



交通数据挖掘作业汇报

应用机器学习方法
识别手机传感器数据中的交通方式

15级交通工程2班 王倩妮 2015112956

内容提要

目录 > >

- 01 数据挖掘方法的选定
- 02 搭建环境
- 03 数据预处理
- 04 神经网络方法
- 05 模型训练结果
- 06 模型用于预测
- 07 支持向量机方法



题目要求

- 训练数据文件夹中是已标定好的出行个体在出行过程中的时间 (s)、经纬度 (度)、速度 (m/s) 和对应交通方式 (1-步行, 2-自行车, 3-公交车, 4-小汽车)。
- 请查阅相关文献资料, 选取合适的特征属性, 利用有监督机器学习算法 (如人工神经网络、支持向量机、决策树、朴素贝叶斯等) 训练出有效的 交通方式分类器, 最后对测试数据文件夹中每个出行个体的交通方式进行 测试分类, 识别出出行方式模式【如步行-小汽车-步行 (1-4-1)、步行-自行车-公交车-步行 (1-2-3-1) 等】, 以及交通方式发生切换的时间点。



数据挖掘方法的选定

机器学习分类方式

✓ 按学习方式分类

- 有监督学习 (supervised learning)
- 无监督学习 (unsupervised learning)
- 半监督学习 (semi-supervised learning)

✓ 按问题归属分类

- 分类 (classification)
- 回归 (regression)
- 聚类 (clustering)
- 降维 (dimensionality reduction)

本问题要求

- 有监督
- 分类问题



数据挖掘方法的选定

有监督的分类问题常用方法

- k-邻近算法
- 决策树
- 朴素贝叶斯（常用于进行文本分类）
- 支持向量机（常用于二元分类）
- 神经网络



本次课程作业选定方法

- 神经网络



搭建环境



编程语言



TensorFlow

Google开发神经网络框架

- ✓ 分类器整体编程语言：Python
- ✓ 神经网络框架：TensorFlow
- ✓ 矩阵运算：Numpy
- ✓ 绘图：Matplotlib
- ✓ IDE：Jupyter Notebook
- ✓



数据预处理

✓ 数据集:

- 训练数据 (共6个个体、17130条数据) (个体编号、时间、经度、纬度、速度、交通方式)
- 测试数据 (个体编号、时间、经度、纬度、速度)

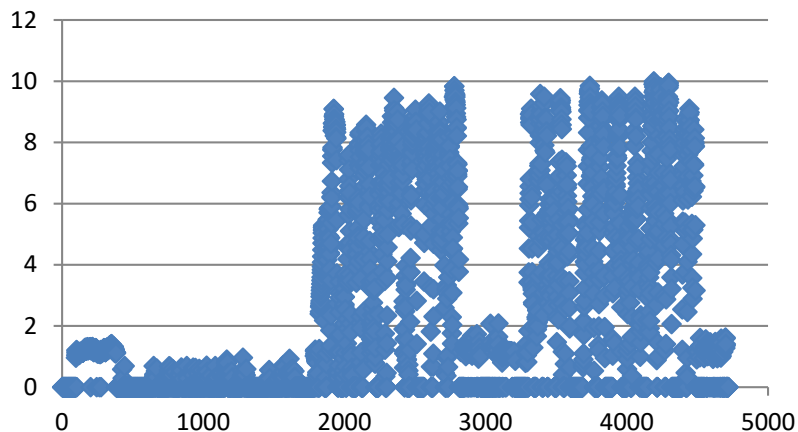
✓ 交通方式:

- 1-步行, 2-自行车, 3-公交车, 4-小汽车

✓ 数据频率:

