**大连理工大学本科毕业设计（论文）**

**基于哈希编码的网络流量分类方法的研究**

**Research on Network Traffic Classification Based on Hash**

学 院（系）：

专 业：

学 生 姓 名：

学 号：

指 导 教 师：

评 阅 教 师：

完 成 日 期：

大连理工大学

Dalian University of Technology

# 摘 要

近十年来，网络规模和网络用户的爆炸式增长为互联网应用带来了光明的前景同时也为网络管理与维护机构以及相关从业者从业者带来了复杂的管理与运维工作。对网络应用以及网络流量的识别与分类研究具有较大的实际应用价值，成为当前网络研究领域的热点。

通过将高维数据量化为哈希值来降低数据的存储开销和检索开销，基于哈希编码的大数据近邻检索技术主要解决大数据环境中数据分类和查找问题。鉴于该特点，探索基于哈希编码的大数据近邻检索能否与网络流量分类研究相结合以及如何实现较优的分类效果是本文研究的重点。

通过阅读相关文献、设计具体算法以及验证实际性能，本文实现了利用基于哈希编码的大数据近邻检索技术对网络流量进行分类的预期目标，本文主要完成了以下具体工作：

（一）研究网络流量分类方法以及大数据近邻检索技术，掌握基于乘积量化的大数据近邻检索方法。

（二）实现一种基于哈希编码的网络流量分类方法，构建对应的网络流量分类模型，选择数据集对该分类模型进行验证，验证结果表明基于哈希编码的网络流量分类方法为可行。

**关键词：网络流量分类；哈希编码；大数据近邻检索；乘积量化**

**Research on Network Traffic Classification based on Hash**

# Abstract

In the past ten years, The explosive growth of network size and number of network user results in promising prospects for many network applications. However, the management and operation of network become more complicated for network management and maintenance organizations. The research on identification and classification of network traffic generated by network applications play an important role in related research and Traffic Classification(TC) is one of the most trending fields in current network research.

The technique of large-scale data Nearest Neighbors(NN) Retrieval based on hashing usually is applied to solve the classification and retrieval of large-scale data by quantizing high-dimensional data into hash values. Considering the factors which mentioned above, the focus of this paper is to explore whether hash-based Near Neighbors(NN) Retrieval of large-scale data can be applied in the classification of network classification and how to implement high performance.

By studying relevant references, designing classifying algorithms and verifying its performance, this paper has accomplished the expected target of classifying network traffic by using the technology of the large-scale data Nearest Neighbor Retrieval based on hashing. The main work of this paper is below:

(1) Research the methods of network traffic classification and the technique of large-scale data Nearest Neighbor Retrieval, be familiar with the implement of Nearest Neighbor Retrieval by Product Quantization(PQ).

(2) Implement one classification method of network traffic base on hashing, build related classification model and then verify its performance with selected datasets which proves that the method of network traffic classification based on hashing is practical.

**Key Words： Network Traffic Classification; Hash coding;** **Large-scale data Nearest Neighbors retrieval; Product Quantization**

目 录

[摘 要 2](#_Toc516599248)

[Abstract 3](#_Toc516599249)

[引 言 1](#_Toc516599250)

[1 相关技术概述 4](#_Toc516599251)

[1.1 网络流量分类 4](#_Toc516599252)

[1.1.1 网络流 4](#_Toc516599253)

[1.1.2 网络流特征 4](#_Toc516599254)

[1.1.3 基于端口匹配 4](#_Toc516599255)

[1.1.4 基于深度包检测 5](#_Toc516599256)

[1.1.5 基于协议解析 5](#_Toc516599257)

[1.1.6 基于统计学习 6](#_Toc516599258)

[1.2 基于哈希的大数据近邻检索 6](#_Toc516599259)

[1.2.1 近邻检索 6](#_Toc516599260)

[1.2.2 基于随机投影的哈希算法 9](#_Toc516599261)

[1.2.3 基于学习的哈希算法 10](#_Toc516599262)

[1.2.4 基于乘积量化的哈希算法 10](#_Toc516599263)

[2 基于哈希编码的网络流量分类方法 15](#_Toc516599264)

[2.1 引子 15](#_Toc516599265)

[2.2 相关算法 15](#_Toc516599266)

[2.2.1 K均值算法 15](#_Toc516599267)

[2.2.2 KNN算法 16](#_Toc516599268)

[2.3 算法描述 16](#_Toc516599269)

[2.3.1 线下训练阶段 17](#_Toc516599270)

[2.3.2 实时分类过程 18](#_Toc516599271)

[2.3.3 算法复杂度分析 19](#_Toc516599272)

[2.3.4 算法参数说明 19](#_Toc516599273)

[2.4 与其他工作不同之处 20](#_Toc516599274)

[2.4.1 使用汉明距离衡量网络流数据相似度 20](#_Toc516599275)

[2.4.2 对网络流数据进行哈希编码 21](#_Toc516599276)

[3 实验结果与分析 22](#_Toc516599277)

[3.1 实验数据集 22](#_Toc516599278)

[3.2 数据预处理 22](#_Toc516599279)

[3.2.1 数据缺失值和空值处理 23](#_Toc516599280)

[3.2.2 命名型属性处理 23](#_Toc516599281)

[3.2.3 冗余值处理 23](#_Toc516599282)

[3.2.4 属性重排列 23](#_Toc516599283)

[3.3 实验环境 23](#_Toc516599284)

[3.3.1 硬件环境 23](#_Toc516599285)

[3.3.2 软件环境 23](#_Toc516599286)

[3.4 实验结果 24](#_Toc516599287)

[3.4.1 分类评价标准 24](#_Toc516599288)

[3.4.2 参数分析 25](#_Toc516599289)

[3.4.3 分类性能 25](#_Toc516599290)

[3.4.4 和传统KNN算法的对比 31](#_Toc516599291)

[3.4.5 实验结果分析 33](#_Toc516599292)

[结 论 34](#_Toc516599293)

[参考文献 36](#_Toc516599294)

[致 谢 40](#_Toc516599295)

# 引 言

随着经济发展水平和人民生活水平的不断提高，近年来我国互联网发展水平稳中有升，体量规模近一步扩大。截至2017年12月，我国网民总数达到7.72亿，互联网普及率达到55.8%，较2016年同期提升了2.6%[[[1]](#endnote-2)]。随着网络部署与应用规模的不断扩大，网络应用的种类与形式益发丰富，即时通讯、网络游戏或视频音乐等类型的应用增势明显。网络基础设施不断完善，无论是宽带接入端口数量、光缆线路长度还是互联网数据中心体量均保持着稳定增长，为国内网络技术的应用和发展提供了坚实基础。

随着互联网持续的高速发展，网络的建设、管理和维护面临着更高的要求。而网络流量的识别与分类技术是网络维管的重要组成部分，在实际工作中发挥着关键性作用。对于网络管理者，网络流量分类是网络状况监控与管理的基础。对于运营商，网络流量分类反映出网络用户流量特征，为预测网络应用的流行趋势提供了数据基础，有助于运营商对网络设备进行针对性的维护和升级。对于网络监管部门，网络流量分类技术在防止网络攻击、网络犯罪以及净化网络环境方面不可或缺。

鉴于网络流量分类在网络应用领域的重要作用，网络流量分类与识别成为网络研究领域的热点内容之一。在不同的时期，研究者们提出和实现了多种不同的方法。目前主流的方法主要分为四类：基于端口匹配、基于深度包检测、基于网络协议解析和基于统计特征学习，这四种方法适用于不同的网络环境，但是均存在一定的缺陷。事实上，考虑到网络环境的复杂多变性，设计具有良好时空稳定性、可扩展性和鲁棒性的网络流量分类器仍具有一定难度。

此外，实际网络发展中出现的新趋势和新特征值得研究者注意。网络骨干网和边缘网带宽大幅度增长，我国2017年国际出口带宽已超过700万Mbps，20M以上宽带用户占比超过90%[1]，而目前普通的服务器与网络设备难以处理高带宽下的网络流量分类任务；应用大量使用私有化协议与加密通信，解析和获取协议特征以及获取网络载荷遇到巨大障碍，难以对网络流量数据进行标记导致相关研究陷入找不到高质量网络流量样本的困境；P2P传输逐渐流行，而其使用的网络协议模型与传统TCP/IP模型存在诸多差异，甚至出现即使识别为P2P协议却无法判断具体类型的问题。真实网络中的网络流量分类系统往往需要在很短的时间内分析大量数据，而现代的网络流量不再和某些特定特征如源端口和目的端口有强依赖关系，分类处理时需要将大量特征纳入考虑，此外强调实时性和准确性的同时，实际网络分类设备所配置的CPU与内存资源有限，造成了较大的开销和有限资源间的矛盾。以上列举的种种新挑战无疑为网络流量分类带来了新的研究难点，增大了网络流量分类问题的研究难度。

近年来，大数据学习成为研究人员关注的焦点，其中基于哈希的大数据近邻检索技术的基本思想为将数据的特征映射为哈希值，再通过汉明距离或哈希表寻找近邻值。高维数据通过某种方法编码为哈希码后，能够以非常少的空间储存，同时在计算相似度和查询近邻时能将原数据间的欧式距离计算转为哈希值间的汉明距离计算，从而大大降低运算所需的开销。该方法的关键在于如何将数据编码为哈希码同时尽可能地保持数据间的相似性。基于哈希的近邻检索技术已被应用于相似图像的检索中，在部分应用场景中具有非常不错的效果。鉴于基于哈希的大数据近邻检索的诸多优点，能否将该技术应用到网络流量分类中从而实现一种基于哈希编码的网络流量分类方法将是本文研究的重点。

