数据挖掘第二次大作业报告

乔嘉林 2016311941
林　丽 2016311940

一. 任务一

1. 数据处理

1) 对缺省数据的处理：枚举类型的数据，按照说明文档中的排列顺序，将？的缺省数据作为一个类别，按照序列号进行记录，对枚举数据进行数值化。

2) 数值数据的分段处理：鉴于年龄数据在 100 以内变化，且相对均匀连续，因此不作处理；fnlwgt 可看做一列无关数据，因其值是经过一定方式对其他数据进行计算的结果，鉴于取值范围在 5 位数，用除以 1000 的方法取高位数进行缩小取值范围。Capital-gain 和 capital-loss 这两列数据也是连续数值型，但是取值差异较大，因此用除以 100 进行取商值处理 。hours-per-week 虽然也是连续值，但是变化范围有限，且均为整型，可作离散数据处理。

2. 算法描述

A. 决策树

本算法使用了 Python 的 skylearn 库。

决策树的构造过程不依赖领域知识，它使用属性选择度量来选择将元组最好地划分成不同的类的属性。决策树的构造过程就是进行属性选择度量确定各个特征属性之间的拓扑结构。

构造决策树的关键步骤是分裂属性。就是在某个节点处按照某一特征属性的不同划分构造不同的分支，其目标是让各个分裂子集尽可能地“纯”。尽可能“纯”就是尽量让一个分裂子集中待分类项属于同一类别。分裂属性分为三种不同的情况：

1) 属性是离散值且不要求生成二叉决策树。此时用属性的每一个划分作为一个分支。

2) 属性是离散值且要求生成二叉决策树。此时使用属性划分的一个子集进行测试，按照“属于此子集”和“不属于此子集”分成两个分支。

3) 属性是连续值。此时确定一个值作为分裂点 split_point，按照>split_point 和<=split_point 生成两个分支。

构造决策树的关键性内容是进行属性选择度量，属性选择度量是一种选择分裂准则，是将给定的类标记的训练集合的数据划分 D“最好”地分成个体类的启发式方法，它决定了拓扑结构及分裂点 split_point 的选择。

属性选择度量算法有很多，一般使用自顶向下递归分治法，并采用不回溯的贪心策略。

用训练出的决策树对测试集进行分类准确率 76.38%。

B. 朴素贝叶斯方法

朴素贝叶斯分类是一种十分简单的分类算法，对于给出的待分类项，求解在此项出现的条件下各个类别出现的概率，哪个最大，就认为此待分类项属于哪个类别。朴素贝叶斯分类的正式定义如下：

1) 设$x = \{a_1, a_2, ..., a_m\}$为一个待分类项，而每个 a 为 x 的一个特征

属性。有类别集合$C=\{y_1,y_2,...,y_n\}$。

2) 计算$P(y_1|x),P(y_2|x),...,P(y_n|x)$。

3) 如果$P(y_k|x)=max\{P(y_1|x),P(y_2|x),...,P(y_n|x)\}$，则$x\in y_k$。

那么现在的关键就是如何计算第 3 步中的各个条件概率。我们可以这么做：

找到一个已知分类的待分类项集合，这个集合叫做训练样本集。

统计得到在各类别下各个特征属性的条件概率估计。即$P(a_1|y_1),P(a_2|y_1),...,P(a_m|y_1);P(a_1|y_2),P(a_2|y_2),...,P(a_m|y_2);...;P(a_1|y_n),P(a_2|y$。

如果各个特征属性是条件独立的，则根据贝叶斯定理有如下推导：

$$P(y_i|x)=\frac{P(x|y_i)P(y_i)}{P(x)}$$

因为分母对于所有类别为常数，因为我们只要将分子最大化皆可。又因为各特征属性是条件独立的，所以有：

$$P(x|y_i)P(y_i)=P(a_1|y_i)P(a_2|y_i)...P(a_m|y_i)P(y_i)=P(y_i)\prod_{j=1}^{m}P(a_j|y_i)$$

C. 神经网络方法

所谓神经网络就是将许多个单一“神经元”联结在一起，这样，一个“神经元”的输出就可以是另一个“神经元”的输入。如图 1-3-1 所示，我们使用圆圈来表示神经网络的输入，标上“$+1$”的圆圈被称为偏置节点，也就是截距项。神经网络最左边的一层叫做输入层，最右的一层叫做输出层（本例中，输出层只有一个节点）。中间所有节点组成的一层叫做隐藏层，因为我们不能在训练样本集中观测到它们的值。同时可以看到，以上神经网络的例子中有 3 个输入单元（偏置单元不计在内），3 个隐藏单元及一个输出单元。

我们用n_l来表示网络的层数，我们将第l层记为L_l，于是L_1是输入层，输出层是L_{n_l}。本例神经网络有参数$(W,b)=(W^{(1)},b^{(1)},W^{(2)},b^{(2)})$，其中$W_{ij}^{(l)}$是第$l$层第$j$单元与第$l+1$层第$i$单元之间的联接参数，$b_i^{(l)}$是第$l+1$层第$i$单元的偏置项。因此，$W^{(1)}\in\Re^{3\times3}$，$W^{(2)}\in\Re^{1\times3}$。注意，没有其他单元连向偏置单元(即偏置单元没有输入)，因为它们总是输出$+1$。同时，我们用s_l表示第l层的节点数（偏置单元不计在内）。