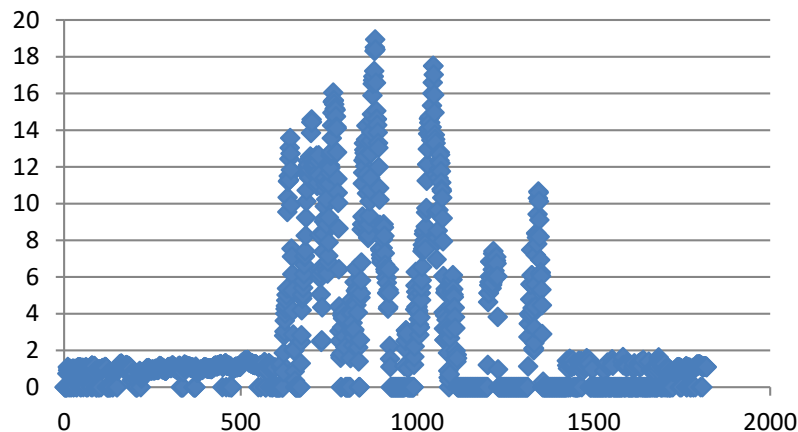
- 1次/s

速度(m/s)



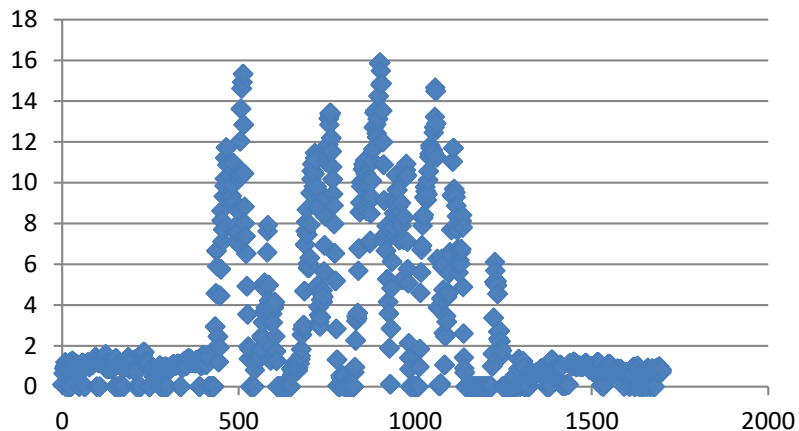
出行个体1: 1-3-1-3-1
步行-公交车-步行-公交车-步行

速度(m/s)



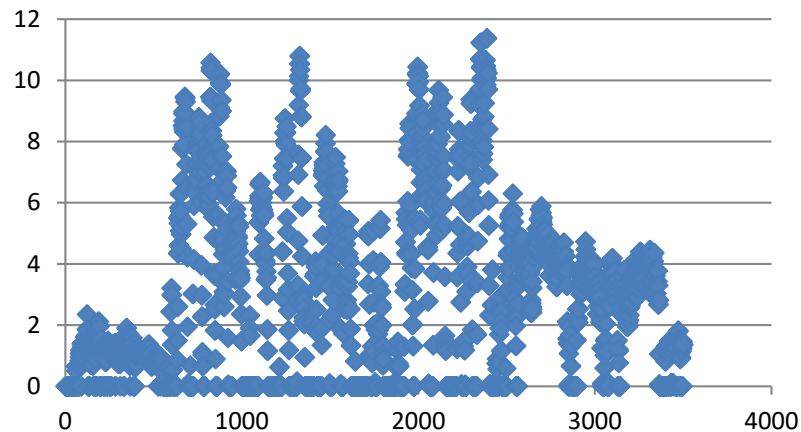
出行个体2: 1-4-1
步行-小汽车-步行

速度(m/s)