有关网络流量分类的研究随着网络和互联网的诞生便已经开始。时至今日，现有的方法或多或少均存在一定的缺陷，现代网络对分类方法的要求苛刻、变化迅速，如何实现分类性能好、资源开销低、可扩展具鲁棒性的网络流量分类设计将一直成为研究重点。本文将通过研究不同的网络流量分类方法，掌握网络流量分类的基本原理，分析网络流量分类模型中的关键部分，实现基于哈希编码的网络流量分类方法的研究。

基于哈希编码的大数据近邻检索技术的关键在于哈希算法，哈希操作结果的好坏将直接影响最后的分类和识别的效果。将网络流量数据转换为哈希码的过程中，需要权衡哈希码的长度。一般地，哈希码越长，所能保存的信息越多，最后分类的准确率越高。但是，哈希码过长可能造成数据稀疏，影响检索性能，并且在网络流量规模大的情况下哈希码将占用更多的空间。对基于哈希编码的大数据近邻检索的研究需要关注如何找出最合适的编码方式以实现最短的编码和最好的识别准确率。

网络数据采集设备能够从某一实际网络流量中采集到超过200个特征，因此网络流量数据的维度较高，如果直接处理则相当容易陷入“维度灾难”，而对于高维特征数据进行处理是哈希方法具有应用优势的领域。本文将通过基于乘积量化的哈希编码方式将网络流量数据编码为整数类型的哈希索引，并利用大数据近邻检索技术对编码后的网络流量数据进行学习与训练构建分类模型，通过优化模型参数和实现细节以达到最好的分类效果和性能。随后使用实际中采集的网络流数据集对实现的流量分类模型进行测试，比较模型中不同实现方式性能间的差异。

本文在组织结构上分为五部分，每部分的内容如下：

引言介绍了本文的研究背景、研究内容和研究目标等内容，同时简单说明本文的组织结构。

第一章主要介绍与本文研究内容密切相关的技术以及研究现状，主要包括网络流量分类相关研究、基于哈希的大数据近邻检索技术等相关内容。

第二章描述了一种基于哈希编码的网络流量分类方法的算法，主要包括算法的具体实现、算法复杂度分析和对算法参数的说明。

第三章主要是对第二章所提出的网络流量分类方法进行了实验验证，并列举相关实验数据同时进行对比分析。

结论部分主要对本文提出的基于哈希的网络流量分类方法在实验中表现出的具体性能进行了分析和总结，指出了该方法的改进之处。

# 1 相关技术概述

## 1.1 网络流量分类

### 1.1.1 网络流

一段时间内，具有相同源IP、目的IP、源端口、目的端口和传输层协议的网络报文序列为一条网络流，其中源IP、目的IP、源端口、目的端口、传输协议称为五元组，网络流分类基于上述五元组进行分类。

### 1.1.2 网络流特征

网络流特征是在网络流传输中得到的统计特征，Moore等人列举了246种统计特征，包括端口号、包大小以及SYN和FIN状态码信息等[[[2]](#endnote-3)]，下面列举出Moore统计的特征中24种在网络流量分类研究中常用的特征。

表1.1 24种常用网络流量特征

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 编号 | 特征名 | 序号 | 编号 | 特征名 |
| 1 | 1 | Server Port | 13 | 34 | ack pkts sent b a |
| 2 | 2 | Client Port | 14 | 43 | unique bytes sent a b |
| 3 | 3 | min IAT | 15 | 44 | unique bytes sent b a |
| 4 | 6 | mean IAT | 16 | 45 | actual data pkts a b |
| 5 | 8 | max IAT | 17 | 46 | actual data pkts b a |
| 6 | 9 | var IAT | 18 | 47 | actual data bytes a b |
| 7 | 17 | min data ip | 19 | 48 | actual data bytes b a |
| 8 | 20 | mean data ip | 20 | 59 | pushed data pkts a b |
| 9 | 23 | var data ip | 21 | 60 | pushed data pkts b a |
| 10 | 31 | total packets a b | 22 | 61 | SYN pkts sent a b |
| 11 | 32 | total packets b a | 23 | 62 | FIN pkts sent a b |
| 12 | 33 | ack pkts sent a b | 24 | 63 | SYN pkts sent b a |

### 1.1.3 基于端口匹配

根据国际互联网代理成员管理局(IANA)所建议的TCP/UDP协议端口号，0~1023号端口为固定端口，为IANA所保留，1024-49151号端口为已注册协议或应用的端口[[[3]](#endnote-4)]。通过分析网络数据包的源端口和目的端口并对照IANA端口注册表进行映射，便可以获取该网络流量对应的服务以及使用的传输层及应用层协议从而完成分类和识别。这种方法在实现上较为简单，也容易实现硬件支持，发展较为成熟。但是随着新型网络协议如P2P和被动FTP等的出现，网络通信者间往往采用随机端口或者双方协商的任意端口，采用该方法容易受到伪装端口或随机端口的欺骗，流量分类的准确率较低。Wei Li等人通过实验发现，该方法容易受到干扰，在某些情况下准确率低于30%[[[4]](#endnote-5)]。尽管准确率较低，但是这种方法开销相对而言非常少，并且端口分析是网络流量识别与分类中仍有重要价值，因此该方法依然在实际应用中存在一定价值。

### 1.1.4 基于深度包检测

早期的包检测方法仅仅检测网络层IP包报头和传输层TCP/UDP报头，而深度包检测不仅检测数据包报头，还会对数据包的载荷进行检测，一般会要求检测超过64字节的载荷内容，检测时对载荷内容进行特征和关键字匹配从而实现对网络流量的分类。总的来说，基于深度包检测的分类技术准确率较高，Moore等人提出一种基于载荷检测的流量分类方法，通过实验验证该方法的准确率达到了99.9%[[[5]](#endnote-6)]，但是该方法实现较为复杂，开销较大。与基于端口匹配的流量分类方法仅仅关心网络流的端口信息不同，深度包检测通过分析网络流传输过程中载荷的内容并从中提取相关的特征内容进行分类，往往能够得到准确的结果，但分析过程中分类系统需要大量的计算以及存储，可能导致网络分类设备的复杂规模和高昂成本。此外随着加密载荷和加密传输的出现，网络流量数据包的载荷对于分类设备而言变得难以分析或是不可见，因此基于深度包检测的方法面临着严峻的挑战。目前基于深度包检测的网络流量分类技术已经被广泛地用在诸多的商业化产品中，例如思科公司的SCE产品线和华为公司的SIG产品线中，在实际的网络管理和运维中发挥巨大作用。

付文亮等人基于FPGA设计了深度包检测技术的网络流量分类架构RocketTC，该架构能够对超过90种网络协议进行识别，准确率达到97%，并且高速处理性能好，但未对加密流量进行分析与验证[[[6]](#endnote-7)]。陈传通对正则表达式匹配应用与网络流量分类进行了研究与实现，利用有穷自动机和加权分组算法提高了处理速度和准确性[[[7]](#endnote-8)]。

### 1.1.5 基于协议解析

网络协议是规定网络流中交换的数据格式以及有关的同步问题的规则，而基于协议解析的网络流量分类方法正是通过解析网络流量所使用的协议来识别网络流量类别。基于协议解析的网络流量分类方法的实现方式根据协议类型有所不同。对于开放的协议，如TCP/UDP等协议，根据RFC文件规定的协议状态机信息或者协议特殊的流量行为后识别协议特征实现网络流量分类。对于新型协议或未知协议，则需要通过软件逆向方法分析协议特征后进行流量分类。

Caballero Juan等人提出称为“投影”的协议逆向方法，并设计出Polyglot系统采用动态方法得到协议相关特征[[[8]](#endnote-9)]。 Yan Hu等人对P2P应用的流量进行了分析，归纳了P2P不同种类流量的特征并提出了多条检测规则[[[9]](#endnote-10)]。Bonfiglio Dario等人在研究Skype的过程中对UDP消息格式进行了逆向研究，对每个可抽取信息的部分如ID、Fun和Frame等进行解析，取得了不错的效果[[[10]](#endnote-11)]。杨季等人改进了基于子图模式的网络流量分类方法，利用传输层子图连接模式的差异进行分类[[[11]](#endnote-12)]。

### 1.1.6 基于统计学习

基于学习的网络流量分类是建立在网络流量统计特征与机器学习之上的流量分类方法，首先采集网络流量的流统计特征作为训练数据，再利用机器学习算法训练得到网络流量分类模型后即可对未知类别的流量进行分类。其中采用的机器学习算法种类众多，包括决策树、SVM、遗传算法、神经网络和朴素贝叶斯等，准确率普遍高于90%。但是基于统计学习的网络流量分类方法存在的普遍问题是对训练数据依赖性非常强，往往对自身训练建模的特定网络数据集中的流量识别性能较高，但是对其他数据集中的流量识别准确率较低。实际上找到一种适应所有网络环境、鲁棒性的学习分类模型是难得的，尤其是在应用到网络拓扑复杂、网络流量巨大的实际网络中的情况下。