我们用$a_i^{(l)}$表示第l层第i单元的激活值（输出值）。

当 $l=1$ 时，$a_i^{(1)}=x_i$，也就是第 i 个输入值（输入值的第 i 个特征）。对于给定参数集合 W,b，我们的神经网络就可以按照函数 $h_{W,b}(x)$ 来计算输出结果。神经网络的计算步骤如下：

$$
\begin{aligned}
a_1^{(2)} &= f(W_{11}^{(1)}x_1+W_{12}^{(1)}x_2+W_{13}^{(1)}x_3+b_1^{(1)})\\
a_2^{(2)} &= f(W_{21}^{(1)}x_1+W_{22}^{(1)}x_2+W_{23}^{(1)}x_3+b_2^{(1)})\\
a_3^{(2)} &= f(W_{31}^{(1)}x_1+W_{32}^{(1)}x_2+W_{33}^{(1)}x_3+b_3^{(1)})\\
h_{W,b}(x) &= a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)}+W_{12}^{(2)}a_2^{(2)}+W_{13}^{(2)}a_3^{(2)}+b_1^{(2)})
\end{aligned}
$$

我们用 $z_i^{(l)}$ 表示第 l 层第 i 单元输入加权和（包括偏置单元），比如，$z_i^{(2)}=\sum_{j=1}^{n}W_{ij}^{(1)}x_j+b_i^{(1)}$，则 $a_i^{(l)}=f(z_i^{(l)})$。

这样我们就可以得到一种更简洁的表示法。这里我们将激活函数 $f(\cdot)$ 扩展为用向量（分量的形式）来表示，即 $f([z_1,z_2,z_3])=[f(z_1),f(z_2),f(z_3)]$，那么，上面的等式可以更简洁地表示为：

$$
\begin{aligned}
z^{(2)} &= W^{(1)}x+b^{(1)}\\
a^{(2)} &= f(z^{(2)})\\
z^{(3)} &= W^{(2)}a^{(2)}+b^{(2)}\\
h_{W,b}(x) &= a^{(3)} = f(z^{(3)})
\end{aligned}
$$

我们将上面的计算步骤叫作前向传播。回想一下，之前我们用 $a^{(1)}=x$ 表示输入层的激活值，那么给定第 l 层的激活值 $a^{(l)}$ 后，第 $l+1$ 层的激活值 $a^{(l+1)}$ 就可以按照下面步骤计算得到：

$$
\begin{aligned}
z^{(l+1)} &= W^{(l)}a^{(l)}+b^{(l)}\\
a^{(l+1)} &= f(z^{(l+1)})
\end{aligned}
$$

将参数矩阵化，使用矩阵－向量运算方式，我们就可以利用线性代数的优势对神经网络进行快速求解。

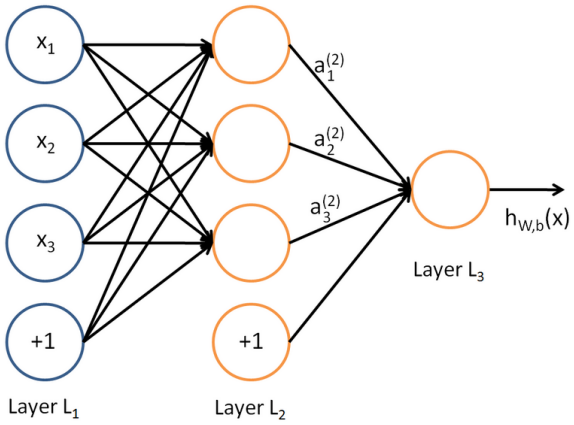


图 1-3-1 神经网络示意图

3. 结果分析

决策树算法是调用 python2.7.12 中的包实现的，朴素贝叶斯是用 Java 语言实现，没有调用已有包，神经网络利用 tensorflow 实现，共两层，每层结点数分别为 10,5，学习率 0.01，最大迭代次数 2000。由下图可见，三种算法中，神经网络的准确率最高。朴素贝叶斯比决策树要高一些。我们认为原因如下：决策树在选择属性进行分支的时候，对属性重要程度有所依赖，与此同时，数据集中并不是每个属性都能够有明确的偏向，比如 fnlwgt，很容易对分类结果产生干扰，因此决策树效果最差。而朴素贝叶斯每一次判断都基于所有属性的先验概率值，相对决策树，对每个属性的重要程度依赖比较平均，但是贝叶斯公式要求变量之间相互独立，数据集中显然有不独立的属性，比如 capital-gain 和 capital-loss 也会对分类结果产生影响。神经网络的非线性 sigmoid 层很好地进行非线性化处理，削弱了一些属性的重要性，同时增强了一些属性的作用，显然更容易充分学习数据集的特征，达到最好的效果。

