文本复制检测报告单(全文标明引文)

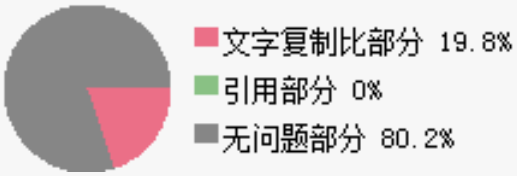
№:ADBD2017R_20171107182339201712172204514014947513339 检测时间:2017-12-17 22:04:51

检测文献: 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现
作者: 基于动态手势的智能终端身份识别技术研究与实现
检测范围: 中国学术期刊网络出版总库
中国博士学位论文全文数据库/中国优秀硕士学位论文全文数据库
中国重要会议论文全文数据库
中国重要报纸全文数据库
中国专利全文数据库
互联网资源(包含贴吧等论坛资源)
英文数据库(涵盖期刊、博硕、会议的英文数据以及德国Springer、英国Taylor&Francis 期刊数据库等)
港澳台学术文献库
优先出版文献库
互联网文档资源
图书资源
CNKI大成编客-原创作品库
学术论文联合比对库
个人比对库
时间范围: 1900-01-01至2017-12-17

检测结果

总文字复制比：19.8% 跨语言检测结果：0%
去除引用文献复制比：19.8% 去除本人已发表文献复制比：19.8%
单篇最大文字复制比：5.9%（Hadoop YARN资源分配与调度的研究）

重复字数：	[21370]	总段落数：	[19]
总字数：	[107691]	疑似段落数：	[15]
单篇最大重复字数：	[6328]	前部重合字数：	[973]
疑似段落最大重合字数：	[5548]	后部重合字数：	[20397]
疑似段落最小重合字数：	[29]		



指 标：☑疑似剽窃观点 ☑疑似剽窃文字表述 ☐疑似自我剽窃 ☐疑似整体剽窃 ☐过度引用

表格：0 脚注与尾注：0

4%（227） 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第1部分（总5655字）
9.5%（746） 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第2部分（总7816字）
0%（0） 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第3部分（总10828字）
0%（0） 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第4部分（总4046字）
0%（0） 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第5部分（总2749字）
3.5%（29） 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第6部分（总837字）
7.1%（651） 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第7部分（总9196字）
22.2%（2090） 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第8部分（总9421字）
14.3%（405） 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第9部分（总2829字）
31.3%（1443） 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第10部分（总4607字）
75.3%（5548） 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第11部分（总7366字）
8.5%（209） 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第12部分（总2448字）
2.5%（31） 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第13部分（总1229字）
43.6%（3749） 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第14部分（总8600字）
57.8%（5132） 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第15部分（总8875字）
4.7%（468） 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第16部分（总9907字）
0%（0） 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第17部分（总2377字）
13.1%（154） 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第18部分（总1174字）
6.3%（488） 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第19部分（总7731字）

（注释：无问题部分 文字复制比部分 引用部分）

疑似剽窃观点 (5)

80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第7部分

1. 实验结果表明，采用本文的算法能有效地提高系统资源利用率，缩短集群的作业执行时间。

80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第11部分

1. 当AMA向RM中的Scheduler组件申请Container时，假设当前轮询的位置的节点Node2且Node2中可分配资源满足AM A所需资源量，Scheduler将Node2中的资源分配给AM A，然而从图4.1可以看出，相对于Node2，Node3的资源负载更小CPU处理速度更快，故将Node3中资源分配给AM A更合理。
2. 最终，根据本章算法编写相应的 Hadoop 资源调度器，并进行实验验证，实验表明本章调度器能够有效的提高集群资源分配的合理化，达到缩短作业执行时间的目的。

80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第14部分

1. 综上所述，现有的恶意代码检测技术有很多，每一种方法都有自身的优缺点。

80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第15部分

1. 还是动态分析，都需要借助恶意代码分析技术和监控技术获得恶意代码的基本属性和执行信息，以便深入理解恶意代码的功能，进一步实现恶意代码的检测和抑制。

1. 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第1部分 总字数：5655

相似文献列表 文字复制比：4%(227) 疑似剽窃观点：(0)

1	城市轨道交通对沿线住宅价格的影响研究 李林芸(导师：任波) -《重庆大学硕士论文》- 2011-10-01	1.4%（81） 是否引证：否
2	基于ZigBee的智能家居网关的研究与实现 张英会(导师：张乃通) -《哈尔滨工业大学硕士论文》- 2012-12-01	1.3%（72） 是否引证：否
3	016-220100902230-刘泉影 刘泉影 -《学术论文联合比对库》- 2013-04-02	1.2%（69） 是否引证：否
4	移动IPv6切换技术的研究 刘希伟(导师：代红) -《辽宁科技大学硕士论文》- 2011-12-05	1.2%（67） 是否引证：否
5	CNU1543011026 -《学术论文联合比对库》- 2014-04-21	1.2%（66） 是否引证：否
6	一种基于AF4iOS框架的移动软件设计方法 陈贝(导师：倪友聪) -《福建师范大学硕士论文》- 2014-04-06	1.0%（59） 是否引证：否

7	基于小波神经网络的人脸图像识别研究	0.6%（33）
	倪洽凯(导师：杨静) -《太原理工大学硕士论文》- 2012-05-01	是否引证：否
8	船舶装备物联网服务平台的规划研究	0.6%（32）
	周志凤(导师：朱岩) -《南京理工大学硕士论文》- 2013-05-01	是否引证：否
9	基于SVM和深度学习的情感分类算法研究	0.5%（31）
	黄志勇(导师：杨富平) -《重庆邮电大学硕士论文》- 2016-05-30	是否引证：否
10	制导炮弹可控滚转执行机构关键技术研究	0.5%（31）
	张皎(导师：姚晓先) -《北京理工大学博士论文》- 2015-06-01	是否引证：否
11	一种精确估计区域北斗接收机硬件延迟的方法	0.5%（30）
	李昕;郭际明;周吕;覃发超; -《测绘学报》- 2016-08-15	是否引证：否

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第一章杨倩晴

1.1 研究背景及意义

随着移动互联网的发展，智能手机已经普遍用于大众的生活，移动互联网带来了信息共享的同时也伴随着对用户隐私的威胁。现如今，由于手机被盗用或者丢失而产生的用户隐私数据泄漏的事件频频发生。因此如何实现准确且长久保证智能终端访问者的合法性成为了很多学者研究的重点。但大多数学者的研究都是基于入口式的，无法在用户进入系统后仍然在后台持续监控用户的合法性。针对上述情况，本文提出了一种基于动态手势的智能终端身份识别方法，实现了在后台24小时不间断的监控用户是否为非法用户。主要包含以下的研究内容：

1．本文采用小波包分解和皮尔森相关系数结合的方法对原始数据进行滤波去噪。首先采用小波包分解的方法将原始数据分解为几个频段，每个频段都含有不同的信息，接着用皮尔森相关系数对相邻的频段进行相关性分析，从而找到动态手势信息和高频噪声的分界点，然后对其进行滤波去噪。

2．本文对传统的K-Means算法在初始中心数目和初始中心位置点的确定上进行了优化。传统K-Means算法初始中心数目和初始中心点的选择是完全随机的，本文对初始中心数目的选择采取枚举的方式，初始中心点位置的选择采取的是半随机化的方式。

3．本文提出了一种基于动态手势的身份识别的方法。根据聚类的结果生成一个打分判决公式，以此为依据对新样本进行预判，还为新的样本数据提供了回收的机制，使模型能够得到不断的优化。

4．为了验证本文方法的可行性，本文搭建了一个基于动态手势的智能终端身份识别系统。客户端采用的是Android系统的智能终端，用此系统对本文提出的方法进行实验，实验结果说明本文方法在实验中的FAR(False Acceptance Rate)值为4.2%，FRR(False Rejection Rate)值为1.92%。证明了本方法模型在实际应用中具有一定的可信性。

随着移动互联网的发展，移动终端开始应用于人们生活中的各种场景。如今越来越多的人开始使用移动支付，比如支付宝或者微信；购物从原来的商场购物模式转变为使用移动应用进行购物，比如淘宝，京东等；打车由原来的出租转变为移动软件应用预约打车，比如滴滴打车，美团打车等；饮食方面越来越多的人愿意在移动应用上进行订餐，比如美团外卖，饿了么等。由此可见人们现在的生活已经离不开智能终端上那些各种各样的应用，它为人们的生活带来了巨大的便利，改善了人们生活的质量。但随着人们对移动终端使用的越来越多，用户个人信息的泄漏和隐私数据的窃取事件也频频发生，于是如何保证访问用户的合法性成为了研究的热点。目前智能终端大多采用数字密码和指纹进行身份的识别，从而保证访问用户的合法性，但这种方式存在缺陷，例如简单的密码容易被破解，而复杂的密码容易被用户遗忘导致用户自身无法正常的访问手机。指纹如果被复制就完全失去了对自己的手机的控制，导致无法挽回的损失。

为了提高智能终端用户的安全性，人们开始研究基于生物特征的身份识别方法，比如利用智能终端自带的摄像头进行人脸识别和虹膜识别，然而这种方式受周围光线环境影响很大，如果光线很暗或者很亮，或者背景中有其他头像等干扰识别的背景图像都无法进行正常的识别；三维动态手势签名是用手机自带摄像头进行动态手势签名的捕捉，容易受环境影响并且实现复杂；语音识别是根据个人的音色的不同进行识别，但是随着现代的科技发展语音合成技术越来越成熟，如果获取用户的录音，则很容易通过系统的语音认证。

智能手机终端中有很多传感器，比如加速度计、陀螺仪、压力传感器等，这些传感器具有价格低廉，体积微小和灵敏度高的优势，因此他们被广泛的应用于智能终端的硬件中。因此这些传感器为动态手势的提取提供了硬件的支持，在用户使用手机的过程中可以实时并且方便的获得各个传感器的数值。并且传感器具有不受周围环境影响的优势，无论周围的温度或者光线如何变化，都不会对传感器所测量的值造成影响。并且每个人有自己固定且稳定的使用习惯，在传感器上体现为相似的数值，因此具有很好的区分真假用户的能力。相比较于之前的生物特征的识别具有稳定性好、受环境影响小、实现成本低等优势，具有很好的研究价值。

基于动态手势的智能终端身份识别技术具有很好的研究价值，在理论上，每个用户使用手机的行为模式是稳定的，是多年使用手机的习惯所形成的，很难被模仿，同时一时之间也不会改变，因此非常适合作为身份识别的依据；在实践上，手机传感器都是自带的不需要额外的设备，并且用户基本随时会携带智能手机，采集数据方便，成本低廉，具有很好的可实现性。

1.2 国内外研究现状

为了保证访问用户的真实性，国内外学者研究了很多种身份识别的方法，目前应用的最广泛的身份识别的方法有：

1.静态密码

静态密码是由用户在注册过程中设定的一组由数字、字母和符号组合的字符串，在之后的登录请求过程中系统将再次对比用户登录过程中输入的密码和注册过程中设置的密码是否一致，如果一致则判定为合法用户，如果不一致则不允许该用户获取手机的访问权限。此方法具有实现简单，成本低廉和识别速度快的优势，但是存在简单的密码容易被破解，复杂的密码容易被用户遗忘的弊端。因此安全性很低，无法长时间保证用户隐私数据的安全性。

2.动态口令

动态口令指的是用户在请求登录的时候会给注册的时候绑定的手机号码发送验证码，系统对验证吗进行匹配，如果一致则允许该用户访问系统，如果不一致则不允许该用户对系统进行访问。此方法具有实现简单，成本低，使用方便无需用户自己记忆密码的优势，但验证码存在被半路拦截窃取的风险，易产生安全问题，并且容易受环境影响，如果信号不好则无法进行验证从而影响用户体验感。

3.指纹

指纹[1]识别依据手指末端的正面皮肤上凹凸不平的纹路来区别不同的人，因此非常适用于身份识别，但是指纹并不是终身不变的，随着年龄的增长指纹是会产生变化的，而且每个人的指纹是否是唯一的还有待验证，如果有人恶意获取了用户的指纹模型并进行仿冒，是可以通过指纹的验证的，因此此方法存在一定的安全漏洞。

4.人脸识别

随着科技的发展，图像识别技术越来越准确，这使得人脸识别[2]作为用户身份识别的方法成为可能。现如今有很多手机都内置了人脸识别功能，最新的iphoneX就以人脸识别作为卖点吸引了很多消费者。但是人脸识别也有局限性，例如无法区分长相非常相似的双胞胎，而且容易受数据采集时环境的影响，比如拍摄人脸时背景不干净或者有其他的头像就会影响到人脸识别的精度。

5.基于加速度传感器的三维手势签名[3]

智能手机中有内置的加速度传感器，根据用户在空中签名的轨迹进行身份识别，此方法的识别难度较大，并且在识别过程中需要用户拿起手机在空中进行签名，用户体验感差，因此并不具有实用性。

由上文可知，这些身份识别的方法各有各的优缺点，因此并不存在一种完美的身份识别方法。在选择身份识别方法时需要根据具体的应用场景和需求再进行选择。目前随着移动互联网的发展，越来越多的人的生活开始离不开互联网，在共享网络带来了便利的同时也带来了相应的安全隐患，比如个人信息的泄漏，隐私数据的盗取，资金的盗用等安全问题。因此如何在后台持续保证手机用户的合法性成为了急需解决的问题。越来越多的学者开始研究基于生物特征的身份识别技术。

基于手势的身份识别技术指的是用户在使用智能终端的过程中，会产生一些静态[4]或者动态[5]的手势。静态指的是一些时刻的传感器所获得的数值，比如击键识别就是一种静态的手势识别。另一种动态的手势识别不仅仅包含静态手势中某一时刻的传感器数值还包括一段时间中所产生的轨迹或者运动特征，比如滑动的速度和距离等。动态手势按照数据采集的方式又分为基于视觉的动态手势和基于数据“手套”的动态手势。基于视觉的动态手势指的是利用手机自带的摄像头对用户的手势进行采集，然后利用采集的数据进行特征提取，最后用相应的算法进行身份识别。此方法容易受环境的影响，采集过程中如果光线过暗或者过亮都会导致图像采集的准确性从而影响后续的识别，同时背景的不干净也会影响识别的准确性。基于数据“手套”的身份识别利用的是手机自带的传感器进行数据的采集，在用户的使用过程中自动在后台采集，并不影响用户前台正常的业务请求，因此具有实用性。并且现在传感器的批量生产使得传感器的价格低廉，而且随着生产技术的提高，传感器的灵敏度和稳定性也越来越好，这使得传感器在智能终端上的应用越来越广泛，为动态手势的识别提供了硬件条件。两种方式相比较，基于数据“手套”的动态手势识别更廉价、更方便、更具有实用性。因此本文主要研究基于数据“手套”的动态手势身份识别技术。

2011年陈文[6]等人提出了一种适用于手机身份识别的流水线机制，采用FGBD（Forward & Backward Gesture Detection）进行手势检测，结合动态规划和DTW（Dynamic Time Warping，动态时间规整）算法，提出了E-DTW（Enhance DTW）算法来对智能终端进行手势识别；2013年刘志丹[7]等人通过采集用户在手持手机进行空中签名的过程中产生的加速度的值，提出了一种基于小波变换理论的手势轨迹特征向量算法用于身份识别；2014年沈超[8]等人提出了一种基于陀螺仪行为特征的智能手机身份识别方法；2014年王尧等人提出了一种基于手机加速度传感器的三维手势身份识别方法，提出了A-DTW（average-DTW）算法；2014年高焕之[9]结合最长公共子序列算法进行数据匹配实现身份识别；2015年沈爱敏[10]提出了一种基于人体生理震颤的静态手势身份识别方法；2015年苗敏敏[11]等人提出了基于能量熵的端点检测方法来识别手势的开始和结束，基于DTW算法提出了半分动态时间规整算法（HDTW）来进行用户身份识别。目前的这些研究成果大多数都是基于非机器学习的算法，比如DTW算法，这些算法的缺点是学习能力弱而且算法复杂性太高；而采用机器学习的算法又大多属于监督学习，比如SVM算法，算法复杂度高。

目前研究的方向大部分都是面向所有的用户，也就是模型的建立不仅与本用户的数据有关还受其他用户的数据的影响，而本文只研究属于本用户的数据，不关心其他用户的数据，在数据的处理上大大降低了难度，并且加快了算法的收敛速度。本文的实现采用非监督的机器学习的算法，并采用样本回收进行模型矫正，优势在于本文采用的算法复杂度低而且具有较强的学习能力，同时获得了较低的FAR和FRR。

1.3 本文的工作与创新

本文的主要研究内容主要分为3个部分：

1、本文对传统的聚类算法K-Means算法在初始中心数目和初始中心位置的确定上进行了优化。传统的K-Means算法对数据比较敏感，不同的类的数量和初始中心位置的不同对聚类效果影响很大。相比传统K-Means算法的完全随机选择初始中心，本文在聚类数目的选择上使用枚举的方法来确定最佳的聚类数目，在K-Means算法初始中心点位置的选择采取半随机化的方式，最后形成了本文半随机化的K-Means聚类算法。

2、本文提出了一种基于动态手势的身份识别的方法。它根据聚类的结果得到用户的所有行为类分别占总行为的比重，以此比重为依据提出了行为相似度的打分公式，以此公式作为判决真伪用户的依据。同时还为新的预测样本增加了回收机制，对于判决的结果与真实情况不符的样本进行加权回收，加大模型学习遗漏的用户行为样本的学习的力度，使模型能够得到不断的完善。

3、基于动态手势的身份识别平台的系统实现。主要实现了以下几个模块：1）客户端（Android系统），主要承担用户交互界面，读取传感器相关数据及其封装，数据传输至后台服务的任务。其中数据封装格式采取通用JSON格式，传输协议采取的TCP/IP协议；2）服务器，主要负责整个身份识别的过程。主要包括数据的预处理，数据的存储，模型的训练，模型的预测，以及结果处理等；

本文的创新点主要分为2个部分：

1）结合传统K-Means算法提出了一种采用半随机化选取初始中心位置和枚举选择初始中心数量的K-Means算法；

2）提出了一种基于动态手势的智能终端身份识别的方法。此方法基于K-Means聚类的结果得出一个打分公式，以此打分公式对新数据进行判别，同时给新来的样本数据提供回收机制，随着样本的增多能不断的对模型公式进行优化。

最后对本文提出的基于动态手势的身份识别方法的有效性进行了实验，实验证明本文方法具有较低的FAR和FRR，能在不影响用户正常使用的条件下有效的监测智能终端是否具有非法的访问。

1.4 论文结构

本文共分为6个章节，具体各个章节的安排如下：

第一章绪论，首先介绍了本文的研究背景和研究的意义，接下来介绍了目前国内外在身份识别领域的研究现状，紧接着介绍了本文的主要研究内容和创新点，最后对本文的整体章节分布进行了介绍。

第二章动态手势身份识别相关技术研究，首先研究了手势识别的方法，为后续手势识别奠定了理论基础，然后研究了相关性分析，为之后的特征选择提供了理论依据，最后研究了聚类算法，为之后建模算法的选择提供技术支撑。

第三章动态手势识别方法，首先详细介绍了本文采用的滤波去噪理论和数据空缺值和冗余的处理，然后介绍了数据的归一化和特征提取。接着介绍了本文优化后的半随机化的K-Means算法，最后详细阐述了本文所提出的基于动态手势的身份识别方法。

第四章系统实现

第五章实验结果与分析，首先介绍了本文的实验环境，然后介绍了实验数据的来源和实验结果的评估标准，最后对K值的选择进行了验证，验证了本文提出的方法的可行性，同时对比了其他方法，证明了本文方法优于其他方法。

第六章总结了本文所做的工作，并对本文的不足进行了分析，最后对未来进行展望。

指　标
疑似剽窃文字表述

1. 论文结构
   本文共分为6个章节，具体各个章节的安排如下：
   第一章绪论，首先介绍了本文的研究背景和研究的意义，接下来介绍了目前国内外在身份识别领域的研究现状，紧接着介绍了本文的主要研究内容和创新点，

2. 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第2部分　总字数：7816

相似文献列表　文字复制比：9.5%(746)　疑似剽窃观点：(0)

1	基于Kubernetes的资源动态调度的研究与实现 杨鹏飞(导师：黄忠东) - 《浙江大学硕士论文》- 2017-01-15	3.5%（270） 是否引证：否
2	智能移动平台融合定位技术综述 裴凌;刘东辉;龚正;赵毅; - 《导航定位与授时》- 2017/9/18 8:	2.9%（224） 是否引证：否
3	基于人工神经网络的上肢关节刚度估算 宋伟任;崔泽;杨洪鑫;韩汪洋;王玉梅;邱国文;鄢旋; - 《计量与测试技术》- 2017-08-30	2.6%（200） 是否引证：否
4	机器学习常见算法分类汇总 -- 程序猿 -- 传送门 -《网络（http://chuansong.me/）》- 2015	2.4%（188） 是否引证：否
5	[转载]机器学习的认知和算法总结【仅供参考】_笨三乐 -《网络（http://blog.sina.com）》- 2015	2.4%（188） 是否引证：否
6	机器学习常见算法分类汇总	2.4%（188）

	- 《网络（http://www.aiweibang）》- 2016	是否引证：否
7	机器学习算法汇总：人工神经网络、深度学习及其它_TW_mathematica	2.4%（188）
	- 《网络（http://blog.sina.com）》- 2017	是否引证：否
8	机器学习常见算法分类汇总 - 爱橙子的OK绷的专栏 - 博客频道 - CSDN.NET	2.4%（188）
	- 《网络（http://blog.csdn.net）》- 2017	是否引证：否
9	神经网络使用情景 - 杨航的专栏 - 博客频道 - CSDN.NET	2.4%（188）
	- 《网络（http://blog.csdn.net）》- 2017	是否引证：否
10	机器学习常见算法分类	2.4%（188）
	- 《互联网文档资源（http://wenku.baidu.c）》- 2016	是否引证：否
11	机器学习常用算法总结 - 竭尽全力的专栏 - 博客频道 - CSDN.NET	2.3%（177）
	- 《网络（http://blog.csdn.net）》- 2017	是否引证：否
12	基于感知机优化的BP神经网络邮件分类算法研究	2.3%（176）
	马秋明(导师：秦志光) -《电子科技大学硕士论文》- 2011-03-01	是否引证：否
13	基于免疫遗传算法的模糊神经网络研究	2.1%（166）
	赵红(导师：史震) -《哈尔滨工程大学硕士论文》- 2007-12-01	是否引证：否
14	胡慧君_带钢缺陷图像分类中的智能算法研究	1.7%（131）
	胡慧君 -《学术论文联合比对库》- 2015-10-17	是否引证：否
15	枣虫害图像自动识别关键技术研究	1.6%（123）
	王东(导师：周桂红) -《河北农业大学硕士论文》- 2011-06-02	是否引证：否
16	含腐蚀缺陷海洋立管剩余强度预测	1.5%（118）
	王超(导师：俞树荣) -《兰州理工大学硕士论文》- 2014-04-21	是否引证：否
17	泵送混凝土可泵性评价	1.5%（115）
	李帅(导师：侯子义) -《河北工业大学硕士论文》- 2015-05-01	是否引证：否
18	基于神经网络的高效智能入侵检测系统	1.4%（113）
	撖书良,蒋嶷川,张世永 -《计算机工程》- 2004-05-20	是否引证：否
19	天然气水合物晶体结构判别及转换方法研究	1.3%（105）
	钟煜(导师：刘武) -《西南石油大学硕士论文》- 2017-05-01	是否引证：否
20	从AlphaGo到智能汽车 AI将改变世界	1.3%（101）
	- 《网络（http://auto.163.com/）》- 2017	是否引证：否
21	小波分解法在冻土路基下土体温度预测中的应用	1.0%（79）
	文斌;吴青柏;刘永智; -《公路交通科技(应用技术版)》- 2012-09-15	是否引证：否
22	基于人工神经网络的遥操作预测仿真	0.9%（68）
	王明明;李世其;朱文革;卢亚夫; -《载人航天》- 2012-09-25	是否引证：否
23	安卓平台下基于传感器的手势识别技术应用研究	0.9%（68）
	李正山(导师：王海婴) -《北京邮电大学硕士论文》- 2014-01-08	是否引证：否
24	基于表面肌电多特征的下肢行走关键模态识别研究	0.7%（58）
	刘亚伟(导师：万柏坤) -《天津大学硕士论文》- 2009-05-01	是否引证：否
25	浅析人工智能的现状及发展趋势	0.7%（56）
	聂志伟; -《数码世界》- 2017-05-01	是否引证：否
26	12212780吴余	0.6%（45）
	- 《学术论文联合比对库》- 2014-04-23	是否引证：否
27	K-Means算法改进及其在森林健康评价中的应用	0.4%（35）
	谭浩(导师：李建军) -《中南林业科技大学硕士论文》- 2015-06-30	是否引证：否
28	在线社会网络中基于属性的重叠社区发现算法研究与应用	0.4%（35）
	王宇欢(导师：易秀双) -《东北大学硕士论文》- 2014-06-01	是否引证：否
29	Apriori关联规则挖掘算法在高校教学管理系统中的应用研究	0.4%（34）
	杨超(导师：陆鑫;李文联) -《电子科技大学硕士论文》- 2013-03-25	是否引证：否
30	基于“合作—参与”计算认知模型的半监督学习算法研究与应用	0.4%（31）
	邓超(导师：郭茂祖) -《哈尔滨工业大学博士论文》- 2009-06-01	是否引证：否

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第二章动态手势身份识别相关技术研究

2.1 手势识别

手势识别分为两大类，分别是静态手势，动态手势。静态手势指的是在某一时刻固定做某个动作，然后通过手机自带的摄像头进行图像的识别并与模板手势进行匹配；动态手势指的是在一定的时序时间内，通过手机内置传感器所产生的数据来记录用户在使用手机的过程中所产生的一段轨迹，比起静态手势动态手势的轨迹中包含了更多的动态变化的数据，信息量更丰富。

基于以上原因本文的研究重点放在对动态手势的识别。

2.1.1 动态手势识别简介

动态手势识别指的是在用户使用手机的过程中，由手机自带传感器采集的手势运动过程中的数据（比如三轴的加速度[12]、三轴的陀螺仪[13]、手指滑屏的速度和距离、手指按压的力度和面积等）来判断当前访问手机的用户是合法用户还是冒名的假用户。手势识别的过程一般涉及数据采集、数据预处理、特征筛选、手势行为建模、手势识别等过程。具体的过程如图2-1所示。

数据采集主要是采集用户在使用手机的过程中产生的各种传感器的数据，将数据存储到本地的数据库，以备后续的数据处理。

数据预处理主要是针对采集数据的过程中产生的噪声、数据遗漏、数据冗余和数据大幅度波动等情况对其进行数据的补偿或者去噪等工作。如果处理不好，会对后续的建模过程产生较大的影响，甚至有可能导致模型建立错误，严重影响后续识别的准确性。

特征筛选主要是对所有提取出来的特征进行一次评估，本文采用互信息法和皮尔森系数法对特征的相关性进行排序，相关性比较弱的特征就会被剔除，因为对模型的贡献不大，甚至会影响到模型的精确度。

手势行为建模是指用特征筛选后的特征向量来建立模型，该模型用来判断当前是否为真实用户的手势行为，模型的好坏直接影响判决的最终结果。因而，建立可靠的模型是手势识别过程中至关重要的部分。

手势识别是将输入的特征向量预处理之后和建立的模型进行匹配，如果当前手势和模型的相似度高则判断为真用户，否则判断为假用户。同一用户在不同时刻的同一手势具有聚集效应，映射到空间时会聚集在附近。而不同用户在同一时刻的手势映射到空间中的差别较大，会有一定的类间距。基于这个类间距模型可以判断出当前输入的手势是合法用户还是冒名的盗用者。

2.1.2 动态手势识别方法

手势识别方法的本质是通过模型对动态手势进行分类，比如对同一用户不同的手势进行分类，或者对同一手势不同用户进行分类。常用的手势识别方法如下所示：

1.模板匹配

模板匹配[14]的原理是将输入的动态手势与训练好的手势模型进行比对，通过一些衡量标准来计算二者之间的相似度，常用的衡量方式有欧几里得距离（Eucledian Distance）、曼哈顿距离（Manhattan Distance）和明可夫斯基距离（Minkowski distance）等。但是输入数据和模板会存在时间序列长度不一致的问题，毕竟对于同一个人在不同的时刻做同一个手势也会存在快慢的问题，因此在进行匹配的时候就需要进行时间上的校准。目前常用的时间匹配算法是离散小波变换[15]（Discrete Wavelet Transformation）算法。

2.人工神经网络

人工神经网络[16][17]（Artificial Neural Network，ANN）简称神经网络(NN)，是基于生物学中神经网络的基本原理，在理解和抽象了人脑结构和外界刺激响应机制后，以网络拓扑知识为理论基础，模拟人脑的神经系统对复杂信息的处理机制的一种数学模型。该模型以并行分布的处理能力、高容错性、智能化和自学习等能力为特征，将信息的加工和存储结合在一起，以其独特的知识表示方式和智能化的自适应学习能力，引起各学科领域的关注。它实际上是一个有大量简单元件相互连接而成的复杂网络，具有高度的非线性，能够进行复杂的逻辑操作和非线性关系实现的系统。重要的人工神经网络算法包括：感知器神经网络[18]（Perceptron Neural Network），反向传递神经网络[19]（Back Propagation），Hopfield网络，自组织映射[20]（Self-Organizing Map, SOM）和学习矢量量化[21]（Learning Vector Quantization，LVQ）等。

3.机器学习[22]

模型的建立与数据类型是分不开的，不同的数据类型会导致建模方式也有所差异。根据学习方式的不同，将机器学习的算法分为监督式学习算法与非监督式学习算法。监督式学习算法是指在训练时，每个数据记录都有一个标签，表示这个数据记录属于哪个类别。训练过程中，会不断比较预测值与标签值的差异，根据差异不断优化，最终使得预测与标签的差异不断减少，直到可以接受范围内。目前业界常用监督算法有SVM[23]、逻辑回归和决策树[24]等。非监督学习算法在训练过程中，对数据不要求记录都需要打上标签，而是根据数据本身的分布特性，能够自称为一类。非监督不需要目标标签，而是学出数据自身类特性，最后对类特性进行映射作为最终的预测结果。典型算法例如K-Means算法和Apriori算法[25]。

2.1.3 手势识别方法对比

手势识别算法没有绝对的优劣之分，对于不同的场景、不同类型的数据和不同的要求来选择不同类型的算法。有的算法识别准确率高，但是模型复杂，时间复杂度高；有的算法时间复杂度低，计算速度快，但是准确率低。结合应用场景，用户在使用手机的过程中如果频繁的收到验证提示，那么一定会影响用户的体验感。结合本文的应用场景，本文主要是对某一个用户的使用行为习惯进行建模，数据量不大，如果使用SVM或者人工神经网络算法，算法的时间复杂度高，模型复杂，运行速度慢，牺牲的时间复杂度并没有带来很大的准确性的提高。如果采用DWT算法，算法复杂度高并且没有自我学习能力。因此考虑使用一些简单的算法来处理数据量并不是那么大的数据。由于数据量小，所以希望模型具有自我学习能力，并且通过样本数据的增加不断的优化模型。

基于以上原因本文最后选择采用非监督学习方式的聚类算法中的K-Means算法，首先，K-Means算法模型简单，时间复杂度低，运行速度快，对于处理数据量不大的样本具有一定的优势；其次，K-Means的可解释强，能够给出合适的理由，更方便策略提供更为丰富的侧率；最后，K-Means能够根据自身业务被修改调整，以便适应具体业务情况，本文结合手势识别场景的

特性，对其进行优化改进，将在3.3小节中进行详细描述。

2.2 相关性分析

2.2.1 互信息法

互信息[26]是信息论理论中的一个重要的概念，与香农提出的比特信息，熵，条件熵，香农公式等是同等重要的概念。它是一种非常有用的信息度量值之一，能够体现随机变量的相关程度。比如两个随机变量分别是X，Y，则它们二者的互信息定义被定义如下：

（2-1）

其中，随机变量的X和Y的联合概率密度是 ，各自的边缘概率分别是和。再结合式子（2-1）可以得到

（2-2）

其中，H表示信息熵，例如随机变量X的信息熵可以表示为

（2-3）

信息熵在信息论中是表示某个随机变量的不确定性大小，熵值越大表示不确定信息越多。那么，通过公式2-2可以得知，互信息含义是体现某个随机变量熵减去本随机量关于另一个随机量条件熵后的信息熵大小。另外，互信息也可以看做是某个随机量确定后，另一个随机量不确定度在这个随机变量上的缩减值。所以，互信息是能够反映两个随机量在非线性上的相关特性。

关于互信息在机器学习上的应用情况，早期在1988年时，Linsker[27]发表了一篇关于感知网络自组织研究中的文章，里面讲述了如何把互信息作为机器学习的学习准则。另外，互信息在特征选择与特征提取领域中也有广泛的应用，因为互信息选择特征采取的是一种无参和非线性的方式来选择特征，能够很好的应用到实际场景中。最后，互信息在机器学习上最为典型的应用场景是十大经典算法中的ID3和C4.5等算法，将互信息作为了选择局部最优特征的准则。

2.2.2 皮尔森相关系数

皮尔森相关系数[28]是统计学中的一个数学概念，是指两个随机变量之间的线性相关性，该值的取值范围是[-1,1]，皮尔森相关系数的绝对值越大表示相关性越大。符号体现两个随机变量相关性的方向，负值表示负相关，正值表示正向相关。其定义的表达式为

（2-4）

其中表示一个样本点的坐标，通过代入统计学中的期望值，方差值

（2-5）

可以得到关系式

（2-6）

由式子（2-6）可知皮尔森相关系数是协方差比上各自标准方差的结果。协方差在一定程度反映了随机变量之间的线性相关程度。若协方差大于0，则表示正线性相关；若小于0时，则负线性相关；若等于0，则表示线性不相关。协方差值的范围是整个实域，并且不能够准确描述两个随机量的相关程度。如图2-2，图中是在二维空间上的数据点分布情况，其中X,Y变量的相关程度较小，但是数据在二维空间上分布比较离散，求出二者的协方差偏大。所以，直接用协方差表示两个随机量的相关程度是不合理的。

为了避免上述不合理的设计，需要在协方差的基础上，除以两个随机量各自的标准方差值，从而得到了最终本文采取的皮尔森相关系数值。皮尔森相关系数的取值范围在-1到1之间。若皮尔森系数值为正值，则两个随机量为正相关性，即当一个随机量增大时，另一个随机变量也增大；等于1表示两个随机量完全等价；若皮尔森系数值为负数，则两个随机量为负相关，即某一个随机量随着另一个随机量增大而减少，当为-1表示完全相反的等价；若皮尔森系数值为0时，则表示两个随机变量不存在线性相关性。

2.3 聚类算法

2.3.1 聚类算法相关理论

聚类算法[29]属于无监督方式分类的统计分析算法，也是数据挖掘中一类重要的算法研究领域。聚类算法，主要是模拟自然数据的一种现象，即“物以类聚，人以群分”。聚类实质上是对数据空间进行划分，是以子空间的内部数据点之间的相似性尽可能大，子空间之间的数据相似性尽可能的小为目标将数据空间划分为若干个子空间。聚类典型的特点是无需标记数据，基于数据分布特点学习，自动将数据归类。聚类算法整体需要经过如下四个步骤，分别是特征选择、数据相似度计算、算法选择、算法验证。聚类算法根据聚类方式不同，大致划分为层次聚类、划分聚类、网格聚类、密度聚类等。目前，聚类算法已经得到了工业界大量的使用，比如利用基于密度聚类的DBSCAN能够检测网络异常流量情况，以及优化城市规划、热点事件传播特性研究等等。

1、划分聚类

给定N个样本的数据集合，设定K个簇类，其中K<N，并且每个簇类至少包含一个样本数据，每个样本必须属于且仅属于某个簇类。划分聚类，基于初始分类后，重新计算相似度，迭代更新分类方法，不断地使聚类效果变好，即让簇类内的距离变小，簇类间的距离变大。在实际应用中，采取启发式的聚类很多，比如K-Means算法，这类算法能够在每次迭代中不断的提高聚类效果，逐渐逼近局部最优解。如果想得到全局最优解，需要多次搜索整个数据，如果数据量太大会使得计算的耗时也非常大。因此，这类算法非常适合中小规模且呈球状型分布的数据集合。根据簇类表示方式的不同，可以分为平均值算法和中心点算法。这些算法中典型的算法是基于平均值算法的K-Means算法。

2、层次聚类

层次聚类，是基于给定的数据集合，将其分层次的相似度进行不断分解，不断迭代直至满足某种要求为止。层次聚类存在两种方案，分别是自底向上方案和自顶向下方案。自底向上是从数据点作为个体簇类出发，每一步合并两个相近的簇类，直到合并完所有的簇类；自顶向下的方法是从所有点为一个簇类开始，每一步进行分裂新簇类，直到满足终止条件（比如少于多少样本认为不再生长等）。层次聚类的优点是当聚类数目发生变化时，算法不需要重新再扫描重新聚类，因为它是一次性获取整个聚类的过程，这样比较适应聚类数目不确定的聚类。层次聚类的缺点是初次构造时需要多次计算簇类内所有数据点的二者距离，并且它是局部最优的贪心算法，不是全局最优。这类算法典型的代表是CURE算法。

3、网格聚类

网格聚类是指对数据空间，根据有限的网络单元，将其划分，这些有限的单元组成了最终的网络结构。其中，处理的最小单元也被称之为网格单元。网格聚类的处理速度快，因为处理的时间是独立于数据对象本身，只与网格数目有关。网格聚类通常采取多分辨率聚类，有利于网格结构的并行处理和增量更新而且效率非常高。网格聚类的聚类效果取决于网格结构中的最底层的颗粒的力度。若是所采取的最小单元的粒度过于细化，那么处理代价会急剧上升，导致网络因性能不足而无法使用；若是最小单元粒度过于粗糙，则会导致聚类效果下降。因此如何选择合适的最小单元成为网格聚类算法的效果与性能好坏的权衡点。另外，网络聚类只是考虑层级上单元的关系，并没有考虑相邻单元的关系。这类的算法典型的代表为STING算法。

4、密度聚类

密度聚类与其它聚类存在着根本性的差异，也就是密度聚类的出发点是根据数据点的密度值进行聚类调整，而其他的聚类基本上是根据各种各样的距离进行聚类的。出发点的不同导致密度聚类具备克服基于距离聚类的缺点，比如基于距离聚类的算法只能够发现数据点的分布是“类圆形”的分类。层次聚类能够很好的将非球形的数据进行分类，但是对于数据密度不均匀，或者过于分散的数据，就很难识别了。另外，层次聚类对于内存以及I/O的消耗都是特别大的，这导致了密度聚类不能够很好的应用到实际场景中的原因。

本文的应用场景是识别智能手机用户的真伪，由于每个用户使用手机的行为习惯不一样，因此能够体现用户的差异性。另外，本文针对每个用户的数据进行分析后发现每个用户的行为是相对有限的，使用的手机场景也是有限的。因此数据会形成有限的行为簇。根据采集来的数据的分布可以得到每个用户的数据大致呈现球状的类效果。因此选择适合分析球状数据的K-Means算法作为本文的算法，因为它能有效的处理呈球状的数据、可解释性也强并且实现简单。