Carlos Bacquet等人提出了一种基于多目标遗传算法的识别方法，初始选择38个流特征并将流量表示为60位的二进制串，经过优化选择了最优的14个特征，预测准确率较高的情况下误报率也较高[[[12]](#endnote-13)]。于孝关等人利用C4.5决策树进行网络流量分类，分类平均准确率达到93%[[[13]](#endnote-14)]。 Riyad [Alshammari](http://hosted.cceu.org.cn:1701/primo_library/libweb/action/search.do?vl(freeText0)=Alshammari%20%2c+Riyad%20+&vl(12428426UI0)=creator&vl(54437159UI1)=all_items&fn=search&tab=default_tab&mode=Basic&vid=dlut&scp.scps=scope%3a(DLUT)%2cscope%3a(pqdt)%2cprimo_central_multiple_fe&ct=lateralLinking)等人使用C5.0、AdaBoost、GP三种算法研究了一种用于识别加密VoIP流量的机器学习算法[[[14]](#endnote-15)]。 Pan等人设计了一种基于WENC和马尔科夫链的方法用于解决HTTPS加密流量分类问题，平均分类准确率达到了90%[[[15]](#endnote-16)]。Jin Yu等人运用多种线性二进制分类器实现了一种模块化的大型网络机器学习分类模型[[[16]](#endnote-17)]。Erman Jeffrey等人提出了一种基于K均值聚类的网络流量分类方法，在S-C方向上和C-S方向上流级准确率都达到了90%[[[17]](#endnote-18)]。Yuan等人实现了一种基于SVM的网络流量分类方法，对校园骨干网中收集的7类流量进行验证，结果表明对于无偏样本准确率达到96.9%，对于有偏样本准确率达到99.4% [[[18]](#endnote-19)]。邓河等人同样实现了一种基于SVM的分类方法对P2P流量进行分类，分类准确率达到了92.38%[[[19]](#endnote-20)]。周文刚等人实现一种基于K均值算法的半监督网络流量分类方法，算法的平均准确率高于90%[[[20]](#endnote-21)]。

## 1.2 基于哈希的大数据近邻检索

### 1.2.1 近邻检索

对于一个数据点q，现在需要从数据集C={c1,c2,c3,c4,…，cn}中找到q的最近邻点p，p满足

(2.1)

其中d是数据的维度，dist是数据间的距离，反映数据间的相似度，如果取欧式距离，则

(2.2)

找到满足条件的p点为最近邻检索，而最近邻检索可以扩展到k近邻检索即KNN（k-NearestNeighbor），将上述寻找p的过程扩展到在n个数据中寻找q的k个近邻点。

目前主要的近邻检索实现方法有以下几种：

(1) 线性扫描方法

线性扫描方法是最简单最朴素的近邻检索实现方法，对于查询数据，逐一计算其与检索数据间的距离，最后排序得到结果数据。这种方法的时间复杂度与需要检索的数据数量和维度成线性关系，在数据量较小的情况下，性能尚可接受，但在大规模数据和高维度数据的近邻检索中线性扫描方法开销过大。

(2) 基于树结构的检索方法

基于树结构的近邻检索方法通过树的数据结构来提升检索性能，其中最经典的算法是KD树算法[[[21]](#endnote-22)]，KD树算法近似于多维空间中的二叉树，不断地使用方差最大的维度的作为分类维度，以该维度的中值作为分类依据将数据点划分到两部分中，最终实现KD树的每个节点上只有一个数据点。查询时从根节点开始，不断向下对分支进行搜索，最终找到近邻。KD树在低维空间中效果尚可，但是在高维空间中检索性能下降明显。

KD树的改进版本有很多，如量化KD树（quantized kd-Tree）[[[22]](#endnote-23)]、Bkd-Tree[[[23]](#endnote-24)]，同时一些其他的方法将分支的划分依据改为旋转矩阵（PCA树，PCA-tree)[[[24]](#endnote-25)]或随机投影向量（随机投影树，Random Projection Trees）[[[25]](#endnote-26)]等，它们的检索效果不同，但是核心思想都是基于树结构划分数据来提高检索性能。

(3) 基于哈希的检索方法

基于哈希的近似近邻检索方法和上面的方法有所不同，哈希方法的基本思路是降维，即将高维的原始数据通过一定的方法映射为哈希值。与原始数据相比，哈希码值在计算开销和存储开销上有较大的改善，理论上能够实现性能良好的近似近邻检索。但是哈希的过程是将原始数据少则十几个特征多则上百个特征映射到几十位的01串上，不可避免地存在失真的问题。因此基于哈希的近似近邻检索方法的关键在于选择出色的哈希方法，使得在尽可能短的哈希串中反映尽可能多的原始数据特征，哈希降维方法应该保证在原始数据空间中相似的数据在汉明空间内依然相似。而近似近邻检索中运用的哈希和传统的哈希有所区别，近似近邻哈希要求相似数据产生更多碰撞，而传统的哈希更期望相似数据能够尽可能避免碰撞以实现最佳的散列效果。

目前基于哈希的近似近邻检索主要有三种实现方式，基于随机投影、基于学习和基于乘积量化，下一节将对这三种实现方式做详细介绍，而本文主要基于乘积量化方法进行实现。

大数据近邻检索技术应用的经典场景之一是相似图像检索。图像被提取出多维特征向量完成图像到数据的转换，数据的每个维度均表示图像的某个特征，如颜色特征、纹理特征等。图像的相似度使用数据向量间的距离表示，在检索相似图像时在海量的图片库中快速寻找向量距离最近的k个向量并将对应的图像作为检索结果输出。大数据近邻检索技术在图像检索上的准确率较高，具有非常丰富的应用，





图1.1 近邻检索用于图像处理

现在举例说明基于哈希的近邻检索的基本原理，假设对上面的三张图片进行哈希编码，如果选用的哈希方法合理则相似的两张表示绿树的图片编码可能编码为01011110和01011100，相互间汉明距离为1，而和另外两张图片差异较大的表示高山的图片可能编码为10010001，与图1和图2间的汉明距离分别为6和5，因此我们通过比较汉明距离来可以判断图1和图2较为相似，而与图3差异较大从而实现对相似图片的分类和识别。

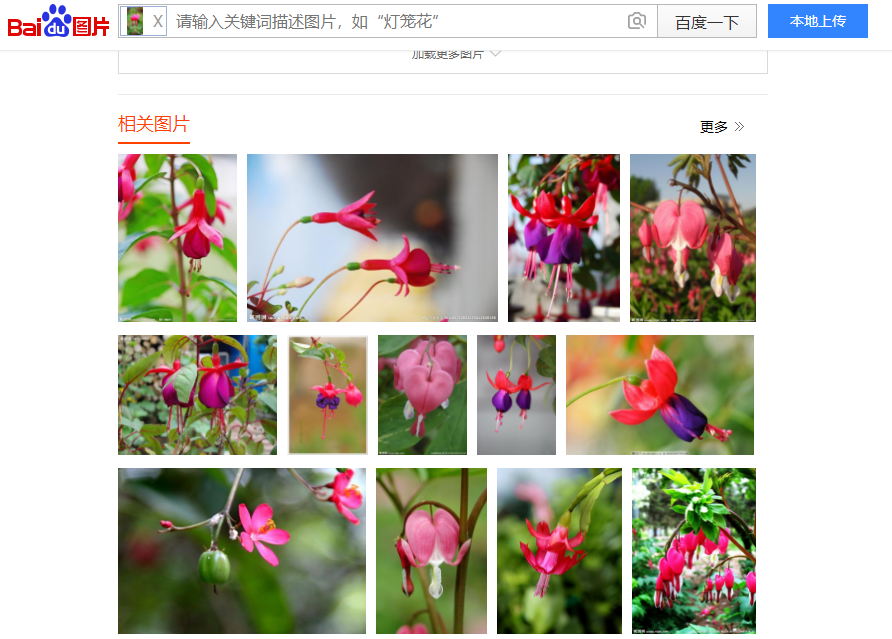


图1.2 商业识图软件

上面是百度公司提供的识图功能[[[26]](#endnote-27)]，可以看到根据提供的图片，软件自动查询到许多相似的图片，其中的原理正是大数据近邻检索。

### 1.2.2 基于随机投影的哈希算法

基于随机投影的哈希算法种类众多，通常是通过函数或者函数族将数据转换为哈希值。

基于随机投影的典型例子是LSH算法，LSH(Locality SensitiveHashing)算法的原理是将相似的数据以尽可能大的概率映射到相同的哈希桶中去，由Indyk Piotr[[[27]](#endnote-28)]等人提出并得到A Gionis等人[[[28]](#endnote-29)]的证明和完善，LSH算法定义了局部敏感哈希函数族和一个函数属于该函数族的条件，最终通过求解满足局部敏感的函数得到哈希编码。

Charikar M提出了一个利用p-稳态分布的随机超平面来分割空间来对数据进行0和1赋值的方法，该方法在形式上非常简洁[[[29]](#endnote-30)]。Datar等人和Chum等人分别提出了基于欧式空间的哈希算法[[[30]](#endnote-31)][[[31]](#endnote-32)]，其特点是不将数据压缩为二进制编码。而对LSH算法的改进则有核化局部敏感哈希算法(Kemelized Locality SensitiveHashing)[[[32]](#endnote-33)]和平移不变核化哈希算法(Shift-Invariant Kernel Hashing)[[[33]](#endnote-34)]等，这两种方法的创新之处是将函数划分的空间扩展到核空间。林朝辉等人提出了高维分布式局部敏感哈希方法，对哈希索引的表达在分布式环境中进行了优化[[[34]](#endnote-35)]。

基于随机投影的哈希算法往往在哈希编码较长时性能表现较好，而在哈希编码较短时编码性能不如其他方法[[[35]](#endnote-36)]。为了达到更佳的哈希编码效果，基于学习的哈希算法是基于随机投影的哈希方法的有力补充。