算法	决策树	朴素贝叶斯	神经网络
准确率	0.761255451139	.08217554204	0.9982

二. 任务二

1. 算法描述

A. K-means 算法

1) 从 N 个文档随机选取 K 个文档作为质心。
2) 对剩余的每个文档测量其到每个质心的距离，并把它归到最近的质心的类
3) 重新计算已经得到的各个类的质心
4) 迭代 2～3 步直至新的质心与原质心相等或小于指定阈值，算法结束

B. DBSCAN 算法

r 领域：给定对象半径为 E 内的区域称为该对象的 r 领域；

核心对象：如果给定对象 E 领域内的样本点数大于等于 min，则称该对象为核心对象；

直接密度可达：对于样本集合 D，如果样本点 q 在 p 的 r 领域内，并且 p 为核心对象，那么对象 q 从对象 p 直接密度可达。

密度可达：对于样本集合 D，给定一串样本点 p1,p2…pn，p= p1,q= pn, 假如对象 pi 从 pi-1 直接密度可达，那么对象 q 从对象 p 密度可达。

密度相连：存在样本集合 D 中的一点 o，如果对象 o 到对象 p 和对象 q 都是密度可达的，那么 p 和 q 密度相联。算法过程如下:

1) 扫描原始数据，获取所有的数据点。

2) 遍历数据点中的每个点，如果此点已经被访问过，则跳过，否则取出此点做聚类查找。

3) 以步骤 2 中找到的点 P 为核心对象，找出在 r 领域内所有满足条件的点，如果个数大于等于 min，则此点为核心对象，加入到簇中。

4) 再次 P 为核心对象的簇中的每个点，进行递归的扩增簇。如果 P 点的递归扩增结束，再次回到步骤 2。

5) 算法的终止条件为所有的点都被访问。

2. 算法实验分析

A. 数据集一

在数据集一上用 K-means 算法聚类，不同 k 值对应的不同 Purity 和 F-Score 如下图 2-1-1 所示。purity 和 F-Score 随着 k 的增加先上升后下降，在 k=11 时达到峰值。

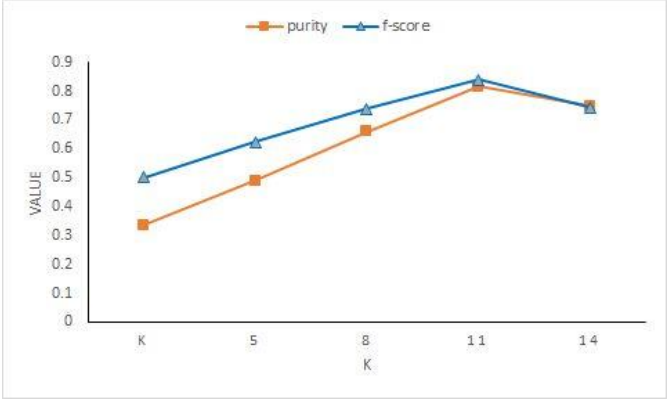


图 2-1-1 purity 和 f-score 随 k 值的变化

在数据集一上用 DBSCAN 算法聚类，不同 r 和 min 对应不同的 Purity 和 F-Score 如下图 2-1-2 所示。在固定 min 为 2 时，准确度和 F-Score 随 r 的增加先上升后下降，在 r 为 9000 时到达峰值。

