华中科技大学

本科生毕业设计[论文]

基于工业大数据的故障诊断模型设计

院　　系　　机械科学与工程学院

专业班级　　机械 1401 班

姓　　名　　张照博

学　　号　　U201410606

指导教师　　金海、吴波

年　　月　　日

学位论文原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包括任何其他个人或集体已经发表或撰写的成果作品。本人完全意识到本声明的法律后果由本人承担。

作者签名：　　年　月　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保障、使用学位论文的规定，同意学校保留并向有关学位论文管理部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权省级优秀学士论文评选机构将本学位论文的全部或部分内容编入有关数据进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于 1、保密口，在　　年解密后适用本授权书

2、不保密口　。

（请在以上相应方框内打“√”）

作者签名：　　年　月　日

导师签名：　　年　月　日

I

摘　要

由于计算机硬件以符合摩尔定律的速度迅猛发展，计算机数据存储、数据传输和分布式计算的成本都大幅度降低。而现代化的工厂中，往往都布置有大量的传感器，而且随着存储成本的降低，读取到的设备信息变得更加丰富，由此便会产生海量的工业数据。

对于这一变化，全世界许多国家都相继提出相应的举措。最初是德国提出了“工业 4.0”的概念，之后美国也随之推出“工业互联网”，我国也相继推出“中国制造 2025”的概念，其核心都指向智能制造。而工业大数据技术则是这些内容中的核心部分。

在工业设备的运行过程中，自然磨损、设备超载、操作不当等多种原因会导致设备的性能发生下降，甚至于产生故障或者是异常。而通过对设备加装传感器进行监控，获取到设备的实时信息并且加以梳理计算，就可以得到设备各个部位的实时运行状态，从而实现对设备的监控。而如果出现了设备故障现象，则可以通过对历史数据进行数据挖掘、清洗形成故障模型，导入设备的最新运行数据进行故障诊断。

设备的故障诊断方法可以按照诊断依据分为三种:基于机理模型的方法，基于数据驱动的方法，基于知识工程的方法[1]。本文将采用基于数据驱动的方法中的基于分类的方法进行故障模型的构建。同时为了对比不同分类方法的性能，本文采用了两种分类方式进行比较。

本文主要研究工作和成果如下：

（1）建立了以决策树算法为基础的故障诊断模型；

（2）实现了以支持向量机为核心的数据驱动方法。对支持向量机的核心原理进行了研究，以风力涡轮机齿轮箱的健康数据、故障数据进行了 SVM 的分类检测，实现了简单的故障诊断；

（3）比对了两种策略的精度以及其他的一些的性能度量。

关键词：故障诊断；工业大数据；数据驱动；决策树；支持向量机

Abstract

As the computer hardware rapidly develops at a rate consistent with Moore's Law, the cost of computer data storage, data transmission, and distributed computing has been greatly Reduced. In the modern factories, a large number of sensors are often arranged, and as the cost of storage decreases, the read device information becomes more abundant, and thus a large amount of industrial data will be generated.

For this change, many countries around the world have put forward corresponding measures in succession. Initially, Germany proposed the concept of "Industry 4.0". After the United States introduced the "Industrial Internet," China has also successively introduced the concept of "Made in China 2025." Its core points to intelligent manufacturing. Industrial big data technology is a core part of these contents.

During the operation of industrial equipment, natural wear, equipment overload, improper operation, and other reasons can cause the performance of the equipment to drop, and even result in failure or abnormality. By adding sensors to the equipment for monitoring, obtaining real-time information from the equipment and combing calculations, the real-time running status of various parts of the equipment can be obtained, thereby realizing the monitoring of the equipment. If there is a device failure, you can perform data mining and cleaning on the historical data to form a fault model and import the latest operating data of the device for fault diagnosis.

The equipment fault diagnosis methods can be divided into three types according to the diagnosis basis: based on the mechanism model method, based on the data-driven method, based on the knowledge engineering method. This article will use a classification-based approach based on a Data-Driven approach to build a fault model. At the same time, in order to compare the performance of different classification methods, this paper uses two classification methods to compare.

The main research work and achievements of this article are as follows:

(1) Establish a fault diagnosis model based on decision tree algorithm, and study the differences between ID3 and C4.5;

(2) A Data-Driven approach based on Support Vector Machines(SVM) is implemented. The core principle of SVM is studied. The classification and detection of SVM is performed based on the health data and fault data of the wind turbine gearbox, and a simple fault diagnosis is realized.

(3) Compare the running time of two strategies, the accuracy rate of fault diagnosis, and some other performance metrics.

Key Words： Fault Diagnosis; Industrial Big Data; Data-Driven Method; Decision Tree; Support Vector Machine

目　录

1 绪论

1.1 选题背景和意义

在计算机行业还未能发展到如今这般规模的时候，人们只能选择抽样的数据、局部的数据和片面的数据，纯粹靠经验、理论、假设和价值观去发现、理解未知领域的规律。而这样做的结果，就是对真实现象的抽象归纳与演绎推理，这就不可避免的包含了各种主观上的因素。同时由于样本的局部性，很多推理归纳出来的结果与实际现象具有极大的偏差。

而如今的所谓大数据，通常都指数据量在“太字节（TB）”即 2 的 40 次方以上，一般情况下难以收集、存储、管理以及分析的数据。而且随着科技进步，大数据对于“大”的含义还在不断地刷新。但是大数据不仅仅只关乎于数据量的大小，而且还与其他的因素有着千丝万缕的关系。

大数据其真正的意义在于：我们可以通过各式各样的传感器，实现与真实世界更加紧密、准确的连接。在得到实时数据后进行整合、挖掘、云计算，去逐步的逼近真实世界，挖掘出那些未曾被我们发现的隐藏规律，建立更加符合真实的数学模型，这是大数据的魅力所在。而在大数据的庞大篇幅中，工业大数据占据着相当重要的地位。工业大数据是智能制造的关键技术，它是联通物理世界与信息世界的桥梁，是推动生产型制造向服务型制造转型的动力之一[2]。

而在工业领域，工业大数据的一大发展方向就是故障诊断，数据的产生和记录贯穿于一台设备从投入生产到损耗的全过程，而通过一定数量的智能传感器，我们可以监控某些生产设备的所有信息，使设备在生产线的实时状态远程监控成为可能，这一方面改善了工作人员的工作环境，另一个更为重要的方面就是提高了设备发生故障或者异常时候的反应速度以及排除故障的效率。

伴随着工业生产水平发展的突飞猛进，工业设备精度越来越高，结构越来越复杂，所以在车间内很多设备的故障都未能得到及时的发现和解决，这

一点很容易对工厂造成巨大的损失.由于设备愈加复杂所导致的设备故障信息数据呈现指数型增长，而运行时所产生的海量数据，采用传统基于机理模型的方式已经很难负载如此巨量规模的数据分析，进行故障诊断了。

此外，工业设备结构极其复杂，不同模块之间可能会产生故障的交集，人工分析或者是传统的先验知识故障检测手段已经无法准确、迅速的完成故障的诊断。因此，结合工业大数据对工业设备所产生的海量数据进行数据挖掘分析建立故障诊断模型，对于提高设备维护效率、迅速有效解决故障、降低维修费用有巨大意义。

1.2 国内外研究现况及发展趋势

1.2.1 国内研究现状

金风科技公司对大量风场的历史故障、异常运行信息进行大数据挖掘与分析，对 SCADA(Supervisory Control and Data Acquisition，监控和数据采集系统)数据的提取、清洗、分析，从而得到了关于电机与桨叶的一系列特征曲线与模态数据，建立了多角度的断裂预警模型，成功实现了提前 90h 的预测性维护，消除了潜在的重大故障隐患，保障了工作人员与设备的安全[3]。南京航空航天大学的张鹏在原有的线性模型基础上进行了大量改进，提出了将卡尔曼滤波器和基于非线性模型相结合的方法，并且将其理论在航空发动机上进行了验证[4]。西安电子科技大学的钟福磊等人在盾构机上建立了基于先验知识与基于数据驱动两种方式的混合模型，并且利用仿真数据进行了验证。南京航空航天大学鲁峰等人对发动机进行了仿真建模并且获得了影响系数矩阵，成功实现了对发动机中的气路故障进行诊断[5]。

1.2.2 国外研究现状

国际权威专家 Frank 将故障诊断的方法总结为三种:基于机理模型的方法、基于数据驱动的方法、基于知识工程的方法[6]。葡萄牙科英布拉大学的 Marco S. Reis 教授和 Geert Gins 在《Industrial Process Monitoring in the Big Data/Industry 4.0 Era: From Detection, to Diagnosis, to

Prognosis》一文中提到，过去的重点都是检测，也就是基于机理模型的一种对比当前数据的检测办法，实现高水平的检测速度和强度是过去 IPM 研究的主要重点。在处理新流程时，这是一个必要的步骤，但是随着时间推进，越来越多的因素阻碍了更先进的监控手段的发展[7]。因此为了解决这些障碍与挑战，我们需要找到一种方法可以找到故障根源，也就是我们当今时代的主流过程监测手段：故障诊断。未来的发展方向更是令人心驰神往，故障预检测，能够根据当前的运行数据获取未来一段时间内的设备运行状态预测。

瑞典吕勒奥理工大学的 Lianwei Zhang 开发除了一套专用于大数据检测以及维护的系统以及一种基于自适应核密度的异常检测（Adaptive-KD）方法，在工业场景中具有极大的使用价值。

1.3 主要研究内容

（1） 研究数据挖掘算法，采集数据进行模型构建，并且可以根据新的数据进行模型改进、重构；

（2） 研究数据处理方法，如离散化等方法提高故障模型的精度；

（3） 设计人机交互界面，提供显示、检索等功能；

（4） 研究模型的改进方案，如决策树中的“剪枝”方法，对模型进行精简，减少模型构建所消耗的资源。

2 故障诊断的总体设计方案

2.1 故障模型的要求

故障模型应该是基于历史数据构建的，由于我们的模型是基于数据驱动，所以对于机械设备方面的先验知识需求量远小于基于机理模型和基于知识工程的构建方式。另外，我们的模型要能接受新的运行数据，对于数据进行测试，从而进行故障诊断的最终目的。

在精度上，模型的精度应该随着数据的不断完善而改进，构建模型所使用的数据越多，那么我们的模型的泛化程度就越高，对于各种实际情况的解读能力就会进一步提升。另外故障模型的基础是基于工业大数据，对于工业大数据的各个环节都会在具体实现环节中一一对应。

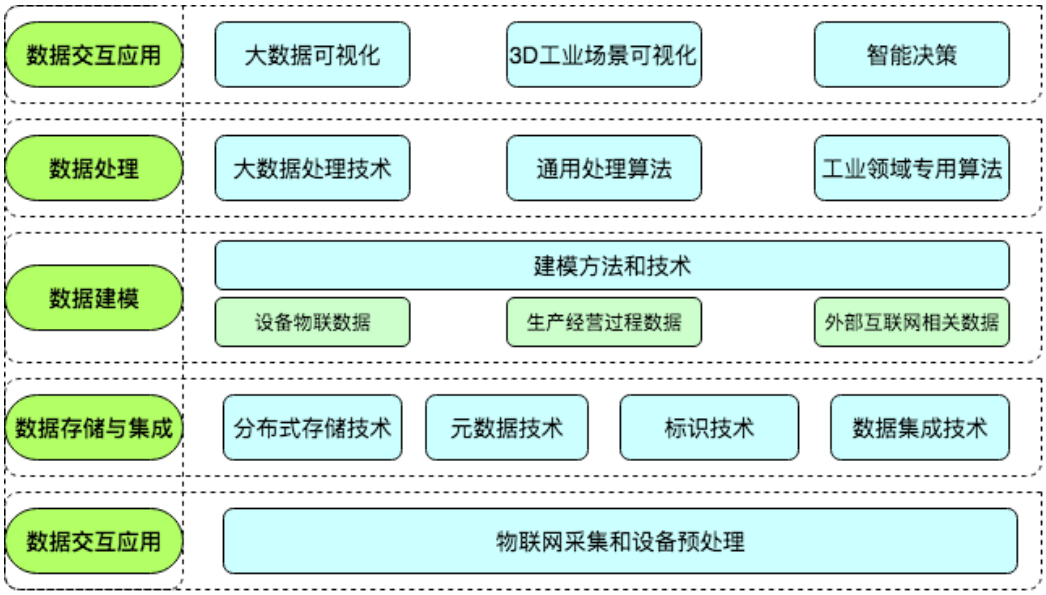

图 2-1 工业大数据技术架构

2.2 决策树建立故障树模型

2.2.1 信息熵

假设当前样本集 D 中有 N 个样本，而整个样本有 k 个分类，每个分类对应的样本数量为N_i，那么对于每个分类，他们各自占据的信息量（也可以理

解为样本分类的频率）为：

$$\mathrm{P}(\mathrm{x_i}) = \frac{N_i}{N} \quad (i = 1,2,3 \cdots k) \qquad (2-1)$$