### 1.2.3 基于学习的哈希算法

基于学习的哈希算法和基于随机投影的方法有所不同，往往是通过不同算法挖掘出数据内部性质和规律并反映到特定的目标函数中，再通过目标函数对数据进行求解得到哈希编码。

基于主成分分析的哈希(Principal Component Hashing)算法，会考虑数据的分布情况，使用数据的若干主方向来训练哈希函数[[[36]](#endnote-37)]。Weiss Y等人提出的谱哈希算法(Speclxal Hashing)是一种挖掘训练数据存在的谱性质的哈希算法，最后求解相应的三角函数来获取特征函数[[[37]](#endnote-38)]，对谱哈希算法的改进还有稀疏谱哈希（Sparse spectral hashing）[[[38]](#endnote-39)]，反转谱哈希（Reversed Spectral Hashing）[[[39]](#endnote-40)]。Gong Yunchao等人提出了迭代量化哈希算法(Iterative Quantization Hashing)，通过一种基于旋转的交替最小化算法以实现量化误差最小值，实验证明哈希整体性能也得到极大的提升[[[40]](#endnote-41)]。Paulevé Loïc等人提出对数据预先进行处理，之后进行哈希编码的做法从而提升算法性能[[[41]](#endnote-42)]。W Liu等人提出了基于锚点图的哈希算法(Anchor Graph Hashing)，通过基于锚点图来发现数据中存在的邻域结构来紧凑表达编码从而实现更好的哈希效果[[[42]](#endnote-43)]。Heo Jae-Pil等人在2012年的CVPR会议提出了球哈希算法(Spherical Hashing)，将分割的依据从传统的超平面扩展到球面，实验结果表明创新性地用球面分割数据点使得检索的准确度得到了很大的提升[[[43]](#endnote-44)]。

基于学习的哈希算法的特点是在哈希编码较短时效果比基于随机投影的方法效果好，但是在编码长度变长时效果提升不明显，甚至不如基于随机投影的方法。

### 1.2.4 基于乘积量化的哈希算法

(1) 向量量化

向量量化是一种降维的方法，核心思想是将数据通过编码来表示。具体的实现方法是寻找量化函数q满足：

q(x), 其中i=1,2,3,…,k ,x (2.3)

量化函数q称为量化器（quantizater），C称为码本（code book），k为码本中码字（code word）的数目，量化得到的ci称为质心。码字所包含的特征向量集合是用表示：

(2.4)

向量量化的过程中不可避免地存在量化误差，量化误差的大小反映出量化器的好坏，一般用向量x与量化得到的质心q(x)间的欧式距离来表示量化误差，对于量化器q的误差，引入均方误差（Mean Squared Error）：

(2.5)

其中d(x, y) = 计算x与y间的欧式距离，p(X)是随机变量X的概率分布函数，对于特定x，p(x)表示随机变量X落在该点的概率。

MSE（q）是所有数据到其量化后质心的欧式距离的期望，明显在MSE值更小时，量化器q的误差较小，量化性能更好。而对于特定数据而言，经过多次调整，理论上一定能找到满足使量化误差最小的量化器，这样的量化器为最优的量化器。一般用Llyod条件来表示最优量化器满足的条件：

(2.6)

(2.7)

式一表示量化器需要将向量x量化到距离最小的质心。式二表示量化的质心需要是向量的期望值。

(2) 乘积量化

乘积量化是一种在量化思想的基础上发展的量化方法，乘积量化的最大特征是将向量划分为子向量分开量化：

(2.8)

其中为子向量，m为子向量数量，为子量化器。

乘积量化将原向量划分为m个子向量，再在子向量空间中对子向量进行量化，每个子向量空间均有码本，最后原向量的量化空间会是子向量的笛卡尔积：

(2.9)

假设每个子向量码本中有个码字，则整个向量空间中的码字总数为，当相同为时，码字总数为。值得注意的是当m=D时，乘积量化退化为标量量化。

乘积量化将向量分为子向量量化有利于处理高维数据，对于子向量而言，因为维度的降低，训练得到最优的量化器更简单，所需要的训练样本更少。

乘积量化的误差同样可以用均方误差来衡量：

(2.10)

实际上影响乘积量化性能的参数有两个：子空间中码字的数量k和子空间的数量m，k和m的选择影响MSE的大小，下图反映了MSE与k，m取值的关系。

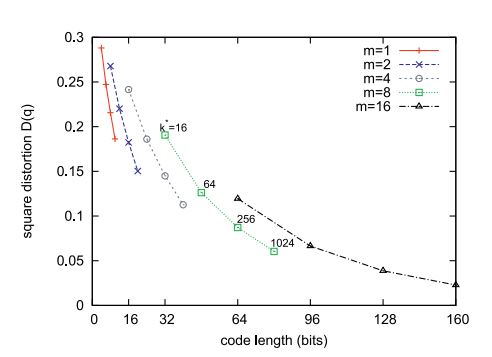


图1.3 乘积量化误差分析

code length为乘积量化最终形成的编码长度，满足，从上图可以看出在编码长度一样的情况下，m较小、k较大的量化器性能比m较大，k较小的量化器性能要好。

经典的乘积量化近邻检索中计算向量间距离可以通过两种方式进行计算，一种是对称的距离计算方法（Symmetric distance computation , SDC），另一种是非对称的距离计算方法（Asymmetric distance computation , ADC）。以下将已经量化完成的向量称为数据向量，将需要检索的向量称为查询向量。两种方法的区别在于查询向量是否用量化后的质心代替原数据计算距离。

假设查询向量为x，而数据向量为y，如果应用SDC方法，则距离为，即查询向量已经被量化，此时计算距离直接计算查询向量量化后对应的质心和数据向量量化后对应的质心间的欧式距离。而如果应用ADC方法，则距离为，即计算查询向量和数据向量量化后对应的质心间的距离，两种距离的具体表示如下：

(2.11)

(2.12)

下图为两种方法的示意图：

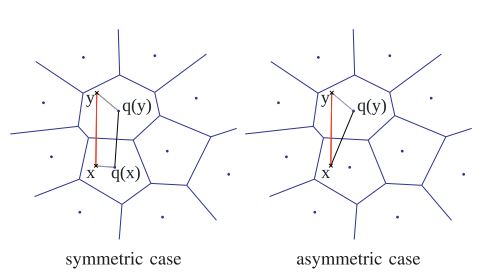


图1.4 SDC和ADC示意图

可以看到ADC方法得到的距离要比SDC方法得到的精确，但是ADC方法需要存储查询向量原始数据并在线进行m次欧式距离计算，而SDC方法量化得到的码字可能已经存在于码本中，只需要取出离线计算后存储的码字间的距离即可。两种方法都需要计算欧式距离，而本文实现了一种用汉明距离代替欧式距离的计算方法来进一步节省时间和空间花销，该方法将在之后说明。

上述的乘积量化方法是一种非常朴素的量化方法，乘积量化过程中并没有考虑量化后量化数据和原始数据间的残差，所以在算法性能上存在较大的改进空间。目前对于乘积量化的改进有以下几种：

Ge T 等人提出的OPQ(Optimized Product Quantization)通过分解出两个子问题以及基于高斯假设的方法最小化量化失来优化PQ[[[44]](#endnote-45)]，Kalantidis Y 等人提出了LOPQ(Local Optimized Product Quantization)算法，通过对每个子空间进行基于旋转和空间分解的局部优化并将粗量化残差编码的方式提高准确率和检索速度[[[45]](#endnote-46)]，L Li等人提出的DSPQ(Distribution Sensitive Product Quantization)方法根据数据的聚合程度动态比较数据分布和改变位排列以提高检索效率[[[46]](#endnote-47)]。

可以看出，乘积量化和上述基于随机投影和基于学习的哈希方法存在一定的不同。后两种方法归根结底是依据哈希函数对数据进行计算求解得到哈希值，而基于乘积量化的哈希方法并不需要通过经验或者学习得到哈希函数才能得到哈希值，甚至处理的最终结果并不一定是传统上的01串而可能仅仅是一个数据点在码本空间内的索引值，即码字的索引。无论在形式上还是原理上，这样的编码方式与基于随机投影和基于学习的哈希方法都不一致，但是基本的原理依然是将原始的高维数据从欧式空间映射汉明空间的哈希索引上，并且都期望数据信息的损失和失真尽可能的小。本文对乘积量化的应用主要是其基本思想，对于许多优化过的乘积量化算法并不涉及过深。

# 2 基于哈希编码的网络流量分类方法

## 2.1 引子

随着网络技术的进步，网络流量分类对于新的实现思路和方法有着更多的诉求。这些诉求主要集中在两方面：（一）更低的计算和存储开销（二）适合处理大数据（三）对于新型流量的识别能力增强。哈希方法将数据编码为二进制串的特性，同时汉明距离相对于欧式距离在计算上的高效性意味着运算时消耗更少的计算和存储开销，适合于高维网络流量数据的处理。关键的问题是如何将网络流量数据映射为哈希编码，满足使不同类别的网络流量尽可能地避免汉明空间内的碰撞而相同类别的网络流量尽可能地发生碰撞。为了解决这个问题，本文结合乘积量化的哈希方法和KNN近邻检索算法提出了一种基于哈希编码的网络流量分类方法。这种方法所构建的分类模型在线下训练阶段利用乘积量化思想和K均值算法对网络流量数据进行学习和分类，从而得到哈希索引进行编码以实现网络流量数据的降维，而在实时分类阶段可以利用离线学习阶段得到的编码快速地利用KNN近邻检索算法对查询网络流量进行分类。相比较其他方法而言，KNN算法中数据距离的计算从欧式空间转换到汉明空间，从而大幅了降低计算和存储开销。而一旦哈希的过程中数据压缩效果不佳，哈希索引性能变坏，原高维网络流量数据映射到汉明空间后将不可避免地出现编码混淆，KNN查询分类的精度必定下降。在权衡算法的开销和分类精度上我们需要对关键参数进行优化和验证。