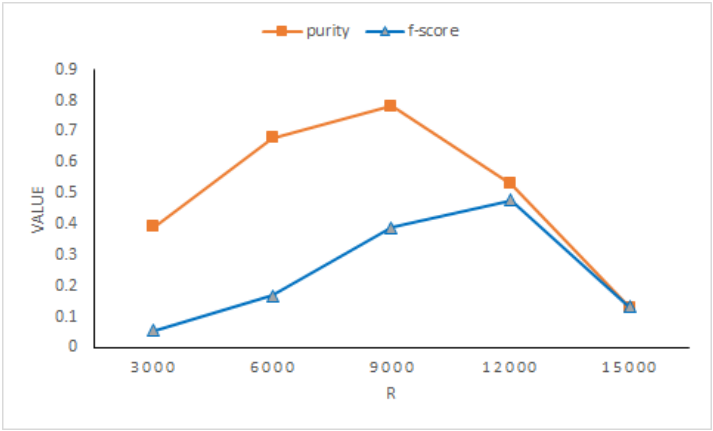


图 2-1-2 purity 和 f-score 在 min=2 时随 r 值的变化

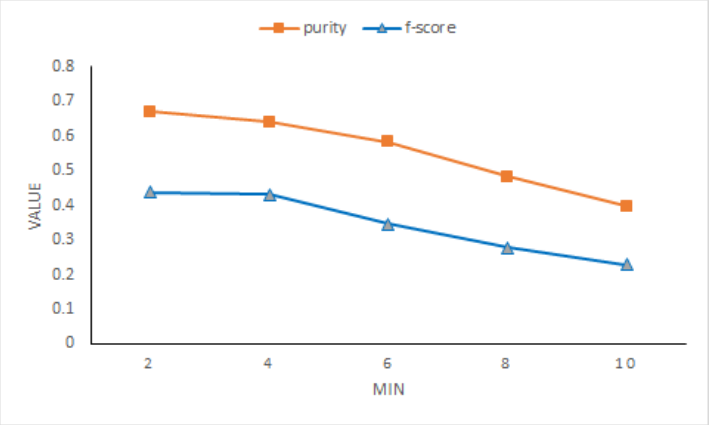


图 2-1-3 purity 和 f-score 在 r=10000 时随 min 值的变化

在固定 r 为 10000 时，准确率和 F-Score 随 min 的增加而下降。如图 2-1-3 所示。

B. 数据集二

在数据集二上用 K-means 算法聚类，不同 k 值对应的不同 Purity 和 F-Score 如下图 2-2-1。purity 随着 k 的上升而上升，F-Score 随着 k 的增加而下降。

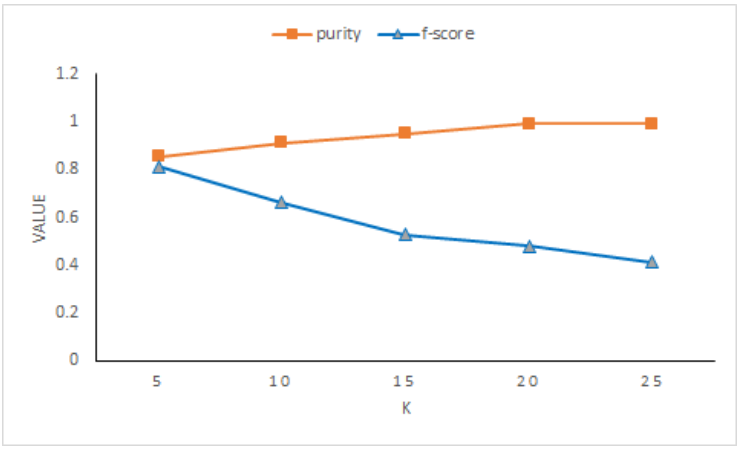


图 2-2-1 purity 和 f-score 随 k 值的变化

在数据集二上用 DBSCAN 算法聚类，不同 r 和 min 对应不同的 Purity 和 F-Score 如下图 2-2-2：在固定 min 为 5 时，准确度和 F-score 随 r 的增加而增加，在 r=1 时接近最大值。

