毕业设计（论文）报告

题目　基于启动时网络流量的安卓应用识别

信息科学与工程　院（系）　信息工程　专　业

学　　号　04013212

学生姓名　刘丹阳

指导教师　秦中元

起止日期　2016 年 12 月 19 日至 2017 年 6 月 1 日

设计地点　南京无线谷 6205

东南大学毕业（设计）论文独创性声明

本人声明所呈交的毕业（设计）论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得东南大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

作者签名：_____

日 期：_____

东南大学毕业（设计）论文使用授权声明

东南大学有权保留本人所送交毕业（设计）论文的复印件和电子文档，可以采用影印、缩印或其他复制手段保存论文。本人电子文档的内容和纸质论文的内容相一致。除在保密期内的保密论文外，允许论文被查阅和借阅，可以公布（包括刊登）论文的全部或部分内容。论文的公布（包括刊登）授权东南大学教务处办理。

作者签名：_____

导师签名：_____

日 期：_____

摘　要

通过网络流量分析识别 Android 应用的能力在许多领域有着重要的影响，如流量管理，恶意软件检测和用户隐私保护等等。目前普遍采用的应用识别策略为深度包检测技术，或是从 HTTP 报文荷载中提取应用指纹，这些策略在流量被加密时是无法使用的。然而在这种条件下，数据包的长度仍可以作为应用识别的特征，因此本文研究通过 Android 应用启动时流量的长度来识别应用。首先在 Android 系统中重复启动数十个常见应用，并捕捉它们启动时所产生的网络流量，对数据进行预处理之后，使用了本文所创新的区分方向的聚类特征方法来做特征工程，之后使用有监督的机器学习方法来识别产生流量的应用，并对所训练出的各分类器进行性能评估。

结果显示不同的学习算法被同一数据集训练之后，通过分析启动时流量长度特征，常见的 Android 应用可以以 90% 以上的正确率被识别，在随机森林算法中识别率高达 99.73%。当被测试的数据集来自不同设备，不同供应商或不同日期时，识别率并不会产生下降。

关键词： 流量分析；应用识别；机器学习；安卓应用

Abstract

The ability to identify Android applications through network traffic have a huge impact in many fields, such as traffic management, malware detection and privacy security. Currently the common method of identifying application is deep packet inspection or extracting fingerprint of application from HTTP message. But these methods are useless when facing encrypted network traffic. In this case, the length of HTTP traffic can be a effective feature to identify Android apps. In this paper we investigate whether the Android apps can be identified by the length of their launch time traffic . Firstly we capture the launch time traffic of 50 popular apps by repeatedly running them. Then the redundant information in the original data is deleted and then the original training set is converted to the standard data format which is accepted by Weka, a machine learning software. Supervised machine learning algorithm is used to train the classifier, including naive bayes algorithm, decision tree algorithm, random forest algorithm and support vector machine algorithm.

The result indicates that popular Android apps can be identified with more than 90% accuracy by using the length of their launch time traffic. The accuracy reaches up to 99.73% when using random forest algorithm. Moreover, the identification accuracy does not decrease when training set is from different dates, different devices or different mobile providers.

KEY WORDS: Application Identification; Network Traffic Analysis; Machine Learning; Android Applications

目 录

1 绪论

1.1 研究背景

随着手机移动平台的发展，我们的生活已经发生了翻天覆地的变化。而在如今繁多的手机平台中，Android 平台无疑是最常见的手机系统平台。2015 年 9 月谷歌在发布会上公布，如今在全球范围内激活的 Android 平台数量已经到达 14 亿台。除此之外，在智能家电市场里，包括电视、冰箱、相机、摄像头等，到处都有安卓的影子。在 2017 年 4 月 12 日，国际数据公司（IDC）发布了全球手机季度追踪报告。数据显示，在 2017 年第一季度，全球智能手机厂商总出货量到达 3.47 亿台，同比增长 4.3% 。Google 的 Android 系统依然在统治着整个移动设备系统市场，如今在中国，Android 系统取得了 9.3% 的年增长率，截止到 2017 年 1 月，市场占有率已经达到了惊人的 83.2% 。对于移动平台而言，是其之上的各种应用组成了整个平台的生态圈。近年，Android 应用数量已达 150 万款，超过了 iOS 成为最大的应用生态圈。

与此同时，通过分析应用产生的 TCP/IP 流量来分辨应用的问题受到了广泛的关注。这种现象有两个主要的原因：首先，通过网络流量来识别客户应用的能力在一些领域内是十分有用的——包括恶意软件检测，流量管理，网络容量规划和了解用户的隐私泄露程度。其次，随着加密和压缩技术被广泛接受和互联网隐私立法的完善，HTTP 荷载变得越来越不易被流量监控者接触，这就使得仅通过 TCP/IP 头部来识别应用的方法是最为有效的。

上述的趋势在 Web 页面的识别中已经被充分的认识。事实上，Web 客户在访问 Web 页面时会产生流量，通过这些流量的 TCP/IP 首部来识别应用的方法在过去几年中已经有了重大的突破，然后，在识别移动应用的领域只有很少的相关工作。尽管已经有一些初步的通过少量应用收集到的信息，但是还没有对应用识别方面综合的学习。这也正是本文研究的重点所在。

1.2 国内外研究现状

由安卓应用所产生的网络流量对其进行识别的问题受到广泛的关注。在文献[1][2][3][4] 中提出，基于流量的应用识别在许多领域有着重要的地位和前景，例如恶意软件检测，流量管理以及保护用户隐私以免泄露。在文献[5] 和文献[6] 中指出，随着加密和压缩技术的广泛应用以及人们对互联网隐私的重视，当前 HTTP 报文荷载越来越不易获得，这就使得通过网络包的长度来识别移动应用是更为有效的。这种趋势在 Web 页面识别中已经引起广泛的重视，文献[4][7][8] 中指出，Web 客户在访问 Web 页面时会产生流量，通过这些流量的 TCP/IP 首部来识别应用的方法在过去几年中已经有了重大的突破。然而这种方法在移动应用领域还未得到广泛的关注，仅有一些微小的工作，例如文献[9] 和文献[10] 中收集了 40-70 个安卓应用并对其进行流量分析。

在当前的移动安卓应用识别中，基于深度包检测（DPI）并分析 HTTP 报文首部来提取应用的指纹是最为常用的方法。然而，这些策略在 HTTP 流量被加密的情况下是无法使用的，因此通过数据包长度的应用检测的优势就体现了出来。

许多应用程序在启动时立即使用网络有几个原因。他们从广告网络中检索新广告或向分析服务发送信息。他们还执行应用程序特定的沟通——例如，电子邮件应用程序在启动时可能会检查新的电子邮件。先前在非移动流量分类化的领域中建立的结果表明，在这样的启动时业务内观察到的分组大小可以产生用于应用标识的良好特征集。具体来说，文献[12] 表明，TCP 流的第一个数据包可以用于将 Internet 应用程序分类为 Web，FTP 和游戏等类别。更相关地，文献[8] 示出了可以仅使用分组大小来执行网页识别。

同时，应用程序在启动期间生成的数据包数量取决于互联网速度和应用程序的应用程序逻辑。两个原因促使我们最小化用于应用程序识别的数据包数量。首先，较少数量的要处理的分组意味着用于识别方法的较少的计算资源需求。第二，更重要的是，用户可以开始在应用中执行动作，诸如在应用的用户界面可用之后立即点击按钮或输入文本。这样的动作可以产生额外的网络流量，其将对应用的启动时间流量增加噪声。通过最小化启动时网络分组的数目，我们确保用户动作的最小干扰，最终确定最佳数据包量为 64 个。

针对本课题的机器学习方法而言，分类器的选择是至关重要的。在文献[11] 中所用的方法是：首先将 TCP 连接中的连续输入或输出数据包分组定义为“突发”，给定一个流量样本，从 TCP 连接中提取输入突发大小，并将它们中的每一个舍入到最接近的 32 个字节，将流量样本视为使用 Jaccard 的系数测量两个样本之间的相似性。通过在训练样本中找到最相似的样本来识别测试样本。

在文献[8] 中，给定了一个流量样本，提取它的数据包大小。一个流量样本被视作多个数据包的总大小。用高斯朴素贝叶斯分类器进行分类。

在文献[7] 中则是给定一个流量样本，首先如方案二一样提取数据包的大小。然后利用 TF-IDF，一种用于信息检索与数据挖掘的常用加权技术，和归一化被应用于特征向量。将多项朴素贝叶斯分类法用于分类。

1.3 研究内容与意义

Android 平台是移动端最大的市场，而 Android 应用识别事关恶意软件检测，流量管理及保护用户隐私等诸多 Android 平台关键部分，因此 Android 应用识别的研究有着重要的意义。当前所常用的 Android 应用识别大多是基于深度包检测（DPI）的，而在 HTTPS 及诸多流量加密技术日益流行的今天，对流量包的内容进行解读变得越来越困难，但是在大部分加密的场景下，如流密码加密，数据包的长度是不会发生改变的，因此使用数据包的长度作为特征是可靠且值得研究的方向。更具体的讲，本文是讲 Android 应用启动时与服务器通信的数据包长度作为特征的。之所以选择启动时流量，是因为每一个 Android 应用在编写时，其网络交互模式是确定的，即不同的应用在启动时往往会有不同的网络交互表现，具有一定的稳定性。除此之外，在应用启动之后，由于每个用户的操作不同，输入不同，造成服务器对应用的反馈不同，因为启动后的流量具有不稳定性，不适合作为 Android 应用识别的特征。

确定特征之后，本文选择了机器学习方法作为分类手段。机器学习是当前计算机界炙手可热的方向，其在预测、回归、分类、聚类等方面都表现出了非常优秀的性能，因此我们选择机器学习方法作为本次毕业设计的分类方法。更具体的讲，由于所采集的每一个 Android

应用启动时流量样本都对应于一个固定的分类，所以属于有监督的机器学习分类方法。后文中将使用各种不同的分类方法，如朴素贝叶斯，决策树，随机森林和支持向量机等等，并比较这些热门的分类算法在应用识别方面的表现，分析各种差异所产生的原因。

最后本文中把基于 Weka 训练的分类器实现为 Java 编写的独立分类器，以形成独立的程序，且可以作为接口用于其他的场景中。然后本文会分析对分类器产生影响的各种不同场景，例如不同日期，不同供应商，不同设备，不同版本等情况下分类器能否依然正确识别出应用，即检测应用的鲁棒性。由于现实的应用环境总是比实验室中复杂，所以这一步也是必要且具有重要意义的。

1.4 本文组织结构

这篇文章的组织结构遵从如下安排：第一章为绪论，主要介绍了关于 Android 目前的发展现状和基于流量的识别技术在国内外发展的现状，对比了国内外相关文献在这方面使用的不同方法，包括不同的提取特征方法，不同的分类器及不同的分类思想。同时，也介绍了基于流量的应用识别技术的研究内容和意义，如上文所讲，在恶意软件检测，流量管理及保护用户隐私方面都有重要的意义。

第二章为技术背景，交代了在我的毕业设计中会用到的所有相关的技术背景。首先介绍了 Android 应用的通信机制，即应用如何与服务器之间进行交流，这是识别应用的基础。其次，要通过网络流量来识别应用，就必须了解网络流量的特征，包括在 TCP/IP 模型下每层流量的特征。本文使用的应用识别方法是基于机器学习中的有监督的分类方法，故在第二章会介绍关于有监督的机器学习分类的背景内容。同时，我的毕业设计是基于 Weka 软件来进行的，在第二章也会介绍该软件的基本情况。

第三章讲了整个应用识别功能的实现过程。按照流程来分，可以分为四部分：一是网络流量的采集，将 Android 应用启动时的流量采集下来；二是数据预处理，即将一手的原始数据经过处理，变成可以作为输入的标准格式的数据；第三步是特征工程，即在经过处理的数据中选取作为分类特征的某些特征，这是影响识别正确率最重要的一步；接着是分类器的选择，目前常见的分类器有朴素贝叶斯分类器，支持向量机分类器，决策树分类器和随机森林分类器等等，本文会比较不同分类算法下分类器的表现；最后是将基于 Weka 实现的分类器独立出来，形成独立的代码和程序。

第四章将本次分类的测试结果分析。在本章，我们将考虑各种不同的因素对分类器产生的影响。首先是不同日期的数据对应用识别率的影响，因为许多应用在不同日期推送的内容会有差异，因此会对网络流量产生影响；其次还会讨论不同运营商、不同设备及不同的应用版本对应用识别率的影响，以检测我们的分类器的鲁棒性。