结合每一个分类的信息量，则此样本总体的信息熵为：

$$\mathrm{Ent(D)} = -\sum_{i=1}^{k} \mathrm{P(x_i)} * \log\big(\mathrm{P(x_i)}\big) \qquad (2-2)$$

2.2.2 信息增益

假设当前样本集 D 中有 N 个样本，每个样本都有一些属性，假设我们目前取属性 A 作为我们计算信息增益的属性。

根据属性 A，我们可以用属性 A 的不同取值（假设有 v 种），将整个样本集 D 分为 v 个子样本集$\mathrm{D_i(i = 1,2,3 \cdots v)}$，每个样本子集的样本数为 N，那么每一个样本子集的频率为：

$$\mathrm{P(D_i)} = \frac{\mathrm{Ni}}{N} = \frac{\mathrm{|Di|}}{\mathrm{|D|}} \quad (\mathrm{i = 1,2,3 \cdots v}) \qquad (2-3)$$

那么该样本集的 A 属性的信息增益即为：

$$\mathrm{Gain(D,A)} = Ent(D) - \sum_{i=1}^{v} (P(D_i) * Ent(D_i) \qquad (2-4)$$

2.2.3 ID3 算法

ID3 是一种以自顶向下递归的方法构造决策树的贪心算法。其决策树的基本生成策略如下:

（1）树以整体样本作为单个节点开始；

（2）如果当前节点中，所有的样本都属于同一类，则该节点成为叶节点，并标记为当前样本的类；

（3）否则，使用前面提到的信息增益作为判断信息，选择信息增益最大的一个属性作，该各个属性值将成为该节点往下进行分支的依据。在这里，我们假设所有的属性都是可分的，即所有的属性都可以在一个离散值集合内取到，如果该属性为连续值，则该属性必须离散化处理[8]；

（4） 基于测试属性的每一个属性值创建一个分支，并将该属性值相同的样本划分到该分支对应的样本子集；

（5）使用类似的方法，递归地形成每个分支延伸对应的样本决策树。一旦一个属性出现在一个节点上，就不必在该节点的后代上考虑这个属性。

ID3 算法虽然简单易用，但是也有很多缺陷：

（1） ID3 算法缺乏对于连续值的处理手段，而在现实生活中，很多的应用环境都是采集到的连续值；

（2）计算信息增益的时候对于样本频率 $P(x_i)$ 有极大的依赖性，有时候会对模型造成很大的偏差；

（3）对噪声较为敏感，所谓噪声也就是一些在生成模型的时候就给定的错误数据；

（4）采用递归的方式形成模型，而且整个决策树的生成过程对于数据多次读取存写，所以算法较为低效，而且无法应用于大数据量的场合下。

下面是利用这些概念获得故障树的过程：

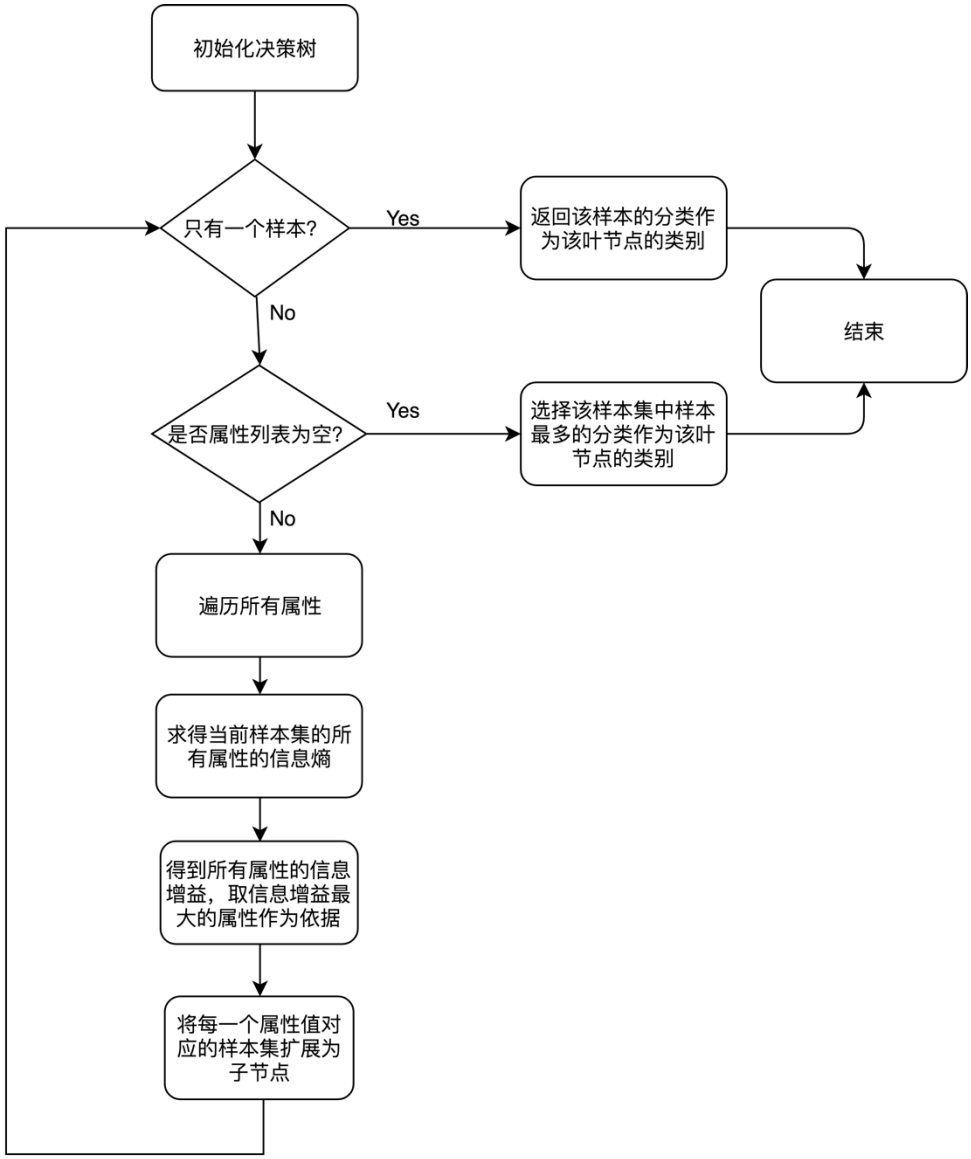

图 2-2　ID3 算法生成决策树流程图

为了提高决策树的模型精度，去除掉模型创建时一些错误数据的干扰，有两种剪枝方法可以用于提高决策树的正确分类能力：

（1）预剪枝方法(prepruning)，该方法通过提前停止树的向下延伸而对树剪枝。在各个节点向下分支之前，判断通往该分支的样本子集中的判断正确率进行对比，如果在当前节点的正确率高于分支后的子节点，那么就停止生长，这就是预剪枝的主要思想。该方法很多的分支都未曾“展开”，降低了过拟合的风险，而且还显著减少了决策树的训练时间和花销；但是另外一方面，一些分支虽然不能提升整体的泛化性能，但是由其再次展开的分支却有可能导致性能显著提高，而且预剪枝的“贪心”本质给这种方法带来了欠

拟合的风险。

（2）后剪枝方法(postpruning)，顾名思义，该方法是预先生成一颗完整的决策树，然后从每一个叶节点往上查看父节点，计算如果该父节点进行剪枝成为叶节点后是否会提高判定精度来决定是否剪枝。这种方式直到无法提高决策树性能为止。对比预剪枝方法，后剪枝方法保留了更多的分支。一般情况下，后剪枝方法的欠拟合风险比较小，泛化性能优于预剪枝方法生成的决策树。但是由于后剪枝方法是在决策树生成后在进行的，所以在训练时间和花销上会比预剪枝方法高得多。

2.3 支持向量机二分类原理

2.3.1 SVM 原理

支持向量机（Support Vector Machine，SVM）是一种经典的二分类模型算法，基本模型定义为特征空间中最大间隔的线性分类器，其学习的优化目标便是间隔最大化，因此支持向量机本身可以转化为一个凸二次规划求解的问题。

对于二分类的支持向量机学习器，假设数据是线性可分的，这时分类学习的目标就是找到一个合适的超平面来将所有样本分割成两个类别，将不同类别的样本分割到两个区域内。

下图中的点集是低维数据样本表示，而这些数据对应的超平面就是中间那根线。对于点来说，线毫无疑问是处于高维的超平面了。随着维数增加，超平面总是比数据的维数多一维。

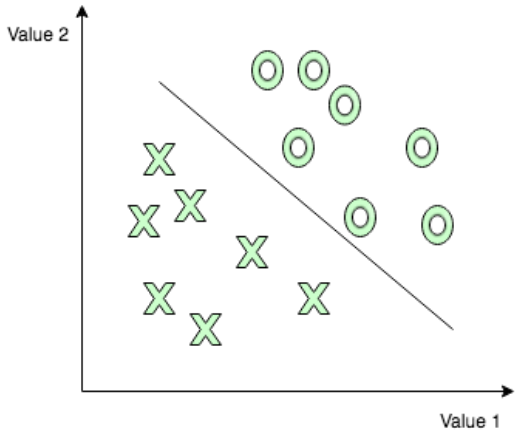

图 2-3 二维超平面

但是这样的超平面可能存在多个，我们应该寻找的最优超平面该如何获取呢？

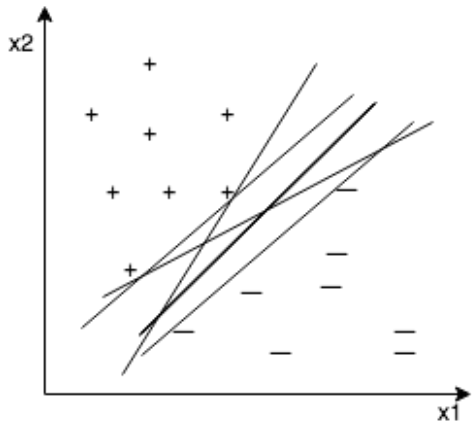

图 2-4 存在多个划分超平面将两类训练样本分开

直观上看的话，我们应该寻找位于两类训练样本“正中间”的超平面作为我们的最优超平面，因为这个超平面对于整个训练样本局部扰动的“容忍”性能最好。换言之，这个超平面所产生的分类结果是最鲁棒的，对于未知的数据的分类能里最强。

在样本空间中，超平面可以通过如下线性方程表示：

$$\boldsymbol{\omega}^{\mathrm{T}}x + b = 0 \qquad (2-5)$$

其中 $\boldsymbol{w}$ = （w1;w2;w3;...;w_d）是当前数据产生的超平面的法向量，它

决定了超平面在高维空间内的方向。b 为超平面的位移，决定了超平面与其所在空间的原点之间的距离。既然有了可以用数学方式表达的超平面，那么样本空间中任意一点到超平面的距离也就可以表示为：

$$r = \frac{|\boldsymbol{\omega}^{T}x + b|}{\|\boldsymbol{\omega}\|} \tag{2-6}$$

假设超平面能够正确分类，即对于（x_i, y_i） ϵ D，则有：

$$\begin{cases} \boldsymbol{\omega}^{T}x + b \geq +1 , y_i = +1; \\ \boldsymbol{\omega}^{T}x + b \leq -1 , y_i = -1; \end{cases} \tag{2-7}$$

经过计算，我们可以筛选距离超平面最近的几个训练样本，使得上式中的等号成立，这几个样本就称之为支持向量，两个异类（在超平面两侧）支持向量到超平面的距离之和为：

$$\gamma = \frac{2}{\|w\|} \tag{2-8}$$

这个距离就被称作“间隔”（margin）：

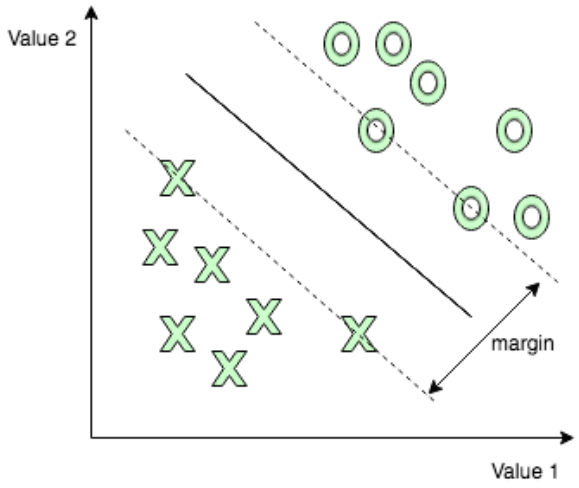

图 2-5 支持向量与间隔

显而易见的，我们的最终目标，就是找到具有最大间隔的超平面作为最优超平面。那么此超平面需要符合下列特点：

$$\max_{w,b} \frac{2}{\|\boldsymbol{\omega}\|} \tag{2-9}$$

$$\text{s.t.} \quad y_i(\ \boldsymbol{\omega}^T x\ +\ b\ )\ \geq\ 1, \quad i = 1,2\ldots,m;$$