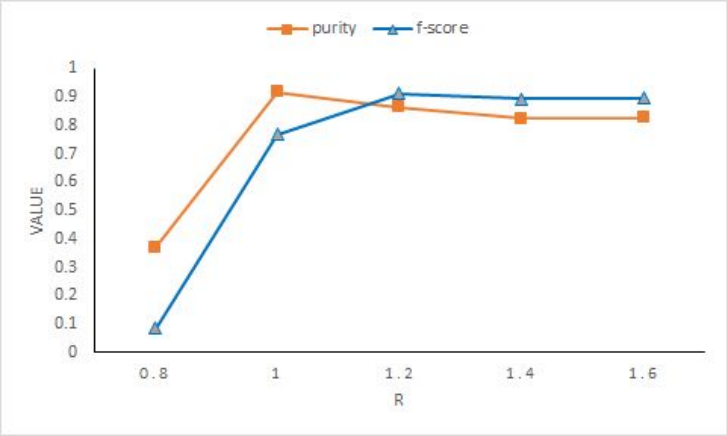


图 2-2-2 purity 和 f-score 在 min=5 时随 r 值的变化

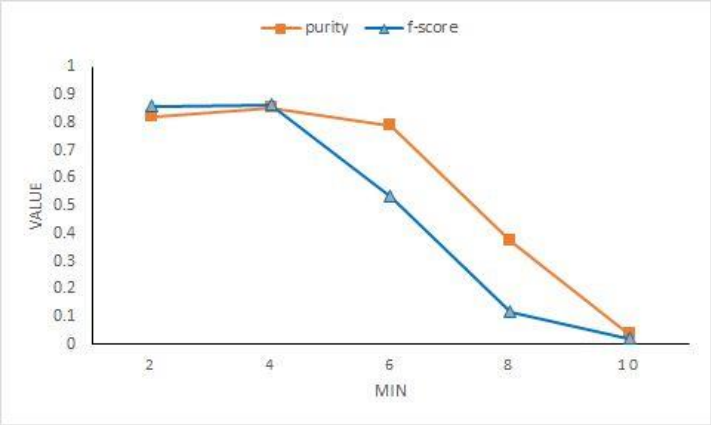


图 2-2-3 purity 和 f-score 在 r=1 时随 min 值的变化

这张图表示，在固定 r 邻域的半径为 1 时，精确度和 F-score 随 minPts 的增加而降低。

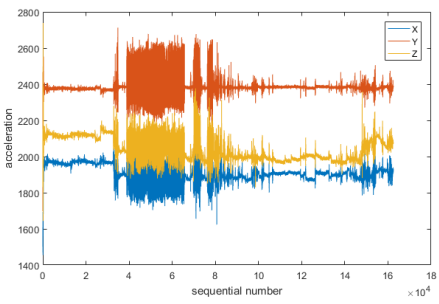
3. 结论

在数据集一上 Kmeans 比 DBSCAN 聚类算法快，应该和数据分布有关，最高准确率两者差不多，Kmeans 稍高一些。

数据集二上 Kmeans 比 DBSCAN 要快一些，准确率上要更高，Kmeans 可以达到 99%左右的准确率，DBSCAN 可以达到 90%左右。

三．任务三

1. 数据分析



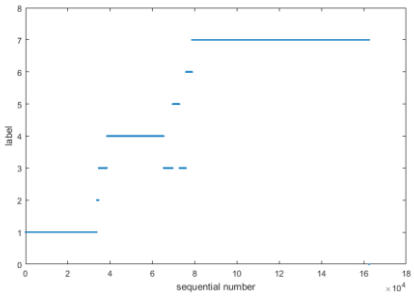


图 3-1-1 1.csv 文件中各方向加速度随时间的变化及其标签

根据图 3-1-1 和图 3-1-2，我们可以发现，单纯从各方向加速度的变化图来看，该时序数据中存在明显的不同模式，但是仅凭观察，可以总结出近三种不同的模式，结合对应的 label 图，不难发现，许多类别序列的特征很难分辨，比如序列开始阶段的第 1 类状态和序列结束阶段的第 7 类数据，仅有微弱区别。

2. 单一活动类别识别

假设已知一段序列属于一个活动类别，可以凭借序列的特征对各序列进行分类。这是典型的时序数据分类的问题。目前常用的方法有：1）基于距离的序列匹配方法，如 DTW；2）基于特征的机器学习分类方法，即对提取出的特征应用决策树，SVM，神经网络等分类方法进行分类。关于时序数据的特征，又主要分为三类：1）基于统计的特征，如一段序列的均值，方差，标准差，众数，中位数和相关函数，滑动平均等等；2）基于时频变换方法的特征，对时序数据进行傅里叶，拉普拉斯变换，将其对应频域的特征进行提取，用于数据分类；3）基于回归模型的特征，主要针对于随机过程，尤其是平稳性易于证明的过程，通过拟合回归对应的模型，将模型的参数作为数据的特征，如 ARMA，ARIMA 等模型。