## 2.2 相关算法

### 2.2.1 K均值算法

K均值(K-Means)算法是一种应用广泛的数据聚类方法，作为一种无监督聚类方法，K-Means实现非常简单，但聚类效果却相当高效。经典的K均值算法的基本思想是在n个数据中寻找k个中心作为类中心，并将每个数据划分到最近的类中，随后在每个类中更新类中心，对每个数据重新划分类，迭代多次直到类中心不再变化。对数据划分类型的衡量标准一般为数据间的欧氏距离。在本文实现的网络流量分类模型中，K均值算法用于线下训练阶段，这个阶段计算网络流量数据间的距离我们使用欧式距离。

K均值算法的伪代码实现和流程图如下：

Begin

随机选取k个质心

do

对n个数据

根据与质心距离划分到k个聚类中

对每个聚类

计算每个聚类中数据均值作为新的质心

util 质心不再变化

End

影响K均值算法性能的因素为初始质心的选取和质心的数量即k值选取，本文在运行K均值算法主要关注k值的变化对算法性能的影响。

### 2.2.2 KNN算法

KNN即K近邻算法和K均值算法同为数据挖掘领域的经典算法，KNN算法与K均值算法不同，它是一种监督算法。KNN的基本思想是对于某个要查询的数据，在训练数据中找到k个相距最近的数据点，根据最大表决原则即统计k个近邻数据类别标签的众值作为预测查询的数据类别的结果。本文在实时分类阶段使用KNN算法，因此衡量相似度的标准是数据间的汉明距离。

KNN算法的伪代码实现如下：

Begin:

计算查询数据与训练数据间的距离

排序得到最小的k个距离

统计k个距离中对应的数据标签

输出出现最多次的标签

End

影响KNN算法性能的主要因素为k值的选取，当k=1时，KNN退化为最近邻检索，k值太小时，KNN算法的结果对数据中“坏值”非常敏感。而较大的k值能够减少数据“坏值”的影响，但是容易受到相距较远的数据的影响，这一点在样本数据较少时非常明显。

## 2.3 算法描述

基于哈希编码的网络流量方法主要分为两部分：训练部分和查询部分，训练部分是通过对训练数据的学习得到分类模型，主要运用K-Means方法进行聚类从而实现对训练数据进行哈希编码，最后实现高维网络流量数据到汉明空间的映射，而查询部分主要应用KNN算法实现对查询流量的预测和分类。整个算法的解决方案流程图如下：

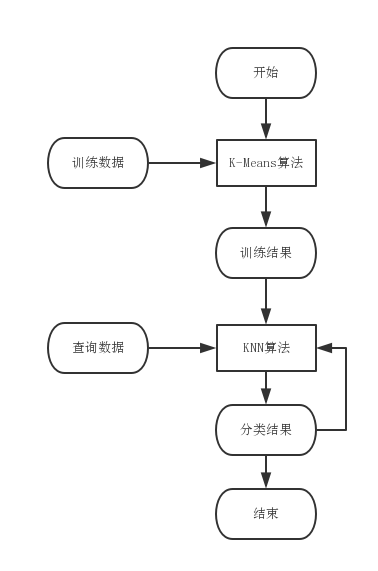


图2.1 整体流程示意图

下面将分为线下训练和实时分类两部分对基于哈希编码的网络流量分类进行说明。

### 2.3.1 线下训练阶段

线下训练阶段的主要工作是对训练数据集进行聚类并根据聚簇得到网络流数据的哈希编码，本文采用的聚类方法是基于乘积量化和K均值算法的聚类方法。具体实现方式是先将网络流量数据划分为多个子向量再在每个子向量空间内运行K均值算法进行聚类，依据欧式距离的大小网络流数据被划分为不同的聚簇，最后将网络流量对应的聚簇索引作为哈希编码输出，因此这里的哈希码并不是01串而是十进制数值。离线训练过程的输出有两个，一是训练数据的哈希编码集合，二是训练过程得到的K均值聚类模型，这些聚类模型将在实现分类过程中对查询的网络流数据进行哈希编码。该过程在线下进行，对于在线的网络流量分类此过程的时间开销可以忽略。

算法1：乘积量化编码算法

输入：网络流量训练数据集train\_set，子向量数M，子空间内聚簇数V

输出：训练数据集哈希编码train\_hash，K均值聚类模型kmeans\_model

ProductQuantizationHashing(train\_set, V, M)

Begin

sub\_set[] = train\_set.split(V);

kmeans\_model[] = Kmeans[M];

for set in sub\_set:

kmeans\_model[i] = Kmeans(set, V);

for data in train\_set :

train\_data =data.split(M) for sub\_data in train\_data:

sub\_code = kmeans\_model[train\_data.index(sub\_data)].predict(train\_data[j]);   
 hash\_code.append(sub\_code)

train\_hash.append(hash\_code)

return train\_hash, kmean\_model

End

### 2.3.2 实时分类过程

实时分类过程主要是对某一待分类的网络流量进行哈希编码，并利用KNN算法检索近邻得到流量的类别。具体的实现过程是对查询向量进行划分，划分得到的子向量数应与线下训练阶段时的流量数据划分得到的数目相同，之后利用线下训练阶段得到的K均值聚类模型对每个子向量进行哈希编码，计算训练数据和查询数据间的汉明距离，最后以汉明距离作为衡量相似性的依据运行KNN算法预测流量类别。需要注意的是训练样本数量过大时，简单的穷举检索并不适用，应考虑使用KD树加速KNN的检索过程，以下算法仅仅表示一种实现思路。

算法2：基于汉明距离的KNN分类算法

输入：查询数据query\_data，子空间数V，K-Means模型kmeans\_model，训练数据机标签train\_tag，训练数据哈希集train\_hash

输出：分类结果result

HammingClassifying(query\_data, V, kmeans\_model, train\_tag, train\_hash)

Begin

sub\_data[] = query\_data.split(V);

for model in kmeans\_model:

sub\_code = model.predict(sub\_data[kmeans\_model.index(model)])

hash\_code.append(sub\_code)

for train\_data in train\_set:

dist[index(train\_data)] = 0

for code in hash\_code:

d = hamming(code, train\_data[index(code)])

dist[index(train\_data)] += d

indexs = kmin(dist)

for index in index:

tag\_list.append(tag[index])

result = most\_item(tag\_list)

return result

End

### 2.3.3 算法复杂度分析

#### 1 线下学习阶段

线下学习阶段主要运行K均值算法，K均值算法的时间复杂度为O(dnkt)，空间复杂度为O(dn)，其中d为数据维度，n为数据量，k为聚类质心数，t为K均值算法迭代数。

#### 2 实时分类阶段

实时分类阶段主要运行kNN算法，kNN在使用KD树实现的情况下查找时间复杂度为O()，其中k为近邻数，n为数据量，d为数据维度，m为常数，可以看到在维度较高的情况下KD树性能下降。

### 2.3.4 算法参数说明

基于哈希编码的网络流量分类方法在线下学习阶段将高维的网络流量数据划分为多个维度较低的子向量，再对每个子向量进行聚类得到哈希编码，在实时分类阶段使用KNN算法查询近邻得到分类结果。影响算法性能的参数主要有数据划分的程度即子向量的个数M、在每个子向量空间聚类的聚簇数V、寻找近邻时的近邻数目k。对算法的优化主要是对以上三个参数进行调整，在下一步的实验中对于这三个参数的测试也将是验证算法性能的重点。

## 2.4 与其他工作不同之处

本文提出的基于哈希编码的网络流量分类方法在本质上和其他基于类别模型网络流量分类方法并没有区别，原理上都是通过训练数据进行学习建立映射关系，再利用映射关系和类别模型对查询数据进行计算得到预测的类别。存在许多相关工作使用其他分类算法如C4.5，朴素贝叶斯等对训练数据进行学习和建模后对网络流量进行分类。本文所实现的方法采用的是基于哈希编码的实现思路，与其他相关工作的不同主要有以下几点：

### 2.4.1 使用汉明距离衡量网络流数据相似度

在数据分类中，如何判断数据是否应该被判断一类即是关于相似度衡量的问题。目前采用的方法多是通过计算数据向量间的距离来表示相似度，而不必考虑这样处理数据的具体含义。而对于如何计算距离，数据处理者和分析者需要根据数据特征和算法特点来合理选择方法。

常见的距离计算方法有欧式距离（Euclidean Distance）、闵式距离（Minkowski Distance）、皮尔森相关系数（Pearson Correlation）和曼哈顿距离（Manhattan Distance）等，其中欧式距离非常适合属性为连续数值型的数据计算距离，而网络流量数据的绝大多数维度都是连续数值型，因此大部分研究都是基于欧式距离判断流数据的相似性。欧式距离的数学表示如下：

(3.1)

对于具有d个维度的数据，使用欧式距离计算时每个维度需要1次幂运算，d个维度总共需要进行d次幂运算和1次开方运算。当数据维度高，数据量大时，计算欧式距离速度受影响较大。