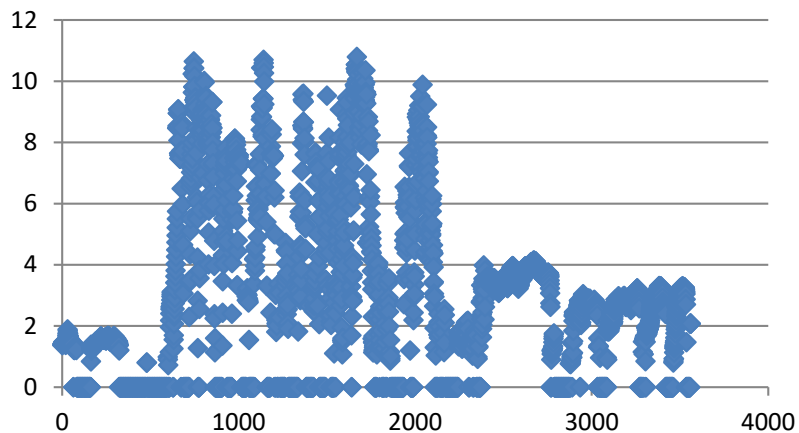
出行个体3: 1-4-1
步行-小汽车-步行

速度(m/s)



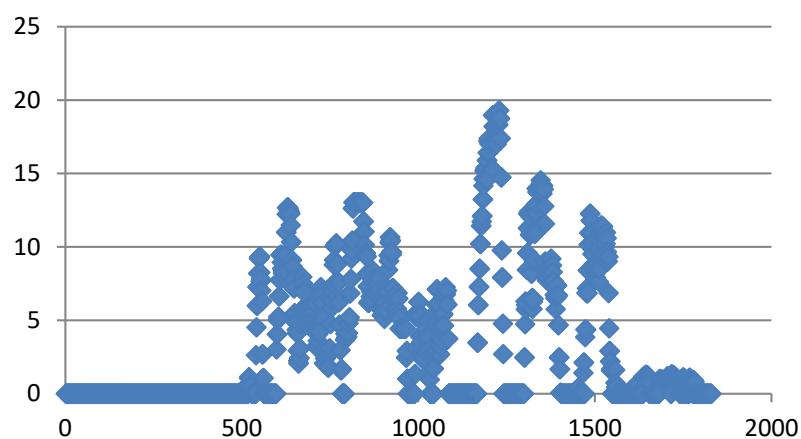
出行个体4: 1-3-2-1
步行-公交车-自行车-步行

速度(m/s)



出行个体5: 1-3-1-2-1
步行-公交车-步行-自行车-步行

速度(m/s)



出行个体6: 1-4-1
步行-小汽车-步行

数据预处理

□ 观察训练数据:

- 步行速度平稳且处于较低水平
- 自行车速度高于步行速度
- 公交车与小汽车速度较高
- 公交车与小汽车的波动性存在差异
-



□ 初步结论:

- 交通方式在一段时间内连续（不存在频繁跳动）
- 交通方式与速度与速度变化有关



数据预处理

□ 输入：

- 因为相关性的存在，所以不能直接输入原始数据，而是应该在考虑相关性的基础上处理速度数据。



□ 数据输入的思路讨论：分组思想

- × 尝试一：每个SCALE长度的数据分为一组。17130条数据分为 $(17130/SCALE)$ 组。
- ✓ 尝试二：滚动分组思想，每条数据与之前(SCALE)与后(SCALE)分为一组，组长 $2SCALE$ 。



数据预处理

□ 尝试一——确定神经网络的输入与输出：

➤ 输入：(17130/SCALE) 条数据

(每组总结为1条数据，具体操作见下一页)

- 每个SCALE内的平均速度 \bar{v}
- 每个SCALE内的最大速度 v_{max}
- 每个SCALE内的速度标准差 $std(v)$

➤ 输出：(17130/SCALE) 条数据

- 每个SCALE对应的交通方式

(转化为one-hot模式)

[[1,0,0,0],
[0,1,0,0],
[0,0,1,0],
[0,0,0,1]]



数据预处理

□ 数据预处理:

