華東理工大學

模式识别大作业

题　　目　运输任务的行驶时间预测

学　　院　信息科学与工程

专　　业　控制科学与工程

姓　　名　唐　山

学　　号　Y30190746

指导教师　赵海涛

完成日期：　**2019** 年 **12** 月 **8** 日

模式识别作业报告——运输任务的时间预测

姓名：唐山　学号：Y30190746

经过半个学期的学习，我在课堂上学到了很多，尽管在上模式识别之前，对机器学习有了大概的认识，但是在赵海涛老师的指导下，对很多学习算法有了狠了更深的了解。但是理论终究还是理论，距离真正运用这些算法还有很长的路要走。尤其是像我这样编程基础几乎为零的新手来说，掌握这些算发展更是需要花很长的时间，因此赵海涛老师布置的大作业对我个人而言虽说有难度，但也是锻炼自己的一种好方法。

在看过 Lintcode 以及 Kaggle 上的竞赛题后，我深知自己的基础不足以完成这些项目，因此在网上浏览许久后，最终确定做一个货车运输任务的时间预测。

一、运输任务的时间预测简介

Butler 汽运公司，位于美国加洲，该公司的主要业务是在当地运送货物。为了更好地制定工作计划，Butler 汽运公司打算评估公司每个运送任务，如果能把运送时间和影响因素构建模型，那就能预测新的任务所需要的时间，据此更好的安排运送任务，增加人员工作效率，提高公司效益。

我们从公司多年的经营经验可以得知，运送任务的里程、其中分送货物的次数和运送时间都有着密切的关系。所以我们将运送时间作为因变量，运送里程、分送次数作为自变量，建立多元回归模型。

二、整体解决方案

通过多元线性分析，将将运送时间作为因变量，运送里程、分送次数作为自变量，与平常的分类问题不同。分析结果并非是 0-1 两种结果。而是有点类似辨识参数的过程。

运输时间的预测是在其以往历史中，选出部分样本，通过训练模型，得到一个表现较为良好的最终模型，因而可以以此作为一个基准，安排运送任务。

我们通过将数据分为训练集以及测试集，利用训练集建立模型后，在训练数据上反复调试模型中的部分参数，使得模型对于训练数据达到较好的效果后，将测试集的数据放入模型中，将模型预测数据与实际数据相比较。从而来对模型做出评价。

2.1 数据结构分析

首先观察原始数据的结构，原始数据的分类如表 1 所示。

表 1 原始数据结果

运送任务编号	行驶里程	分送次数	行驶时间
1	100	4	9.3
2	50	3	4.8
3	100	4	8.9
4	100	2	6.5

表中的数据只是截取了原始数据中的一部分。

第一列数据表示的是数据的编号，这列数据对算法求解没有任何作用，因此在读取数据时，选择不读取这一列的数据。

第二列数据表头为行驶里程，是货车单次行驶时的路程。

第三列数据的表头为分送次数，即货车运输任务需要完成几次。

第四列数据的表头为行驶时间，即货车完成运输任务需要耗费的时间，这也是构建模型需要达到的目的。

目的暂时只考虑运输路程以及运输次数的因素，并在所有的数据中选择二八分的形式，即训练集为数据的百分之八十，测试集为数据的百分之二十。将数据整理好后，放在 CSV 文件中。分为命名为 test.csv 和 train.csv。

2.2 数据读入

数据分为训练数据和测试数据，训练数据为 train.csv，测试数据为 test.csv。在 Python 中利用 numpy 的 genfromtxt 方法打开训练数据。同时将特征与标签分别放在 x_data 和 y_data 中。最终得到数据如图：

```
[[  nan   nan   nan   nan]
 [  1.  100.    4.    9.3]
 [  2.   50.    3.    4.8]
 [  3.  100.    4.    8.9]
 [  4.  100.    2.    6.5]
 [  5.   50.    2.    4.2]
 [  6.   80.    2.    6.2]
 [  7.   75.    3.    7.4]
 [  8.   65.    4.    6. ]
 [  9.   90.    3.    7.6]
 [ 10.   90.    2.    6.1]
 [ 11.   50.    5.    7. ]
 [ 12.   45.    6.    7. ]
 [ 13.   85.    5.    9.6]
 [ 14.   65.    2.    6.5]
 [ 15.   40.    6.    7.2]
 [ 16.   50.    3.    6. ]
 [ 17.   75.    5.    9.9]
 [ 18.   75.    4.    8.6]
 [ 19.   65.    3.    4.9]
 [ 20.   90.    3.    7.5]]
```

