



机器学习与模式识别课程报告

题目：利用 Logistic 模型的分类问题

组 员： 王倩妮

班 级： 交通 2015-02 班

学 号： 2015112956

任课教师： 郝 莉

2018 年 4 月 22 日

一、问题描述

利用 Logistic 回归与随机梯度上升算法来预测患有疝病的马的存活问题

二、数据准备

2.1 数据描述

数据来源于 UCI 机器学习库，数据中包含 368 个样本，每个样本有 28 个特征，数据中包含医院检测马疝病的一些指标，数据中存在约 30% 的确实状况，还有一些指标较为主观而难以测量。

2.2 数据预处理

2.2.1 数据预处理一般方法

- 使用可用特征的均值来填补缺失值；
- 使用特殊值来填补缺失值，如-1；
- 忽略有缺失值的样本；
- 使用相似样本的均值填补缺失值；
- 使用另外的机器学习算法预测缺失值。

2.2.2 本节数据预处理方法

在本阶段需完成两项工作，其一是所有缺失值必须用一个实数值来替换，因为使用 numpy 进行后期运算不允许包含缺失值，考虑到使用 Logistic 回归，因此选择使用 0 来代替所有缺失值，选用 0 替换，在更新时不会影响系数。

```
weights = weights + alpha * error * dataMatrix[randIndex]
weights = weights    if dataMatrix[randIndex] = 0
```

此外由于 $\text{sigmoid}(0) = 0.5$ ，对于预测结果不具有任何倾向性，因此上述做法不会对误差项造成影响。且数据集特征值一般不为 0，也满足“特殊值”的要求，因此使用 0 填补缺失的特征值。

工作二是如果发现一条数据的类别标签缺失，则直接删除该条数据。因为对于类别标签的预测存在一定难度。

三、建模与预测

3.1 从疝气病症预测病马的死亡率实施步骤

- 1) 收集数据：给定数据文件。

- 2) 准备数据：用 `python` 解析文本文件并填充缺失值。
- 3) 分析数据：可视化并观察数据。
- 4) 训练数据：使用优化算法，找到最佳系数。
- 5) 测试算法：为了量化回归效果，需要观察错误率。根据错误率决定是否退到训练阶段，通过改变迭代次数和步长等参数来得到更好的回归系数。
- 6) 使用算法：实现一个简单的命令程序来收集马的病症并输出预测结果。

3.2 使用 Logistic 方法进行预测

使用 Logistic 回归方法进行分类所需工作不多，只是需要把测试集上的每个特征向量乘以最优化方法得来的回归系数，再讲该乘积结果求和，最后输入 Sigmoid 函数中即可。Sigmoid 函数值小于 0.5 则标签为 0，否则为 1。

● 预设函数部分

在使用本案例的测试函数前，已经定义了一些用于进行 Logistic 回归的函数，现介绍如下：

➤ loadDataSet 函数

loadDataSet 用于数据读入，通过打开 txt 文件，按照行进行读入后利用 `split` 进行分割，划分特征和标签，并进行存储。为了方便计算，将 `X0` 统一补充为 1.0。

```
def loadDataSet():
    dataMat = []; labelMat = []
    fr = open('testSet.txt')
    for line in fr.readlines():
        lineArr = line.strip().split()
        dataMat.append([1.0, float(lineArr[0]), float(lineArr[1])])
        labelMat.append(int(lineArr[2]))
    return dataMat, labelMat
```

➤ sigmoid 函数

sigmoid 函数用于还原 Sigmoid 函数，进行运算。

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

```
def sigmoid(inX):
    return 1.0/(1+np.exp(-inX))
```

➤ gradAscent 函数

Sigmoid 函数的输入为 z ，可通过下列公式得出：

$$z = w_0x_0 + w_1x_1 + \dots + w_nx_n$$

若写成向量形式，则 x 为特征向量（输入数据）， w 为最佳参数，因此需要通过一些最优化方法寻找这个最佳参数。梯度上升方法即是方法之一。当然随后也将扩展到随机梯度上升法与改进的随机梯度上升法。

在二维平面内，梯度可以定义为：

$$\nabla f(x,y) = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix}$$

梯度算子总沿函数数值增长最快的方向变化，此处的找到了方向，而移动的步长设为 α 。梯度上升的迭代公式可以用下式表示：

$$w := w + \alpha \nabla_w f(w)$$

此迭代直至迭代次数达到某一值或者算法达到某个误差允许的范围内停止。

在 `gradAscent` 函数中输入特征值与标签值，首先现将值转化为 `numpy` 的 `matrix` 形式，之后再通过循环更新系数。（此处不太懂：梯度是连续的概念，为何推广到离散的数据后是用 `dataMatrix.transpose()* error`）

```
def gradAscent(dataMatIn, classLabels):#一般的梯度上升寻找最大值
    dataMatrix = np.mat(dataMatIn)           #convert to NumPy matrix
    labelMat = np.mat(classLabels).transpose() #convert to NumPy matrix
    m,n = np.shape(dataMatrix)
    alpha = 0.001
    maxCycles = 500
    weights = np.ones((n,1))
    for k in range(maxCycles):                #heavy on matrix operations
        h = sigmoid(dataMatrix*weights)      #matrix mult
        error = (labelMat - h)               #vector subtraction
        weights = weights + alpha * dataMatrix.transpose()* error #matrix mult
    return weights
```

