Machine Learning HW2 report

R05944013 網媒一 高滿馨

1.Logistic Regression Function

使用cross entropy 當cost function, 把trainData和weight做內積後, 送進sigmoid function 得到最終的預測值，使用adagrad algorithm來調整learning rate。最後classify的方式是，假如最終值的結果< 0.5 就是 0, 否則就是1。

$$C = -\frac{1}{n}\sum_{x}\left[y \ln a + (1-y)\ln(1-a)\right],$$

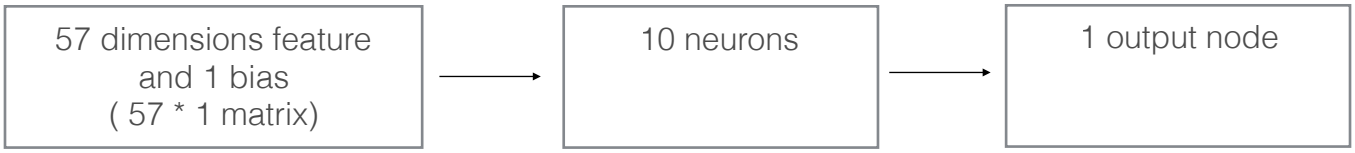
```
def sigmoid( X ):

    d = 1.0/(1.0 + np.exp( -1.0*X))
    return d
```

code fragment: sigmoid function

```
def gradientDescent( trainData, yHead, weight, count ):

    J_History = zeros( shape = ( iteration, 1 ) )
    accumulate = 0

    for x in range( 0, iteration ):


        prediction =  sigmoid( trainData.dot(weight) )


        for i in range( featureNum+1 ):

            #get trainData
            tmp = trainData[ :, i ]
            tmp.shape = ( count, 1 )

            #compute gradient
            derivative = ( ( ( prediction - yHead )*tmp ).sum() )/count

            #update accumulate ( adagrad algorithm )
            accumulate = accumulate + derivative*derivative
            #compute learning rate
            learningRate = alpha/(delta+sqrt(accumulate))

            #update weight
            weight[i][0] = weight[i][0] - learningRate*derivative

        J_History[x][0] = computeErrorRate( trainData, yHead, weight )
        print("finish iteration " + str(x) + ", error is "+ str( J_History[x][0] ) )

    return weight, J_History
```

code fragment: gradient descent

2. 方法2是使用3層的neural network來做training，第一層是input layer，第二層是hidden layer，neuron數可以調整，但試過多種參數後，最後決定使用10個neurons，最後一層是outputnode，可以直接output出最終值，假如小於0.5就判定為0，否則判定為1。sigmoid function則是使用hyper tangent。有使用adagrad algorithm來調整learning rate，另外有再去計算momentum來更新，效果確實有比較好。最後最好的model是使用adagrad和momentum，iteration為100圈的架構。



input layer　　hidden layer　　output layer

neural network架構圖

結果上傳kaggle的部分，最後neural network最好的model有比logistic regression的效果好。可能是因為中間那層hidden layer有做到feature transformation，所以原本無法被分開的一些測資，最後都可以順利地被classify，因此效果比logistic regression的效果好。

3.這次的作業中，在實作完基本的架構後，optimization的部分比較著重在learning rate調整的部分。有想過要用統計的方式去計算correlation，看是否能夠降維，不過看過data後覺得大部分的data都是0，感覺correlation不會有太好的效果，所以就沒有去測試。adaptive learning rate的部分，有把之前學過的多種演算法都拿來試試看，包括RMSProp，Adam等等，不過這些演算法雖然可以有效降低training error rate，找到真正的最低點，但反而會造成overfitting，準確度反而沒有提升。尤其是RMSProp，training error可以降低到0.0，但public set的準確度很低。因此，可以看出調整learning rate來得到optimize的效果是有限的。