

**模式识别大作业**

题 目 基于iris数据集的logistic回归

学 院 信息科学与工程

专 业 控制科学与工程

组 员 程梁

指导教师 赵海涛

**完成日期： 2018 年 10 月24日**

**模式识别作业报告——logistic regression**

Logistic 回归简介：线性回归模型形式简单，易于建模，对于回归任务此算法能够通过数据集学得一个通过属性线性组合来进行预测的函数，即



是其d个属性，一般用向量形式写成



但是若要做分类任务该线性回归模型显然行不通，这时需要考虑一个单调可微函数G,令



得到“广义线性模型”，这样将分类任务的真实标记y与线性回归模型的预测值联系起来。考虑二分类任务，其输出标记y={0,1},而线性回归模型产生的预测值f(x)是实值，利用单调可谓函数G将实值f(x)转换为0/1值。最理想的函数莫过于单位阶跃函数，即预测值大于零则就判为正例1，小于零则判为反例0，为临界值时可任意判断。但该函数的最大缺点是不连续，如果选择单位阶跃函数，不利于后续工作的展开。人们希望找到一个在一定程度上能够近似单位阶跃函数的替代函数，而且此函数是单调可微的。Sigmoid函数正是这样的一个函数。其形式为：



该函数有一个很好的性质，即



此性质为后续的性质提供了方便，而将Sigmoid函数作为,代入上式中即可得到logisitic回归的模型



Logistic 回归的基本原理：logistic回归的基本原理与线性回归十分相似。首先找到一个合适的预测函数，即分类函数,其根据输入数据预测相应的输出。然后构造损失函数Cost，该函数表示模型预测输出与训练数据的真实输出值之间的偏差。综合考虑所有训练数据的损失，将Cost求和或者取平均记为函数。表示所有训练数据与实际输出的偏差。现在需要找到我们假设的一组参数使得所得到的达到最小，即最优化问题。找到函数最小值的方法有很多，我们这里采用的是梯度下降法。

具体过程：1.首先构造预测函数



这里介绍的是线性边界，对于非线性边界不做重点的介绍。

函数的数值有特殊的意义：它表示结果取1的概率，因此对于类别1与类别0的概率分别为：



构造Cost函数：

取似然函数：

对似然函数取对数：



使得最大似然估计取最大值时的即为要求的参数，同时也可使



因为加了一个负号，则要求使达到最小值的值，可用梯度下降法求解最优的。

梯度下降法求解的最小值：







代入上式最终结果可得



故



根据表达式可发现，logistic回归的梯度下降表达式与线性回归十分相似，下面将上述表达式转化为向量矩阵间的运算。







x为已知数据集，y为实际输出，待定参数为Ø；





根据上式可得

将所有的参数组合为一个向量即得到梯度下降的矩阵形式



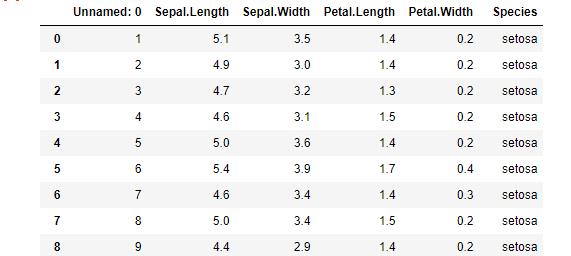
E表示了预测值与真实值的误差向量。以上是logistic回归在梯度下降法寻找最佳参数的步骤，其中参数m可以省略。

案例展示：本次作业采用了较为简单的iris数据集，其数据结构如下

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
| 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 5.1 | 2.5 | 3 | 1.1 | versicolor |
| 6.3 | 3.3 | 6 | 2.5 | virginica |

表中数据为整体数据中截取的一部分。前四列为四个特征就是我们模型的输入数据，最后一列为类别。由于logistic回归只能做二分类问题，故在处理数据时我们将最后一个类别virginica及其对应的特征全部删去，只保留前两个类别，并且将类别setosa归为0，类别versicolor归为1。使用处理后的数据对问题进行求解。