由此可知，为了得到具有最大间隔的最优超平面，只需要最小化$\|w\|^2$即可。于是，上式可以改写为：

$$\min_{w,b}\ \frac{1}{2}\ \|\boldsymbol{\omega}\|^2 \tag{2-10}$$

$$\text{s.t.} \quad y_i(\ \boldsymbol{\omega}^T x_i\ +\ b\ )\ \geq\ 1, \quad i = 1,2\ldots,m;$$

这就是“支持向量机”（Support Vector Machine，SVM）这一方法的基本型。

2.3.2 对偶问题

由支持向量机方法的基本型，我们可以知道这是一个带约束的凸二次规划问题，解决这个问题的较高效的办法是：转化为对偶问题（dual problem）。

具体的解决方式为给每条约束添加拉格朗日乘子 $\partial_i \geq 0$，则上述问题的朗格朗日函数可写为：

$$\boldsymbol{L}(\boldsymbol{\omega}, b, \alpha) =\ \frac{1}{2}\ \|\boldsymbol{\omega}\|^2\ \pm\ \sum_{i=1}^{m} \alpha_i \big(1 - y_i(\omega^T x\ +\ b)\big) \tag{2-11}$$

其中$\alpha_i = (\alpha_1; \alpha_2; \ldots; \alpha_m)$，由拉格朗日乘数法的思想，令$L(\omega, b, \alpha)$对$\omega$和$b$的偏导为0可以的得到：

$$\boldsymbol{\omega} =\ \sum_{i=1}^{m} \alpha_i y_i x_i\ , \tag{2-12}$$

$$\mathbf{0} =\ \sum_{i=1}^{m} \alpha_i y_i\ , \tag{2-13}$$

利用两个式子代入到$L(\omega, b, \alpha)$中可以消去ω和b，就得到了关于基本型的对偶问题：

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \boldsymbol{x_i^T x_j} \qquad (2-14)$$

$$\text{s.t.} \quad \sum_{i=1}^{m} \alpha_i y_i = 0 \ ,$$

$$\alpha_i \geq 0, \quad i = 1,2,3,\ldots,m.$$

通过求解出的α_i，我们可以计算得出$\boldsymbol{\omega}$和 b，从而得到最终具体的模型：

$$f(x) = \boldsymbol{\omega}^T x + b = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x_i^T x} + b \ . \qquad (2-15)$$

$$\omega^* = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x_i} \qquad (2-16)$$

$$b^* = -\frac{\max\limits_{i:y_i=-1} \omega^{*T} x_i + \min\limits_{i:y_i=1} \omega^{*T} x_i}{2} \qquad (2-17)$$

在上述的解答过程中，因为基本型有着不等式的约束条件，所以需要满足 KKT（Karush-Kuhn-Tucker）条件：

$$\alpha_i \geq 0 \ ; \qquad (2-18)$$

$$y_i f(x_i) - 1 \geq 0 \ ; \qquad (2-19)$$

$$\alpha_i (y_i f(x_i) - 1) = 0 \ . \qquad (2-20)$$

于是对于任何的训练样本，都要有$\alpha_i = 0$ 或者 $y_i f(x_i) = 1$的限制条件。且必须是满足$y_i f(x_i) = 1$的样本才会出现在最大间隔边界上，说明这一个样本是支持向量。这一点也就代表着，训练完成后大部分的训练样本都不需要进行保留，只需要留下支持向量的那些样本即可。因为非支持向量的α_i都等于 0，去除之后不会对模型产生什么影响。

模型建立完毕之后，在对新的数据点进行分类时，实际上就是将这个新的数据点x^*代入到分类函数 $f(x) = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x_i^T x} + b$ 中，若 f(x)得出来的结果大于 0，则为正类，否则为负类。

2.3.3 SVM 核函数

假如训练样本线性不可分，放到我们最开始的例子里面的意思就是：我们无法用一条直线将所有的样本正确分类为两类。如下面的“异或”问题：

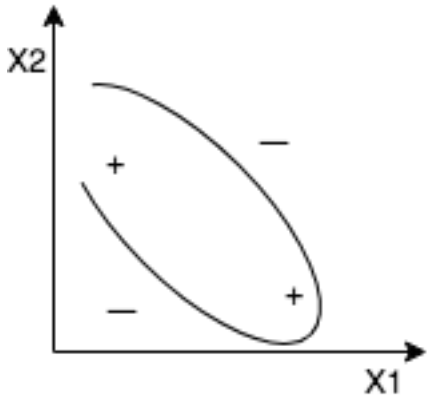

图 2-6 异或问题

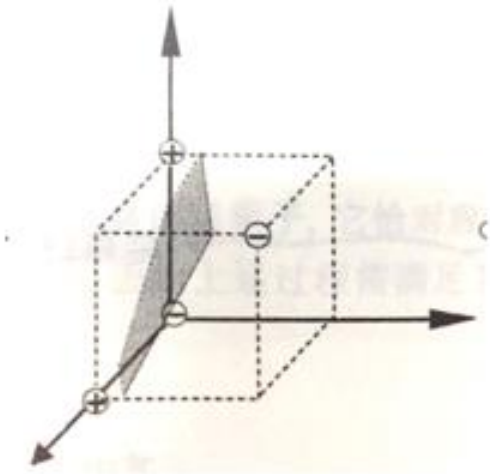

图 2-7 非线性映射

对于这种情况，我们可以将原始空间的训练样本映射到一个更高维度的特征空间，使得样本在这个特征空间内线性可分[9]。

令$\phi(x)$表示将 x 映射到高维空间中的特征向量，那么对比前面线性可分的模型表示，同理可以得到相应的高维空间模型为：

$$f(\mathbf{x}) = \omega^{\mathrm{T}}\phi(x) + b \qquad (2-21)$$

同理可得：

$$\min_{w,b} \ \frac{1}{2} \ \|\boldsymbol{\omega}\|^2 \qquad (2-22)$$

$$\text{s.t.} \quad \mathrm{y_i}(\ \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\phi}(x_i) \ + \ b\ ) \ \geq \ 1, \quad \mathrm{i} = 1,2\ldots,\mathrm{m}.$$

同样可以得到对偶问题，如下：

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_{\mathrm{i}} - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_{\mathrm{i}}\alpha_j y_i y_j \boldsymbol{\phi}(\boldsymbol{x_i})^{\boldsymbol{T}}\boldsymbol{\phi}(\boldsymbol{x_j}) \qquad (2-23)$$

$$\text{s.t.} \quad \sum_{i=1}^{m} \alpha_{\mathrm{i}} y_i = 0 \ ,$$

$$\alpha_{\mathrm{i}} \geq 0, \quad \mathrm{i} = 1,2,3,\ldots,\mathrm{m}.$$

映射后的特征空间维数可能会很高，会极大地加大我们训练时间和内存开销，直接计算$\boldsymbol{\phi}(\boldsymbol{x_i^T})\boldsymbol{\phi}(\boldsymbol{x_j})$ 不一定可行。为了避开这个不稳定的区域，我们可以假设存在这样一个函数：

$$\kappa(x_i, x_j) = \ <\phi(x_i), \phi(x_j) = \ \ \phi(x_i)^T\phi(x_j) \qquad (2-24)$$

即在高维特征空间中映射向量的内积，与在原始样本空间中通过核函数$\kappa(\mathrm{x_i}, \mathrm{x_j})$计算后的结果等效。这就是“核技巧”。上述式子可以改写为：

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i\alpha_j y_i y_j \kappa(x_i, x_j) \qquad (2-25)$$

$$\text{s.t.} \quad \sum_{i=1}^{m} \alpha_{\mathrm{i}} y_i = 0 \ ,$$

$$\alpha_{\mathrm{i}} \geq 0, \quad \mathrm{i} = 1,2,3,\ldots,\mathrm{m}.$$

求解之后可以得到：

$$\mathrm{f}(x) = \ \ \boldsymbol{\omega}^T\phi(x) + b = \ \sum_{i=1}^{m} \alpha_i y_i \kappa(x_i, x_j) + b \ . \qquad (2-26)$$

在线性不可分的情况中，核函数的选择是影响 SVM 模型的性能至关重要的因素。在明确特征映射的形式之前，我们并不知道具体选择哪一种核函数，

而且核函数的选择也隐式的定义了特征空间。如果在模型的构建过程中选择了错误的核函数，那么由此建立的支持向量机模型的性能将会下降许多。

下列几种常用的核函数：

表 2-1 常用核函数表达式及其参数说明

名称	表达式	参数
线性核	$\kappa(x_i, x_j) = x_i^T x_j$	
多项式核	$\kappa(x_i, x_j) = (x_i^T x_j)^d$	$d \geq 1$ 为多项式的次数
高斯核	$\kappa(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$	$\sigma > 0$ 为高斯核的带宽
拉普拉斯核	$\kappa(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|}{\sigma}\right)$	$\sigma > 0$
SIGMOD 核	$\kappa(x_i, x_j) = \tanh(\beta x_i^T x_j + \theta)$	Tanh 为双曲正切函数 $\beta > 0, \theta < 0$

此外，函数相互组合也可以得到核函数：

（1） 如果κ_1 和 κ_2 都是核函数，那么对于任意正数$\gamma 1, \gamma 2$其线性组合：

$$\gamma 1\kappa_1 + \gamma 2\kappa_2 \qquad (2-27)$$

也是一个核函数；

（2） 如果κ_1 和 κ_2 都是核函数，那么核函数的直积：

$$\kappa_1 \otimes \kappa_2 \ (x, z) = \kappa_1(x, z) \ \kappa_2(x, z) \qquad (2-28)$$

也是一个核函数；

（3） 如果κ_1 是一个核函数，那么对于任意函数 $g(x)$：

$$\kappa \ (x, z) = g(x)\kappa_1(x, z)g(z) \qquad (2-29)$$

也是一个核函数。

3 具体方案设计

3.1 数据获取

目前主要的数据获取手段是通过互联网上的共享数据集，当前已经获得的数据集合有两个，其中一个用于初步测试，一个用于最终模型构建：

一个是来自罗马的一家通信科学研究所：Semeion Research Center of Sciences of Communication

该数据主要用于测试模型，数据特性如下：

Data Set Characteristics:	Multivariate	**Number of Instances:**	1941	**Area:**	Physical
Attribute Characteristics:	Integer, Real	**Number of Attributes:**	27	**Date Donated**	2010-10-26
Associated Tasks:	Classification	**Missing Values?**	N/A	**Number of Web Hits:**	55309

图 3-1 钢板数据集属性

第二个数据集来自 Github 上一个 Fault Diagnosis 项目的自带的风力涡轮内部齿轮箱数据集。该 Github 项目地址为：Gearboxdata/Gear-Box-Fault-Diagnosis-Data-Set

该数据集内的数据分为两类，即正常运行数据和故障/异常状态下的数据。每一类数据下又按照 0-90 的不同载荷百分比，每 10 个百分点负载一个层次分为 10 种运行状态。合共 20 个文件，一共 2021119 条记录，每条记录包括载荷百分比在内一共 5 个属性。

