基于决策树算法的用户行为模型研究

刘天祺 (学号: 1320151097)

(北京理工大学 计算机学院, 北京 100081)

Research on the Model of User Behavior Based on Decision Tree Algorithm

Tianqi Liu

(School of Computer Science, Beijing Institute of Technology, Beijing 100081, China)

Abstract: Sensors in mobile phones can collect a large amount of user information every day. Application software can predict user behavior based on this information to facilitate the user's life. In this experiment, the real data collected by the sensors and their labels are supplied for supervised learning to predict the labels corresponding to the data without labels. This experiment uses decision tree learning algorithm to predict the label.

Key words: data mining; sensor; supervised learning; decision tree

摘 要: 手机中的传感器每天可采集大量用户信息, 供应用软件预测用户行为, 为用户生活提供便利。本实验中通过对一批传感器采集的真实数据和它们的标签进行监督式学习, 从而预测没有标签的数据所对应的标签。本实验采用决策树的学习算法。

关键词: 数据挖掘; 传感器; 监督式学习; 决策树

中图法分类号: TP301 **文献标识码**: A

1 引言

本文实验中，从不同用户身上采集数据集，其中包括特征文件(feature file) 和标签文件(label file)。从中挑选一部分数据集，仅保留其特征文件，作为测试集(test set)，剩余部分数据集作为训练集(training set)，数据格式如图 1-1 所示。

特征文件中有 41 列特征，其中第 1 列是数据的时间戳，第 2 列是心率，第 3-15 列、第 16-28 列、第 29-41 列分别是三组集成传感器采集的数据。每组集成传感器采集的数据分别由 1 列体温数据，3 列 I 3D 型加速度数据，3 列 II 3D 型加速度数据，3 列 3D-gyroscope 型数据和 3 列 3D-magnetometer 型数据组成(共计 13 列)。标签文件的行数与特征文件行数相等，其每一行代表与特征文件中对应行记录相对应的分类标签 ID。标签 ID 为 0 至 24 的整数，分别代表不同的行为，其中 0 意为无行动。

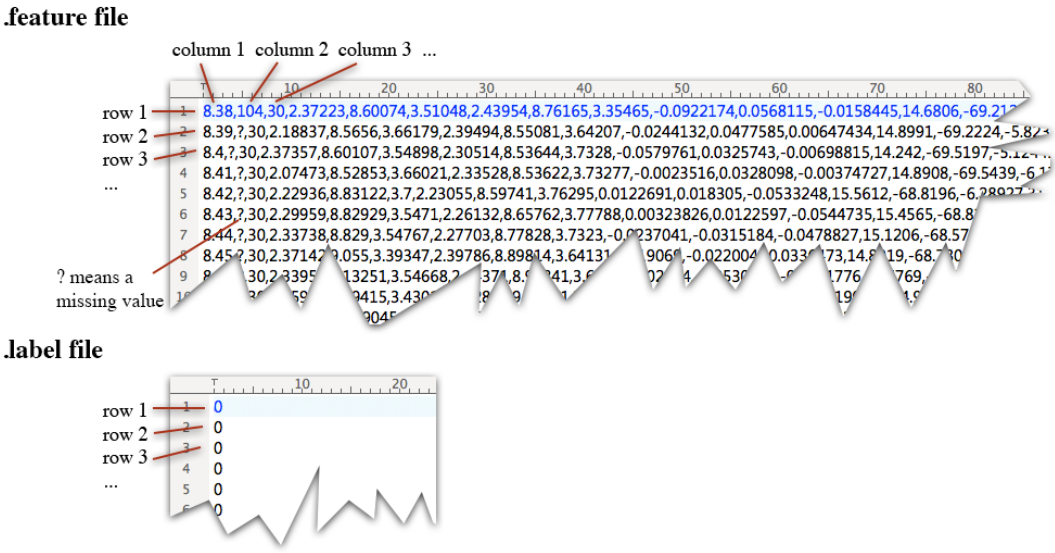

图 1-1 数据格式图

本文实验中，取 5 数据集作为训练集，记为 A,B,C,D,E，取 3 组数据集作为测试集，记为 X,Y,Z。实验目标为从训练集 A,B,C,D,E 中建立起分类模型，用来预测测试集 X,Y,Z 的标签。