最后一章为总结与展望，在这一章里将会总结我在毕业设计中所做的工作及成功，并对本次设计的应用前景及 Android 应用识别领域的发展前景进行讨论。

2 技术背景

2.1 Android 应用中的网络通信机制

Android 应用的网络通信机制一般分为三种：Socket 接口，HttpURLConnection 接口和 HttpClient 接口。而基于 Socket 接口的网络通信现在已不常用，常用的是 HttpURLConnection 接口和 HttpClient 接口方法。

首先要明确的是，二者都是基于 HTTP 协议的通信方法，在 HttpURLConnection 方法中，Http 通信中的 POST 和 GET 在请求方法上有所不同。GET 请求往往用于获得静态的页面，我们也常把参数放在 URL 字符串的后面来传递给服务器端。而 POST 的请求方法则是在 Http 请求中写入参数。因此在编写 HTTP 程序中，应该先确定要用哪种通信方法，然后根据所选择的具体方式来选择相应的编程方式。HttpURLConnection 是 URLConnection 类的继承类，二者都为抽象类。往往用 URL 的 openConnection 方法来获得其对象。要注意的是，HttpURLConnection 默认使用的是 GET 方法，如需设置 POST 方法，则需要使用 setRequestMethod 语句来设置。

Apache 提供的 HttpClient 方法也可以用于 Http 操作。在这种方法下对于 GET 和 POST 的操作也有所不同。在使用 POST 方法时，需要传递的参数的传递需要通过 NameValuePair 来保存。事实上，HttpClient 是对 Java 所提供的方法的一种封装。在 HttpClient 接口中所统一封装的 HttpPost 和 HttpResponse，事实上就是上段中 HttpURLConnection 中所描述的输入输出操作，以此大大减小了操作的繁琐程度。

对比 HttpURLConnection 接口和 HttpClient 接口，虽然 HttpClinet 接口支持更多更细节的操作，如压缩、连续池、代理、COOKIE 等等，但是相应的，这对 Android 开发人员提出了更高的要求，虽然对细节的操作更简单，但是代码的复杂程度比 HttpURLConnection 方法要高不少，普通人员要驾驭起来很难，官方也越来越少的提供对这种方法的支持。而 HttpURLConnection 方法已经包括了大部分我们所常用的网络操作，屏蔽了冗余的复杂操作，更适合开发人员直接调用，目前官方对 HttpURLConnection 方法的优化也越来越好，因此在 Android 编程中往往使用这种方法更多。

2.2 HTTP 协议

无论是 HttpURLConnection 还是 HttpClinet 方法，都是基于 HTTP 协议的。我们当前使用的网络多为 TCP/IP 体系，整个网络结构分为四层，与 OSI 七层模型的对比如图2.1所示。

HTTP 协议在 TCP/IP 分层模型中属于应用层的协议，全称为 HyperText Transfer Protocol，即超文本传输协议。OSI 七层模型与 TCP/IP 四层模型都是分层模型，在这种模型中，上层都是调用下层的功能，下层集合其下所有层一起对上层提供服务的接口。对于 HTTP 协议而言，其功能是建立在下层的 TCP 协议，IP 协议和物理接口层的功能之上的。

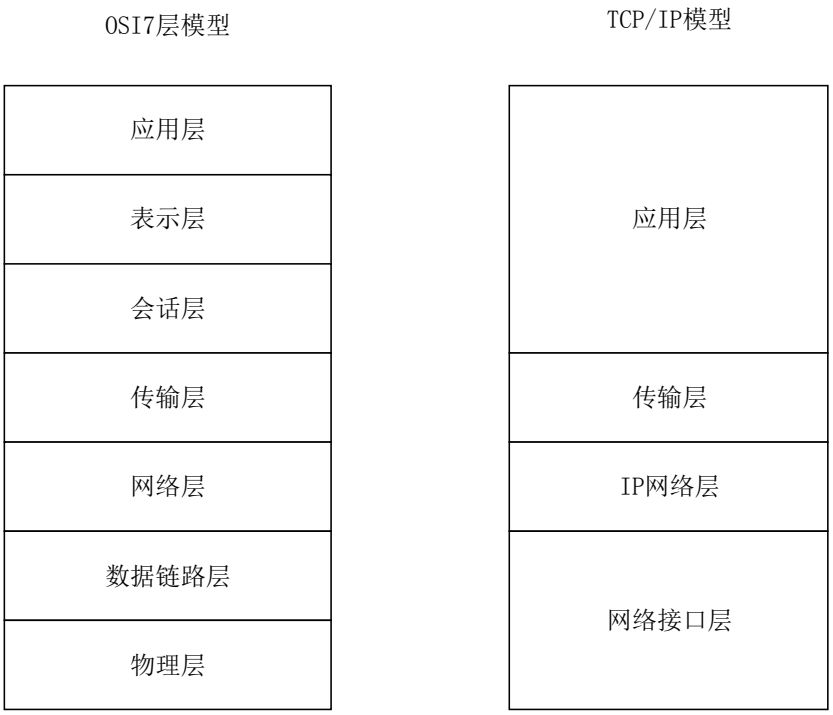


图 2.1 TCP/IP 模型与 OSI 模型对比

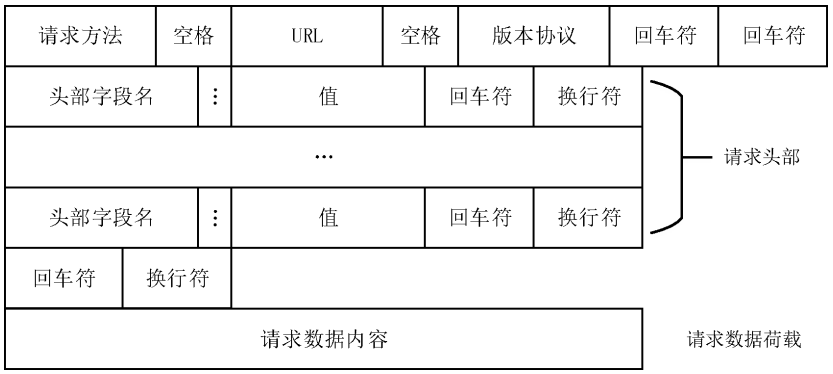
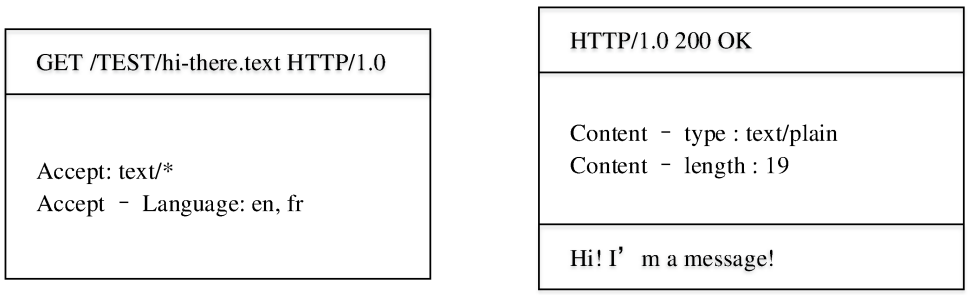


图 2.2 HTTP 报文结构

HTTP 报文是由一系列 ASCII 码组成的，也就是说 HTTP 报文是面向文本的。HTTP 报文可以分为两类：请求报文和响应报文。

- 请求报文：HTTP 请求报文的结构如图2.2所示。主要包括请求行，请求头部，空行和请求数据四个部分。其中，请求头一般由三个字段组成，分别是请求方法，URL 和 HTTP 协议的版本。他们之间以空格分隔，例如，GET /main.html HTTP/1.0，意为请求获得 maim.html 文件，且使用 1.0 版本的 HTTP 请求。请求头部的格式为关键字/值对组成吗，每对占一行，值和关键字之间用英文的冒号“:”所间隔。请求头部用来通知服务器有关客户端所发出的请求信息，通常包括产生请求的浏览器类型，客户端可以识别的内容类型集合和所请求的主机名。空行的作用为告知服务器，此处以下不再有请求头部。而请求数据仅在 POST 方法中出现，因为 POST 方法用于用户要提交数据给服务器，所要提交



（a）请求报文 （b）响应报文

图 2.3 响应报文和请求报文

的数据就填写在这个部分。在真实通信场景中最常用的请求方法是 GET 方法和 POST 方法。

- GET 请求：这是最常见的一种 HTTP 请求报文。例如在我们点击某个链接时，浏览器向服务器端发送的就是 GET 报文，以获得所请求页面的具体内容，并以此来构建页面显示给用户。服务器相应 GET 请求时会把构建页面所需要的数据放在响应报文的数据部分，发送到客户端。在 GET 方法中，所请求的参数和对应的值放在 URL 后，URL 的结尾处以问号“?”来表征，问号后为请求参数。例如：/main.jsp?id=211&op=bind，从中可以看出，问号后为 GET 所发送的请求数据，各个数据之间用“&”符号隔开。

- POST 请求：与 GET 请求相比，POST 请求支持更长的数据长度，在 POST 方法中客户端可以给服务器发送更多的数据信息。在 POST 方法中，格式为名称/值的形式。POST 方法的数据字符串不会显示在请求行中，而是会保存在“请求内容”部分，数据间隔符也是“&”符号。通常 POST 会用于页面提交表单的请求，且 POST 也能完成 GET 所能完成的任务。但在具体设计时使用哪种方法还是要具体情况具体分析，不要盲目的选择 POST 方法。

- HEAD 请求：HEAD 请求只要求服务器接受后返回响应头部，而相应内容部分为空。在我们只想要查看某个页面当前的可达性状态时，使用 HEAD 请求是一种十分高效的方法，因为在传输过程中省去了页面内容。简言之，HEAD 请求就像 PING 请求一样，用来检测某个页面当前的状态。

• 响应报文：HTTP 响应报文由状态行，消息报文和响应正文三个部分所组成。与请求报文相比较，只有第一行不同。在响应报文中这一行是状态信息，而在请求报文中这一行是请求信息。状态信息通过一个状态码来说明所请求的资源情况。状态行的格式为“HTTP-Version Status-Code Reason-Phrase CRLF”，其中第一部分为 HTTP 协议的版本信息，第二部分为服务器所发回的响应状态代码，第三部分为状态代码的描述，以文本形式呈现。最常见的状态码为 200 OK，即客户端请求成功。

2.3 有监督的机器学习分类方法

当代计算机界已经掀起了机器学习的潮流，其意在从大量的已知数据中提取出隐含在背后的，之前不被人们所发现的，且有价值的潜在有意义信息。机器学习的主要工作是分析和设计一些可以让电脑程序自身进行“学习”的算法，这类算法是让计算机自己从数据中获得直接观察不易得到的，深层次的规律信息，以此来发觉数据背后的有用信息。所以，机器学习是一种新的编程方法，不需要人们来总结经验和输入逻辑。人类只需要把大量的输入数据给计算机，然后计算机自己就可以根据算法和逻辑来总结，归纳和挖掘数据背后的规律和逻辑。这个过程就叫做训练，训练后可以得到一个模型，之后同样的工作都可以交由这个模型来代替人工去处理，一个输入样本足够准确和广泛，且训练算法完备的模型，可以做出比人类更加准确的判断。

机器学习总的来说可以分为有监督的机器学习和无监督的机器学习。

- 有监督的机器学习：所谓有监督的学习，就是说每一个数据都有对应的一个标签，计算机程序可以从数据中得知该情况下的正确输入应该是什么，我们希望在这种样本的训练之下，所训练的模型在面对没有对应标签或者说答案的数据下也能给出正确或者靠谱的答案，从而答案对未知预测的目的。根据数据的输出结果是连续还是离散，有监督的机器学习可以继续分为分类问题和回归问题两类。举个简单的例子，判断一个肿瘤是不是恶性属于分类问题，判断房价在某一天会涨到多少就是回归问题。它们在语音处理，自然与原处理，图像识别，垃圾邮件判断和拦截，网页检索，股票预测等方面都有着重要的意义。

- 无监督的机器学习：与有监督的学习相比，无监督的学习中，样本集里的每一个数据样本都没有对应的正确输出结果信息。无监督的机器学习希望从这些没有标签的数据中，找出其中有共性的相同或相似的样本，比如判断在哪一个样本点处数据聚集，哪些样本特征的出现频率更高等等。无监督的学习希望从大量的无标签数据中挖掘出其中有意义的信息，常见的例子有聚类，寻找关联规则，离群点发现等等。