通过上述分析，我们你采用基于特征学习的方法对序列进行分类。

A. 特征提取

1）通过计算该序列的平稳性，其原序列和一阶差分序列均不平稳，所以不采用回归模型参数估计的方法进行特征提取。

2）我们对文件 1.csv 中的 x 方向上的加速度数据进行按 label 分类，绘制其频谱图，如下图 3-1-2，3-1-3，3-1-4，3-1-5 所示：

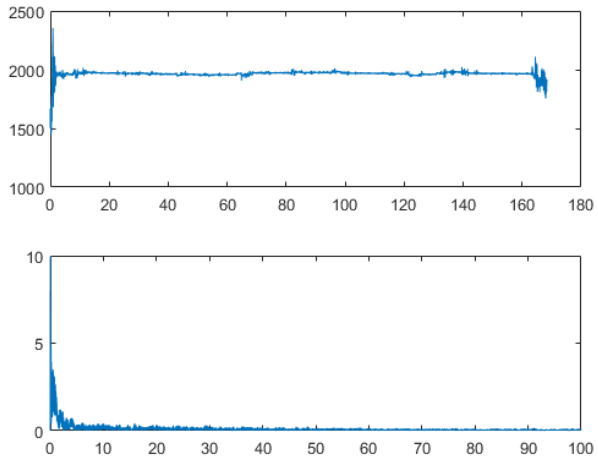


图 3-1-2 label 为 1 的序列及其频谱图（上面的图的横纵坐标分别是序列编号和加速度值；下面的图的横纵坐标分别是频率和幅值）

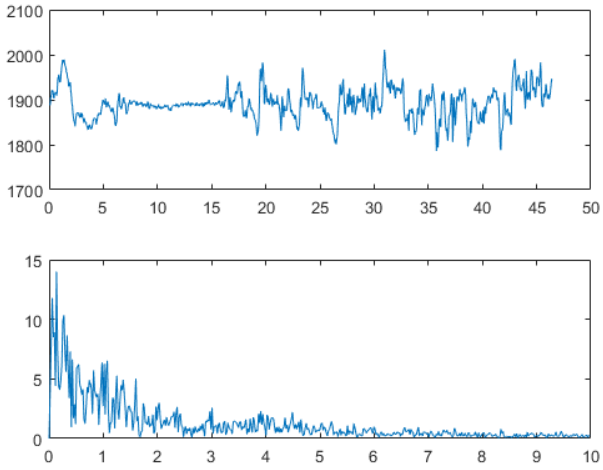


图 3-1-3 label 为 2 的序列及其频谱图（上面的图的横纵坐标分别是序列编号和加速度值；下面的图的横纵坐标分别是频率和幅值）

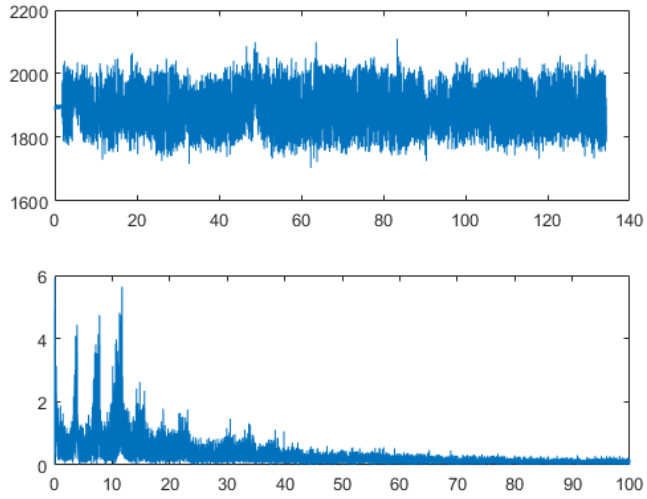


图 3-1-4 label 为 4 的序列及其频谱图（上面的图的横纵坐标分别是序列编号和加速度值；下面的图的横纵坐标分别是频率和幅值）

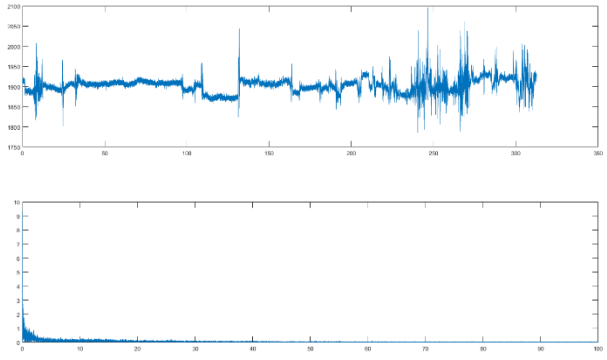


图 3-1-5 label 为 7 的序列及其频谱图（上面的图的横纵坐标分别是序列编号和加速度值；下面的图的横纵坐标分别是频率和幅值）

根据频谱图我们发现，不同类别的序列中，仅 label 为 4 的序列的频

谱在大于 10 的频段有明显的幅值，其他序列的频谱衰减十分迅速，仅在频率范围为 0-5 之间有明显可见的值，且十分相似。用决策树算法尝试后仅能得到 28%的分类准确率。

因此，频域的数据特征不足以用来区分不同类别的序列。

3）由于数据的采样频率为 52Hz，我们将 52 个数据点即 1s 内的数据划分为一个序列，形成维度（52，3）矩阵形式。计算该序列每个方向的平均值，方差，最大值和最小值，相关函数等统计量。用决策树，随机森林，逻辑回归等算法均能得到比依据原始数据分类更高的准确率。

	x	y	z	空间方向
平均值	ave_x	ave_y	ave_z	sqrt（ave_x^2+ave_y^2+ave_z^2）
最小值	min_x	min_y	min_z	sqrt（min_x^2+min_y^2+min_z^2）
最大值	max_x	max_y	max_z	sqrt（max_x^2+max_y^2+max_z^2）
方差	var_x	var_y	var_z	sqrt（var_x^2+var_y^2+var_z^2）
相关函数	ate_x(10)	ate_y(10)	ate(10)	

表 3-1-1 统计特征表

B. 分类算法比较

1）算法运行环境：处理器 i5-3470 内存 8GB

2）算法测试精确度比较

图 3-1-6 三种分类算法在不同统计特征下的精确度

可见在特征相同的情况下，逻辑回归方法优于随机森林优于决策树算法。同时统计特征中相关函数对逻辑回归方法的结果影响很小。逻辑回归的最高准确率为 59.65%。相比基于原始数据的平均值这一特征的分类结果 34.16%提高了 25 个百分点。

3. 活动转换点检测

活动转换点的检测主要有两种方法，一种是基于序列比较的滑动窗

口，自底向上，自顶向下等方法，另一种是基于概率模型，即最常用的隐马尔科夫模型的方法（hmm）。

首先我们尝试通过无监督学习的方法即 hmm 模型检测转换点。但是由于元数据存在大量的震荡，高频数据幅值低，但是频域宽度大，因此用元数据作为 hmm 模型的输入时，输出的状态预测结果仍然会有许多高频成分。

1）对原始数据进行滤波

调用 scipy.signal 模块中 iirdesign 设计低通滤波器参数，最后选取的参数为通带频率 0.001，阻带频率 0.002，通带最大衰减 1dB，阻带最小衰减 40dB，应用切比雪夫滤波。