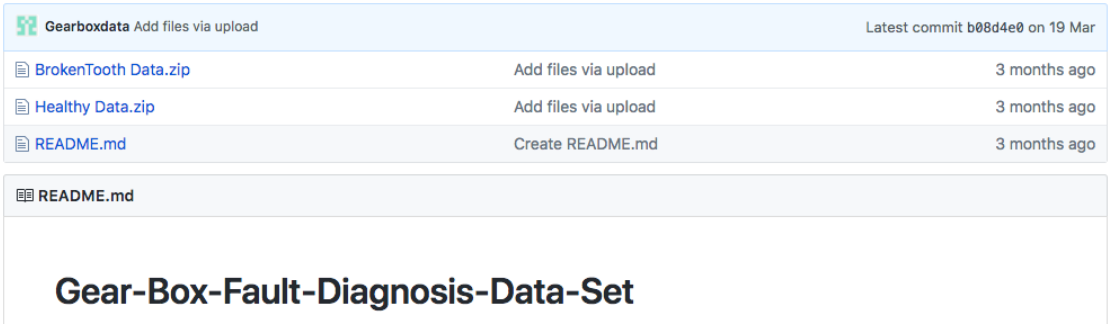

图 3-2 风力涡轮齿轮箱数据集属性

这些数据虽然是连续的，但是经过一定的修改，比如固定为小数点后一位精度，这样可以很轻易的将其离散化，虽然对于精度有一定的影响，但是为了是的数据更为集中，不会出现一条数据记录就是一个分支的情况，离散化势在必行。而且后期新增了基于信息熵的离散化功能，可以很方便的进行区间划分从而提高预测的准确率。

数据整理我采用的是 C++编码来实现的，因为 C++对于数据读取有较好的速度支持，而且格式化能力也比较完善。主要的代码如下(GearData.cpp)：

```
for (int i = 0; i < 10; ++i)
{
    file="/Users/zhangzhaobo/Documents/Graduation-Design/Data/BrokenTooth
Data/b30hz"+hz[i]+".txt";
    ifstream in(file);
    while(in>>data[0])
    {
        out<<setprecision(2)<<data[0]<<"\t\t";
        for (int i = 1; i < 4; ++i)
        {
            in>>data[i];
            out<<setprecision(2)<<data[i]<<"\t\t";
        }
        out<<endl;
    }
    cout<<file<<" is done!"<<endl;
    in.close();
}
```

3.2 数据存取

3.2.1 本地环境

如此大量的数据，采用文本读取这种方式很容易出现错误，所以结合数据库知识，最终选定了 Mysql 数据库作为工业大数据架构的数据存储层。

在安装好 Mysql 之后，建立 Graduation_Design 数据库，在其中建立了 gear 表格作为风力涡轮齿轮箱数据的存储表。表格信息如下：

```
mysql> show columns from gear;
```

```
+----------+------------------+------+-----+---------+-------+
| Field    | Type             | Null | Key | Default | Extra |
+----------+------------------+------+-----+---------+-------+
| id       | int(10) unsigned | NO   | PRI | NULL    |       |
| Sensor1  | float            | NO   |     | NULL    |       |
| Sensor2  | float            | NO   |     | NULL    |       |
| Sensor3  | float            | NO   |     | NULL    |       |
| Sensor4  | float            | NO   |     | NULL    |       |
| Load     | float            | NO   |     | NULL    |       |
| category | int(10) unsigned | NO   |     | NULL    |       |
+----------+------------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

图 3-3 齿轮箱数据存储格式

数据的存入与读取都是依赖于 Java 的一个外部包 mysql-connector-java.jar（JDBC），导入至本地项目后可以调用 JDBC 中的内置类，通过实例化一个数据库连接对象进行数据的存取。为了封装驱动，连接，会话等 JDBC 内容，新建了一个 Mysql_Connect 类提供数据库连接(Connect())，会话(getStatement())，断开连接(Dis_Connect())三个数据库常用的功能。

存储的过程中，由于数据量的问题，如果采用单条记录提交一次的方式进行两百万条数据的存储，那么一共需要两个小时，但是采用 JDBC 自带的批处理操作 Batch，可以将这个时间减少一半。具体操作如下：

```
String INSERT = getInsertQuery(id, Name, line);
statement.addBatch(INSERT);
id++;
count++;
//执行批量执行
if (count>40000) {
    statement.executeBatch();
    count = 0;
}
```

通过批处理操作，每一次与数据库的交互都能提交四万条数据，可以极大地减少 Mysql 连接，认证等的时间花销，提高存储效率。

而读取数据的时候，由于 Decision Tree 算法与 Support Vector Machine 算法需要的数据结构不同，所以定义了两个数据读取类，分别为：

ReadData.java 与 SVMReadData.java，在 ReadData.java 中定义了静态方法 getSelectQuery()提供给所有需要生成查询语句的类调用。

另外每一个数据库读取类都提供了 readTrainData()和 readTestData()两类读取方式，提供给用户有选择的从数据库中读取出指定数量大小的数据记录。

数据读取的时候还需要一个 Parameter 类进行辅助，在这个类里面可以定义训练集与验证机的比例，训练集或者验证集的大小。每一个 Parameter 类内部定义了静态变量：训练样本数，测试样本数，以及二者之间的比例三个变量值。三者之间相互调节，并且因为静态变量的特性作用于全局。这样做的好处是不论在项目中定义多少个 Parameter 实例化对象，只需要在任意处调用调节方法进行修改，就可以直接作用于全局，保证不同方案之间的数据量一致性。

3.2.2 大数据平台环境

【实验环境】 HPC 高性能计算机集群

【操作系统】 Linux RedHat

【硬件配置】 2*8 核 CPU，64GBDDR3 内存，300GB 本地 SAS 磁盘，千兆以太网卡，InfiniBand 网卡，Lustre 文件系统，RedHat Enterprise Linux6.2，内核版本 2.6.32

【计算框架】 Hadoop

生产环境中可以采用计算机组件 HPC 标准集群，基于大数据软件框架进行数据存储、计算任务分摊，缓解巨量数据下的存取、建模压力，这也是工业大数据计算常用的手段之一，符合工业大数据架构中的第二层与第四层的部分技术需求。

本次实验采用了两个计算节点，分别为 node61 和 node39。在两个平台上部署 Hadoop 框架之后，将 node61 设置为主节点，node39 设置为从节点，也就是将 node61 作为 NameNode，node39 作为 DataNode。将数据从本地使用 WinSCP 上传至/home 目录下后即可提交任务至 node61 进行计算。

在本次实验中，利用 Hadoop 使得原始数据中出现次数超过一次的数据在输出文件中只出现一次。实现的原理是：

在 MapReduce 流程中，Mapper 的输出结果<key，value>会经过 hadoop 的 shuffle 过程，聚集成<key，value-list>的形式之后再提交给 Reducer。所以从设计好的 Reducer 输入可以反推出 Map 的输出 key 应为数据，value 任意。继续反推，Map 输出数据的 key 为数据，而在这个实例中每个数据代表输入文件中的一行内容，所以 Map 阶段要完成的任务就是在采用 Hadoop 默认的作业输入方式之后，将属性值与分类值分别保存至 Key 和 Value 中，Mapper 中的结果经过 shuffle 过程之后形成<key,value-list>交给 Reducer。Reducer 运行的时候通过查看每个 key 对应的 value-list 是否有两个值，来确定是否有重复信息。

```
Integer count=0;
int times = 0;
for (IntWritable value : values) {
    count+=value.get();
    times++;
}
if (times==1){
    context.write(new Text(key),new IntWritable(count));
}
else if (count%times==0){
    context.write(new Text(key),new IntWritable(count/times));
}
```

假设 length 代表 value-list 的长度，而 count 代表 value-lst 的所有元素的和。那么我们可以通过简单的判定 count%length 求余是否等于 0 来确定是否含有相异的分类结果。如果结果不为 0，代表有相异的分类结果，则抛弃这条数据；否则直接输出 key 和一个 count/length 到 HDFS 上的输出文件即可。整体过程的命令如下：

```
hdfs dfs -rm /output/result/*
hdfs dfs -rmdir /output/result
hadoop jar hadoopClear.jar hadoopClear /input/ /output/result
hdfs dfs -cat /output/result/*
rm result.txt
hdfs dfs -get /output/result/part* result.txt
```

经过几次实验，读取了 30 万条数据进行去重清洗，最后得出的平均重复率为：0.1933%，即 30 万条数据中一共有 58 条数据的属性值重合，其中异分类结果为 0。

3.3 决策树实现

3.3.1 连续属性值离散化

因为获取的数据为浮点数表示的连续值，如果给每个取值开一个分支显然不可行，就算是将精度降低到 0.1，那么五个属性延展开来最后还是会超过上亿种分支的可能，显然这个数量级对于及于决策树的分类问题是很不友好的，所以在进行模型构建之前需要进行数值离散化处理。

离散化常用方法为二分法。给定样本集 D 与连续属性 a，二分法试图找到一个划分点 t 将样本集 D 在属性 a 上分为 a ≤ t 与 a ＞ t。但是这一点对于我们的数据规模和跨度不合适，所以采用一种基于熵的离散化方法[10]。

当频率（或概率）分布具有最大的属性值个数时，熵（或信息）被最大化[11]。当我们单独将某一个属性所有的值与对应的分类结果拿出来做一个单属性样本集 X 的时候，我们可以自定义若干个区间对属性进行区间分割，然后根据前面的信息熵的计算公式，计算出这个样本集的信息熵 H(X)。另外根据凸函数的性质：

$$f(\lambda x_1 + (1-\lambda)x_2) \le \lambda f(x_1) + (1-\lambda) f(x_2) \qquad (3-1)$$

由参考文献中提到的离散化算法（EADC），可以将每个连续属性离散化为若干区间，区间数目由数据本身决定（但是最小区间划分数目为 10 个），之后找到合并两个区间后使得合并前后熵差最小，然后保存划分点，继续合并直到达到最佳平衡为止。是否达到最佳平衡的度量公式为：

$$h = (k_{max} - 1) * H(p) - H_{max}(p) * (k-2) \qquad (3-2)$$

式中，k_{max} 表示最大区间数，$H_{max}(p)$ 表示最大熵值。在实际操作中，对其进行简化，取 40 个划分区间下的对应值。

连续属性离散化之前，决策树的数据精度设置为 1，正确率一直在 30%

左右；离散化后，数据精度设置为 0.1，正确率随着训练数据量增长而增长。训练数据量为 1000 条左右时正确率为 35%左右，当数据量提升到 20000 条左右时，准确率有 53%左右，而 SVM 数据因为不需要划分区间，所以正确率的增长与数据量的线性关系并不明显，从 1000 条数据到两万条数据仅仅波动了 1%的正确率不到。可见决策树的正确率的提高，在初期更加依赖于训练数据量的积累。

整个离散化的过程如下：

（1） 从本地数据库读取设备运行记录数据，格式化后以实参形式传入到 EADC 离散化方法中；

（2） 在离散化方法中，针对单一的属性，取出其所有的值和分类数据，并且按照属性值进行排序；

（3） 排序后根据初始区间数划分区间，并且利用公式（2-2）的熵的计算公式，计算出改属性的初始熵，并且将度量数值 C_k 预设为 0 ；

（4） 先遍历整个区间集合，依次尝试合并两个相邻区间，记录每一次使合并前后的熵差，最后选择熵差最小的两个相邻区间进行合并，并且重置划分点，保存合并后的熵值；

（5） 根据上面的公式（3-2）提供的度量公式，计算出当前合并和后的度量值 $C_{k-1} = h$；

（6） 如果 $C_{k-1} > C_k$ ，那么令 $k = k - 1$，跳回到第(4)步循环进行区间遍历、合并、计算；

（7） 如果 $C_{k-1} < C_k$ ，保存当前的区间划分，结束区间划分进程；

（8） 将最后的区间划分点返回至 EADC 方法调用者，由其对实际数据根据划分点进行离散化处理。

离散化流程图如下：

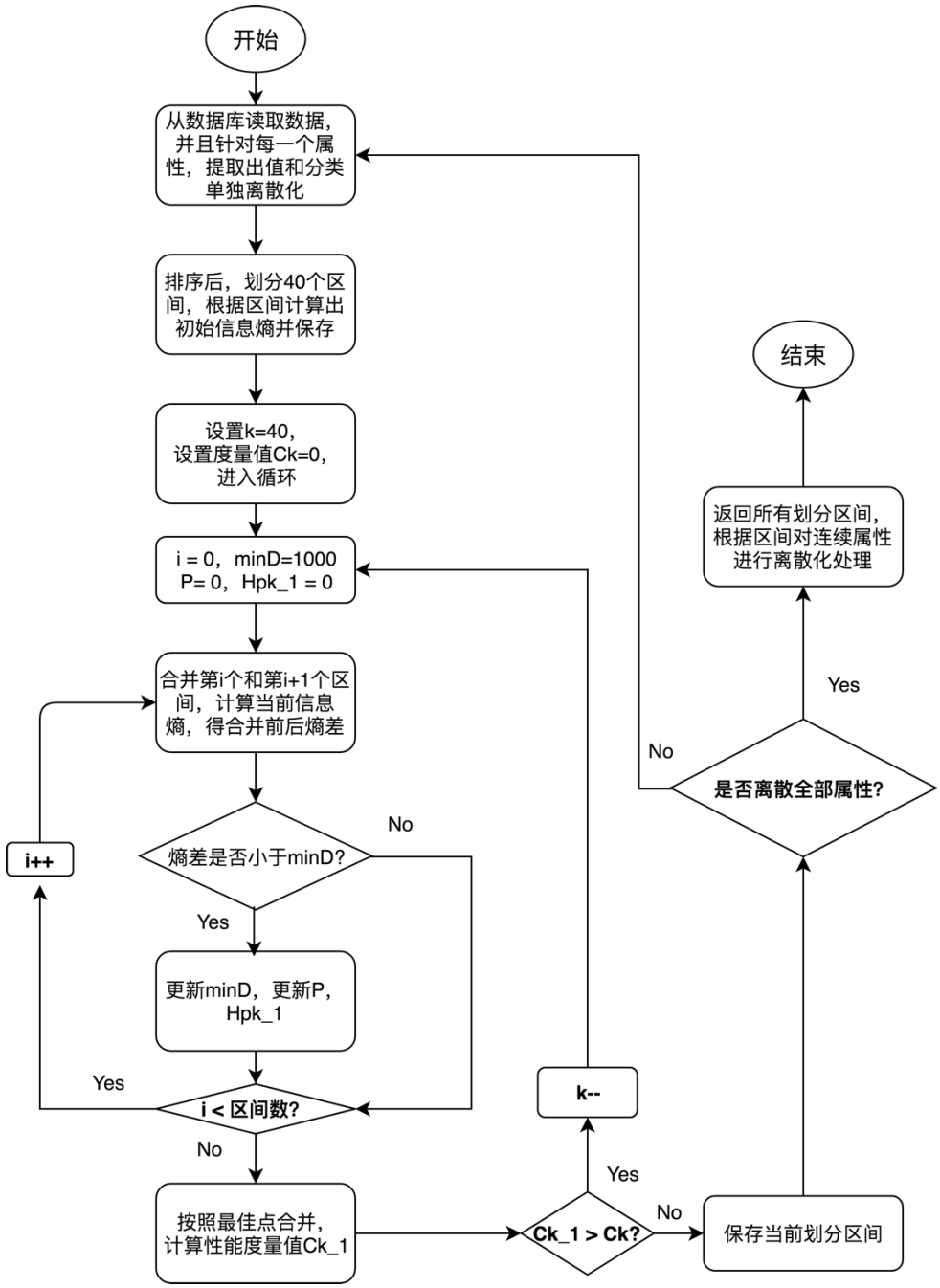

图 3-4 连续属性值离散化流程图

3.3.2 样本初始化

决策树的核心算法是 ID3 算法，其他的数据结构，数据处理等都是辅助内容，但是样本初始化在整个体系中也是举足轻重的。

首先定义属性名列表 attribute，也就是四个传感器的位置和负载百分比，然后是在此基础上增加一个分类属性名重新定义一个列表

attributr_Names，作为读取数据库内容时候的列名：