显然，在本次毕业设计中，我们用到的是有监督的机器学习，更具体的讲，是基于有监督的机器学习的分类操作。本文中，将要采集的数据集是众多 Android 应用启动时所产生的网络流量，从流量中提取这样或那样的特征，以输入分类器进行分类。由于对于样本集中的每一个流量样本，其所属的 Android 应用都已经被我们所知，所以属于有监督的机器学习模型。

2.4 Weka 简介

Weka 全称为怀卡拓智能分析坏境（Waikato Environment for Knowledge Analysis），取其英文名称首字母组成 Weka，发音类似于新西兰国所特有的一种不会飞翔的鸟类，如图2.4所示。因此 Weka 也取这种鸟类为其公司的标志。

Weka 是一款基于 Java 语言所编写的机器学习和数据挖掘分析软件，也是 GNU 协议下所分发的开源软件，是一套数据分析的完整工具和系统，包括数据的预处理工具，各种不同的学习算法和对于结果的评估方法。同时，Weka 有可视化的图形用户界面，易于用户操作。在 Weka 系统中，各种最先进的机器学习算法和数据预处理的工具都汇集于其之中，以此使

图 2.4 Weka

用户有能力便捷灵活的将各种前沿的数据处理算法用于所获得的最新数据集中，以获得其背后所蕴藏的规律和逻辑。同时，Weka 为机器学习或是数据挖掘的各个阶段都提供了完备的支持，包括一手数据的预处理方法，集成了各种前沿高效的机器学习算法，以及对模型的评估方法。通过可视化界面，用户可以便捷的操作不同组件，比较不同算法的效果，以获得问题的最佳解决策略。

Weka 系统中囊括了数据挖掘和机器学习中包括非监督学习和监督学习中所有常用的方法：回归、聚类、分类、关联原则以及特征的选择。对于待将处理的数据进行预处理是机器学习中重要的一个环节。Weka 系统中提供了许多数据预处理的方法，且完全基于可视化界面，方便灵活。Weka 可以接受的数据类型一般分为两种，一种是以 ARFF 文件为代表的文件；另一种方法是直接读取数据库表中的元素数据。

Weka 一般用于以下几种工作场合：第一种是将已有的机器学习算法用于某个用户获得数据样本集，然后从输出中分析其中逻辑和规律，以获得对原始数据更深层次的了解；第二种是将上述第一种方法所得到的模型，用于预测某个未知样本，来得到一个可靠的预测答案；第三种是用各种不同的机器学习方法对于同一个样本数据集进行处理，然后比较不同算法所表现出的性能，以获得针对该问题最佳的学习算法，不仅如此，大部分的学习算法都带有由用户调节的参数，通过调节同一算法的不同参数也可以获得不同的性能，然后对于性能进行评估。

Weka 的主界面部分叫做 Weka GUI 选择器，通过其右侧的四个选项来提供机器学习所能用到的四个主要应用程序。鼠标点击各个按钮即可进入到相应的图像用户界面。

在四个部分中，最为人们所常用的用户图形界面叫做探索者（Explorer）。通过图形界面上册不同的标签选项和界面上的表单填写，可以实现 Weka 提供的所有机器学习常用方法。例如，数据集的导入仅需要用户去单击几个按钮，即可从 ARFF 或其他数据文件中读取数据样本集，然后建立随机森林的分类模型。Weka 拥有有好的交互操作界面，某功能在当前不可选择时，按钮会适时地变成灰色；鼠标停留在某个按钮时，会给出该功能的使用提示；Weka 提供了许许多多的机器学习算法，在选择了算法后，所有关于算法的默认参数都是普遍适应的最佳参数；这一切都使得 Weka 可以实现更便捷的操作。

一个缺陷也存在于探索者窗口，即在每一次读取数据的时候，都必须把对应 ARFF 或 CSV 数据格式中的所有数据一次性都读入。因此，这种批量读入的方式不可以用于一个

ARFF 文件所不能存贮的情况，在这种情形下就需要使用知识流界面。

知识流（KnowledgeFlow）界面如上所言，可以用来处理大型的数据样本集，这一操作时通过增量方式的算法来实现的。用户可以自行定义处理样本数据流的方法和顺序。在知识流界面，用户在界面上任意的拖动代表不同学习部分的图形部件，并按照一定的次序组合在一起。总的来讲，知识流界面允许用户自行的组合数据源，数据预处理部分，机器学习算法，模型评估以及各个部分的可视化模块，形容了所谓的数据流。

实验者（Experimenter）界面旨在解决用户在选择分类器和回归方法中的参数问题，即对于一个问题来讲，用哪种分类器或回归方程，可以得到最好的结果？在实验者界面，用户可以比较不同的机器学习算法和方案。虽然在探索界面也可以完成这个操作，但是在实验界面，用户可以让计算机自动的测试不同参数、过滤器和分类器，使其重复运行在同一组数据集之上，实现比较和评估。

简单命令行（Simple CLI）界面为没有命令行操作的计算机系统提供了通过简单命令行进行交互的功能，在其中可以直接进行 Weka 命令。

2.5 本章小结

本章的第一部分介绍 Android 应用的网络通信机制。从 Android 应用的编写上来讲，调用网络通信模块可以用 HttpClient 接口和 HttpURLConnection 接口两种方式，前者默认支持 HTTP 协议的 POST 请求，而后者默认支持 HTTP 协议中的 GET 请求。在大部分情况下在 Android 编程中使用的是后者，虽然 HttpClient 方法具有更多更详尽的参数，但与此同时带来了更复杂的语句和调用方法；而 HttpURLConnection 接口方法已经涵盖了 Android 编程中常用的网络需求，且更简单易懂，不易出错。

本章的第二部分介绍 HTTP 协议。我们所使用的网络属于 TCP/IP 网络模型，由上到下分为应用层，传输层，IP 层和物理接口层，HTTP 层属于应用层的协议，往往用于 WWW 万维网中，同时 Android 应用也是使用 HTTP 协议。HTTP 报文可以分为请求报文和响应报文两种，常用的请求方法有 GET 和 POST 两种，其中 GET 用于获得某个页面的内容，POST 用来提交表单，同时也可以实现 GET 的功能，但是不能盲目的使用 POST 方法。响应报文用于服务器响应客户的请求报文，其中包含了构建页面的所有元素。

本章的第三部分用于介绍机器学习，更具体的讲是机器学习算法中的有监督的机器学习算法。机器学习可以分为有监督的学习和无监督的学习两种，二者的区别在于样本数据集中的每一个样本是否已有对应的标签。若每一个样本数据都有对应的正确标签，那问题就属于有监督的机器学习问题。而若数据集中每一个数据都没有对应标签，需要计算机使用机器学习算法来从数据集中自行的发现其中的逻辑和规律，这种问题属于无监督的机器学习算法。若一部分数据含有标签，另一部分没有标签，这种问题成为半监督的机器学习算法。本文要解决的问题是基于 Android 应用启动时流量的分类问题，因为每一个样本集都有对应的分类标签，所以属于有监督的机器学习问题中的分类问题。

本章的第四部分介绍 Weka 的基本信息。Weka 是本文所使用的机器学习应用软件，可以分为探索者、知识流、实验者和简单命令行四部分。探索者是最常用的功能，用于读取样本数据集，选择机器学习算法，训练模型和评估模型等所有常用的功能，但是其数据导入方法只适用于小规模的数据集，在面对较大的数据集时，我们往往使用知识流方法，在知识流中，我们可以读取大规模的数据集，同时通过图形窗口拖拽的图形部件的方法形成数据流，

以处理较大规模的数据样本。实验者界面用于帮助用户比较不同算法或同种算法下不同参数的性能和表现，在实验者界面 Weka 可以自动的运行不同参数不同算法的各种命令，省去用户重复操作的时间，大大的提高效率。简单命令行界面用于没有命令操作行界面的系统，Weka 通过这个部分来提供简单命令行界面用于用户与计算机的交互，可以直接执行 Weka 中的种种命令。

3　应用识别功能的实现

3.1 实现概述

本文把通过 Android 应用启动时流量完成应用识别这一任务分为五个阶段，如图3.1，首先采集 Android 应用启动时所与服务器通信所产生的网络流量，更具体的讲，本文所研究的基于有监督的机器学习算法下 Android 应用识别问题的训练数据集采集的问题。Android 应用在与服务器通信时会产生一系列的网络流量，而所有流量可以分为基于 Android 程序逻辑的和基于用户差异操作的，后者属于高噪声流量，因为不同用户的操作有着不可预测性，其网络流量对于应用的识别没有意义。所以本文的主要任务是抓取 Android 应用基于自身程序逻辑所产生的网络流量，这部分流量中，在 Android 应用启动时所产生的流量又是最容易控制抓取的，因为其后的周期性逻辑流量不易被抓取。所以本文要基于 Android 应用启动时的网络流量来进行应用识别。之后存在 Android 应用选择的问题，本文选择了安卓应用市场上最常用的 50 个应用，按照下载安装量排列，并且去除掉了与我们的设备，安卓版本不兼容的部分应用。对于网络流量的捕捉，本文采用了基于 Windows 的 Android 模拟器的方法来抓取流量，这种方法可以保证所有应用的流量全都经过电脑网卡，易于捕捉。若以主机作为移动设备的代理服务器方法存在时延不稳定等问题，对于流量的捕捉造成困难。利用 Android 模拟器的方法对于 Android 系统版本的更换、设备型号的更换、运行商的变换等问题都带来极大的便利。对于流经网卡的流量，本文使用 Wireshark 软件来抓取其产生的流量，由于流经网卡的流量分为 Windows 进程产生的流量，Android 系统与系统服务器的流量和应用产生的流量三部分，因此需要对前两种流量进行过滤。对于 Windows 进程所产生的流量，基于数据包的协议可以对其进行过滤，因为 Android 应用所产生的流量是 HTTP 协议，而在保证系统不打开浏览器的情况下，Windows 并不会凭空产生 HTTP 协议的流量；对于 Android 系统及其他非目标应用所产生的流量，本文通过目的 IP 地址进行过滤，因为这些流量都是定期与某固定服务器通信的，只要运行 Android 模拟器一段时间，将所有非操作下产生的流量的目的 IP 地址全部过滤即可。完成过滤规则后，启动应用同时开始捕获流量，此时不要人为操作，直到应用不再产生流量时停止捕获，这样即可保存下 Android 应用启动时与服务器通信的 CSV 格式纯净流量。

然后对数据进行预处理，包括将采集下来的数据转换为 Weka 所能识别的标准数据格式，比如 ARFF 和 CSV 格式，前者为 Weka 所特有的数据集格式，后者为 Excel 环境也支持的逗号分隔符格式；之后再对于数据集中的数据进行离散化与维度规约，考虑如何去掉冗余数据，如何将数据整合，以提高每个维度的代表性，或者提高每个维度的权重；对于所采集下来的流量数据，如何处理可以使其成为 Weka 所接受的标准机器学习数据 ARFF 格式。第一部分所抓取的 CSV 文件中有着大量的冗余信息，例如数据包的序号，协议类型，源/目的 IP 地址，时间和详细信息等等，本文所需要的特征是长度，因此除长度之外的信息全部删除，且需要将同一应用多次启动的样本全部集中到同一文件中。这部分操作本文编写了基于 VB

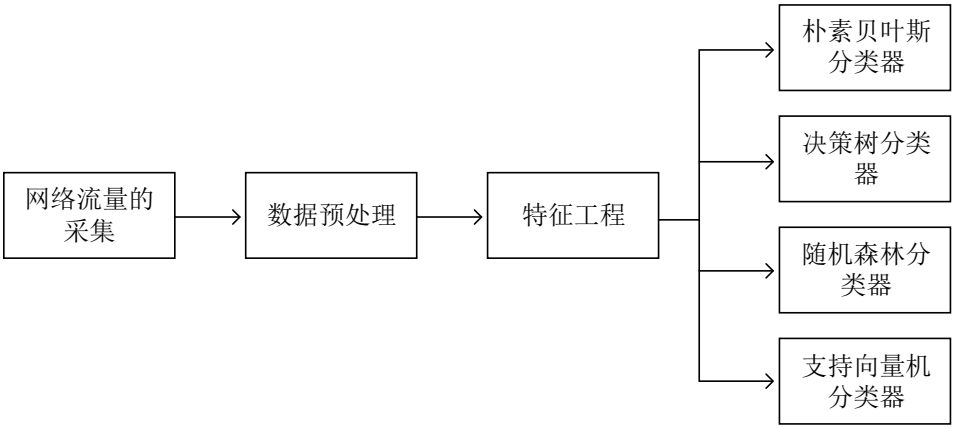


图 3.1 系统流程图

语言的脚本程序来自动化实现，以将 CSV 格式标准化为 Weka 可以接受的类型，之后将该 CSV 格式输入 Weka，在软件内可以完成 CSV 指 ARFF 的格式转换。保存的流量包的长度从 0 到 1500 不定，如果全部作为特征的话会导致分类器效率低下，且分类准确率没有保证。因此本文将长度属性进行维度规约，以 32Byte 为单位，将各属性降维，这样每一个属性有着更强的代表性，大大提高了分类器的效率和准确率。

