

**模式识别大作业**

题 目 Logistic回归

学 院 信息科学与工程学院

专 业 信息与通信工程

组 员 桂少军

指导教师 赵海涛

**完成日期： 2018 年 10 月23日**

**模式识别作业报告——Logistic回归**

组员：桂少军

经过几周模式识别学习，在赵海涛老师的辛勤指导下，我对模式识别的Logistic回归算法有了一定的了解，并通过本次针对猫数据集的识别实验来巩固所学内容。

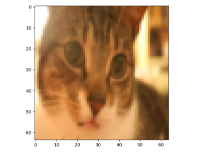
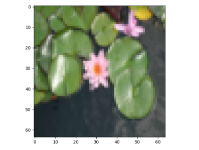
下面将详细说明我的解决过程。

1. **整体解决方案**

**1.1数据分析**

数据集里面分为训练集和测试集,以HDF5文件格式存储。训练集和测试集里面分别有209张和50张64\*64\*3的图片。猫的对应label为1，不是猫的label则为0。

图片示例如下：

cat（label=1） non-cat(label=0)

图1

首先要将数据集导入，代码如下所示：

|  |
| --- |
| import numpy as np  import h5py  def load\_dataset():  train\_dataset = h5py.File('datasets/train\_catvnoncat.h5', "r")  train\_set\_x\_orig = np.array(train\_dataset["train\_set\_x"][:]) # your train set features  train\_set\_y\_orig = np.array(train\_dataset["train\_set\_y"][:]) # your train set labels  test\_dataset = h5py.File('datasets/test\_catvnoncat.h5', "r")  test\_set\_x\_orig = np.array(test\_dataset["test\_set\_x"][:]) # your test set features  test\_set\_y\_orig = np.array(test\_dataset["test\_set\_y"][:]) # your test set labels  classes = np.array(test\_dataset["list\_classes"][:]) # the list of classes    train\_set\_y\_orig = train\_set\_y\_orig.reshape((1, train\_set\_y\_orig.shape[0]))  test\_set\_y\_orig = test\_set\_y\_orig.reshape((1, test\_set\_y\_orig.shape[0]))    return train\_set\_x\_orig, train\_set\_y\_orig, test\_set\_x\_orig, test\_set\_y\_orig, classes |

接下来对数据进行一下处理，将训练集和测试集的维度降低并转置，并将数据集标准化：

|  |
| --- |
| m\_train = train\_set\_y.shape[1]  m\_test = test\_set\_y.shape[1]  num\_px1 = train\_set\_x\_orig.shape[1]  num\_px2 = train\_set\_x\_orig.shape[2]  train\_set\_x\_flatten = train\_set\_x\_orig.reshape(train\_set\_x\_orig.shape[0], -1).T  test\_set\_x\_flatten = test\_set\_x\_orig.reshape(test\_set\_x\_orig.shape[0], -1).T  train\_set\_x=train\_set\_x\_flatten/255  test\_set\_x=test\_set\_x\_flatten/255 |

数据处理之后的结果如下：

训练集降维之后的维度：(12288, 209)

训练集\_标签的维数 : (1, 209)

测试集降维之后的维度: (12288, 50)

测试集\_标签的维数 : (1, 50)