```
String[] attribute = new String[] {"Sensor1","Sensor2","Sensor3", "Sensor4", "Load"};
String[] attribute_Names = new String[] {"Sensor1","Sensor2","Sensor3","Sensor4","Load",
"category"};
```

当属性列表定义完毕之后，就会进入 Decision Tree 算法的样本读取方法，readSample()。在这个方法中，通过在前面数据库模块定义的 ReadData 类中的 readTrainData()方法读取出数据之后进行样本整合。具体的操作为：

（1） 定义一个 Sample 类，内含属性名及其对应的属性值；

（2） 对读取出来的二维数组逐行读取，每一行定义为一个 Sample 实例；

（3） 按照所有实例的分类，定义与分类数目相同的链表，读取当前样本的分类，并且将当前 Sample 添加到相应的链表上去。如果没有这个分类，就新添加一条链表。

（4） 最后返回的数据结构为类别为键，包涵所有此类样本的链表为值的键值对 Map。

样本初始化整体流程如下：

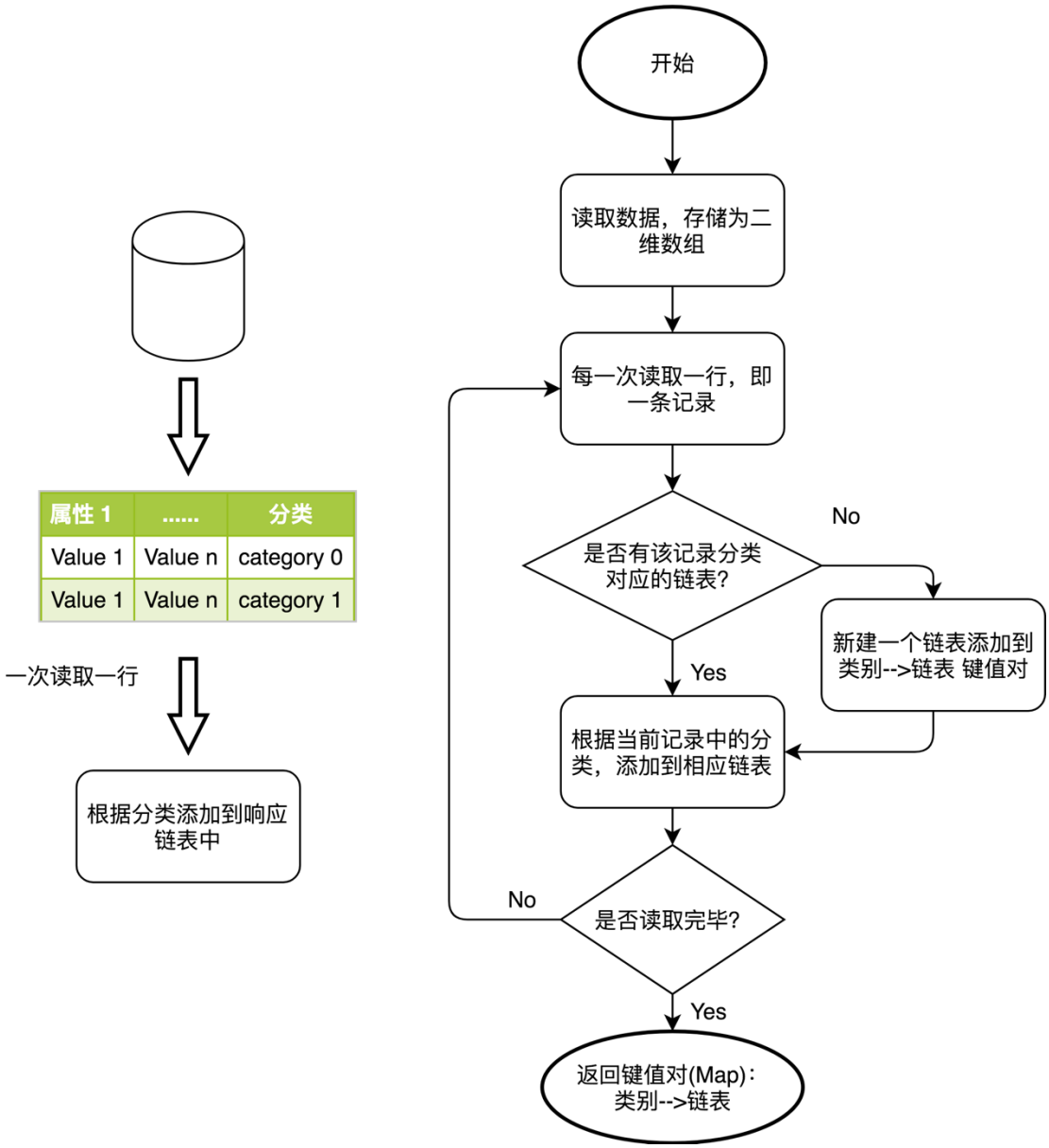

图 3-5 样本初始化流程图

3.3.3 生成决策树

在样本初始化完毕后，就进入生成决策树的阶段。调用定义的generateDecisionTree()方法，传入样本集和属性列表。就可以得到一颗基于此样本集，用 ID3 算法构建的故障树了。

```
//生成决策树
Object decisionTree = generateDecisionTree(samples,attribute);
```

在这个方法当中，ID3 算法担任了求出信息增益最大的属性的责任。整个generateDecisionTree()方法采用递归的方式，不断地向下延伸分支，直到

将当前节点归类为叶节点才会停止。

流程解释如下；

（1） 判断是否当前分支的样本数目，如果为空或者分类只有一种，那么将当前样本的类别作为叶节点的分类；

（2） 如果属性用完，或者是同一个属性值对应的样本子集中无法通过数目将此节点归类，那么采用后验分布进行归类；

（3） 如果上述条件都不满足，那么就使用 ID3 算法计算出当前的分支属性，并且读取该分支属性的各个属性值构成分支向下延伸（进入递归），直到遇到上述的条件成为叶节点为止；

（4） 当满足生成叶节点的条件时，则递归止步于当前节点，当前节点作为叶节点返回至其父节点，直至抵达根节点为止。

整体流程如下：

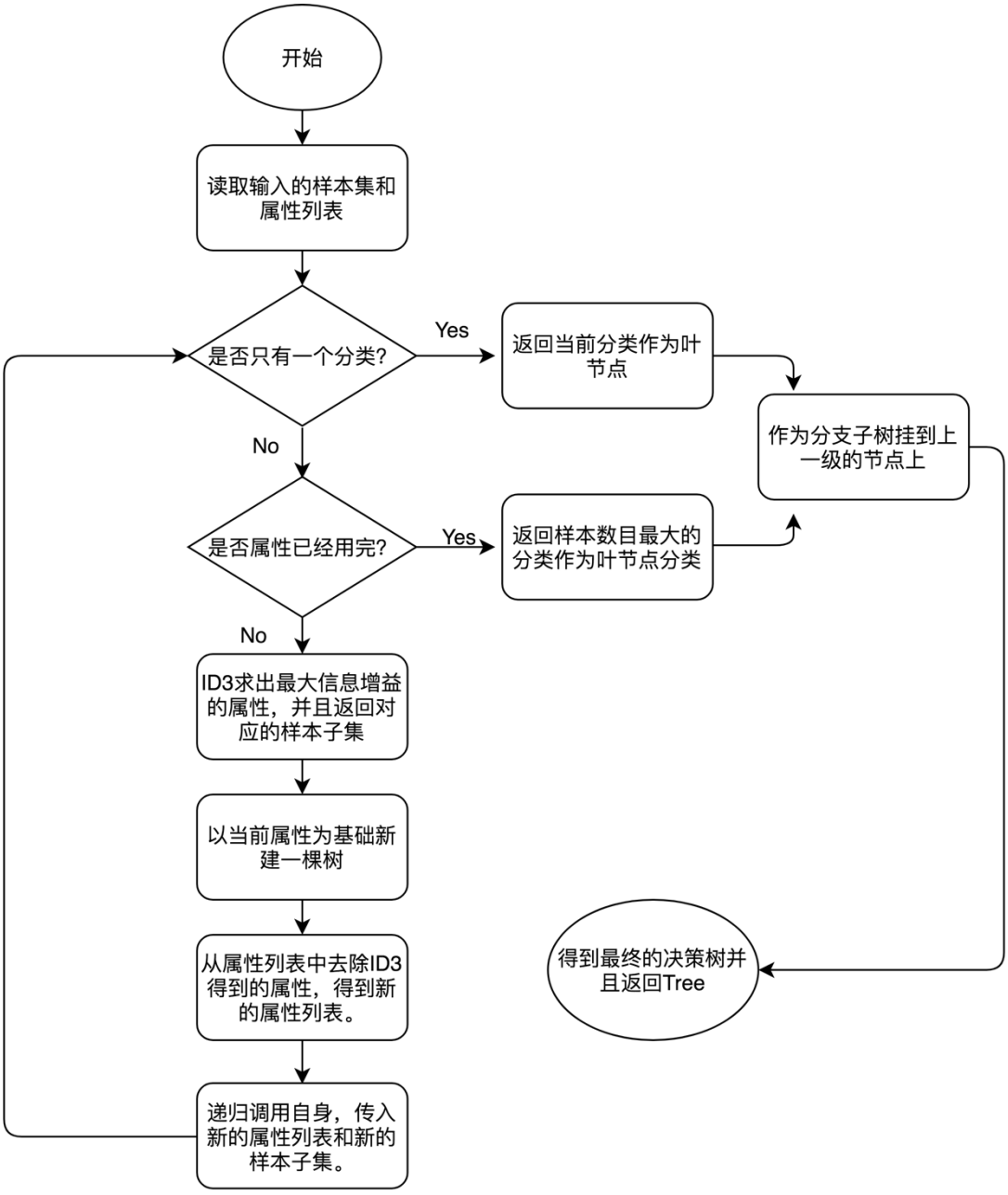

图 3-6 决策树生成流程示意图

至此，整个决策树就已经训练完毕了。我们从这一系列的操作中得到了一个 Tree 类的实例对象。每一个 Tree 都是由一个根节点和一系列的分支组成，除了叶节点不是 Tree 类型外，其他的子节点都是 Tree 类型。这一点也为我们提供了一个良好的搜索环境。只要检测当前节点是否为 Tree 类型，就可以判定是否已经搜索到了子节点了，这一特性在后面的数据测试中将会用到。

而在上面生成决策树时用到的 ID3 方法，下面进行详细的解释：

（1） 传入一个样本集，一个属性列表；

（2） 样本集形式为分类与此分类对应的所有样本，注意此处为分类而不是属性值；

（3） 对每一个属性值进行信息增益的计算，具体的实现方式为：

a) 读取当前属性值，拿到一个键值对，【所属类别-->样本集】

b) 解析键值对，分解出 key 和 value，其中 key 为类别，value 为此类别所有的样本

c) 对于 Value 里边读出来的每个样本，分别读取当前属性下的值，然后建立起来当前属性值相同的所有样本的样本集；

d) 建立起了所有属性值对应的样本子集后，再在此样本子集的基础上按照分类的不同进行子集划分，相当于是二次划分样本集；

e) 最终得到的数据结构如下：

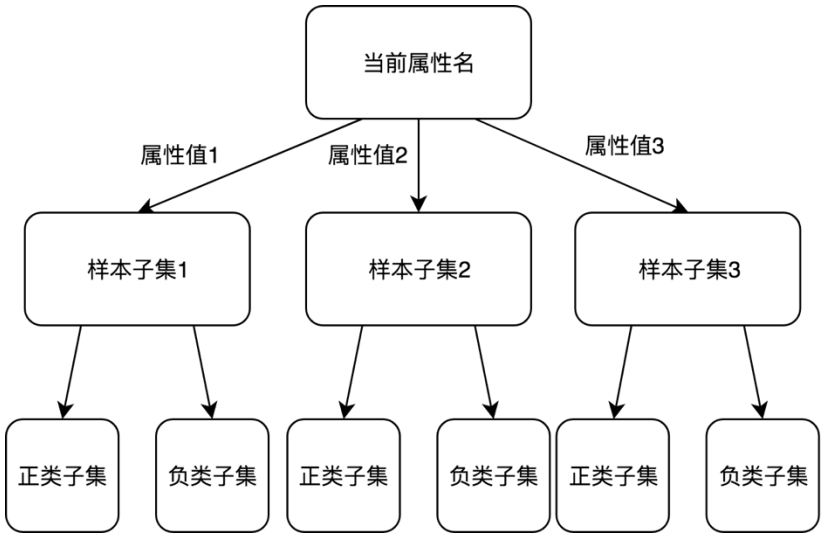

图 3-7 ID3 信息增益计算的数据结构

f) 根据上面的数据结构，计算每一个属性值的信息熵，然后结合所有的属性值计算出这个属性所对应的信息增益，最后得到信息增益最大的那个属性，即可将此属性的下标，对应的信息增益以及由这个属性所衍生出来的样本集打包成一个数组返回。

g） 至此，ID3 算法执行完毕，返回一个数组供 generateDecisionTree()方法选择下一个分支的属性，并且递归调用自身产生子树分支。

3.3.4 数据测试类

数据测试类接受一个模型 obj，一个属性名列表 Attr_Name，一个测试数据（属性值）列表 TestData，这三个参数用于测试数据；另外由于需要对外开放测试结果，还接受一个字符串变量 line 用于递归返回最终结果。下面是 TestData.java 的定义：