之后进行特征工程，如何从数据集中选择输入分类器的分类特征，即特征工程。从样本集中提取用于分类的特征，本文所研究的特征是基于“启动时流量的长度”之上的，一切都由流量包的长度，或者说大小，推演而来；之后将处理完毕的数据输入分类器，在分类器的选择上，我们需要比较各个不同的分类器在该问题上的表现，取表现最优的分类器实现为独立分类器，这就完成了基于 Android 应用启动时流量长度来进行应用识别的任务。首先介绍了 TF-IDF 算法，即词频 -逆文档频率方法，词目即某属性的取值，文档即针对某一应用的样本，语料库即所有应用综合的样本数据集。词频指的是该词目在文档中出现的频率，TF 值越高，表示这个词目对于该文档有着更强的表征性，是更好的特征。而逆文档频率指的是整个语料库中包含该词目的文档比例，IDF 的值越低，表示这个词的表征性越弱，因为在整个样本空间内都是一个常见的词汇，没有代表性。所以整个词频 -逆文档频率算法倾向于选择出对于某文档更有代表性的词目，过滤掉在整个语料库中都常见的属性值。针对本文所研究的基于启动时流量的 Android 应用识别问题，启动时流量有着方向区别，即分为从服务器发向客户端的流量和从客户端发往服务器的流量两部分，而这两部分对于问题的重要性是不同的。由客户端发往服务器的流量往往是基于 Android 程序的内部逻辑，有着更强的稳定性和识别表征性，而服务器返回客户端的流量由于不同日期投放的广告，不同用户的兴趣信息不同等等因素，有着极强的不稳定性。因此需要将这两部分流量区分，本文采用正负长度值的方法来区分，即由客户端发往服务器的流量为正长度，由服务器返回客户端的长度为负长度，这样即可区分这两部分流量。经由 32Byte 单位化和正负长度值两步，就形成了本文所采用的独一无二的特征工程方法，本文称之为区分方向的聚类特征。

最后本文会介绍不同的分类器算法的特征。朴素贝叶斯分类方法是最简单也最常用的分类算法，其前提条件是数据样本集中的各个特征属性具有强的独立性，主要思想是根据贝叶斯定理，基于条件概率公式，条件之间互相独立和最大后验准则，选择概率最多的假设。决

策树方法是利用样本数据集生成一棵二叉树，每个节点成为分裂点，分裂的判别条件是根据某个属性值的分类，而生成一棵决策树最常见的方法是 C4.5 方法，其思想为计算出整个样本集的信息熵，之后计算每个属性对于分类的划分信息熵，二者作差求得信息增益，求得所有属性的信息增益后，计算各个属性的分裂信息，作为辅助条件。之后求出每个属性对于数据集的信息增益率，取信息增益率最大的一个属性作为本次循环确定的分裂点。之后递归调用这个算法，直到完成一棵决策树的构建。随机森林方法是指对于样本进行随机取样，得到许多样本子集后对于属性再次进行随机取样，针对这个样本子集进行决策树的构建，这样可以形成不计其数的决策树，称之为森林。待判别样本输入后，所有决策树的输出结果中最多的那一个，即输出的众数，作为随机森林的输出结果。随机森林有许多优点，比如不必对数据集中的特征进行挑挑选选，因为有随机取样的过程，且对于维度很高的数据处理有着算法上的优势；同时随机森林也有着缺点，比如当训练数据集中的数据特征属性有着不同的权重或复杂的优先级时，随机取样就会对分类结果产生偏差的影响，这种情况下不适宜使用随机森林算法。

3.2 网络流量的采集

3.2.1 Android 应用选择

在安卓应用平台 Google 市场上大概有 200 万个应用程序。我们依靠 Google 市场应用的排行榜来识别出 50 个到 2017 年 1 月为止最流行的（安装次数最多的）免费应用程序。接下来确定这些应用程序的子集与本文后续会用到的所有设备及不同的 Android 系统兼容。剔除掉部分与本文中测试设备和系统不兼容的应用后，本文所选择的应用如表3.1 所示。

表 3.1 选取的 50 个应用

360 浏览器	360 清理大师	360 天气	360 影音	iReader	QQ 空间
UC 浏览器	WPS	百度贴吧	暴风影音	大众点评	斗鱼
虎扑体育	金山词霸	酷狗音乐	酷我音乐	鲁大师	美图秀秀
搜狐视频	腾讯漫画	天猫	同花顺	我查查	喜马拉雅
小猿搜题	中华万年历	Bilibili	Keep	腾讯 QQ	QQ 音乐
爱奇艺	唱吧	今日头条	京东	聚美优品	快手
美团	腾讯新闻	腾讯视频	网易云音乐	唯品会	小红书
QQ 浏览器	高铁管家	人人网	虾米音乐	QQ 邮箱	链家
新浪微博	知乎				

3.2.2 网络流量的捕捉

Android 应用与服务器通信过程中会产生一系列网络流量，其中可以分为两部分：一是程序自身的逻辑产生的，Android 应用在定时定期与服务器通信的内容，比如当日所要投放的广告，对于该用户的兴趣推荐，以及对服务器上传该用户的 COOKIE 等等；另一部分是用户在操作过程中，基于用户的操作所产生的流量，比如用户点击某个模块，或者用户输入的内容需要上传到服务器等等，这一部分的流量是难以控制的，也是没有共性的，或者说，是噪声很大的，对本文中的应用识别任务几乎没有帮助。所以，基于 Android 应用逻辑所产生的流量更具有稳定性，更适合用来做应用识别用。因此，本文选取 Android 应用启动时与服务器通信所产生的流量作为样本。采用 Android 应用的启动时流量，一是因为在启动时没有用户各异操作的干扰，所有的流量都是基于 Android 程序编写时的逻辑所产生，具有稳定性；二是这部分流量易于确定和捕捉，只要在 Android 应用打开，且用户没有操作时所产生的流量，都可以认为属于 Android 应用的启动时流量，可以用于 Android 应用识别。在设备方面，为了简化本文的网络流量获取，本文选择了基于 Windows 平台的 Android 模拟器的方式来抓取流量，因为在这种条件下，所有流量全都流经计算机的网卡，易于捕捉。

对于网络流量的采集，目前常用的方法有 TCPdump，Fiddler 和 Wireshark 等等。本文使用 Wireshark 软件来完成 Android 应用启动是流量的采集。Wireshark 是一款免费的且开源的网络数据包分析软件，通常用来捕捉网络中出现的问题并进行流量分析，也用于各种通信协议的分析和改进，同时在网络教育方面也有着广泛的应用，比如网络安全中的报文分析和协议分析。Wireshark 有时也被称作 Ethereal，这是软件的原名。除此之外，Wireshark 是跨平台的软件，使用当前 Qt 框架的发行版本来执行其端口，并使用 pcap 技术来捕捉网络流量包，可以在 Windows，Linux，macOS 等很多平台上运行。

在实际环境中，情况要复杂得多。首先，计算机无时无刻不在进行网络数据包的交换，因此在抓取 Android 应用启动时流量时，会有许许多多的干扰流量，其中包括了计算机各进程与服务器通信所产生的网络流量，Android 系统中不同应用与网络服务器通信所产生的流量；其次，由于本文要采集的是 Android 应用启动时的流量，如何定义“启动时”的概念也会影响流量抓取。针对前一个问题，本文使用了基于端口，目的地址和协议的过滤方法来排除干扰流量。首先对于计算机进程来讲，这部分流量和 Android 模拟器所产生的流量最大的差别在于协议上，在我们不去手动操作浏览器相关应用的条件下，计算机不会自动产生 HTTP 协议的流量，然而 Android 应用与网络服务器通信所产生的网络流量几乎全部是 HTTP 协议，因此以 HTTP 协议作为过滤语句，我们可以获得 Android 模拟器所产生的这部分流量。其次，为了过滤出目标应用的流量，首先要保证不对其他应用进行操作，且当前时刻下 Android 模拟器只在运行所要采集流量的一个应用；其次，有许多应用时 Android 系统所自动运行的，比如与应用市场的通信等。针对这部分干扰流量，本文在 Android 模拟器启动后，让其运行一段时间，这段时间所产生的所有流量，都可以认为是 Android 系统与各服务器定时的业务通信，可以针对这部分目的 IP 地址进行过滤。经过 HTTP 协议过滤和 IP 地址过滤这双重过滤，我们可以得到比较纯净的目标 Android 应用流量。除此之外，本文将“启动时流量”定义为：Android 应用启动的一刻开始捕获，在没有任何人为操作的情况下，直到该应用不再产生流量为止，这部分流量成为“启动时”，本文通过该种方式来获得一系列 Android 应用的启动时网络流量。

根据上述操作在 Wireshark 中将流量捕获后，保存为 CSV 格式文件，就完成了启动时流

量的捕获。本文对每个 Android 应用重复启动 20 次，并去除其中有明显干扰的样本，剩下的有效样本即为所获取的 Android 应用启动时流量。

3.3 数据预处理

本文所捕获的 Android 应用启动时流量的格式为 CSV 文件，而其中的每一条记录为“No.-Time-Source-Destination-Protocol-Length-Info （即序号 - 时间 - 原 IP 地址 -目的 IP 地址 -网络协议 - 包长度 -详细信息）”格式，其中有许多对本文无用的冗余信息，需要对每个样本进行处理。本文所需要的特征是每个样本的长度，这就需要我们对于原始数据进行标准化的格式处理，包括删除冗余信息，重新编号（这是因为原 No. 即序号选项是该报文在报文流中所占的位置序号，这其中有很多的冗余报文），以及转换为 Weka 所能接受的数据格式。

由于包长度可以从零至上千，如此多维度的属性下，数据在特征空间内变得更加稀疏，这很可能造成分类学习算法的效果不理想。因此我们要进行维度规约。维度规约即降低各个属性所占空间的维度，以创建新属性的方式，通过重新编码或某种变换，将稀疏的旧属性更多的聚合到新属性上来，提高每一个属性的代表性和在分类器中的权重，以此来提高后续分类算法的效率和准确性。

3.3.1 标准数据格式

Weka 所支持的数据集标准格式为 ARFF 格式，其全称为 Attribute-Relation File Format（属性 -关系文件格式）。这种文件类型是面向文本的，由 ASCII 码表示的文本文件，用来描述一组样本及其属性的关系。ARFF 文件中的各个实例之间并无先后的顺序关系，也没有逻辑关系，各个实例之间互相独立。一个 ARFF 文件是一个样本集，其中定义了属性，并给出了一系列独立的实例。属性有以下几种：预定数字型（nominal），只能选择预定的几个数据值作为属性；实数型（numeric）可以取实数域内的任意数值；此外，还有字符串型（String）、日期型（data）和关系型（relation）等等。整个 ARFF 文件可以分为标题部分（header）和数据部分（data）。一个标准的 ARFF 文件样例如图3.2所示。

在上述标准格式中，百分号“%”起始的行为注释行，一般用来说明该文件的用途和意义等。@relation 行描述了这个数据集的名称。@attribute 后所跟的单词为这个样本集中的各个特征，以 @attribute outlook{sunny, overcast, rainy} 为例，这一句定义了一个属性 outlook，并且定义了 outlook 的三个不同取值：sunny，overcast 和 rainy。下面是所有样本实例中的 outlook 属性只有这三个取值。@data 为数据集开始的标志，其后全都是样本实例，格式为逗号分隔符形式，各个属性排列顺序与 @attribute 行中定义的特征顺序一致。

针对本文所要研究的问题来讲，所有属性都应是 numeric 形式，为 Android 应用启动时与服务器通信所产生流量的长度，每个实例所属的类别即为 @attribute class{...} 括号中的内容。

