華東理工大學

模式识别大作业

题　　目　预测收入高低

学　　院　信息科学与工程

专　　业　信息与通信工程

组　　员　陈志刚

指导教师　赵海涛

完成日期：　2019 年 11 月 20 日

模式识别作业报告——预测收入高低

组员：陈志刚

经过半个学期的模式识别原理与应用课程的学习，我了解到了模式识别的一些常用方法，对这些方法的原理有了一定的掌握。模式识别是一门实践性很强的课程，仅仅学习理论知识是不够的，只有把知识熟练运用到实践当中，获得一定的结果，才能算是真正掌握了这门课程，所以赵海涛老师布置的这个大作业能够很好的锻炼我们理论运用于实际的能力。

经过网上查找相关题目及数据集，分析数据，我最终确定编写一个预测收入高低的模型，准备采用的模型是逻辑回归模型，下面将详细说明我的解决过程。

一、预测收入数据集介绍

我选择的题目来自 kaggle InClass Prediction Competition，名称是 Rebuild MIS583，它是中山大学资管硕班深度学习的作业竞赛，该竞赛的目标是通过已给的数据集，构建机器学习模型来预测工资收入的高低，数据集中将收入情况分为了两类，rich(>50K) and poor(<=50K)。该数据集包含 14 个特征以及收入>50K 或<=50K 的标签，其中部分数据如表 1 所示。

表 1 数据集示例

age	39	50	38
workclass	State-gov	Self-emp-not-inc	Private
fnlwgt	77516	83311	215646
education	Bachelors	Bachelors	HS-grad
education_num	13	13	9
marital_status	Never-married	Married-civ-spouse	Divorced
occupation	Adm-clerical	Exec-managerial	Handlers-cleaners
relationship	Not-in-family	Husband	Not-in-family
race	White	White	White
sex	Male	Male	Male
capital_gain	2174	0	0
capital_loss	0	0	0
hours_per_week	40	13	40
native_country	United-States	United-States	United-States
income	<=50K	<=50K	<=50K

从表中可以看出该数据集包括年龄、工作类别、教育程度、婚姻状况、职位、种族、社会关系、性别、每周工作时间、祖国、等属性，共 8 个离散属性、6 个连续属性，最后是收入水平，有<=50K 和>50K 两类。

该竞赛提供的数据均为 csv 文件，为了方便构建模型，它已经将原始数据进行了 one-hot 编码，方便输入模型进行训练，包括 X_train.csv(训练集)、Y_train.csv(训练集标签)和 X_test.csv(测试集)三个文件。其中训练集有 32561 个样本，测试集有 16281 个样本。

本次实验主要是利用 X_train.csv 和 Y_train.csv 对构建的预测收入高低模型进行训练，训练完成后，对 X_test.csv 中的数据进行预测，结果保存为 submission.csv 文件，上传到 kaggle 上和真实样本标签进行对比来评价构建模型的性能。

二、整体解决方案

2.1 方案分析

本次实验所研究的问题是二分类问题，于是我考虑用二分类的机器学习方法来构建模型，了解到二分类通常有以下几种方法：

(1) 贝叶斯分类器：它通过预测一个给定的元组属于一个特定类的概率，来进行分类。朴素贝叶斯分类法假定一个属性值在给定类的影响独立于其他属性的——类条件独立性。

(2) 决策树：是一种简单但广泛使用的分类器，它通过训练数据构建决策树，对未知的数据进行分类。决策树的每个内部节点表示在一个属性上的测试，每个分枝代表该测试的一个输出，而每个树叶结点存放着一个类标号。

(3) 支持向量机：把分类问题转化为寻找分类平面的问题，并通过最大化分类边界点距离分类平面的距离来实现分类。

(4) 逻辑回归：逻辑回归是一种广义线性回归，属于概率型非线性回归，它是研究二分类观察结果与一些影响因素之间关系的一种多变量分析方法。

在模式识别课程中，上述四种方法我们都有学习到。由于本实验的数据集中包含取值连续的属性，而贝叶斯分类器要计算 $P(x|c_k)$ 的值，如果 x 未在训练集中出现，则无法计算，解决办法一个是对连续属性离散化，但对区间划分的粒度不好控制，还有一个是假设服从某种概率分布，但也有可能产生假设错误的情况。所以考虑到这个问题，本实验采用决策树、支持向量机、逻辑回归分别对数据进行建模。