图 1 原始数据的读入

由图 1 可以看出，在数据第一行会出现 nan 的数据，nan 的产生是因为数据表格中出现了非数字的数据类型，导致数据读取为 nan。同时第一列将运输任务编号也被读取进来，为了保证这些数据对后续模型的训练不产生影响，因此在训练之前需要将数据进行处理。使得程序能够正常运行。

2.3 数据处理

将数据中的特征以及标签分别给 x_data 和 y_data。利用切片形式，将数据中的运输任务编号以及表头都去掉。只留下有用的特征以及标签。

代码如下：

```
data = np.genfromtxt("C:\\Users\\ASUS\\Desktop\\data\\duoyuan\\train.csv",
delimiter=',')
x_data = data[1:, 1:-1]
y_data = data[1:, -1]
```

2.4 训练数据建立

由于采用的 numpy 对数据进行的读取，因为训练数据通过对数据的切片，所有的特征都存在于一个 ndarray 的数组中。即数组的每一个行都对应一个任务编号，而每一列都对应一个特征。此外每一个任务编号都对应一个索引，通过这个索引能够在 y_data 中找到对应的标签，即运输耗费的时间。

2.5 线性回归模型的建立

在获得了上述训练数据的基础上，我们的模型采用最小二乘法来进行拟合，最小二乘法（又称最小平方法）是一种数学优化技术。它通过最小化误差的平方和寻找数据的最佳函数匹配。利用最小二乘法可以简便地求得未知的数据，并使得这些求得的数据与实际数据之间误差的平方和为最小。

我们利用最小二乘法来拟合训练数据集中的数据构造了自变量$\mathrm{X}=\{1, x_{i1}, x_{i2}\}$和因变量$\mathrm{Y}=\{y\}$，得到了一组参数$\theta=\{\theta 0, \theta 1, \theta 2\}$。然后就可以用这组参数去估计预测数据的运输时间，即$Y_{pred}=X_{test}\theta$。

最小二乘表达式为：

$$Z=\frac{1}{2n}\sum_{i=0}^{n}(Yi-(\theta 0+Xi1*\ \theta 1+Xi2*\ \theta 2))^{2} \qquad (1)$$

通过梯度下降来不断的优化θ各项参数的值，最后便可通过模型来对不同的运输任务来进行时间预测。

2.6 调试及预测

调试过程即为对模型参数的不断优化，由于是通过梯度下降来进行参数的优化，因而需要对每一个θ参数的梯度进行计算。

根据式子$Y_{pred} = X_{test}\theta$，将其展开可有：

$$Y = \theta 0 + Xi1 * \theta 1 + Xi2 * \theta 2 \qquad (2)$$

其中：

Xi1，Xi2 分别代表数据的两个特征，$\theta_{0,1,2}$ 为权重。

而每项参数的梯度分别为：

$$\frac{dZ}{D\theta 0} = \frac{1}{n}\sum_{i=0}^{n}((\theta 0 + Xi1 * \theta 1 + Xi2 * \theta 2) - Yi) \qquad (3)$$

$$\frac{dZ}{D\theta 1} = \frac{1}{n}\sum_{i=0}^{n}((\theta 0 + Xi1 * \theta 1 + Xi2 * \theta 2) - Yi) * Xi1 \qquad (4)$$

$$\frac{dZ}{D\theta 2} = \frac{1}{n}\sum_{i=0}^{n}((\theta 0 + Xi1 * \theta 1 + Xi2 * \theta 2) - Yi) * Xi2 \qquad (5)$$