```
public static String    TestData(Object obj, Object[] Attr_Name, Object[] TestData,String line) {
```

基于既定的决策树模型，从根节点出发，根据传入的测试数据选择分支，一路向下，直到抵达叶节点，或者无法找到向下的分支为止。流程解释如下：

（1） 判断传入的 obj 类型是否为 Tree，如果不为 Tree，代表着已经到了叶节点，可以根据当前节点的值来找到对应的分类；

（2） 如果 obj 是 Tree 类型，找到 obj 对应的属性名，然后根据属性名找到测试数据的值，同时新建立一个未曾使用的属性名列表和对应的测试数据列表（长度等于原来的列表减一）；

（3） 遍历当前节点的每个分支，找到属性值等于测试数据的那一条；

（4） 将分支对应的子树以及（2）中建立的两个新列表，还有 line 传入新的 TestData()中，递归调用，返回 line；

数据测试类被定义为静态类，所以可以直接在 GUI 后台程序中调用而不用考虑是否定义实例。在 GUI 中的命令处理环节，有两个部分运用到了 TestData()这个方法，一个是 test，这个命令可以单独对一组数据进行测试；而另外一个 autotest 则是对已经加载好的数据测试文件进行解读后，直接批量测试，最后得出准确率，结合 SVM 下的准确率一起显示在消息提示栏。每一次测试数据的整体流程图如下：

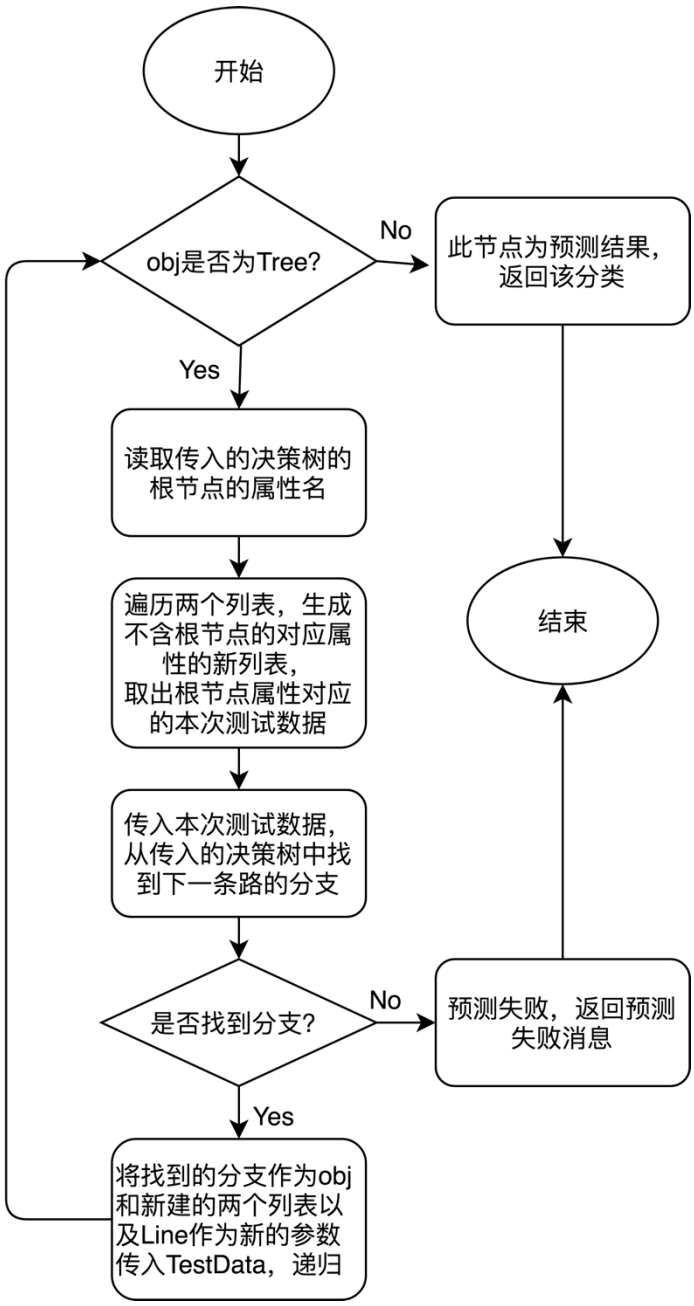

图 3-8 数据测试流程

3.4 支持向量机实现

SVM 由于其复杂性较高，而且网络上已经有了相当成熟的软件包可以直接取用，所以我最后借鉴了台湾大学林智仁教授的 LibSVM 包中的 Java 部分。

这个包的对外内容由四个，分别是 svm_toy，svm_scale，svm_train，svm_predict。我用到的是后面两个 svm_train 和 svm_predict。其中

svm_train 是训练模型用到的实现代码，svm_predict 是在实际使用过程中使用模型进行数据分类的实现代码。

为了配合 GUI 的显示，将这些内容整合到了一个类：ZZB_SVM.java 中，确保可以全面的调用这些包内的方法并且返回需要的数值。代码如下：

```
import java.io.IOException;
import java.text.NumberFormat;