2.3.2 K-Means算法理论

K-Means算法[30][31]是机器学习中十大经典算法之一，也是聚类算法中的典型，更是工业上应用较多的算法之一。K-Means算法目的是将一个数据集合划分为若干个簇类，让属于同一簇类的数据对象之间能够高度相似，不同类的数据对象之间能够有较大的差异。该算法的基本思想是首先取定K个中心以及K个中心的初始位置，再计算每个数据点离各个中心点的距离，根据距离的远近将其划分到不同的中心类中。接着是迭代替换中心位置，直到中心位置稳定下来，收敛。其示意图如图2-2所示：

K-Means的算法实现思想是EM的算法思想，主要的算法流程如下：

1．人为设置K值，且随机取K个样本点作为初始中心点，第K个初始中心点的向量表示为,其中n表示特征向量的维度；

2．根据K个中心计算M个样例所属的类别，根据式子（2-6）计算

（2-6）

其中表示样本属于类；

3．重新计算各个类目的中心，即

（2-7）

其中表示样本i属于j类，表示属于j类的样本数目；

4．根据式子（2-8）比较新旧簇类中心的距离平方和

（2-8）

算法迭代的终止条件为J接近于0（此时表示聚类已经达到稳定和收敛）或者迭代次数超出了最大的迭代次数。

5．重复上述步骤2和4直到算法收敛或者超出限定的迭代次，最后得到最终的K个聚类中心。

上述的流程中，K-Means使用的衡量样本间相似度的度量方法是欧式距离。衡量数据的距离还有其他的度量，比如曼哈顿距离，切比雪夫距离，这些距离都是属于闵可夫斯基距离的一个特例值，基于范数值的不同而不同。还有其他的度量方式，比如汉明距离，夹角余弦距离和杰卡德距离等。本文采取的是欧式距离，因为欧式距离能够很好的反映样本在数据空间上的差异。

从上述K-Means算法的聚类过程可以看出传统的K-Means存在两个比较关键的点，分别是K值的确定，以及初始K个类中心的确定。这两点如果选择的不当，会导致聚类效果陷入无法收敛或者仅仅局部最优的情况。针对这个情况目前有很多学者对K-Means初始中心的选择做了优化，提出了很多改进的K-Means算法。典型的有K-Means++[32]，ISODATA，Kernel K-means。

1、K-means++[32]

K-means++与K-means算法基本相似，在选择初始中心时具有差异，也就是针对K-means的完全随机化选择初始中心进行了优化。K-means++优化过程如下：假设共有K个初始中心，已经确定了n个初始聚类中心，其中n<K；当需要选择第n+1个聚

类初始中心时，采取的是选择离n个初始中心最远的样本点作为n+1的初始类中心。这种选择策略主要是基于远离已有的初始中心，更容易成为新的初始中心。这个改进算法简单且有效，但是容易受到异常点影响而导致效果变差。

2、ISODATA[33]

ISODATA算法主要是解决K-means算法中K值的确认，因为K值的确认直接影响聚类效果。在很多实际场景中，应用者可以根据实际场景情况，做出经验上的K值确定。然而，当对应一个陌生的场景，经验知识缺失时，并且数据量很大，维度也很高，那么这种K值试错的代价是相对很高的。针对这种问题，ISODATA对其进行了改进，主要思想是基于样本点数目情况，过少的样本点的类可以直接舍弃，过多的样本点可以进行分裂，也就是说类似生物学中的细胞分裂，与细胞死亡的过程模拟。这类算法能够对K值进行优化，本文的场景具备先验知识，且数据量不大，维度也不高，从而这类算法不适应。

3、Kernel K-means[34]

Kernel K-means算法没有对传统的K-means中的两个典型缺陷进行修改，而是考虑其依赖的度量值的改进，即欧式距离计算样本相似度。考虑一些场景不适合直接应用欧式距离，因为维度过高，导致无法计算，比如上亿级的维度，那么此时需要降维度，而又不是移除给特征公式，从而利用支持向量机的核函数思想，利用核函数将高纬度进行保信息的映射，从而能够达到提升聚类效果。

总体上来说，这些改进都完善了K-means的算法模型，但是所付出的代价是加大了算法的复杂度和模型的复杂度，导致模型的泛化能力不够，或者不利于实现。针对这些问题，本文结合业务和动态手势数据的特点，提出了一种基于半随机化选择初始中心的K-Means算法的动态手势智能终端身份识别方法。

2.4 本章小结

本章介绍了本文所用到的一些关键技术，其中第一小节包括动态手势识别的简介，几种常见的动态手势识别方法，包括模板匹配、人工神经网络和机器学习。接下来对比了几种手势识别方法的优劣。第二小节详细介绍了两种相关性分析的方法，有互信息法和皮尔森相关系数，为之后的特征选择奠定了理论基础。最后一小节详细介绍了聚类算法的相关理论，介绍了划分聚类、层次聚类、网格聚类和密度聚类，紧接着对本文选择的K-Means算法的步骤进行了详细说明，同时介绍了几种优化后的K-Means算法。

指　标

疑似剽窃文字表述

---

1. 该模型以并行分布的处理能力、高容错性、智能化和自学习等能力为特征，将信息的加工和存储结合在一起，以其独特的知识表示方式和智能化的自适应学习能力，引起各学科领域的关注。它实际上是一个有大量简单元件相互连接而成的复杂网络，具有高度的非线性，能够进行复杂的逻辑操作和非线性关系实现的系统。
2. 空间进行划分，是以子空间的内部数据点之间的相似性尽可能大，子空间之间的数据相似性尽可能的小为目标将数据空间
3. 若干个簇类，让属于同一簇类的数据对象之间能够高度相似，不同类的数据对象之间能够有较大的差异。

3. 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第3部分　总字数：10828

相似文献列表　文字复制比：0%(0)　疑似剽窃观点：(0)

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第三章动态手势识别方法

在数据挖掘的过程中，建模是数据挖掘过程的核心部分。本章对传统K-Means算法的初始中心数目和位置的确定做了优化，并且基于聚类的结果提出了一种基于动态手势的身份识别方法。考虑到预测结果的重要性，给新样本提供了回收机制，形成闭环的结构，最终使系统能够自适应的识别伪用户。

3.1 数据预处理

经过智能终端直接采集的数据，由于采集环境的不稳定，难免会有受到环境噪声的影响。因此为了避免对后续特征提取和建模产生影响，本文对原始数据进行去噪、空缺填充和归一化的预处理。

3.1.1 动态手势数据的滤波

在采集动态手势数据的过程中，由于环境的不稳定，比如手臂的抖动、行走时的晃动或者坐车的颠簸，这些因素都会导致数据采集的过程中出现小范围的波动，或者突然的极值。具体如图所示，因此原始数据需要经过去噪[35]处理后才能用于后续的步骤。

由图可知，传感器在一段时间内采集的数据并不稳定，会有细微的波动。图-是手机静止状态下采集的加速度[36]的数据，可以看出即使在静止状态下，传感器也会有细微的波动。

这些波动相比采集的动态手势的数据属于高频信号，由此可见，对智能终端进行数据采集的过程中有用的数据主要集中在低频阶段，而干扰主要集中在高频的部分。因此，为了解决这个问题本文采取滤波的方式对加速度和陀螺仪这种并不是很稳定并且容易受到环境影响的数据进行去噪处理。

一般常用的平滑去噪的方式有以下几种：

1.简单移动平均线（Simple Moving Average）

简单移动平均线的原理是以一段时间内的平均值来代替当前时刻的瞬时值，它通过求平均值的方式让波动对数据产生的影响降低到最小。具体公式如下所示：

（3-1）

以加速度为例，其中ASMAnow表示当前时刻加速度传感器采集的加速度瞬时值，i表示当前时刻，i～（i-m+1）表示i时刻之前的m个时刻，m表示滤波窗口的大小，即以多少个瞬时值的平均值来表示瞬时值。窗口值取值的大小直接影响到滤波的效果，m值越大，平滑的效果越好，但同时会覆盖掉一些有用的信息，一些手势数据的变化也被滤掉了，导致无法提取有用的信息。m值太小，虽然不会过滤掉有用的信息，但是也保留了高频的干扰信号，对后续特征的提取和建模也会产生影响。因此简单移动平滑去噪效果的好坏直接取决于窗口值的设定。

2.低通滤波（Low-Pass Filter）

由上文分析，本文采集数据的干扰信号主要来自高频区，因此采用低通滤波的方法能有效的过滤掉高频的干扰。低通滤波器使用的最广泛的是巴特沃斯滤波器[37]和切比雪夫滤波器[38]。

低通滤波的原理是对于小于截止频率的信号会保留下来，对于高于截止频率的信号就会被过滤掉，但实际上工程中只能实现在截止频率附近的信号的削弱和远大于截止频率的信号的去除，因为有一个渐变的范围，这个范围的大小通过滚降速度来衡量。滤波器的阶数越高，滚降速度越快，但实现复杂，阶数越低，实现简单，但滚降速度越慢。因此如果采取低通滤波的方式来进行去噪，核心在于截止频率和滤波器阶数的选择

3.小波包分解（Wavelet Packet Decomposition）[39][40]

通过上文的分析可知，简单移动平均线受窗口值影响太大，窗口值太小无法起到平滑的作用，窗口值太大则容易过滤掉有用的信息。低通滤波器也同样存在这样的问题，关键在于截止频率的选择。因此采用小波包分解的方式，不仅可以分解出低频的有用信号，同时也分解出高频的噪声信号，同时结合皮尔森系数法对其进行预处理，具体原理如下。

首先采用小波包分解[41]将原始数据进行3层小波包分解，如式子-所示：

（3-2）

其中origin表示原始数据，d0～d7表示将原始数据从低频至高频分解为8个频域段的信号，下标越大表示所在的频域越高。具体的分解效果图如图3-1所示：

由图可知低频的信号的变化平缓稳定，高频区的信号变化很快，与原始信号进行比较可知，低频信号图与原始信号图的相似度很高，因此动态手势的信息主要涵盖在低频的部分，而高频区主要是手机内部传感器的工频噪声、人类肢体的生理晃动和环境的噪声。将动态手势的有用信号从原始信号中提取出来的关键在于重构的过程中组合分量的选择，组合分量选择过少就无法完整的描述过程中动态手势的轨迹，组合分量过多就无法达到平滑去噪的目的。于是采用皮尔森系数作为选择重构的依据。

皮尔森相关系数公式如下：

（3-3）

分子部分是X和Y两个变量的协方差，分母部分是两个变量的标准差的乘积。协方差是用来表示两个随机变量相关程度的指标，如果两个随机变量随着一个的变大另一个也变大，则这两个变量正相关，协方差为正值；反之两个变量负相关，协方差为负值。这个式子的值越大表示二者的相关性越大，反之相关性越小。原始数据经过小波分解后的8个频域的信号，相邻的频域的信号两两之间求皮尔森系数值，由于动态手势的信号主要集中在低频区，噪声主要集中在高频区，因此低频区之间的相关性会很高，高频区之间的相关性也会很高，在低频与高频直接的过渡区的相关性会很低，因此对分解后的信号相邻信号之间两两求皮尔森系数值，会呈现出先下降后上升的趋势。通过计算找到转折点，就可以有效的将动态手势信号从噪声信号中分离出来。具体的计算值如表3-1所示：

表3-1 皮尔森系数

频段	R	频段	R
d0-d1	0.962317	d4-d5	0.835712
d1-d2	0.942156	d5-d6	0.719934
d2-d3	0.697792	d6-d7	0.581236
d3-d4	0.789214		

由表可知d2到d3是一个下降的趋势，d3到d4是一个上升的趋势，因此d3就是动态手势与高频噪声信号的分界点。因此过滤掉d3之后的高频噪声信号，再将d0～d3的低频信号进行重构得到的重构后的信号如图3-2中的去除噪声后的波形，可以看出重构后的信号不但保留了原来信号的信息，同时信号更平滑了。

3.1.2 动态手势数据的空缺和冗余处理

在数据采集的过程中常出现一段时间内数据出现大量的空白情况，可能是因为设备传感器的不稳定，也可能是外界或者人为的干扰产生的。对于不同的数据和不同的情况本文采取不同的措施。

1．直接丢弃

有的特征采集的过程中数据分布的非常密集以至于每次采集都不是采集一个瞬时点的数据，而是采集一小段间隔的平均值作为瞬时值，对于这种情况一小段的空白数据段对模型的影响不大，所以采取丢弃的方式把空白数据直接丢弃，在本文中对于陀螺仪的数据空白采取的是这种直接丢弃的方式。

2．平均值替代

如果当前采集的特征的重要性很大，在模型的建立过程中起着很重要的作用，则不适合采用直接丢弃的方式，本文对于特征取值变化不大的数据采取平均值替代的方式，因为特征数据的波动很小，所以用一小段的平均值替代当前空白区的瞬时值是合理的。在本文对于滑屏起始位置采取的是平均值替换的方式。

3．手工填补

手工填补方式指的是根据数据的分布情况和变化趋势，预测空白处应该呈现出怎样的数据，以预测值来替换空缺的数值。

冗余数据指的是在数据集合中存在一段连续的完全相同的记录，在数据值上完全无法区分，只有时间记录不同，其他的完全相同，这表示在某一时间对记录进行插入时出现了重复，此时这个记录是冗余的。本文采取相邻记录之间两两比较的方式进行去重。

3.1.3 动态手势数据的归一化

在机器学习和数据挖据中，归一化是很重要的一个步骤，它的目的是使得不同取值范围的特征具有可比性，而且它可以加快求解的速度和提高精度。如图3-3所示，如果不同特征之间的取值范围相差很大的话，数据就会呈现出扁平的状态，此时如果采取梯度下降的方式，将会经历很多次的迭代才能收敛，有时候甚至无法收敛。如图3-3中的折现所示，很曲折，表示算法的求解速度会很慢。

如果将数据进行归一化，数据的分布如图3-4所示，数据的分布就会呈现一个比较圆的状态，此时进行梯度下降很明显下降的次数减少了，在图上的呈现就是少走了很多弯路，因此将数据进行归一化有利于算法的收敛速度。

另一方面是因为归一化有可能提高精度，当分类器需要计算样本之间的距离时如果一个特征值的取值范围非常大，则这个取值范围特别大的特征对距离的计算将会产生很大的影响，从而导致与实际情况不符（例如所有特征值中有一个特征的取值范围很小但是此特征对模型很重要，但是由于取值很小而无法影响到距离的计算从而造成计算的偏差）。

但是也不是所有的模型都要进行归一化，比如概率模型就不需要进行归一化，因为概率模型并不关心变量的取值的大小，只关心变量的分布和变量之间的条件概率，比如决策树，因此变量的取值对模型不会产生影响所以不需要进行归一化。

其他情况特征的数据是否要进行归一化取决于模型是否具有伸缩不变性，比如SVM（Support Vector Machine）如果在特征的各个维度进行不均匀的伸缩变化后，它的最优解是会改变的，与原来不变化之前的最优解不等价，此时对于这种模型，如果所有的特征数据的分布范围很接近则可以不用进行归一化，否则必须进行归一化，以此来防止特征中某一个范围很大或者范围很小的特征占据主导地位。像逻辑回归这种模型，经过各个维度的不均匀伸缩后，最优解仍然保持和原来的一致，对于这类模型，是否进行归一化理论上并不会改变最优解，所以不会影响解的最终结果。但是，由于在实际工程算法实现时经常采用迭代的方式，如果目标函数的形状过于“扁”，这将会影响算法收敛的速度，甚至导致算法无法收敛。因此无论是对于伸缩可变的模型还是对于伸缩不可变的模型，都建议先采用归一化的方式对特征数据进行预处理，为后续的建模做好准备工作。

本文采用线性函数归一化(Min-Max scaling)的方式对数据进行归一化。线性归一化的公式如下：

（3-4）

其中X表示归一化之前的特性向量，min（X）表示此特征向量取值范围的最小值，max（X）表示取值范围中的最大值，通过这个线性公式将将原始数据转换到[0,1]的取值范围，实质是对数据进行了等比例的缩放，用原始的数据减去最小值再除以最大值与最小值的差，最后得到归一化后的特征向量，以此来解决各个特征向量取值范围相差过大导致的不同取值范围的特征无法在同一标准下进行度量的问题。通过这种方式进行归一化的优势是实现起来快速简单。

3.2 特征提取

特征提取主要的目的是为了缩减数据的维度，提高之后模型的运行效率，以及去除无效的数据。其基本任务是从众多特征中找出最能反映目标属性的特征，主要依赖特征的相关性来寻找。考虑特征关联的多样性，本文同时考虑属性特征与目标特征的线性关系以及非线性关系。特征提取的具体过程如图3-5。

首先，输入原始数据需要经过一定的数据预处理，这里的处理方式也是3.1小节提到的数据去噪、数据变换、归一化等；接着数据分为两路进行，一路是经过非线性的互信息方式进行相关性的排序，另一路是经过线性的皮尔森系数方式对特征进行相关性的排序；最后采取一定策略对两种方式选择的特征进行合并得出最终的有效特征集合。

数据需要先确定正负样例，本文先选的某个用户为正样例，则其他的用户就为负样例。利用接口调用python中的sk-learning包中的工具获取互信息值和皮尔森相关系数的大小；接着直接按照大小排序，得到了两组有序的特征序列。皮尔森系数排序的结果如表3-2，互信息排序的结果如表3-3。

表3-2 皮尔森相关系数排序表

字段字段名字皮尔森相关系数值相关极性

mingy 最小gy 0.2824 正相关

avgergy_x 平均gy_x -0.2351 负相关

表3-3 互信息排序表

字段字段名字互信息值

avgergy_y 平均gy_x 0.312

基于上述的排序，本文在汇总时，采取交替取特征的方式，选择特征重要性高的，基本思想是类似于排序算法中的归并算法实现。最终，得到相关性排序靠前的几个特征如表3-4所示：

表3-4 筛选后的特征表

字段字段名字皮尔森相关系数值互信息值

3.3 半随机化K-Means算法

3.3.1 K-Means中K值的选择

传统K-Means[42]算法关于初始中心个数K的选择采取的是随机的方式，而本文聚类的数据是用户使用手机的行为，由于每个用户的行为都是经过很长的时间形成的并且种类是有限的，所以理论上是可以枚举出所有的K值。基于以上原因，本文对K-Means算法中的K值的确定采取枚举的方式，通过遍历多个K值，并分别计算各个取值下聚类效果的评价指标，最后选择聚类效果最好的K值。

聚类效果评估指标可以分为带标签的评估方式和不带标签的评估方式。对于带标签的评估指标，衡量聚类是相对精准的，但是由于本文是针对某一用户的行为数据进行聚类，并不存在标签，并且聚类结果是用于后续的打分模型，不是用来进行分类，因此不适合使用带标签类型的指标。所以本文采取不带标签的指标。目前使用比较广泛的不带标签的指标主要有以下几种：

1．仅考虑类内紧凑性的CP指标值。首先通过式子（3-5）计算出属于i类的所有样本点离i类中心点的距离的平均值。

（3-5）

然后通过式子（3-6）求出K个类的类内距离的平均值来体现整体的聚类效果，值越小表示越紧凑。

（3-6）

2．仅考虑类间的隔离性，SP指标值的计算如式子（3-7）：

（3-7）

其中，

（3-8）

表示类间距离的组合数目， 表示类间的距离，通过求任意两个类中心距离和的平均值，来表示聚类效果，值越大表示类间的隔离性越好。

3．同时考虑类内紧凑性与类间隔离性，戴维森堡丁[43]指数指标：

（3-9）

其中，

（3-10）

的分子部分体现了两个类类内的紧凑性，分母部分体现了两个类类间的隔离性。该值越大表示两个类的类内间距很大，即不紧凑，类间间距很小，隔离性差，因此式子的值越大聚类的效果就越差，其值越小表示类内越紧凑，类间越隔离，聚类效果越好。最后通过取最大值来表示最差的情况，以此来表示类i的聚类效果，最后求均值表示整体的聚类效果。

4．直接考虑样本点的紧凑性与隔离性，邓恩指数指标：

（3-11）

针对簇类m与n的样本的紧凑性与隔离性，式子（3-11）的分子体现了两个簇类的类间距离，取两个类的样本的最小距离作为两个类的类间距离，接着取所有两两组合的簇类中最小的最为整体的类间距离；式子（3-11）的分母部分是取每个类内的最远距离作为类内距离，取所有类类内的最大值表示整体的类内特性。其值越大表示类间距离越大，类内距离越小，聚类效果越好。

根据本文需求，最终选择了戴维森堡丁指标。因为单独考虑类内紧凑性的CP值和单独考虑类间隔离性的SP值都不够准确，无法全面和客观的比较聚类效果。而邓恩指数指标，类内和类间的距离直接通过最大值和最小值来表示，容易受异常样本点的影响，不够稳定，因此本文选择相对稳定的戴维森堡丁指标作为衡量K-Means聚类效果好坏的依据。

由于本文的研究对象是对某一个用户的行为进行聚类，每个用户的行为习惯都是长年累月累积下来的，而且比较稳定。因此聚类的数目是很有限的，根据这个特点本文采用枚举的方式，对聚类初始的类中心的数目进行遍历，用戴维森堡丁指标对聚类的效果进行评估，最后选择最佳的聚类数目。

3.3.2 半随机化选择初始中心

由于每个用户使用手机的行为习惯是常年累月所形成的，并且行为类的数量有限，因此数据之间的差值不会很大。基于以上原因，本文采取半随机化的方式替代之前传统K-Means算法中的完全随机化方式来选择初始中心，目的是为了使算法快速收敛并改善模型的效果。在使用半随机化之前需要对数据进行归一化，因为传统K-Means采用的是欧式距离，没有进行取量纲化，这容易使最后聚类的效果偏向于某个取值比较大的特征，从而影响聚类的准确性，因此需要先对数据进行归一化，使得每个特征的取值都在0和1之间。本文选择简单有效的极值法进行归一化。数据经过归一化处理后进入半随机化的处理，本文以二维举例说明，具体步骤如下所示：

1．首先确定维度界限，按照上述方式归一化后，边界为0或1；

2．根据K值，将数据空间在各个维度在上均等划分为K个数据区域，二维空间就会被划分为个数据区域，三维空间就会被划分为K3个数据区域，以此类推，然后统计所有区域的样本数量并按照降序排序；

3．基于上述的排序，选择前K个区域作为候选的种子区域，并且这些种子区域满足维度上的唯一。维度的唯一性指的是如果某区域被选中，和选中区域在同一行或者同一列的区域都不能够再作为候选的区域，然后从剩余的区域选择区域样本数量最

多的区域作为候选区域；以此类推，最后选出K个候选区域；具体例子如图3-1，当选中B区，那么ACEH就不能够被选择，只能够在DFGI中选择，在候选集合中，继续排序，选择了F，那么类似，只剩下G区域可选择，从而最终选择了BFG三个区域作为随机区域。

4．基于3中的候选区域，图中选出来的为BFG区域，在各自的区域中采取均匀随机选取的方式，选择K-Means的初始中心值。图中红色点标记的样本为随机选择的初始中心。至此K-Means算法初始中心的选择就结束了（图中BFG区的红色样本点为选定的初始中心）。

3.4 基于动态手势的身份识别方法

3.4.1 打分预测模型

算法模型的建立包括训练过程和预测过程，一般二者是相互分开和独立的，但是本文考虑到无监督模型形成的模型，可能泛化能力不够，或者收敛速度不够而影响模型的效果。因此在预测验证后，还将对样本进行回收，并且对满足一定条件的样本进行加权并重新进入训练过程，对模型进行优化，如此使得样本的权重得到改变，模型得到适当的调整，使得模型不断的优化和完善，最终逼近最优的模型。

图中的打分预测模型，主要是由两个部分组成，一部分是单点聚类效果，另一部分是类的加权效果。假设新样本点x预测为当前真用户的概率大小为P，新样本点被划分至某个行为类后，类中心μ与新样本x的距离设为d，根据K-Means聚类误差公式（距离平方和公式），计算出新样本带来的误差值为：

（3-12）

其中，x表示样本，表示某个类中心的向量，则单个样本的误差值即为距离值。对于模型而言，误差越小（距离越小），则样本属于当前真用户的某个行为类的概率越高，从而P值越高。由此可知P值与d值成反比例关系。

根据经验以及相关应用，例如温度的衰减公式和信号的衰减公式，都是呈指数族的关系，因此假设P值与d值服从指数族函数关系，如式子（3-13）：

（3-13）

K-Measn聚类适合球形分布的数据和高斯分布的数据，因此假设样本服从高斯分布，即。其中为已知的某个行为类的中心向量，基于统计求出该类中心样本的方差值，根据式子（3-14）分别求出α和β。

（3-14）

通过上述式子可以求出某一行为类的概率值，但是由于每个行为类所占总行为的比重不一样，这表示有的行为为用户主要的行为，有的行为为次要的行为，而主要的行为的置信度更高，因此在概率的基础上还需增加置信度的权重来表示最终的打分公式。具体公式如下所示：

首先计算得到x样本属于行为类k，计算公式为：

（3-15）

确定样本所属行为类后，通过统计学计算得到公司-中的参数α、β和μ，从而确定最终的打分公式，如下所示：

（3-16）

其中，，表示类k的样本集合，表示类K的类中心向量，表示样本的权重值。初始训练数据的样本权重为等权，即

（3-17）

其中N表示训练数据的集合大小。

3.4.2 样本加权

Yoav Freund和Robert Schapire于1995年提出了AdaBoost[44]（Adaptive Boosting）算法，该算法的核心是通过改变样本的权重和弱分类器的权重来实现算法的自适应增强，从而达到算法最优的状态。具体实现的方式是每次分类器分错的样本将被增大权重并且参加下一次训练分类器的过程中，算法将进行多次训练分类器，当算法低于某个预设的错误率或者算法迭代次数大于预设的最大迭代次数值时，则结束算法的迭代，完成算法的输出。

Adaboost 算法思想主要有以下三步：

1. 首先，初始化训练数据的权值分布，如果有N个样本，每一个训练样本在最初训练时将被赋予相同的权值1/N；

2. 其次，样本若是经过训练弱分类器后的结果是预测准确，那么该样本点的权重会被下降，因为该样本所以提供的信息已经被算法学习到了，从而降低权限；若是预测结果不准确，那么该样本点会被加权，以便算法能够学习到。这样通过改变样本的权重，通过权重突出没有学习到的信息，从而能够达到算法效果的提升。

3. 最后，针对弱分类器，是需要根据弱分类器预测准确性来给出不同的权重。弱分类器预测越准，那么以为这该分类的权重需要加大，从而达到提升整体效果的预测；若是弱分类器预测准确度不够，就降低其权重，甚至可以直接去掉该分类器保证整体算法效果不被拉低。总而言之，是好的弱分类器需要被重视，坏的弱分类器需要打压的策略来改变弱分类器的权重。

本文的想法来源于Adaboost，起初对训练样本设置的值都是等权的，当判决的结果与实际不符时则表示样本的比重需要被调整以便分类器更好的学习此类样本的特性，而与Adaboost不同的是，Adaboost在重复训练的时候初始样本的比重也会被调整，而本文对初始样本的权重不做调整，只把测试样本作为特殊样本进行权重的调整，这样的优势在于既能让模型快速的学习到样本的所有特征，又能保证原来模型的稳定性，不会产生过度调整导致原来已经学习的特征被丢弃。另一点和Adaboost不同的是，Adaboost是基于误差进行样本权重的调整，而本文算法调整样本权重的依据是最终调整的结果在下次进行训练模型时是否能让模型学到上次没有学到的特征，以达到完善模型的目的。

预测时，在判决验证时具有以下4种情况，每种情况需要结合实际应用场景以及含义进行不同的处理。

1. 如果属于本用户的新样本的数据被预测为非本用户，这种错误是体现了模型的泛化能力不够，导致模型没有学到此样本的特性，需要额外添加其他手段使得模型能够快速学习到这类样本的特性，本文对这种情况进行加权处理，使得最终的得分值能够得到提升，达到预测的结果为本用户数据的目的；

2. 如果属于本用户的新样本被预测为本用户，则说明模型能够正常学习到当前新样本的特性，并且能够正常运行，这种样本特性和之前训练样本的特性是一致的，不需要加权直接回收即可，相当于权重为1；

3. 如果不属于本用户的新样本被预测为本用户，这表示模型过于泛化，也就是模型并没有完全收敛，或者由某个类样本不足导致的，此时需要对这个样本进行加权使得样本最终的得分能够下降，以便模型能够收敛，不至于过于泛化；

4. 如果不属于本用户的新样本被预测为非本用户，这种就是模型正常运行，且无需回收，因为模型主要是对属于本用户的样本进行学习，避免学习不属于本用户的样本的特性，所以对这种情况不予回收，相当于权重为0。

下面将详细讲解本属于本用户的新样本却被预测为非本用户这种情况，如何通过对新样本进行加权来修正模型，使得模型能够学习到这种特性。假设新样本是属于本用户的行为数据，其样本点的特征向量为，并且该样本点所属的用户的行为类为k，k类的类中心为，根据上一节的打分公式新样本的得分如下：

（3-18）

根据判决条件可知时，会把新样本点判决为非本用户的数据点，其中T值表示判决阈值。为了表达的含义更为清晰，将式子调整成如下形式：

（3-19）

将公式（3-20）代入（3-19）得到最终的表达式

（3-20）

其中式子（3-20）表示样本到所属类中心的距离值。

（3-21）

假设新样本的归一化的权重为，那么通过上述的式子可以变得如下式子：

（3-22）

（3-23）

其中，分别表示加权后的距离值与分值，如果要满足则需要满足：

（3-24）

值得注意的是，在调整样本的权重全是赋值与之前训练数据等权中的值，也就是说的值是包含上次训练属于本类，以及增加一个等权的新样本的情况，从而为一个常数。式子中除了其他参数都已经确定取值，只有未知，调整式子为如下的形式：

（3-25）

由式子（3-25）可知这是一个关于一元二次不等式，通过上述式子可以求出取值范围，选择最小值作为相对最佳的值，尽量通过较少的改变来调整样本的权重比达到提高得分的目的。并且的前提条件是必须是大于0且小于1的值。

另外，上述不等式存在无解的情况，这说明样本点离该类的中心很远，导致通过加权的方式也无法达到要求。这种情况本文采用样本伸缩的方式来解决。如图所示如果将样本向量进行延长，在无限延长的过程中样本距离类中心的距离是先缩短后增大，存在一个最小值，由几何知识可知最小值为类中心在新样本向量延长线上的投影所在的点，由此可以计算出新样本的最佳权重值。

如上图，伸缩最佳点在处，也就是垂直交界处，其中是类中心向量与样本向量的夹角值。由此可以得到：

（3-26）

通过上述方法可以求出新样本到每个类中心的最佳值，这里的最佳值指的是满足了条件的最小变化值，或者没有满足条件的最佳变化方向。本文的策略是优先选择能够满足要求的最小的值，如果无法满足上述不等式，则选择最佳变化方向中的的最小值。如此这样得到新样本的权重值。由于权重进行过归一化，所以同时需要更新训练数据中的权重值，假设训练数据的样本初始权重为。如下式子更新：

（3-27）

其中，，表示样本点i更新后的样本权重值，表示训练数据样本集合权重的和，训练集合数据具有封闭性，从而该值为1。

类似的对于非本用户样本预测为本样本，不同在于d的修改目的是使得得分能够下降从而能够正确地排除该样本点为本用户数据点。

3.5 本章小结

本章节首先介绍了本文对数据的预处理，包括采用小波包分解和皮尔森系数法对原始数据进行滤波去噪、原始数据对空缺值和冗余的处理、数据的归一化，接着介绍了本文如何对特征进行筛选和提取，然后重点介绍了本文对K-Means算法的优化，主要是对初始中心数目选择的优化和初始中心点选择的优化，最后通过上述调整K-Means的聚类效果，基于优化后的聚类，本文结合实际应用场景，提出了一种基于动态手势的身份识别方法，并增加样本回收加权机制，最终使得模型影响数据，数据不断优化模型，使模型不断的趋于最优。

第四章动态手势识别系统

本章主要讲述动态手势识别系统以及核心功能的实现。主要对总体设计，客户端设计，服务端设计进行分别讲述。其中，最为核心的服务端，本章重点介绍线上线下的实现过程。

4.1 系统总体设计

本系统的设计目的是为了能够增强对用户隐私的保护，能够及时的发现伪用户，以供后续做相关的保护措施。本文提出了一种动态手势识别完整的系统，见图4-1，该系统能够满足快速识别伪用户的功能，达到对用户及时的保护。

图 4-1 动态手势识别系统框架

本系统的设计主要分为三大部分，分别是智能手机端的客户端逻辑部分，数据通信接口逻辑部分，后台服务逻辑部分。网络通信是完成客户端与服务端的连接功能；客户端是完成与人交互以及相关业务逻辑处理；服务端是识别系统核心，完成对用户真伪识别的逻辑。

手机客户端部分，主要目的是能够收集用户数据，并且能够顺利传输至后台服务中；能够与用户进行人机交互功能；以及能够根据后台服务的结果做出相应的安全策略保护用户的隐私，将在4.2中详细介绍。

后台服务端部分，采取两个核心流程部分设计，一方面是数据训练流程，主要目的是训练预测模型，这里会涉及到存储，数据预处理，数据属性选择，数据训练，以及模型存储等问题；另一方面是数据预测流程，目的是快速做预测，要求实时效果，对用户无感知，这里主要涉及加载模型，特征快处理，快预测等逻辑。将在4.3中详细介绍。

网络通信部分，核心功能是完成客户端与服务端的通信，其中的客户端通信模块的实现在客户机上，服务端通信模块在后台服务端实现。网络传输中数据包中的数据格式采取的json数据格式。客户端与服务端通信采取的是XXX框架，能够稳定的在客户端与服务端之间进行传输。

4.2 客户端

客户端包括前端交互界面、数据库、数据编解码和通信模块。交互产生的传感器的数据通过本地数据处理模块后存入本地数据库，当新数据量达到一千时，数据从数据库输出，输入至数据编解码模块进行编码，将数据打包成json格式输入至通信模块进行传输，通信模块同时还负责接收服务器返回的数据。具体如下：

1.客户端交互界面

客户端的前端交互模块包括3个部分，分别是注册、登录和采集。注册部分需要填写用户名、密码、性别和年龄等个人信息，登录部分包括用户名和密码，数据采集部分包括用户名、已采集数据的数量、文字阅读交互页面和当前各个传感器采集的数值，具体如图4-2所示：

图4-2 客户端人机交互界面

2.数据库

主要采集的数据类型如表4-1所示，包括用户注册的信息（用户名、密码、性别、年龄），用户手势滑动种类（单击、双击、轻浮、长按、滑动），加速度（三轴的加速度和平均值），陀螺仪（三轴的陀螺仪和平均值），速度（X轴和Y轴速度），位置（开始和结束的位置及距离）和压力（按压力度及面积）等信息。

表4-1 数据采集的字段

数据类型字段

用户注册信息用户名、密码、性别、年龄

用户手势滑动种类单击、双击、轻浮、长按、滑动

加速度三轴的加速度和平均值

陀螺仪三轴的陀螺仪和平均值

速度 X轴和Y轴速度

位置开始和结束的位置及距离

按压力度按压力度及面积

4.3 服务端

服务端的实现主要采取两个方向实现，分别是离线的模型训练，在线的模型预测。目的是能够快速响应线上请求的同时，能够及时更新模型，使得系统能够及时根据变化调整系统预测。后台服务端的实现详细框架图如下图4-3

图4-3 后台服务系统框架

数据经过网络传输后到服务端，数据流统一进入数据存储后，一方面，直接流向数据库后，接着数据训练从中拉取数据进行训练过程；另一方面，流向数据预测直接进行进行预测。在逻辑上后台将其划分了三部分，分别是数据存储、数据训练、数据预测。

数据存储，主要包含了解析请求的数据；拉取后台mysql数据库数据的逻辑；以及存储数据的mysql数据库。数据解析的过程主要由DataParse抽象类完成，主要的任务是实现多种数据结构的解析，例如json，xml，libsvm等等。不同数据形式都用一

个实施类来完成，比如JsonDataParse、XmlDataParse、LibsvmDataParse类，它们的核心方法是继承父类中的Parse解析方法，对数据的解析。类的详情如图4-4。

图 4-4 解析数据类簇

另外，存储接口中，重要的模块就是存储逻辑模块，主要是实现基于sql语句对数据库数据进行读写操作。主要通过类MyMysql实现，类中实现连接msyql数据库，以及重要的查询接口、写接口、以及其他辅助类接口，类的详情介绍见类图4-5。下面详细介绍连接mysql的整个细节，如下4-6时序图：

图 4-5 MyMysql类结构图

图 4-6 连接mysql时序图

数据训练过程，也就是本系统的离线部分，主要是能够批量处理大量数据，做到每天能够输出一个相对准确的预测模型文件。主要包括特征工程处理模块，基于优化版K-Means模型训练模块，以及输出模型模块，以及批量加载不同权重的新数据模块。经过特征工程处理后的数据，以及经过加权的数据，汇总提供给模型训练，以便输出一个相对精准的模型文件。详细的实现细节见4.3.1小节的离线训练模块的实现

数据预测过程，也是本系统的在线部分，主要设计目的就是能够满足线上的快速响应。主要包括特征计算，模型加载，以及模型预测。特征计算主要以快速拉取特征为准，模型加载主要以定时加载模型更新模型，模型预测则是结合特征计算以及模型加载的输出，来完成最终结果的预测。详细的实现细节见4.3.2小节在线预测模块的实现。

4.3.1 离线的模型训练

离线训练模块，需要对数据进行批量处理，主要用于处理非实时的计算任务。离线训练模块的输入是通过Mysql类中的Query函数获取的数据，以及一批通过加权模块后新样本数据。训练数据过程中，主要是涉及到特征工程类DealFeature静态类，该类主要是提供静态方法来处理特征；特征选择类SelectFeat主要是用于对特征属性的选择，是一个抽象类，需要继承类必须实现的方法是Select方法；模型训练抽象类ModelTrain，主要实现Train函数与SaveModel函数，对模型进行训练以及保存。

首先，若是有需要数据整合，则通过静态类提供MegerData方法，把不同数据来源的数据进行统一整合；接着，通过一些过滤器对数据进行过滤，比如处理缺省值的函数DealDefaultValueByFeatNam，处理重复数据的UniqueData函数，归一化数据的NormalFeature函数，以及其他可随机扩展的其他函数等等。其中，处理函数也可以通过提供额外参数来增加处理细节的控制，比如归一化，可以利用参数deal_way给业务方提供不同的归一化方式；之后，利用SelectFeat抽象类，所有不同处理特征提取的算法实施算法类都继承于SelectFeat，实现其中的Select抽象方法；之后，利用ModelTrain抽象类进行模型的训练，主要是在其集成类中实现Train函数，这样可以支持其他训练模型的实现，以方便多个做验证对比；最后，通过实现ModelTrain类的SaveModel方法将模型是保存到文件中。其中详细的训练时序图如4-7，以及相关的类结构图见图4-8。

图4-7 离线训练时序图

图4-8 离线训练模块的相关类图

4.3.2 在线的预测模型

在线预测模块主要是满足能够快速响应线上的请求，而不影响用户的体验，从而这里的实现时同步方式完成的。那么，这必须要避免耗时过大的逻辑处理，以及重复工作的处理。根据预测模块，需要两个输入，一个数据特征向量，一个是模型对象。数据特征向量，可以通过类SampleData来保存数据特征向量。但是，对于模型不是固定，这里对模型进行抽象得到了一个模型的抽象类BaseModel，该类主要是能够做到自身模型的变更，这里抽离出3个方法，分别是Initialize，ReloadModel，IsUpdate。Initialize用于实现初始化工作；ReloadModel用户实现加载模型工作；IsUpdate用于判定是否需要加载模型。获取到数据以及模型后，需要进行预测那么就需要预测类，然而不同的模型其预测的方式也不同，本文对预测类同样也进行了抽离，得到了BaseModelPredict预测抽象类。BaseModelPredict类主要包含Initialize，Predict抽象方法，Initialize完成初始化功能，Predict是完成具体的预测过程，上述的类结构详情见图4-9.