将 θ 以其负梯度方形进行更新，更新方式为：

$$\theta = \theta - lr * \frac{dZ}{D\theta} \qquad (6)$$

通过训练最终可以得θ的值，分别为：

$$\theta 0 = 0.024258103，\theta 1 = 0.08311285，\theta 2 = 0.2423386$$

其训练结果如图 2：

```
theta0 = 0.024258103020545783 , theta1 = 0.08311285692081907 , theta2 = 0.24233863868739758,
```

图 2 权重 θ 的参数

为了得到预测结果，首先要建立起测试数据表，测试数据表需要与训练数据表相对应的进行整合以及排序，因为此实验的测试集来自于数据集，因此可以将测试集中的数据放进模型，可将模型预测的结果与真实值进行对比。而对测试集的数据处理与训练集的样本处理一致，将测试集的特征和标签分别放在 x_test，y_test 中。预测结果如图 3：

```
predict is  [8.16231815 9.37401134 4.42223959 7.40030257 7.9890925  7.40030257
 6.98473829]
```

图 3 测试集的预测结果。

将测试集的数据以及预测结果放在下列表格中（表 2）

表 2 测试集的数据和预测结果

运输任务编号 i	行驶里程（英里）	分送次数	行驶时间	预测结果	误差（%）
21	95	1	7.2	8.16231815	13.36
22	95	6	9.9	9.37401134	5.31
23	50	1	5.6	4.42223959	21.03
24	80	3	7.5	7.40030257	1.32
25	90	2	7.8	7.9890925	2.42
26	80	3	7.2	7.40030257	2.781
27	75	3	7	6.98473829	0.218

将模型以及测试样本在空间中描述出来，如图 4：

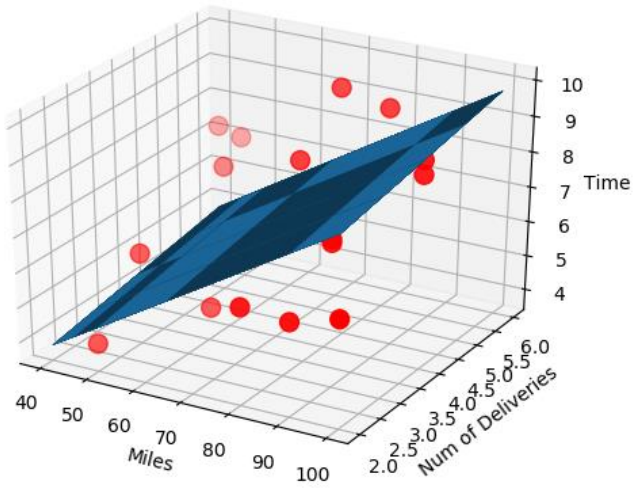

图 4 测试集的预测结果。

三、作业总结

因为机器学习方法掌握的并不多，因此选择了一个基础性的项目，但其实用性比较强。在刚接触到这个项目时，也还是有些无从下手，一是对数据的处理方式不太懂，二是对编程有些难以下手。但是经过一步一步的尝试后，最终将模型训练出来了。而且根据最后的训练结果，能够看出大部分预测结果的误差都在百

分之十之内。说明算法还是有一定效果的。因此在对于一些金融预测还有房价预测中，可以采用这种方法去进行简单的预测。而对于一些分类问题，也可以加入一个激活函数后，也能实现分类功能。

但是也存在有些样本的误差太大，在某些风险代价大的项目中，这种算法便不适用。可能是由于算法比较简单，并且考虑到的因素太少。

四、其他发现和展望

4.1 其他发现

此项目用到的多元分析方法是在已知有用特征的情况下，采用的对每个特征不同的权重来预测运输任务的耗费时间。一旦受到干扰，这个模型便无法有效的做出预测。此外一旦特征值变得更多，便无法对特征值做一个比较正确的处理，即可能会对一些无用特征赋予权重，导致过拟合的现象。