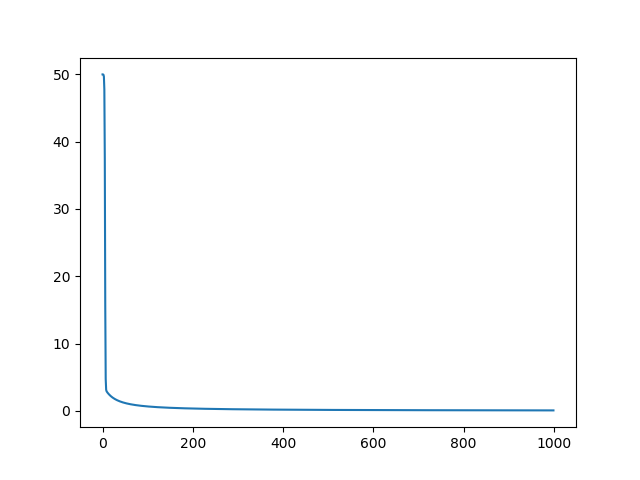
数据读入与处理：本次作业的实现主要是采用python编程，故数据的读入与处理采用的是pandas包完成，将数据集读取为数据框架，如下图所示：



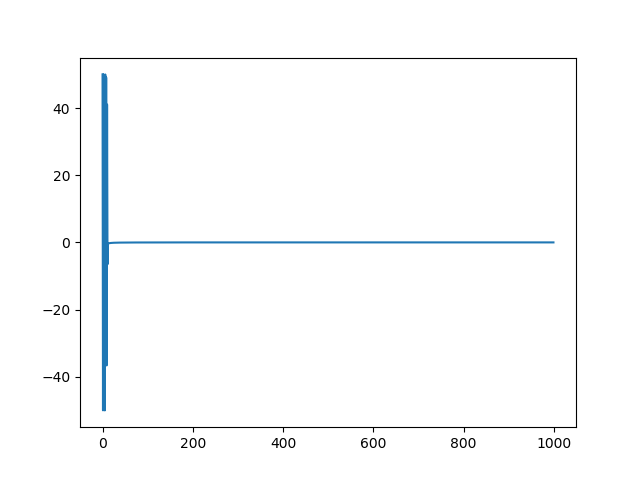
对于上图数据，在数据框架下将Species中的两个类别分别归为0和1。然后将Species这一列读出作为真实的输出数据，然后删去Species，将剩余的数据作为输入数据转换为矩阵，然后删去标序的第一列，然后再为此矩阵添上一个全为1的列作为它的第一列。这是就得到了我们需要的输入数据（矩阵）。关于logistic回归算法的原理上文已经阐述的较为清晰，这里就不再多做赘述。接下来就是编程实现：该算法的核心步骤如下所示：

|  |
| --- |
| def Training(data,label):  dataMatrix = np.mat(data).astype(np.float64)  labelMatrix = np.mat(label).transpose().astype(np.float64)#将输入输出转化为矩阵形式  dataMatrix = dataMatrix[:,1:6]#取输入矩阵的2~5列  m,n = np.shape(dataMatrix)  a = np.ones((m,1))  dataMatrix = np.c\_[a,dataMatrix] #为输入矩阵增添一个全为1的列作为其第一列  w = np.ones((n+1,1)) #初始参数全部置为1  alpha = 0.001 #取学习率，即下降步长  c = []  for i in range(1000):  prediction = Sigmoid(dataMatrix \* w)  loss = prediction - labelMatrix  d = np.ones((1,m))  c.append(np.ravel(d\*loss)) #保存每次学习的损失  w = w - alpha \* dataMatrix.transpose() \* loss #梯度下降更新参数  return w,c |

由于数据样本较少，且没有训练集。所以我保留了每次训练之后预测值与真实值的误差作为观察算法运行的依据。下图是该算法迭代1000次，学习率为0.01所得的误差变化曲线。



有训练结果可知误差最终稳定在0.07左右；增大学习率，将学习率调为0.01,所得误差图像为：



由训练结果可知，误差在前期会出现剧烈抖动，在一定次数趋于平稳并且稳定在0.02之间。再增大学习率，所得误差图像与上面相同，但误差会渐渐趋近于0。可以认为，当误差小于一定值时，可以得到最佳的参数估计值。

作业总结：

由于自己的编程能力较为薄弱，加上时间紧张，所以这次作业选择了一个较为简单的题目，并且更加注重算法原理的推导，并且logistic回归优化并不仅仅局限于批量梯度下降，也可以通过随机梯度下降法，牛顿法等等都可以实现参数的优化，同时对于模型过拟合与欠拟合时的处理方法，这些方法也都值得我们去尝试。但此次实验也存在一些缺陷，由于数据量较小，且没有训练集，只能通过误差曲线来观察模型的工作情况。最后，通过此次作业自己也对logistic回归的梯度下降过程有了一个全面的了解与认识，同时也增加了对模式识别算法的兴趣。