图 4-9 在线预测模块的相关类图

由于模型加载的过程是相对耗时，而且又不需要时刻加载，基本是天级的更新，从而这里将其从预测主线程中切离出去，形成子线程。本文将这部分逻辑实现通过两个线程实现，主线程用于接收用户请求，并且响应，另一个线程用于定时检查模型文件，用于更新模型。

主线程实现的时序图如图4-10。详细过程下：

首先，当用户请求过来时，通过函数GenFeatData将样本数据对象SampleData赋值；

其次，通过实施预测类KMeansModelPredictor中的Predict方法来对数据进行预测；

最后，将预测结果打包，提供给输出。

这个过程中，子线程不断的守护模型文件是否更新，若是更新则加载模型，并且修改模型，切换为最新的模型，切换在线运行模型的标记。

图 4-10 在线预测模块的时序图

4.4 本章小结

本章首先给出了系统的总体设计框图并对其进行了描述，然后分别介绍客户端和服务端的实现，客户端介绍了人机交互界面和数据库的存储格式，然后介绍了服务端三大框架，分别是数据存储、数据训练、数据预测。最后重点介绍了离线的数据训

练过程，以及在线的数据预测过程。

第五章实验结果与分析

为了验证本文算法的可行性，本文在Android手机上开发了一款基于动态手势身份识别的原型，选择Android操作系统的原因是目前市面上使用最广泛的就是Android操作系统的手机，另一个总要因素是因为其开源性，网络上有丰富的代码可供参考。本章对第三章中K-Means半随机化中心点的选择做了评估，对本文算法的有效性进行评估，同时对适应性进行了验证？

5.1 实验环境

Android操作系统的应用层开发语言为Java，所以需要Java的运行时环境，本文的Java环境为JDK1.8，对于Android集成开发工具主要有Eclipse和Android Studio，Eclipse为早期开发工具，而Android Studio是2013年Google新推出的开发环境，不仅具有Eclipse所有的功能，还有一些Eclipse所不具备的优势，因此本文选择Android Studio作为应用开发环境。同时还需要Android SDK4.0。因此本文的实验环境为JDK1.8 + Android Studio + Android + Android SDK 。服务器的软件配置采取的是Linux中的CentOS作为服务系统，C++作为程序实现的语言编辑器采用vim7.4，编译采用gcc version 4.8.2 (GCC)，连接采用Apache Web服务器。

5.2 实验数据的准备

本实验的数据来自于作者研发的智能终端APP的采集，采集的工具如图5-1所示，主要负责人机交互界面，采集传感器相关数据并进行封装，把数据传输至后台服务。其中数据封装格式采取通用JSON格式，传输协议采取TCP/IP协议。

一共采集了30个用户的数据，采集的数据类型如表5-1所示，包括用户注册的信息（用户名、密码、性别、年龄），用户手势滑动种类（单击、双击、轻浮、长按、滑动），加速度（三轴的加速度和平均值），陀螺仪（三轴的陀螺仪和平均值），速度（X轴和Y轴速度），位置（开始和结束的位置及距离）和压力（按压力度及面积）等信息。

表5-1 数据集合

数据类型字段

用户注册信息用户名、密码、性别、年龄

用户手势滑动种类单击、双击、轻浮、长按、滑动

加速度三轴的加速度和平均值

陀螺仪三轴的陀螺仪和平均值

速度 X轴和Y轴速度

位置开始和结束的位置及距离

按压力度按压力度及面积

数据类型字段

基于这些数据，进行多组实验，每个用户为一组，其他用户作为该用户的负样例，并将每个用户的数据集合分成训练数据以及测试数据，大致按照2:1的比例进行分层采样，以便验证。

5.3 试验结果评估标准

本次实验使用的验证指标，分别是FAR[45]，FRR[46]，以及EER[47]。FAR是指认假率，即“把不应该匹配上的用户当成匹配上的用户”的概率值；FRR是指拒真率，即“把应该匹配上的用户当成不能匹配上的用户”的概率值；FAR和FRR随着阈值的增大，FRR也会增大，而FAR会减小，在某一处二者的值会相等，用EER表示相等错误率，即EER为FRR和FAR相等时的概率值。具体如表5-2和公式（5-1）所示：

表5-2 混合矩阵表

实际标签

预测标签真假

真 a b

假 c d

（5-1）

其中：

a表示真正，即将正类预测为正类数；

b表示假正，即将负类预测为正类数，被视为误报；

c表示假负，即将正类预测为负类数，被视为漏报；

d表示真负，即将负类预测为负类数；

5.4 实验过程与结果分析

本文实验取某个用户的数据作为正样例，随机等比例方式获取其他用户数据作为负样例。接着，采取随机层次采样，将上述数据集合按照2:1的方式分为训练数据集合和测试数据集合。通过公式(0-2)对测试集合进行预测打分，之后将每次预测结果进行加权加入到训练的数据集合中，不断的训练和完善打分模型。下面将详细说明测试数据的测试结果。

5.4.1 K值的验证

为了选择初始的K值，本文对K值的取值进行了一次遍历，分别求出不同K值下聚类的效果，效果用公式（5-3）进行评估。（5-3）

图5-2表示了不同K值对聚类效果的影响，由图可知随着类数目K的增加，聚类效果值d先减后增，在K=2时，d有最小值6.86，其中d计算公式如式子(5-3)。这说明当前用户的行为类主要分为2种，表示用户在使用手机看文章的过程中行为比较单一并且每种行为在数据的分布上都比较集中。

为了进一步验证这个观点，本文查看了当前用户典型特征TOP2的数据分布情况，如图5-3所示。图中体现了用户的数据分布大致呈现2个类的情况，进一步验证了本文K值选择方法的正确性。

5.4.2 FAR-FRR效果图

为了获取最佳的阈值，用测试数据求出在不同阈值下的FAR和FRR曲线，具体如图5-4所示，随着阈值的增加，FAR值不断下降，FRR值不断增大；当阈值为16%时，二者基本相等，那么此时的EER为3.6%。可以得出最佳阈值为16%。由于本文的应用场景是手机用户，如果FRR太高会导致正常用户频繁的被拒绝访问自己的手机，用户体验感会大大降低，因此需要将FRR降低到最小，但是随着FRR的减小，FAR会增大，FAR增大的话就会降低手机的安全性，所以既要考虑用户体验感，又要保证安全性。因此本文将阈值设置为8%，此时的FRR为1.92%，相应的FAR为4.2%。

5.4.3 多种算法的对比

为了验证本文算法具有可行性，并且相比较于其他算法具有优势，本文用相同的一批测试数据应用在不同的算法上，并求出各个算法的FAR和FRR值，如图5-5所示：

整体来看，各种算法的效果由高到低依次是：本文算法，随机森林，逻辑回归，传统K-Measn，贝叶斯。由图可知本文算法相对比随机森林在FAR上具有24%的提升，在FRR上有70%的提升，相对传统K-Means算法在FAR，FRR上分别具有50%和71%的提升。因此无论在FAR指标还是FRR指标，本文算法的效果都是相对比较好的。

本文的算法在满足FRR值相对较低的情况下，同时也保证了FAR值也相对比较低，这样可以保证在不打扰用户正常使用的情况下，尽可能准的识别伪用户，从而在原有的身份认证的基础上能够有效防止用户信息的泄漏，以及保护手机的安全。

5.5 本章小结

本章首先对实验环境的配置和搭建做了介绍，然后对实验数据的提取和准备做了说明，接下来介绍了基于动态手势的智能终端身份识别方法的评估标准FAR和FRR，然后验证了K值的选择并给出了本文基于动态手势的智能终端身份识别方法的评估效果图，最后给出了本文算法和其他算法的比较。

6. 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第6部分　总字数：837

相似文献列表　文字复制比：3.5%(29)　疑似剽窃观点：(0)

1	城市轨道交通对沿线住宅价格的影响研究 李林芸(导师：任波) -《重庆大学硕士论文》- 2011-10-01	3.5%（29） 是否引证：否

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第六章杨倩晴-

6.1 全文工作总结

随着移动互联网的发展，智能终端成为了人们生活中不能离开的一部分，日常的购物、消费、联系和理财等都离不开手机。随之而来的安全隐患也日益凸显。在此背景下，本文提出了一种基于动态手势的智能终端身份识别方法，并实现了对应的系统。本系统在不影响用户正常使用手机的前提下能有效保护智能终端的安全性，防止冒用伪用户登录系统。本文主要的工作内容和研究成果如下：

1、本文对传统的聚类算法K-Means算法在初始中心数目和初始中心位置的确定上进行了优化。相比传统K-Means算法的完全随机选择初始中心，本文在聚类数目的选择上使用枚举的方法来确定最佳的聚类数目，在K-Means算法初始中心点位置的选择采取半随机化的方式，最后形成了本文半随机化的K-Means聚类算法。

2、本文提出了一种基于动态手势的身份识别的方法。根据聚类的结果得到用户的所有行为类分别占总行为的比重，以此比重为依据提出了行为相似度的打分公式，以此公式作为判决真伪用户的依据。同时还为新的预测样本增加了回收机制，对于判决的结果与真实情况不符的样本进行加权回收，加大模型学习遗漏的用户行为样本的学习的力度，使模型能够得到不断的完善。

3、基于动态手势的身份识别平台的系统实现。主要实现了以下几个模块：1）客户端（Android系统），主要承担用户交互界面，读取传感器相关数据及其封装，数据传输至后台服务的任务。其中数据封装格式采取通用JSON格式，传输协议采取的TCP/IP协议；2）服务器，主要负责整个身份识别的过程。主要包括数据的预处理，数据的存储，模型的训练，模型的预测

，以及结果处理等；最后对系统的准确性进行了验证，得到了1.92%的FRR和4.2%的FAR。

6.2 进一步展望

本文的基于动态手势的身份识别系统还存在一些不足，以后可以从以下几个方面进行改进：

1．本文主要分析的是用户阅读的行为习惯，以后可以考虑分析用户所有的行为习惯，增大用户的样本数

7. 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第7部分　总字数：9196

相似文献列表　文字复制比：7.1%(651)　疑似剽窃观点：(0)

1	基于云计算的分布式存储技术 唐京伟; -《中国传媒科技》- 2013-08-08	2.7%（249） 是否引证：否
2	云计算在教育信息化中的应用研究 张鹏; -《信息通信》- 2012-12-15	2.5%（232） 是否引证：否
3	Hadoop YARN资源分配与调度的研究 李媛祯(导师：杨群) -《南京航空航天大学硕士论文》- 2015-01-01	2.4%（221） 是否引证：否
4	浅析广电行业如何应用云计算 林宗达; -《中国有线电视》- 2012-06-15	2.3%（216） 是否引证：否
5	关于云计算及其建设的探讨 张建国; -《山西电子技术》- 2012-10-15	2.0%（188） 是否引证：否
6	基于Google的云计算实例分析 蔡键;王树梅; -《电脑知识与技术》- 2009-09-05	2.0%（188） 是否引证：否
7	云计算在图书馆建设与信息服务中潜在价值探析 李征; -《大学图书馆学报》- 2011-01-21	2.0%（186） 是否引证：否
8	基于云计算的数字图书馆信息服务平台 陈宫;牛秦洲; -《情报科学》- 2012-05-05	2.0%（185） 是否引证：否
9	实达云计算海量数据挖掘效率大大提升_hulala9876 -《网络（http://blog.sina.com）》- 2014	2.0%（182） 是否引证：否
10	【智慧建筑4MC之云计算Cloud Computing】_智慧运维-smartFM -《网络（http://blog.sina.com）》- 2015	1.9%（172） 是否引证：否
11	云计算在人才管理中的展望 朱建新; -《上海企业》- 2011-12-10	1.9%（172） 是否引证：否
12	浅谈云计算技术在服务业中的应用 赵红; -《信息系统工程》- 2012-10-20	1.8%（170） 是否引证：否
13	创新与安全,云计算的两只跷跷板 薛芊; -《信息安全与通信保密》- 2009-06-10	1.8%（167） 是否引证：否
14	浅析云计算 李婧; -《科技信息》- 2011-05-15	1.5%（142） 是否引证：否
15	云计算在高校教学中的应用 沈沛; -《企业家天地(下半月刊)》- 2014-06-23	1.5%（142） 是否引证：否
16	云计算在图书馆的应用初探 沈奎林;杜瑾; -《图书情报工作》- 2010-12-31	1.5%（135） 是否引证：否
17	云计算在数字图书馆应用 赵力; -《科技信息》- 2011-04-15	1.5%（135） 是否引证：否
18	云计算在中小企业的电子商务中的应用研究 姚天祥;徐运红;刘双霞; -《电脑知识与技术》- 2011-05-15	1.5%（135） 是否引证：否
19	云计算时代的数字图书馆发展探析 韦妙; -《中国教育技术装备》- 2011-10-15	1.5%（135） 是否引证：否
20	云计算与江西昌北高校图书馆联盟 李雪萍;吴青林; -《江西图书馆学刊》- 2012-11-30	1.5%（135） 是否引证：否
21	浅析云计算在故障诊断中的应用 王春键;卢文忠;刘丙杰; -《计算机与网络》- 2010-05-26	1.5%（135） 是否引证：否
22	公安云计算平台的建设探讨 吕益民; -《中国人民公安大学学报(自然科学版)》- 2013-02-15	1.5%（135） 是否引证：否
23	基于云计算技术的高校招生管理系统研究 石允剑;袁家斌; -《中国教育信息化》- 2013-02-10	1.5%（135） 是否引证：否

24	对国内IT行业在“低碳”面前尴尬处境的思考	1.5%（135）
	汤慧; -《警官文苑》- 2010-09-15	是否引证：否
25	大数据与云计算	1.5%（135）
	李永宏; -《统计与管理》- 2013-12-20	是否引证：否
26	云计算环境下的信息安全问题研究	1.4%（128）
	刘宏;邱平; -《中国信息界》- 2014-01-20	是否引证：否
27	一种Hadoop Yarn的资源调度方法研究	0.5%（50）
	李媛祯;杨群;赖尚琦;李博涵; -《电子学报》- 2016-05-15	是否引证：否
28	基于Hadoop一种移动云计算本地化调度算法的研究	0.4%（35）
	刘瑞祥;汤艳; -《计算机应用与软件》- 2015-07-15	是否引证：否
29	基于模式匹配和协议分析的入侵检测系统的军事应用研究	0.4%（33）
	曾宪金(导师：桂卫华) -《中南大学硕士论文》- 2006-03-01	是否引证：否
30	基于多维评价模型及改进蚁群优化算法的云计算资源调度策略	0.4%（33）
	张乐乾 -《学术论文联合比对库》- 2015-04-01	是否引证：否
31	无线传感器网络覆盖优化与目标定位技术研究	0.3%（31）
	黄月(导师：吴成东) -《东北大学博士论文》- 2013-07-01	是否引证：否

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第七章田野-绪论

以下三段是我的摘要部分，论文检测使用。

云计算技术的趋于成熟为企业提供了一种可行的、廉价的处理大数据的解决方案。Hadoop 是Apache基金会下一个开源的分布式存储和并行计算平台，由于具有高可靠性、容易扩展和高容错性等优点，目前在大数据处理领域得到了广泛的应用。在云计算应用方面，作业调度和资源分配问题一直都是一个需要重点关注和解决的问题。在云计算平台上同一时刻经常有多个作业需要调度运行，每个作业又被分为若干个子任务单独运行，因此如何协调各个任务的资源分配和调度问题至关重要。Hadoop的资源调度管理框框架YARN提供了三种内置的资源调度器。但是随着应用的延伸，现有的资源调度算法已经不能满足需求，在许多情况下制约了Hadoop系统性能的提升。因此，对合理地进行资源分配与作业调度的研究，能够提高系统的资源利用率，从而减少作业执行时间，最终使得平台的整体性能得到充分提升。

资源调度问题是NP-hard问题。NP-Hard问题不能在多项式时间内求解。群智能算法在求解此类问题上有着不俗的表现，如遗传算法，蚁群算法，布谷鸟算法等。针对布谷鸟算法在后期收敛性较弱、求解精度不高的不足，本文提出一种混合遗传布谷鸟优化算法，在布谷鸟算法的基础上，引入遗传算法。保留布谷鸟算法的强全局搜索能力的同时，结合了遗传算法局部收敛性好的特点，加快了算法在后期的收敛速度。同时，本论文对Hadoop YARN的资源管理和分配机制进行了深入分析，对Hadoop资源和作业调度过程进行建模，将混合遗传布谷鸟算法应用于YARN资源调度。该算法可以通过YARN的管理机制获取节点CPU速率、内存容量、负载等信息，结合任务的资源申请需求，将各个任务分配到适合的资源节点。同时在调度的过程中，加入了对任务优先级的调整，对资源量需求较大和即将完成的任务，提高其调度优先级，避免资源量需求较大的任务长时间得不到调度以及即将完成的任务陷入长时间等待的困局。

研究和测试结果表明，本文提出的混合遗传布谷鸟算法相比布谷鸟算法，在标准函数的最优值求解过程中表现优异。同时，将该算法应用到Hadoop资源调度时，能有效地提高系统资源利用率，缩短集群的作业执行时间。

本课题首先对云计算和Hadoop相关技术进行了研究，其中对作业调度算法进行了深入研究，对其作业调度算法进行了引入群智能理论的改进创新，设计与实现了新的作业调度算法，并对改进后的作业调度算法进行实践验证。本章首先阐述本课题的选题背景和意义，明确工作定位与内容。然后说明本课题的创新点。最后将阐述本论文的组织结构。

7.1 课题背景与意义

近年来，随着互联网技术与应用的飞速兴起与发展，网络服务已经成为普通居民生活中不可或缺的一部分，不仅参与网络活动的用户数量在持续增长，单个用户的网络活动内容也在持续增多。随着用户网络活动的增多，由此产生的数据也在持续的爆炸性的增长。在使用各种面对如此海量数据的冲击，传统的数据存储处理系统已经显得有些力不从心，单台计算机的性能增长已经落后于用户和企业对互联网传输、处理及存储海量数据的能力的需求。而随着硬件设备与软件技术的不断发展，构建能提供可靠服务的分布式集群的技术也逐渐成熟，“云计算”的概念也就在此背景下传播开来。现在云计算已经广泛应用于电子商务、金融、天文计算等各个领域。

云计算（Cloud Computing）是分布式计算（Distributed Computing）、并行计算（Parallel Computing）、网络存储（Network Storage Technologies）、虚拟化（Virtualization）、负载均衡（Load Balance）等传统计算机和网络技术发展融合的产物。云计算是通过使计算分布在大量的分布式计算机上，而非本地计算机或远程服务器中，云计算支持用户在任意位置、使用各种终端获取应用服务。所请求的资源来自“云”，而不是固定的有形的实体。应用在“云”中某处运行，但实际上用户无需了解、也不用担心应用运行的具体位置。只需要一台笔记本或者一个手机，就可以通过网络服务来实现我们需要的一切，甚至包括超级计算这样的任务。

在云计算的领域中，国外的公司一直都是先行者和开拓者。谷歌、微软和亚马逊等大型互联网公司在早期就开始探索并布局云计算方面的应用。云计算服务的可靠性需要众多技术的保证，如虚拟化、并行计算、分布式存储等。谷歌作为行业的巨头

，在领域内具有极其强大的技术能力。2004年前后，Google发布了三篇论文，分别介绍了GFS、MapReduce和BigTable技术，在当时的互联网领域引起了极大地反响。在随后的时间中，谷歌在云计算领域又进行了深入的研究和原型的开发并对云计算进行大力推广，在其他公司和开发者的响应下，云计算得以快速地发展和应用，Hadoop就是在这一背景下诞生并且得到了极大的关注。

Hadoop是一个开源的分布式数据处理平台，它包含了云计算平台的各个设计要点。它的核心是由两部分构成：Hadoop分布式文件系统HDFS（Hadoop Distributed File System）以及运行于HDFS之上的分布式并行计算框架MapReduce。其中，HDFS是谷歌论文中介绍的GFS技术的实现，它提供了具有高可靠性、高扩展性、高容错性的文件系统。Hadoop的MapReduce框架则是谷歌MapReduce论文的开源实现，运行于HDFS之上，提供了多台计算机并行合作计算处理的能力，并且针对HDFS的特点进行了适配和调整。这两个核心部分相辅相成，使得Hadoop不仅能够提供对大数据的优秀处理能力并且能长时间可靠地运作。

Hadoop的出现使得普通的开发人员编写开发并运行分布式处理程序变得更加简单便捷。Hadoop平台屏蔽了分布式程序运行需要关注的细节，使开发人员得以专注于业务代码的编写和调试。同时，平台也统一了MapReduce编程接口，规范了开发流程，使不同的开发人员能够在同一个平台上编写运行不同的任务。Hadoop的出现大大节省了企业的设备购置成本，普通的计算机资源组成的集群就能够替代超级服务器，同时还具有更好的可靠性、高扩展性及高容灾能力。其统一的编程接口也使得代码可以很好地复用，大大降低了开发成本。Hadoop的开源也使得中小型企业和个人开发者可以通过闲散机器快速搭建集群并拥有较强的大数据处理能力，从而为大数据处理的研究做出更大的贡献。总之，Hadoop各方面的优点使其快速的流行开来，并且有许多的开发者志愿维护社区并优化Hadoop性能。

由于Hadoop的强大的大数据处理能力及其他诸多优点，基于Hadoop的应用越来越多。尤其是互联网领域的许多公司，都积极搭建Hadoop集群来运行处理大数据任务。如Facebook借助集群来支持其用户的爱好分析和机器学习相关任务；百度的Hadoop集群用于进行用户搜索记录分析和数据挖掘方向的工作。淘宝则使用集群处理电子商务的订单相关数据，并对用户进行画像分析。许多高校的实验室也通过搭建自己的小规模集群来处理实验数据等工作，在缩短数据处理时间和提高工作效率方面有着良好的效果。

随着Hadoop集群应用和规模的不断扩展，其在资源管理和作业调度上的问题也逐渐暴露出来，成为影响集群性能提升的主要瓶颈之一，对本问题的改进能使集群在不扩展规模和提升单机性能的基础上有比较明显的性能提升。在Hadoop的MRv1及以前的版本中，其资源调度和作业控制功能是由集群的主节点负责的。由于资源调度的复杂性，在此种情况下，随着待调度作业的增多、集群的扩大，主节点的负载越来越重，其调度所需时间也越来越多，调度性能出现不可避免的下降。在Hadoop进行重构后的的MRv2（即YARN）架构中，资源管理和作业控制功能进行了分离。资源调度功能由主节点的ResourceManager负责，而作业的控制交由各个应用程序对应的控制器ApplicationMaster（AM）进行管理。具体来说AM主要负责向资源调度申请适量的资源容器，在容器中运行任务，跟踪任务的运行状态并对进程进行监控，对失败的任务进行处理和反馈。YARN中对MRv1主节点两种职责的分离大大减轻了主节点的运行负载，提高了Hadoop集群的工作性能。

目前，Hadoop的作业调度器作为核心组件之一，只提供了三种常用的资源分配算法：FIFO算法，Capacity调度算法，Fair调度算法。随着集群应用范围的不断扩大，系统内置的几种资源分配算法并不能满足广大用户和作业的差异化分配需求。FIFO调度算法虽然有简单易行，资源调度开销较小的优点，但是对于调度的公平性上有所不足，同时也比较难以兼顾数据本地性，适用情形较为狭隘。

Capacity调度算法在使用方面较为复杂，需要用户掌握大量的系统信息，准确估算作业的实际运行情况并针对实际设置参数；Fair调度算法实现复杂，过于强调任务数量的负载均衡，对于集群的异构环境考虑不足，同时也忽略了集群各节点的实时负载能力，因此在作业调度方面表现不佳。综上所述，针对Hadoop资源管理和作业调度的研究能使Hadoop集群性能得到很好地提升。目前已有许多研究人员致力于研究如何提高Hadoop资源管理及作业调度效率。

7.2 课题目标与任务

所谓作业调度就是将系统资源分配给相应的作业满足其运行需求的过程，也可以被称为资源调度。在Hadoop集群中通常会有大量的作业同时提交运行，每个作业又被分为若干个Map任务和Reduce子任务，并且作业类型和资源需求也不尽相同，在此种情况下，设计一个好的资源管理和作业调度器，对于提高集群的资源利用率，降低作业的平均完成时间，保证集群的高吞吐量、负载均衡，满足用户的使用需求有着至关重要的作用。Hadoop平台内置了三种资源调度器：分别为FIFO调度器、Capacity调度器、Fair调度器。目前Hadoop 调度架构YARN的默认调度器为Capacity调度器。 Hadoop的调度器采用了可插拔的设计，如果内置的三种调度器表现不能满足用户对于调度性能的需求，用户可以根据集群环境和实际应用的需求设计符合要求的作业调度器。

本课题的目标是研究并实现基于群智能算法进行资源调度的算法。本文针对群智能算法的不足之处，提出一种改进的群智能算法，并将通用型的改进智能算法进行调整应用于Hadoop作业的资源调度，并针对Hadoop资源管理的特定模型进行优化和调整，使算法能够用于Hadoop资源调度，并且相比于平台默认的算法在性能上有所提升。

7.3 课题创新点

本论文主要研究了Hadoop YARN框架的资源管理和作业调度的特点。针对资源调度的不足之处，本论文的工作概述为如下两方面：

一、NP-Hard问题是指不能再多项式时间内可解决的问题，此类问题随着问题规模的增长，求解问题所消耗的时间通常是指

数形式地增加。资源调度已被证明是NP-Hard问题。群智能算法是一种通过借鉴或模拟自然界生物生存或繁衍的规律，在问题的解空间内按照此类经验来寻找最优解，如遗传算法，蚁群算法，布谷鸟算法等。群智能算法已经在实验和实际生产实践中证明是一类可以在限定时间内找到问题的最优解或近似最优解的算法。本论文针对布谷鸟算法在后期收敛性较弱、求解精度不高的不足，本文提出一种混合遗传布谷鸟优化算法，在布谷鸟算法的基础上，引入遗传算法。保留布谷鸟算法的强全局搜索能力的同时，结合了遗传算法局部收敛性好的特点，加快了算法在后期的收敛速度。标准测试函数表明，本文提出的混合遗传布谷鸟算法相比布谷鸟算法，加快了后期的收敛速度，求解精度得到提高，算法的整体性能得到提升。

二、资源调度问题是一个NP-Hard问题，采用传统的调度算法并不能很好的对系统资源进行分配。由于智能算法是一种有效地求解最优化问题的手段，本论文通过对Hadoop YARN的资源管理和分配机制进行深入分析，将混合遗传布谷鸟算法应用于YARN资源调度。该算法首先通过YARN的管理机制获取节点CPU速率、内存容量、负载等信息，其次对任务的资源申请需求进行统计，按照算法的特点对节点资源和任务的资源需求进行适当的编码，进一步根据算法选择适当的节点来分配任务。同时在调度的过程中，加入了对任务优先级的调整，对资源量需求较大和即将完成的任务，提高其调度优先级，避免资源量需求较大的任务长时间得不到调度以及即将完成的任务陷入长时间等待的困局。在论文的最后进行了实验验证。实验结果表明，采用本文的算法能有效地提高系统资源利用率，缩短集群的作业执行时间。

7.4 论文结构

本文主要阐述了基于布谷鸟算法的改进型混合遗传布谷鸟算法的研究与实现，并将该算法应用于Hadoop资源调度，结合系统资源分配的限制和需求对算法进行调整。文章的整体架构如下：

第一章是论文的绪论部分，主要概述了本课题的研究目标及意义，明确了相关的研究工作和内容。并阐述了本课题的创新点，最后对论文组织结构进行介绍。

第二章的主要内容是Hadoop平台发展和架构及其核心技术的介绍。首先介绍了Hadoop平台及其发展背景，并重点介绍了其核心技术和重要概念。在本章的最后阐述了当前Hadoop平台内置的三种默认的调度算法，并对现有资源调度算法的不足进行了分析。

第三章研究对群智能算法的改进。NP-Hard问题是不能在多项式时间内可以解决的问题，随着问题规模的增长，解决时间会以指数级别的速度增加。资源调度问题就是一类常见的NP-Hard问题。群智能算法是一种能在有限时间内求解此类问题近似最优解的方法。布谷鸟算法是一种较为新型的群智能算法。本章针对布谷鸟算法在后期收敛速度偏慢，局部寻优精度不高的缺点，提出了一种混合遗传布谷鸟算法，在保留布谷鸟算法全局搜索性较好特点的基础上，融合了遗传算法局部寻优精度较高的特点，并在最后对算法进行测试。

第四章研究如何将改进后的群智能算法应用到Hadoop YARN资源调度管理。本章首先说明了YARN对资源的管理方式，对YARN的资源管理方式进行建模分析，将节点的计算资源和任务的资源需求申请进行编码。针对Hadoop资源管理的实际应用，对智能算法进行调整，使得每次迭代之后的解符合资源调度的规则。同时，在调度的过程中加入了调度优先级的概念，避免出现需求资源较大的任务长时间得不到调度以及作业在即将完成时长时间停滞的现象。

第五章搭建测试环境进行测试。通过实际作业的运行验证本文提出的调度算法的有效性。

第六章进行了总结和展望。对本课题的主要工作内容进行总结，在现有的研究成果基础上提出了研究的不足之处以及对进一步的研究进行展望。

法

The maturity of cloud computing technology provides enterprises with a viable and affordable solution to big data. Hadoop is Apache Foundation's next open source distributed storage and parallel computing platform, due to its high reliability, easy expansion and high fault tolerance, etc., is now widely used in the field of big data processing. In cloud computing applications, job scheduling and resource allocation has always been a problem that needs to be focused and solved. In the cloud computing platform, there are often multiple jobs that need to be scheduled and run at the same time. Each job is divided into several sub-tasks and run separately. Therefore, how to coordinate the resource allocation and scheduling problem of each task is very important. Hadoop's Resource Scheduling Management Framework YARN provides three built-in resource schedulers. However, with the extension of the application, the existing resource scheduling algorithm can no longer meet the demand, which in many cases restricts the performance of the Hadoop system. Therefore, the research of resource allocation and job scheduling reasonably can improve the system resource utilization, reduce the job execution time, and ultimately improve the overall performance of the platform.

Resource scheduling problem is NP-hard problem. NP-Hard problems can not be solved in polynomial time. Swarm intelligence algorithm has good performance in solving such problems, such as genetic algorithm, ant colony algorithm, cuckoo algorithm and so on. Aiming at the shortcomings of the cuckoo algorithm such as low convergence rate and low solution accuracy in the later period, a hybrid genetic cuckoo optimization algorithm is proposed in this paper. Based on the cuckoo algorithm, a genetic algorithm is introduced. While preserving the strong global search ability of cuckoo algorithm, combining with the good local convergence of genetic algorithm, the convergence speed of the algorithm is accelerated. At the same time, this thesis deeply analyzes the resource management and allocation mechanism of Hadoop YARN, and models the Hadoop resource and job scheduling process. The hybrid genetic cuckoo algorithm is applied to YARN resource

scheduling. The algorithm can get the node CPU speed, memory capacity, load and other information through the YARN management mechanism, and assign each task to the appropriate resource node according to the resource request of the task. At the same time in the scheduling process, joined the priority adjustment of the task, the larger demand for resources and the upcoming tasks to be completed, to improve its scheduling priority, to avoid resource-demanding tasks for a long time without scheduling and imminent completion Task into a long wait for the dilemma.

The research and test results show that the hybrid genetic cuckoo algorithm presented in this paper is superior to the cuckoo algorithm in solving the optimal value of the standard function. At the same time, when applied to Hadoop resource scheduling, this algorithm can effectively improve system resource utilization and shorten the execution time of the cluster.

指　标
疑似剽窃观点
1. 实验结果表明，采用本文的算法能有效地提高系统资源利用率，缩短集群的作业执行时间。
疑似剽窃文字表述
1. 计算支持用户在任意位置、使用各种终端获取应用服务。所请求的资源来自“云”，而不是固定的有形的实体。应用在“云”中某处运行，但实际上用户无需了解、也不用担心应用运行的具体位置。只需要一台笔记本或者一个手机，就可以通过网络服务来实现我们需要的一切，甚至包括超级计算这样的任务。 在云计算的领域中， 2. FIFO算法，Capacity调度算法，Fair调度算法。随着集群应用范围的不断扩大，系统内置的几种资源分配算法并不能满足 3. 资源调度问题是一个NP-Hard问题，采用传统的调度算法并不能很好的对系统资源进行分配。由于智能算法

8. 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第8部分　总字数：9421

相似文献列表　文字复制比：22.2%(2090)　疑似剽窃观点：(0)

1	03700_1120379028_赵颖 赵颖 - 《学术论文联合比对库》- 2014-12-28	8.4%（796） 是否引证：否
2	Hadoop环境下的动态资源管理研究与实现 赵颖(导师：吴刚) - 《上海交通大学硕士论文》- 2015-01-14	6.9%（650） 是否引证：否
3	BH20109263109_李孟庭 李孟庭 - 《学术论文联合比对库》- 2013-01-05	5.7%（541） 是否引证：否
4	李盼_10Z0700_Hadoop关键技术及其改进策略研究 李盼 - 《学术论文联合比对库》- 2013-01-15	5.5%（518） 是否引证：否
5	HADOOP调度算法及其改进策略研究 李盼(导师：余重秀) - 《北京邮电大学硕士论文》- 2013-01-14	4.9%（457） 是否引证：否
6	异构环境下MapReduce离线调度算法的研究 陈珩(导师：过敏意) - 《上海交通大学硕士论文》- 2014-01-01	4.3%（403） 是否引证：否
7	BH20099257125_卡哇伊 卡哇伊 - 《学术论文联合比对库》- 2012-12-28	4.0%（377） 是否引证：否
8	马鹏将 马鹏将 - 《学术论文联合比对库》- 2015-01-04	1.3%（123） 是否引证：否
9	基于Hadoop的文本分类研究 刘丛山(导师：杨煜普) - 《上海交通大学硕士论文》- 2012-02-01	0.7%（69） 是否引证：否
10	通信网云计算平台资源调度策略与算法研究 汪洋(导师：陈海) - 《南昌大学硕士论文》- 2013-05-28	0.7%（69） 是否引证：否
11	基于HDFS的电子文件集中存储和检索系统 张宸(导师：谢立;茅兵) - 《南京大学硕士论文》- 2012-05-01	0.7%（67） 是否引证：否
12	20121236535-王德龙-Hadoop平台下作业调度算法的研究与改进 王德龙 - 《学术论文联合比对库》- 2015-04-07	0.6%（60） 是否引证：否
13	主资源公平调度算法在YARN中应用研究 陈昕;刘朝晖; - 《电脑知识与技术》- 2016-04-12 1	0.6%（56） 是否引证：否
14	基于Hadoop云平台的数据挖掘技术在天气数据的应用研究 杨利娟(导师：张永军) - 《北京邮电大学硕士论文》- 2015-01-04	0.4%（40） 是否引证：否
15	基于hadoop的地震数据分布式存储策略的研究	0.4%（38）

	冯翔(导师：文必龙) -《东北石油大学硕士论文》- 2014-06-08	是否引证：否
16	基于Hadoop的海量数据管理系统	0.4%（34）
	多雪松;张晶;高强; -《微计算机信息》- 2010-05-05	是否引证：否
17	基于HADOOP的数据挖掘研究	0.4%（34）
	杨宸铸(导师：向宏) -《重庆大学硕士论文》- 2010-11-01	是否引证：否
18	云计算背景下基于FPGA的文件管理系统与Web缓存的紧耦合研究与分析	0.3%（32）
	孔雪(导师：祝永新) -《上海交通大学硕士论文》- 2011-12-28	是否引证：否

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第八章 Hadoop相关知识

8.1 Hadoop平台概述

Hadoop是来源于Apache Lucene的一个开源的网络搜索引擎Nutch子项目。Nutch项目起始于2002。在2004年前后，Google陆续发表了关于分布式计算框架MapReduce和分布式文件系统GFS的论文。Doug Cutting等人受到论文描述方法的启发，尝试实现MapReduce框架并与NDFS（Nutch Distributed File System）相结合，用于对Nutch引擎的算法支持。由于结合后的系统在Nutch系统中表现优秀，在2006年，NDFS和MapReduce系统从Nutch中分离出来并被完善成一个独立且完整的系统，Hadoop便这样诞生了。同时Yahoo!决定开始采用Hadoop进行分布式计算，此项举动也大大推进了Hadoop的推广。到2008年初，Hadoop已经被Google、Yahoo!等众多互联网公司所应用和改进。同年4月，Hadoop打破TB级数据排序的世界纪录，有力地证明了Hadoop系统集群强劲的工作性能。虽然目前Spark等框架如火如荼，Hadoop作为一个入门的、典型的分布式计算系统，仍然有其旺盛的生命力，对其研究及改进仍然有着重要的意义。

到目前为止，Hadoop的架构发展经历了两个主要的版本，即MRv1（MapReduce version 1），Hadoop 1.x与Hadoop 0.x的架构版本和MRv2（MapReduce version 2），Hadoop 2.x的架构版本。目前Hadoop2.x是现在较为流行的版本，其引入了集中资源管理系统YARN，将资源管理与计算框架分离开来，使得Hadoop可以轻松与其他分布式计算框架相结合，构建功能更加强大的分布式计算系统，使得Hadoop的生命力得到进一步增强。本论文主要针对MRv2中的资源管理和作业调度过程进行分析和改进，提出自己的优化思路。

在本章接下来的小节中，将会详细介绍Hadoop的相关核心技术。

8.2 Hadoop分布式计算框架MapReduce

MapReduce是Google与2003年公开发表的一个用于大规模数据集并行计算的编程框架，其在Google内部已经被广泛的使用。MapReduce提出的初衷就是为了解决海量网络数据和检索日志的难题。Google作为世界上使用量最大的引擎，积累的庞大无比的数据，而且每天的数据量还在不断的增加中。面对如此海量数据处理的压力，即使是简单的PageRank计算或者建立数据倒排索引都变得异常困难。因此，面对超出单机计算能力的数据处理需求，并行处理计算成了必然的选择。MapRedece计算框架即是为解决这一问题而提出的。MapReduce是基于一种分治的思想，将数据处理分散到各个计算节点。该计算模型封装了简单的数据处理接口，屏蔽了处理分布式计算所需要注意的细节，用户只需要实现两个简单的函数Map和Reduce，即可高效的实现分布式计算。用户无需关心框架是如何处理数据在各个节点上的存储和在节点间的复制或移动，如何对单个任务或计算节点的异常进行处理，极大地节省了开发人员的精力。