3.3.2 离散化与维度规约

网络流量包的长度是从 1-1500 甚至更多的，这连续的上千个属性会降低分类器的效率和准确率，一个未经离散和维度规约的原始实例的特征可能为 @attribute 0 numeric 一直到 @attribute 1500 numeric 有一千多个特征。所以本文针对连续的属性进行离散化，以 32Byte 为单位对原始的属性集进行维度规约，这样可以把原本稀疏的特征值聚集起来，使每个特征

```
@relation weather.symbolic

@attribute outlook {sunny, overcast, rainy}
@attribute temperature {hot, mild, cool}
@attribute humidity {high, normal}
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}

@data
sunny,hot,high,FALSE,no
sunny,hot,high,TRUE,no
overcast,hot,high,FALSE,yes
rainy,mild,high,FALSE,yes
rainy,cool,normal,FALSE,yes
rainy,cool,normal,TRUE,no
overcast,cool,normal,TRUE,yes
sunny,mild,high,FALSE,no
sunny,cool,normal,FALSE,yes
rainy,mild,normal,FALSE,yes
sunny,mild,normal,TRUE,yes
overcast,mild,high,TRUE,yes
overcast,hot,normal,FALSE,yes
rainy,mild,high,TRUE,no
```

图 3.2 ARFF 标准格式样例

更有代表性，同时提高了分类器的效率和分类的准确率。

3.3.3 脚本程序实现

本文将数据预处理工作分为三步，首先是剔除原始数据中的冗余信息，然后将数据变化为 Weka 所能接受的标准数据格式，最后将特征离散化。基于 VB 语言，首先删除除了 Length 之外的列，之后将 Length 列转置，可以得到一个实例行。对所抓取的所有 Android 应用启动时流量包进行这个操作，并将这些行放到同一个文件中，可以得到针对该 Android 应用的数据集，再将每个数据以 32Byte 为一个单位进行单位化，即可得到适应于 Weka 和流量识别的 CSV 格式样本集，而 CSV 到 ARFF 格式的转换通过 Weka 软件内部探索者文件转换操作可以实现。具体的 VB 代码实现见附录中。

3.4 特征工程

特征工程是使用数据中的已知属性来创造使得机器学习算法得以运行的过程。同时特征工程是机器学习的基础，虽然不是机器学习中一个正式的话题，但其在机器学习的应用中占了非常重要的比例。从数据的一系列属性中选出用于分类的特征是消耗时间且困难的，这需要许多的专业知识。换句话说，解决一个机器学习问题的时候，特征工程就是决定输入 X 是什么的过程。特征指的是对于分类或者预测而言有用的信息，任何对于分类模型产生作用的

属性都可以作为一个特征。特征工程的过程就是从样本集中的一系列属性中选取，转换出对于分类器而言最佳的特征的过程。常用的特征构建方法有 TF-IDF，N-Gram 方法等等。

3.4.1 TF-IDF

TF-IDF 是 Term Frequency-Inverse Document Frequency 的简写，是一种用来反映某一个词在文件中的重要程度的数字统计方法。TF-IDF 由两个统计量所决定，一是该次在文档中出现的频率（Term Frequency），二是该词在语料库中出现的频率（Inverse-Document Frequency），通常用来求某个数据在整个文档中的权重特征。某词的 TF-IDF 值随着这个词语在文档中出现的频率上升而上升，但是会随着这个词在整个语料库中出现的频率下降而下降。

- 词频（Term Frequency）

  词频记为 tf(t,d)，最简单的统计方法为记录原始词目 t 在文件 d 中出现的次数。如果我们用 $f_{t,d}$ 来表示某词目在文档中出现的次数，那最简单的 TF 算法即为 $tf(f,d) = f_{t,d}$，除此之外，其他的可能性还有很多。例如布尔频率，即只将词目出现与否做区别，$tf(f,d) = 0\ or\ 1$；用词目出现的数量除以文档的总长度，此时计算方法为

$$tf(f,d) = \frac{f_{t,d}}{number\ of\ words\ in\ d} \tag{3.1}$$

  常见的 TF 计算方法见表3.2。

表 3.2 不同的 TF 计算方法

权重策略	TF 计算方法
二进制	0,1
原始计数	$f_{t,d}$
词目出现频率	$f_{t,d}/\sum t' \in d f_{t',d}$
对数归一化	$1 + log(f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{max_{t' \in d} f_{t',d}}$
double normalization K	$K + (1-K) \cdot \frac{f_{t,d}}{max_{t' \in d} f_{t',d}}$

- 逆文档频率（Inverse Document Frequency）

  逆文档频率是衡量一个词目所能提供的信息量的多少的，也就是说，用来衡量这个词汇在所有的数据集中是常见还是稀有。其计算方法为用文档的总数除以包含该词目的文档数，并对结果取对数，以获得某个词目的逆文档频率。计算方法为

$$idf(t,D) = log\frac{N}{|d \in D : t \in d|} \tag{3.2}$$

  其中，N 为语料库中所有的文档数量；$|d \in D : t \in d|$ 为所有 t 出现过的文档的数量，如果该词目在语料库中没有出现过，就会导致除以 0 的没有意义的局面，因此这种情况下的计算方法为 $1 + |d \in D : t \in d|$。其他不同计算逆文档频率的方法见图3.3。

表 3.3 不同的 IDF 计算方法

权重策略	IDF 计算方法
一元化	1
逆文档频率	$log\frac{N}{n_t} = -log\frac{n_t}{N}$
平滑化的逆文档频率	$log(\frac{N}{1+n_t})$
最大化的逆文档频率	$log(\frac{max_{t' \in d} n_{t'}}{1+n_t})$
可靠化的逆文档频率	$log\frac{N-n_t}{n_t}$

综合上述 TF 和 IDF 的计算，得到 TF-IDF 系数的计算方法：

$$tfidf(t,d,D) = tf(t,d) \cdot idf(t,D) \tag{3.3}$$

当一个词目的词目频率值很高，而在全体语料库中的逆文档频率很低时，这个词目的 TF-IDF 值很高，这样的计算方法可以去除掉在所有文档中出现频率都很高的常见词目。在本文所研究的 Android 应用识别场景下，语料库就是所有的数据集，指所有应用的启动时流量样本的集合，而文档指的是对于某一应用的数据集。若某个特征在这一应用的样本集下出现频率很高，那这个特征的 TF-IDF 值就会很大。相反的，若某个特征在所有应用的样本空间下出现频率都很高，那该特征就不具有表征某一应用的特殊性，因此其 TF-IDF 值就会低。

3.4.2 区分方向的聚类特征

本文对于 TF-IDF 方案进行一定的改进，TF-IDF 方法虽然可以取得样本集中更有代表性的数据包特征长度，但是会丧失很多的方向和时序信息，因为在 TF-IDF 中并不区分时序和流量的进出；但是由服务器发往客户端的网络流量和由客户端发往服务器的网络流量在应用识别的背景下有着重大区别，前者包括该日服务器所投放的广告，对于不同用户的兴趣推荐，以及根据用户操作的反馈等等，具有很大的不稳定性；而后者更多的是由于 Android 应用编写时的网络时序逻辑所决定，对于应用识别有着更重要的意义。所以将进出的流量进行区分对于应用识别效率的提高有着重要的意义。因此，本文将由服务器发往客户端的流量记作“-”，即长度记作负数，将客户端发往服务器的流量即为“+”，这样就对发往服务器和返回客户端的流量进行了区分。

同时，以 1Byte 为单位长度下，样本集中的数据包长度可以从 -1500 到 1500，三千个特征显然会使特征空间变得稀疏，降低了分类器的运行效率和分类准备率，因为，本文以 32Byte 为单位对样本集进行维度规约，将特征空间降为 -50 到 50，以提高分类器的运行效率和分类准确率。

3.5 分类器的选择

在已经获得的数据样本集的基础上，通过机器学习算法来训练出一个分类函数或者说构建一个分类模型，这就是分类器（Classifier）。分类器可以根据分类函数或者分类模型，把每一个样本实例都映射到某一个确定的类别，也就是对每一个实例都给出自己可靠的预测。构建分类器之前，需要获得足够大的样本，用来训练分类器。同时，也需要测试集，用来测试分类器的准确性。有时候我们也会用部分样本数据集来作为测试数据集，检查分类器的性能。常用的用来训练分类器的分类算法有朴素贝叶斯，决策树和随机森林等。

3.5.1 朴素贝叶斯分类器

朴素贝叶斯算法是一种根据贝叶斯定理来实现的常用分类算法，前提是依靠样本集各个特征之间的强独立性。具体来讲，朴素贝叶斯算法是最简单的构建分类器的方法，是一种可以给未知实例分配标签的模型，朴素贝叶斯算法将实例的特征表示为特征向量，同时为实例分配某个确定的集合中的标签。朴素贝叶斯算法不只是某一种单一的构建分类器的算法，而是一系列基于特征之间互相独立来实现的分类算法的总称。特征之间互相独立意味着，朴素贝叶斯会从不同的特征入手，分别计算基于该特征下，实例应该属于哪一个分类，而不会去考虑这些特征之间的相互关联。

在有监督的机器学习环境下，朴素贝叶斯分类器可以非常有效率的被构建。在许多实际应用中，朴素贝叶斯模型的参数估计可以采用最大似然法；换句话说，人们在不了解贝叶斯定理的情况下，也可以使用朴素贝叶斯分类器进行机器学习操作。

总的来说，朴素贝叶斯模型是一种条件概率模型，给定一个待分类的实例，其 n 个特征向量（需互相独立）用一个特征向量 $\mathbf{x} = (x_1, ..., x_n)$ 来表征，这个特征向量会给予该实例概率 $p(C_k|x_1, ..., x_n)$ 对应于可能存在的 k 个分类结果。这个公式存在的问题是，如果样本集中特征总数 n 足够大，或者每个特征可以有相当多可能的取值，那么基于这种模型下所计算的概率值是不可行的。因此我们重新计算概率模型，使其更加的有效。根据贝叶斯定理，条件概率可以表示为：

$$p(C_k|\mathbf{x}) = \frac{p(C_k)p(\mathbf{x}|C_K)}{p(\mathbf{x})} \tag{3.4}$$

上述公式也可表述为：

$$posterior = \frac{prior \times likelihood}{evidence} \tag{3.5}$$

在实际运算中，分母并不依赖于 $\mathbf{C}$ 并且每一个特征的值都是确定的，因此只有分子需要计算，而分母是一个确定的常数。所以分子等价于联合概率模型 $p(C_k, x_1, ..., x_n)$，利用链式法则和条件概率的公式，这个公式可以被分解为：

$$\begin{aligned} p(C_k, x_1, ..., x_n) &= p(x_1, ..., x_n, C_k) \\ &= p(x_1|x_2, ..., x_n, C_k)p(x_2, ..., x_n, C_k) \\ &= p(x_1|x_2, ..., x_n, C_k)p(x_2|x3, ..., x_n, C_k)p(x_3, ..., x_n, C_k) \\ &= ... \\ &= p(x_1|x_2, ..., x_n, C_k)p(x_2|x3, ..., x_n, C_k)...p(x_{n-1}|x_n, C_k)p(x_n|C_k)p(C_k) \end{aligned} \tag{3.6}$$

现在需要用到“朴素”这一条件了，条件之间互相独立的假设，即在所有类别集合 $\mathbf{C}$ 中，对

于 $i \neq j$ 得每个不同的特征 $\mathbf{F_i}$ 之间都独立，而独立的条件之间并不会有概率上的影响。这意味着：

$$p(x_i|x_{i+1},...,x_n,C_k) = p(x_i|C_k) \tag{3.7}$$

因此，联合概率模型可以表达为：

$$\begin{aligned} p(C_k,x_1,...,x_n) &\propto p(C_k,x_1,...,x_n) \\ &\propto p(C_k)p(x_1|C_k)p(x_2|C_k)p(x_3|C_k)\cdots \\ &\propto p(C_k)\prod_{i=1}^{n} p(x_i|C_k) \end{aligned} \tag{3.8}$$