2.2 数据读入

本次程序主要用 Python 编写，为进行数据训练及测试，首先要对 csv 数据进行读取。本次实验读取数据使用了 Pandas 模块，将训练集读取为数据帧(Data Frame)，读取的数据部分展示如图 1 所示：

	age	fnlwgt	sex	capital_gain	capital_loss	hours_per_...	Federal-gov	Local-gov	Never-wor...	Private	Self-emp-i...	Self-emp-n...
0	39	77516	1	2174	0	40	0	0	0	0	0	0
1	50	83311	1	0	0	13	0	0	0	0	0	1
2	38	215646	1	0	0	40	0	0	0	1	0	0
3	53	234721	1	0	0	40	0	0	0	1	0	0
4	28	338409	0	0	0	40	0	0	0	1	0	0
5	37	284582	0	0	0	40	0	0	0	1	0	0
6	49	160187	0	0	0	16	0	0	0	1	0	0
7	52	209642	1	0	0	45	0	0	0	0	0	1
8	31	45781	0	14084	0	50	0	0	0	1	0	0
9	42	159449	1	5178	0	40	0	0	0	1	0	0
10	37	280464	1	0	0	80	0	0	0	1	0	0
11	30	141297	1	0	0	40	0	0	0	0	0	0
12	23	122272	0	0	0	30	0	0	0	1	0	0
13	32	205019	1	0	0	50	0	0	0	1	0	0
14	40	121772	1	0	0	40	0	0	0	1	0	0
15	34	245487	1	0	0	45	0	0	0	1	0	0
16	25	176756	1	0	0	35	0	0	0	0	0	1
17	32	186824	1	0	0	40	0	0	0	1	0	0
18	38	28887	1	0	0	50	0	0	0	1	0	0
19	43	292175	0	0	0	45	0	0	0	0	0	1
20	40	193524	1	0	0	60	0	0	0	1	0	0

图 1 读入的原始数据

由于数据帧不便于数据操作，可将其转为数组形式(array)。将输入特征值和标签分别存于 x_data 和 y_data 两个数组中。对于测试数据的读取操作，与训练集相似，但需要存储 ID。数据读入部分 Python 程序如下所示：

代码如下：

```
import pandas as pd
import numpy as np
# 数据读入
x_data = pd.read_csv('D:/研一/模式识别/大作业/数据/X_train.csv')
y_data = pd.read_csv('D:/研一/模式识别/大作业/数据/Y_train.csv')
x_data = np.array(x_data)
y_data = np.array(y_data)
```

2.3 数据预处理

可以发现读入的数据的值过大或过小，不便于输入模型进行训练，于是首先要对数据进行标准化。数据的标准化是将数据按比例缩放，使之落入一个小的特定区间，去除数据的单位限制，将其转化为无量纲的纯数值，便于不同单位或量级的指标能够进行比较和加权。标准化的优点主要是：(1)加快梯度下降求最优解的速度；(2)有可能提高精度。

在标准化之后需要将读入的数据划分成训练集和测试集，训练集用来训练，测试集用来评价构建模型的好坏。整个预处理部分的 Python 程序如下所示：

代码如下：

```
from sklearn import preprocessing
x_data = preprocessing.MaxAbsScaler().fit_transform(x_data)
x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.3, random_state=2)
```

