dot(train, w_list)), dtype= float))**(-1) # classification function
error= np.mean(-(target* np.log(f+1e-30)+ (1- target)* np.log(1- f+1e-30))) #cross entropy function
```

logistic regression：$\Delta w_i = -\eta(\Sigma - (y^n - f(x^n))x_i^n)$

```
def logistic_regression(x_list, yn_list, w_list): #regularization, x_list, y_list-> hole batch list
    gradient = np.dot(x_list.T* (-1), (yn_list- 1.0/ (1.0+ np.exp(-np.dot(x_list, w_list))))) / len(x_list)
    return gradient
```

Deep Neural Network

neural network 是現今機器學習的主要趨勢，概念為將training feature乘上各自的weight投影到hidden layer上，經過一層或多層的hidden layer最後將classification的預測結果output出來計算和正確label之間的cost，透過backpropagation回傳cost並更新各層weight參數，這裡我只使用一層hidden layer，hidden layer中有64個node，每個neuron的activation function為sigmoid funtion，基本上每個neuron的運作方式和上述的logistic regression是相同的，以下為公式及程式上的應用：



Forword propagation:

a_i^j : *activation of unit i in layer j*

w^j : *weight mapping from layer j to layer j + 1*

z_i : *input of layer i + 1*

$a^1 = x$

$a^2 = \sigma(z_2) = \sigma(a^1 w^1)$ ， $a_0^2 = 1$

```
###Forward propagation###
a = [train]
for l in range(len(weight)):
    z= np.dot(a[l], weight[l])
    activation= 1.0/ (1.0+ np.exp(-z))
    if l < len(weight)-1:
        activation.T[0].fill(1.0)
    a.append(activation)
```

$a^3 = \sigma(z_3) = \sigma(a^2 w^2)$

Back propagation:
δ_j^l : *error of node j in layer l*

$\delta^3 = -[y/a^3 - (1-y)/(1-a^3)] * [a^3 * (1-a^3)]$
$\delta^2 = \delta^3.(w^2)^T * [a^2 * (1-a^2)]$
$w^j <- w^j - \eta * a^j.\delta^{j+1}$

```
###Backward propagation###
error= -(target/(a[-1]+1e-30)-(1-target)/ (1-a[-1]+1e-30))* (a[-1]*(1-a[-1]))
deltas= [error]#* sigmoid_prime(a[-1])]
for l in range(len(a)-2, 0, -1):
    deltas.append(np.dot(deltas[-1],weight[l].T)*(a[l]*(1-a[l])))
deltas.reverse()
for i in range(len(weight)): # a1*delta2, a2*delta3
    g_sum[i]= adagrad(g_sum[i], np.dot(a[i].T,deltas[i]))
    weight[i]-= lr* np.dot(a[i].T,deltas[i])/ g_sum[i]
```

Performance

logistic regression best:

learning rate:	0.1
iteration:	5000
train set number:	3800
validation set number:	201
adaptive learning rate:	adagrad
result:	0.93 (in public set)

DNN best:

learning rate:	0.15
iteration:	1500
train set number:	3000
validation set number:	1001
adaptive learning rate:	adagrad
result:	0.96 (in public set)

比較起來，logistic regression的traing loss下降幅度較慢，且精確度相對較低，DNN的優勢在於其參數量大，參數更新也較為穩定且明確，精確度相對較高，每次iteration都會隨機抽取一段validation set，數量大約在train set數量的1/3~ 1/10，當validation loss減小幅度開始減緩後會記住目前validation loss最低的weight參數，以做為最終輸出的model參數，可以有效避免model的overfitting。

比較其餘參數調整的影響，兩種方法都是使用adaptive learning rate訓練速度較快，參數更新較不會震盪; DNN hidden layer的neuron數大約在32～64之間會得到較好的精確度，若持續增加neuron數不僅訓練速度緩慢，精確度也會下降，此外，增加hidden layer的數目，對精確度的提昇並不會很顯著; training iteration超過3000可能會有overfiitting的情況產生。