本文实验中，由于已知大量特征和标签，故使用决策树学习的监督式学习分类算法，成功预测出 3 组测试集数据的标签，至于预测结果的正确率，还有待老师检查。

本文后续部分组织如下。第 2 节回顾相关工作，第 3 节详细陈述使用的数据挖掘方法，第 4 节报告实验结果，第 5 节对讨论本文的方法并总结全文。

2 相关工作

徐川龙, 顾勤龙, 姚明海[1]在《一种基于三维加速度传感器的人体行为识别方法》一文中，分别使用 SVM 和 BP 神经网络作分类器对传感器采集的加速度数据进行识别，对人体行为识别的识别率达到 98%和 93%。在本文中，使用 DT 作为分类器，对数据特征进行识别。

3 本文的方法

决策树学习算法是应用广泛的一类监督式学习算法,主要用于解决离散的输入与输出映射问题,即对离散函数进行估计。本文实验中，由于传感器采集的第 2 列特征(心率)出现部分数据丢失现象，所以需要先对数据集进行筛选，取出具有完整的 41 列特征的每行数据和其相对应的标签作为筛选后的训练集。训练时，使用 python 库 scikit-learn 中的决策树分类器(DecisionTreeClassifier)，对特征和标签进行函数估计。预测时，输入测试集中的特征数据，由训练出的函数识别特征，得到相对应的标签。

4 实验

4.1 实验设置

4.1.1 实验环境

本文实验环境为 Ubuntu 14.04 (64-bit)，Python 2.7.6。

python 安装 NumPy, scikit-learn 包。

实验中编写的.py 程序，例如：a.py，通过 chmod +x a.py 命令，赋予其执行权限。

4.1.2 筛选数据

编写数据筛选程序，对数据集进行筛选，去除心率特征丢失的数据。程序分为两部分，对于训练集，输入为原始的特征文件和对应的标签文件，输出为筛选后的特征文件和对应的标签文件；对于测试集，输入为原始的特征文件，输出为筛选后的特征文件。

筛选训练集数据的程序如下：

```
#!/usr/bin/python
#for A,B,C,D,E
from sys import argv
__author__ = 'TaQini'

if len(argv) != 3:
    print 'usage: '
    print ' $ ./sample_with_label.py <feature_file> <label_file>'
    print 'example: '
    print ' $ ./sample_with_label.py A.feature A.label'
    exit(0)

feature_file  = open(argv[1],'ro')
label_file = open(argv[2],'ro')
feature_sample_file = open(argv[1]+'.sample','w+')
label_sample_file = open(argv[2]+'.sample','w+')

# read data
fdata = feature_file.readlines()
ldata = label_file.readlines()
feature_file.close()
label_file.close()

# del all item having '?'
feature = []
label = []
for item in range(len(fdata)):
    if '?' not in fdata[item]:
        feature.append(fdata[item])
        label.append(ldata[item])

feature_sample_file.writelines(feature)
feature_sample_file.close()

label_sample_file.writelines(label)
label_sample_file.close()
```

筛选测试集数据的程序如下：

```
#!/usr/bin/python
#for X,Y,Z
from sys import argv
__author__ = 'TaQini'

if len(argv) != 2:
    print 'usage: '
    print ' $ ./sample_without_label.py <feature_file>'
    print 'example: '
    print ' $ ./sample_without_label.py X.feature'
    exit(0)

feature_file  = open(argv[1],'r')
sample_file = open(argv[1]+'.sample','w+')

# read data
data = feature_file.readlines()
feature_file.close()

# del all item having '?'
result = []
for item in data:
    if '?' not in item:
        result.append(item)

sample_file.writelines(result)
sample_file.close()
```

4.1.3 整合训练集文件

将训练集筛选后得到的特征文件和对应的标签文件，分别整合为 1 个文件合集。

整合文件命令如下：

```
$ cat A.feature.sample B.feature.sample C.feature.sample D.feature.sample
E.feature.sample > training.feature
$ cat A.label.sample B.label.sample C.label.sample D.label.sample
E.label.sample > training.label
```

4.1.4 特征识别

编写特征识别程序，预测测试集中特征对应的标签。程序的输入为训练集特征文件、训练集标签文件、测试集特征文件，输出为测试集特征文件相应的标签文件，程序如下：

```
#!/usr/bin/python
import numpy as np
from sys import argv
from sklearn import tree
__author__ = 'TaQini'

if len(argv) != 5:
    print 'usage: '
    print  '  $  ./fit.py  training_feature  training_label  predict_fecture
predict_label'
    print 'example: '
    print ' $ ./fit.py training.feature training.label X.feature X.label'
    exit(0)

feature_file = open(argv[1],'ro')
label_file = open(argv[2],'ro')

X = np.array([eval(item) for item in feature_file.readlines()])
Y = np.array([eval(item) for item in label_file.readlines()])

feature_file.close()
label_file.close()

clf = tree.DecisionTreeClassifier()
clf = clf.fit(X, Y)

predict_file = open(argv[3],'ro')
predict_lable = open(argv[4],'w+')

pX = np.array([eval(item) for item in predict_file.readlines()])
predict_file.close()

pY = clf.predict(pX)
for i in pY:
   predict_lable.write(str(i)+'\n')
predict_lable.close()
```