2.4 逻辑回归算法原理及程序实现

Logistic 回归用于解决二分类问题，根据已知的训练集进行模型训练，再用此模型对新的数据进行分类预测，如图 2 所示。

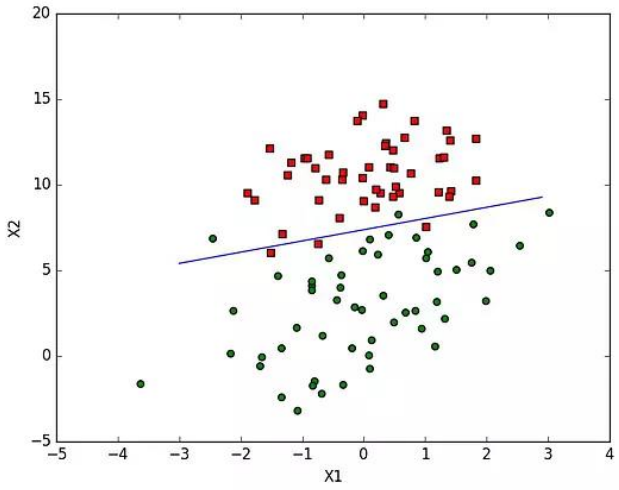

图 2 基于 logistic 回归分类示例

回归问题常见步骤是：寻找 g 函数；构造 J 函数；想办法使得 J 函数最小并求得回归参数。logistic 回归的 g(x)函数为：

$$g(x)=\frac{1}{1+e^{-x}} \tag{2-1}$$

其中，g(x)表示取 1 的概率值。图像如图 3 所示。

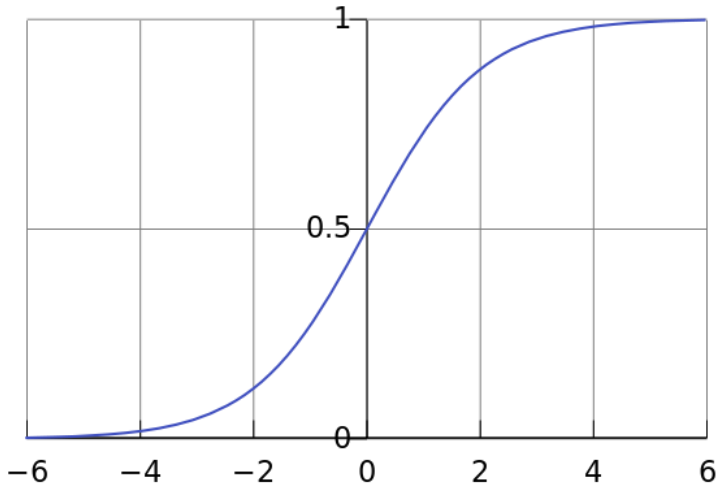

图 3 logistic 函数图像

那么对于输入 x 的分类结果对于类别 1 和类别 0 的概率分别为：

$$p(y=1|x;\omega,b)=g\left(\omega^T x+b\right)=\frac{1}{1+e^{-\left(\omega^T x+b\right)}} \tag{2-2}$$

$$p(y=0|x;\omega,b)=1-g\left(\omega^T x+b\right)=\frac{e^{-\left(\omega^T x+b\right)}}{1+e^{-\left(\omega^T x+b\right)}} \tag{2-3}$$

那么对于构造损失函数 J，它们基于最大似然估计推到得到的：

$$\begin{aligned} L(\theta)&=p(Y|X;\theta)=\prod_{i=1}^{m}p\left(y^{(i)}|x^{(i)};\theta\right)\\ &=\prod_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right)^{y^{(i)}}\right)\left(1-h_\theta\left(x^{(i)}\right)\right)^{1-y^{(i)}}\end{aligned} \tag{2-4}$$

取似然函数：

$$l(\theta)=\sum_{i=1}^{m}\left(y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right)+\left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right)\right) \tag{2-5}$$

通过梯度下降求解梯度最小值：

$$\frac{\partial J(\theta)}{\partial\theta_j}=\left(y^{(i)}-h_\theta\left(x^{(i)}\right)\right)x^{(i)} \tag{2-6}$$

其中，k 为迭代次数，α 为学习率。

代码如下：

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


def logistic_regression(x, y, iterations, alpha, print_step):
    x = np.append(x, np.ones((x.shape[0], 1)), axis=1)
    w = np.random.random((x.shape[1],1))
    for i in range(iterations):
        z = np.dot(x, w)
        pre = sigmoid(z)
        error = pre - y
        grad = np.dot(x.T, error)
        # print(grad.shape)
        w -= alpha*grad
        if i % print_step == 0:
            e = np.exp(-10)
```