4.2 展望

在进行这个算法前，可以对数据进行预处理。将有用特征保留下来，再用这个算法，应该能得到更加准确的效果。

由于此项目的特征值少，因此并没有对数据进行预处理，如果以后有较为复杂的数据集进行实验，可以尝试将预处理放到算法里面。

五、附录

多元分析的 python 代码为：

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn import linear_model


data =
np.genfromtxt("C:\\Users\\ASUS\\Desktop\\data\\duoyuan\\train.csv",
delimiter=',')
x_data = data[1:, 1:-1]
y_data = data[1:, -1]
print(data)
test =
np.genfromtxt("C:\\Users\\ASUS\\Desktop\\data\\duoyuan\\test.csv",
delimiter=',')
print(test)
x_test = test[1:, 1:-1]
y_test = test[1:, -1]
print(x_test)

lr = 0.0001
theta0 = 0
theta1 = 0
theta2 = 0
epochs = 1000


def computer_error(theta0 , theta1 , theta2 , x_data , y_data):
    total_Error = 0
    for i in range(len(x_data)):
        total_Error += (y_data[i] - theta1 * x_data[i,0] - theta2 *
x_data[i,1] - theta0) ** 2
    return total_Error/float(len(x_data))


def gradient_discent_runner(x_data,y_data ,lr, theta0,
theta1,theta2 ,epochs):
    m = float(len(x_data))
    #print("1111111111111111",theta0)
    for i in range(0, epochs):
```

```
        theta0_grad = 0
        theta1_grad = 0
        theta2_grad = 0
        for j in range(0, len(x_data)):
            theta0_grad += (1/m) * ((theta0 + theta1 * x_data[j, 0] +
theta2 * x_data[j, 1]) - y_data[j])
            theta1_grad += (1/m) * ((theta0 + theta1 * x_data[j, 0] +
theta2 * x_data[j, 1]) - y_data[j]) * x_data[j, 0]
            theta2_grad += (1/m) * ((theta0 + theta1 * x_data[j, 0] +
theta2 * x_data[j, 1]) - y_data[j]) * x_data[j, 1]
        #print("0000000000000000",theta0_grad)
        theta0 = theta0 - (lr * theta0_grad)
        theta1 = theta1 - (lr * theta1_grad)
        theta2 = theta2 - (lr * theta2_grad)
    #print("theta012,,,====", theta0, theta1, theta2)
    return theta0, theta1, theta2


print("theta0 = {0} , theta1 = {1} , theta2 = {2},
computer_error={3}".format(theta0 , theta1 , theta2,
computer_error(theta0 , theta1 , theta2 , x_data , y_data)))
print("Running....")
theta0, theta1, theta2 = gradient_discent_runner(x_data , y_data , lr ,
theta0 , theta1 , theta2 ,epochs)


print("After Running theta0 = {0} , theta1 = {1} , theta2 = {2},
computer_error={3}".format(theta0 , theta1 , theta2,
computer_error(theta0 , theta1 , theta2 , x_data , y_data)))

ax = plt.figure().add_subplot(111, projection='3d')
ax.scatter(x_data[:, 0], x_data[:, 1], y_data,c='r',marker='o',s=100) #
点为红色三角形
x0 = x_data[:, 0]
x1 = x_data[:, 1]
x0, x1 = np.meshgrid(x0, x1)
#plt.scatter(x0,x1)
#plt.show()
x_test1 = x_test[:, 0]
x_test2 = x_test[:, 1]
z = theta0 + theta1 * x0 + theta2 * x1
c = theta0 + theta1 * x_test1 + theta2 * x_test2
print(z)
print("predict is ",c)
```

```
ax.plot_surface(X=x0,Y=x1,Z=z)
ax.set_xlabel('Miles')
ax.set_ylabel('Num of Deliveries')
ax.set_zlabel('Time')
plt.show()

model = linear_model.LinearRegression()
model.fit(x_data,y_data)
x_predict = model.predict(x_test)
print(model.score(x_test,y_test))
print(x_predict)
```