MapReduce计算模型的处理对象均是以<key,value>形式存储的键值对。如果输入数据不是满足<key,value>形式的键值对，模型会要求输入数据规范为<key,value>形式：如{Dear，Bear，River}将按照字数统计的需求，格式化为{<Dear,1>，<Bear,1>，<River,1>}。规范化的好处是框架可以方便的处理各类数据，而不会因为格式问题陷入僵局。MapReduce框架只需用户实现两个自定义函数map和reduce，分别对数据进行两个规约，即可以并行的方式处理输入数据集。通常，map函数负责处理输入的<key,value>键值对，生成一系列以<key,value>形式存储的中间结果。再经过shuffling过程，将相同key值的键值对分配到一个计算节点，在此过程中，MapReduce框架不对程序运行中间结果做更改。最后由reduce程序对shuffling的结果进行处理并产生输出文件。在以上过程中，MapReduce框架会自动将输入文件划分为M份小的输入文件存入HDFS文件系统，同时产生M个map任务待调度执行。在资源充足的集群中，map任务可以在M个计算节点上并行运行，生成的中间结果按照key值分配到R个计算节点上，那么reduce程序也可以在R个计算节点上并行运行。

图2.3即展示了MapReduce框架的工作流程。以下为MapReduce框架工作的详细流程描述：（1）首先用户需要实现数据处理所需要的map函数和reduce函数，之后连同数据集一起提交到集群。（2）当用户提交任务并启动之后，MapReduce框架会将输入数据集按照一定的大小划分为M个小文件，用户可以按照需求自行配置划分文件的大小。（3）在MapReduce集群系统中，节点机器会被划分为两类不同职责的机器，一类是控制节点，在一般情况下由一台机器来执行控制节点的职能；其余的所有机器被划为另一类，即工作节点。控制节点的主要任务是将map和Reduce任务分配给工作节点。当其发现有资源闲置并且有待调度的map或者reduce任务存在时，控制节点会给待调度任务分配运算资源以运行任务。这样，同一个输入作业的不同的map任务便在多台机器上并行运行（4）工作节点接受到调度执行的map任务之后，读取本地HDFS存储的相对应的输入文件，此时，如果输入文件没有在本地，则需要从其他存储节点进行网络传输拷贝。在此种情况下，集群的网络将会有临时的高负载，如果有比较多的经过调度待执行的map任务存在数据本地性无法得到满足的情况，则集群的网络通信受到比较严重的影响，继而影响数据的传输，最终对任务的执行产生较大的影响。在工作节点读取完输入数据之后，调用map函数对输入的数据键值对进行处理，产生的中间结果先暂时在内存中缓存，其结果也是键值对的形式。（5）当map任务运行完毕时，缓存在内存

中的中间结果将会写入到磁盘，在写入的过程中，MapReduce框架会按照一定的规则将key值进行分类，通常是使用Hash的方法，将中间结果分到R个结果集合，分别将其组成文件写入HDFS系统，同时MapReduce框架也会将Hash处理后的结果集的位置通知调度系统，以便于在新的一轮资源分配和作业调度的过程中充分考虑reduce任务的本地性，将reduce任务调度到合适的工作节点进行运行。（6）当任务调度器将一个reduce任务调度到合适的节点之后，类似于map任务调度，系统也会知悉待处理的数据是否在本节点存在，需要从何处拷贝等消息。由于每份reduce待处理的中间结果集通常含有多个key值，因此，当节点对数据读入内存之后，还需要将中间结果数据集进行排序，以方便reduce程序处理和之后多个reduce输出结果的合并。此次中间结果的排序过程连同第（5）步中对中间结果的Hash处理的步骤一起的过程，叫做shuffle过程。在此步骤中的排序过程，并不一定需要在reduce任务调度完成之后在进行，通常可以和第（5）步中的Hash过程合并执行。（7）调度执行的reduce任务会按照用户的reduce函数对排序后的中间结果集进行处理，对每个key值对应的集合进行合并，并将最终结果输出到文件系统中。（8）当所有的map任务和reduce任务执行完毕，控制节点会收到一份通知，对任务的执行过程进行统计与总结，将生成的结果和输出文件推送给用户。通常来讲，输出结果会是R份文件，这R份文件可以简单合并，也可以作为下一个MapReduce任务的输入直接写入文件系统。

8.3 Hadoop的分布式文件系统HDFS

Hadoop的数据存储系统是HDFS（Hadoop Distribute File System），它也是Hadoop云计算平台的一项核心技术。HDFS能提供高吞吐量的数据存储访问服务，作为大数据处理集上的分布式存储系统来说非常有优势。HDFS的高吞吐率是通过“一次写入多次读取”的流式读取文件数据的模式来实现的，在此种模式下，系统中的文件不能被修改，以此为代价换取较高的数据吞吐率。

8.3.1 HDFS的工作机制

HDFS被设计成一种Master/Slaver结构（即主从结构），一个典型的系统包括一个名字节点（NameNode）和N个数据节点（DataNode），如图2.4所示，NameNode中存储有包含完整的文件系统命名空间和文件块映射表的元数据，是文件系统的控制和监控中心；文件数据则按照一定的规则存在诸多DataNode上，同时DataNode提供了文件的操作管理服务。

由于绝大部分的MapReduce程序对数据集的操作符合“一次写多次读”的流操作模式，待处理文件包括Map任务产生的中间结果集文件都符合这一特点。因此HDFS在设计之初就考虑了MapReduce程序的这一特性，简化了分布式文件存储系统的数据一致性的问题，将数据文件以数据块（Block）而不是以连续文件的形式存储在系统中。在用户上传文件的过程中，HDFS管理系统就对文件切分成固定大小的数据块进行存储，默认的配置的数据块大小为64MB，每个数据块的副本总数为3，这些数据块会被分散到各个存储节点来进行冗余存储，这样不仅能使数据块得以方便地并行访问，数据本地性得到满足的可能性也大大的提高了，同时也减小了因为节点故障而导致数据丢失的风险，为MapReduce程序的运行提供了良好的基础。

Hadoop的文件目录管理结构是传统的层次型组织结构，文件系统命名空间与现在常见的大多数文件系统相像，符合用户的使用习惯，用户可以较为方便的移动、创建和删除文件。不过Hadoop不支持对用户访问文件权限的控制，也不支持多用户对于磁盘空间使用的限制。正如上文所述NameNode是HDFS的控制中心，也负责管理和维护HDFS系统的命名空间，并且管理数据块（block）与文件和DataNode的映射。这些信息都以文件的形式保存在本地磁盘上，同时，任何对文件、文件属性或命名空间的修改都以日志的方式记录在本地磁盘。Editlog事务日志即是用于记录HDFS系统对文件操作记录的，FsImage文件则存有整个文件系统的命名空间，以及数据块与文件的映射、文件的属性等。当客户端发起任何此类操作，NameNode会通过记录的映射关系确定操作的数据块或文件的位置，进而根据用户的需求，进行相应的操作并进行日志的记录。任何文件或数据块均不会在NamNode上保存，仅有文件的组织结构和数据块在DataNode上的存储信息。该文件系统会定期或在启动时将事务日志中记录的文件操作与元数据进行合并，生成新的FsImage保存到本地文件系统并对旧的日志进行清除，这个过程称为一个检查点（checkpoint）。

8.3.2 HDFS的可靠性分析

Hadoop分布式文件系统（HDFS）可以在大量廉价的通用硬件所组成的集群上可靠运行是最初的设计目标之一。鉴于通用硬件相比专业设备的故障发生率较高，而且在计算机领域故障的发生也被认为是常态而不是异常，在HDFS设计与实现时就对故障的发生原因和可能导致的结果做了充分的考虑和准备，在系统中设计和实现了较为完善的错误处理机制来保证系统能提供可靠的存储服务。

（1）数据块冗余机制

副本的存放是HDFS稿可靠性和高读取性能的关键。在HDFS中，文件不是以整体存放于数据节点，而是被分割成大小相等的数据块存储于不同的数据节点。如果存放数据块的节点由于故障导致无法链接到集群，则会使得有数据块存放于本节点的文件不可访问。为了避免出现此种情况，HDFS将每个数据块按照系统设置生成一定数量的副本，分别存放于不同的数据节点，数据块副本的映射和位置信息存于NameNode。当数据节点出现故障时，HDFS会根据文件的数据块对应的副本数量信息找到少于设置副本的数据块，并自动进行复制备份，分散其到其他运行良好的数据节点中。

大型的HDFS集群往往布置在多个机架上，甚至有异地的数据中心，在同一机架上的数据节点的通讯效率远远高于位于不同机架的节点。然而由于不同机架同时发生故障的概率更为微小，为了避免在单个机架出现断电等故障导致服务无法继续运行，将数据块的副本备份到不同的机架是一个很好的提高可靠性的策略。同时在机架间的带宽闲置时，分散任务到相邻机架也能很好的提高系统性能。通过平衡机架间的数据传输带宽的占用与HDFS的容灾能力，在默认副本数为3的情况下，三分之二的副本保存在本地与同机架的机器上，三分之一放置在相邻的机架上，这样既兼顾了数据的可靠性也不会因为副本的距离产生较

多的数据通讯成本。在更多副本的情况下，可以参考机架间的带宽占用率与系统设计的容灾能力进行权衡设计副本保存的策略。

（2）心跳通信机制

数据冗余机制保证了系统存在故障节点时的可靠性，同时系统发现故障节点之后能够快速恢复数据块的冗余程度。而系统对故障节点的发现依赖于心跳通信机制。在HDFS中，NameNode与DataNode通过心跳机制来进行信息交流。每隔固定的时间，DataNode将会发送一个心跳数据包，汇报本节点当前的状态。NameNode也会定期查看处理各DataNode的心跳包，如果NameNode发现心跳信号缺失，并在规定时间内没有再与之建立连接，则认为该DataNode与网络断开连接，将其标记为宕机状态，不会再将新的IO请求发送给该节点，同时通过元数据来查找缺失的数据块，将其进行备份复制并分散到其余活跃节点中，保证数据的冗余程度满足要求。

（3）数据完整性检测机制

由于硬件的不可靠性，存储于节点的数据块可能发生异常，或者系统从某个DataNode获取的数据在网络传输的过程中发生突变。为了解决数据异常的问题，HDFS采用文件校验的方式，对HDFS文件内容的校验和（checksum）进行检查。当用户提交文件到系统时，系统会对切割后的数据块进行校验计算校验和，并将校验后的结果隐藏存储于对应文件的同一级的命名空间中。当用户或任务发起对数据的需求后，系统从中取出数据块并对其进行校验，将校验结果与预存的校验和进行比对，比对无误则说明数据块无损坏，如果比对失败，则系统将会向其他节点发起副本传输的请求。

（4）元数据冗余机制

在HDFS中，数据可靠性的保证十分依赖于元数据存储文件FsImage和事务日志Editlog这两个核心文件。如果这两个文件发生损坏，则整个集群将无法工作。为保证两个核心文件的完整性，NameNode支持将此核心配置文件设置多个副本，在设置完成后，任何针对这两个文件的修改将同步修改所有副本，副本间通过强一致性来保持同步关系。由于对数据一致性的要求较高，不可避免的会降低NameNode的性能，然而在测试中表明，这一策略对NameNode事务处理速度的降低并不明显，相比于系统可靠性的提升，这个代价是可以接受的。

8.4 Hadoop的资源管理方式

在本章的第一小节介绍了Hadoop的发展以及Hadoop资源管理架构的变迁。本小节将详细介绍Hadoop资源管理及作业调度的方式。

在Hadoop MRv1资源管理架构中，JobTracker是集群资源与作业的控制中心，其职能相当于集群的管理者，负责集群的资源管理和作业控制功能，其中作业控制功能包括作业调度、作业监控、对失败作业的容错等。当用户程序（JobClient）提交了一个作业之后，其作业的信息会发送至JobTracker进行统计分析。TaskTracker是MRv1中的另一个重要的组件，其在每台计算节点上均有一个实例，它的主要工作就是监控所在机器的资源现状及任务的运行情况，并定时向JobTracker进行汇报。由此可见，JobTracker不仅要管理系统内所有的计算资源，还要兼顾作业调度的实现，在MapReduce任务非常多的时候，会对JobTracker节点造成很大的压力，致使该节点运行速度变慢，不能很好的完成资源管理和任务调度的需求，甚至会出现内存空间不足导致JobTracker fail的错误。同时，Jobtracker也存在单点故障的问题，在整个系统中，JobTracker只有一个实例在运行，并且没有很好的运行保护措施可以在JobTracker崩溃之后处理崩溃日志并从失败中恢复JobTracker的运行。此外，在TaskTracker资源管理端，仅仅以map/reduce任务的数目来表示资源的占用情况，如果发生两个内存密集型任务分配到同一计算节点的情况，该节点将容易出现内存溢出（OOM）的错误。再者，在MRv1架构中，为保证reduce和map资源都能够调度运行，TaskTracker端的资源被强制划分为只允许Map任务调度运行的Map Slot和只允许Reduce任务调度运行的Reduce Slot，在系统运行的过程中Map/Reduce任务时常不能得到有效的平衡，此种做法造成了资源的浪费，降低了集群的资源利用率。

由于MRv1，Hadoop项目组对MRv1框架进行了全方位的重构和改进，在经过优化设计和实现之后，下一代MapReduce管理框架YARN（Yet Another Resource Negotiator）诞生了。YARN的一个重大改变就是将MRv1中的JobTracker的两个主要功能，即资源管理和作业控制（包括作业监控、容错等），拆分为两个单独的守护进程进行运行管理，这两个进程分别是一个全局的资源管理器（Resource Manager，RM）和每个应用程序对应的应用管理器（Application Manager，AM）。通过对JobTracker的拆分，大大降低了MapReduce任务增多时对主节点的压力，从而使得集群能够容纳更多的计算节点和承载更多的作业运行，这也是YARN的基本设计思想。

YARN 的基本架构如图2.5所示，它由一个资源管理器RM和多个节点管理器（Node Manager）组成，节点管理器在每个计算节点上都会运行，因此YARN在总体架构上仍然与第一代相同，是一个Master/Slaver结构，资源管理器RM为Master角色，节点管理器NM为Slaver，在Master与Slaver之间通过心跳机制来传递信息，资源管理器通过了解汇总各节点资源的使用和空闲情况，结合待调度任务的需求来分配资源和进行任务调度。对比在MRv1架构设计中，由JobTracker 负责所有节点资源的管理、分配和任务调度，YARN框架在设计中采用了两级调度策略：第一层调度由资源管理器RM管理执行，管理所有系统的资源并进行调度分配；第二层调度交由每个应用对应的应用管理器AM来执行调度，将申请到的资源分配给自己的子任务，同时AM也需要通知对应资源所在节点的管理器NM来启动对应的任务进行运行。接下来将详细讲述此三个YARN的重要组成部分。

8.4.1 资源管理器 (Resouce Manager, RM)

在YARN资源管理框架中，资源管理器RM拥有系统级别的资源管理和分配的全部权限，它由两大重要部分构成：调度器

(Scheduler)和应用程序管理器(ApplicationsManager)。调度器是RM实现资源调度的核心组件，它会根据集群的资源容量、作业队列、任务优先级等约束条件，并依据相应的策略完成将集群资源分配给各个应用程序的过程。在RM中，调度器只完成纯粹的资源调度功能，并不负责监控和跟踪应用程序的后续运行状态。此外也不会对由于应用程序bug或硬件故障等原因导致的任务失败做出响应。调度器仅依赖于应用程序提出的资源申请和系统设定的资源分配算法进行工作，它的资源调度单位是容器（container）。容器是一个对系统资源进行封装后的抽象概念，可以包含CPU、内存、磁盘、网络等元素。使用该容器的进程对节点资源的使用量不能超过该容器的上限，同时容器也提供了一个相对隔离的环境来运行任务，从而实现了任务隔离的目的。在本论文所采用的YARN版本中只提供了对CPU和内存的容器封装支持。在YARN中内置了三个可以直接使用的资源调度器：先进先出（FIFO）调度器、公平（Fair）调度器和计算能力（Capacity）调度器，默认使用的调度器为Capacity Scheduler。此外，调度器的调度策略被设计成一个可插拔的组件，用户可以根据自身的需求来设计符合需要的调度策略来替换现有的调度器。

8.4.2 应用管理器(Application Master, AM)

应用程序管理器（ApplicationsManager)负责接受作业的提交，并且负责于应用程序申请分配得到的第一个容器上启动应用程序的专属的ApplicationMaster（AM），并提供任务失败时重新启动AM的服务。AM的主要有以下功能：与RM管理器提交包含一定资源需求的容器申请，其中容器所包含的资源通常由用户来设置；在RM分配给AM容器之后，AM负责将申请到的容器进行再分配来运行子任务；在任务运行在所申请的容器后，与容器所在节点保持心跳连接，同时监控和跟踪任务的运行状态，如果出现任务运行失败的情况，AM将重新提交包含任务运行所需资源的申请，以便重新运行任务。

8.4.3 节点管理器 (Node Manager, NM)

节点管理器在所有可用的计算节点上均有一个实例运行。它的主要功能是监控容器内进程的资源使用状况，并通过心跳包向RM进行汇报，具体包含各个Container的运行状态、运行的Application列表、节点运行时的状态信息等。RM则会根据节点的失败信息或调度算法的计算结果告知NM需要启动和关闭container列表、Application列表等信息。

8.5 Hadoop现有的作业调度算法

作业调度器是的关键组件之一，它的职能是当检测到中存在空闲的工作槽时，根据特定的调度算法从作业队列中选取作业，再进一步选取一个或多个任务作为回复，派发给去执行。调度器性能的好坏将直接影响整个系统的吞吐量和服务效率。本小节将重点研究的作业调度算法及其改进策略。

当前，应用比较广泛的Hadoop作业调度算法包括：FIFO调度算法、Yahoo公司研发的计算能力调度（Capacity Scheduler）算法和Facebook公司研发的公平份额调度（Fair Scheduler）算法。

8.5.1 FIFO调度算法

在Hadoop发展的初期，MapReduce框架所处理的作业主要是单用户提交的大规模批处理作业，因此在MRv1架构中，最早使用也是默认使用的作业调度算法是FIFO算法，所有的作业都被提交到同一个Job Queue中，然后JobTracker根据作业的提交顺序来确定任务的调度顺序。

指　标
疑似剽窃文字表述

1. 用户只需要实现两个简单的函数Map和Reduce，即可高效的实现分布式计算。用户无需关心框架是如何
2. 即可以并行的方式处理输入数据集。通常，map函数负责处理输入的<key,value>键值对，生成一系列以<key,value>形式存储的中间结果。
3. 处理并产生输出文件。在以上过程中，MapReduce框架会自动将输入文件划分为M份小的输入文件
4. 在MapReduce集群系统中，节点机器会被划分为两类不同职责的机器，一类是控制节点，在一般情况下由一台机器来执行控制节点的职能；其余的所有机器被划为另一类，即工作节点。控制节点的主要任务是将map和Reduce任务分配给工作节点。当其发现
5. 心跳通信机制。在HDFS中，NameNode与DataNode通过心跳机制来进行信息交流。每隔固定的时间，DataNode将会发送一个心跳数据包，汇报本节点当前的状态。NameNode
6. 任务的运行情况，并定时向JobTracker进行汇报。由此可见，JobTrack
7. 由JobTracker 负责所有节点资源的管理、分配和任务调度，YARN框架在设计中采用了两级调度策略
8. 申请到的资源分配给自己的子任务，同时AM也需要通知对应资源所在节点的管理器NM来启动对应的任务进行运行。接下来将详细讲述此三个YARN的重要组成部分。
9. 调度器是RM实现资源调度的核心组件，它会根据集群的资源容量、作业队列、任务优先级等约束条件，并依据相应的策略完成将集群资源分配给各个应用程序的过程。在RM中，调度
10. 监控和跟踪任务的运行状态，如果出现任务运行失败的情况，AM将重新提交包含任务运行所需资源的申请，以便重新运行任务。
11. Hadoop现有的作业调度算法
    作业调度器是的关键组件之一，它的职能是当检测到中存在空闲的工作槽时，根据特定的调度算法从作业队列中选取作业，再进一步选取一个或多个任务作为回复，派发给去执行。调度器性能的好坏将直接影响整个系统的吞吐量和服务效

率。本小节将重点研究的作业调度算法及其改进策略。

9. 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第9部分　总字数：2829

相似文献列表　文字复制比：14.3%(405)　疑似剽窃观点：(0)

1	Hadoop YARN资源分配与调度的研究 李媛祯(导师：杨群) -《南京航空航天大学硕士论文》- 2015-01-01	13.1%（371） 是否引证：否
2	5-颜廷帅 颜廷帅 -《学术论文联合比对库》- 2014-12-22	1.2%（33） 是否引证：否
3	S312060010+颜廷帅 颜廷帅 -《学术论文联合比对库》- 2014-12-31	1.2%（33） 是否引证：否

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

在MRv2中，FIFO调度器支持任务的优先级调度，通过设置5个不同的优先级队列，根据队列的优先级的高低，从高到低依次选择不同队列，在同级队列中，按照作业的提交顺序进行作业选择调度和资源分配。FIFO调度算法的设计思想简单易实现，且用户无需对系统进行额外的配置，调度器运行算法的代价也较小。相对而言，FIFO调度算法也存在比较明显的不足之处：首先该算法对各个作业的资源需求考虑不足，容易造成系统不同计算资源利用率的失衡，例如集群中计算密集型作业过多使得内存资源闲置，造成了系统整体性能的下降；其次，算法对于作业优先级的限制过于严格且不支持抢占式执行，无论对于急于执行的高优先级作业还是低优先级作业都十分不友好，常常不能满足用户对于时限的要求；再者，在同队列中的先到先服务的方式对于大作业之后提交的小作业的执行十分不利，采用FIFO调度算法将极大地延长小作业的等待时间，最终使得作业的平均完成时间过长，往往不能达到用户的要求。

8.5.2 Capacity调度算法

Capacity调度算法是一种可以用于多用户集群环境的调度策略，它是一种多队列的调度算法，调度器按照配置对各个队列分配一定比例的资源，用户可以更改系统设置来对每个队列设置资源获取比例以保证系统的资源获取，同时也可以设置资源使用上限防止队列无限制申请资源。每个队列并不是严格按照配置分配固定的资源，而是根据需求动态分配，当其他队列有剩余资源未能加以利用时，调度器会将资源暂时回收，分配给其他队列来加速集群的运行。当原有队列有新的任务提交需要资源运行时，调度器将会对其他队列发起资源回收请求，在被占用资源上的容器中的任务运行结束后，资源将会被返回原有队列，如果该队列在等待一段时间后无法得到资源返还，默认地，Capacity调度器将会强制停止其它队列上的一些任务来返回之前征用的资源。

Capacity调度器实际上是三级资源分配模型，调度流程如图2.8所示，当一个节点上有空闲资源的时候，系统会依次对Job Queue（或用户队列）、Application（或作业）和Container（或请求）进行三次分配过程：首先，调度器对队列进行选择，Capacity调度器将基于优先级的队列结构结合各个队列的资源需求申请转换成一个树形结构，类似于二叉树的搜索思想，对队列结构进行遍历，然后选择一个队列在队列中根据之后的规则选择一个适合的Container进行调度，根据队列调度结果后的变化，对队列的结构进行适当的调整；其次，在系统选择一个队列之后，Capacity Scheduler按照队列内的调度算法对作业进行调度排序。通常队列内的调度算法为FIFO算法或者DRF（Dominant Resource Fairness）算法。DRF算法是一种考虑多种资源（主要是CPU和内存）需求调度的公平资源分配算法。由于不是研究重点，在此不做详细叙述。最后，在上一阶段选中一个应用程序之后，调度器需要对应用程序内的Container进行比较选取。在同一个应用程序中，Container也有可能具有不同的属性，如不同的优先级，节点的选取，申请的资源量等，调度器最终会根据配置的策略选择一个Container进行资源分配。

Capacity Scheduler作为YARN架构默认的调度算法，它在资源合理分配、队列资源限制、安全控制等方面有着不错的表现。它通过多层次的资源调度策略来保证队列中的作业能获得基本的运行资源，同时也限制了作业不能无限制的申请资源侵占其他用户的资源需求，这两点保证是相辅相成的。另外，队列的闲置资源可以暂时借用给其他队列，在需要的时候也可以在一定时间内收回暂借出去的资源来保证作业的运行，说明资源分配具有灵活性。为了支持多用户同时使用集群，Capacity Scheduler也增加了对于作业和文件的权限管理，有效地降低了单个用户误操作对其他用户造成影响。

8.5.3 Fair调度算法

Fair调度算法是由Facebook研发和推广的多用户调度器，其实现方式相比Capacity Scheduler较为相似，也采用了以队列划分资源的方式。Fair Scheduler设计的初衷是让系统中的各个作业能够公平地分配到系统资源。Fair调度器对于小规模的作业比较友好，能在比较理想的时间内得到处理结果。交互性作业也能在一定时间内得到回复。

不同于Capacity Scheduler对于队列的树形组织结构，Fair Scheduler管理系统中待调度作业的方式是使用作业池（pool）。算法将用户提交的作业放置到一个待调度的池中，在默认情况下，各个用户所提交的作业会有相同的概率被调度器选中分配资源并进入运行状态，但是系统也支持管理员进行配置，按照用户优先级或者作业类型提高其任务被调度的概率。类似Capacity调度器，Fair调度器也会保证所有用户有一个最低的资源占有率，使其任务保持运行，而不会陷入停滞，同时如果用户的最小资源份额没有被充分利用，Fair调度器也会将其进行再分配，按照一定的规则分配给其他的用户进行任务运行。Fair Scheduler也支持资源抢占，因此当用户的最小资源份额被其他用户占用时，在任务等待超时后仍然没有得到其他用户任务运行结束返回资源，调度器会终止其他用户在其抢占资源上的运行程序，并返回资源给等待用户。Fair Scheduler也有工作良好的负载均衡机制，其作用是将系统中作业所含的任务尽可能平均地分配到各个计算节点上，用户也可以依据自己的需求进

行负载均衡的定制。

8.6 Hadoop常用调度算法的不足

由上述分析可知，YARN内置的三种调度算法均能完成基础的资源管理和作业调度功能，但是在分配的过程中均存在某些缺陷。FIFO算法是应用最早的算法，它的优点在于算法设计思想简单、调度过程易于理解，算法运行开销很小，对于RM节点运行压力不大。其不足之处在于忽略了不同作业的差异化需求，并且不支持作业抢占，影响排在大作业之后小作业的执行，同时算法也没有考虑机器的实际负载情况。Capacity Scheduler和Fair Scheduler在一定程度上克服了FIFO调度算法的不足，是两种比较常用的调度策略。然而Fair Scheduler实现复杂，且没有考虑集群中各节点计算能力的差异，因此在Map/Reduce任务数量均衡的情况下，各个节点的负载情况有着不小的差异，在异构的环境下算法调度表现不佳。Capacity Scheduler通过多队列的支持提高了调度效率和系统的资源利用率，但是调度器需要用户手动填写配置文件并为作业选择队列而无法由系统自动化完成，为此用户需要了解系统的整体环境配置，在大型系统中，不当的设置会成为调度器性能发挥的阻碍。目前有许多研究人员和大数据从业者对Hadoop资源管理和作业调度算法进行研究和改进。

指　标

疑似剽窃文字表述

---

1. 队列，根据队列的优先级的高低，从高到低依次选择不同队列，在同级队列中，按照作业的
2. 设计思想简单易实现，且用户无需对系统进行额外的配置，调度器运行算法的代价也较小。相对而言，FIFO调度算法也存在比较明显的不足之处：首先
3. 队列有剩余资源未能加以利用时，调度器会将资源暂时回收，分配给其他队列来加速集群的运行。当原有队列有新的任务提交需要资源运行时，
4. 3 Fair调度算法
   Fair调度算法是由Facebook研发和推广的多用户调度器，其实现方式相比Capacity Scheduler较为相似，
5. 系统中作业所含的任务尽可能平均地分配到各个计算节点上，用户也可以依据自己的需求进行负载均衡的定制。
   8.6 Hadoop常用调度算法的不足
   由上述分析可知，

10. 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第10部分　总字数：4607

相似文献列表　文字复制比：31.3%(1443)　疑似剽窃观点：(0)

1	Hadoop平台下基于遗传算法的作业调度的研究与改进 燕明磊(导师：薛涛) -《西安工程大学硕士论文》- 2015-05-24	18.9%（869） 是否引证：否
2	燕明磊_2012076_Hadoop平台下基于遗传算法的作业调度的研究与改进 燕明磊 -《学术论文联合比对库》- 2015-03-11	18.9%（869） 是否引证：否
3	燕明磊_2012076_Hadoop平台下基于遗传算法的作业调度的研究与改进 燕明磊 -《学术论文联合比对库》- 2015-03-11	18.9%（869） 是否引证：否
4	一种自适应步长布谷鸟搜索算法 郑洪清;周永权; -《计算机工程与应用》- 2012-05-21 1	7.9%（365） 是否引证：否
5	一种小规模多种群布谷鸟算法 郑巧燕;莫愿斌;刘付永;马彦追; -《计算机应用与软件》- 2014-10-15	2.5%（113） 是否引证：否
6	基于CS-SVM的网络热点话题变化趋势预测 邱仕坦; -《福州大学学报(自然科学版)》- 2014-05-28 1	2.1%（98） 是否引证：否
7	改进布谷鸟搜索算法求解批量流水线调度问题 郑洪清; -《计算机系统应用》- 2014-10-15	2.0%（94） 是否引证：否
8	随机交叉粒子群优化算法 王联国;洪毅; -《计算机工程与应用》- 2009-06-01	1.9%（88） 是否引证：否
9	基于觅食算子的粒子群优化算法 王联国;洪毅;赵付青; -《计算机应用与软件》- 2009-11-15	1.7%（79） 是否引证：否
10	一种多智能体混合蛙跳算法 王联国;代永强; -《计算机工程》- 2013-07-15	1.5%（71） 是否引证：否
11	求解无约束优化问题的改进布谷鸟搜索算法 苏芙华;刘云连;伍铁斌; -《计算机工程》- 2014-05-15	1.5%（71） 是否引证：否
12	基于高斯扰动的布谷鸟搜索算法 王凡;贺兴时;王燕; -《西安工程大学学报》- 2011-08-25	1.1%（52） 是否引证：否
13	基于布谷鸟搜索算法的主题爬虫策略设计 钱竞远;杨辉华;刘振丙; -《仪器仪表用户》- 2017-06-08	1.1%（51） 是否引证：否

第九章混合遗传布谷鸟算法

资源调度算法是一种组合优化问题，也是一种典型的NP-Hard问题，目前不能在短时间内求得精确的最优解。群智能算法是一种很好的求解目标优化问题的算法，其思想为模拟或借鉴自然界生物生存活动的规律，来减少搜索的频次，使得算法能快速且有效的找到这类问题的最优解，或者在有效时间内找到近似最优解。目前已有一些群智能算法应用于资源调度。然而这些群智能算法都有有待改进的空间。因此，本论文首先对群智能算法进行研究和改进，在对其性能进行提升后，探讨将其应用于资源调度的可能性于有效性。

布谷鸟算法是2009年提出的一种群智能算法，在许多测试和实际应用中已经证明了其相对于其他算法的优秀性，然而它也有后期收敛慢，计算精度不足的缺点。本章将对布谷鸟算法进行研究，并对其进行改进和优化。

9.1 布谷鸟算法

9.1.1 莱维飞行

自然界中很多动物的觅食行为是随机行为过程。随机过程指的是下一步的移动取决于当前位置，选择哪个方向取决于所使用的数学模型。莱维飞行（levy flight）是具有截尾概率分布步长的随机游走。莱维分布是一种连续概率分布，并且其在运动过程中的跳跃性非常大。图3-1是levy flight的轨迹示意图。

9.1.2 布谷鸟算法的数学原理

布谷鸟算法（CS）是基于布谷鸟随机寻窝产卵过程和莱维飞行而产生的。布谷鸟算法具有以下优点：操作简单，参数少，无需在处理优化问题时重新匹配参数。布谷鸟寻觅巢穴的路径是类似随机搜索的方式。为了模拟和简化布谷鸟寻巢的方式，规定以下3个理想的状态：(1)每只布谷鸟每次只产一个卵，即一个解。鸟巢孵化位置遵循随机选择的方式。(2)位置最好的鸟巢即最优的解将被保留到下一代。(3)可供寄生的巢数量n是固定的，布谷鸟蛋被鸟巢主人发现的概率也是固定的，为Pa (Pa ∈ [0，1])。在以上三种规则的基础上，布谷鸟寻窝路径和位置更新公式如下：

$$Xi(t+1) = Xi(t) + \alpha \oplus L(\lambda) \quad (i = 1, 2, \ldots, n) \quad (3\text{-}1)$$

式中 Xi(t)表示第i个鸟窝在第t代的鸟窝位置；α表示步长控制量；⊕为点乘； L(λ) 服从Levy分布，并且L~ u = t –λ (1 < λ ≤ 3)。在鸟巢更新位置后，将随机数r (r ∈ [0，1])与Pa 对比，若r > Pa ，则对Xi(t+1) 进行随机修改，否则不做任何操作。最后对解的最优值yi(t + 1)进行保留，此时仍记为Xi(t+1)。

9.2 遗传算法

9.2.1 遗传算法产生的背景

遗传算法(Genetic Algorithms GA)源自上世纪六七十年代，由 John Holland教授与其同事、学生们提出，依据达尔文的进化论的主要思想–物竞天择，适者生存，目的是通过模拟生物进化过程构造人工系统模型。后来随着相关理论著作的出版，以及在机器学习和性能优化等领域的应用，该模型理论正式定名为遗传算法。

9.2.2 遗传算法的原理

9.2.3 遗传算法的特点

遗传算法是一种优化算法，因而需要满足最优性原理。最优性原理是针对现实生活中的这一类问题：这个问题可被分为若干个子问题，对于任意的子问题t都只依赖于当前阶段的状态，而与t之前的所有其它子问题的状态均无关，这是一种多阶段决策的问题。这类问题中一种最通用但是低效的方法是枚举法，该方法的基本思想是穷举解集范围内的所有解，然后依据求解目标找出最优解；这种算法的一种改进算法是搜索算法，它的基本思想是把搜索范围缩小到解集的某一个子集或几个子集的范围内，通过缩小搜索域的方式可以大大的减少计算量；此外，这类问题还有另外一种效率较高但不具有通用性的解决办法，启发式算法：它针对每个待求解的问题必须明确它的特定的启发式规则。遗传算法兼顾了以上两类算法的优点，具有通用性，高效性，并且可以应对问题规模的动态增长，此外还具有以上两类算法不具备的一些特点：

(1) 遗传算法以种群作为出发点，从本质上来说是并行的，这是遗传算法最本质的特点。根据进化理论，种群是物种进化的基本单位，种群间的遗传操作是并行的；同时在种群内部，个体间的遗传操作也是并行的，因此具有种群与个体两个层面上的并行。

(2) 遗传算法具有智能性，物种在进化中遵守物竞天择，适者生存的进化法则。适应环境的个体有更大的几率生存下去，并且把自己的基因传给后代；而适应环境能力差的个体生存的几率较小或者被淘汰，因此进化的过程就是一个智能选择优良个体淘汰劣质个体的过程。

(3) 遗传算法具有灵活性，遗传算法可以通过目标函数和适应度函数来控制种群进化的方向，因此在实际问题中可通过修改控制这两个函数来应对不同的应用场景。

9.3 混合遗传布谷鸟算法

9.3.1 混合遗传布谷鸟算法模型

在基本的布谷鸟算法中，步长是由levy flight产生的随机长度。因此在搜索过程中，布谷鸟算法能比较容易跳出局部最优值

，进而增大了搜索到全局最优的概率。然而在局部搜索的过程中由于levy flight产生的步长具有随机性，因此局部求最优值的精度比较差，局部收敛性较弱。因此，布谷鸟算法在算法的后期收敛比较慢，需要较多次数的迭代才能收敛到局部最优值，进而比较得出全局最优值。在算法运行的理想情况下，解空间的一部分解应该在附近搜索局部最优值，即可能的全局最优值，另一部分通过大步长的levy flight跳出当前局部去搜索其他可能的全局最优值。

因此针对这个问题，对布谷鸟算法进行改进。遗传算法具有比较好的局部搜索性，能比较快的使算法收敛到局部最优值。本论文将布谷鸟算法和遗传算法进行结合搜索，提出了混合布谷鸟遗传算法（GCS），能比较好的处理好全局寻优能力和局部收敛性之间的关系。在布谷鸟算法求解的过程中，适应性较好的值周围可能存在是局部最优值，因此全局最优的可能性也较大，因此对此部分解应用遗传算法，使其尽快收敛到局部最优，其他适应值较差的值继续应用levy flight随机游走，保持算法的全局搜索性。

9.3.2 混合遗传布谷鸟算法的实现

根据第二章的分析，GCS算法的具体过程与步骤如下：

步骤1（初始化）：首先对解空间进行适当的编码，以便适应遗传算法的交叉遗传的需要。随机生成n个鸟巢，搜索空间的维数为m，最大迭代次为itermax，适应值较好的阈值为fV (fV∈ [0，1])，找出当前最优鸟窝位置Xb(0) (b∈{1，2，…，n})和最优解fmax。

步骤2（循环体）：保留上一代最优鸟窝的位置Xb(t-1)，t为正整数，表示迭代的代数。对适应值比较好的前p个(p = n × fV)解，应用遗传算法，使其尽快收敛到局部最优值。取适应值较高的两个值进行交叉遗传，对新的解空间内p+2个值进行适应值计算，取前p个较优值。对剩余的n-p个解应用布谷鸟算法的更新方式按照公式3-1更新位置。经过两种算法的并行计算，得出一组新的鸟窝位置，即得到了新的一组解。对比组内各个解的适应度，让适应度较好的鸟窝代替适应度差的。最终保留前n个解，使得进行遗传和迭代之后解空间保持不变，从而得到下一代鸟窝位置。

步骤3：找出上一步中最后得到的解空间中最优的一个鸟巢Xb(t)，并判断是否满足输出条件（迭代次数达到限制或是其他限制条件）。

9.4 算法性能测试与分析

9.4.1 实验测试设计

本文通过求四个标准测试函数的最小值为例进行仿真实验，对该算法的性能进行评估，测试平台为matlab7.1和windows7，机器的cpu主频为2.10GHz，内存为4.00 GB。测试函数如下：

f1(x) = xi2

f2(x) = 10d + i=1d[xi2-10cos(2πxi)]

f3(x) = i=1n-1[100xi+1-xi22+(1-xi)2]

f4(x) = i=1dxi2-10cos2πxi+10

实验中算法参数设置为：鸟窝规模为20，阈值fV为0.2，用于遗传算法按照函数的不同设置不同的精度进行编码，精度按照表3-1所示。使用基本CS算法和GCS算法对上述四个测试函数分别进行寻优测试，最终测试结果采用独立运行20次后的平均值。当算法的全局最优值收敛到目标精度值时停止算法。