**1.2算法原理及程序实现**

**Logistic回归实际上不是回归，而是分类。是从线性回归中衍生出来的分类策略。当y值为只有两个值时(比如0，1)，线性回归不能很好的拟合时，用logistic回归来对其进行二值分类。**

**我们的目的就是训练出图片的每个像素所对应的权重值，使得成本函数最小，从而找到最佳的权值，达到能够预测的效果。**

**实现过程如图2所示：**

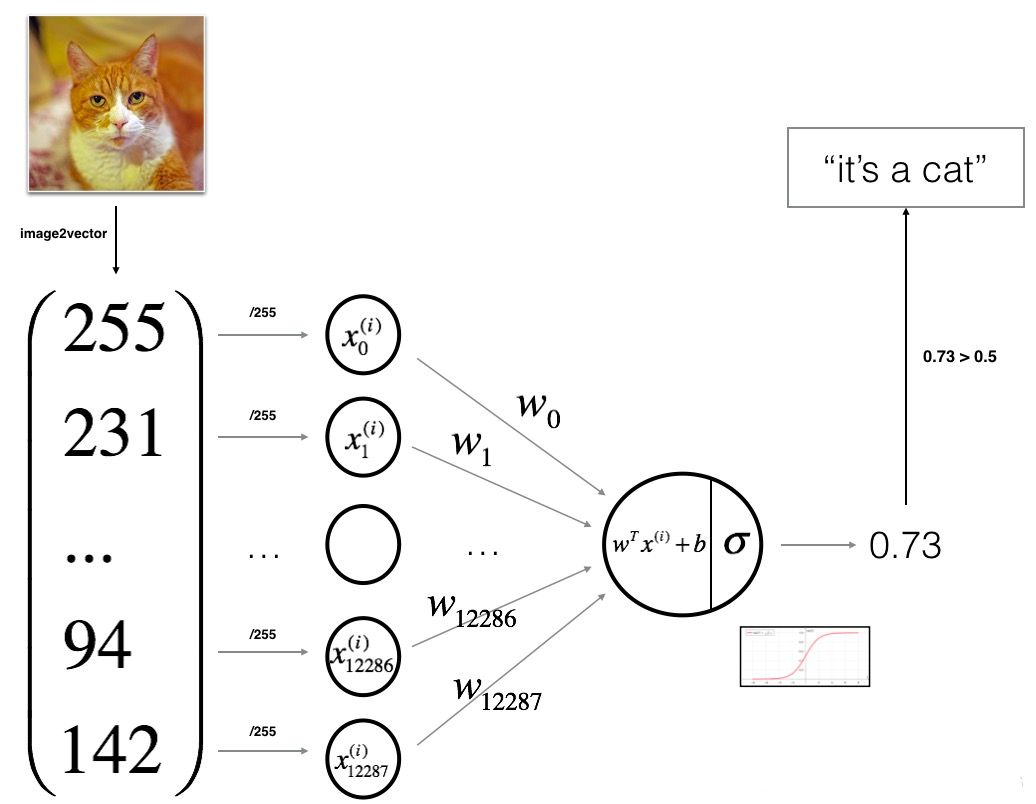


图2

1.2.1公式推导

对于输入:

(1)

(2)

(3)

然后通过对所有训练样例求和来计算成本:

(4)

其中，式（3）可由最大似然估计推导而来，以下为推导过程：

这里逻辑函数为：

(5)

于是，可得估计函数为：

(6)

这里，我们的目的是求出一组值，使得这组可以很好的模拟出训练样本的类值。

由于二值分类很像二项分布，我们把单一样本的类值假设为发生概率，则：

(7)

(8)

可以写成概率一般式：

(9)

 由最大似然估计原理，我们可以通过m个训练样本值，来估计出值，使得似然函数值最大：

(10)

这里，为m个训练样本同时发生的概率。对求log，得：

(11)

式（11）即为最后结果。

1.2.2具体程序实现

算法过程主要有以下3个部分：

1. 定义模型结构（例如输入特征的数量）   
2. 初始化模型的参数   
3. 循环：

3.1 计算当前损失（正向传播）

3.2 计算当前梯度（反向传播）

3.3 更新参数（梯度下降）

现在构建sigmoid，需要使用 sigmoid（w^T\*x + b）计算来做出预测。

|  |
| --- |
| def sigmoid(z):      s = 1/(1+np.exp(-z))      return s |

接着就可以初始化我们需要的参数w和b了。

|  |
| --- |
| def initialize\_with\_zeros(dim):      w = np.zeros((dim,1))      b = 0      assert(w.shape == (dim, 1))      assert(isinstance(b, float) or isinstance(b, int))      return (w,b) |

初始化参数的函数已经构建好了，现在就可以执行“前向”和“后向”传播步骤来学习参数。我们现在要实现一个计算成本函数及其渐变的函数propagate

|  |
| --- |
| def propagate(w, b, X, Y):      m = X.shape[1]  #正向      A = sigmoid(np.dot(w.T,X)+b)      cost = (-1/m)\*np.sum(Y\*np.log(A)+(1-Y)\*(np.log(1-A)))  #反向      dw = (1/m)\*(np.dot(X,(A-Y).T))      db = (1/m)\*(np.sum(A-Y))      assert(dw.shape == w.shape)      assert(db.dtype == float)      cost = np.squeeze(cost)  assert(cost.shape == ())  #创建一个字典保存dw,db    grads={            'dw': dw,            'db': db          }  return (grads,cost) |

现在，我要使用梯度下降更新参数

|  |
| --- |
| def optimize(w,b,X,Y,num\_iterations,learning\_rate,print\_cost=False):      costs = []      for i in range(num\_iterations):          grads,cost = propagate(w,b,X,Y)          dw = grads['dw']          db = grads['db']          w = w-learning\_rate\*dw          b = b-learning\_rate\*db          if i%100 == 0:              costs.append(cost)          if (print\_cost) and (i%100 == 0):              print("迭代次数：%d ，误差值：%f" % (i,cost))      params = {'w': w,'b': b}      grads = {'dw': dw, 'db': db}  return (params, grads, costs) |

预测一下：

|  |
| --- |
| def predict(w,b,X):      m = X.shape[1]      Y\_prediction = np.zeros((1,m))      w = w.reshape(X.shape[0],1)      A = sigmoid(np.dot(w.T,X)+b)      for i in range(A.shape[1]):          if A[0,i]>0.5:              Y\_prediction[0,i]=1          else:              Y\_prediction[0,i]=0      assert(Y\_prediction.shape == (1,m))      return Y\_prediction |