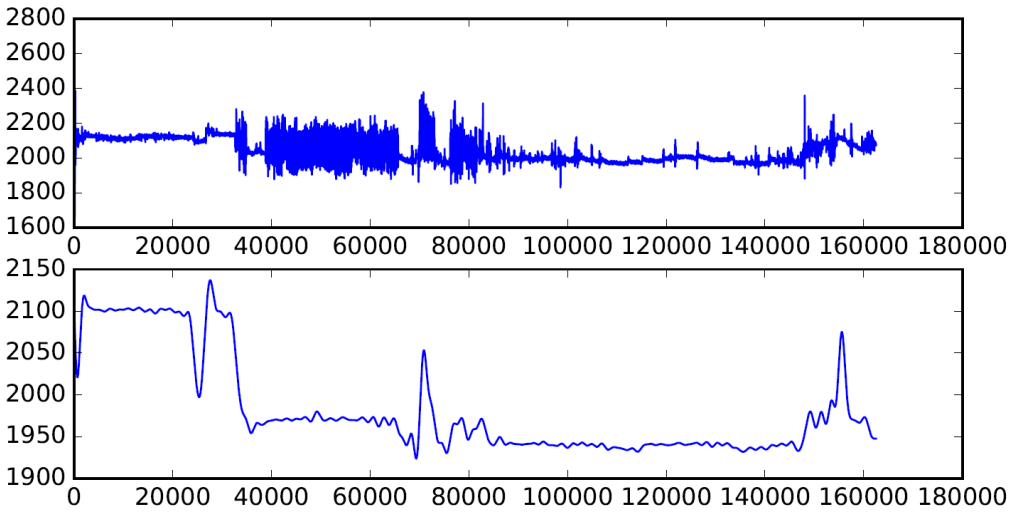


图 3-2-1 1.csv 中的 z 方向的数据及其滤波后的情况

2）在未滤波的数据上进行状态转移点检测，除去 label 为 0 的个别点，我们将数据分为 7 个类别。下面三幅图中最上面的图是文件中的数据标签，我们可以认为一条竖直线就是一个转移点。中间图是用 HMM 检测出的数据状态，可见能够识别出所有的状态转移点，但是也会产生很多并非真实转移点的情况，造成极大的干扰。下面的图是对 HMM 的输出进行滤波后将结果进行 7 等分得到的状态图，相比之下丢失了大量真实的转移点。这里对 HMM 输出状态序列滤波的参数是，通带 0.005，阻带 0.001，通带最大衰减 1dB, 阻带最小衰减 40dB。

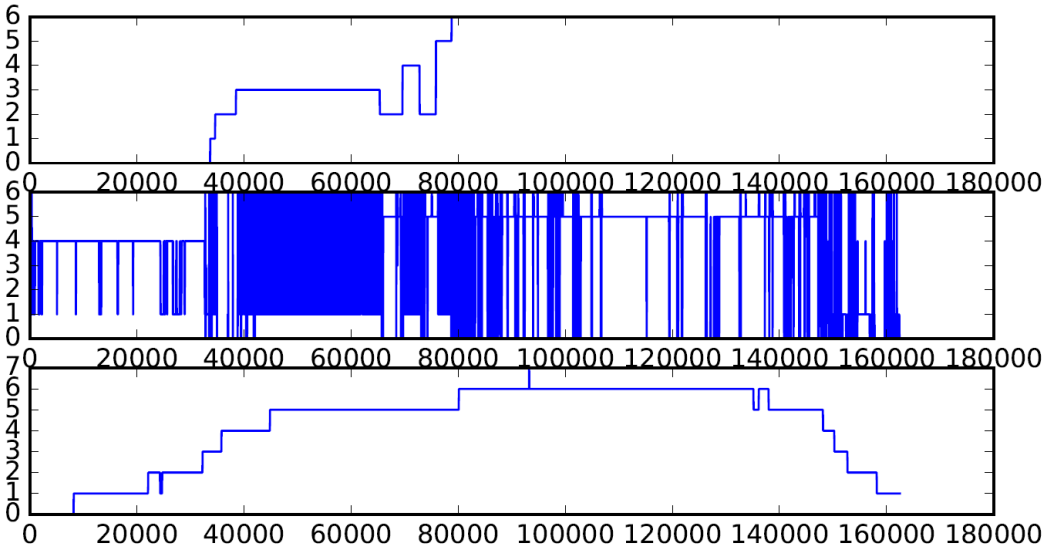


图 3-2-2 1.csv 中原始数据 HMM 转移点检测

3）在滤波后的数据上进行 HMM 状态分类的结果：

下面三幅图中最上面的图依然是文件中的数据标签，我们可以认为一条竖直线就是一个转移点。中间图是用 HMM 检测出的数据状态，可见能够识别出所有的状态转移点，但是也会产生很多并非真实转移点的情况，造成极大的干扰。下面的图是对 HMM 的输出进行滤波后将结果进行 7 等分得到的状态图，相比之下丢失了大量真实的转移点。这里对 HMM 输出状态序列滤波的参数与 2）中一致 。