表3-1 用于遗传算法的精度

函数维数搜索范围理论最优值遗传算法精度目标精度

f1 10 [- 100，100] 0 1/2048 1E - 5

f2 20 [- 5.12，5.12] 0 1/2048 200

f3 30 [- 30，30] 0 1/2048 7000

f4 10 [- 5.12，5.12] 0 1/2048 20

性能评价采用如下方法：1、对算法收敛到目标精度值所需的迭代次数进行比较；2、在一定的迭代次数下，对算法的收敛速度和计算精度进行比较。

9.4.2 实验结果及分析

将算法的迭代次数设定在200次，各算法的目标收敛精度如表3-1所示。表3-2和图3.1~3.4为标准函数在各收敛精度下每个算法独立运行20次后迭代次数对比结果展示。由表3-2可以看出：对单峰函数f1，GCS比CS有了3~4个数量级的提高;对高维函数f2，单峰病态函数f3，复杂多峰函数f4，GCS也比CS在适应值上有很大提高，平均迭代次数均有显著的降低。

表3-2 用于遗传函数的精度

函数算法最小适应值最大适应值平均适应值平均迭代次数

f1 CS 0.5181 4.7514 1.3523 464.650

GCS 0.0004 3.1542 0.0348 231.100

f2 CS 115.728 134.259 124.020 3973.700

GCS 108.956 133.145 114.019 2362.800

f3 CS 16180 78693 26510367.350

GCS 2953.5 20712 7425.2214.750

f4 CS 16.880 28.772 22.452 270.700

GCS 9.054 30.245 17.249 176.250

图1至图4是函数 f1 ，f2 ，f3 ，f4 采用 CS 算法和 GCS算法分别单独运行20次的平均适应度的对比图（采用对数比较）。可以从图中看到，GCS在收敛速度上较快，在寻优精度上表现较好。综上所述，改进后的GCS对比CS在寻优精度和收敛速度方面都较为优秀。

布谷鸟算法尽管表现优秀，但也存在后期收敛速度较慢、精度有待提高等缺陷。本文提出了一种新的混合遗传布谷鸟算法(GCS)。经过测试表明，改进后的混合遗传布谷鸟算法在后期收敛速度和寻优精度上有所提高。

9.5 本章小结

本章主要介绍了布谷鸟算法及其改进方式。布谷鸟算法是一种较为新型的群智能算法，它是通过模拟布谷鸟寻窝产卵以及结合莱维飞行而设计的。本章首先介绍布谷鸟的生活习性化及莱维飞行过程的数学原理。然后对该算法的优势与缺点进行了阐述。针对其后期收敛较慢，局部计算精度不高的缺点，提出结合遗传算法进行修正。最后将布谷鸟算法进行改进，并且通过实验对改进的布谷鸟算法与原有的布谷鸟算法进行了性能比较，结果显示改进的布谷鸟算法的性能要优于原本的布谷鸟算法。

指　标

疑似剽窃文字表述

---

1. 类似随机搜索的方式。为了模拟和简化布谷鸟寻巢的方式，规定以下3个理想的状态：(1)每只布谷鸟每次只产一个卵，
2. 后来随着相关理论著作的出版，以及在机器学习和性能优化等领域的应用，该模型理论正式定名为遗传算法。

   9.2.2 遗传算法的原理

   9.2.3 遗传算法的特点

   遗传算法是一种优化算法，因而需要满足最优性原理。最优性原理是针对现实生活中的这一类问题：这个问题可被分为若干个子问题，对于任意的子问题t都只依赖于当前阶段的状态，而与t之前的所有其它子问题的状态均无关，这是一种多阶段决策的问题。这类问题中一种最通用但是低效的方法是枚举法，该方法的基本思想是穷举解集范围内的所有解，然后依据求解目标找出最优解；这种算法的一种改进算法是搜索算法，它的基本思想是把搜索范围缩小到解集的某一个子集或几个子集的范围内，通过缩小搜索域的方式可以大大的减少计算量；此外，这类问题还有另外一种效率较高但不具有通用性的解决办法，启发式算法：它针对每个待求解的问题必须明确它的特定的启发式规则。遗传算法兼顾了以上两类算法的优点，具有通用性，高效性，并且可以应对问题规模的动态增长，此外还具有以上两类算法不具备的一些特点：

   (1) 遗传算法以种群作为出发点，从本质上来说是并行的，这是遗传算法最本质的特点。根据进化理论，种群是物种进化的基本单位，种群间的遗传操作是并行的；同时在种群内部，个体间的遗传操作也是并行的，因此具有种群与个体两个层面上的并行。

   (2) 遗传算法具有智能性，物种在进化中遵守物竞天择，适者生存的进化法则。适应环境的个体有更大的几率生存下去，并且把自己的基因传给后代；而适应环境能力差的个体生存的几率较小或者被淘汰，因此进化的过程就是一个智能选择优良个体淘汰劣质个体的过程。

   (3) 遗传算法具有灵活性，遗传算法可以通过目标函数和适应度函数来控制种群进化的方向，因此在实际问题中可通过修改控制这两个函数来应对不同的应用场景。
3. 布谷鸟算法模型

   在基本的布谷鸟算法中，步长是由levy flight产生的随机长度。
4. 实验测试设计

   本文通过求四个标准测试函数的最小值为例进行仿真实验，对该算法的性能进行评估，测试平台为matlab7.1和windows7，机器的cpu主频为2.
5. 基本CS算法和GCS算法对上述四个测试函数分别进行寻优测试，最终测试结果采用独立运行20次后的平均值。

11. 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第11部分　总字数：7366

相似文献列表　文字复制比：75.3%(5548)　疑似剽窃观点：(0)

1	Hadoop YARN资源分配与调度的研究 李媛祯(导师：杨群) -《南京航空航天大学硕士论文》- 2015-01-01	75.0%（5527） 是否引证：否
2	一种Hadoop Yarn的资源调度方法研究 李媛祯;杨群;赖尚琦;李博涵; -《电子学报》- 2016-05-15	8.3%（613） 是否引证：否
3	Hadoop Yarn 框架原理及运作机制 - Focus,Just For Today - 博客频道 - CSDN.NET -《网络（http://blog.csdn.net）》- 2017	2.7%（198） 是否引证：否
4	马鹏将 马鹏将 -《学术论文联合比对库》- 2015-01-04	2.0%（149） 是否引证：否
5	李芳林_并行推荐算法的研究与实现	2.0%（149）

	李芳林 - 《学术论文联合比对库》- 2015-01-02	是否引证：否
6	大数据计算框架及其资源调度机制研究 - 豆丁网	2.0%（147）
	-《互联网文档资源（http://www.docin.com）》- 2016	是否引证：否
7	Hadoop YARN大数据计算框架及其资源调度机制研究	1.3%（98）
	董春涛;李文婷;沈晴霓;吴中海; -《信息通信技术》- 2015-02-15	是否引证：否
8	SA12225077_徐磊_1	1.2%（87）
	徐磊 -《学术论文联合比对库》- 2014-08-19	是否引证：否
9	SA12225077徐磊	1.2%（87）
	徐磊 -《学术论文联合比对库》- 2015-01-04	是否引证：否
10	573+刘盼红	0.9%（65）
	-《学术论文联合比对库》- 2015-03-24	是否引证：否
11	SA12225077_徐磊_2	0.7%（50）
	徐磊 -《学术论文联合比对库》- 2014-09-18	是否引证：否
12	152130200058.徐辉.学术硕士	0.7%（49）
	-《学术论文联合比对库》- 2015-07-27	是否引证：否
13	基于Hadoop的密度聚类算法并行化分析与研究	0.7%（49）
	徐辉 -《学术论文联合比对库》- 2015-05-19	是否引证：否
14	基于Hadoop的密度聚类算法并行化分析与研究	0.7%（49）
	徐 辉 -《学术论文联合比对库》- 2015-06-08	是否引证：否
15	基于Hadoop的密度聚类算法并行化分析与研究	0.7%（49）
	徐 辉 -《学术论文联合比对库》- 2015-06-09	是否引证：否
16	基于Hadoop的密度聚类算法并行化分析与研究	0.7%（49）
	徐 辉 -《学术论文联合比对库》- 2015-06-09	是否引证：否
17	yarn的工作流程 - 军军的专栏 - 博客频道 - CSDN.NET	0.6%（41）
	-《网络（http://blog.csdn.net）》- 2017	是否引证：否
18	Spark入门实战系列--4.Spark运行架构 - yirenboy的专栏 - 博客频道 - CSDN.NET	0.5%（38）
	-《网络（http://blog.csdn.net）》- 2017	是否引证：否
19	并行图挖掘算法的研究与实现	0.4%（31）
	王虹旭(导师：吴斌) -《北京邮电大学硕士论文》- 2015-03-10	是否引证：否
20	王虹旭-2012110718-并行图挖掘算法的研究与实现-0112	0.4%（31）
	王虹旭 -《学术论文联合比对库》- 2015-01-12	是否引证：否
21	面向多用户环境的MapReduce集群调度算法研究	0.4%（31）
	陈重韬; -《高技术通讯》- 2017-04-15	是否引证：否

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第十章基于GCS的Hadoop作业调度算法的设计与实现

10.1 Hadoop资源调度介绍

10.1.1 Hadoop资源调度模型分析

Hadoop2.x支持内存和CPU两种不同资源类型的管理和分配，资源调度器采用动态资源分配机制，通过处理由应用程序提交的Request请求来分配这两种资源。

当NM通过ResourceTracker协议向RM注册、汇报节点状态时，它会提交本节点中可分配的CPU和内存资源总量。在系统中默认值是每个NM具有8GB内存，8个虚拟CPU核心。NM周期性地主动向RM发出请求并汇报资源信息，而应用程序通过ResourceRequest向RM请求资源，一个请求包含了以下字段：

（1）Priority：表示资源优先级（0优先级最大，数字越大优先级越小）。

（2）ResourceName：表示请求资源的位置（*：表示任意位置资源；/rack-name：在这个机架上任意一个节点申请资源；/host-name：在某个节点申请资源）。

（3）Capability：资源的规格，包括上面提到的两种资源的量（例如：2GB内存，一个虚拟CPU核）。

（4）Num_Container：请求满足上面资源规格描述的Container数量。

（5）Relax_locality：是否强制要求本地性。因为AM会根据数据副本的数量提交多个资源请求，在响应ResourceRequest时，会按照本地作业（NODE_LOCAL），本机架作业（RACK_LOCAL）和任意位置（OFF_SWITCH）的顺序进行响应。这样有一个被响应就会使得对应的较大范围的请求被撤回（例如本地作业请求被响应则本机架作业请求和任意位置请求被撤回）。

调度器会根据一系列条件进行调度，首先是选择队列，由于队列具有以下两种属性，调度器需要考虑这两种属性来确定队列：

（1）最小容量：调度器需要努力满足这个最小容量的值，在一个多队列的环境下，调度器会选择队列实际容量和最小容量

差值最大的那些队列，把资源分配给这些队列，这样做是为了保证这些队列内的作业能够保持执行的效率。当一条队列空转时，其资源会分配到其他队列供其他队列的应用使用，直到此队列再一次有应用提交并申请资源。

（2）最大容量：在资源情况比较紧张时，队列也会设置一个最大容量。最大容量是队列的一个硬性要求值。在任何时刻，队列的资源占有量都不能超过这个值。一般而言，为了提高队列效率，这个值不会被设置，这样，一个比较繁忙队列能够在其他队列都不工作时完全使用集群资源（占用100%的资源），在其他队列有应用时再逐步释放。在队列中，存放了该队列的用户提交的应用，调度器会根据资源情况和应用的请求响应资源请求。新的HadoopYARN实现的是两层调度机制：在第一层中，上述的ResouceRequest会被应用程序提交到RM中，而调度器的唯一工作就是结合由应用提交的ResouceRequest和各节点的资源情况，确定是否对ResouceRequest进行响应，如果响应这些请求则会在节点上启动一个到多个Container；如果没有响应则在下一次节点状态更新的时候再判断是否需要响应这些请求。在第二层中，应用程序的AM从RM中定期取回分配到的Container，接着在应用内部将这些Container分配给自己要完成的任务。在HadoopYARN中，分配资源要经过以下步骤：

(1)NM用心跳信息向RM汇报节点情况。

(2)RM中的ResourceTracker服务会返回一个应答，包含要启动和释放Container的信息。

(3)服务向调度器转发NODE_UPDATE事件。

(4)调度器把释放的资源重新标记为可用，再按照一定策略将节点上的资源分配给应用程序。

(5)AM向RM发送心跳信息。

(6)RM把分配的Container应答给AM。

(7)AM把Container分配给自己内部的任务。

设计调度器的关键就是完成第四步的策略，选择合适的调度策略进行资源分配，可以使得资源被更为合理的使用，从而提高应用的执行效率。

10.1.2 资源调度存在的问题

资源调度是Hadoop中一个重要的研究领域。不同于Hadoop0.x中的JobTracker组件，HadoopYARN中的资源调度器是一个“纯调度器”，不再从事任何与具体应用程序相关的工作，如不负责监控或跟踪应用程序的执行状态，不负责重新启动因应用程序执行失败或硬件故障而产生的失败任务等，而是仅仅负责根据各个应用程序的资源需求对整个集群的资源进行统一的管理和分配。资源分配的单位用“资源容器”（Container）表示，Container是一个冬天资源分配单位，它将内存与CPU资源封装在一起以限定每个任务使用的资源量，所以如何对资源进行合理的分配，使得哪个资源分配给哪个Container执行，关系到整个集群中的作业执行效率。当用户向YARN中提交一个应用程序后，YARN将分两个阶段运行该应用程序，首先是启动AM，其次是由AM创建应用程序，并为它申请资源，监控整个过程直到运行完成。在AM资源申请过程中，调度器采用轮询的方式为其分配资源，而并未考虑到是否有其他更优的资源可供AM使用。图4.1描述了当前Hadoop调度器的资源分配策略。当AMA向RM中的Scheduler组件申请Container时，假设当前轮询的位置的节点Node2且Node2中可分配资源满足AM A所需资源量，Scheduler将Node2中的资源分配给AM A，然而从图4.1可以看出，相对于Node2，Node3的资源负载更小CPU处理速度更快，故将Node3中资源分配给AM A更合理。

对于Hadoop集群资源调度问题，可以把Container看作是一个关于资源分配的映射目标，可行解空间就是资源分配策略的集合。在可行解空间中，如何为每一个Container提高最优的资源，这是一个无法确定的问题，同时由于无法确定最优解，使得资源的分配问题是一个NP-Hard问题。

10.1.3 合理调度分配资源的重要性

当前，Hadoop 技术越来越成熟，资源调度问题也越来越重要。在 Hadoop YARN 中，集群资源以大小不等的 Container 为单位进行分配，随着集群规模越来越大，集群中各个节点不可能是同构的，同一作业在不同计算能力的节点资源上执行时，其执行效率有很大的差异，所以选择一个合适的节点为 Container 分配资源非常重要。 Hadoop 集群中，影响作业执行时间的因素有很多，但其中一个重要的因素为运行作业中所属任务的资源的性能。如果一个作业中的任务被分配到一个性能相对较差的节点上，如负载高、CPU 处理速度慢或当前节点上没有运行任务所需的数据副本，此时执行该任务需要花费较长的时间，这在一定程度上大大的延长了作业的执行时间，甚至造成了集群资源的浪费。 针对上述问题，本章首先分析 Hadoop 资源调度的请求机制与模型，详细剖析了资源调度中存在的问题，在此基础上，引入蚁群算法和粒子群算法，实现 Hadoop 资源的合理化分配。最终，根据本章算法编写相应的 Hadoop 资源调度器，并进行实验验证，实验表明本章调度器能够有效的提高集群资源分配的合理化，达到缩短作业执行时间的目的。

10.2 基于GCS的资源调度算法设计

10.2.1 Hadoop资源调度模型分析

本节将混合遗传布谷鸟算法应用于Hadoop资源调度问题，Hadoop集群模型化描述如下：

集群环境模型记为G={E,R,J,C}，其中，E为布谷鸟所产的卵的位置的集合，即可行解的集合；R为Hadoop节点资源集合；J为待调度作业集；C为容器(container)集合，资源集合R={R1,R2,…,Rn}由n个节点及其资源构成，其中，第i个节点上的资源Ri={cpuSpeedi,Mi,resMi,loadi}。cpuSpeedi为节点i上的CPU运行速率，节点i的总资源量用Mi表示，resMi为节点i的空闲资源量，resMi≤Mi，0≤i＜n，loadi表示节点i的负载。作业集J={J1,J2,…,Js}表示当前集群上运行的作业量为s，第j个作业Jj={resTotalJj,resMJj,progj},0≤j＜s，其中，resTotalJj为Jj所需资源的总量；resMJj为分配给Jj的实际资源量；progj表示Jj的运行进度。容器集C={C1,C2,…,Ct}表示集群中所有申请了资源的容器集合，第m个容器

Cm={Jmj,resMCm},0≤j < s,0≤m < t，Jmj为Cm所属的作业，resMCm为容器Cm所申请的资源量。

对容器进行资源分配时，对任意Ri∈R，Cm∈C，如果资源Ri可以满足容器Cm所申请的资源量，则允许Ri为容器Cm分配执行任务所需要的资源。满足容器Cm的资源Ri可能有不止一个。

容器选择资源节点可以表示成为一个t维度的向量（Xd = X1X2X3…Xt）其中0 < Xi < m（d为第d次迭代，Xi的值表示Ci容器选择的资源节点，例如 X5 = 1023，表示第一个容器选择第一个节点，第二个容器没有选择节点，第三个容器选择第二个节点，第四个容器选择了第四个节点。

为了保证节点资源得到充分、合理的分配，使得集群可以正常运行。本论文规定：当算法满足以下两个约束条件时，单个布谷鸟的一次迭代过程视为结束。

条件一：对所有的资源节点，不能分配任何额外容器。即当前所有资源节点已经饱和分配。条件二：所有的容器都已经分配给指定的资源节点，并且资源节点没有超负荷分配运行。

10.2.2 基于GCS的资源调度算法的设计说明

在本论文的第三章已经说明了改进后的GCS算法拥有较好的寻优能力。本小节将说明如何将GCS算法应用于Hadoop资源调度，以及针对具体的细节进行算法的调整。在GCS算法的应用中，一次布谷鸟搜索和一次遗传算法的求解表示一次资源分配方案的生成。下面详细介绍本章算法的具体流程：

（1）初始化种群：集群Hadoop环境下，无论是Map或Reduce，节点的CPU速率、内存量与负载等因素对任务的执行效果具有至关重要的影响。一般而言，任务不分配到CPU执行速度低、内存量小、负载较重的节点上执行，而应该在高性能节点（如CPU速率高、负载小、对应作业失败率小的节点）上启动，以此减少任务执行时间。

Hadoop YARN资源调度框架中，AM负责作业启动与监控，RM负责集群中所有资源的管理和分配，并监控、管理各节点上AM的状态。AM与RM之间的通信通过心跳传输机制实现。本章设计并实现了一个资源调度器，通过该资源调度器从NM获取节点CPU速率、作业失败记录、内存容量及负载情况等信息。

在算法初始化时，随机生成k组解，每组解的形式如4.2.1小节中描述的t维向量，表示各个容器选择资源节点的情况。抛弃不符合终止条件的解，并重新生成相同数量的解，直到符合调节的解数量为k或者达到尝试上限(attempt limit)为止。如果算法达到了上限，不合格的解按照以下第（2）步中产生新的不合格解时的步骤进行解的合理化。

（2）生成新的解：

布谷鸟算法：在一次迭代过程中，布谷鸟通过一次levy飞行来寻找下一个可能的解的位置。通过莱维飞行得出下一个可供产卵的巢的位置，即可行解。在执行一次莱维飞行之后得出的新的解并不一定符合终止条件。此时需要计算每个资源节点资源分配情况，列出所有超出资源总量的节点，清理超出可分配资源量的容器。然后尝试分配容器到有结余资源的节点，直到无法分配任何容器到所有节点为止。

遗传算法：选取适应度最优的两个解，将其进行交叉遗传，相比于随机搜索，交叉遗传有较大的概率得到更优的解。对资源分配进行交叉遗传的具体做法为：选取适应值最好的两个解向量，将两个向量的编码各自从中点分成相等的两段（对于可选容器为奇数，即可行解的编码不能分成均等的两段的情况，则随机分为首长尾短或首短尾长的两个片段），然后进行交叉遗传，即互换首尾的编码，组成两个新的编码，即生成了两个新的可行解。例如最优的两个解的编码分别为X1=1230321、X2=2310132，经过交叉遗传产生的新的解为Y1=1230132、Y2=2310321。

在容器清理和再分配的过程中，加入了对容器优先度的考虑，如果随机清理，存在会让即将完成的任务和需求资源较大的容器在较长的时间内得不到调度，这将极大的影响任务的完成时间。因此，在调度的时候应该给予此类任务较高的优先度。在资源调度算法迭代过程中的容器优先级计算公式如下式（4-1）所示：

PRm=a×resMCmm=0tresMCmn+b×progj （4-1）

在上述公式中，PRm表示容器C_m的调度优先级。a,b均为常数，分别对应容器和容器所属作业进度的权重值。在默认情况下，取a=b=1，此时需求资源较大的容器和即将完成的作业的所属任务申请的容器的调度优先级是近似相等的。

（3）适应度计算：当资源分配满足终止条件时，算法的一次迭代过程结束，根据以下公式（4.5）计算当前所得解的目标函数EPk的值：

（4.5）

其中：Wj代表Jj运行进度的权重值，计算方法如下式（4.6）所示。

（4.6）

10.3 基于GCS的资源调度算法实现

10.3.1 主要数据结构介绍

Hadoop调度器负责资源分配，本节以Capacity调度器的源代码为例介绍主要的实现类：

（1）CapacityScheduler类主要作为Capacity调度器的资源管理。当RM收到来自NM发送的心跳信息后，将向CapacityScheduler发送一个Scheduler EventType的NODE_UPDATE事件。调度器接收到NODE_UPDATE事件后，首先释放已成功或失败的Container。如果此时调度器有资源空闲，调度器就会尝试在这个节点上按照调度分配策略响应应用程序提交的Resource Request。

（2）SchedulerNode类用于节点对资源的管理。在Hadoop YARN中，调度器通过汇集各个应用程序的资源请求进行资源分配。集群初始化时，程序从状态机rmNode中获得相关信息，然后不断维护节点上的SchedulerNode以更新集群资源情况

，并确认是否能在此节点上分配资源。当一个Container在节点上启动时，SchedulerNode类会调整与可分配资源相关的内部属性信息值，并记录这个Container的情况。

（3）Scheduler Application类主要用于记录应用程序对集群资源的占用情况和程序所在队列情况。调度器通过维护SchedulerApplication以监控集群资源的分配，当某个应用程序过度占用资源时，调度器会通过心跳信息通知应用程序释放Container，如果应用程序仍不释放资源，调度器将请求RM杀死此Container。同时，SchedulerApplication中包含了此应用程序提交的ResoureRequest，在每次allocate接口被调用时，AM发来的程序请求会被更新到SchedulerApplication中，以供调度算法在调度时进行计算。

10.3.2 关键接口的实现

资源调度器是RM的一个核心部件，通过RM的组件和服务与其他组件进行交互。Hadoop保留了一系列的接口供RM其他模块调用，本章资源调度器借鉴Capacity调度器的实现方式，实现了与资源调度相关的接口，具体接口如下：

（1）GcsSchedulerNode：节点信息接口，继承自SchedulerNode，保留了节点当前硬件状态和资源状态，为信息素矩阵的初始化提供相关的属性信息。在资源调度过程中，GcsSchedulerNode会根据分配或释放Container情况动态更新节点资源情况，使调度器能够正确显示集群资源情况。

（2）GcsSchedulerApp：应用程序接口；包含了该应用程序提交的资源请求信息和当前已获取的资源信息。在调度器进行资源分配时，获取应用程序中的最新资源请求信息以更新数据结构，为下一步资源分配提高数据参考，从而避免因资源请求超出节点容量而造成的资源分配失败。GcsSchedulerNode与GcsSchedulerApp均以映射表Map的形式存储在调度器中，以便在资源分配时供调度器查阅。

（3）GcsScheduler：本章资源调度器的关键结构，类似于CapacityScheduler类。GcsSchedulerHadoop调度器负责资源分配，本节以Capacity调度器的源代码为例介绍主要的实现类：实现了调度器与RM的其他组件交互的接口，主要包括handle函数和allocate函数。handle函数是调度器与Eventdispatcher的接口，通过这个接口，Eventdispatcher会把调度器事件（SchedulerEvent）以参数的形式转发给调度器，调度器会根据事件的不同做出相应的动作。allocate函数主要用于ApplicationMasterService的调用，通过该函数，调度器汇集从AM发来的各种请求，然后根据这些信息进行相应操作，如释放Container，更新ResourceRequest，把节点从黑名单中加入或者移除等。

10.4 本章小结

本章针对Hadoop集群的资源调度问题，将第三章提出的混合遗传布谷鸟算法应用于Hadoop资源调度。算法在应用的过程中，充分考虑了节点的各个因素对布谷鸟选择新巢的影响，在修正莱维飞行和遗传算法产生的新解时考虑了节点的CPU频率、内存、负载及容器所属作业在节点上失败率等因素，做到了对集群性能较为全面的评估。以此为前提，资源分配能够更加充分地利用系统资源。在最后简略阐述了算法实现所需要注意的事项和实现的接口。

指　标
疑似剽窃观点
1. 当AMA向RM中的Scheduler组件申请Container时，假设当前轮询的位置的节点Node2且Node2中可分配资源满足AM A所需资源量，Scheduler将Node2中的资源分配给AM A，然而从图4.1可以看出，相对于Node2，Node3的资源负载更小CPU处理速度更快，故将Node3中资源分配给AM A更合理。 2. 最终，根据本章算法编写相应的 Hadoop 资源调度器，并进行实验验证，实验表明本章调度器能够有效的提高集群资源分配的合理化，达到缩短作业执行时间的目的。
疑似剽窃文字表述
1. 资源调度模型分析 Hadoop2.x支持内存和CPU两种不同资源类型的管理和分配，资源调度器采用动态资源分配机制，通过处理由应用程序提交的Request请求来分配这两种资源。 当NM通过ResourceTracker协议向RM注册、汇报节点状态时，它会提交本节点中可分配的CPU和内存资源总量。在系统中默认值是每个NM具有8GB内存，8个虚拟CPU核心。 2. 当一条队列空转时，其资源会分配到其他队列供其他队列的应用使用，直到此队列再一次有应用提交并申请资源。 3. 在队列中，存放了该队列的用户提交的应用，调度器会根据资源情况和应用的请求响应资源请求。新的HadoopYARN实现的是两层调度机制：在第一层中，上述的ResouceRequest会被应用程序提交到RM中，而调度器的唯一工作就是结合由应用提交的ResouceRequest和各节点的资源情况，确定是否对ResouceRequest进行响应，如果响应这些请求则会在节点上启动一个到多个Container；如果没有响应则在下一次节点状态更新的时候再判断是否需要响应这些请求。在第二层中，应用程序的AM从RM中定期取回分配到的Container，接着在应用内部将这些Container分配给自己要完成的任务。在HadoopYARN中，分配资源要经过以下步骤： (1)NM用心跳信息向RM汇报节点情况。 (2)RM中的ResourceTracker服务会返回一个应答，包含要启动和释放Container的信息。

(3)服务向调度器转发NODE_UPDATE事件。
(4)调度器把释放的资源重新标记为可用，再按照一定策略将节点上的资源分配给应用程序。
(5)AM向RM发送心跳信息。
(6)RM把分配的Container应答给AM。
(7)AM把Container分配给自己内部的任务。
设计调度器的关键就是完成第四步的策略，选择合适的调度策略进行资源分配，可以使得资源被更为合理的使用，从而提高应用的执行效率。
10.1.2 资源调度存在的问题
资源调度是Hadoop中一个重要的研究领域。不同于Hadoop0.x中的JobTracker组件，HadoopYARN中的资源调度器是一个“纯调度器”，不再从事任何与具体应用程序相关的工作，如不负责监控或跟踪应用程序的执行状态，不负责重新启动因应用程序执行失败或硬件故障而产生的失败任务等，而是仅仅负责根据各个应用程序的资源需求对整个集群的资源进行统一的管理和分配。

4. 当用户向YARN中提交一个应用程序后，YARN将分两个阶段运行该应用程序，首先是启动AM，其次是由AM创建应用程序，并为它申请资源，监控整个过程直到运行完成。在AM资源申请过程中，调度器采用轮询的方式为其分配资源，而并未考虑到是否有其他更优的资源可供AM使用。图4.1描述了当前Hadoop调度器的资源分配策略。
5. 对于Hadoop集群资源调度问题，可以把Container看作是一个关于资源分配的映射目标，可行解空间就是资源分配策略的集合。在可行解空间中，如何为每一个Container提高最优的资源，这是一个无法确定的问题，同时由于无法确定最优解，使得资源的分配问题是一个NP-Hard问题。
6. 合理调度分配资源的重要性
当前，Hadoop 技术越来越成熟，资源调度问题也越来越重要。在 Hadoop YARN 中，集群资源以大小不等的 Container 为单位进行分配，随着集群规模越来越大，集群中各个节点不可能是同构的，同一作业在不同计算能力的节点资源上执行时，其执行效率有很大的差异，所以选择一个合适的节点为 Container 分配资源非常重要。 Hadoop 集群中，影响作业执行时间的因素有很多，但其中一个重要的因素为运行作业中所属任务的资源的性能。如果一个作业中的任务被分配到一个性能相对较差的节点上，如负载高、CPU 处理速度慢或当前节点上没有运行任务所需的数据副本，此时执行该任务需要花费较长的时间，这在一定程度上大大的延长了作业的执行时间，甚至造成了集群资源的浪费。针对上述问题，本章首先分析 Hadoop 资源调度的请求机制与模型，详细剖析了资源调度中存在的问题，在此基础上，引入蚁群算法和粒子群算法，实现 Hadoop 资源的合理化分配。
7. cpuSpeedi为节点i上的CPU运行速率，节点i的总资源量用Mi表示，resMi为节点i的空闲资源量，resMi≤Mi，0≤i＜n，loadi表示节点i的负载。作业集J={J1,J2,…,Js}表示当前集群上运行的作业量为s，第j个作业Jj={resTotalJj,resMJj,progj},0≤j＜s，其中，resTotalJj为Jj所需资源的总量；resMJj为分配给Jj的实际资源量；progj表示Jj的运行进度。容器集C={C1,C2,…,Ct}表示集群中所有申请了资源的容器集合，第m个容器Cm={Jmj,resMCm},0≤j＜s,0≤m＜t，Jmj为Cm所属的作业，resMCm为容器Cm所申请的资源量。
对容器进行资源分配时，对任意Ri∈R，Cm∈C，如果资源Ri可以满足容器Cm所申请的资源量，
8. 集群可以正常运行。本论文规定：当算法满足以下两个约束条件时，单个布谷鸟的一次迭代过程视为结束。
9. Hadoop YARN资源调度框架中，AM负责作业启动与监控，RM负责集群中所有资源的管理和分配，并监控、管理各节点上AM的状态。AM与RM之间的通信通过心跳传输机制实现。本章设计并实现了一个资源调度器，通过该资源调度器从NM获取节点CPU速率、作业失败记录、内存容量及负载情况等信息。
在
10. 当RM收到来自NM发送的心跳信息后，将向CapacityScheduler发送一个Scheduler EventType的NODE_UPDATE事件。调度器接收到NODE_UPDATE事件后，首先释放已成功或失败的Container。如果此时调度器有资源空闲，调度器就会尝试在这个节点上按照调度分配策略响应应用程序提交的Resource Request。
11. 在Hadoop YARN中，调度器通过汇集各个应用程序的资源请求进行资源分配。集群初始化时，程序从状态机rmNode中获得相关信息，然后不断维护节点上的SchedulerNode以更新集群资源情况，并确认是否能在此节点上分配资源。当一个Container在节点上启动时，SchedulerNode类会调整与可分配资源相关的内部属性信息值，并记录这个Container的情况。
12. 调度器通过维护SchedulerApplication以监控集群资源的分配，当某个应用程序过度占用资源时，调度器会通过心跳信息通知应用程序释放Container，如果应用程序仍不释放资源，调度器将请求RM杀死此Container。同时，SchedulerApplication中包含了此应用程序提交的ResoureRequest，在每次allocate接口被调用时，AM发来的程序请求会被更新到SchedulerApplication中，以供调度算法在调度时进行计算。
10.3.2 关键接口的实现
资源调度器是RM的一个核心部件，通过RM的组件和服务与其他组件进行交互。
13. 在资源调度过程中，GcsSchedulerNode会根据分配或释放Container情况动态更新节点资源情况，使调度器能够正确显示集群资源情况。
14. 在调度器进行资源分配时，获取应用程序中的最新资源请求信息以更新数据结构，为下一步资源分配提高数据参考，从

而避免因资源请求超出节点容量而造成的资源分配失败。GcsSchedulerNode与GcsSchedulerApp均以映射表Map的形式存储在调度器中，以便在资源分配时供调度器查阅。

15. 实现了调度器与RM的其他组件交互的接口，主要包括handle函数和allocate函数。
16. allocate函数主要用于ApplicationMasterService的调用，通过该函数，调度器汇集从AM发来的各种请求，然后根据这些信息进行相应操作，如释放Container，更新ResourceRequest，把节点从黑名单中加入或者移除等。
17. 节点的CPU频率、内存、负载及容器所属作业在节点上失败率等因素，做到了对集群性能较为全面的评估。以此为前提，

12. 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第12部分 总字数：2448

相似文献列表 文字复制比：8.5%(209) 疑似剽窃观点：(0)

1	Hadoop YARN资源分配与调度的研究 李媛祯(导师：杨群) -《南京航空航天大学硕士论文》- 2015-01-01	8.5%（209） 是否引证：否

原文内容 红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第十一章测试

11.1 测试环境

实验环境为8台计算机构成的Hadoop异构集群，操作系统为CentOS7.0，JAVA版本为JDK7，Hadoop版本为2.7.2。每台计算机的CPU和可用内存如下表5-1所示：

表5-1 Hadoop集群各个计算节点的资源状况

PC1 PC2 PC3 PC4 PC5 PC6 PC7 PC8

CPU频率(GHz) 1.6 1.7 1.9 2.1 2.1 2.1 2.8 3.0

核心数 2 2 2 4 4 4 8 8

内存(GB) 4 4 8 4 8 8 16 16

11.2 测试结果及分析

本论文通过分别选取一个数据密集型作业和计算密集型作业对应用混合遗传布谷鸟算法资源调度的Hadoop集群进行测试，并与最常用的Hadoop YARN默认调度器之一Capacity 调度器进行对比。其中数据密集型选取字数统计WordCount作为测试作业，计算密集型作业选取蒙卡罗特方法估算圆周率PiEsTamator作为测试作业。

实验一：选取Hadoop自带的数据密集型作业测试用例WordCount。WordCount是字数统计测试用例，是一个典型的数据密集型作业集。实验的测试数据采用了一组2015年7月的网络新闻汇总，数据来源为数据堂网站公开分享。本实验将数据进行分词处理后上传到Hadoop文件系统HDFS。在试验测试中分别同时提交8个和16个WordCount作业，每个作业设置为处理一天的数据信息。每一天的信息数据由至少300个txt文档组成，每个文档对应一个Map任务进行处理。因此8个作业构成的作业集中的Map数量要超过2400，此时集群已不可能同时全部进行调度执行，集群负载处于超额状态。分别将作业集提交运行20次的最终测试结果如下表5-2所示：

表5-2 不同规模WordCount作业集在集群上的运行时间

作业集(WordCount) 算法平均时间(s) 最优时间(s) 最差时间(s)

8个 GCS 624 553 689

Capacity 679 603 768

16个 GCS 1219 1127 1295

Capacity 1385 1280 1478

由表5-2可知：使用混合遗传布谷鸟算法（GCS）的集群的作业集执行时间相比使用Capacity默认调度器的集群有了显著的缩短，从而可以证明在集群满负载运行的情况下，本论文提出的混合遗传布谷鸟算法能够大幅提高集群的效率。

实验二：选取测试用例蒙卡罗特方法估计圆周率PiEstamator。蒙卡罗特方法是一种利用随机模拟和统计试验的方法，是一种典型的计算密集型作业。测试作业集N * M1 * M2表示作业集由N个作业组成，每个作业设置M1个Map数量，每个Map中模拟投掷的点数为M2个。最终测试结果如下表5-3所示：

表5-3 不同规模PiEstamator作业集在集群上的运行时间

作业集(N*M1*M2) 算法平均时间(s) 最优时间(s) 最差时间(s)

4 * 25 * 25 GCS 58.442 49.283 67.184

Capacity 62.594 49.923 72.754

4 * 100 * 100 GCS 189.546 176.235 210.252

Capacity 243.493 192.324 278.312

8 * 100 * 100 GCS 304.127 265.236 345.714

Capacity 385.272 324.416 436.335

8 * 200 * 200 GCS 467.231 412.658 525.247

Capacity 553.755 465.251 614.573

由表5-3可以看出，在集群规模较小的情况下，采用混合遗传布谷鸟算法的集群作业集执行时间与Capacity资源调度算法相

差不大。随着作业集规模的增大，集群的资源分配更为紧张，合理地对系统资源进行分配能有效地降低作业集的执行时间，可以看到采用新算法的集群执行时间有着更为明显的降低，从而可以证明本论文提出的算法应用到Hadoop具有较好的资源调度效果。

上述两种类型的测试结果的表明，本论文提出的混合遗传布谷鸟算法应用到Hadoop，无论对计算密集型作业还是数据密集型作业都可以有效地进行调度执行，并且在表现上均优于Hadoop内置的常用的计算能力调度器，是一种有效的实用的资源分配算法。

11.3 问题与改进

尽管在测试与应用的过程中，应用了混合遗传布谷鸟算法的资源调度器对比Hadoop内置的计算能力调度器有着较为明显的进步，然而本算法也存在一些不足之处。

一、本算法在设计时默认作业优先级等同，多用户使用的情况下也不能针对多用户设定用户优先度。例如在运行一批大型作业的时候，一个用户A需要运行一个紧急限时任务，由于此时不能设置任务优先级，会对A用户造成不好的用户体验。关于作业优先级调度如何与本算法的资源调度器相结合需要进一步的研究与测试。

二、本论文提出的混合遗传布谷鸟算法对系统资源进行分配前，首先要进行布谷鸟算法和遗传算法的迭代操作，这在一定程度上增加了资源调度的成本。具体针对额外成本的检验与测试，需要进一步的研究与实验。

三、本论文对提出的算法仅进行了较为简单的测试，对数据量更大和计算节点更多的使用环境下的表现有待进一步的验证和测试。

11.4 本章小结

数据密集型作业和计算密集型作业是两种典型的Hadoop作业。本章分别通过WordCount和PiEstamator这两个常见的数据密集型和计算密集型作业对应用了新算法的Hadoop集群进行了实验测试。多次的测试结果与同等条件下使用Capacity调度器的集群进行了对比，结果证明采用新算法的Hadoop集群作业集执行时间更短，集群资源的利用率更高，从而说明了新算法的有效性。