➤ `stocGradAscent0` 函数

由于一般的梯度上升法每次更新回归系数是都需要遍历整个数据集，计算量较大，不适合数据量较大的情况，因此改用随机梯度上升法，该方法以此仅用一个样本点来更新回归系数。

在此函数中 `h` 与 `error` 都是数值，避免了梯度上升法向量相乘造成的大运算量。

```
def stocGradAscent0(dataMatrix, classLabels):#随机梯度上升
    dataMatrix=np.array(dataMatrix)
    m,n = np.shape(dataMatrix)
    alpha = 0.01
    weights = np.ones(n)    #initialize to all ones
    for i in range(m):
        h = sigmoid(sum(dataMatrix[i]*weights))
        error = classLabels[i] - h
        weights = weights + alpha * error * dataMatrix[i]
    return weights
```

通过结果可以看到此时的分类效果较差，通过绘制系数变化情况发现系数波动性较大，原因是存在一些不能正确分类的样本点，每次迭代时引发剧烈的变化。因此接下来产生了改进的随机梯度上升法。

➤ `stocGradAscent1` 函数

改进的随机梯度上升法步长 `alpha` 在每次迭代都会作调整，`alpha` 会随着迭代次数的上升而不断减小，但不会减小到 0，这样保证了多次迭代后新数据仍然具有一定的影响。

此外 α 每次减小 $\frac{1}{j+i}$, 其中 j 是迭代次数, i 是样本下标, 这样避免了 α 严格下降。

且改进的随机梯度上升算法每次回随机选取样本来更新回归系数, 减少了一定的周期性波动。

综上改进的梯度上升算法较未改进前收敛速度更快、没有出现周期性波动。

```
def stocGradAscent1(dataMatrix, classLabels, numIter=150):#改进了的随机梯度上升
    m,n = np.shape(dataMatrix)
    weights = np.ones(n)    #initialize to all ones
    for j in range(numIter):
        dataIndex = list(range(m))#需要加一个 list()保证不报错
        for i in range(m):
            alpha = 4/(1.0+j+i)+0.0001    #alpha decreases with iteration, does not
            randIndex = int(np.random.uniform(0,len(dataIndex)))#go to 0 because of the constant
            h = sigmoid(sum(dataMatrix[randIndex]*weights))
            error = classLabels[randIndex] - h
            weights = weights + alpha * error * dataMatrix[randIndex]
            del(dataIndex[randIndex])
    return weights
```

➤ classifyVector 函数

此函数用于按照 sigmoid 函数的分类规则进行分类。

```
def classifyVector(inX, weights):
    prob = sigmoid(sum(inX*weights))
    if prob > 0.5: return 1.0
    else: return 0.0
```

● 主体部分

介绍完成各提前定义函数后, 进入用 Logistic 回归预测疝病马存活部分。

在 colicTest 函数部分, 首先打开训练文件, 将特征与标签分开, 利用 stocGradAscent1 的改进的随机梯度上升算法进行运算, 接下来对比分类结果与标签结果测试准确率, 并将准确率结果返回。

在 multiTest 函数中调用 colicTest 函数, 进行多次 Logistic 回归, 并返回 10 次的平均结果。

其代码如下:

```
def colicTest():
    frTrain = open('horseColicTraining.txt'); frTest = open('horseColicTest.txt')
    trainingSet = []; trainingLabels = []
    for line in frTrain.readlines():
        currLine = line.strip().split('\t')
        lineArr = []
        for i in range(21):
            lineArr.append(float(currLine[i]))
        trainingSet.append(lineArr)
        trainingLabels.append(float(currLine[21]))
    trainWeights = stocGradAscent1(np.array(trainingSet), trainingLabels, 1000)
    errorCount = 0; numTestVec = 0.0
    for line in frTest.readlines():
        numTestVec += 1.0
        currLine = line.strip().split('\t')
        lineArr = []
        for i in range(21):
```

```

        lineArr.append(float(currLine[i]))
        if int(classifyVector(np.array(lineArr), trainWeights))!= int(currLine[21]):
            errorCount += 1
    errorRate = (float(errorCount)/numTestVec)
    print ("the error rate of this test is: %f" % errorRate)
    return errorRate

def multiTest():
    numTests = 10; errorSum=0.0
    for k in range(numTests):
        errorSum += colicTest()
    print ("after %d iterations the average error rate is: %f" % (numTests, errorSum/float(numTests)))