- 确定SCALE=60（每分钟进行统计）

STEP1: 选取文件，得到速度列、出行方式列

STEP2: 计算此文件能够分为几个组，每组长度为SCALE

STEP3: 计算训练数据中每个组的神经网络输入

STEP4: 计算训练数据中每个组的神经网络输出（占比最大）

STEP5: 转化神经网络输出为one-hot形式

STEP6: 得到用于神经网络训练使用的数据data_x与data_y



数据预处理

□ 尝试二——确定神经网络的输入与输出：

➤ 输入：17130条数据

• 滚动每2SCALE内的平均速度 \bar{v}

• 滚动每2SCALE内的最大速度 v_{max}

• 滚动每2SCALE内的速度标准差 $std(v)$

➤ 输出：17130条数据

• 每个SCALE对应的交通方式
(转化为one-hot模式)

[[1,0,0,0],
[0,1,0,0],
[0,0,1,0],
[0,0,0,1]]



数据预处理

□ 数据预处理:

- 确定SCALE=60 (滚动组长=120)

STEP1: 选取文件, 得到速度列、出行方式列

STEP2: 按滚动规则, 计算训练数据中每条数据的神经网络输入[平均速度, 最大速度, 速度标准差]

STEP3: 直接输入每条数据对应的出行方式

STEP4: 转化神经网络输出为one-hot形式

STEP5: 得到用于神经网络训练使用的数据data_x与data_y



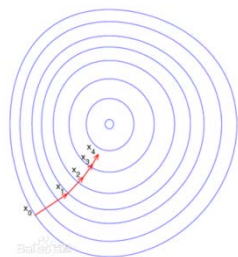
神经网络方法

✓ 基本工作原理

- 反向传播
- Loss(又称为cross_entropy or cost)

$$H(X = x) = - \sum_x p(x) \log q(x) \quad \text{交叉熵计算}$$

- 梯度下降



使Loss最小

- Accuracy

神经网络输出为每种出行方式的概率，如 $[0.9, 0.02, 0.03, 0.05]$ ，识别结果即为最大的一项。

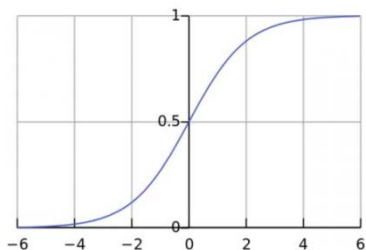
$$\text{Accuracy} = \frac{\text{准确识别数据条数}}{\text{数据条数总数}}$$



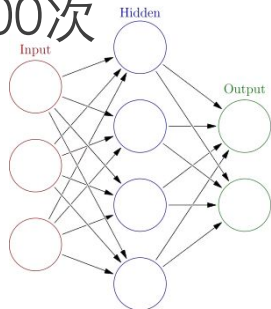
模型训练结果

✓ 尝试一参数设置

- 神经层数：4层
- 每层的神经元数：10个元
- 激活函数选取：softmax函数
- 梯度下降优化器选取：
GradientDescentOptimizer
- 学习率：0.001
- 全局or随机学习：全局
- 循环次数：10000次