再加上上述的各特征之间独立的假设，随机变量 $\mathbf{C}$ 的条件分布可以表示为：

$$p(C_k|x_1,...,x_n) = \frac{1}{Z}p(C_k)\prod_{i=1}^{n} p(x_i|C_k) \tag{3.9}$$

在这里 $\mathbf{Z} = p(\mathbf{x})$ 是一个由 $x_1,...,x_n$ 确定的常数。

到这里我们已经讨论出了条件独立下的朴素贝叶斯模型，将此概率模型与判决规则结合起来，就构成了朴素贝叶斯分类器。最常见的规则是选择概率最大的假设，这就是常见的最大后验概率准则或者成为 MAP 决策规则。综上所述，朴素贝叶斯分类器分配标签 $\hat{y} = C_k$ 的法则如下：

$$\hat{y} = \underset{k\in 1,...,K}{\arg\max}\ p(C_k)\prod_{i=1}^{n} p(x_i|C_k) \tag{3.10}$$

3.5.2 决策树分类器

决策树是一种基于二叉树的分类模型，分为决策特征节点，分支和叶子结点三部分构成。在分类过程中，每个决策特征节点处由实例的某个特征属性来决定去往哪一个分支。所分类别的标签存放在叶节点处，决定了该实例被分到哪一类。所以，经过决策树算法训练所生成的分类器，只需要保存一个二叉树的结构，对每个实例的分类，就是从根节点开始，根据每一个特征来一级一级向下搜索，直到达到最低端的叶节点处，得知其分类信息。例如，一个根据天气的外观、湿度和风力来进行是否适宜户外运动的分类器如图3.3所示。

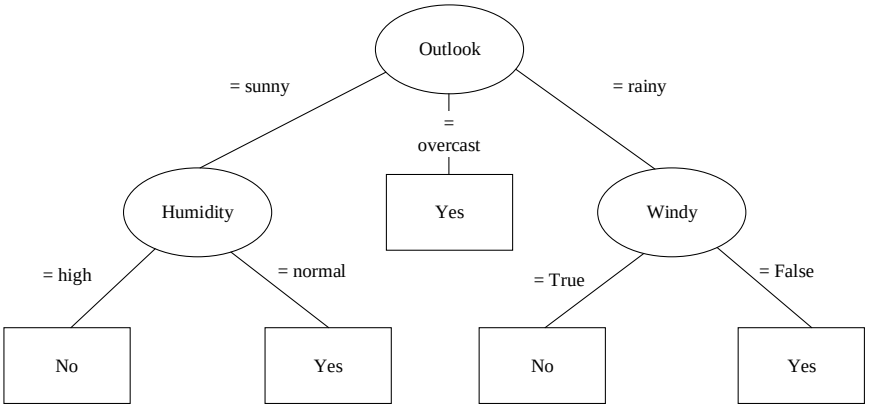


图 3.3 一个决策树实例

最为广泛使用的决策树算法是 C4.5 算法，这种算法对于数据集的特征选择以各属性的信息增益率作为标准。这种算法的生长树算法可以利用 TreeGrowth 的决策树归纳算法，如

图3.4，其中 E 为输入的训练集，F 为输入的属性集。这个算法以递归的方式来选择用以划分数据的最佳属性，并且使该决策树生长，到结束条件成立为止结束。

```
TreeGrowth(E, F)
      if stopping_cond(E, F) = true then
          leaf = creatNode()
          leaf.label = Classify(E)
          return leaf
      else
          root = creatNode()
          root.test_cond = find_best_split(E, F)
          令 V = {v | v是root.tst_cond的一个可能输出}
          for each v ∈ V do
              Ev ={e | root.Test_cond(E) = v and e ∈ E}
              child = TreeGrowth(Ev, F)
              添加child为root的子节点，并将（root—>child）标记为v
          end for
      end if
return root
```

图 3.4 决策树归纳算法框架

算法的工作原理如下，首先依据属性将数据集 S 划分为不同的类别，计算每一个类别 $\mathbf{C_k}$ 的信息增益，信息的增益的计算方法为：

$$Entropy(S) = -\sum_{i=1}^{m} p_i log_2 p_i \tag{3.11}$$

其中，S 为整个数据样本集，$p_i(i = 1, 2, ..., m)$ 为样本集中分别对应于 m 个不同类别的样本在总体样本集中所占的比例。

接下来用属性 A 来划分总体 S，得到该属性不同值所划分出的样本子集，计算每个样本子集的信息熵，即划分熵值。假设属性 A 的取值是离散的，且共有 k 个不同的取值，则样本总体被分割为 $S_1, S_2, ..., S_k$，A 所划分 S 得到的信息熵为：

$$Entropy_A(S) = \sum_{i=1}^{k} Entropy(S)_i \tag{3.12}$$

其中，$|S_i|$ 和 $|S|$ 分别是 S_i 和 S 中所包含的样本实例个数。

若属性 A 的取值是连续的，则将其离散化。先将属性 A 的各种可能取值按递增的顺序排列，计算每两个相邻的属性值的中点，称其为属性 A 的分裂点。对每个可能存在的分裂点，计算信息熵：

$$Entropy_A(S) = \frac{|S_L|}{|S|} Entropy(S_L) + \frac{|S_R|}{|S|} Entropy(S_R) \tag{3.13}$$

其中 S_L 和 S_R 分别指针对该分裂点左右划分出的样本子集，选择计算得出的信息熵最小的一个分裂点作为最佳分裂点，该分裂点所求得的划分熵值就是连续取值属性 A 所划分样本总体 S 得到的划分信息熵值。

接下来计算信息增益。每个属性所计算出对应于该属性的划分信息熵，即用数据样本集 S 总的信息熵减去按照某一属性 A 所计算出的划分信息熵，得到信息增益 $Gain(S, A)$，即：

$$Gain(S, A) = Entropy(S) - Entropy_A(S) \tag{3.14}$$

得到所有属性的信息增益之后，再计算各个属性的分裂信息，之后计算对应的信息增益率。其中，分裂信息用作信息增益的辅助调节作用：

$$SplitE(A) = -\sum_{i=1}^{k} \frac{|S_i|}{|S|} log_2 \frac{|S_i|}{|S|} \tag{3.15}$$

而信息增益率是将某一属性的分裂信息 $SplitE(A)$ 作为分母，信息增益 $Gain(A)$ 作为分子：

$$GainRatio(A) = \frac{Gain(A)}{SplitE(A)} \tag{3.16}$$

这种调节方法的目的是，当属性 A 的某一取值的实例数目越大时，分裂信息就越大，可以一定程度上抵消属性 A 的这种取值对信息增益 $Gain(A)$ 所带来的影响。

之后，比较各个属性的信息增益率 $GainRatio$ 的值，取值最大的一个属性作为这一次计算所得出的分裂节点，其不同的取值作为不同的分支，完成本次树的生成。之后，对本次分裂节点之下的各个分支，递归运行上述算法，求子树，最终得到 C4.5 方法决策树。J48 决策树就是 Weka 中利用 C4.5 算法生成的决策树实例。

3.5.3 随机森林分类器

顾名思义，随机森林内包括不计其数的树，这种树是决策树。也就是说，随机森林是由不计其数的决策树分类器组成的，这种分类器的输出由所有决策树的输出中最多的一个类别决定，即决策树分类输出中的众数，流程图如图3.5。首先通过 Boot-Strap 方法从样本集中随机提取不同的样本构成样本子集，且各子集中可以存在重复元素。用这些样本子集分别训练决策树，以构成随机森林。测试数据的分类结果由所有决策树独立分类产生的结果中出现次数最多的那一个决定。

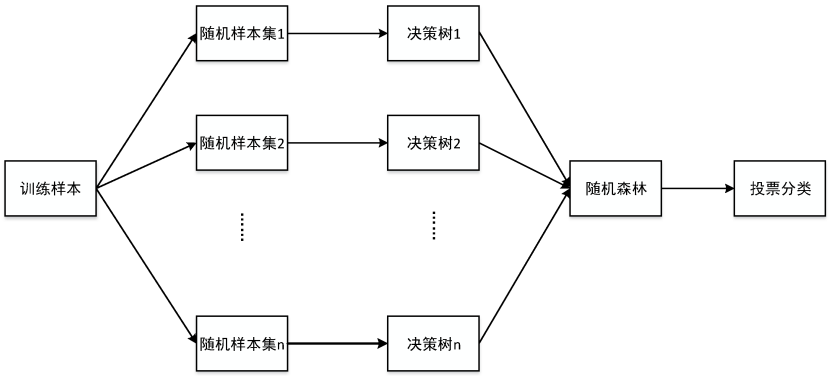


图 3.5 随机森林算法流程图

随机森林的伪代码实现如表3.4。其中 Boostrap 方法指的是从样本数据集中随机选取一子集对决策树进行训练，这也正是“随机”的含义。在每个需要分裂的节点，从全部属性中取出其中少量属性，在这少量属性中按照上述 Gini 算法或者其他算法选取最佳的分裂策略。之后递归运行上述算法，直到一颗决策树完全生成。按照这种步骤生成大量的决策树，这就形成了一篇决策森林。

与决策树相比，随机森林算法在以下方面有着更加优良的表现：

表 3.4 实现随机森林伪代码

伪代码 Random Forest
输入：1. 训练集 $S=(x_i,y_i),i=1,2,...,n,X,Y\in R^d\times R$ 2. 待测试样本 $x_i\in R^d$ $For\ i=1,2,...,N_{tree}$ (1) 对初始总体样本集 S 进行 Boostrap 随机取样，抽取出各个样本子集 S_i (2) 对 S_i 生成一棵不剪枝的树 h_i a. 从 d 个特征中随机选取 M_{try} 个特征 b. 在每个节点上从 M_try 个特征中依据 Gini 指标选取最优特征 c. 分裂直到树生长到最大 End
输出：1. 树的集合 $h_i,i=1,2,...,N_{tree}$ 2. 对待测样本 x_i 决策树 h_i 输出 $h_i(x_i)$
分类答案：$f(x_t)=majority\ \ vote\ h_i(x_i)_{i=1}^{N_{tree}}$

1) 在大部分数据集之上的表现优异；
2) 能够处理大量的维度很高的数据，并不用从中选择并舍弃部分属性或特征，因为算法会自动随机取样；
3) 训练器模型生成的速度快，且可以给出各个属性对于解决问题的权重和属性间的相互影响；
4) 较之其他算法，随机森林分类算法更加适应并行化运算。

在某些方面，随机森林算法也有着自己的缺点：

1) 当面对噪声较大的数据样本时，随机森林在解决针对这些样本的回归或分类问题容易出现过拟合现象；
2) 当样本数据训练集中的属性特征有着不同的权重，或有着复杂的优先级时，则随机森林在随机取样的过程中会对结果产生较大的偏差影响，面对这种样本训练集，随机森林所产生的分类或回归结果是不可靠的。

3.5.4 支持向量机分类器

在有监督的机器学习分类算法中，支持向量机（Support Vector Machine，SVM）是一种适用广泛的算法。支持向量机算法在数学和统计学上有着坚实的基础。SVM 算法可以将经验

误差最小化，并且同时在特征空间内使得几何边缘最大化，这也是为什么支持向量机算法也被称为最大边缘分类器。从所适应的数据集来讲，支持向量机算法对于高维度的样本数据集有着优良的表现，可以有效的规避维数灾难。其分类算法的最大特点是，从数据集中按照某种规则选择某个样本子集，并将其作为决策分类边界，这个样本子集就是所谓的支持向量。

最开始的支持向量机算法只支持二元分类，即按照支持向量将数据样本空间一分为二，之后确定每一个样本属于哪一个类别。所以在支持向量机分类器中，对于一个 N 维的数据集，需要一个 N-1 维的支持向量来将数据空间划分为两份，进行分类。而这一个支持向量的选择的原则则是能够最大限度的区别出两个不同的分类，有时也称支持向量为一个超平面。随着机器学习理论的发展，如今的支持向量机不仅支持二元分类，也扩展到了多元分类的领域，下面以线性支持向量机为例介绍 SVM 算法。

若数据样本集 D 中包含 N 个不同的训练样本，则有：

$$D = \{(\mathbf{x_i}, y_i)|\mathbf{x_i} \in R^p, y_i \in -1, 1\} \qquad i = 1, 2, ..., N \tag{3.17}$$

其中，y_i 表示对于点 $\mathbf{x_i}$ 的二元分类的结果，只有 -1 和 1 两种取值可能。$\mathbf{x_i}$ 为训练集中维数为 p 的向量。分类目标是寻找一个维数为 p-1 的超平面，来将 $y_i = 1$ 和 $y_i = -1$ 的样本点进行分割。而超平面可以用一个方程来表示：

$$\mathbf{w} \cdot \mathbf{x} - b = 0 \tag{3.18}$$

其中，“$\cdot$”表示向量之间的数量积运算，$\mathbf{w}$ 表示垂直于超平面的向量，即法向量；参数 $\frac{b}{||\mathbf{w}||}$ 是沿法向量方向，从超平面的原点起始的位移大小。

