分类问题（logistic regression）

算法.小强

1、 数据格式

（1） 数据

b,30.83,0,u,g,w,v,1.25,t,t,01,f,g,00202,0,+

b,32.33,7.5,u,g,e,bb,1.585,t,f,0,t,s,00420,0,-

（2）属性格式

A1: b, a. A2: continuous. A3: continuous. A4: u, y, l, t. A5: g, p, gg. A6: c, d, cc, i, j, k, m, r, q, w, x, e, aa, ff.

A7: v, h, bb, j, n, z, dd, ff, o. A8: continuous. A9: t, f. A10: t, f. A11: continuous. A12: t, f. A13: g, p, s.

A14: continuous. A15: continuous. A16: +,- (class attribute)

（2） 数据分析

数据维度为 16，其中最后一维是分类值，取值“+”或者“-”。可以看到我们的任务是二分类问题，使用 logistic regression（简称 LR）进行分类是可以较方便进行的。对于 LR 算法，我们一般将类别设置为 0 或者 1，因此在这里规定“+”为 1，“-”为 0. LR 算法是一种有监督学习算法，通常的输入数据为（X,Y）形式，X 是特征向量，Y 是分类类别。在这里第 16 维数据为 Y，而前 15 维，作为 LR 模型的输入向量，即 X 输入向量。经过统计，可以看到该数据中类别为 1 与 0 的比例为 307：383，数据上相差不大，因此不需要太考虑非平衡问题。

数据中格式有很多种，字符型（a、b），连续型数值（很大或者很小的数值），还有 00202 这类数值等，首先需要将字符型格式转换为数值型，以利于计算。但是因为不知道每一维数据所表示的含义，这存在一定的困难。因此只能靠观察数据形式，可以看到第一维只有 a 或者 b 两类值（确实值为？，共 12 个），比例为 210:468, 直觉上可能是性别（男女），因此可以设置为 0 或者 1，这里规定 a 为 0，b 为 1。对于类似第二维数值型的我们先不处理。类似第 4、5、6、7、13 维数据，并不清楚表征的意思，但猜测是一些选项，因此只需要能设值将这些字符型表征为离散数值即可（这也是初步的方法）。在这里第四维数据表示为 u、y、l、t 分别用 1，2，3，4 表示。第五维用 g、p、gg 使用 1、2、3 表示。第六维使用 1, 2,3,4,5, 6,7,8,9,10,11,12,13,14 分别表示 ff,d,j,k,i,aa,m,c,w, e, q, r,cc, x。第七维使用 1, 2,3, 4,5,6,7,8,9 分别表示 ff,dd,j,bb,v,n,o,h,z。而对于第 9、10、12 维数据，可以看到只有 t、f 两种值，结合当前数据为信用卡业务，初步判断为 t 代表 true，而 f 代表 false，因此可以设置为 t 为 1，而 f 为 0. 第十三维使用 1、2、3 表示 g、p、s。对于缺失值，我们先暂不做处理，按以上方式先处理原数据。处理后得到 fcrx.data 文件。

（3）数据是存在缺失值，由于包含缺失信息的条目数据只有 37 条，不是很多，在这里我们选择 listwise deletion 的处理方式（当然还有其他一些补足缺失信息的方法，可以尝试一下），也就是删除存在 missing data 的数据行，只保留有完整数据的数据行。处理后得到 cfcrx.data 数据文件。

(4)处理完数据之后，我们会发现，各个维度的数值偏差很大，比如第 15 维上的数最大可以达到几万，而其他数一般就在 1，相差非常悬殊。这可能是由于量纲引起，因此需要做无量纲处理。无量纲或者标准化处理是将数据按比例缩放，使之落入一个小的特定区间，去除单位限制，使得不同单位或者量级的指标能够进行比较和加权。在这里，采用比较简单的离差标准化，对第 1-15 维分别做标准化处理。得到文件 ncfcrx.data.

2、 算法实施

（1） 划分训练集、测试集

这里随机选择 90%的数据作为训练集，10%的数据作为测试集。为了平衡起见，我们在类别为 1 以及 0 的数据中分别选择 90%作为训练集，余下的 10%数据作为测试集。得到 train.data, test.data

(2)算法训练及测试

使用的是 python sklearn 进行 logistic regression 进行测试，准确率 precision 可以达到 92.4%。

Python 代码：

```
import numpy as np
from sklearn import linear_model

def interlist(a, b):
    count=0
    for i in range(len(a)):
        if a[i] == b[i]:
            count = count+1
    return count

filepath = 'E:/MF/mltask'
if(os.path.exists(filepath)):
    train = np.loadtxt(filepath+'/train.data')
    X_train = train[:,0:15]
    Y_train = train[:,15]
    test = np.loadtxt(filepath+'/test.data')
    X_test = test[:,0:15]
    Y_test = test[:,15]
    classifier = linear_model.LogisticRegression(C=1e6)
    probas = classifier.fit(X_train, Y_train)
    #probas_test = probas.predict_proba(X_test)
    preClass = probas.predict(X_test);
    precision = interlist(preClass,Y_test)*1.0/len(Y_test)
    print precision
```