最后是主程序：

|  |
| --- |
| def model(X\_train,Y\_train,X\_test,Y\_test,num\_iterations=2000,learning\_rate=0.5,print\_cost=False):      w, b = initialize\_with\_zeros(X\_train.shape[0])      parameters,grads,costs = optimize(w,b,X\_train,Y\_train,num\_iterations,learning\_rate,print\_cost)      w, b = parameters['w'],parameters['b']      Y\_prediction\_train = predict(w,b,X\_train)      Y\_prediction\_test = predict(w,b,X\_test)      print("训练集的准确度为：",format(100-np.mean(np.abs(Y\_prediction\_train-Y\_train)\*100)),"%")      print("测试集的准确度为：",format(100-np.mean(np.abs(Y\_prediction\_test-Y\_test) \* 100)), "%")      d = {              'costs': costs,              'Y\_prediction\_train': Y\_prediction\_train,              'Y\_prediction\_test': Y\_prediction\_test,              'w': w,              'b': b,              'num\_iterations': num\_iterations,              'learning\_rate': learning\_rate}      return d  print("============测试model=============")  d = model(train\_set\_x, train\_set\_y, test\_set\_x, test\_set\_y, num\_iterations=2000, learning\_rate=0.005, print\_cost=True)  #绘图  costs = np.squeeze(d['costs'])  plt.plot(costs)  plt.xlabel("iterations(per hundred)")  plt.ylabel("costs")  plt.title("learning\_rate= "+str(d['learning\_rate']))  plt.show() |

1.3程序结果及分析

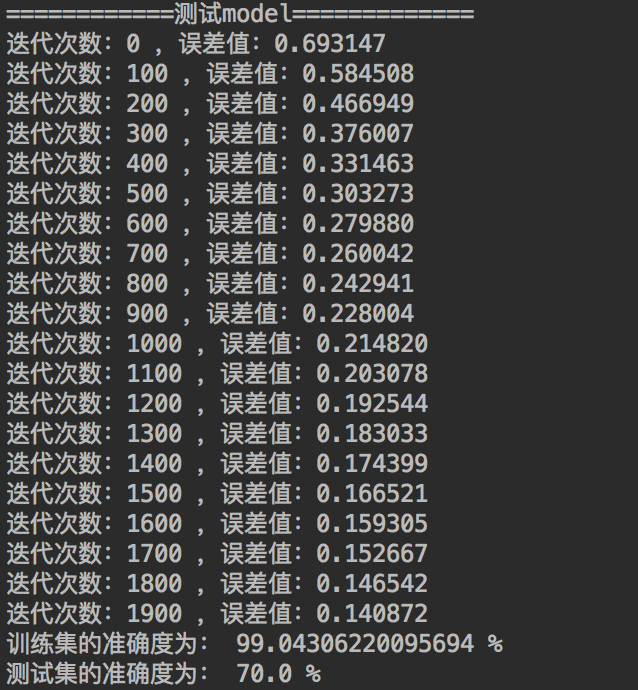
 

图3

分析：从结果可以看出，训练集的准确度高达99%，而测试集的准确度只有70%，出现了过拟合。这个原因首先要归结于数据集太少，可以通过增大数据集来改善，也可以通过增加隐藏层的数量以及正则化来改善。

改善方案:通过5层（包含输入层和输出层）的神经网络来训练，结果如图4所示，可以看出训练集的准确度还是99%，而测试集的准确度已经提升到了78%

其实最根本的解决问题还是要增加数据集的数量，才能大幅度的改善结果。

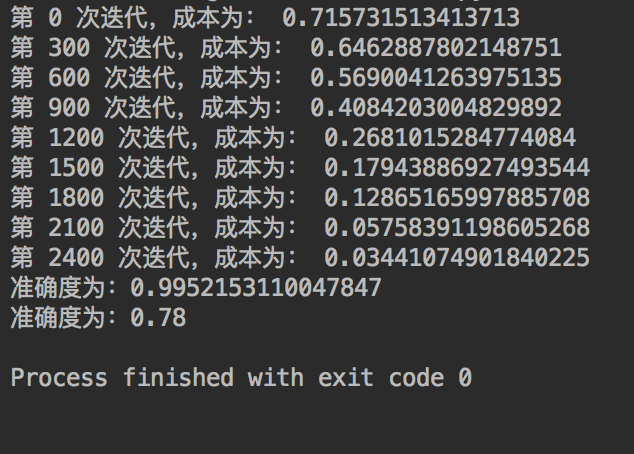


图4

**二、小组分工**

程序设计及编写：桂少军

程序调试：桂少军

实验报告：桂少军

**三、作业总结**

本次实验刚开始做了lintcode上广告点击率的题目，用了I1～I13的13个计数特征来进行训练，但是最后训练集和测试集的准确度都只在80%附近，所以放弃了这个数据集。后来试了一下iris鸢尾花的数据集，效果比较不错，但是老师说这个太简单，所以最后改用相同的算法来训练猫的特征集，没有遇到太大的麻烦。

这次的大作业增加了我对模式识别的兴趣，让我对logistic回归算法有了更深的理解。感谢赵老师教学帮助，并且在作业过程中给了我们大家很多指导。