本文采用汉明距离（Hamming Distance）衡量网络流数据间的相似度，汉明距离的数学表示如下：

(3.2)

对于编码为d位的哈希编码，则计算数据间汉明距离时需要进行计算d次异或运算。相比较于欧式距离，汉明距离在计算开销和速度上更具优势。

基于哈希编码的网络流量分类方法将网络流量数据转换为哈希编码后利用汉明距离计算数据间的相似度，在大数据处理环境下，采用KNN等分类算法对网络流数据分类时能够减少大量计算开销，从而大大加速了分类算法的运行速度。

### 2.4.2 对网络流数据进行哈希编码

为了利用汉明距离在计算开销上的优势，基于哈希编码的网络流量分类方法基于乘积量化思想实现将网络流数据编码为哈希码，从而避免了网络流量数据的“维度灾难”，这在网络流量分类的研究中较为少见。传统的网络流量分类研究更注重基于较少的网络流特征来通过原始数据进行分类，而这样做忽略了大多数其他的网络流特征，由于哈希编码后对高维度不敏感，所以本文实现的方法能够选择更多的特征。

将数据映射为哈希编码往往通过构造哈希函数后求解方程来完成，在前文介绍的基于哈希的大数据近邻检索的三种实现方式中基于随机投影和基于学习的方式本质上都是基于哈希函数，而本文所实现的哈希编码采用了应用较少的乘积量化方式，乘积量化作为一种相对而言并不传统的哈希编码方式在实现上非常简洁，对于高维数据的哈希处理速度相当快。

# 3 实验结果与分析

## 3.1 实验数据集

实验数据集[[47]](#endnote-48)采用Andrew W. Moore 和Denis Zuev等人在文献[[[48]](#endnote-49)][[[49]](#endnote-50)]中使用的数据集，该数据集是一个经典数据集，诸多网络流量分类研究基于此数据集进行实验。该数据集由10个在同一天的不同时段得到的网络流数据集组成，每个数据集中包括不少于2万条的网络流数据，其中的数据由Moore实现的高性能网络管理器[[[50]](#endnote-51)]在全双工千兆以太网中收集。每条网络流统计有248个特征，并在最后标出该网络流种类标签，标签的标定来自于基于内容的分析，在文献2详细地描述了该数据集。

该数据集中网络流种类和数据量详见下表：

表3.1 数据集流量统计表

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | total |
| WWW | 18211 | 18559 | 18065 | 19641 | 18618 | 16892 | 51982 | 51695 | 59993 | 54436 | 328092 |
| MAIL | 4146 | 2726 | 1448 | 1429 | 1651 | 1618 | 2771 | 2508 | 3678 | 6592 | 28567 |
| FTP-CONTROL | 149 | 100 | 1861 | 94 | 500 | 48 | 83 | 63 | 75 | 81 | 3054 |
| FTP-PASV | 43 | 344 | 125 | 22 | 180 | 109 | 94 | 102 | 1412 | 257 | 2688 |
| ATTACK | 122 | 19 | 41 | 324 | 122 | 134 | 89 | 129 | 367 | 446 | 1793 |
| P2P | 339 | 94 | 100 | 114 | 75 | 94 | 116 | 289 | 249 | 624 | 2094 |
| DATABASE | 238 | 329 | 206 | 8 | 0 | 0 | 36 | 43 | 15 | 1773 | 2648 |
| FTP-DATA | 1319 | 1257 | 750 | 484 | 248 | 364 | 307 | 386 | 90 | 592 | 5797 |
| MULTIMEDIA | 87 | 150 | 136 | 54 | 38 | 42 | 36 | 33 | 0 | 0 | 576 |
| SERVICES | 206 | 220 | 200 | 113 | 216 | 82 | 293 | 220 | 337 | 212 | 2099 |
| INTERACTIVE | 3 | 2 | 0 | 2 | 0 | 1 | 25 | 26 | 29 | 22 | 110 |

续表3.1

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | total |
| GAMES | 0 | 1 | 0 | 0 | 0 | 0 | 3 | 0 | 3 | 1 | 8 |
| total | 24863 | 23801 | 22932 | 22285 | 21648 | 19384 | 55835 | 55494 | 66248 | 65036 | 377526 |

该数据集格式为arrf文件，这是一种由数据分析和挖掘软件Weka[[[51]](#endnote-52)]所使用的文件格式，Weka是由新西兰怀卡托大学使用Java开发的一款开源软件，我们可以在Weka软件中将文件转为csv等其他格式。

## 3.2 数据预处理

本文对实验数据的预处理较为简单，而Weka具有强大的数据预处理功能，本文直接使用它来预处理数据集，并将数据集转为csv文件。

### 3.2.1 数据缺失值和空值处理

在实际网络环境采集过程中存在数据缺失或是数据留空的情况，在使用数据集进行实验之前需要对数据集中的缺失值进行处理。考虑到空缺值在整个数据集中占绝少数，本文采用从简处理。填补空缺值的原则为：（一）对于数值型属性（numeric attribute），使用该特征其他数据的均值填补。（二）对于命名型属性（nominal attribute， 类似于枚举型），使用该特征其他数据的众值填补。填补操作使用Weka中的ReplaceMissingValues Fliter，这是一种非监督的数据Fliter。

### 3.2.2 命名型属性处理

数据集中存在为数不多的命名型属性，例如属性编号为65等属性，它们的取值为“Y”和“N”，不便于后续编程处理，因此需要将这些命名型变量转为二进制变量，即0和1。处理的原则为（一）“Y”转为1（二）“N”转为0。处理操作使用Weka中的NominaltoBinary Fliter，这是一种非监督的数据属性Fliter。

### 3.2.3 冗余值处理

数据方差过小的属性过于相似，对于分类器而言为冗余数据，本文将数据方差为0的属性删除，这一步通过Weka的RemoveUseless Fliter实现。

### 3.2.4 属性重排列

数据集中数据属性按照Moore采集顺序排列，本文按照数据的均值升序重新排列属性，使得属性按照值“越来越大”的顺序排列。这主要是考虑基于哈希的网络流量分类方法中线下学习时乘积量化运行K均值聚类算法时衡量数据相似度以欧式距离衡量相似度，当平均值较大的属性和平均值较小的属性划分到相同子矩阵时计算距离时均值较小的属性间的距离对结果影响忽略不计，出现均值较大的属性“掩盖”均值较小的属性的情况，导致造成不必要的信息丢失而影响分类性能。

## 3.3 实验环境

### 3.3.1 硬件环境

实验算法运行和性能测试均在配置为单核Intel(R) Xeon(R) CPU E5-2650以及2GB内存的主机中完成，操作系统为CentOS 6.8。

### 3.3.2 软件环境

实验算法由Python2.7编写完成，数据处理软件包括Weka与Excel软件。

## 3.4 实验结果

### 3.4.1 分类评价标准

衡量分类器性能好坏需要通过不同的分类指标来完成，现在简单介绍本文所使用的分类指标。

1. 分类评价术语
   1. 真正（True positives, TP）：正确分类的正样本数，类别为A的网络流量被预测为类别A
   2. 假正（False positives，FP）：错误分类的负样本数，类别不为A的网络流量被预测为类别A
   3. 真负（True negatives，TN）：正确分类的负样本数，类别不为A的网络流量被预测为其他类别
   4. 假负（False negatives，FN）：错误分类的正样本数， 类别为A的网络流量被预测为其他类别
2. 准确率

准确率（accuracy）表示对样本正确分类的比率，即分类正确的样本除以样本总数。一般来说，准确率越高，分类器性能越好。对于网络流量分类，分类正确表示网络流分类得到的类别标签和实际的类别标签匹配，否则为分类不正确。分类正确的样本数包括正确分类的正样本TP和负样本TN，准确率的计算公式为

(4.1)

1. 召回率

和准确率相比，召回率（recall）关注的重点在于正样本分类的效果，表示正样本被正确预测的比率，反映了分类器对某个类别网络流数据的预测准确率，其中TP为正样本被正确预测，FN表示正样本被错误预测，召回率的计算公式为：

(4.2)

1. 精度

精度（precision）表示预测为正的样本中正样本的比率，反映了实际预测中某种类别的网络流真正为此类别的概率，其中TP为正样本数，而FP为负样本数。精度的计算公式为：

(4.3)

1. 分类时间

分类时间（predict time）表示线上对于一个样本进行预测时需要的时间，反映了分类器的分类速度。注意分类时间并不包括对分类器进行训练的时间，因为分类器的训练在线下进行，对线上实时分类并无影响。

1. 鲁棒性

鲁棒性（robust）表示对于不同测试集的分类准确率稳定性，如果分类器的准确率稳定性好，则该分类器鲁棒性好。

### 3.4.2 参数分析

基于哈希编码的网络流量分类算法的参数主要有3个：子向量数M，子向量聚类数V，检索近邻数k，V和M影响线下学习阶段的编码过程，而k影响实时分类阶段的近邻查找程度，实验的重点为找到使基于哈希的网络流量分类算法性能最好的一组值并进行分析。