第十二章田野

12.1 本文总结

随着互联网的发展，人们的活动越来越依靠网络，由此产生和记录的数据量也越来越大，处理和分析海量的数据成为了各企业的重大需求。Hadoop平台是一个应运而生的开源分布式计算平台，它的高可靠性、良好的扩展性、强大的数据处理能力的特点使其迅速流行开来。在实际应用中，如何提升和改善Hadoop的性能有着重要意义。资源调度是分布式计算平台的一个重要的性能提升点。资源调度是一个典型的NP-Hard问题，很难用传统方法得到最优解。因此许多学者研究如何将群智能算法应用于资源调度，并针对资源调度的特点进行改善。

布谷鸟算法是一种新型的群智能算法。研究表明相比于遗传算法，蚁群算法等群智能算法，布谷鸟算法具有较好的求解能力。然而布谷鸟算法仍然存在些许不足。本论文详细介绍了布谷鸟算法和遗传算法的实现原理和特点，分析了两种群智能算法的优势与不足，提出了一种基于布谷鸟算法的改进算法，即混合遗传布谷鸟算法。针对布谷鸟算法在迭代后期收敛较慢、计算精度不足的缺点，在算法中混合遗传算法，对布谷鸟算法的局部搜索性进行改进。通过标准函数测试，结果表明改进后的混合遗传布谷鸟算法相比布谷鸟算法具有更好的后期收敛性和计算精度。

本论文对Hadoop YARN的资源管理方式和调度算法进行了深入的研究，详细分析和阐述了Hadoop内置资源调度算法的实现原理、使用场景以及不足之处。在深入理解YARN管理计算资源的基础上，将混合遗传布谷鸟算法应用到Hadoop资源调度。针对算法的特点，建立资源分配的模型，对计算机节点资源和任务的申请需求进行合适的编码，以便于进行布谷鸟寻巢和遗传的迭代。在实际应用的过程中，加入了对任务优先程度的考虑，以免出现资源需求较大的任务“饿死”的现象。最后搭建Hadoop集群，通过实验测试验证了使用混合遗传布谷鸟算法的Hadoop集群相比默认的Capacity调度器有更好的表现。

12.2 研究展望

基于对布谷鸟算法的研究，本论文提出了一个优化后的混合遗传布谷鸟算法。针对Hadoop YARN资源调度的特点，建立了模型进行研究和分析，最终将改进的算法应用到Hadoop资源调度，从而优化了作业集的执行时间，提高了集群的系统性能。但是本论文的研究仍有可以完善和拓展的方向。

一、本论文提出的改进后的群智能算法，即混合遗传布谷鸟算法，虽然相比布谷鸟算法有了一些进步，但是仍有可以改进的空间。例如对参数的选取，仍然可以继续研究和优化。

二、基于混合遗传布谷鸟算法的资源调度算法在进行资源分配操作前，首先需要对布谷鸟寻巢和基因遗传进行多次的迭代，对于作业集的执行是一种额外的开销，在一定程度上延长了作业集的执行时间。后续应该对如何减少迭代的时间消耗，提高有限时间内的资源分配效率进行进一步的研究。

三、由于硬件条件的限制，本论文的最终实验集群节点较少，新的资源调度算法在更大规模集群上的有效性需要进一步测试与验证。

14. 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第14部分 总字数：8600

相似文献列表 文字复制比：43.6%(3749) 疑似剽窃观点：(0)

1	基于操作码序列的静态恶意代码检测方法的研究 卢占军(导师：丁宇新) -《哈尔滨工业大学硕士论文》- 2012-12-01	35.6%（3063） 是否引证：否
2	10S051019-卢占军-丁宇新 卢占军 -《学术论文联合比对库》- 2012-12-12	34.9%（2998） 是否引证：否
3	BH2009925453 -《学术论文联合比对库》- 2012-12-11	34.9%（2998） 是否引证：否
4	BH2009921397 -《学术论文联合比对库》- 2012-12-04	34.7%（2983） 是否引证：否
5	11S051036-陈晨-丁宇新 陈晨 -《学术论文联合比对库》- 2013-11-27	26.7%（2295） 是否引证：否
6	11S051036-陈晨-丁宇新 陈晨 -《学术论文联合比对库》- 2013-11-14	26.7%（2295） 是否引证：否
7	基于操作码行为深度学习的恶意代码检测方法 陈晨(导师：丁宇新) -《哈尔滨工业大学硕士论文》- 2013-12-01	26.5%（2276） 是否引证：否
8	基于动态污点分析的恶意代码行为依赖图挖掘技术的研究与实现 尤作赛(导师：王勇军) -《国防科学技术大学硕士论文》- 2012-09-01	2.7%（230） 是否引证：否
9	姓 名 莫冬春 学 号 201317170102 班 级 计网1301 专 业 计算机网络技术... -《互联网文档资源（http://www.worlduc.c）》- 2016	1.5%（126） 是否引证：否
10	恶意代码行为挖掘关键技术研究 解培岱(导师：卢锡城) -《国防科学技术大学博士论文》- 2013-10-01	1.1%（96） 是否引证：否
11	基于滑动窗口的数据流预测聚集查询处理的研究 肖裕权(导师：周肆清) -《中南大学硕士论文》- 2011-06-30	0.9%（80） 是否引证：否
12	基于语义的Deep Web数据源分类研究 刘伟平(导师：王宇平) -《西安电子科技大学硕士论文》- 2012-01-01	0.6%（53） 是否引证：否
13	TEA+MEA混合胺液脱除天然气中CO_2吸收性能 唐建峰;青霞;张新军;徐明海;黄彬;李晶;史泽林;杨帆; -《天然气化工(C1化学与化工)》- 2015-10-14 1	0.4%（35） 是否引证：否
14	太阳能—地源热泵系统运行特性实验研究 高源(导师：李素芬) -《大连理工大学硕士论文》- 2014-05-01	0.4%（34） 是否引证：否
15	足部三维重构的关键技术研究 田苗苗(导师：杨秋翔;刘磊) -《中北大学硕士论文》- 2013-05-25	0.4%（31） 是否引证：否
16	基于程序语义的静态恶意代码检测系统的研究与实现 袁雪冰(导师：丁宇新) -《哈尔滨工业大学硕士论文》- 2009-12-01	0.4%（31） 是否引证：否
17	可控僵尸网络模拟平台的研究与实现 丁澄天(导师：刘贵松) -《电子科技大学硕士论文》- 2012-03-01	0.3%（29） 是否引证：否

原文内容 红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第十三章朱鹏博

计算机的普及以及互联网技术的飞速发展，带给人们方便的同时，也附带着各种各样的威胁。如今计算机信息安全面临着巨大的挑战，各种网络安全事件频发，给个人、企业甚至政府机关带来了巨大经济损失。在众多的攻击事件中，恶意代码是其中的主要的攻击手段。

随着信息共享技术的发展及使用，恶意代码的发展速度和更新频率变得越来越快，各种未知恶意代码层出不穷，给分析人

员带来巨大的挑战。一般来说，恶意代码分析技术分为静态分析和动态分析，静态分析技术从语法、语义的层面去理解程序的行为，以期望获取程序在运行过程中的信息，而不需要运行程序。动态分析技术指在可控环境下实际运行程序，监控执行过程中的程序行为，记录程序执行的信息。传统的基于特征匹配的检测方法对于各种变形的恶意代码显得无能为力。随着机器学习和数据挖掘技术的快速发展，已经有许多研究将这些技术用于恶意代码的检测，并且取的了不错的效果。如何应对恶意代码的爆发式增长，尤其是对未知恶意代码及其变种得到较好的检测效果成为恶意代码检测技术的研究重点。

本文对基于机器学习算法的恶意代码检测方法做出了改进，并根据该方法设计并实现了一个恶意代码检测系统。与传统方法不同的是，本文提供多于一种的抽象化方式，接着使用Eclat算法对中间码特征序列进行频繁项集分析，根据分析结果在多种中间码特征序列中做出选择，得到对分类效果最为明显的中间码序列，并根据此中间码序列生成概率矩阵，然后以此概率矩阵作为代表恶意代码的特征。首先，对恶意代码样本进行查壳和脱壳处理。其次，对恶意代码样本进行反汇编处理，得到样本的汇编文本，并从中提取出汇编操作码序列；然后根据定义的多种抽象化方式分别对汇编操作码序列进行抽象处理，分别得到各抽象方式对应中间码序列；接着，使用n-gram算法获得中间码特征序列，并使用Eclat算法对中间码特征序列的频繁项集进行分析，分析出对分类效果最明显的中间码特征序列，并依据该中间码特征序列生成概率矩阵作为样本特征；最后，使用机器学习算法实现恶意代码的检测。在实验中，通过准确率、精确率和召回率三个指标对实验结果进行评估。通过实验结果的对比分析，本文使用的方法相比传统方法有一定的优越性。

飞速发展的互联网技术推动人类社会的不断进步，但凡事皆有利弊，互联网技术的不断革新也促进了恶意代码的发展壮大，其危害也变得越来越来越难以控制。在人们所熟知的恶意代码中，尤以木马、病毒、蠕虫等恶意代码的传播最为广泛,社会各界深受其害，给个人、企业甚至政府造成了难以估量的损失。根据国内知名互联网安全厂商奇虎360发布的《2016年中国互联网安全报告》显示,仅2016年被确认检测到的就有1.9亿多的新增PC端恶意程序，可以预测实际新增的恶意程序将会远远大于这个数字，足以说明互联网安全问题已迫在眉睫。近年来，敲诈者病毒也频繁爆发，仅在中国国内就发生了两次规模较大的传播,遭受该病毒攻击的电脑用户至少达到了490多万，并且大部分受害者并不清楚何时感染的以及如何被感染的。相比于PC端，移动端的情况虽然好了很多但也不容乐观，仅2016年就截获了1400多万针对Android平台的新增恶意程序，大部分为资源消耗类，但是也出现了大量的勒索类软件，同年大约有17多万新增的勒索类软件，170多万台手机遭受攻击。相比于PC端，用户更习惯在移动端存储个人信息，不法分子也开始将目光聚焦在了截获盗取个人信息上，在截获的此类恶意代码中，窃取短信信息的占67.4%，窃取收集银行信息的占34.8%，窃取联系人信息的占了10.0%，窃取手机通话记录的占了3.7%，有2.0%的窃取了社交软件聊天记录。在信息共享技术高度发达的今天，人们对于个人隐私的保护也越来越关注。以下总结了2017年上半年发生的典型安全事件：

（1）2017年上半年爆发了多次攻击事件，比较知名的有“WannaCry”、“暗云III”、“Petya”等多种类型的病毒木马，为社会各界再次敲响了互联网安全的警钟。根据相关报告显示,仅在2017年上半年,就有超过10亿次的PC端病毒被腾讯安全反病毒实验室拦截，同2016年下半年相比增长了30%。其中，在“WannaCry”的刺激下，勒索类病毒增长明显，在一个季度之内就增长了13.39%，而最具传染力的勒索病毒“PolyRansom”占了所有勒索病毒的78.84%。

（2）2017年3月下旬，苹果公司遭受网络犯罪团伙的勒索，该犯罪团伙声称掌握了3亿多的用户账号信息，并且可以远程清除iCloud账户。

（3）2017年4月中旬，据外国的一家媒体hackread报道，1亿多条优酷账户信息的数据库在暗网被售卖。

（4）2017年4月下旬，有媒体报道12306官方网站再现安全漏洞，用户个人信息遭到泄露。

（5）2017年5月中旬，国务院某款App的页面被植入了色情广告，经过排查确定为当地运营商HTTP被劫持导致H5页面被插入广告。

在众多的互联网安全事件中,尤以恶意代码最具威胁性,其带给社会各界的经济损失占很大比例，主要原因有三点。第一，技术较为成熟，恶意代码的出现可以追溯到上世纪八十年代初期，经过几十年的不断壮大与技术积累，恶意代码变得越来越复杂，其破坏力也不断增长；第二，犯罪成本低，由于信息共享技术的发展，恶意代码的开发工具也趋于自动化，并且极易获得，开发变得越来越方便，而传播方式也由原来的被动传播进化为主动传播，使得用户极易被感染，据统计80%以上的用户曾遭受过恶意代码的侵袭。第三，获利方便，恶意代码出现的初衷是为了技术炫耀，如今在利益的驱动下，更多的恶意代码作者有了经济利益诉求，促进了恶意代码的发展，这也是勒索软件出现的原因。现如今,提高互联网安全已经成为国家的一项重要基本战略。

很大一部分网络安全事件发生都是由恶意代码引起的,并且根据以往案例来看,往往都是恶意代码造成一定的损失之后,针对该恶意代码的分析及检测技术才会被提出。出现这种情况的原因无非有两个:首先是恶意代码越来越复杂,并且种类繁多,传播形式多种多样,使得用户很容易被感染;其次是恶意代码检测技术不够成熟,尽管很多学者为恶意代码检测做了很多研究,也提出了很多检测方法,但是理论到应用总是需要时间,而这些时间给了恶意代码去进一步变异的可能。此消彼长下,恶意代码的危害始终存在,而检测与反检测的斗争不会停止。

近年来,恶意代码的使用与各种各样的经济甚至政治利益互相牵扯,其危害性和隐藏性日益增强,破坏的目标、目的以及要达成的后果更加具有针对性。现如今的恶意代码编写者已经不单单是为了技术炫耀，更多的恶意代码的编写者开始利用自己的技术，危害他人以达到经济或者政治诉求。据统计，中国木马产业链一年的收入已逾上百亿元,黑色产业链正在逐渐成型。研究出一种可以对恶意代码进行有效检测的方法已经迫在眉睫。

综上所述,恶意代码的危害无处不在,不仅给个人、企业带来了巨大的损失,甚至可能给国家安全带来不可预期的危害。对于个

人,恶意代码的入侵会导致个人隐私暴露在开放的互联网环境下，这些信息可能被犯罪分子利用，造成个人的经济损失或名誉损失；对于企业，商业机密的泄露，如果是核心技术外泄，那么企业的竞争力将会大打折扣，对发展造成不良影响；对于国家而言，信息时代背景下，信息安全作为国家安全的重要组成部分，其扮演的角色也越来越重要，信息安全已经成为了国家的一项重要发展战略。因此研究更加有效的恶意代码检测技术是非常具有现实意义的。

上世纪80年代，Apple II[2]作为真正意义上的第一个病毒诞生了，自此拉开了恶意代码检测与反检测的战争序幕，众多的学者开始投入大量的精力和时间到恶意代码的对抗中。矛与盾从来都是相互促进的，恶意代码的发展同样促进了检测技术的进步，目前已经有各种各样的检测技术被应用在了商业环境中。Sung[3]等人提出了一种针对恶意代码变种的静态检测技术，该方法的主要是基于系统调用来检测恶意代码的，首先将恶意代码通过逆向工程技术得到其汇编文本文件，然后通过分析汇编文本信息，得到系统调用序列，最后根据序列相似度判断该程序是否是恶意的。张波云[4]等人也提出了相似的检测方法，不同的是该方法在提取系统调用信息时程序处于运行状态，一般是通过在虚拟环境中执行恶意代码程序，提取系统的调用序列，然后使用n-gram算法提取调用序列特征，在检测恶意代码时程序处于运行状态的分析方法被称为动态分析方法。对于恶意代码的分析方法，恶意代码从来都不缺少应对手段，混淆技术应运而生。同样为了解决混淆技术带来的困惑，有学者提出了基于语义分析的检测方法，该方法试图分析指令运行时的语义，一般的分析方法有符号执行[5]、模型检验[6]、逻辑推理证明等。Cousot.P 和 Cousot.R[7]等人提出了程序分析构造和逼近不动点语义理论，使得程序的语义分析有了理论依据。M.Christodorescu[8]使用行为自动机表述恶意代码，并引入了一种抽象模式库来表示自动机的符号，最终恶意代码将会被表达为库中符号表示的自动机，最后使用模型检验实现预测。之后他又提出了一种基于语义的恶意代码检测方法，恶意代码的行为通过迹语义来描述，采用抽象解释方法检测恶意代码的行为[9]。D.Preda[10]同样使用了抽象解释的思想对恶意代码进行检测，再度证明了该方法对于混淆技术具有一定的对抗能力。Singh[11]通过分析存在于汇编文本中的数据流信息，利用线性时态逻辑语义模型检测程序是否存在恶意行为。Kinder[12]结合程序流程图和函数之间的调用关系，使用计算逻辑树描述恶意代码。李佳静等人[13]通过分析函数调用及其调用序列之间的依赖关系，使用有穷自动机描述程序的行为信息。王晓洁和王海峰[14]通过语义描述恶意代码的行为，实验证明该方法对使用了混淆技术的恶意代码也具有相当明显的检测效果。孔德光等人[15]提出了一种签名提取算法，对于恶意代码检测的准确率有了较大的提高。G.Tahan等人[16]提出了一种自动签名提取算法，该算法主要被应用在高速恶意代码的过滤装置中。Y.Tang等人[17]提出了一种基于正则表达式的签名算法，这种方法能否更加准确的产生基于漏洞的签名。Y.Chen[18]等人提出了在网络层没有任何主机分析的蠕虫执行的脆弱性驱动的签名，实验效果非常好。现有的基于签名的恶意代码检测技术通过特殊的字符串特征来判断，其准确率非常高，但是其缺点是不能检测新出现的恶意代码，并且需要不断的更新特征库。现在大部分研究用基于 n-gram 序列的字节序列代替二进制特征码序列，这会提高分类的准确率。Robert[19]等人提出了用操作码序列作为特征，然后使用文本分类的方法实现检测，并解决了数据不平衡问题[20]。Schultz[21]等人第一次提出了应用数据挖掘模型来检测恶意代码，他们提取三种特征并使用不同的分类方法：程序的头文件信息，字符串信息，字节的序列，应用基于签名、基于规则的学习器 Pipper、朴素贝叶斯等方法进行分类。 研究表明使用机器学习方法能提高准确率。后来 Kolter[22]使用 n-gram 算法提取字节序列作为特征，改进的决策树算法取得了很好的分类效果。 在参考文献[23]中，作者提出了使用 n-gram 算法提取特征后使用信息增益的方法来选择一些分类效果好的特征，并使用 K 近邻，基于 TFIDF 的分类器、 朴素贝叶斯、支持向量机、决策树等分类方法，并取得了很好的实验效果。 Kolterh 和 Maloof[22]研究了恶意代码的家族的分类，基于恶意代码的功能行为，使用多分类方法将恶意代码分为蠕虫、木马、后门、病毒等，这更加细化了分类的结果，有助于对每一种恶意代码的研究，发现它们的共性，这也为以后的语义分析等方法奠定了基础。文献[24]中作者提出了一个层次特征选择的方法，即使用 n-gram 算法提取特征后选择那些出现频率高于某个阈值的特征，这种方法对于检测恶意代码的变种很有效。Raja[25]等人应用数据挖掘方法实现恶意代码的检测，他通过反汇编技术提取了恶意代码的操作码序列，使用了一种新的在文本分类领域的特征选择方法 CPD（Categorical Proportional Difference）。CPD 用来度量一个特征的区分能力，最终分类效果相对比较好。Dolev[26]提倡使用操作码来作为恶意代码的中间表示。操作码是机器语言的一个操作的一部分，它包含着指令的行为和程序的控制。近年来，操作码特征已经被用来检测蠕虫的变种和一些间谍软件[27]。将操作码提取出来作为标签，然后产生签名来判别恶意代码的变种。后来有些学者提取操作码并将其转化成操作码序列来检测未知的恶意代码[19]，实验使用三种分类算法取得了很好的实验效果。在文献[28]中，作者提出了使用变长的指令序列作为特征，并使用 Bagging算法得到了很好的实验效果。也有人使用了十六进制码作为特征[29]。在恶意代码检测技术中使用操作码序列作为特征的研究相对还是比较少的，但是研究结果发现操作码序列是一种比较好的特征表示方法。 在文献[30]中，作者使用程序的控制流程图并用三种不同版本的黑客防御工具设计了一个分类算法，并取得了很好的实验效果。Ismail B[31]提出了将程序控制流程图和函数调用拓扑用于将未知的恶意代码归类。使用函数调用拓扑的缺点是，攻击者能使用相似的函数调用或者改变函数调用的序列来逃避检测。 Halvar[32]利用程序控制流程的拓扑图的同构来实现检测。因为同一种族的恶意代码的拓扑图基本相似，这种方法也是适合检测恶意代码的变种。Igor[33]提出了一个新的检测未知恶意代码族的方法。该方法是基于操作码序列的出现的频率，并挖掘了每一个操作码序列的相关性。通过大量实验对比分析，该方法是非常有效的。Perdisci[34]等人提出了从PE文件提取一些特征，如标准和非标准部分的数目，可执行部分的数量以及 PE 头文件的熵信息，并使用不同的机器学习模型实现分类。后来他们开发了一个快速统计恶意代码的检测工具[35]。

综上所述，现有的恶意代码检测技术有很多，每一种方法都有自身的优缺点。这为后面的研究提供了基础的同时也带来了很多挑战。本文提出了一种新的恶意代码检测方法，结合了特征码、行为及机器学习的方法，提出了基于操作码序列的静态恶意代码检测方法，能更好的检测恶意代码。

本文改进了一种基于机器学习算法的恶意代码检测方法，并基于该方法实现了一个恶意代码检测系统。方法主要是使用汇编操作码的抽象化技术将汇编码表示为中间码，并结合Eclat算法对中间码的频繁项集进行分析，预测出一种最优的中间码序列，然后使用n-gram算法提取中间码特征序列生成概率矩阵，作为代表恶意代码的特征，最后使用机器学习算法进行建模。当抽象方式有多种的时候，这种方法能够针对n-gram算法提取出的中间码特征序列，预测出一种对分类效果最好中间码特征序列作为生成概率矩阵的依据。

在仿真实验中，本文提供了两种抽象方式，分别记为Abs1和Abs2，Abs1是根据作者的理解对汇编码提出的抽象方式，Abs2是文献[1]中提出了抽象方式，本文将会分别使用本文方法和文献方法进行实验并分析对比实验结果，给出结论。实验主要步骤有：首先，为了逃避病毒检测系统的检测，一般的恶意代码作者都会对恶意程序进行加壳处理，所以本文的第一步就是对恶意代码进行查壳和脱壳处理；其次对恶意代码样本进行反汇编处理，得到样本的汇编文本，并从中提取出汇编操作码序列；然后根据已配置的抽象方式对汇编操作码序列进行抽象化处理，得到各抽象方式对应的中间码序列；接着使用n-gram算法获得中间码特征序列，并对中间码特征序列的频繁项集进行分析，预测对分类效果最明显的中间码特征序列，并依据选择出的中间码特征序列生成概率矩阵；最后对比了随机森林算法、支持向量机以及K邻近三种机器学习算法使用概率矩阵作为输入的分类效果，结果显示随机森林算法效果最为显著和稳定。另外根据本文提出的方法设计并实现了一个恶意代码检测系统。本文的工作有以下几点：

第一：收集实验样本，并对样本进行预处理操作，然后提取基于文本的汇编操作码序列。

第二：本文改进了一种基于机器学习算法的恶意代码检测方法。通过汇编操作码抽象化技术、Eclat算法频繁项集分析和n-gram算法提取特征序列获得概率矩阵，并以此作为代表恶意代码的特征，最后使用机器学习算法构建分类模型。

第三：对本文提出的方法进行实验仿真，并对实验结果进行对比分析，得出结论。

第四：结合实验仿真的结果分析，针对本文提出的改进方法，采用模块化编程技术实现了一个恶意代码检测系统。

本文总共分为四章：

第一章为绪论，主要介绍课题背景及意义。并且详细阐述了国内外对于恶意代码检测技术的研究现状和存在的问题，最后介绍了本文的研究内容和章节安排。

第二章是相关理论与关键技术。首先介绍了恶意代码的定义和分类，然后对现有恶意代码分析技术、检测技术和反检测技术做了相关介绍。

第三章是基于机器学习算法的恶意代码检测方法。首先对汇编操作码的特点进行了分析，并介绍了有关学者对汇编码抽象化的一些研究；然后概要的介绍了本文提出方法的大体流程，接着对方法的各部分进行了详细说明，包括数据预处理、概率矩阵的生成过程和恶意代码的分类；最后根据本文方法进行了实验仿真，并和传统方法的实验结果进行了对比分析，得出结论。

第四章是系统设计与实现。首先介绍整个系统的架构设计，包括系统的功能分析及组成模块，并对各模块的功能进行了简单介绍；接着对各模块的实现方式进行了详细介绍；最后对本章进行总结。

第五章为总结与展望。概括总结了本文的主要研究成果和不足，对未来的可研究方向进行了展望。

本章首先介绍了恶意代码相关概念，并对现有的恶意代码检测技术以及反检测技术进行了详细介绍；然后对恶意代码的反检测技术做了相关介绍；最后对恶意代码的分析技术做了总结。

恶意代码也成为恶意软件，是对各种敌对和入侵软件的概括性术语。包括各种形式的计算机病毒、蠕虫、特洛伊木马、勒索软件、间谍软件、广告软件以及其他的恶意软件。形式上多种多样，可以是可执行文件、脚本、插件等等。其违背使用者的意愿去执行一些操作，损害用户的利益以达到入侵者不可告人的目的。

根据不同的依据，恶意代码有很多种不同的分类方法，没有一种标准的分法，但是常见的种类有：计算机病毒、蠕虫、特洛伊木马、间谍软件、勒索软件等等。下面对几种恶意代码做简要介绍：

(1)计算机病毒。病毒是早期产生的最主要的恶意代码之一，病毒是能够自我繁殖并寄生在其他程序中的代码，这个被寄存的程序被称为宿主程序，但是病毒不能单独运行，必须通过激活宿主程序并满足一定条件下，病毒就能干扰电脑正常工作，扰乱或破坏已有存储的信息，甚至引起整个系统不能正常工作。一般而言计算机病毒通常由三个单元和一个标志构成：引导模块、感染模块、破坏表现模块和感染指标。1、引导模块是指将计算机病毒感染的宿主程序设法引导安装到

计算机操作系统中，为以后的感染、破坏两个后期模块提供前期的有效准备，一般而言不同的计算机病毒有不同的引导操作，而且引导操作往往是隐蔽的，不易被用户察觉和发现的。2、感染模块包括两个部分，一个是用来激活感染功能的判断部分。该模块提供一个感染

的标志，用来判断计算机是否被感染。另一个是执行感染功能部分。这一部分主要的功能就是监控宿主满足条件的时机，并及时的将计算机病毒存入到系统特定的位置。3、破坏表现模块与感染模块一样包括两个部分，一是具有触发破坏表现功能的判断部分。二是具有破坏表现功能的实施部分。计算机病毒一般具有寄生性、传染性、隐藏性、破坏性、潜伏性等特征。

（2）特洛伊木马。木马分为客户端和服务端，客户端安装在攻击者的主机上是控制端，服务端安装在受害者的机器上。木马可以使攻击者远程控制受害者的主机，造成受害者信息丢失等问题。木马有很好的隐蔽性，通过模仿正常的系统文件命名、与其他程序绑定、进程注入及拦截系统调用的方法伪装自己。木马也有很好的自启动性和自恢复性。常见木马有远程访问型木马、键盘记录型木马、密码发送型木马、FTP 型木马以及破坏型木马等。

（3）蠕虫。蠕虫是一种可以独立运行、自我复制及自动传播的恶意程序。它通过网络、共享文件、电子邮件、移动存储设备以及有漏洞的主机等自我复制和传播。蠕虫的传播速度非常快，根据它的危害性可以简单分为无害型、消耗型和破坏型。无

害型蠕虫感染主机后会产生很多垃圾文件减少系统的可用空间；消耗型蠕虫感染主机后，发送大量扫描数据包，消耗主机的 CPU 和内存资源，与此同时增加了网络的负载，降低网络的性能；破坏型蠕虫感染主机后会删除和破坏程序和文件，有时会泄露一些重要信息。

（4）后门。它是一种运行在目标系统中，能够绕过安全控制机制获得对系统的访问权，为攻击者提供通道的恶意代码。

指　标

疑似剽窃观点

---

1. 综上所述，现有的恶意代码检测技术有很多，每一种方法都有自身的优缺点。

疑似剽窃文字表述

---

1. 恶意代码层出不穷，给分析人员带来巨大的挑战。一般来说，恶意代码分析技术分为静态分析和动态分析，
2. 恶意代码及其变种得到较好的检测效果成为恶意代码检测技术的研究重点。
   本文对基于
3. 查壳和脱壳处理。其次，对恶意代码样本进行反汇编处理，得到样本的汇编文本，并从中提取出汇编操作码序列；
4. 恶意代码的发展同样促进了检测技术的进步，目前已经有各种各样的检测技术被应用在了商业环境中。
5. 现有的基于签名的恶意代码检测技术通过特殊的字符串特征来判断，其准确率非常高，但是其缺点是不能检测新出现的恶意代码，并且需要不断的更新特征库。现在大部分研究用基于 n-gram 序列的字节序列代替二进制特征码序列，这会提高分类的准确率。
6. 在恶意代码检测技术中使用操作码序列作为特征的研究相对还是比较少的，但是研究结果发现操作码序列是一种比较好的特征表示方法。
7. 使用函数调用拓扑的缺点是，攻击者能使用相似的函数调用或者改变函数调用的序列来逃避检测。
8. 该方法是基于操作码序列的出现的频率，并挖掘了每一个操作码序列的相关性。通过大量实验对比分析，该方法是非常有效的。
9. 这为后面的研究提供了基础的同时也带来了很多挑战。本文提出了一种新的恶意代码检测方法，结合了特征码、行为及机器学习的方法，提出了基于操作码序列的静态恶意代码检测方法，能更好的检测恶意代码。
   本文改进了一种基于机器学习算法的恶意代码检测方法，
10. 本文的第一步就是对恶意代码进行查壳和脱壳处理；其次对恶意代码样本进行反汇编处理，得到样本的汇编文本，并从中提取出汇编操作码序列；
11. 进行实验仿真，并对实验结果进行对比分析，得出结论。
    第四：结合实验仿真的结果分析，针对本文提出的改进方法，
12. 第一章为绪论，主要介绍课题背景及意义。并且详细阐述了国内外对于恶意代码检测技术的研究现状和存在的问题，最后介绍了本文的研究内容和章节安排。
    第二章是
13. 根据不同的依据，恶意代码有很多种不同的分类方法，没有一种标准的分法，但是常见的种类有：计算机病毒、蠕虫、特洛伊木马、间谍软件、勒索软件等等。下面对几种恶意代码做简要介绍：
    (1)计算机病毒。
14. 计算机病毒通常由三个单元和一个标志构成：引导模块、感染模块、破坏表现模块和感染指标。1、引导模块是指将计算机病毒感染
15. 满足条件的时机，并及时的将计算机病毒存入到系统特定的位置。3、破坏表现模块与感染模块一样包括两个部分，一是具有触发破坏表现功能的判断部分。
16. 木马分为客户端和服务端，客户端安装在攻击者的主机上是控制端，服务端安装在受害者的机器上。木马可以使攻击者远程控制受害者的主机，造成受害者信息丢失等问题。木马有很好的隐蔽性，通过模仿正常的系统文件命名、与其他程序绑定、进程注入及拦截系统调用的方法伪装自己。木马也有很好的自启动性和自恢复性。常见木马有远程访问型木马、键盘记录型木马、密码发送型木马、FTP 型木马以及破坏型木马等。
17. 蠕虫是一种可以独立运行、自我复制及自动传播的恶意程序。它通过网络、共享文件、电子邮件、移动存储设备以及有漏洞的主机等自我复制和传播。蠕虫的传播速度非常快，根据它的危害性可以简单分为无害型、消耗型和破坏型。无害型蠕虫感染主机后会产生很多垃圾文件减少系统的可用空间；消耗型蠕虫感染主机后，发送大量扫描数据包，消耗主机的 CPU 和内存资源，与此同时增加了网络的负载，降低网络的性能；破坏型蠕虫感染主机后会删除和破坏程序和文件，有时会泄露一些重要信息。
18. 它是一种运行在目标系统中，能够绕过安全控制机制获得对系统的访问权，为攻击者提供通道的恶意代码。

15. 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第15部分　总字数：8875

相似文献列表　文字复制比：57.8%(5132)　疑似剽窃观点：(0)

1	基于操作码序列的静态恶意代码检测方法的研究	26.0%（2307）
	卢占军(导师：丁宇新) -《哈尔滨工业大学硕士论文》- 2012-12-01	是否引证：否

2	10S051019-卢占军-丁宇新 卢占军 - 《学术论文联合比对库》- 2012-12-12	25.2%（2233） 是否引证：否
3	BH2009925453 - 《学术论文联合比对库》- 2012-12-11	25.2%（2233） 是否引证：否
4	BH2009921397 - 《学术论文联合比对库》- 2012-12-04	24.8%（2204） 是否引证：否
5	恶意代码检测中若干关键技术研究 陈良(导师：李斌;陈斌) - 《扬州大学硕士论文》- 2012-04-01	12.3%（1091） 是否引证：否
6	094616006_冯本慧 冯本慧 - 《学术论文联合比对库》- 2013-09-23	11.4%（1008） 是否引证：否
7	基于数据挖掘与机器学习的恶意代码检测技术研究 冯本慧(导师：王加阳) - 《中南大学硕士论文》- 2013-09-01	10.7%（946） 是否引证：否
8	基于操作码行为深度学习的恶意代码检测方法 陈晨(导师：丁宇新) - 《哈尔滨工业大学硕士论文》- 2013-12-01	9.4%（832） 是否引证：否
9	11S051036-陈晨-丁宇新 陈晨 - 《学术论文联合比对库》- 2013-11-27	8.6%（766） 是否引证：否
10	11S051036-陈晨-丁宇新 陈晨 - 《学术论文联合比对库》- 2013-11-14	8.6%（766） 是否引证：否
11	结合语义的统计机器学习方法在代码安全中应用研究 孔德光(导师：奚宏生) - 《中国科学技术大学博士论文》- 2010-05-01	8.6%（764） 是否引证：否
12	恶意代码行为挖掘关键技术研究 解培岱(导师：卢锡城) - 《国防科学技术大学博士论文》- 2013-10-01	2.0%（181） 是否引证：否
13	浅谈病毒与反病毒 梅净缘; - 《电脑与信息技术》- 2012-04-15	1.4%（124） 是否引证：否
14	液压支架电液控制系统的研究与实现 郭科伟(导师：李志敏) - 《重庆大学硕士论文》- 2012-04-01	0.4%（36） 是否引证：否
15	基于沙箱技术的恶意代码行为自动化检测方法 李志勇(导师：李伟明) - 《华中科技大学硕士论文》- 2015-05-01	0.3%（31） 是否引证：否
16	基于污点跟踪的固件漏洞定位研究 戴忠华;费永康;赵波;王婷; - 《山东大学学报(理学版)》- 2016-04-18 1	0.3%（30） 是否引证：否

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

后门可以使攻击者远程控制目标主机，危害无穷。后门提供的通道有几种类型：本地权限提升、远程命令行访问、单命令远程执行、远程控制等。

（5）Rootkit。它是指帮助攻击者获取主机管理权限后，实现维持拥有管理权限的程序[36]。通常攻击者通过后门获取管理权限，并使用 Rootkit 维持管理权限使的恶意代码能隐藏在目标系统中。Rootkit 分为用户模式和内核模式。用户模式通过通道插入恶意代码、覆盖文件、API 钩子和 DLL 注入等方式达到目的。而内核模式通过安装恶意的设备驱动程序、打补丁、修改内存中运行的内核以及虚拟伪造系统的方式实现。

（6）间谍软件。它是在未授权的情况下窃取用户的信息并通过网络发送给攻击者的一种恶意代码。这种恶意代码不仅仅能泄露目标主机的数据信息，还可以提供恶意代码的植入接口使得被侵系统受到更加严重的破坏。

（7）广告软件。它是指在未经用户授权的情况下和别的程序捆绑在一起，以便经常弹出一些用户不想接受的广告。这种恶意程序目的是通过这种强制的方式做商业宣传。一些广告插件的安装会降低主机的性能。广告软件主要的危害是弹出一些色情或者恶意的广告，这会给用户带来很大的困扰。

（8）恶意网页脚本。它是指在网页中嵌入一些用脚本语言编写的有恶意行为的代码。当用户点击带恶意脚本的网站后，脚本通过修改目标系统的注册表、下载病毒或者加载木马程序等方式对被侵系统实施破坏行为。

目前用于商业的恶意代码检测软件中，一般采用的都是基于“特征码”的检测技术，基本思想是，当新的恶意代码被发现后，对其进行采集取样，分析代码构成，提取有用的特征码，然后将新的特征码加入已有数据库中，用户更新病毒库之后，就会使用新的特征库去匹配恶意代码，如果匹配成功则进行相应的处理。但是特征码检测技术的缺点是只能对已知恶意代码进行有效的将测，对与未知或者稍加变动的恶意代码无能为力。因此，在恶意代码检测领域提出了启发式检测算法来预防和检测新的恶意代码。根据对恶意代码分析原理的不同对现有恶意代码检测方法进行分类，主要分为基于特征码的检测技术、基于行为的检测技术、基于启发式的检测技术、基于语义的检测方法和基于机器学习算法的检测技术等。下面将详细介绍几种检测技术。

（1）基于特征码的检测技术

基于特征码的检测方法是使用最古老和最广泛的方法。被Symantec等多有著名病毒检测厂商所使用，是目前已知的所有恶意代码检测方法中最简单、开销最小的方法，广泛用于文件类型的病毒检测中。检测软件的核心就是恶意代码特征库的完全性，当需要扫描某个程序是否有恶意企图时，启动特征扫描提取特征，然后再与特征库进行匹配，如果匹成功，则判断该程序是

恶意的。此技术的关键在于如何选取最能代表恶意程序的特征值。采用该方法，检验结果准确，鲜有误报情况，但该方法对于未知或者变形恶意代码无能为力。还有，这种方法使得特征库不断增加，这需要用户经常更新特征库，随着时间的流逝，特征库会越来越庞大，这会影响检测的速度和系统的性能。

（2）基于行为的检测技术

基于行为的检测方法是利用恶意代码的特有行为来检测恶意代码的方法。恶意代码的行为有相对的稳定性和已于检测的特点，比如特定的系统调用，恶意代码要完成自身逻辑功能，即完成对系统的入侵和破坏，就必须获取系统非法权限，调用系统的资源[37]，这样通过分析恶意代码的行为就可以方便的分析检测出恶意代码。当程序运行时，监控其行为，如果发现了异常行为，则立即报警。一般用于检测恶意代码的行为特征如下：

1) 对特定文件执行写操作：有些恶意代码时依附而生，所以在其执行时，就要将自身代码附加在感染文件中，可以监控是否有异常写操作。

2) 监控系统调用序列：某些系统调用序列可以体现某种程度的程序语义。系统调用是用户态和内核态的唯一接口，恶意代码想要获取高级权限实施破坏行为，就必然要经过系统调用接口。