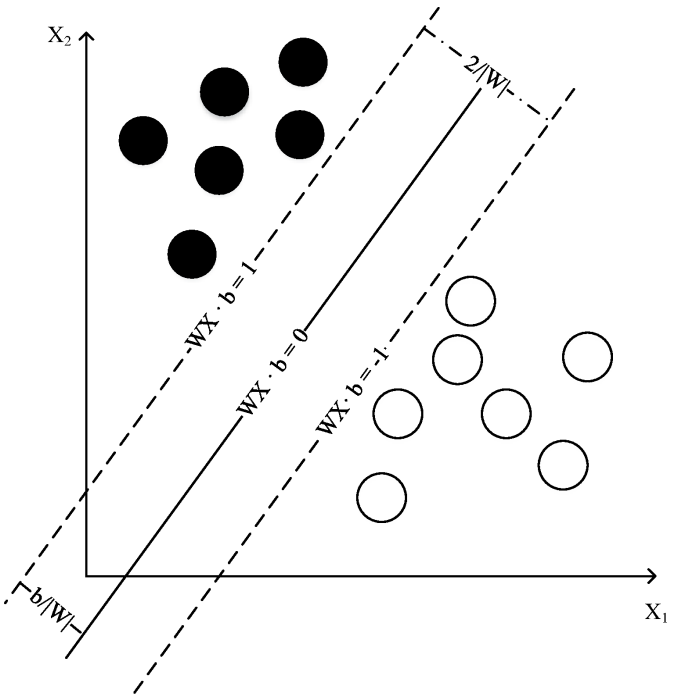


图 3.6 SVM 最大边缘超平面

如果样本数据集可以用线性超平面进行分割，可以选择两个超平面，并且保证两个超平面之剑没有样本点存在，然后保证两个超平面之间的距离最大。这两个超平面方程可以表示

为：

$$\mathbf{w} \cdot \mathbf{x} - b = 1 \tag{3.19}$$

$$\mathbf{w} \cdot \mathbf{x} - b = -1 \tag{3.20}$$

由空间几何的知识得知，两个平面之间的距离为 $\frac{2}{||\mathbf{w}||}$，该距离在 $||w||$ 最小时取到最大值。为了保证没有样本点落在两个平面之间和边缘之上，要添加约束条件：

$$\mathbf{w} \cdot \mathbf{x_i} - b \geq 1 \tag{3.21}$$

$$\mathbf{w} \cdot \mathbf{x_i} - b \leq -1 \tag{3.22}$$

此二式也可以概括为

$$\mathbf{y_i}(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1 \tag{3.23}$$

综上所述，对于线性条件下的支持向量机的最大边缘化等价于最小化如下的目标函数：

$$f(\mathbf{w}) = \frac{||\mathbf{w}||^2}{\mathbf{2}} \tag{3.24}$$

在 Weka 环境中，集成了许多基于支持向量机的算法。在本文中取序列最小优化算法（Sequential Minimal Optimization, SMO）作为支持向量机算法的代表算法进行分类。

3.6 本章小结

本章的第一部分介绍了如何对 Android 应用启动时的网络流量进行采集以及 Android 应用的选择。本文采用了 Android 模拟器的方法，选取了 Android 市场上目前下载排行榜前 50 名的应用作为样本，每个应用重复启动 15 次，采集这些 Android 应用启动时与服务器所产生的流量。在流量采集时会有噪声流量，这部分流量通过 Wwireshark 的过滤功能去除。之后保存下来的文件格式为 CSV 格式，且有许多冗余信息，本文编写了基于 VB 的脚本来实现自动化数据预处理，转换为 Weka 所能接受的标准格式。

本章的第二部分和第三部分介绍了如何对数据进行处理和特征选择。首先以数据包的长度作为特征，存在特征空间维度太高的问题。因此以 32Byte 为单位对长度特征进行维度规约，同时将数据格式转化为 Weka 所能接受的标准 ARFF 格式。对于特征的选择，常用的方法为 TF-IDF 方法，但是对于本文所研究的问题，Android 应用出入的方向对于应用识别有着重大的影响，所以对于进出客户端的流量加以正负号的区别，这样就可以区分出这两部分流量，本文称之为区分方向的聚类特征。

本章的第四部分介绍了分类中常用的几种分类器算法。包括常用的基于贝叶斯定理的朴素贝叶斯分类算法，基于空间划分的支持向量机算法，决策树算法和基于众多决策树的随机森林算法，并比较各种算法的特点。

4　测试与结果分析

将所有原始数据经过预处理和特征工程之后，得到 50 个 Android 应用启动时所产生流量长度的数据集。其中长度以 32Byte 为单位进行维度规约，且按照流量出入的方向对长度特征进行正负变化加以区分：由客户端发往服务器的流量长度标记为正，而从服务器返回客户端的流量长度标记为负。然后把用该数据集训练不同的分类器，如朴素贝叶斯分类器，决策树分类器，随机森林分类器和支持向量机分类器，得到训练之后的分类模型。

在测试所生成分类器的方法上本文主要使用了 K 折交叉验证的方法。所谓 K 折交叉验证，是指将训练数据集中的数据等分为十份，然后依次取其中九份做份训练集来训练出分类模型，剩余一份作为测试集对分类模型进行测试，得到分类器表现的各个指标。

4.1 测试环境

本次系统测试所处的软硬件环境如表4.1所示。

表 4.1 测试软硬件环境

项目	参数
CPU	Intel(R) Core(TM) i5-3230M CPU @ 2.60GHz 2.60GHz
RAM	4.00GB（3.70GB 可用）
硬盘大小	500GB
系统版本	Windows 7 64 位企业版
测试软件版本	Weka 3.8

4.2 分类器 k 折交叉验证下输出分析

表4.2为以主要指标来衡量的经过数据集训练之后的各分类器表现，其中各个指标参数的意义为：

- Correctly Classified Instances：训练集中分类正确的数据个数，指训练集中的数据经过分类器模型分类后，所得到的分类输出与原数据集中对于该实例的分类标签一致的样本个数。
- Incorrectly Classified Instances：训练集中分类错误的数据个数，指数据集中的数据经过分类器模型分类后，所得到的分类输出与原数据集中对于该实例的分类标签不同的样本个

表 4.2 不同算法在 k 折交叉验证下表现

指标	朴素贝叶斯	决策树	随机森林	支持向量机
Correctly Classified Instances	98.92%	92.03%	99.73%	90.95%
Incorrectly Classified Instances	1.08%	7.97%	0.27%	9.05%
Kappa statistic	0.99	0.92	0.99	0.90
Mean absolute error	0.01	0.01	0.01	0.03
Root mean squared error	0.02	0.06	0.03	0.14
Relative absolute error	1.13%	9.03%	13.91%	97.96%
Root relative squared error	14.66%	39.93%	21.48%	98.29%

数。

- Kappa statistic：Kappa 统计指数，取值范围在 $[0,1]$ 内的一个实数。Kappa 统计指数用来衡量经过所训练的分类器分类之后的分类结果与将测试数据集经过随机分类之后的分类结果之间的区别度。Kappa 指数越接近于 1，表示该分类器的分类结果与随机分类完全不同，即分类器的表现良好；Kappa 指数越接近于 0，表示该分类器的分类结果与随机分类完全相同，即分类器完全无效；而当 Kappa 指数接近于 -1 时，表示该分类器的分类效果比随机分类还要糟糕。通常来说，Kappa 指数与模型的 AUC 指数的变化趋势一致，所以这两个指数有一定的相关性。
- Mean absolute error：平均绝对误差，指所有所有数据样本与平均值的差别的绝对值的平均，取值范围为 $[0,1]$。
- Root mean squared error：相对均方根误差，指观测所得值与真实值偏差的平方除以观测次数比值的平方根，取值范围为 $[0,1]$。
- Relative absolute error：相对绝对误差，以百分数的形式来描述。
- Root relative squared error：相对均方根误差，以百分数的形式来描述。

比较表4.2中各分类器的表现，可以发现随机森林算法可以达到最好的分类准确率，高达 99.73%，其次是朴素贝叶斯算法，分类准确率达到 98.92%，决策树算法的分类准确率为 92.03%，而在本文所研究的 Android 应用分类的问题上表现最差的是支持向量机算法，分类准确率仅有 90.95%。究其原因，因为本文的数据样本集属于高维度特征样本，对于这种样本随机森林算法不会舍弃其中部分特征，由其随机取样的算法可以使得所有维度都得到充分的使用，以此获得最高的分类准确率。而作为随机森林算法的基础，决策树算法显得相对单薄，在 Android 应用分类问题上，每个应用的启动时流量往往不能够单单通过某一单一特征来进行分裂区分，因此决策树算法得到的准确率相对较低。朴素贝叶斯算法作为基于贝叶斯定理的稳定算法，也如期得到了一个不错的分类成功率。而支持向量机算法之所以得到最低

的分类准确率，原因类似于随机森林，是因为本次数据集的维度很高，很难通过一个超平面准确的将样本做一个二元划分。

对于不同分类算法，其模型所需要的训练时间也有所不同，如表4.3所示。朴素贝叶斯模型所需训练时间最短，这是因为朴素贝叶斯算法基于贝叶斯定理和条件概率，本身就是一个简单的算法，所需训练时间很短。随机森林模型比决策树模型所需时间明长许多，这是因为随机森林算法中要建立不计其数的决策树，并行化运算下总时间会有所减少，但也会比对应的决策树算法消耗更长的训练时间。支持向量机算法消耗了最多的时间，这是因为支持向量机算法旨在将样本数据集在特征空间内进行划分，本文所研究的问题又是多元问题的高维特征分类问题，数据集过高的维度和本文过多的类别标签导致了最长的训练时间，且得到了最低的分类效率。

表 4.3 训练不同分类器所用时间

	朴素贝叶斯	决策树	随机森林	支持支持向量机
训练时间 (s)	0.12	0.54	1.23	6.1

4.3 属性选择

经过的前文所述的维度规约和特征工程，样本数据集中的长度特征从 $[-50, 50]$，过多的特征导致支持向量机模型的训练时间过长，且分类准确率低下。进行属性的权重排列和选择对于提高分类器的训练效率和分类准确率都有着重要的意义。本文使用了各属性的信息熵增益的方法来评估各属性对于分类的重要性，结果如表4.4所示；同时重要性最低的十个特征如表4.5所示。

表 4.4 重要程度最高的 10 个特征

重要性排名	1	2	3	4	5	6	7	8	9	10
特征	8	10	7	17	-7	-2	18	-8	-11	19

从表4.4中可以发现，重要性靠前的特征全都是正数，也就是由客户端发往服务器的流量。这也说明了由服务器发往客户端的流量对于 Android 应用识别有着更强的重要性。这是因为这部分流量完全基于不同 Android 应用的内在网络逻辑，具有更强的稳定性，在识别中也应占有更高的权重。值得一提的是，重要性程度排名最低的 10 个特征的权重为 0，即去掉

表 4.5 重要程度最低的 10 个特征

重要性倒序排名	1	2	3	4	5	6	7	8	9	10
特征	-49	-48	50	-1	-44	-3	2	1	-9	-50

这些属性对于分类器的分类效率会有提升，同时对分类准确率没有影响。本文将这些属性去除后，重新测试每种模型的训练时间，结果如表4.6，可见去除无用特征对于分类器的效率有着明显的提升。

表 4.6 去除无用特征之后训练不同分类器所用时间

	朴素贝叶斯	决策树	随机森林	支持支持向量机
训练时间 (s)	0.10	0.51	1.02	5.5

4.4 基于不同日期流量的应用识别

当前 Android 应用更新速度快，几乎每个应用每周都要进行更新。因此，研究隔日 Android 应用启动时流量作为样本集时，在前日流量所训练模的分类模型下的表现有着重要意义。除此之外，大部分 Android 应用在启动时会投放广告，而所投放的广告经常会发生变化，这会对从服务器返回客户端的流量产生较大影响。同时，不同日期下用户的浏览习惯会发生较大变化，而服务器对于客户的兴趣推荐这部分流量也会在不同日期产生影响。为此，本文针对几个常用的应用，每各一周采集一次流量，持续两个月，得到隔日流量样本集，并将其作为测试样本，测试训练集的鲁棒性，结果见表4.7。