```

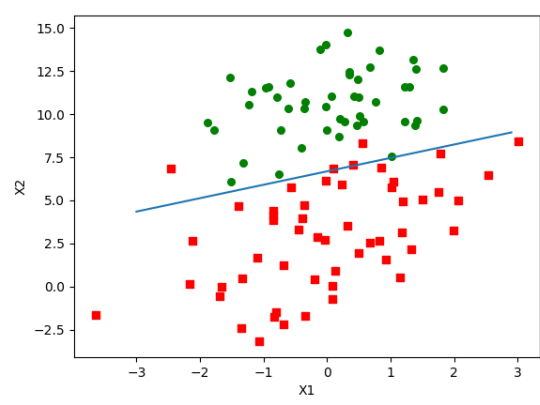
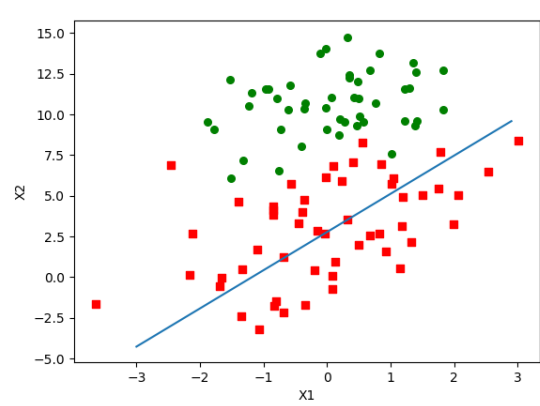
四、模型结果

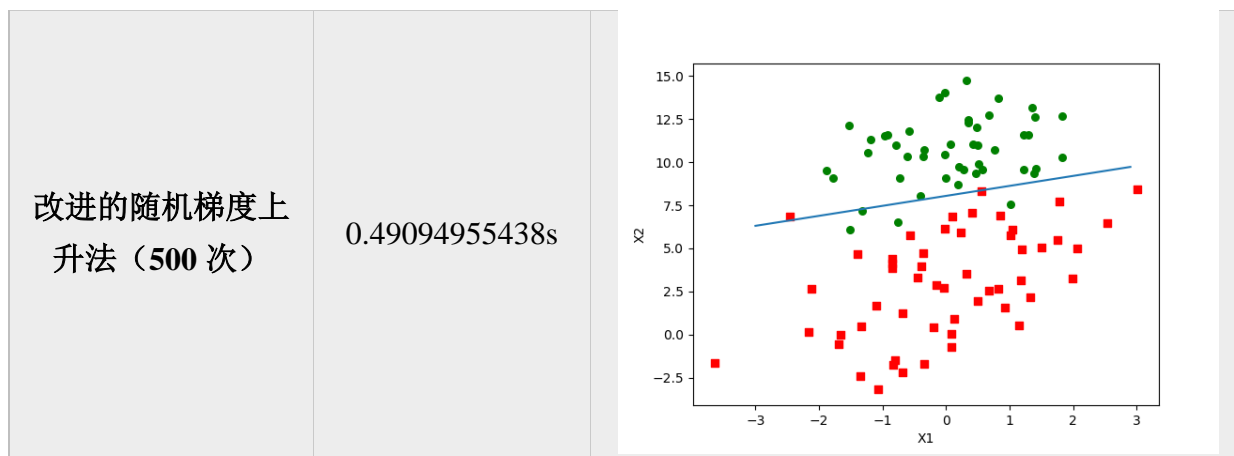
4.1 测试数据结果

程序清单 5-1 至 5-4 都使用了 testSet.txt 进行了验证，下面是采用 3 种方法进行 Logistic 回归得到的运行时间与划分结果。

结果表明，在运行速度方面：随机梯度上升法>改进的随机梯度上升法>梯度上升法，在准确率方面：改进的随机梯度上升法≈梯度上升法>随机梯度上升法。

表 1 各方法时间、结果比较

方法名称	运行时间	划分结果
梯度上升法	0.01563817s	
随机梯度上升法	0.000737848s	



4.2 从疝气病症预测病马的死亡率结果

执行“三”中代码可以用于预测疝病马死亡状况，由运行结果可见，利用 Logistic 回归的方法进行分类，错误率在 30%-40%之间。

```
In [24]: multiTest()
__main__:14: RuntimeWarning: overflow encountered in exp
the error rate of this test is: 0.328358
the error rate of this test is: 0.328358
the error rate of this test is: 0.313433
the error rate of this test is: 0.343284
the error rate of this test is: 0.313433
the error rate of this test is: 0.373134
the error rate of this test is: 0.417910
the error rate of this test is: 0.373134
the error rate of this test is: 0.328358
the error rate of this test is: 0.328358
after 10 iterations the average error rate is: 0.344776
```

图 1 Logistic 分类结果

五、优缺点分析与心得体会

本次课程作业，使用 Logistic 回归进行分类问题，Logistic 回归的计算代价不高，是很常用的分类算法。基于随机梯度上升的 Logistic 回归分类器能够支持在线学习。但 Logistic 回归算法缺点很明显，其一般只能解决两个类的分类问题。此外，Logistic 回归容易欠拟合，导致分类的精度不高。由疝气病马分类结果来看，Logistic 回归分类准确率差强人意。