4.1.5 实验步骤

1. 解压数据集文件，将所有文件移至同一目录下。编写实验所需的上述程序，依次命名为 sample_with_label.py、sample_without_sample.py、fit.py，并保存至数据集各文件所在的目录下。

2. 对训练集 A,B,C,D,E 的原始数据，分别执行 sample_with_label.py 程序，进行数据筛选，得到筛选后的特征文件 ?.feature.sample 和标签文件 ?.label.sample。

例如，对于训练集 A，执行命令： ./sample_with_label.py A.feature A.label，命令执行结束后，在同一目录下得到 A.feature.sample 文件和 A.label.sample 文件。

3. 通过 cat 命令整合训练集文件，得到训练集特征文件集合 training.feature 和标签文件集合 training.label。具体命令参考 4.1.3 节内容。

4. 对测试集 X,Y,Z 的原始数据，分别执行 sample_without_label.py 程序，进行数据筛选，得到筛选后的特征文件 ?.feature.sample。

例如，对于测试集 X，执行命令： ./sample_without_label.py X.feature，命令执行结束后，在同一目录下得到 X.feature.sample 文件。

5. 对测试集 X,Y,Z 筛选后的特征文件，分别通过执行 fit.py 程序，进行特征识别，得到预测的筛选后的特征文件对应的标签文件 ?.label。

例如，对于测试集 X，执行命令： ./fit.py training.feature training.label X.feature.sample X.label ，命令执行结束后，在同一目录下得到 X.label 文件。

4.2 实验结果

X,Y,Z 三组测试集预测结果待老师评估。

本文实验中，为测试预测结果的准确度，即识别率（正确识别标签数/标签总数），编写了评估程序 evaluate.py 和批量测试程序 test.py。

评估程序统计预测标签与实际标签中吻合数量，输出识别率，代码如下：

```
#!/usr/bin/python
from sys import argv
real = open(argv[1]).readlines()
perdict = open(argv[2]).readlines()
rdata = [eval(i) for i in real]
pdata = [eval(i) for i in perdict]
cnt = 0
for i in range(len(rdata)):
    if rdata[i] == pdata[i]:
        cnt += 1
print 100.0*cnt/len(rdata)
```

批量测试程序用于批量测试识别率，输入训练集序列、测试集序列，程序对训练集和测试集中的每一项，分别做训练集与测试集，并输出识别率，代码如下：

```
#!/usr/bin/python
from subprocess import *

training_set = 'ABCDE'
test_set = 'ABCDE'

for i in training_set:
    for j in test_set:
        if i!=j:
            pf = Popen(('./fit.py '+i+'.feature.sample '+i+'.label.sample
'+j+'.feature.sample '+j+'.label').split(),stdout=PIPE)
            pf.communicate()
            pe = Popen(('./evaluate.py '+j+'.label.sample
'+j+'.label').split(),stdout=PIPE)
            print i,j,pe.communicate()[0],
```

测试结果如表 4-1 所示：

表 4-1 训练集相互测试结果

测试集 / 训练集	A	B	C	D	E
A	-	62.6	29.6	46.4	43.9
B	56.6	-	52.9	50.2	62.4
C	40.9	46.1	-	37.6	44.7
D	60.5	54.8	52.2	-	63.0
E	35.4	47.7	32.8	50.7	-

5 讨论与结束语

本文使用决策树算法对传感器采集的用户数据进行分类，预测用户行为，但由于实验中选取的特征过于复杂，可能导致决策树学习时出现过学习(over-fitting)问题，导致决策树过于复杂,损失了泛化能力。解决过学习问题需要运用“奥坎姆剃刀”原则，可通过提前停止（设置阈值）策略和剪枝（应用最小描述长度 MDL 方法，降低决策树复杂度）策略实现。

References:

[1] 徐川龙，顾勤龙，姚明海．一种基于三维加速度传感器的人体行为识别方法[J]. 计算机系统应用,2013,(6).doi:10.3969/j.issn.1003-3254.2013.06.030.