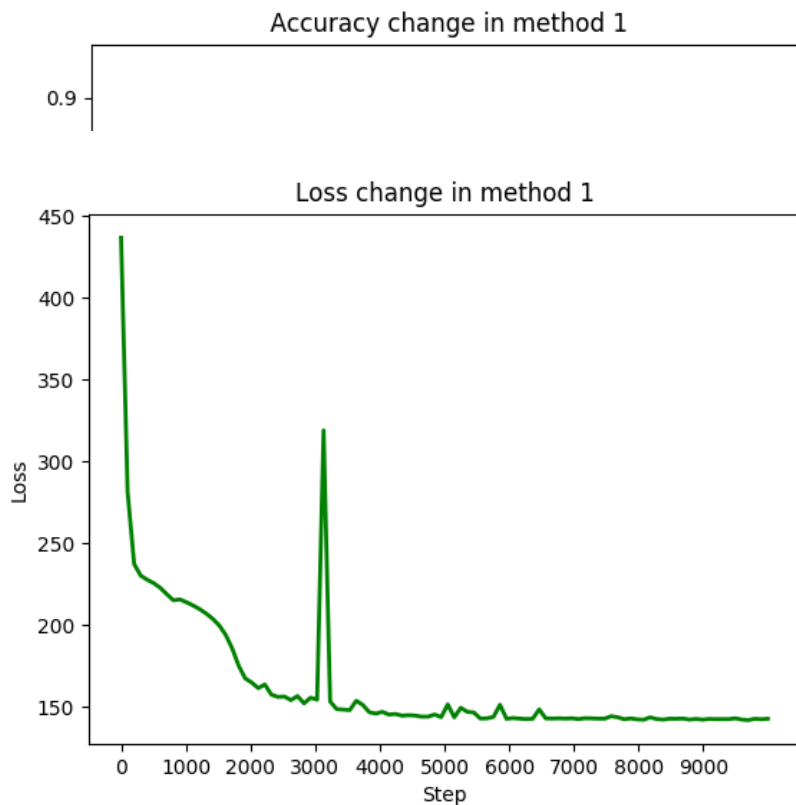


Softmax函数

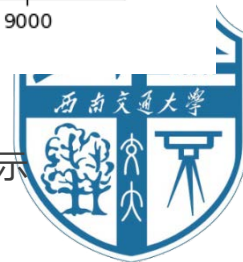


神经网络示意

✓ 尝试一训练过程指标变化图



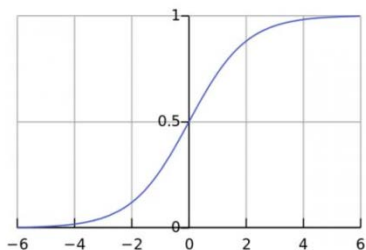
随着训练次数增加loss与accuracy的变化如上图所示



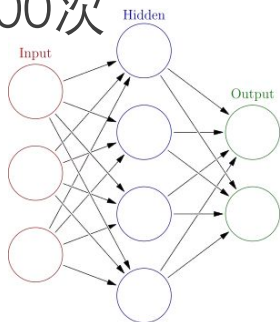
模型训练结果

✓ 尝试二参数设置

- 神经层数：4层
- 每层的神经元数：10个元
- 激活函数选取：softmax函数
- 梯度下降优化器选取：
GradientDescentOptimizer
- 学习率：0.001
- 全局or随机学习：全局
- 循环次数：25000次