3) 修改内存总量：恶意代码为了完成特定的恶意意图，经常会常驻在内存中，并且不能被覆盖，那么将会减少系统内存的总量，使得该段内存不受系统内核控制。

（3）基于启发式的检测技术

启发式检测方法是对恶意代码特征提前设定一个阈值，在对文件进行扫描后，当提取的特征和恶意代码特征的相似度达到一定的值，这认定该文件是恶意代码。例如一些恶意代码都会固定的对一些内核函数进行调用，通常这些调用的顺序是有一定的规律，因此利用对内核函数的名称和调用次数进行分析，可以构建一个恶意代码对内核函数的特征。启发式方法属于主动防御技术，对未知的恶意代码检测具有明显的效果，因此，这种方法在现如今的商业开发被重点应用。启发式检测可分为静态启发和动态启发。

静态启发方法其实是对传统的特征识别方法的一种扩展，通过分析程序对系统API的调用序列作为特征，有领域专家根据自身经验，研究总结出某些恶意代码的行为特征，当对行为进行监控时，此类特征一旦被发现，就立即报警并做相应的处理。这种方法能够有效的检测出已知的恶意代码，并发现部分未知的恶意代码，但在发现恶意代码的时候，系统往往已经被感染。另外，行为检测是对系统进行实时的监控，因此可能会持续占用大量内存、CPU等系统资源。在商业领域中，该方法主要用于辅助性检测。

动态启发式技术主要的工作原理是在计算机系统中划分出一各独立的虚拟环境，当发现可疑程序时，并不立即停止，而是让其继续运行。“沙盒”技术就是动态启发式技术的一种，沙盒会对可疑程序的行为进行记录，直到恶意代码完全暴露后，它在执行回滚操作，使计算机恢复到执行可疑程序之前的状态。近年来病毒检测厂商已经将沙盒技术应用与商业的查杀工具中并进行了推广。

（4）基于语义的检测技术

基于语义的检测技术是现在研究的热点。因为混淆技术只是通过插入垃圾指令、改变指令顺序及寄存器重新分配等方法来改变程序，但是程序的基本语义是等价的。通过分析恶意程序，抽象程序指令的行为并建立其行为模型，使得该模型既描述恶意程序的基本行为，又具有很强的泛化能力。这样因其有很强的泛化能力，使得检测恶意程序的变种更加方便快捷。除此之外，也可以检测未知类型的恶意代码。现阶段基于语义的检测方法分为基于内存和函数调用的方法。M.Christodorescu[8][9]提出了一套抽象理论和语义框架，使用自动机描述程序的行为，通过抽象理论描述程序的行为建立抽象模式库，并将其作为自动机的符号表，最终经恶意行为描述为自动机表述的模板，最后通过模型检测方法检测样本是否含有恶意行为。模型检测是通过遍历系统所有状态空间，看其中是否有一条符合的路径状态。之后，他还提出迹语义这一概念，将迹语义作为程序的基本语义，并定义了等价的条件，通过抽象解释的方法给出了近似的检测算法。抽象解释理论为解决不可判定和复杂问题的逼近求解提供可很好的构造方法。基于函数调用的方法是将程序中使用的函数提取出来，并结合程序的控制流程图，通过图的同构、线性时态逻辑、计算逻辑树、有穷状态机及下推自动机等方法描述恶意行为，最终通过模型检测完成恶意代码的检测。

（5）基于机器学习的检测技术

基于机器学习和数据挖掘的检测方法。随着检测技术的不断发展，机器学习和数据挖掘的方法已经被开始应用在恶意代码检测的领域。主要应用分类、关联规则挖掘、序列模式分析以及聚类等几种技术。主要思想是利用数据挖掘技术从现有的数据中挖掘一些有意义的模式，用机器学习技术归纳出已有样本的特征，然后根据特征的相似性等完成分类的任务。其中，最主要的是选择好的特征和有效的分类器。检测步骤如下：首先，要分析样本确定提取哪种特征或者特征序列；其次，根据特征的特性选择合适的特征选择方法从所有提取的特征中选择一些分类效果好的特征；最后，根据实际情况选择较好的模型实现分类。

恶意代码检测与反检测技术总是相互促进，相辅相成。检测技术的进步也带动了反检测技术的发展，当前出现了各种各样的恶意代码反检测技术，现总结如下：

（1）加壳技术

恶意代码作者为了防止自己的程序被检测软件发现，利用一些软件技术给恶意代码加外壳，可以是利用算法将自己伪装成正常程序，或者利用特殊的算法将自己压缩或加密，使得检测软件很难检测。但是这些“壳”都有一个特点就是，他们先于程序获得执行控制权，然后把伪装后的程序还原，再把执行权交还给原始代码：是一类自修改代码。

（2）反虚拟执行技术

不可否认，虚拟执行的系统和真实系统或多或少存在差异[41]。比如，硬件上，调试器总是会设置硬件断点，而虚拟机总是在模拟硬件，这和真实的硬件是有差别的；执行环境，内核地址空间，对于虚拟机和真正的机器是不同的，还有调试器必须挂靠某些进程来插桩进程用于监控；应用程序，虚拟机和调试器都有外部应用程序，对进程可见，用于检查运行环境。一些指令在虚拟机环境中，执行时间总是远远长于真实环境，一个经常执行此类指令的程序能够指示它在虚拟机环境中运行。

（3）代码迷惑技术

恶意代码迷惑技术是指通过某种程序代码变换，改变自身在空间和时间上的结构，但是完成相同的逻辑功能。恶意代码在进行迷惑处理之后，使得逆向工程分析变得难以进行。迷惑技术本身是一种保护软件的手段，但是常常被用来对抗分析和检测。恶意代码的迷惑技术可以有效的对抗恶意代码的静态分析技术和动态反汇编技术。目前主要有基于加密的迷惑技术和基于代码变换的迷惑术。其中代码变换主要指在源程序中，利用等价指令替换、指令位置交换、添加新指令等手段改变程序形式，但逻辑功能保持不变。

恶意代码分析是确定恶意代码意图的过程，是实行恶意代码检测的必要前提。恶意代码分析的直接结果是用于实现恶意行为建模的元数据信息，如指令流、API调用序列等，为后续恶意代码的检测工作提供必要的支持。

恶意代码的分析技术一般可分为静态分析和动态分析。

（1）静态分析技术

静态分析技术是指对被测软件的源程序或者二进制码进行扫描，从语法、语义的层面去理解程序的行为，以期望获取程序在运行过程中的信息，而不需要运行程序。

要进行恶意代码的静态分析，首先需要对恶意程序进行反汇编，常用的反汇编工具有：W32DASM、objdump、PEid、HIEW、IDA Pro等。

静态分析技术由于不会运行程序，因此不会对计算机系统造成任何伤害，其分析效率相对动态分析而言较高，同时由于静态分析技术从程序本身入手，因此可以获得程序的全部信息，分析结果较为全面。但是由于静态分析技术的前提条件是对程序进行正确的反汇编，现如今很多恶意代码编写者常常会对恶意代码进行加壳、加密或者压缩使得恶意程序很难被正确的反汇编。总之，如果恶意代码无法被正确的反汇编，那么静态分析将会失效。

（2）动态分析技术

动态分析技术是指在可控环境下实际运行程序，监控执行过程中的程序行为，记录程序执行的信息。由于动态分析需要先运行程序，所以为了防止恶意代码对当前环境的破坏，系统在普或者恶意代码相关信息之后，会自动恢复到恶意代码执行前的最初状态，防止影响下一次的分析结果，但是动态分析技术能够获得恶意代码执行时的真实信息，可以有效地解决静态分析中譬如加壳、加密的干扰。

当前最流行的动态分析技术是动态污点分析技术[38]，它的基本原理是将一切不信任的外部数据标记为污点，然后跟踪标记为污点的数据的传播情况，并记录相关的系统调用或者指令执行等相关信息，然后以此信息进行检测。动态污点分析能够记录恶意代码更细粒度的精确特征，是当前非常热门的恶意代码检测技术。

动态分析技术也存在缺陷，比如开销大，一次只能分析一条路径，恶法应对恶意代码存在多路径的问题，同时由于恶法模拟出一个完全真实的计算机环境，对某些环境敏感的恶意代码无法进行有效的检测，因为恶意代码能够检测到虚拟机或者仿真机存在的情况，从而隐藏自身的真实行为，也无法知道某些恶意代码何时才会触发，动态分析技术也会受到行为层的混淆技术的干扰[39]如等价行为替换、模拟序列或者混淆序列等。于是有学者开始尝试静态分析与动态分析相结合[40]的方式进行恶意代码的检测，充分利用两种分析技术的优点。

两种分析技术各有优缺点，静态分析技术开销小，关注的是恶意程序本身的语法或者结构特征，动态分析技术开销大，关注的是恶意代码的行为特征，各有侧重点。静态分析技术分析全面，可获得恶意代码的全部信息，但获取特征的方式一般都是无导向的，因此可能包含大量无用信息，也易受代码迷惑技术的影响，动态分析技术能够获得程序的真实行为信息，但一次只能获得一种行为并且与当前的检测环境相关，信息不够全面。当前应用最为广泛的技术还是静态分析技术。总之，无论是静态分析还是动态分析，都需要借助恶意代码分析技术和监控技术获得恶意代码的基本属性和执行信息，以便深入理解恶意代码的功能，进一步实现恶意代码的检测和抑制。

本章首先介绍了恶意代码的定义以及恶意代码的分类，具体介绍了病毒、特洛伊木马、蠕虫等恶意代码的特征和危害。其次，介绍了恶意代码的检测技术，详细阐述了基于特征码的检测技术、基于行为的检测技术、基于启发式的检测技术、基于语义的检测技术和基于机器学习的检测技术。最后，针对目前主流的恶意代码反检测手段以及分析技术做了详细说明。

本章对汇编操作码的特点进行分析，研究了众多的汇编码抽象方式后，改进了一种基于机器学习算法的恶意代码检测方法。该方法主要包括三个部分：数据预处理、概率矩阵的生成以及恶意代码分类，然后分别对方法的三个部分进行了详细介绍，其中概率矩阵的生成方法是本章的核心，也是本方法与其他传统方法的主要区别。最后对本文方法进行了实验仿真，证明本方法对于恶意代码的检测具有实际意义。

大部分的基于机器学习算法的恶意代码静态检测技术都会使用汇编码进行分类模型构建，汇编操作码有以下特点：

第一，汇编语言是面向机器的程序语言，是一种用文字助记符来表示机器指令的符号语言，它是目前所有编程语言中最接近机器码的一种语言。它的针对性特别强，需要对机器硬件进行精确的控制，所以它的每一条指令都是极致细化的，这也导致了汇编操作码的局限性。当使用n-gram算法时，一般来说n的取值不会太大，n条少量的汇编操作码并不能反映出程序行为的一般特征。如果对汇编操作码进行抽象，得到更高维度抽象的中间码，同样是n条中间码，其反映的行为特征将会更加明确及

有意义，所以对汇编码进行抽象对于恶意代码的检测是存在积极意义的。

第二，将汇编操作码进行抽象化处理能够显著的提升实验的效率。本文使用概率矩阵作为恶意代码的特征，即机器学习的输入，而x86汇编操作码的种类就有100多种，当使用2-gram、3-gram、4-gram算法时，概率矩阵的列数将分别是1002、1003、1004，即使后续对概率矩阵进行降维操作也是非常低效的。

针对以上对汇编操作码特点的分析，Kranz J等人[42]提出了一种名为GDSL的中间码来对x86汇编指令做出抽象，以期望表达更多的程序语义信息，Alam S等人[43]提出了一种被称为MAIL（Malware Analysis Intermediate Language，MAIL）的中间码表示方法，这些方法的确能够很好的归纳汇编指令的特征，但是王冰和方勇[1]认为其精简度不够，所以提出了一种新的指令抽象方式并且应用到了恶意代码检测领域，取得了不错的检测效果。

可以看出对于汇编指令的抽象化方式仁者见仁，很难去判定某种抽象方式绝对优于另一种抽象方式，不过可以肯定的是，任何一种指令抽象方式都有其适用范围，那么如何在复杂的互联网环境中保证检测系统始终选择最适合抽象化方式是本章要解决的问题。

基于机器学习算法的恶意代码检测方法一般流程如图3-1。首先对样本进行预处理，该阶段包括对样本进行查壳脱壳以及对样本反汇编得到其对应的汇编文本；然后对汇编文本直接使用n-gram算法提取特征作为机器学习算法构造预测模型的输入；最后使用预测模型对恶意代码进行预测，得到预测结果。

图3-1 一般算法流程

由上一节分析可知，汇编操作码直接应用于恶意代码的检测时会有一定的局限性，所以本文改进了一种恶意代码的检测方法，主要流程如图3-2。首先是样本预处理，该步骤包括对所有的样本进行查壳脱壳处理，并对脱壳后的样本进行反汇编，得到其汇编文本文件；其次，根据汇编文本文件提取汇编操作码，并根据不同的抽象方式对汇编操作码进行抽象，得到对应的中间码序列；然后使用n-gram算法提取样本的中间码特征，接着对各中间码特征序列进行频繁项集的相似度分析，选择相似度最小的中间码特征序列，作为生成概率矩阵的依据；最后，使用机器学习算法完成分类检测。

图3-2 本文算法主要流程

数据预处理阶段和一般检测算法并没有多大区别，包括恶意代码的查壳与脱壳、反汇编。

壳是指在计算机软件中一段专门保护软件不被非法修改或反编译的程序。它们一般都先于程序运行，拿到控制权，然后完成保护软件的功能。

壳通常分为两类：压缩壳和加密壳。压缩壳出现较早，可追溯到DOS时代，使用压缩壳可以帮助缩减PE文件大小，隐藏PE文件内部代码和资源，便于网络传输和隐藏。压缩壳通常有两种用途，一种是单纯用于压缩普通的PE文件，另一种则会对源文件产生较大的变形，严重破坏PE文件头，通常用于压缩恶意程序。常见的压缩壳有：Upx、ASpack、PECompat。加密壳，也称为保护壳，它主要的功能是防止逆向分析技术，保护PE文件不被逆向分析。加密壳保护的文件通常比PE源文件大得多。常见的加密壳有：ASProtector、Armadillo、 EXECryptor、Themida、VMProtect。

现阶段主要的查壳手段有两种：基于特征码和基于信息熵。

基于特征码：同样的加壳方法会使得加壳后的PE文件在特定的位置有相同的字节序列。相同的序列即为该加壳方式的特征，然而这种基于特征码的查壳方法，只能检测出已有的并且加入到特征库的加壳方法。

基于信息熵：经过加壳的PE文件，其结构会产生变化。壳一般分为加密壳和压缩壳，加密和压缩会使 PE文件变成随机性更大的无结构形式。然而，熵是用来衡量不确定性的指标，这样经过加壳的PE文件的熵一般会大于源PE文件的熵。

目前有很多加壳工具，常用的有ASPACK、PE PACK、UPX、PECOMPACT 等。相应的也有一些查壳软件有PEID、FILEINFO、FILE SCANNER等，本文主要是用PEID的批量模式查壳。

反汇编指将机器代码转换为汇编代码、低级转高级的意思，常用于逆向工程领域。目前网络上许多“免费软件”，PSP、PS、NDS游戏机的破解和苹果系统的越狱都跟反汇编息息相关。

基本的反汇编算法有两种：线性扫描反汇编和递归下降反汇编。

线性扫描反汇编算法从程序的机器指令进行全盘扫描，从程序的第一个指令字节开始，对所有检测到的指令以线性模式逐条反汇编，顺序的把所有字节码转换成指令，直到遇到程序的终结点。

递归下降反汇编算法重视控制流的概念，该算法按照指令是否为转移指令决定下一步操作。将整个过程分为两类，要么顺序遍历，要么跳转到另一个程序模块。当遇到指令转移指令时，将此位置作为一个分岔点，然后顺序反汇编每一个分支，每一个分支中如果又有分岔点，做同样的递归处理，直到遇到程序的终结点。

反汇编工具有很多，如有IDA Pro、C32Asm、W32DASM、花指令清除器1.2 等等。本文使用objdump工具对恶意代码样本进行批量反汇编，主要使用objdump -d指令，配合PowerShell脚本完成恶意代码样本的反汇编。

根据不同的抽象方式对汇编操作码做抽象化处理，得到各个中间码序列，以此来尝试从不同视角来理解程序。

本次实验会使用两种抽象化方式，分别记为Abs1和Abs2，其中Abs1是作者根据对汇编码的理解提出的抽象方式，中间码种类为9，Abs2是文献[1]中提出的抽象化方式，中间码种类为5。下面分别对Abs1和Abs2进行详细介绍。

Abs1将汇编操作码抽象为9类中间码：

（1）数据传输，比如：MOV、PUSH、POP、IN、OUT等；

（2）算术运算，比如：ADD、SUB、CMP、DIV、MUL等；

（3）逻辑运算，比如：NOT、AND、OR、TEST等；

（4）移位运算，比如：SAL、SHL、SAR、SHR等；
（5）字符串操作，比如：MOVS、STOS、CMPS等；
（6）过程控制，比如：JMP、JZ、JP等；
（7）标志位控制，比如：CLC、STD、STC等；
（8）CPU控制，比如：HLT、WAIT、NOP等；
（9）其他，伪指令及不常用的指令，比如：SEGMENT，ASSUME，END等。
这9种中间码的表示方法如表3-1：
表3-1 Abs1抽象中间码的表示
种类表示方法
数据传输 DATA_TRANS
算术运算 ALTH_OPE
逻辑运算 LOGIC
移位运算 SHIFT
字符串操作 STR_OPE
过程控制 PRO_CTRL
标志位控制 FLAG_CTRL
CPU控制 CPU_CTRL
其他 OTHER
Abs2将汇编操作码抽象为5类中间码：
（1）数据传输，比如：MOV，PUSH，IN等；
（2）运算，其中包含算数运算指令(ADD、SBB等)和逻辑运算指令(OR，XOR，TEST等)；
（3）程序转移，这是指影响程序正常流程的指令，比如：JMP，CALL，RET，INT等；
（4）处理机控制，比如：NOP，HLT，WAIT等；
（5）其他，伪指令及不常用的指令，比如：SEGMENT，ASSUME，END等。

指　标
疑似剽窃观点
1. 还是动态分析，都需要借助恶意代码分析技术和监控技术获得恶意代码的基本属性和执行信息，以便深入理解恶意代码的功能，进一步实现恶意代码的检测和抑制。
疑似剽窃文字表述
1. 后门可以使攻击者远程控制目标主机，危害无穷。后门提供的通道有几种类型：本地权限提升、远程命令行访问、单命令远程执行、远程控制等。 2. 通常攻击者通过后门获取管理权限，并使用 Rootkit 维持管理权限使的恶意代码能隐藏在目标系统中。Rootkit 分为用户模式和内核模式。用户模式通过通道插入恶意代码、覆盖文件、API 钩子和 DLL 注入等方式达到目的。而内核模式通过安装恶意的设备驱动程序、打补丁、修改内存中运行的内核以及虚拟伪造系统的方式实现。 3. 它是在未授权的情况下窃取用户的信息并通过网络发送给攻击者的一种恶意代码。这种恶意代码不仅仅能泄露目标主机的数据信息，还可以提供恶意代码的植入接口使得被侵系统受到更加严重的破坏。 4. 它是指在未经用户授权的情况下和别的程序捆绑在一起，以便经常弹出一些用户不想接受的广告。这种恶意程序目的是通过这种强制的方式做商业宣传。一些广告插件的安装会降低主机的性能。广告软件主要的危害是弹出一些色情或者恶意的广告，这会给用户带来很大的困扰。 5. 它是指在网页中嵌入一些用脚本语言编写的有恶意行为的代码。当用户点击带恶意脚本的网站后，脚本通过修改目标系统的注册表、 下载病毒或者加载木马程序等方式对被侵系统实施破坏行为。 6. 软件中，一般采用的都是基于“特征码”的检测技术，基本思想是，当新的恶意代码被发现后，对其进行采集取样，分析代码构成，提取有用的特征码，然后将新的特征码加入已有数据库中，用户更新病毒库之后， 7. 特征库去匹配恶意代码，如果匹配成功则进行相应的处理。但是特征码检测技术的缺点是只能对已知恶意代码进行有效的将测，对 8. 恶意代码检测领域提出了启发式检测算法来预防和检测新的恶意代码。根据对恶意代码分析原理的不同对现有恶意代码检测方法进行分类，主要分为基于特征码的检测技术、基于行为的检测技术、基于启发式的检测技术、基于语义的检测方法和基于机器学习算法的检测技术等。 9. Symantec等多有著名病毒检测厂商所使用，是目前已知的所有恶意代码检测方法中最简单、开销最小的方法，广泛用于文件类型的病毒检测中。 10. 匹配，如果匹成功，则判断该程序是恶意的。此技术的关键在于如何选取最能代表恶意程序的特征值。 11. 还有，这种方法使得特征库不断增加，这需要用户经常更新特征库，随着时间的流逝，特征库会越来越庞大，这会影响

检测的速度和系统的性能。

12. 当程序运行时，监控其行为，如果发现了异常行为，则立即报警。一般用于检测恶意代码的行为特征
13. 执行时，就要将自身代码附加在感染文件中，可以监控是否有异常写操作。

2) 监控系统调用序列：某些系统调用序列可以体现某种程度的程序语义。系统调用是用户

14. 系统调用接口。

3) 修改内存总量：恶意代码为了完成特定的恶意意图，经常会常驻在内存中，并且不能被覆盖，那么将会减少系统内存的总量，使得该段内存不受系统

15. 例如一些恶意代码都会固定的对一些内核函数进行调用，通常这些调用的顺序是有一定的规律，因此利用对内核函数的名称和调用次数进行分析，可以构建一个恶意代码
16. 启发式方法属于主动防御技术，对未知的恶意代码检测具有明显的效果，因此，这种方法在现如今的商业开发被重点应用。启发式检测可分为静态启发和动态启发。

静态启发方法其实是对传统的特征识别方法的一种扩展，通过分析程序对系统API的调用序列作为特征，有领域专家根据

17. 研究总结出某些恶意代码的行为特征，当对行为进行监控时，此类特征一旦被发现，就立即报警并做相应的处理。这种方法能够有效的检测出已知的恶意代码，并发现部分未知的恶意代码，但在发现恶意代码的时候，系统往往已经被感染。另外，
18. 动态启发式技术主要的工作原理是在计算机系统中划分出一各独立的虚拟环境，当发现可疑程序时，并不立即停止，而是让其继续运行。“沙盒”技术就是动态启发式技术的一种，沙盒会对可疑程序的行为进行记录，直到恶意代码完全暴露后，它在执行回滚操作，使计算机恢复到执行可疑程序之前的状态。
19. 因为混淆技术只是通过插入垃圾指令、改变指令顺序及寄存器重新分配等方法来改变程序，但是程序的基本语义是等价的。通过分析恶意程序，抽象程序指令的行为并建立其行为模型，使得该模型既描述恶意程序的基本行为，又具有很强的泛化能力。这样因其有很强的泛化能力，使得检测恶意程序的变种更加方便快捷。除此之外，也可以检测未知类型的恶意代码。现阶段基于语义的检测方法分为基于内存和函数调用的方法。
20. 模型检测是通过遍历系统所有状态空间，看其中是否有一条符合的路径状态。之后，他还提出迹语义这一概念，将迹语义作为程序的基本语义，并定义了等价的条件，通过抽象解释的方法给出了近似的检测算法。抽象解释理论为解决不可判定和复杂问题的逼近求解提供可很好的构造方法。基于函数调用的方法是将程序中使用的函数提取出来，并结合程序的控制流程图，通过图的同构、线性时态逻辑、计算逻辑树、有穷状态机及下推自动机等方法描述恶意行为，最终通过模型检测完成恶意代码的检测。
21. 随着检测技术的不断发展，机器学习和数据挖掘的方法已经被开始应用在恶意代码检测的领域。主要应用分类、关联规则挖掘、序列模式分析以及聚类等几种技术。主要思想是利用数据挖掘技术从现有的数据中挖掘一些有意义的模式，用机器学习技术归纳出已有样本的特征，然后根据特征的相似性等完成分类的任务。其中，最主要的是选择好的特征和有效的分类器。检测步骤如下：首先，要分析样本确定提取哪种特征或者特征序列；其次，根据特征的特性选择合适的特征选择方法从所有提取的特征中选择一些分类效果好的特征；最后，根据实际情况选择较好的模型实现分类。

恶意代码检测与反检测技术总是相互促进，相辅相成。

22. 比如，硬件上，调试器总是会设置硬件断点，而虚拟机总是在模拟硬件，这和真实的硬件是有差别的；执行环境，内核地址空间，对于虚拟机和真正的机器是不同的，还有调试器必须挂靠某些进程来插桩进程用于监控；应用程序，虚拟机和调试器都有外部应用程序，对进程可见，用于检查运行环境。一些指令在虚拟机环境中，执行时间总是远远长于真实环境，一个经常执行此类指令的程序能够
23. 恶意代码检测的必要前提。恶意代码分析的直接结果是用于实现恶意行为建模的元数据信息，如指令流、API调用序列等，为后续恶意代码的检测工作提供必要的支持。

恶意代码的分析技术一般可分为静态分析和动态分析。

24. 工具有：W32DASM、objdump、PEid、HIEW、IDA Pro等。

静态分析技术由于不会运行程序，因此不会对计算机系统造成任何伤害，其分析效率相对动态分析而言较高，同时由于静态分析技术从程序本身入手，因此可以获得程序的全部信息，分析结果较为全面。但是由于静态分析技术的

25. 由于动态分析需要先运行程序，所以为了防止恶意代码对当前环境的破坏，系统在普或者恶意代码相关信息之后，会自动恢复到恶意代码执行前的最初状态，防止影响下一次的分析结果，但是动态分析技术能够获得恶意代码执行时的真实信息，可以有效地解决静态分析中譬如加壳、加密的干扰。
26. 动态污点分析能够记录恶意代码更细粒度的精确特征，是当前非常热门的恶意代码检测技术。
27. 分析技术各有优缺点，静态分析技术开销小，关注的是恶意程序本身的语法或者结构特征，动态分析技术开销大，关注的是恶意代码
28. 行为特征，各有侧重点。静态分析技术分析全面，可获得恶意代码的全部信息，但获取特征的方式一般都是无导向的，因此可能包含大量无用信息，也易受代码迷惑技术的影响，动态分析技术能够获得程序的真实行为信息，但一次只能获得一种行为并且与当前的检测环境相关，信息不够全面。
29. 本章首先介绍了恶意代码的定义以及恶意代码的分类，具体介绍了病毒、特洛伊木马、蠕虫
30. 检测技术，详细阐述了基于特征码的检测技术、基于行为的检测技术、基于启发式的检测技术、基于语义的检测技术和

基于机器学习的检测技术。最后，针对目前主流的恶意代码反检测

31. 对于恶意代码的检测具有实际意义。
大部分的基于机器学习算法的恶意代码静态检测技术都会使用

32. 首先是样本预处理，该步骤包括对所有的样本进行查壳脱壳处理，并对脱壳后的样本进行反汇编，得到其汇编文本文件；其次，

33. 计算机软件中一段专门保护软件不被非法修改或反编译的程序。它们一般都先于程序运行，拿到控制权，然后完成保护软件的功能。

34. 现阶段主要的查壳手段有两种：基于特征码和基于信息熵。
基于特征码：同样的加壳方法会使得加壳后的PE文件在特定的位置有相同的字节序列。相同的序列即为该加壳方式的特征，然而这种基于特征码的查壳方法，只能检测出已有的并且加入到特征库的加壳方法。
基于信息熵：经过加壳的PE文件，其结构会产生变化。壳一般分为加密壳和压缩壳，加密和压缩会使 PE文件变成随机性更大的无结构形式。然而，熵是用来衡量不确定性的指标，这样经过加壳的PE文件的熵一般会大于源PE文件的熵。
目前有很多加壳工具，常用的有ASPACK、PE PACK、UPX、PECOMPACT 等。相应的也有一些查壳软件有PEID、FILEINFO、FILE SCANNER等，本文主要是用PEID的批量模式查壳。

35. 基本的反汇编算法有两种：线性扫描反汇编和递归下降反汇编。
线性扫描反汇编算法从程序的

36. 汇编工具有很多，如有IDA Pro、C32Asm、W32DASM、花指令清除器1.2 等等。本文使用

这5种中间码的表示方法如表3-2：

表3-2 Abs2抽象中间码的表示

种类表示方法

数据传输 ASSIGN

运算 TEST

程序转移 JMP

处理机控制 CTRL

其他 OTHER

抽象化过程的实现流程如下：

（1）首先将所有的汇编操作码与中间码映射关系存储在配置文件中，配置文件格式如图3-3。其中IC（Intermediate Code）代表中间码，Abs1代表抽象名称，asm_code代表汇编操作码；

（2）循环读取配置文件的每一行。符号‘#’表示忽略其后面的内容，符号‘@’表示其后为抽象化方式的名称，从下行开始直到结束或者遇到符号‘@’，中间的内容都是该抽象方式的具体内容，符号‘：’前表示中间码名称，符号‘：’后面的内容以符号‘，’分隔，每一个都表示一种汇编操作码；然后将读取到的内容存储到类型map<string,map<string,string>>中，三个string类型分别表示抽象方式名称，汇编操作码、中间码。

图3-3 抽象化配置格式

（3）读取汇编文本文件，遍历其中的每一个汇编操作码，对其应用每一种抽象方式，得到对应的中间码表示，最后得到N种中间码序列，N表示配置文件中抽象方式的个数。

n-gram是计算机语言学和概率论范畴内的概念，目前n-gram被广泛应用于自然语言的自动分类功能。该模型基于第n个词出现只和其前n-1个词相关的假设，对于处理依靠序列关系分类的问题具有很强的优势。

为避免多次读写文件影响系统性能，本文在得到中间码序列之后，继续在内存中使用n-gram算法得到中间码特征序列，并将中间码特征序列写入文件中。具体算法描述如表3-3：

表3-3 n-gram算法获取中间码特征序列描述

算法1：应用n-gram算法获取中间码特征序列

Input: IC_sequenceOutput: n-gram sequenceindex=start_indexfrom index to size(IC_sequence)-1{if index+n < size(IC_sequence){ret.push_back(<IC_sequence[index:index+n-1]>)}else{break;}}return ret

（1）设置当前索引为起始位置；

（2）判断当前位置是否合法，如果合法，转步骤（3）；否则转步骤（4）；

（3）判断当前位置索引加上n值是否合法，如果合法，则将当前位置开始往后的n-1个序列作为一个整体加入结果集，并转步骤（2）；否则，转步骤（4）；

（4）返回结果集并结束算法。

该算法结束时，将会得到一个n-gram中间码特征序列，每一个序列依次对应一个恶意代码样本。本文将同一类别的恶意代码的中间码特征序列放在同一个文件中，每一个序列占用一行。

对中间码应用n-gram算法提取特征后，对于本文，将会得到两种不同的中间码特征序列，本节将使用Eclat算法得到这两种中间码特征序列的频繁项集，并利用频繁项集得到中间码特征序列的平均相似度，然后选择平均相似度最小的一种中间码特征序列作为概率矩阵的生成依据。

本文使用Eclat算法来分析特征文本的频繁项集。尽管Apriori算法是最广为人知的关联规则挖掘算法，但是算法需要多遍扫描数据库因而会产生大量的候选项集，支持度的计算也很耗时。相对于Apriori算法，Eclat算法采用了等价类、深度优先遍历、求交集等策略，支持度计算效率有很大改善。

Eclat算法采用的数据结构分为两部分：项和事务。Eclat算法在遍历数据库时，将事务划分到项下，使得该算法相较于Apriori算法可以基于集合运算很方便的得到频繁项集。将该算法应用到本次实验，项对应n-gram特征序列，事务对应文本的行号。应用Eclat算法得到频繁项集的思路如下：

（1）首先生成项对应的事务集，如图3-4中左上角表格，第一列的ABCDE代表n-gram特征序列，第二列T1-T9代表行号；

（2）把所有项作为一个集合，并求该集合所有的子集，如图3-4中橘色线标出的集合；

（3）对子集中的所有项对应的事务集合求交集，作为该子集对应的事务集合Ti，如图3-4蓝色线指向的集合；

（4）如果集合Ti的基数大于设定的阈值，即判定该项为频繁项。

本文只统计单一的频繁项，并不会统计各个项组合的情况，因此本文对该算法做了简化，具体算法描述如表3-4：

（1）设置遍历起始行；

（2）循环遍历文件的每一行，分割该行字符串，将每一个特征序列作为key值，将行号插入到value所表示的集合中，直到文件结束；

（3）遍历步骤（2）统计到的结果，如果某一key值对应的value中存储的值的个数超过设定的阈值，则将该key值，即就是特征序列插入到结果集；直到循环结束返回结果集。

表3-4 Eclat算法描述

算法2：Eclat算法获取频繁项集

Input: n-gram IC_sequence fileOutput: Frequent Item Setsvector<string> n_gram_IC_sequences = read(n-gram IC_sequence file)index=start_indexfrom index to size(n_gram_IC_sequences)-1{vector<string> items=split(n_gram_IC_sequences[index],’,’)from i = 0 to size(items)-1 {count[items[i]].insert(index)}}ret = getFrequentItems(count,Threshold)return ret

图3-4 Eclat算法示例图

在抽象方式一定的情况下，分别对每一个类别标签下的中间码特征序列应用算法2，得到各个类别在同一中抽象方式下的频繁项集，接着本文使用平均相似度评估该中间码特征序列生成的频繁项集对于恶意代码分类的效果，平均相似度越低，代表分类效果越明显，反之，代表分类效果差。频繁项集的平均相似度定义如式（3-1）：

AVGlike=cardA*α,βAα≠βcard(α∩β)Ccard(A)2*αAcard(α)（ 3-1）

式中α，β分别代表各类的频繁项的集合，A表示各类频繁项集的集合，card(A) 表示集合A的基数。例如，集合

A={α,β,γ}，其中集合α频繁项为{a,b,c,d}，集合β频繁项为 {a,b,e,f}，集合γ频繁项为{c,d,f,g}，那么

AVGlike=3*(cardα∩β+cardα∩γ+cardβ∩γ)C32*(4+4+4)=512≈41.67%

得到了各中间码特征序列生成频繁项集的平均相似度之后，再从中选取平均相似度最小的中间码特征序列，作为生成概率矩阵的依据。

应用特征分析步骤选择出的中间码特征序列，统计分析出其对应的概率矩阵，具体算法流程如表3-5所示。

（1）遍历选择的中间码特征序列文件，并获得每个词组在该行中出现的频率；

（2）对每一行循环N次，N代表n-gram算法中的n值，这将会组成该抽象方式下每一种可能出现的n-gram词组，记录该行出现这种形式词组的频率；

（3）将该行中间码特征序列频率分布作为代表该恶意代码的概率分布，继续对下一行做重复操作，直到文件结束。

表3-5 获取概率矩阵的算法描述

算法3：应用中间码特征序列获取概率矩阵

Input: n-gram IC_sequence fileOutput: Probability matrix fileforeach line in n-gram IC_sequence file{line_pros = getWordProbabilityInLine(line)foreach IC_1 in AbsX{foreach IC_2 in AbsX{:foreach IC_N in AbsX{matrix_line += line_pros[“IC_1:IC_2:⋯:IC_N”])}}} }writeFileLine(class，matrix_line)clear(matrix_line)}

最终本文将会获得一个含有类别标签的概率矩阵文件，用于机器学习算法的模型训练以及评估。至此，特征提取部分全部完成。

随机森林是一种集成学习算法，该算法在学习过程中将产生多个决策树，每棵决策树会根据输入数据集产生相应的预测输出，算法采用投票机制选择类别众数作为预测结果。随机森林的训练模型生成流程如图3-5所示。

图3-5 随机森林算法训练模型生成过程

首先采用bootstrap对数据进行采样，样本总数为N，那么每次也随机采样N个样本作为单个决策树的训练数据集，采样是有放回的采样，所以并不是使用样本空间中的每一个样本作为单个决策树的训练集数据。在每个节点，算法首先随机选取m（m<<M）个变量，从它们中间找到能够提供最佳分割效果的预测属性；然后，算法在不剪枝的前提下生成单个决策树；最后从每棵决策树都得到一个分类预测结果。如果是回归分析，算法将所有预测的平均值或者加权平均值作为最后输出，若果是分类问题，则选择类别预测众数作为最终预测。

本文通过实验对比分析，最终选择决策树的个数T为500，随机属性个数m为6。

很多研究已经证明支持向量机（Support Vector Machine，SVM）是一种强大的分类工具，可以被广泛的应用到不同的领域。与随机森林算法不同，在SVM训练中，从输入数据到输出结果的过程并不清晰，也难以解释，因此SVM属于黑盒算法。SVM算法的基本思想：通过定义核函数将输出数据映射到高维特征空间上，并在此空间中构造一个最有分类超平面（或者一组超平面），使得高维特征空间内的各个类的边缘间隔最大化。定义这些超平面的向量就被称为支持向量，如图3-7中被正方形圈出的o和x：

图3-6 SVM分类示意图

SVM算法的关键在于核函数，常用的核函数主要有四种：

（1）线性核函数：Kx,y=x·y;

（2）径向基函数：Kx,y=exp-γx-y2,γ>0;

（3）多项式核函数：Kx,y=γx+y+rd,γ>0;

（4）二层神经网络核函数：Kx,y=tanhγx·y+r;

式中的γ，r和d都是核参数。

SVM算法是最初设计是为解决二分类问题的，但是本此实验是一个多分类问题，所以需要对SVM算法做一些改动。目前针对SVM多分类问题主要有两种解决方案：直接法和间接法。

直接法：即直接修改目标函数，但是这种算法复杂度较高，很难实现，只适合解决规模较小的问题。

间接法：通过组合多个二分类器实现多分类器功能，常见的方法有两种：一对一法和一对多法。一对多法可能会出现训练集分布不均的情况，甚至可能会有未分类的情况出现，所以本文使用一对一的方法，其算法描述如表3-6所示。对每一个类别的训练集，都和类别集合中的其他类别训练集使用SVM算法构建二分类模型，如果类别总数为n时，二分类SVM算法将会执行n+1*n2次，接着待检测样本使用每一个二分类模型进行预测，最后选取最频繁的预测值作为总的预测结果。这也是libsvm中SVM多分类的实现的方式。

表3-6 SVM实现多分类算法描述

算法4：应用SVM实现多分类

Input：multi-classes trainsetOutput：sample classification resultforeach class_i in classeset{foreach class_j in classeset except class_i{multi_classification_svm_model.insert(svm_model(class_i_trainset, class_j_trainset))}}return getMostFrequentClass(predict(sample,multi_classification_svm_model))

K近邻（K-Nearest Neighbor，KNN）算法属于一种无参惰性学习方法。无参类算法不会对数据的分布做任何的假设，而惰性学习方法则不要求算法具备显性学习过程。算法的主要思想是：通过对样本进行建模，使得可以用一个向量表示一个样本，然后采用一种度量方式，度量方式一般使用欧式距离或者曼哈顿距离，计算出离被测样本最近的K个已知类别的样本，接着

统计这K个样本的类别次数，则判定被测样本的类别与出现次数最多的类别相同，为了避免出现平票的情况，K一般选择为奇数。

KNN算法步骤如下：

（1）首先对特征向量进行归一化处理，本文采用线性函数的转换方法：

y=(x-min)/(max-min) (3-2)