public class ZZB_SVM {
    public static Float main() throws IOException {
        SVMReadData sr = new SVMReadData();
        Parameter par = new Parameter();
        String trainFileName = sr.readTrainData(par);
        String testFileName = sr.readTestData(par);
        //训练使用的数据以及训练得出生成的模型文件名。
        String[] trainFile = { trainFileName, "model.txt" };
        //测试数据文件，模型文件，结果存放文件
        String[] predictFile = { testFileName, "model.txt","predict.txt" };
        System.out.println(".....SVM Start.....");
        long start=System.currentTimeMillis();
        svm_train.main(trainFile); //训练
        System.out.println("Usage of Time : "+(System.currentTimeMillis()-start));
        //预测
        float x = svm_predict.main(predictFile);
        return x;
    }
```

该类的 main()方法最终将返回一个测试正确率的浮点数。静态方法 main()的调用位置在人机交互界面类 GUI.java 中调用并且显示。

在对 LibSVM 的使用过程中，均使用默认的选项进行模型训练。下面是对于 SVM 分类建模过程有较大影响的参数解释：

（1） SVM 类型设置（SVM type）：默认为 C-支持向量分类机，参数 C 是惩罚系数，C 的数值与对错误分类的惩罚成正比例关系。因此，惩罚系数对于模型预测的精度有较大的影响；

（2） 核函数设置（kernel type）：默认为径向基函数（Radial Basis Function，简称为 RBF 函数），在本次试验中采用高斯核函数，见表 2-1 中第三条；

（3） gamma：核函数中的 gamma 函数设置，此背景下默认为 0.25；

（4） eps：允许的终止判据，类似迭代精度(默认 0.001)

（5） shrinking：是否使用启发式，0 或 1(默认为 1)

3.5 人机交互界面设计

人机交互界面采用一个名为 GUI.java 的类来实现，其采用 Java 的 awt 和 Swing 两个专用于 GUI 编程的自带库来实现各个组件。在 ZZB_JCS.java 的 main 方法中定义一个 GUI.java 的实例化对象后，其以线程的方式独立于主线程存在，不过在主线程运行完毕之前会将构建的决策树模型以及其他一些会在人机交互中用到的变量，通过定义的静态方法传入到 GUI 的实例对象中，从而实现从后台到前台的过渡。

GUI 整体由一个大框架，12 个 Label 文字标签、1 个文字输入栏，3 个按钮组成。初步效果如下：

图 3-9 人机交互界面展示

另外配有两个菜单栏提供功能显示：

图 3-10 菜单栏

3.5.1 界面组件介绍

在主程序的后半段，会将决策树算法处理数据后生成的决策树模型传入到 GUI 线程实例中，然后在此基础上，将决策树模型分段在 GUI 上展示。Line1-Line10 就是用于展示决策树模型的。最上方的“指令输入框”作为提示，表示下面的输入框的作用。在文本输入框中键入对应的指令即可调用不同的系统命令执行包括测试数据，自动加载测试数据，退出，决策树展示换行等不同的功能。部分命令按钮会在首次加载 GUI 的时候以弹窗形式告知。

图 3-11 初始化人机交互指令提示弹窗

按照提示，在命令框中输入“HELP”会显示所有的命令：

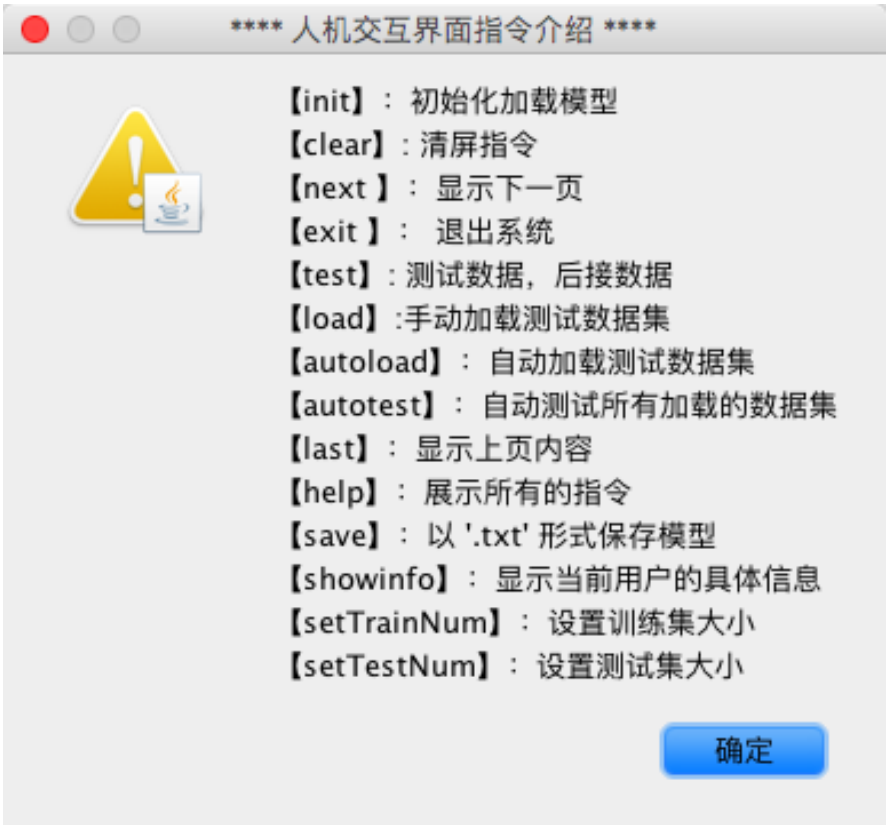

图 3-12 help 命令弹窗

底部的两个按钮则是用于查看决策树模型的辅助按钮。Button-CLEAR 用于直接清空 Label 的输出，恢复到模型展示初始状态，也就是初始化的时候的样子，效果等同于在命令输入框输入 clear 这条指令。而 Button-NEXT 则是等同于在命令框输入 next 这条指令的效果，调取下一个十行决策树模型展示。

在最底部还有一个 Label，这个 Label 用于响应如 test，Load 等无法直接在界面组件上展示效果的各种命令。如果命令错误，那么在底部会显示 Error 报错；如果是进行数据测试，那么会在最后这一条 Label 上展示出结果；如果是如加载数据的“Load”指令，则会在底部显示加载结果。

3.5.2 操作命令介绍

下面解释下输入框能够接受的几条定义的命令：

（1） init：初始化决策树输出，内部加载数据，在 init 执行前所有的其他命令都无法操作；

（2） clear：清屏效果，将 Line1-Line10 以及底部提示栏还原为初始状态；

（3） next：读取下一个十行决策树内容显示在界面上；

（4） last: 读取上一个十行决策树内容显示在界面上；

（5） test [data(value1 value2 value3 …)]：调用决策树模型测试输入的数据：数据格式如上。如果没有在命令框中输入数据，那么系统会查询是否已经加载了外部数据，如果有，则自动读取外部数据，否则按照预设的数据（固定的示例）进行计算演示；其结果输出到底部消息提示栏的为分类结果；

（6） load: 效果类似于菜单栏的 open

（7） autoload：自动加载测试数据；搜索当前目录下的测试数据文本，如果要跨越目录可以使用左上角的菜单栏->Open，选择文件读入或者在命令框中输入“LOAD”加载文件读取窗口；

（8） autotest：自动将目前存储的所有测试数据进行分类预测，并且统计正确率。同时调用 ZZB_SVM.main()测试 SVM 下的结果，将两种模型准确度输出到底部消息提示栏；

（9） save：将当前的决策树存入到指定文件夹，效果等同于菜单栏中的 save 选项；

（10）help：输入后可以跳出弹窗显示所有命令提示；

（11）showinfo：显示当先使用者的信息，包括工作目录、使用者、操作系统、训练集大小、测试集大小、模型输出行数、页面数等；

（12）exit：退出 GUI 界面，效果等同于默认的退出按钮或菜单栏的 exit 选项；

（13）settrainnum：设置训练集大小，后必须跟一个整形参数，另外重置后必须再次进行初始化，否则会报错；

（14）settestnum：设置测试集大小，后必须跟一个整形参数，重置后也必须进行初始化操作。

3.5.3 适用场景

此 GUI 界面适用于智能工厂环境，因为需要预装数据库，Java8 等软件，所以需要性能较为优越的计算机，才能保证初始化后能够迅速得到模型并且初始化 GUI 界面。另外 GUI 界面需要标准键盘输入才能与之交互，如果可以使用一些实物按钮来封装命令或者对应着 GUI 界面上的某些按钮将会有较好的效果。

每一次运行都是从主程序开始运行。因为该模型属于数据驱动方式，所以每一次模型的建立都依赖于历史运行数据。同时测试数据的时候也需要在本地读取一份需要测试的数据集，而且要根据相应的格式存储。所以需要用 PLC、Arduino 或者树莓派等控制器接受来自传感器的数据之后，进行简单的处理再存储到本地或者是串口发送到 PC 端。当然也可以将这份数据处理的工作放到运行程序的 PC 上，不过这样的话负载不够均衡。

在本设计中，由于无法进行实物制作，所以 PC 需要从 Mysql 动态读取数据，如果直接使用的话，中间还存在一个数据转储的问题，这对于网络资源、运算能力都是极大地浪费。所以在实际的操作中完全可以用串口或者是局域网将控制器和 PC 进行连接，然后由 PC 端接收后直接使用即可。

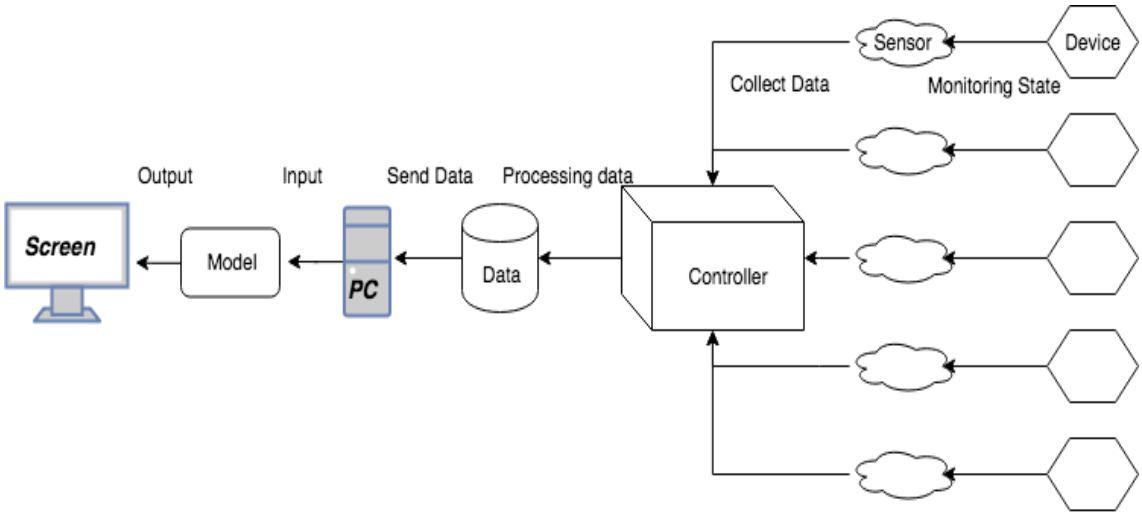

图 3-13 应用场景示意图

另外，如果投入实际使用，也可以将所有程序打包发布为 EXE 可执行文件，安装到本地后直接使用而不用去部署开发环境，而且可以根据工厂里面的设备导出不同类型的可执行文件，适配于大部分的生产场合。

具体的使用对象可以根据工厂的实际生产环境改变。比如在车床上，可以将传感器加载在车床主轴、顶尖、丝杠等耦合的地方。一个区域的变化极容易引起另外的区域变化的设备上，采用数据驱动的方式可以很方便的收集数据并且处理成精确度较高的模型。

4 性能分析

4.1 性能度量

本次设计采用了两套方案，我们就需要评估方法来对性能进行评估以便我们选择更好的模型构建方式。性能测试采用“测试集”来测试模型对于新样本的分类能力。最后以测试集上的误差来作为实际误差的近似。

测试集的获取的方式有几种，本次采用“留出法”。即将一个样本集 D 划分为两个，一个作为模型训练集 S，一个作为模型测试集 T，满足 D=S∪T 且 S∩T=∅，常见的划分方式为分层抽样数据后取出三分之一到五分之一作为测试集。这一点模型通过 Parameter 这个类中的静态变量来定义。