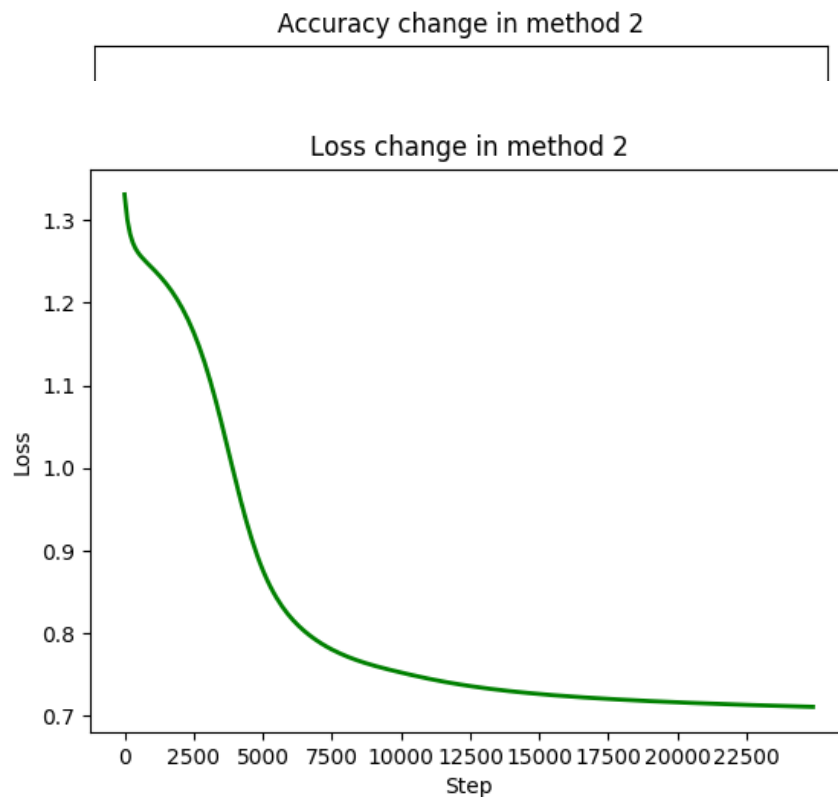


Softmax函数



神经网络示意

✓ 尝试二训练过程指标变化图



随着训练次数增加loss与accuracy的变化如上图所示




模型训练结果


推测在出行方式识别的过程中：步行与自行车方式较容易进行识别，公交车与小汽车不易识别，因此分方式验证准确率。

✓ 尝试一分方式准确率结果

 步行:


 自行车:

 公交车:


 小汽车:


暂时没算出具体数值

✓ 尝试二分方式准确率结果

 步行:

 自行车:

 公交车:

 小汽车:

暂时没算出具体数值



神经网络方法

✓ TensorFlow操作步骤

• STEP1: 定义全局变量

- 输入数据尺度 `X_IN_SIZE=3`
- 输出数据尺度 `N_CLASS = 4`
- 学习效率 `LR=0.01`

```
def add_layer(input_data,in_size,out_size,activation_function=None):  
    Weight=tf.Variable(tf.random_normal([in_size,out_size]))  
    biases=tf.Variable(tf.zeros([1,out_size])+0.1)  
    wx_b=tf.matmul(input_data,Weight)+biases  
    if activation_function==None:  
        outputs=wx_b  
    else:  
        outputs=activation_function(wx_b)  
    return outputs
```

• STEP2: 定义添加层函数

• STEP3: 定义输入、输出数据的placeholder

```
X = tf.placeholder(tf.float32, [None, X_IN_SIZE])  
Y = tf.placeholder(tf.float32, [None, N_CLASS])
```

• STEP4: 添加层（4层网络，每层网络宽度为10）

```
LAYER_UNIT=10  
hidden_layer1=add_layer(X,X_IN_SIZE,LAYER_UNIT,activation_function=tf.nn.softmax)  
hidden_layer2=add_layer(hidden_layer1,LAYER_UNIT,LAYER_UNIT,activation_function=tf.nn.softmax)  
hidden_layer3=add_layer(hidden_layer2,LAYER_UNIT,LAYER_UNIT,activation_function=tf.nn.softmax)  
y=add_layer(hidden_layer3,LAYER_UNIT,N_CLASS,activation_function=tf.nn.softmax)
```



神经网络方法

✓ TensorFlow操作步骤

- STEP5: 定义Loss(又称为cross_entropy or cost)

```
cross_entropy = tf.reduce_mean(-tf.reduce_sum(Y* tf.log(y),reduction_indices=[1]))
```

- STEP6: 定义梯度下降优化器

```
train_step = tf.train.GradientDescentOptimizer(LR).minimize(cross_entropy)
```

- STEP7: 定义神经网络分类准确率

```
correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(Y,1))  
accuracy = tf.reduce_mean(tf.cast(correct_prediction , "float"))
```

- STEP8: 保存模型

```
saver = tf.train.Saver()  
save_path = saver.save(sess, "my_net/save_net.ckpt")
```

- STEP9: 完成计算图的定义并运算

```
init = tf.initialize_all_variables()  
sess = tf.Session()  
sess.run(init)  
for i in range(25000):  
    _,loss_=sess.run([train_step,cross_entropy], feed_dict={X: data_x, Y: data_y})  
    accuracy_=sess.run(accuracy, feed_dict={X: data_x, Y: data_y})  
    if i % 100== 0:  
        print ('trainstep', '%s'%i, 'loss=', loss_, '|accuracy=', accuracy_)
```