### 3.4.3 分类性能

#### 1 分类准确率

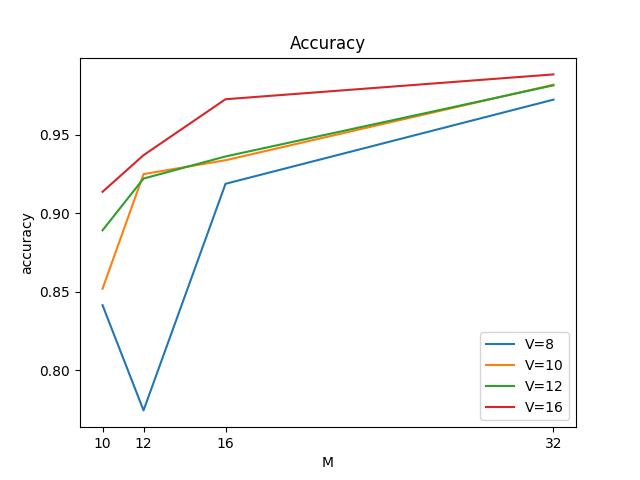
首先，我们选择Moore测试集中的5个数据集，将每个数据集分别作为训练集和测试集验证分类模型并统计平均准确率，表现最佳的一组实验结果如下：

表3.2 5个数据集交叉验证的准确率

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 |
| 1 | 0.98847 | 0.955222 | 0.955624 | 0.934442 | 0.933235 |
| 2 | 0.942305 | 0.990898 | 0.964291 | 0.931102 | 0.895533 |
| 3 | 0.97907 | 0.980378 | 0.990843 | 0.907703 | 0.949273 |
| 4 | 0.973228 | 0.947801 | 0.954831 | 0.989381 | 0.959468 |
| 5 | 0.963972 | 0.951655 | 0.95612 | 0.963664 | 0.988607 |
| 平均 | 0.969409 | 0.965191 | 0.964342 | 0.945258 | 0.945223 |

如上表所示，无论我们将哪个数据集作为训练集，其训练得到的分类模型对所有集合的平均分类准确率均为95%左右，该准确率是较为理想的，同时对于不同的测试集分类模型的准确率较为稳定，说明对于Moore数据集该分类方法具有一定的鲁棒性。另外当训练集数据和测试集数据取自同一个数据集时，分类准确率最高，例如当从第3个数据集中取得训练集，并将剩下的数据作为训练集即进行Holdout验证时，准确率达到了99.1%。

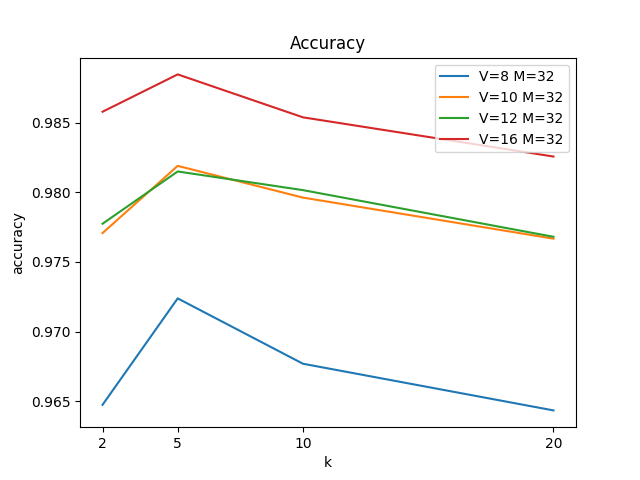
下面我们讨论分类模型的参数对分类准确率的影响，我们使用Moore集中的第1个数据集用于验证和说明。在k=5时，使用占比0.7的训练样本进行训练并进行Holdout验证，子向量聚类数V、子向量数M对准确率的影响如下图：

图图3.1 V、M对分类准确率影响

结果表明参数V和M对分类模型的整体分类准确率影响明显。当M一定时，V值较小则聚类过于粗糙，分类准确率不够理想，当V增大到为网络流量实际类别数左右时，分类模型分类准确率上升到较高水平，之后再继续增大但分类模型性能提升不明显。当V一定时，随着M的增大即划分网络流量数据得到的子向量数量增加，分类模型的分类准确率呈上升趋势。当M值取32，V值取16时准确率达到最高值98.847%。总的来说，当V和M较大时，分类准确率较高，但是此时一条网络流量编码长度也较长。

以上实验结果符合分类模型的原理，当网络流量划分得到的子空间数量较少时，乘积量化编码得到的哈希码数量较少，整体编码长度较短，网络流量数据的高维特征难以映射到如此短的哈希编码中，数据信息被过度压缩。当子空间聚类数较少时，K均值算法分类粗糙，所得到的哈希索引对子向量类别反映不够充分，子向量间发生过度碰撞，不同类别的网络流量数据被归为相同类别，导致分类准确率下降。而当划分的子空间数和每个子空间内聚类数选择合适时，分类模型准确率较高。

以上分析的是线下训练阶段乘积量化和哈希编码过程的参数V和M，现在分析实时分类阶段KNN算法的k的取值对模型分类准确率的影响。

图图3.2 k对分类准确率影响

可以看到k取值对整体分类准确率有一定影响，但影响并不大。在参数V和M确定时，k=5时分类模型的整体分类准确率达到最高值，在KNN算法中k的含义是检索近邻的数量，实验结果表明k的取值并不是越大越好，需要根据数据特点合理选择k值以确保最佳的分类准确率。实际上分类模型的准确率并不能完全反映分类性能的好坏，对于基于哈希编码的网络流量分类模型而言，k值不同，整体分类准确率似乎差别不大，但在下面的召回率和精度的分析中，我们发现k值对分类模型分类性能的影响是明显的。另外考虑到分类准确率表现尚可，我们将V=16、M=32、k=5作为一组理想的参数以便于之后的分析，即之后的实验如未加说明，则该组参数为默认参数。

#### 2 召回率与精度

Moore数据集中网络流量被分为WWW、MAIL等类别，对于某种具体类别的网络流量的分类效果的评价需要通过回归率和精度来进行分析，实验中我们通过混淆矩阵（confusion matrix）得到了不同种类流量的各项评价指标，下表为表现最好的一组实验数据，数据基于对于第一个数据集的Holdout验证获取：

表3.3 V=16、M=32、k=5时网络流量召回率、精度表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| ATTACK | 1 | 0.48 | 0.65 | 33 |
| DATABASE | 0.94 | 0.97 | 0.95 | 74 |
| FTP-CONTROL | 0.91 | 0.89 | 0.9 | 46 |
| FTP-DATA | 0.99 | 0.99 | 0.99 | 378 |
| FTP-PASV | 0.67 | 0.36 | 0.47 | 11 |
| INTERACTIVE | 0 | 0 | 0 | 2 |
| MAIL | 0.98 | 0.99 | 0.99 | 1230 |
| MULTIMEDIA | 0.65 | 0.71 | 0.68 | 24 |
| P2P | 0.92 | 0.8 | 0.86 | 106 |
| SERVICES | 0.96 | 1 | 0.98 | 64 |
| WWW | 0.99 | 1 | 1 | 5491 |
| avg / total | 0.99 | 0.99 | 0.99 | 7459 |

如表所示，对于不同的网络流量，分类模型的预测表现差异较大。对于样本数量较多的网络流量类型例如WWW、MAIL、FTP-DATA和SERVICES类型，分类模型的预测精度和召回率均相当出色。而对于其他类型的网络流量如ATTACK、MULTIMEDIA等类型，预测精度和召回率都存在缺陷。出现这种情况的原因可能的原因是训练样本不平衡，即数据集中WWW等类型的样本过多，导致了训练得到的分类模型不够完善。

值得注意的是，对整体分类准确率影响较小的k的取值在不均衡的测试样本下对精度和召回率影响很大。以下为k=50时，基于哈希编码的网络流量分类模型的精度和召回率结果：

表3.4 V=16、M=32、k=50网络流量召回率、精度表

|  | precision | recall | f1-score | support |
| --- | --- | --- | --- | --- |
| ATTACK | 0.92 | 0.33 | 0.49 | 33 |
| DATABASE | 0.9 | 0.86 | 0.88 | 74 |
| FTP-CONTROL | 0.92 | 0.52 | 0.67 | 46 |

续表3.4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| FTP-DATA | 0.98 | 0.92 | 0.95 | 378 |
| FTP-PASV | 0 | 0 | 0 | 11 |
| INTERACTIVE | 0 | 0 | 0 | 2 |
| MAIL | 0.96 | 0.98 | 0.97 | 1230 |
| MULTIMEDIA | 0.64 | 0.38 | 0.47 | 24 |
| P2P | 0.81 | 0.72 | 0.76 | 106 |
| SERVICES | 0.65 | 1 | 0.79 | 64 |
| WWW | 0.99 | 0.99 | 0.99 | 5491 |
| avg / total | 0.97 | 0.97 | 0.97 | 7459 |

上表中，support表示样本数量，可以看到尽管整体上分类准确率没有明显差距，但是具体到某种类型的网络流量上预测表现却大不相同。对于WWW、MAIL等类型的流量，k=50的分类模型预测表现依旧不错，然而对于FTP-PASV等类型的流量，分类模型的预测性能下降明显。k值增大后，预测性能出现更为严重的不平衡，这种现象出现的原因是KNN分类算法的缺陷，即在样本不平衡的情况下，样本数量大的数据干扰了样本数量小的数据的分类。

以上召回率和精度的分析表明本文实现的基于哈希编码的网络流量分类方法对于特定种类的网络流量分类性能还存在不足，需要在算法设计和具体实现上进行优化与改进，改进的重点应在对不平衡数据样本的处理上。