其中y为归一化处理之后的值，x为处理前的值，max和min分别代表一个样本中出现的最大值和最小值。

（2）使用一种度量方式计算待检测恶意代码样本和所有已知类别数据样本的距离，本文使用欧式距离；

i=1k(xi-yi)2（3-3）

（3）按照距离递增进行排序；

（4）选取和当前待测恶意代码样本距离最小的K个点；

（5）确定前K个点所属类别的出现频率；

（6）返回前K个点出现频率最高的类别作为当前点的预测类别，则目标函数为：

fv=argmaxc∈Ci=1kδ(c,f(vi)) (3-4)

其中，函数fv表示特征向量v的类别，如果f(vi)的类别和c相同，则δ(c,fvi=1，否则δ(c,fvi=0。

本文使用KNN算法来做实验仿真分析，通过实验选择分类较好的K值为5。

为了确保模型能对未知或新到达的对象进行正确预测，需要对模型性能进行评估，避免模型可能存在过拟合问题。

k折交叉验证技术能够解决过拟合问题，因此被广泛应用于分类器性能评估领域。k折交叉验证并不需要使用整个数据集，相反，它会将数据集划分为训练集和测试集两部分。这样，基于训练集得到的模型就可以通过测试集来完成性能测评。重复执行n次k折交叉验证后，就能够根据n次检验的平均正确率实现对模型的真实评估。本次实验n取值3，k取值为10，即就是3重10折交叉验证。

对于恶意代码的检测系统，一般采用准确率（Accuracy），精确率（Precision）和召回率（Recall）来评价其结果。用TP表示恶意样本被正确分类的数量；FN表示恶意样本被判定为正常样本的数量；TN表示正常样本被正确分类的数量；FP表示正常样本被判定为恶意样本的数量。对于多元分类模型，准确率、精确率和召回率的计算公式如下，其中I表示分类个数：

Average Accuracy=1Ii=1ITPi+TNiN（3-5）

Average Precision=1Ii=1ITPiTPi+FPi（3-6）

Average Recall=1Ii=1ITPiTPi+FNi（3-7）

实验环境：Intel酷睿i5-7300HQ@2.50GHz 双核，8.00G内存，操作系统为Microsoft Windows 10，编程语言主要是C++。本文算法实现采用C++语言编程，利用CodeBlocks集成工具实现相关算法的程序设计。

本文使用的实验数据来自Kaggle上微软发起的一个恶意代码分类的比赛，使用的数据集的大小为136GB，其中样本种类分布如表3-7：

表3-7 恶意程序数据集

恶意程序家族	数量（单位，个）
Ramnit	1541
Lollipop	2478
Kelihos_ver3	2942
Vundo	475
Simda	42
Tracur	751
Kelihos_ver1	398
Obfuscator.ACY	1228
Gatak	1013
total	10868

应用3.4.3小节特征分析中提到的算法，得到特征分析的结果如图3-7所示，由图3-7可以看出，对于2-gram算法，Abs1对应的中间码特征序列得到的频繁项集平均相似度较小，本文提出的算法将会选择该中间码特征序列生成概率矩阵；而在3-gram和4-gram算法中，Abs2对应生成的中间码特征序列得到的频繁项集平均相似度较低，本文提出的算法将会选择该中间码特征序列生成概率矩阵。

为了证明特征分析阶段选择生成的概率矩阵确实会得到更好的结果，本次实验将Abs1以及Abs2对应的概率矩阵都生成，并且使用机器学习算法来对样本进行分类，对比分类效果。同时为了分析出哪种机器学习算法使用本文生成的概率矩阵分类效果最佳，本次实验将会分别使用随机森林算法、SVM和KNN算法对实验进行分类。

图3-7 平均相似度对比图

下面对实验结果进行分析。

1. 准确率比较

下面通过对3种n-gram、2种抽象方式和3种机器学习算法的18种不同的组合来对比准确率。

图3-8中，横坐标表示不同的机器学习算法，纵坐标为准确率。其中蓝色柱状抽象方式Abs1，橘色柱状表示抽象方式Abs2，RF代表随机森林算法，并且下面的图中均表达此种意义。

对比图3-8的结果，当n=2时可得出如下结论：

图3-8 n=2时准确率对比图

（1）平均准确率：SVM>随机森林算法>KNN。

（2）算法稳定性：随机森林算法>KNN>SVM。

（3）是否符合特征分析结果：符合，不论哪种机器学习算法，Abs1对应的准确率始终高于Abs2对应的准确率，所以实验结果符合特征分析判断的结果。

对比图3-9的结果，当n=3时可以得出如下结论：

图3-9 n=3时准确率对比图

（1）平均准确率：随机森林算法>KNN>SVM。

（2）算法稳定性：随机森林算法>KNN>SVM。

（3）是否符合特征分析结果：符合，不论何种机器学习算法，抽象方式为Abs2时，准确率均高于抽象方式为Abs1时的准确率。

对比图3-10的结果，当n=4时可以得到如下结论：

（1）平均准确率：KNN>随机森林算法>SVM。

（2）算法稳定性：KNN>随机森林算法>SVM。

图3-10 n=4时准确率对比图

（3）是否符合特征分析结果：符合，不论采用何种机器学习算法，抽象方式为Abs2时，其准确率均大于抽象方式为Abs1时的准确率。

2. 精确率比较

下面通过对3种n-gram、2种抽象方式和3种机器学习算法的18种不同的组合来对比精确率。

图3-11中，横坐标表示不同的机器学习算法，纵坐标为精确率。其中蓝色柱状表示抽象方式Abs1，橘色柱状表示抽象方式Abs2，RF代表随机森林算法，并且下面的图中均表达此种意义。下面通过不同的n值和不同的机器学习算法来对比精确率。

图3-11 n=2时精确率对比图

对比图3-11的结果，当n=2时可得出如下结论：

（1）平均精确率：SVM>随机森林算法>KNN。

（2）算法稳定性：KNN>随机森林算法>SVM。

（3）是否符合特征分析结果：符合，无论采用何种机器学习算法，抽象方式为Abs1时，精确率都大于抽象方式为Abs2的情况。

对比图3-12的结果，当n=3时可以得出如下结论：

（1）平均精确率：随机森林算法>KNN>SVM。

（2）算法稳定性：随机森林算法>KNN> SVM。

（3）是否符合特征分析结果：比较不符合，除了SVM算法的精确率在Abs2时大于Abs1，随机森林算法和KNN算法都不符合特征分析结果。

图3-12 n=3时精确率对比图

对比图3-13的结果，当n=4时可以得出如下结论：

（1）平均精确率：KNN>SVM>随机森林算法。

（2）算法稳定性：KNN>随机森林算法> SVM。

（3）是否符合特征分析结果：符合，抽象方式为Abs2时，精确率都大于抽象方式为Abs1的情况。

图3-13 n=4时精确率对比图

3. 召回率比较

下面通过对3种n-gram、2种抽象方式和3种机器学习算法的18种不同的组合来对比召回率。

图3-14中，横坐标表示不同的机器学习算法，纵坐标为召回率。其中蓝色柱状表示抽象方式Abs1，橘色柱状表示抽象方式Abs2，RF代表随机森林算法，并且下面的图中均表达此种意义。下面通过不同的n值和不同的机器学习算法来对比召回率。

对比图3-14的结果，当n=2时可得出如下结论：

图3-14 n=2时召回率对比图

（1）平均召回率：随机森林算法>SVM> KNN。

（2）算法稳定性：随机森林算法>KNN> SVM。

（3）是否符合特征分析结果：基本符合，除了KNN算法在Abs1时召回率小于Abs2的情况，其他两种机器学习算法都在Abs2时召回率大于Abs1。

对比图3-15的结果，当n=3时可得出如下结论：

（1）平均召回率：随机森林算法> KNN>SVM。

（2）算法稳定性：随机森林算法>KNN> SVM。

图3-15 n=3时召回率对比图

（3）是否符合特征分析结果：符合，三种机器学习算法在Abs2时召回率都大于Abs1时的召回率。

对比图3-16的结果，当n=4时可以得出如下结论：

图3-16 n=4时召回率对比图

（1）平均召回率：随机森林算法> KNN>SVM。

（2）算法稳定性：随机森林算法>KNN> SVM。

（3）是否符合特征分析结果：符合，三种机器学习算法在Abs2时召回率都大于Abs1时的召回率。

4. 实验结果分析

通过对实验结果的对比分析，可以得出如下结论：

（1）本文提出的方法相比于传统方法有一定的优越性。在2-gram算法中，本文提出的方法将会选择Abs1对应的中间码特征序列生成概率矩阵，而传统的方式只能使用Abs2的中间码特征序列，图3-17显示了本文方法和传统方法在准确率、精确率和召回率指标上的差距，可以看出本文与传统方法相比会得到更好的结果，并且在2-gram时这三个指标的平均值是18种组合里最高的；对于3-gram算法和4-gram算法，本文选择的抽象方式Abs2，与传统方法相同，都会得到准确率、精确率和召回率较高的结果。

图3-17 n=2时本文方法和传统方法对比图

（2）随机森林算法是三种算法中稳定性最好的，并且在准确率、精确率和召回率三个指标中的结果比较令人满意，所以在要求平均性能较好的情况下，优先选择随机森林算法。SVM算法是最有可能出现峰值的算法，但是也是相对来说最不稳定的一个算法。相比而言，KNN算法是最不出众的算法，虽然稳定性优于SVM算法，但是其平均性能不如随机森林算法。所以本文实现的机器学习算法的恶意代码检测系统将会选择随机森林算法来进行检测。

本章首先分析了汇编操作码的特点，在分析了相关文献之后，提出了一种基于机器学习的恶意代码检测算法；接着简要介绍了该算法的主要流程，然后对算法的各步骤进行了详细的说明，重点对概率矩阵的生成进行了详尽的解释说明，然后简单介绍了机器学习算法、SVM算法和KNN算法的基本原理和在本实验中的使用情况；最后对实验仿真结果进行了分析，并给出结论。

本文实现的系统主要目的是：用户能够使用一个相对友好的界面去提交想要检测的程序，系统接收到程序之后，能够快速而准确向用户反馈检测结果。

系统用例图如图4-1所示：

图4-1 恶意代码检测系统用例图

选择上传程序功能：用户上传需要检测的本地程序，现阶段系统仅支持PE文件，即格式为EXE、DLL、OCX、SYS和COM格式的文件。上传程序之前会对文件格式进行检测，如果用户上传文件格式不符合要求，则拒绝上传并提示用户重新选择文件进行上传。

查看检测结果功能：用户上传了需要检测程序后，系统后台会返回检测结果，并由本地展示检测结果。

反馈结果：当用户认为结果与实际情况不符或者检测结果不可信时，可向系统替提交反馈，以便系统进行再次确认。

指　标

疑似剽窃文字表述

---

1. n-gram被广泛应用于自然语言的自动分类功能。该模型基于第n个词出现只和其前n-1个词相关的假设，对于处理依靠序列关系分类的问题具有很强的优势。
2. 计算出离被测样本最近的K个已知类别的样本，接着统计这K个样本的类别次数，则判定被测样本的类别与出现次数最多的类别相同，
3. 用TP表示恶意样本被正确分类的数量；FN表示恶意样本被判定为正常样本的数量；TN表示正常样本被正确分类的数量；FP表示正常样本被判定为恶意样本的数量。

17. 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第17部分　总字数：2377

相似文献列表　文字复制比：0%(0)　疑似剽窃观点：(0)

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

针对上述用例图，本文提出了一种系统的设计框架，如图4-2所示：

图4-2 系统框架图

系统共包含5个模块：UI模块，通信模块，检测模块，存储模块和模型训练模块。

UI模块主要有两个功能：上传待检测程序和展示检测结果。

通信模块主要由两个功能：对用户上传的程序进行存储和以及向用户传输检测结果。

检测模块负责对用户提交的程序给出检测结果。当得到用户上传的程序之后，对程序做MD5运算得到哈希码，首先使用该

哈希码去查询数据库，如果在数据库中查询成功，则表示该程序已经被检测过，直接返回检测结果；否则使用系统的检测模型，对程序进行检测，得到分类结果之后，将结果交给通信模块向用户报告结果，并且将分类结果交给存储模块进行存储。

存储模块负责对检测结果进行存储并负责向模型训练模块提供训练样本。

模型训练模块负责定时或者主动的对存储模块中的数据进行重新训练，更新检测模型。

UI模块功能比较简单，为了方便恶意代码的检测，本文使用了CS架构实现检测系统，用户只需在网页上传需要检测的程序，然后可以在本地查看检测结果。

开始界面展示如图4-3所示，选择上传需要检测的文件界面如图4-4。

图4-3 web端首页展示

图4-4 上传程序界面

上传文件之后，点击开始检测，得到检测结果如图4-5所示：

图4-5 检测结果展示图

通信模块是UI模块和后台模块交互的唯一接口，通信模块主要有以下功能：

（1）保证客户端和服务端的正常交互。

（2）接收客户端上传的程序文件或者反馈信息，并通知相应模块。

（3）将检测模块的分析结果返回给用户。

通信模块主要使用tomcat7.0.82完成。

检测模块是系统的核心模块，该模块接收来自通信模块的消息，当被告知有程序需要检测，检测模块首先会对程序求MD5值，然后将该MD5值传给存储模块，由存储模块先在数据库中去查找该MD5，如果查找成功，则返回存储的检测结果给检测模块，由检测模块将检测结果返回给通信模块；如果在数据库中查找失败，则使用系统中现有的预测模型对该程序进行检测。检测完成之后，首先将结果传输给存储模块进行存储，再将结果交由通信模块。具体过程如图4-6所示。

当数据库中不存在待检测程序的MD5值时，程序将会根据系统已有的检测模型对程序进行检测。具体过程如图4-7所示。首先，对程序进行预处理，包括查壳和脱壳以及反汇编过程；然后从汇编文件中提取汇编码，并根据现有预测模型的抽象方式和n-gram中n的取值提取程序的特征码序列，并构建概率矩阵，图中AbsX表示抽象方式编号为X；最后使用概率矩阵作为预测模型的输入，得出该程序的检测结果。

存储模块主要使用MySQL数据库对PE文件的检测结果进行存储，数据库的设计也比较简单，目前数据库中存在两个表Tb_Malicious_Detected和Tb_Types，Tb_Malicious_Detected表存储所有被检测过的程序及其检测结果，表Tb_Types存储本系统所能识别的所有类别。表Tb_Malicious_Detected结构如图4-8所示：程序的MD5值作为主键，存储类型为32位的char类型，detect_time表示检测时间，if_feedback表示检测结果是否被用户反馈为结果不可信，取值为0或者1，0表示未反馈，1表示反馈，默认值为0，detect_result表示检测结果，存储类型为int类型。将具体类型名称和类型编号对应关系存储在表Tb_Types中，在表Tb_Malicious_Detected中不直接存储类型名称的原因是：随着系统的使用，数据必然会越来越多，将会存在大量的同一类型的程序，这个字段如果存储为字符串类型，会占用大量的磁盘空间。表Tb_Types格式如图4-9所示，自增id为主键，int_type表示程序类型的整型表示，str_type表示程序类型的字符串表示，不直接使用int_type为主键主要考虑到今后可能会对程序类型的划分方式做调整，可以对int_type进行不同解释达到这个目的。目前可识别的类型共有10种，包括3.6.2小节种提到的的9种恶意代码类型和1中正常类型。

图4-8 表Tb_Malicious_Detected结构

图4-9表Tb_Types结构

随着系统的使用，数据量必然会增大，如何利用这些不断增长的数据去使得系统的检测性能更好，这就需要系统不定时的利用已有数据去主动学习，不断改进系统的检测模型。模型训练模块和第三章所述的过程完全一样，只不过数据的标签需要从数据库中获取。

模型训练模块时序图如图4-10所示。首先从数据库中读取所有的标签数据，然后从文件系统中读取上一次构建模型时生成的n-gram中间码序列，本系统中共有6个文件：2-gramAbs1、2-gramAbs2、3-gramAbs1、3-gramAbs2、4-gramAbs1和4-gramAbs2；接着将文件系统中指定位置存储的PE文件进行预处理、抽象化并使用n-gram算法得到6种不同的中间码特征，将其加入到各自对应的中间码特征文件中，此时，需要将6个中间码特征序列文件存储更新到文件系统，并选择删除或者移动所有的PE文件，防止下次进行模型训练时重复被使用；再对这6个文件进行特征分析，选择出频繁项集的平均相似度最小的一种，依据该中间码特征序列生成概率矩阵，最后使用随机森林算法得到检测模型，并存储检测模型、n-gram算法n值和对应的抽象方式，以便检测模块使用。

图 4-10 模型训练模块时序图

本章主要介绍了基于机器学习算法的恶意代码检测系统的设计与实现。先概括介绍了该系统的整个设计框架；然后对各个模块的功能、设计思路和实现方式分别进行了分析讨论。

相似文献列表　文字复制比：13.1%(154)　疑似剽窃观点：(0)

1	恶意代码检测中若干关键技术研究 陈良(导师：李斌;陈斌) -《扬州大学硕士论文》- 2012-04-01	13.1%（154） 是否引证：否

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第十四章朱鹏博

计算机的普及和高速发展的互联网技术带给人们方便的同时，用户信息的安全性受到的威胁也越来越严重，加强互联网安全已经成为了国家的一项重要战略。在众多的网络安全事件中，尤以恶意代码的危害最大，给个人、企业甚至政府造成了巨大的损失，所以恶意代码检测技术成为了信息安全领域重要的研究方向。本文主要研究了基于机器学习算法的恶意代码检测技术，介绍了信息安全和机器学习领域的相关理论，分析了恶意代码检测技术的国内外研究现状，在研究分析了大量恶意代码以及相关机器学习算法之后，提出了一种基于机器学习算法的恶意代码检测方法，通过实验证明了本文提出的方法对恶意代码的分类具有明显的效果，并依据该方法设计实现了一个恶意代码检测系统。

本文完成的主要工作有：

（1）对实验样本进行收集和预处理操作，研究了相关的恶意代码检测分析技术。

（2）提出了一种基于机器学习算法的恶意代码检测方法，主要是使用汇编操作码的抽象化技术，并结合数据挖掘技术选择合适的抽象方式对应的中间码特征序列生成概率矩阵，作为代表恶意代码的特征。当n-gram算法中n值有多个时，这种方法能够针对特定的n值，选择合适的中间码特征序列作为概率矩阵的依据，并由此概率矩阵构建效果比较好的分类模型。通过实验分析，本文提出的方法具有一定的优越性。在三类机器学习算法中，随机森林算法针对于本文方法生成的概率矩阵，能够训练出综合性能相对较好的分类模型。从整个实验结果分析，本文提出的方法具有较高的准确率、精确率和召回率。

（3）设计并实现了一个恶意代码检测系统，采用模块化编程实现系统，增加了系统的灵活性。

本文虽然提出了一种基于机器学习算法的恶意代码检测方法，并且基于该方法实现了一个恶意代码检测系统，但由于时间及各方面条件的限制，本文提出的算法还未在大规模实践中得到验证，这也是下一步着重需要解决的问题。同时在基于机器学习算法的恶意代码检测领域中还有一些问题需要进一步的研究解决。

（1）恶意代码种类层出不穷，其反检测的方法也多种多样，保证得到正确的汇编操作码是恶意代码静态分析的前提。如何有效的对抗反检测技术是未来的一个研究方向。

（2）本文提出的概率矩阵，有一定的适用范围，比如当中间码类别数目为c时，概率矩阵的列数是cn，指数n表示n-gram中n的取值，随着n的增大，概率矩阵列数呈指数级增长，并且对于大多数的列，其值为0。研究如何有效的存储概率矩阵并针对稀疏矩阵做出优化需要后续的研究。

（3）对于汇编操作码的抽象方式，并没有过多的参考文献可供参考，如何对汇编操作码进行有效的抽象化处理也需要进一步的研究。

（4）伴随着数据集的急剧增长，算法的运行时间也在不断增加，对于该算法如何应用在分布式计算框架中以解决算法计算效率问题，也需要在未来的研究中不断探索。

指　标

疑似剽窃文字表述

1. 计算机的普及和高速发展的互联网技术带给人们方便的同时，用户信息的安全性受到的威胁也越来越严重，加强互联网安全
2. 造成了巨大的损失，所以恶意代码检测技术成为了信息安全领域重要的研究方向。本文主要研究了基于机器学习算法的恶意代码检测技术，介绍了信息安全和机器学习领域的相关理论，分析了恶意代码检测技术的国内外研究现状

19. 80022497643220302_基于动态手势的智能终端身份识别技术研究与实现_第19部分　总字数：7731

相似文献列表　文字复制比：6.3%(488)　疑似剽窃观点：(0)

1	基于Intent的SDN北向接口研究综述 张岩;张忠平;张曼君; -《信息通信技术》- 2016-02-15	1.2%（96） 是否引证：否
2	在OpenFlow网络中提供路由服务功能的设计与实现 李鹏飞(导师：王文东) -《北京邮电大学硕士论文》- 2013-01-10	1.2%（95） 是否引证：否
3	专题地图总体设计智能化理论与方法研究 马俊(导师：王光霞) -《解放军信息工程大学博士论文》- 2013-04-16	1.1%（85） 是否引证：否
4	BNF(巴科斯范式) - Hefee的专栏 - 博客频道 - CSDN.NET -《网络（http://blog.csdn.net）》- 2017	1.1%（85） 是否引证：否
5	基于物联网仿真平台的动态可配置网络系统的设计与实现 陈博(导师：张雷) -《北京邮电大学硕士论文》- 2014-12-25	0.9%（67） 是否引证：否
6	盘锦职业技术学院网络性能管理系统的设计与实现	0.8%（61）

第十五章杨二

本章将对本课题在研究与实现时所涉及到技术进行介绍。主要包括对SDN基础知识、ONOS控制器、北向意图接口（Intent NBI）模型的介绍。另外还有一些辅助性的知识例如NEMO引擎、搜索服务器Elasticsearch和BNF范式。

15.1 SDN基础知识

随着网络的迅速发展和网络应用数量的急剧增加，网络的功能和结构日趋复杂，随着网络承载功能的不断扩展，路由器的转发单元变得臃肿不堪。网络中部署的设备和协议在相当长的时间内不会发生变化。这种冗余且难以维护的网络必然难以满足未来的需求。网络经过不断发展，进入了高速变革的时代，未来网络将具有灵活性、扩展性和简单性的基本特性[6]。

软件定义网络（SDN）的出现为这一问题提供了解决思路。SDN在斯坦福大学的Clean Slate项目中被提出，并且迅速成为了行内的焦点。随着研究的不断深入，该技术得到了各方的支持和认可，并且成为了未来网络发展的主流方向[7]。

SDN将网络设备的转发和控制进行分离，将设备中的控制权放在集中的控制器中进行管理，减轻了网络设备的负担[8]。同时集中式的控制屏蔽了底层的网络设备的差异。而且采用可编程的方式开发控制器，用户可以针对自己的需求对网络进行按需配置，更加灵活和自由。而当SDN与网络虚拟化相集合时，逻辑网络可以根据业务需求进行配置和迁移，不受物理网络的限制[9]。

15.1.1 SDN基础架构

SDN的架构方式有很多种,其中ONF的SDN架构图最为经典。如图2-1所示，SDN的典型架构共分三层，自顶向下分别为应用层，控制层和转发层[10]。

最底层为控制层，即SDN处理集中逻辑的层面，主要是处理底层交换机传递而来的转发信息。主要由服务器集群组成，其主要功能是进行拓扑管理，流量统计，南北向的接口管理等等。控制器会完成抽象网络资源到实际的底层网络资源的转发的转换。控制器接受上层应用层转来的资源申请和能力调用，转化为底层网络的转发策略，通过OpenFlow协议下发至对应的物理资源中。

转发层的能力可以被定义为为底层连接设备提供管理和转发功能，对于上层其会提供灵活的可编程能力。上层通过调用提供的编程接口进行数据的按策略转发。底层连接的设备可以是物理交换机也可以是虚拟软交换机。

位于最上层的是应用层。通过对SDN的底层基础资源的分析和调度，各种网络应用可以根据网络需求在应用层进行开发，在体验SDN技术灵活性的同时体现出自身应用的优越性。整个应用层有两个接口，分别是南向接口和北向接口。北向接口通过HTTP描述性语言与控制层进行交互，例如restful接口。应用层为用户屏蔽了底层的具体结构差异，但是其需要根据业务和面向对象进行信息模型的定义和抽象。然而应用层的模型并没有标准化的规范，而这也将是未来的重点标准化内容。

15.1.2 OpenFlow协议

OpenFlow协议可以实现上层控制器与底层OpenFlow网络设备之间的通信。控制器通过向网络设备发送OpenFlow支持的消息来对对网络设备进行控制，网络设备通过Openflow协议将自身的信息反馈到控制器中。从而实现了控制器对于底层设备的掌控。

OpenFlow交换机主要由三部分组成：安全通道，交换机通过安全通道与SDN控制器建立连接；多级流表，用来匹配交换机收到的报文；组表，用来对流的动作与策略进行定义[11]。

OpenFlow协议支持的消息类型一共有三种，分别是Controller-to-Switch，Asynchronous和 Symmetric[12]。控制器通过这三种类型的消息与交换机进行信息交互，实现对下层交换机的整体控制，其中包括建立连接，消息处理和流表下发。在OpenFlow的交换机中，仍然有一张路由表控制交换机中的数据流量的转发，然而对于流表的生成维护和控制则完全由上层的控制器来控制[13]。这完美体现出了SDN的控制与转发分离的思想。而流表项被定义为策略动作和关键字组成的策略规则。在实际的项目服务中，管理人员可以通过对流表中的关键字进行配置来决定具体的转发规则和转发策略

如今，随着SDN的发展，越来越多的传输协议也被提出，但是由于OpenFlow的发展较为成熟且相关规范已经得到普通承认，基于OpenFlow来实现SDN仍然是未来网络发展的主流趋势[14]。

15.1.3 SDN应用场景

现今SDN的主要应用场景主要集中在各运营商、数据中心，还有一些政企客户和大型的互联网公司。

15.1.3.1 数据中心

数据中心中的常见网络需求主要表现在其存在大量的租户、流的转发路径复杂且多样化、各类虚拟机需要智能迁移和部署

、网络需要整体集中化的管理，另外还包含一些绿色工程等方面的工作[15]。

而SDN的控制与转发分离，逻辑集中的特点使得网络的控制权可以集中于一点，分布式的进行控制，满足了其对网络的集中化管理的需求。而利用SDN的网络开发能力以及网络虚拟化的特性，可以通过构建虚拟网络的方式进行智能机的部署，在进行网络迁移时也不用考虑底层的物理结构[16]。

在SDN的部署可行性方面，数据中心的运营和维护具有相对的封闭性，通常由其运营商进行统一的开发和部署，在改造时也可进行统一的规划，这也为SDN的部署创造了条件。SDN与数据中心的结合能够充分体现出SDN的优越性，也能让数据中心最大地利用SDN的优势。

15.1.3.2 政企网络

政企网络中最重要的一项网络指标就是需要保证网络的完全性和可靠性，因此需要对网络有集中管控的功能；同时由于政企网络中承载的业务量较大，网络复杂性较高，对网络设备的管理存在一定的难度；并且网络复杂后，整个网络的按需配置也成为了一个难点。

SDN使得转发与分离控制，网络设备中的控制功能被集成在控制器中，网络设备可以变为简单的插拔设备，而不需要考虑其中的逻辑策略，这使得网络设备变得简单；而集中式的逻辑控制也解决了政企网络中要求集中管控的需求；通过对简单设备进行集中管控即可进行按需的业务分发，提高了网络的灵活性[17]。

在SDN的部署可行性方面，政企中的网络主要由政企内部部门进行维护与管理，在部署SDN时，可以进行统一规划和改造，具有较高的可行性。

15.1.3.3 运营商网络

运营商网络中的网络结构较为复杂，网络状态主要表现为网络的整体范围大、覆盖范围广；运营商网络中需要保证网络的安全性；并且在网络情况中会出现由多个运营商共同运营等特点[18]。

而SDN含有转发与控制分离的特性，这可以实现底层物理设备的复用，较少底层物理设备的数量，降低成本；SDN的集中控制的特性可以实现大范围网络的集中和智能管理，从而提高虚拟网络的运行效率，带动网络的全局优化；SDN本身的虚拟化和多元化也有利于用户网络更好向人工智能，机器学习方向发展，增加网络服务内容与质量。

许多运营商已经开始对SDN进行了试验部署，例如德国电信已经在云数据中心等环境中试验部署SDN；而NTT也在美国和日本对SDN搭建了试验环境。

15.1.3.4 互联网公司

SDN的最终目的仍然在于开放上层的网络能力，为上层的应用所服务。互联网公司利用的正是这一点，它们将业务部署在SDN架构中，这也是SDN的重要应用场景。

SDN的北向接口，向上层的应用层提供标准化、规范化的网络能力，上层应用通过此接口进行底层网络资源的获取、分析和调配，从而完成业务流程的处理。且上层完全屏蔽了底层的网络细节，使得应用层的处理更加清晰，提升了用户体验。

国内大型企业如腾讯、百度等都在加速SDN的实验室部署，利用SDN实现Qos等服务，最终实现整体服务质量的提升。

15.2 ONOS控制器

15.2.1 ONOS的产生背景

随着SDN的发展，基于SDN技术开发的开源控制器越拉越多，然而他们中的大部分缺乏可扩展性、可靠性，抽象层的过于简单性和性能上的缺陷使得他们并不能用于商业化产品。而ONOS把握住这些对于SDN控制器平台所需的关键特性，形成了一个一体化的商业级网络操作系统。

15.2.2 ONOS的核心架构

ONOS将服务提供商放在首位，它具有较强的可靠性和灵活性，并且具有良好的性能。同时它还配备了强大的南向接口和北向接口。整个ONOS架构一共包含四个核心，分别是北向接口抽象层、分布式核心、南向接口抽象层、软件模块化[19]。如图2-2所示。

15.2.2.1 北向抽象层

ONOS的北向抽象层是面向上层应用，主要分为Intent架构和全局网络视图。Intent架构从上层的租户意图出发，屏蔽了底层网络的细节，使得上层的服务可以在不了解底层运行细节的情况下直接请求服务。

全局网络视图为应用提供了当前的虚拟网络和物理网络整体情况，包含了其中的设备信息[20]。应用利用全局网络视图层提供的接口进行编程获取网络识图的信息。

整个北向抽象层利用接口将应用程序与底层的网络细节进行隔离，并且可以对网络事件进行隔离；而从网络操作系统的角度看，网络操作系统与应用进行了隔离，使得其可以管理来自多个应用的请求；从业务的角度看，提高了应用开发的速度。

15.2.2.2 分布式核心

分布式核心是ONOS架构特征的关键，分布式核心对上层的应用提供组件。在此种架构中，每个底层设备都会被分配连接到一个关键的内部组件，当这个组件出现故障时，设备会主动选择另外一个网络组件进行连接，此过程无需人为干预，实现了自动化的故障处理。这种方式使得系统的稳定性和可靠性得到了提升，并且还提升了系统的故障处理效率。

15.2.2.3 南向接口抽象层

ONOS的南向抽象层通过对象的方式对网络中的常见网络元素进行表示，分布式核心层通过抽象层抽象出来的简单对象进行网络的操作和维护。也就是说南向抽象层为分布式核心屏蔽了底层的物理细节。而南向抽象层也具有良好的扩展性，它能自动

识别设备的插拔，通过插拔设备进行设备和协议的添加，从而实现对多种设备的管理和控制。

15.2.2.4 软件模块化

软件模块化是ONOS一大结构特征，ONOS的软件模块化使其能够方便得进行软件的添加、改变和维护。分布式核心平台的三层架构作为ONOS的主体架构也反映了ONOS的软件模块化特性，核心平台能够约束其中的子系统中的规模并且保证模块的可扩展性；从接口的角度来说，模块分别连接到不同的接口，保证每个模型能够不依赖与其他模型而进行独立更新，并且不会影响ONOS的整体系统，确保了软件能够进行稳定的更新。 ONOS通过树形的结构来控制这些模块之间的关系，更是加强了这种结构原则。ONOS对模块的大小进行适当的控制，并且在模块与模块之间通过API相关联，保持适当的依赖关系，使其成为一个非循环性的结构图。

软件模块化保证了系统结构的具有完整性和连贯性；简化了系统的测试结构，使得系统可以进行更多的集成测试；使得组件之间具有良好的可扩展性和灵活性；降低了系统软件和模块的维护难度。

15.3 Intent NBI模型

SDN的北向接口差异较大，研究者希望能规范北向接口，并且用一种通用的语言对网络接口进行描述。在此背景下，多个企业组织开始设计与研发一种新的SDN北向应用程序接口[21]。该种应用程序只需要指定业务需求，而无需关注底层的网络细节，这种新的应用程序接口被称为“基于Intent的北向接口模型(Intent NBI)”。在这种应用程序接口模型中，由控制器决定了租户Intent被映射到底层网络设备中的具体方式，使得网络最后呈现为租户期望的状态[22]。租户在利用这种模型时不需要考虑底层的实现技术和实现细节，而需要关注自己想要什么。它是一种基于上层的模型，从租户的需求的角度进行模型抽象，最后才考虑上层与底层的映射策略问题[23]。这种模型使得应用屏蔽了对底层网络网络的细节，用户在使用时易于上手。该种模型接口与平台的具体接口无关，具有良好的可移植性。许多研发小组开始与标准组织和开源社区一起工作，促进此接口模型的规范化和通用化的开发[24]。

由于不同的北向接口方案面向的用户和面向的业务不一致，导致开放北向接口差异较大，用户很难将它们全都完全掌握，因此关于北向接口的规范化制定将成为SDN北向接口研究中的关键[25]。接下来将介绍三种具有代表性的北向接口模型。

15.3.1 OpenStack上的Congress

Congress的面向对象是云平台的管理用户，它对租户提供对网络整体进行管控的策略和服务，使得租户可以利用它对网络设备的资源进行策略下发和安全性监控等功能，并且以此来保障上层的应用业务被合理自动部署[6]。

在以前进行策略的配置需要进行手动配置，例如需要通过发送邮寄的方式将程序添加到网络，通过防火墙的过滤保证信息安全，然后才能被加入到特定的存储中。而这种方式很难对多用户的请求快读响应。

Congress认为云服务可以展示为“表”，它利用OpenStack中各个服务的底层表作为策略的抽象体，在租户进行策略制定时，通过对底层表的操作进行策略的制定[26]。

服务名称数据表

Nova servers,flavors,floating ips,hosts

Neutron networks,subnets,ports,routers,security group

Cinder services,volumes,snapshots

Keystone users,tenants,role

Glance images,tags

15.3.2 Opendaylight上的GBP

ODL上的GBP将底层的资源抽象为端点模型，端点模型中所包含的网络元素可以是一个独立的主机也可以是一个网络中的网络组件。它为端点设置了组模型，及可以将相同属性的端点定制为一个组，并且采用协议来描述组与组之间的策略规则。GBP的抽象数据模型可以被描述为

Opendaylight上的NEMO

NEMO是华为提出的一种基于Intent的北向接口语言。它通过自己创建的类SQL的NEMO语言来对模型进行描述，整个NEMO架构中主要由两部分组成，分别是NEMO模型和NEMO引擎。

1.NEMO模型。NEMO模型主要由三部分构成对象、操作和结果。对象指网络中的资源和服务，操作是调整网络资源和服务的行为，结果用来限制网络资源和服务的最终状态。用户利用这些模型和NEMO语言进行服务的描述。NEMO还提供NEMO引擎以能够控制计算、存储和网络[27]。表示了ODL中的NEMO模型。

2.NEMO引擎。NEMO还提供NEMO引擎以能够控制计算、存储和网络。NEMO引擎主要由四部分组成租户管理模块、语法语义解析模块、虚拟网络构建模块和物理网络管理模块。通过这些模块它可以实现的功能是：

（1）为每个租户创建VN Space，并防止VN Spaces之间的冲突

（2）检查租户输入的语法

（3）通过特定的规则将意图映射成为虚拟网络

（4）根据物理网络的资源和状态将虚拟网络映射到物理网络中

15.4 其他相关技术

15.4.1 搜索服务器Elasticsearch

Elasticsearch是一个基于Lucene的搜索服务器，它为用户提供了一个分布式的全文搜索引擎并且是提供了Restful Api。它

采用多shard的方式保证数据安全，并且提供自动resharding的功能。它可以进行快速地存储，搜索和分析海量数据。

由于Elasticsearch的底层是基于Lucene的，在Lucene中存在一些概念和术语。Document是用主要数据源，用来进行索引和搜索，其中包含一个或者多个Field，Field中包含与Lucene交互的数据；Field是Document的一个组成部分，其中含有两个关键词name和value；Term表示一个搜索的不可分割的最小单元；Token中包含Term和这个Term在文档中的起始位置和类型。

而Elasticsearch本身也有一些关键的概念。主要分为数据层面和服务层面，从数据层面来说的概念有：Index

而对于Elasticsearch本身来说，也有一些关键的概念。从数据层面来说，主要四个概念Index,Document,Document Type和Mapping。Index代表Elasticsearch存储数据的逻辑区域，可以将它看做是关系型数据库中的一个数据库单元，一个Index可以构建在一个或者多个shard上；Elasticsearch中的Document代表存储的实体数据，类似于关系型数据库中的一行，document由多个field组成，不同的document中的同名field具有相同类型。Mapping用来存储field的相关映射信息，不同的document type会有不同mapping[28]。

从服务层面来说，Elasticsearch含有Node,Cluster,Shard,Replica的概念，分别表示server实例，多个node的集合，数据分片和shard备份。

15.4.2 BNF范式

巴科斯范式(BNF: Backus-Naur Form 的缩写)是由 John Backus 和 Peter Naur 首次引入一种形式化符号来描述给定语言的语法[29]。它不仅能严格地表示语法规则，而且所描述的语法是与上下文无关的。它具有语法简单，表示明确，便于语法分析和编译的特点。BNF表示语法规则的方式为：非终结符用尖括号括起。每条规则的左部是一个非终结符，右部是由非终结符和终结符组成的一个符号串，中间一般以“：：=”分开。具有相同左部的规则可以共用一个左部，各右部之间以直竖“|”隔开。

在本课题中对传统的BNF范式进行一定的扩展，扩展的语法规则如下：

<> 尖括号用于分隔作为语法元素名称的字符串。

::= 定义运算符。这在生产规则中用于将规则定义的元素与其定义分开。正在定义的元素出现在运算符的左侧，定义该元素的公式出现在右侧。

{} 大括号组中的包含元素应该被明确定义。

[] 方括号表示公式中的可选元素。括号内的公式部分可以明确指定或省略。

| 替换符。竖线表示，竖线右边的部分是竖线左边部分的可替换项。如果竖线所在位置没有被{}或者[]包含，则会为生产规则定义的元素指定一个完整的替代方法。如果竖线所在位置被包含在{}或者[]中，则它指定了最内侧的括号中的替代方案。

15.5 本章小结

本章对本课题在研究与实现时所涉及到技术进行了介绍。主要包括对SDN基础知识、ONOS控制器、北向意图接口（Intent NBI）模型的介绍。另外还有一些辅助性的知识例如NEMO引擎、搜索服务器Elasticsearch和BNF范式。为后续的研究和实现提供了基础。