模型用于预测

□ 测试数据预测思路:

STEP1: 选取测试文件, 得到速度列、出行方式列

STEP2: 按滚动规则, 计算训练数据中每条数据的神经网络输入[平均速度, 最大速度, 速度标准差]

STEP3: 得到输入数据data_x

STEP4: 带入训练好的神经网络进行预测, 并将输出结果按概率高低转化为出行方式。

STEP5: 使用平滑手段进行平滑, 去除跳变的数据, 保证连续性/每隔一定长度取众数, 保证连续性

STEP6: 依据每个测试数据对应的交通方式, 得出交通方式切换时



心得体会

□ 心(qiang)得(lie)体(tu)会(cao):

- 同样的输入与输出，神经网络的效果与参数的设定有很大关系
(神经网络层数、神经层的宽度、学习效率、激活函数选择、梯度下降优化器选择、损失函数度量方式……)
- 一定要对于tensor的维度有着清楚的认知
- 学习效率过大可能会出现loss=nan的状况，此时应当以10的数量级递减
- 未解：为何在初始时会有accuracy不变的情况？预测的准的一直不变？

□ 改(wei)进(jie)方(zhi)向(mi):

- 由于练习数据量的限制，本次用全部训练数据进行模型训练和准确率计算。今后可以使用“交叉验证”的方式或增加数据量的方式改进，更为科学。
- 模型有很大的进一步优化的空间。



然后.....



支持向量机方法

- 由于神经网络的结果太糟心，不忍上台汇报。所以几小时前临时改换支持向量机方法一试，结果：

全局准确率 0.9284880326911851
步行准确率： 0.93853591160221
自行车准确率： 0.9604938271604938
公交车准确率： 0.9656453110492108
小汽车准确率： 0.7923387096774194

!!!!

- 借助sklearn

scikit-learn

Machine Learning in Python

sklearn.svm.SVC

C-Support Vector Classification.

The implementation is based on libsvm. The fit time complexity is more than quadratic with the number of samples which makes it hard to scale to dataset with more than a couple of 10000 samples.

The multiclass support is handled according to a one-vs-one scheme.



支持向量机方法

□ 输入：SCALE=60

- 原始速度
- 原始速度时间序列下的一次指数平滑值，阻尼=0.5
- 滚动每2SCALE内的平均速度 \bar{v}
- 滚动每2SCALE内的最大速度 v_{max}
- 滚动每2SCALE内的速度标准差 $std(v)$

□ 输出：

- 直接输出出行方式所属类别[1,2,3,4]



支持向量机方法

□ 6个样本数据的5项输入数据共同构成data_x,1项输出数据共同构成data_y。

□ 随机选取80%作为训练数据

□ 剩余20%为验证数据

□ 使用sklearn的SVC模块输入训练数据集
进行训练（默认参数）

□ 输入验证数据计算准确率

□ 将测试数据输入模型进行预测，并保存结果

□ 结果经平滑/众数处理，编程
找出换乘时间。（原理：变化即
保存）

```
import random
a=range((len(data_x)))
b=random.sample(a,int(0.8*len(data_x)))
c=list(set(a).difference(set(b)))
data_xnew=[]
data_ynew=[]
Data_xnew=[]
Data_ynew=[]
for i in b:
    data_xnew.append(data_x[i])
    data_ynew.append(data_y[i])
for i in c:
    Data_xnew.append(data_x[i])
    Data_ynew.append(data_y[i])
data_ynew=np.array(data_ynew)
```

```
clf = SVC()
clf.fit(data_xnew, data_ynew.ravel())
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
y=clf.predict(Data_xnew)
count=0
for i in range (len(y)):
    if y[i]==Data_ynew[i]:
        count+=1
accuracy=count/(len(y))
print(accuracy)
```





谢谢聆听!