表 4.7 不同日期流量测试下模型的表现

指标	朴素贝叶斯	决策树	随机森林	支持向量机
Correctly Classified Instances	100%	100%	100%	100%
Incorrectly Classified Instances	0%	0%	0%	0%
Kappa statistic	1	1	1	1
Mean absolute error	0	0	0.02	0.22
Root mean squared error	0	0	0.03	0.27
Relative absolute error	0%	0%	4.9%	50%
Root relative squared error	0%	0%	7.07%	57.74%

由上图可知，四种分类器对于不同日期的流量的识别都有着 100% 的识别率。究其原因，部分因为本文所采集隔日流量的持续时间太短，在这个时间段内目标 Android 应用在网络逻辑上并没有较大的更新和变动，且所投放的广告和推荐虽在内容上有所不同，但数据包长度并无太大差异。即使这部分流量有着较大变化，那在分类器训练过程中也不会将这不问不稳定的流量作为权重较高的特征，因为对分类器的表现影响可以忽略。

4.5 基于其他不同条件下的应用识别

除了不同日期下 Android 应用在网络流量方面会表现出差异以外，本文认为不同的移动设备类型，不同的网络服务供应商也会对 Android 应用启动时流量产生一定的影响，因此本文采集了不同设备（华为 Mate 9/三星 Galaxy S6/小米 MI 5 等）下 Android 应用的启动时流量，和在不同的网络服务供应商下的 Android 应用启动时流量，分别见表4.8和表4.9。

表 4.8 不同设备流量测试下各分类模型的识别率

设备	朴素贝叶斯	决策树	随机森林	支持向量机
华为 Mate 9	100%	100%	100%	100%
三星 Galaxy S6	100%	100%	100%	100%
小米 MI 5	100%	100%	100%	100%

表 4.9 不同服务商流量测试下各分类模型的识别率

服务商	朴素贝叶斯	决策树	随机森林	支持向量机
联通	100%	100%	100%	100%
电信	100%	100%	100%	100%
移动	100%	100%	100%	100%

综上所述，所有分类模型在不同的设备和网络服务供应商环境下对于测试集的分类成功率为 100%。这说明 Android 应用的启动时流量主要取决于应用本身的网络逻辑，只要是同一个软件，不论在什么设备和网络服务供应商下运行，其网络逻辑和产生的流量都是类似的。当然，如若某 Android 应用对于不同设备或网络服务供应商提供了完全不同的应用版本，那将会对分类器产生较大的影响。

4.6 本章小结

本章的第一部分分析了同一训练集训练下，朴素贝叶斯分类器、决策树分类器、随机森林分类器和支持向量机分类器在 k 折交叉验证下的表现。结果表示随机森林分类器有着最佳的分类准确率，但是需要较长的训练时间。朴素贝叶斯分类器作为经典的分类算法表现良好，且所需训练时间短。决策树分类器由于本身模型较为简单，在分类效果上表现一般。而支持向量机分类器在针对本文所研究的 Android 应用分类问题上表现不佳，且模型所需的训练时间最长，这是因为本文的训练集特征维度太高，且类别标签太多，对支持向量机算法而言非常困难。

本章的第二部分和第三部分分析了系统的鲁棒性。主要验证了在不同日期流量、不同供

应商流量和不同设备流量下分类器的表现，结果显示各分类器表现都很好，具有较强的鲁棒性。究其原因，本文的研究时间跨度太短，可能使一些潜在的问题没有暴露，这也是后续研究所要着眼的地方。

5 总结与展望

5.1 研究总结

本文所要研究的基于启动时流量的 Android 应用识别分为四个步骤：首选选择样本 Android 应用，并在 Windows 平台下的 Android 模拟器上安装。之后使用 Wireshark 捕捉 Android 应用启动时的流量，这个过程中存在冗余流量过滤的问题，解决该问题使用的方法是 Wireshark 中基于流量包协议和源/目的 IP 地址过滤的方法，只保留目标应用的启动时网络流量；其次对于所保存的 CSV 格式的 Android 应用启动时流量，其中存在许多冗余信息，本文所需要的信息只有数据包的长度，因此需要对文件格式进行处理，只保留分类器所需要的特征；然后对于 Weka 所能接受的标准数据格式，进行特征工程，选择可以输入分类器的特征，这一步本文基于 TF-IDF 方法进行改进，使用了区分方向的聚类特征，即对于客户端发送到服务器的流量和服务器返回客户端的流量进行正负号的区分，且按照 32Byte 为单位对长度特征进行维度规约，这两步之后完成了数据的预处理；之后选择不同的分类器，本文分别使用了朴素贝叶斯分类器，决策树分类器，随机森林分类器和支持向量机分类器进行模型训练，并对训练之后的模型进行 k 折交叉验证下的评估；评估结果显示随机森林算法有着最佳的分类准确率，而支持向量机算法对于本文所研究的问题表现不佳，究其原因，是因为本文样本集的特征维度太高，且特征标签太多，不适用于支持向量机算法；除此之外，为了检测系统的鲁棒性，本文利用不同日期、不同设备和不同供应商下所产生的测试流量样本对分类模型进行检验，结果显示分类器鲁棒性良好。

5.2 未来展望

首先，本文针对 50 个常用的 Android 应用进行流量采集和识别，样本大小较小，要实现所有通用 Android 应用的识别目标，还需要扩大样本集的大小。Android 应用的数据采集和预处理可以归并为一个机械的过程，本文采用了人工的数据采集和基于 VB 的数据预处理方法，在未来的工作中可以使用自动化的方法同时实现这两步工作，这对于样本集的扩大有着至关重要的意义。

其次，本文所界定的“启动时”标准是人为的观测应用在无人为操作的情况下不再产生网络流量，这种方法存在一定的误差，很可能会漏掉许多对于分类有着重要意义的样本流量，且应用程序很可能在加入网络之前已经启动，因此如何获得真实的流量跟踪，也是后续研究需要重视的一个问题。

此外，从启动时间流量识别应用程序的能力也有助于对后续（非启动时间）流量进行分类，实际上，在启动时数据包中识别出应用程序启动后，后续数据包可能属于同一个应用程序。这个启发式方法可以用于从在户外环境收集的流量跟踪中提取应用程序的标记的网络跟踪，这样的数据集对于未来的流量分析研究将是有价值的。

文献[13] 和文献[14] 分析了 HTTP 头以提取标识应用程序的字符串模式。此类字符串通常由第三方广告或分析服务用于为应用程序分配唯一标识符，然而其中也说明了，当应用程序加密是使用 TLS 协议的 HTTP 流量时，不能使用这种深度包检测方法。但这仍然对于应用识别有着重要意义。在文献[10] 中提到了基于包大小的应用程序识别方法集成到软件定义网络（SDN）平台，并使用它来研究 70 个应用程序的可识别性。这将是应用识别与 SDN 环境的高效结合，对于未来的研究有着重要意义。文献[15] 表明，安装了 14 个特定应用程序的智能手机的身份可以从应用程序生成的后台 3G 网络流量中以 90％的概率确定。在文献[?] 中，研究了在 7 个应用程序中执行某些用户操作时生成的网络流量。示出了当产生流量的应用（例如 Gmail）已知时，可以以超过 95％的准确度来识别用户动作，例如发送电子邮件或打开聊天。这项工作可以被认为是对我们的文件的补充。在从其启动时间网络流量识别应用之后，可以在后续网络流量上使用经训练以区分在应用内执行的用户动作的方法。

参考文献

[1] Callado A, Kamienski C, Szabó G, et al. A survey on internet traffic identification[J]. IEEE communications surveys & tutorials, 2009, 11(3).

[2] Chen S, Wang R, Wang X F, et al. Side-channel leaks in web applications: A reality today, a challenge tomorrow[C]//Security and Privacy (SP), 2010 IEEE Symposium on. IEEE, 2010: 191-206.

[3] Zhou W, Zhou Y, Jiang X, et al. Detecting repackaged smartphone applications in third-party android marketplaces[C]//Proceedings of the second ACM conference on Data and Application Security and Privacy. ACM, 2012: 317-326.

[4] Miller B, Huang L, Joseph A D, et al. I know why you went to the clinic: Risks and realization of https traffic analysis[C]//International Symposium on Privacy Enhancing Technologies Symposium. Springer International Publishing, 2014: 143-163.

[5] Sicker D C, Ohm P, Grunwald D. Legal issues surrounding monitoring during network research[C]//Proceedings of the 7th ACM SIGCOMM conference on Internet measurement. ACM, 2007: 141-148.

[6] White A M, Krishnan S, Bailey M, et al. Clear and Present Data: Opaque Traffic and its Security Implications for the Future[C]//NDSS. 2013.

[7] Herrmann D, Wendolsky R, Federrath H. Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier[C]//Proceedings of the 2009 ACM workshop on Cloud computing security. ACM, 2009: 31-42.

[8] Liberatore M, Levine B N. Inferring the source of encrypted HTTP connections[C]//Proceedings of the 13th ACM conference on Computer and communications security. ACM, 2006: 255-263.

[9] Le A, Varmarken J, Langhoff S, et al. AntMonitor: A system for monitoring from mobile devices[C]//Proceedings of the 2015 ACM SIGCOMM Workshop on Crowdsourcing and Crowdsharing of Big (Internet) Data. ACM, 2015: 15-20.

[10] Qazi Z A, Lee J, Jin T, et al. Application-awareness in SDN[J]. ACM SIGCOMM computer communication review, 2013, 43(4): 487-488.

[11] Cisco C V N I. Global Mobile Data Traffic Forecast Update, 2015–2020 White Paper, 2016[J].

[12] Dai S, Tongaonkar A, Wang X, et al. Networkprofiler: Towards automatic fingerprinting of android apps[C]//INFOCOM, 2013 Proceedings IEEE. IEEE, 2013: 809-817.

[13] Xu Q, Liao Y, Miskovic S, et al. Automatic generation of mobile app signatures from traffic observations[C]//Computer Communications (INFOCOM), 2015 IEEE Conference on. IEEE, 2015: 1481-1489.

[14] Yao H, Ranjan G, Tongaonkar A, et al. Samples: Self adaptive mining of persistent lexical snippets for classifying mobile application traffic[C]//Proceedings of the 21st Annual International Conference on Mobile Computing and Networking. ACM, 2015: 439-451.

[15] Stöber T, Frank M, Schmitt J, et al. Who do you sync you are?: smartphone fingerprinting via application behaviour[C]//Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks. ACM, 2013: 7-12.

[16] Conti M, Mancini L V, Spolaor R, et al. Analyzing android encrypted network traffic to identify user actions[J]. IEEE Transactions on Information Forensics and Security, 2016, 11(1): 114-125.

[17] Alan H F, Kaur J. Can Android Applications Be Identified Using Only TCP/IP Headers of Their Launch Time Traffic?[C]//Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks. ACM, 2016: 61-66.

[18] 王靖华, 何迪. 基于数据包字节长度的线性自回归 (Autoregression) 和支持向量分类机 (SVM) 的网络流量预测建模与分析 [J]. 微型电脑应用, 2005, 21(11): 1-3.

[19] 黄志根, 陈健, 王珊. 一种基于包长和时间间隔的网络流量分类方法 [J]. 电子测量技术, 2011, 34(11): 109-112.

致　谢

岁月如梭，四年的本科生活即将画下句号。一路上有老师和同学们的陪伴和支持，收获良多。在此，我要感谢所有给过我指点帮助的老师和同学们。

首先，我要感谢秦中元老师。秦中元老师严谨的治学态度和谦和的为人对我有着很大的影响，在我遇到问题时老师总能给我一条最正确的道路。同时，秦中元老师锻炼了我独立解决思考和解决问题的能力，从最初的选题开始，到研究方向的把握，难点的解决，都离不开秦老师的悉心指导。我相信这对于我以后的科研生活和工作都会产生很积极的影响。

另外，我要感谢刘楚君，杨佳伟同学，潘桂鑫同学，黄进瑞同学，祖剑君学长和陆凯学长对我的支持和鼓励。在毕业设计期间，大家的交流讨论对我的学习产生了极大的促进作用，每周的组会上都会从大家的发言中汲取到许多有用的知识。

最后，我要感谢我的父母，在四年的大学生活里给予我无微不至的支持和关怀，给我足够的空间发展自己的兴趣和爱好，为我提供了优越的生活和学习环境，使我可以全身心的投入到学校生活中。在你们的支持下，我才得以安心完成学业。

在即将毕业之际，我真诚的感谢所有指导、鼓励和帮助过我的所有人，祝你们未来的道路一帆风顺！