```
        NLL = -np.sum(y*np.log(pre+e) + (1-y)*np.log(1-pre+e))
        acc = np.sum(np.round(pre)==y)/(y.shape[0]*1.0)
        print('iteration{}:NLL loss{}, train accuracy{}'.format(i, NLL, acc))
    return w

def logistic_regression_predict(x, w):
    x = np.append(x, np.ones((x.shape[0], 1)), axis=1)
    z = np.dot(x, w)
    preds = sigmoid(z)
    return np.round(preds)

weights = logistic_regression(x_train, y_train, 70, 0.1, 5)
preds = logistic_regression_predict(x_test, weights)
```

2.5 支持向量机及决策树调包实现

本次实验除了自己编程实现逻辑回归之外，还通过调用 scikit-learn 包中 SVM 函数实现了支持向量机模型构建，调用 DecisionTreeClassifier 函数实现了决策树模型的构建

代码如下：

```
from sklearn.svm import SVC
from sklearn.metrics import classification_report
# 多项式核函数
svc = SVC(kernel='poly', degree=5, gamma=3, coef0=0)
svc.fit(x_train, y_train)
preds = svc.predict(x_test)
print(classification_report(y_test, preds))
```

```
from sklearn.tree import DecisionTreeClassifier
cart = DecisionTreeClassifier(min_samples_leaf=10)
cart.fit(x_train, y_train)
preds = cart.predict(x_test)
```

2.6 评定指标

为了比较不同算法模型的实验结果，本文采用了准确率、F1 度量和受试者工作特征(Receiver Operating Characteristic, ROC)曲线的评价指标来对模型进行评估。

准确率很好理解，即预测正确的样本数除以总的样本数；F1 度量能够同时考虑查准率(Precision, P)与查全率(Recall, R)进行两个指标的综合评判，其值越高，则算法模型的性能越好；ROC 曲线则能反映出算法模型的泛化性能的好坏。查准率、查全率以及 F1 度量的计算公式分别如式(2-7)、(2-8)、(2-9)所示。

$$P=\frac{\text{正确有并发症样本数}}{\text{预测有并发症样本数}} \tag{2-7}$$

$$\mathrm{R}=\frac{\text{正确有并发症样本数}}{\text{实际有并发症样本数}} \tag{2-8}$$

$$\mathrm{F1}=\frac{2\times P\times R}{P+R} \tag{2-9}$$

2.7 调试及预测结果

在最终的调试过程中，我发现逻辑回归模型学习迭代 70 次左右之后，模型拟合最佳，未产生过拟合现象，于是模型迭代 70 次后停止迭代。图 4 给出了训练过程中的训练集损失值以及准确率。

```
iteration0:NLL loss99319.9405381525, train accuracy0.23983073983073983
iteration5:NLL loss70268.9886680574, train accuracy0.7602033852033852
iteration10:NLL loss86303.33919363529, train accuracy0.7029074529074529
iteration15:NLL loss70268.98864872212, train accuracy0.7602033852033852
iteration20:NLL loss70268.98864872212, train accuracy0.7602033852033852
iteration25:NLL loss70225.7873569699, train accuracy0.7603398853398854
iteration30:NLL loss107438.87007923872, train accuracy0.6328828828828829
iteration35:NLL loss50282.13818533836, train accuracy0.8277368277368278
iteration40:NLL loss99227.5353559836, train accuracy0.6604559104559105
iteration45:NLL loss164504.6639367731, train accuracy0.43765356265356264
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: RuntimeWa