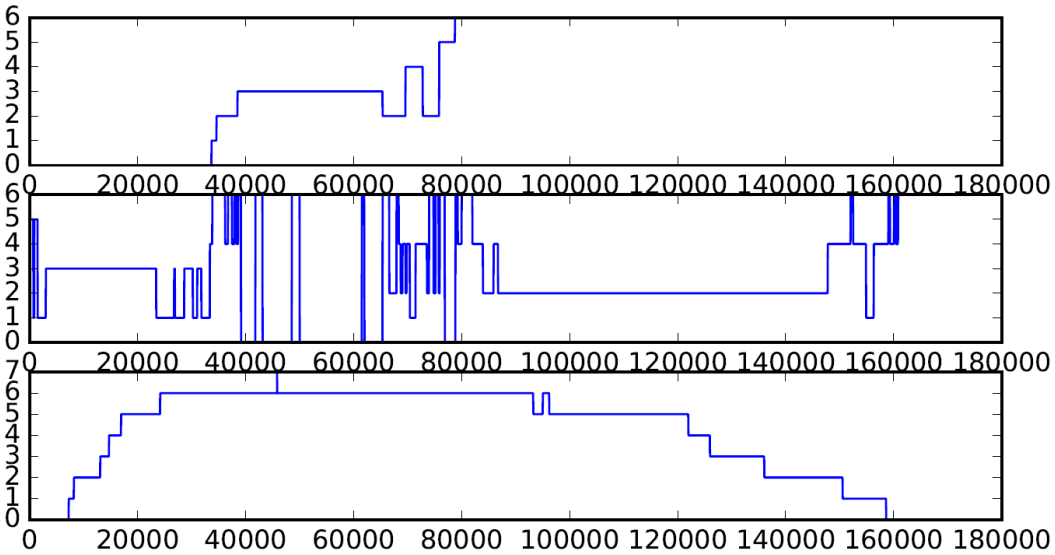


图 3-2-3 1.csv 中滤波后 HMM 转移点检测

4）应用 HMM 得到的状态转移点

可见基于 HMM 得到的结果震荡依然严重，虽然包括了所有的状态转化点，但是不进行滤波的话会有大量假转换点产生，滤波之后又会失去一些真实转换点，很难达到折中的平衡结果。在这种情况下，我们选择使用滤波后的数据输入到 HMM 模型中，获取状态分类结果，不进行

进一步滤波，直接输出其所有转移状态，即遵循不漏掉任何真实转移点的原则。

同时，我们又做了另外一个假设：我们默认每一秒内的状态是不会发生改变的，利用序列分类的方式，将到来的序列按照 52 个数据点为一个序列进行切割，对每一个序列进行分类，这样即使原本不是同一状态的序列被切割到一类序列中，仅产生 1s 的误差也是可以接受的。

用该方法获得的序列状态如图所示：测试集的所有数据组成的。

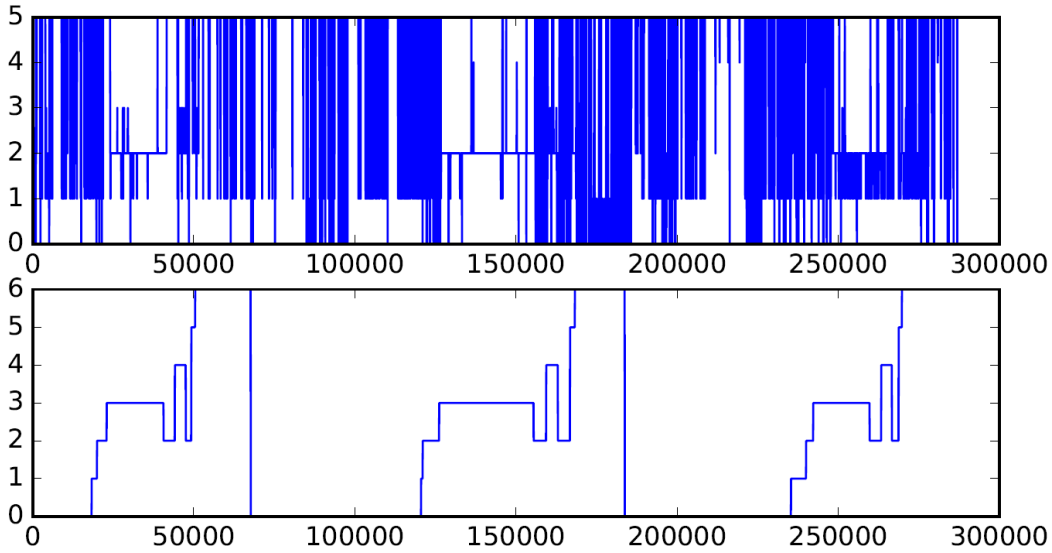


图 3-2-4 13-15.csv 数据按 1s 间隔进行序列判断得到转移点

4. 多活动类别识别

对于一个序列的多活动类别识别有两种方法，一是利用上面的结论，首先对数据进行状态转移点的检测，再对检测出的每一个单一类别进行单一活动分类。我们用 HMM 的结果和逻辑回归分类方法进行了尝试，整体单个时间点的分类准确率只有 15.0080256822 %。但是某些段内的准确率很高。列举一些较高准确率的类别如下：

段标签	预测准确率
0	100%
0	23.0765%
2	62.162%
4	100%
5	20%

另一种方法是对每一个时间点的状态进行判断，即不依赖于序列的特征，我们设计了一个 4 层前向神经网络模型，神经网络参数为：层数 4，对应的每层节点数分别为：100，50，20，10；学习率 0.01，最大迭代次数 4000。对原始数据进行训练，每一个时间点输出一个活动类型，结果 43.27%。

图 3-3-1 四层神经网络训练过程的 loss 衰减