```
private static int rate = 2;
private static int trainNum = 20000;
private static int testNum = trainNum/rate;
```

所有的数据数量都来自于这个类中的静态变量。

在 GUI 界面中提供了 autotest 进行批量的数据测试，最终结果显示在消息栏。里面有 DecisionTree 的精度和 SVM 的精度表示。而由【错误率+精度=1】就可以知道错误率。

另外,还引入查准率（precision）P 和查全率（recall）R 的概念。对于二分类问题，分类结果的混淆矩阵和上述两种概念的定义如下：

表 3-1 混淆矩阵

真实情况	预测结果	
	正例	反例
正例	TP（真正例）	FN（假反例）
反例	FP（假正例）	TN（真反例）

查全率 P 定义：

$$P = \frac{TP}{TP + FP} \qquad (4-1)$$

查准率 R 定义：

$$R = \frac{TP}{TP + FN} \qquad (4-2)$$

4.2 决策树性能度量

决策树的精度对数据数量很敏感，数据量少的时候精度较低，数据量提升之后精度会随之提高。而 SVM 则对数据的数量要求较低，在连续属性离散化的部分曾介绍过。下面是用各种大小的训练数据量下得出来的“数据量-精度”图：

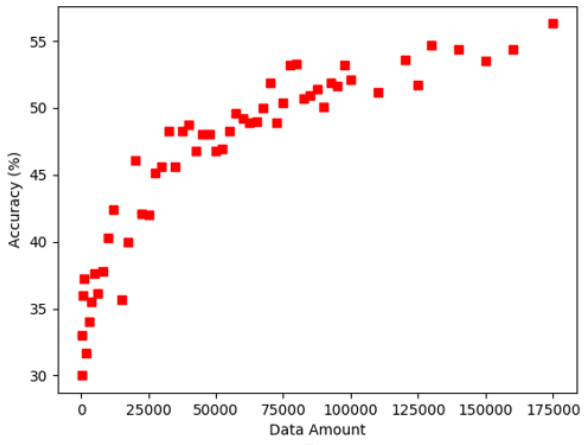

图 4-1 决策树“数据量-精度”关系图

下图中分别显示了精度（Accuracy）、查全率（Precision）、查准率（Recall）在各种数据下的对应值。

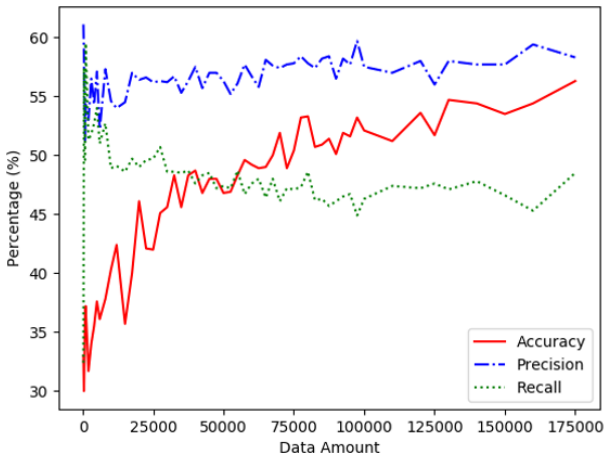

图 4-2 三种度量与数据量的关系图

从两图分析可知：

（1） 整体上决策树的预测结果精度与数据量有关，据趋势可以看出来，数据量越大，预测精度就会越高；

（2） 数据量与查全率的关系不大，查全率是尽可能将更多的情况考虑到的数值度量；

（3） 可以看到查准率随着数据量的提高甚至略有下降趋势，查准率表示对预测结果正确性要求的度量；

（4） 在数据量较大的时候，查全率与查准率二者之间会产生矛盾。如果希望将大部分的正例都选出来，那么可以通过提高选取的样本数量提高来实现，但是这样的话就会使得查准率降低，如我们上面图形中所示；而如果希望查准率高，那么就尽量让选中正例的范围变大，因为我们有连续属性离散化的过程，所以当数据较多时，离散化区间较多，区间细分程度更高，以至于查准率会稍有降低。

4.3 支持向量机性能度量

支持向量机的性能度量主要是精度，在运行时间上 SVM 比决策树同等数据量规模下所花费的时间更多，所以主要对比二者之间的精度。

相比于决策树，SVM 显然受数据量的规模影响小得多，而且 SVM 由于其本身性质，对于错误数据的敏感程度取决于惩罚系数的大小。所以哪怕错误的数据不是很多，在很多情况下也会造成较大的影响。

下面是 SVM 的精度与数据量的关系图：

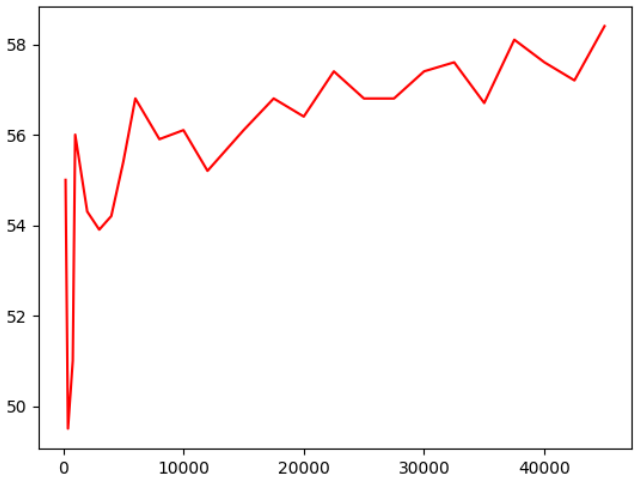

图 4-3 SVM “数据量-精度”图

从图中分析，对比决策树从 30%~60%的波动，SVM 50%~60%的精度波动范围显然受数据量的影响较小，但是从整体趋势上看，SVM 的精度也会随着数据量的增大微弱的提高。

另外，下表中利用多项式拟合与对数拟合进行了整体趋势的推导，可以预见决策树在数量级更高时有更好的表现：

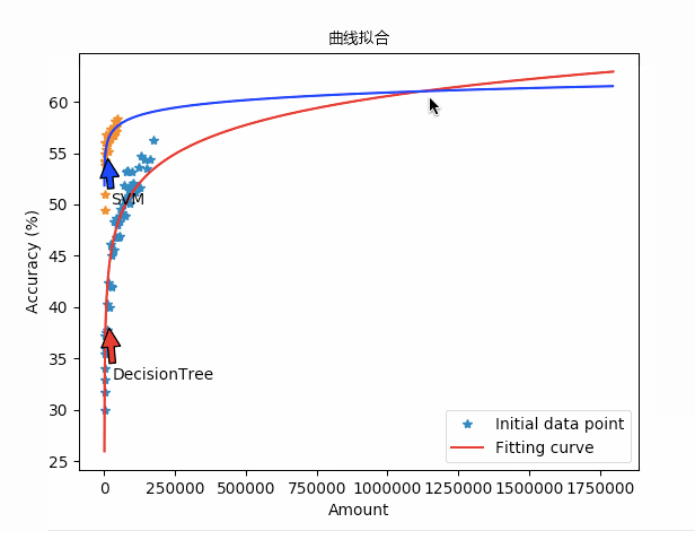

图 4-4 决策树、SVM 精度拟合曲线

5 结论

5.1 全文总结

本次设计内容总体达到了预期进度，成功的建立了故障诊断的模型和人工交互界面。并且通过划分训练集和测试集，实现了对故障或者异常情况的推理。

本次设计所采用的数据均为连续性数据，在支持向量机的模型中影响不大， 但是对于决策树模型的精度影响十分之大。后来经过 EADC 算法的离散化处理，决策树模型的精度有了很大的进步。而且在性能分析中提到：决策树对于数据量十分敏感，在运行环境的最大承载能力下，决策树的精度能达到 58%左右，而且上升趋势仍然十分明显。由此可见如果将决策树模型应用于实际生产环境，辅之以足够的运算能力，也会有一定的实际应用价值。同时因为模型的计算过程较快，所以数据的更新迭代可以周期性的展开，确保数据驱动所使用的数据都是近期的历史数据，紧随着生产设备的实际情况而更新。不用过多的拘泥于过往的先验知识。

当然，基于数据驱动模型的缺陷也是显而易见的：对于未曾出现过的故障类型或者是异常数据，就会有无法处理或者是判定错误的现象，在这一点上钟福磊的混合故障诊断模型显然更加具有实用性。

另外借鉴与 LibSVM 的支持向量机模型由于其集成度很高，内部相当完善，所以在数据量的变化过程中具有较好的表现。但是由于其在运算时所需要的时间与内存等相较于决策树模型更多，而且其精度与数据量的关系远没有决策树那么明显。

所以在实际的应用中，决策树适合于数据量较大，对于精度要求高，易于理解的场合中。而且本次设计的 GUI 界面配套了决策树的输出，利于操作人员理解使用。而 SVM 更加适合于数据量较小，但是对于精度有一定要求的场合。

5.2 展望

本次毕设过程中，我学习到了很多的知识，尤其是对于算法实现，数据选取，数据挖掘，模型优化等内容有了长足的认识与进步。而在模型建立过程中，还有很多理论和技术上的问题需要解决：

（1） 本次设计对于各个属性的选取并未经过严格考虑，在实际应用中应该提前对数据进行筛选和清理，比如计算各个属性参数之间的关联度剔除一些很明显的错误训练样本和验证样本，这样不仅能降低训练、测试开销，而且还能提高模型的预测精度；

（2） 对于模型的处理问题，剪枝方法未曾实现，而且剪枝虽然能提高模型对一些特定数据的精度并且减少预测开销，但是在一定程度上也限制了被剪枝模型对于更大范围数据的分类能力；

（3） 另外，在数据获取部分，应该使用一些机械方面的预备知识，对数据检测模块进行优化，使得获得的数据在计算关联度之前就先天利于模型构建；

（4） SVM 模块的内容并未深入，参数均采用默认类型。在实际使用时，可以调节一些参数，使得支持向量机模型更加适合于不同的工作环境；

（5） GUI 的功能可以进行扩充，同时改进人机交互体验；

（6） Java 与 Mysql 的使用在设计过程中还有待改进与优化的余地。

如果完成了上面所述的内容，配合基于数据驱动的模型只需提前设置好规范的数据格式即可适应于各类环境的特性，相信构建的模型会具有更大的实用价值。

致谢

光阴荏苒，四年大学生活即将走进尾声。这四年内我收获良多，对于那些曾给予我帮助，或者引领我走上正确的道路的人，我内心深处抱以深深的感激之情。

首先需要感谢的是我的指导老师金海教授，感谢老师为我选择的课题以及对我的指导。在我人生中最为迷茫的时候，金老师给了我莫大支持，让我坚定了走下去的信念。每一次与金老师的沟通，总能让人感觉如沐春风，感念至深，心头的不安与忐忑消失殆尽。未来几年内能够师从金老师，我感到非常幸运，也一定会严格要求自身，以最高的标准来完成导师学业与导师分配的任务！同时感谢实验室的 SCTS/CGCL HPCC 为我提供了实验环境。

同时也要感谢机械学院吴波教授。虽然我们交流甚少，但是吴波老师在对我论文的指点，以及帮助我解决图纸问题上给予了很大的帮助！

另外还要感谢胡友民老师，开题答辩时他一阵见血的指出我的不足之处，点明了我后期要努力的方向。毕业答辩之前的图纸也是来源于胡老师，在本次毕设过程中给了我很大的帮助，十分感谢！

还要感谢我的室友方舟同学，因为跨院系，所以很多消息都是通过他来给我解答，在我的论文写作过程中他也给予了我相当大的支持，让我规避了很多常识性的错误。另外，感谢他在后期对大纲结构、论文格式、图纸排版的指点，让我省下了很多时间。

最后要感谢的是我的父母， 感谢他们无私的支持，以及最长久的陪伴，感谢他们用双手，勤苦工作二十余年，为我创造了一个安然成长的环境。身处武汉异地求学，碰到过这样那样的挫折，心情潮涨潮落起伏不定，但是他们始终是我最坚实的后盾，让我可以在求学路上大步前行无需顾及身后，因为我知道他们一直都在！在未来，我一定更加勤奋的学习，不辜负父母对我的殷切期盼，争取早日成为家中顶梁柱，让父母放下身上的重担，享受生活！

参考文献

[1] 钟福磊. 工业大数据环境下的混合故障诊断模型研究[D].西安电子科技大学,2015.

[2] 卫凤林,董建,张群.《工业大数据白皮书(2017 版)》解读[J].信息技术与标准化,2017(04):13-17.

[3] 王建民.工业大数据技术[J].电信网技术,2016(08):1-5.

[4] 张鹏. 基于卡尔曼滤波的航空发动机故障诊断技术研究[D]. 南京: 南京航空航天大学, 2009.

[5] 徐德民, 刘富樯, 张立川等. 基于改进连续-离散无迹卡尔曼滤波的水下航行器故障诊断 [J]. 西北工业大学学报, 2014, 32(5): 756-760.

[6] Frank P M. Fault diagnosis in dynamics systems using ana-lytical and knowledge-based redundancy: a survey and some new results[J]. Automatica, 1990, 26(3): 459-474.

[7] Reis M S, Gins G. Industrial Process Monitoring in the Big Data/Industry 4.0 Era: From Detection, to Diagnosis, to Prognosis[J]. Processes, 2017, 5(3): 35.

[8] 张媛.采用数据挖掘技术中 ID3 决策树算法分析学生成绩[J].科技信息,2009(06):537

[9] 周志华 著.机器学习[M], 北京: 清华大学出版社, 2016 年 1 月. (ISBN 978-7-302-206853-6)

[10] 贺跃,郑建军,朱蕾.一种基于熵的连续属性离散化算法[J].计算机应用,2005(03):637-638+651.

[11] COVER TM, THOMAS JA. Elements of information theory[M].New York: John Wiley & Sons, 1991.

[12] 鲁峰, 黄金泉, 孔祥天. 基于变权重最小二乘法的发动机气路故障诊断[J]. 航空动力学 报, 2011, 26(10): 2376-2381.

[13] 朱霄珣. 基于支持向量机的旋转机械故障诊断与预测方法研究[D].华北电力大学,2013.

[14] 易辉. 基于支持向量机的故障诊断及应用研究[D].南京航空航天大学,2011.

[15] 王振华,杜宇波.基于 ESMD 和 SVM 的滚动轴承故障诊断[J].现代制造技术与装备,2018(01):122+124.

[16] Yang Li,Yan Qiang Li,Zhi Xue Wang. Fault Diagnosis of Automobile ECUs with Data Mining Technologies[J]. Applied Mechanics and Materials,2011,1069(40).

[17] Xiao Rong Cheng,Qiong Wang. An Improved ID3 Algorithm for Power Equipment in Green Power Engineering[J]. Applied Mechanics and Materials,2013,2488(340).

[18] Huan Huang,Natalie Baddour,Ming Liang. Bearing fault diagnosis under unknown time-varying rotational speed conditions via multiple time-frequency curve extraction[J]. Journal of Sound and Vibration,2017

[19] Guo Ping Li,Qing Wei Zhang,Ma Xiao. Fault Diagnosis Research of Hydraulic Excavator Based on Fault Tree and Fuzzy Neural Network[J]. Applied Mechanics and Materials,2013,2308(303).

[20] 盛博, 邓超, 熊尧等. 基于图论的数控机床故障诊断方法[J]. 计算机集成制造系统, 2015, 06: 1559-1570.

[21] 李晗, 萧德云. 基于数据驱动的故障诊断方法综述[J]. 控制与决策, 2011, 26(1): 1-9+16.

[22] 刘强, 柴天佑, 秦泗钊. 基于数据和知识的工业过程监视及故障诊断综述[J]. 控制与决策, 2010, 25(6): 801-807+813.

[23] Zhang, Liangwei. Big Data Analytics for Fault Detection and its Application in Maintenance[J], 2016

[24] Jay Lee, Hung-An Kao, Shanhu Yang. Service innovation and smart analytics gor Industry 4.0 and big data environment[J]. Percedia CTRP, 2014, 16:3-8.

[25] 邸文君,宫秀军.基于Hadoop架构的数据驱动的SVM并行增量学习算法[J].计算机应用,2016,36(11):3044-3049.

[26] 赵华,苏东,乔文生.TBM主变速箱的状态监测与故障诊断[J].建筑机械化,2003(06):44-45+43.

[27] 徐牧. 基于SVM的变压器故障诊断研究[D].安徽理工大学,2017

[28] 罗雨滋,付兴宏.数据挖掘ID3决策树分类算法及其改进算法[J].计算机系统应用,2013,22(10):136-138+187.

附录