iteration50:NLL loss85652.849670441, train accuracy0.7072754572754573
iteration55:NLL loss70116.51552752013, train accuracy0.7607152607152607
iteration60:NLL loss89638.8038657969, train accuracy0.6929429429429429
iteration65:NLL loss70082.80978243634, train accuracy0.7607835107835108
iteration70:NLL loss54347.75589401688, train accuracy0.8136431886431886
```

图 4 训练过程输出

将训练好的三种模型在测试集上进行测试，表 2.1 分别给出了逻辑回归模型、SVM 模型和决策树模型在测试集上的准确率以及 F1 度量的值。

表 2.1 逻辑回归模型、SVM 模型以及决策树模型的准确率及 F1 度量

模型	准确率	F1 度量
逻辑回归模型	0.8324	0.89
SVM 模型	0.8532	0.91
决策树模型	0.8532	0.9

由于只有逻辑回归模型的输出是类别的概率值，根据 ROC 曲线的定义，可以画出逻辑回归模型预测值的 ROC 曲线，如图 5 所示。

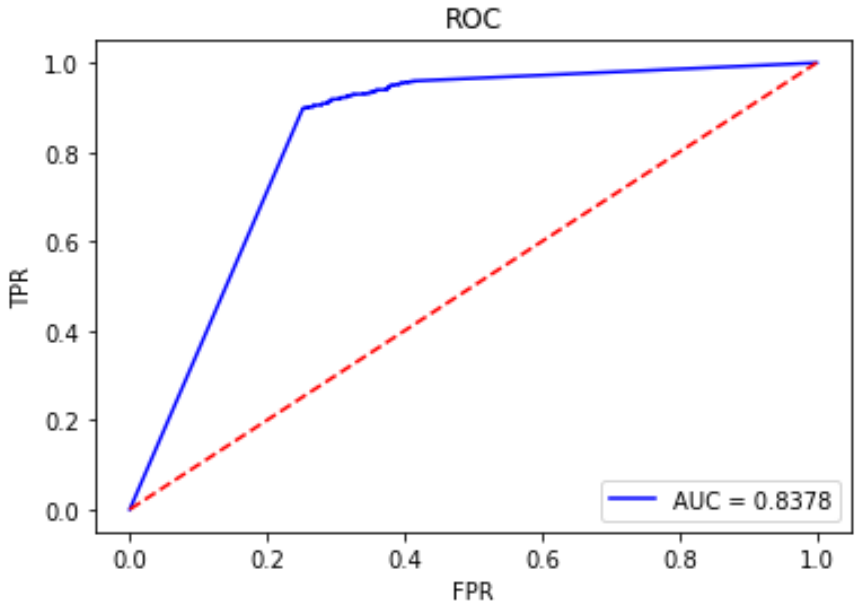

图 5 逻辑回归模型的 ROC 曲线

同时由于本实验题目及数据集来自与 kaggle，于是我对 X_test.csv 测试集进行预测，并将预测结果保存为 submission.csv 上传到 kaggle 上，图 6 给出了 kaggle 上给出的 Score。其中第二行的提交为逻辑回归模型的预测结果，第一行的提交结果为我使用 XGBoost 构建决策树模型预测的结果(该结果是本次实验所有结果里面最好的)。

Submission and Description	Private Score	Public Score
submission.csv 19 hours ago by Zhigang Chen add submission details	0.87838	0.86609
submission.csv a day ago by Zhigang Chen add submission details	0.79749	0.79258

三、小组分工

程序设计及编写：陈志刚

程序调试：陈志刚

实验报告：陈志刚

四、作业总结

通过这次大作业的编程实践，我深刻体会到学懂一个知识点与会应用一个知识点还是有很大的区别，在课堂上某种算法的原理、推导可能学得很扎实，但真正将这个算法运用到实际当中又可能出现诸多问题，编程能力是一方面因素，但并不是全部。这次作业中，我在数据预处理部分就遇到了一些问题，首先要处理数据缺失值的问题，然后数据归一化最开始我也没有考虑在内，导致后面出现了一系列的问题。在模型构建过程中，一些矩阵运算的编程实现以及维数的选取等问题相继出现，还好网络资源丰富，通过网上查找相关解决办法顺利完成了模型构建。模型训练过程中，学习率、迭代次数的选取又让我摸不着头脑，通过逐渐尝试，慢慢找到了较为合适的取值。最终完成了逻辑回归模型的构建。此外我还通过调用函数包的方式实践了 SVM 模型、决策树模型的构建。

这次的大作业除了让我对课上所学内容进行巩固复习之外，更重要的是锻炼培养了我的动手实践能力，让我有能力将一个理论层面的东西应用到实际当中去，此外也提升了我的编程能力，让我补足了先前缺漏的一些编程方面的基础知识。总之，这次大作业收获很多，之后有时间我也会动手实践其它学习到的方法，以进一步加强对各个知识点的理解程度。

附：文件说明

本次附件一共包含有：

1 大作业报告；

2 最终的 Python 实现程序源码：poor_and_rich.ipynb

3 导出了预测数据集：predictions.csv