#### 3 空间性能

哈希编码的优点之一是能够大量节省空间开销，实验中统计了Moore数据集中第1个数据集占用的空间和按照基于哈希的网络流量分类算法将其编码为哈希码后占用的空间。

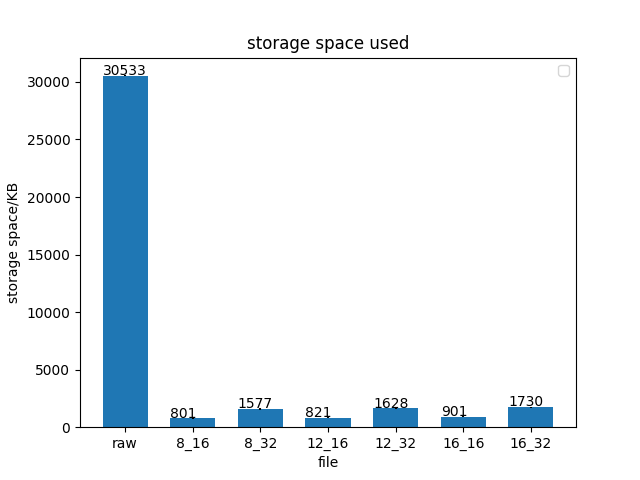


图3.3 存储空间对比

其中横坐标表示数据的处理方式，raw表示原始数据，而如8\_16例则表示按照V=8、M=16进行编码，可以看到对原始的网络流量数据进行哈希编码后，数据所需的存储空间大大减少。在分类准确率最高的情况下即V=16、M=32的情况下，编码后数据占用空间最大，为1730KB，但相比于原始数据仍减少了95%的占用空间。实验结果表明基于哈希编码的网络流量分类方法在空间上的优势是相当明显的，能够大大减少算法运行时的内存空间开销。

#### 4 分类时间

网络流量分类中分类时间是衡量分类模型性能的关键指标，关系着分类器能否达到实时性要求，能否尽快地对网络流量进行处理。对于基于哈希编码的网络流量分类模型的平均分类时间统计如下表：

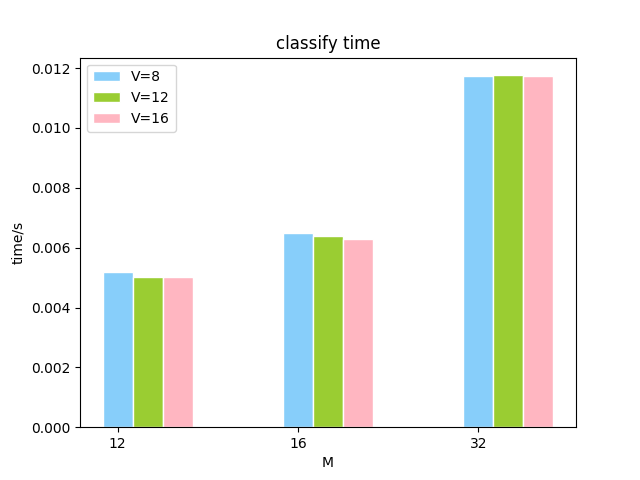


图3.4 分类时间对比

实验结果表明随着M值增大，分类模型对某一网络流量所需的预测时间也在增加。当M增大时，网络流量被分为更多的子向量，而对每个子向量计算汉明距离的时间开销是一定的，更多的子向量意味着计算时间开销更大，分类算法的时间复杂度增加。而当V增大时，例如V=8增大到V=16时，哈希编码从3位增加到4位，但是分类时间并无增加趋势，这一点可能与Python程序中汉明距离的计算方法有关，具体原因需要进一步分析。总的来说，当M=32、V=16、 k=5的情况下，基于哈希编码的网络流量分类模型分类整体准确率达到98.8%，而对某一网络流量进行预测的平均分类时间和最大分类时间不到0.012秒。在V=16、M=12、k=5的情况下分类准确率达到95.1%，而平均分类时间约为0.006秒。

### 3.4.4 和传统KNN算法的对比

本文所实现的基于哈希的网络流量分类方法在实时分类阶段使用的分类算法为KNN算法，而传统的KNN算法衡量数据相似性往往基于欧式距离。因为在线下训练阶段本文所实现的分类方法已将训练数据编为哈希索引，因此本文的KNN算法基于汉明距离来衡量网络流量数据间的相似性。我们利用Moore数据集的前5个数据集进行验证，分别测试两种实现方法的准确率和预测时间，现对两种实现方法的分类性能做简单分析。

基于欧式距离的KNN算法和基于汉明距离的KNN算法分类准确率对比如下：

表3.5 欧式KNN和汉明KNN分类准确率对比

|  |  |  |
| --- | --- | --- |
|  | euclidean | hamming |
| 1 | 0.97479555 | 0.97425928 |
| 2 | 0.98151519 | 0.97829436 |
| 3 | 0.98125 | 0.98008721 |
| 4 | 0.98070595 | 0.97995812 |
| 5 | 0.97921478 | 0.97875289 |
| 平均值 | 0.9794963 | 0.97827037 |

可以看到相比于传统KNN而言，基于哈希的网络流量分类方法在准确率上并无明显差异，这是因为传统KNN分类准确率已经相当高了，平均值达到了97.9%。

而上述两种方法的分类时间对比如下：

表3.6 欧式KNN和汉明KNN分类时间对比

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | euclidean | | hamming | |
|  | 平均值/s | 最大值/s | 平均值/s | 最大值/s |
| 1 | 0.01251202 | 0.02806187 | 0.01186733 | 0.033549 |
| 2 | 0.01189115 | 0.02679706 | 0.0116676 | 0.030694 |
| 3 | 0.01151583 | 0.02956891 | 0.0113546 | 0.029888 |
| 4 | 0.01094815 | 0.03086996 | 0.01130607 | 0.030742 |
| 5 | 0.01081509 | 0.03139305 | 0.01136875 | 0.031155 |
| 平均值 | 0.01153645 | 0.02933817 | 0.01151287 | 0.033549 |

上表的结果令人意外地表明，相比较于传统的基于欧式距离的KNN，基于哈希编码的KNN分类的预测时间与之基本持平，甚至在最大预测时间这一指标上还落后不少。考虑到计算汉明距离要比计算欧式距离时间开销小得多，这一结果实在让人难以满意。

在基于哈希编码的网络流量分类方法中，对一条网络流量预测的过程大体上分为两步，第一步是将该流量的特征数据编码为哈希索引，第二步是在KNN中计算哈希索引的汉明距离。对于两阶段的时间开销列表如下：

表3.7 基于汉明的KNN分类时间分析表

|  |  |  |
| --- | --- | --- |
|  | step1/s | step2/s |
| 1 | 0.00945976 | 0.00240757 |
| 2 | 0.00930502 | 0.00236258 |
| 3 | 0.00919686 | 0.00215774 |
| 4 | 0.00918068 | 0.00212539 |
| 5 | 0.00926835 | 0.0021004 |
| 平均值 | 0.00928213 | 0.00223074 |

表中step1表示编码过程，step2表示KNN计算过程。可见对于基于哈希编码的网络流量分类方法，其分类预测时间约80%的部分来自于编码过程的开销。在刨去编码开销后，对于编码完成后的计算过程与表4.4中数据对比，其时间约为使用原始数据计算欧氏距离即传统方法所需时间的20%，该速度仍低于预期速度。导致使用汉明距离的算法设计在时间性能上未能充分发挥优势的原因可能是在基于Python的实现程序中对于汉明距离的计算开销要高于理论值。

另外关于两种方法空间性能上的对比在上一小节中有所反映，由于传统KNN使用的是原始数据，因此在内存占用上会比基于哈希的KNN要大。在训练数据更大的情况下，该对比更加明显。

以上本文实现的基于哈希的KNN方法与传统KNN的方法的对比均在暴力检索实方式现的KNN下完成，对于利用Kd-Tree或Ball-Tree等数据结构加速的KNN算法下性能对比的工作限于水平本文未进行。

### 3.4.5 实验结果分析

通过实验验证，本文所实现的基于哈希编码的网络流量分类方法在分类准确率上、空间性能上表现尚可，同时具有一定鲁棒性，然而对于某些种类的网络流量预测性能存在缺陷，在时间性能上未能达到预期效果。对该方法的改进可以从以下两方面着手：（一）数据集样本不平衡问题（二）Python程序中汉明距离计算开销。

# 结 论

本文对现有的网络流量分类方法和基于哈希的大数据近邻检索进行了研究，在现有理论和实践的基础上将两者进行了结合，实现了一种基于哈希编码的网络流量分类方法，通过实验验证了该分类方法的性能。

基于哈希编码的网络流量分类方法重点解决网络流量分类中传统的基于端口匹配和基于深度包检测方法对于新型网络流量分类上准确率较低的问题，同时为基于统计特征学习的分类方法提供了一种新的实现思路。本文提出的方法基本思路是将网络流量数据映射为若干个哈希索引值，并利用哈希索引间的汉明距离作为衡量网络流量间相似度的标准进行分类。具体的实现方式分为线下训练和线上分类两个过程，线下训练过程通过乘积量化和K均值算法进行哈希索引编码，线上分类过程通过KNN算法实现分类。

将基于哈希编码的大数据近邻检索技术应用到网络流量分类主要是出于以下目的：（一）与欧式距离的计算相比，汉明距离的计算通过异或运算进行，在计算速度上具有优势。（二）网络流量数据的统计特征数量达到200多个，在不进行特征选择的情况下网络流量分类陷入“维度灾难”，哈希编码的思想能够大大降低网络流量数据的维度，同时大大减少数据所占的存储空间和内存开销。正因为如此，本文在实验中对以上两点进行了测试与验证。

通过设置算法参数变量，本文对实现的基于哈希编码的网络流量分类模型的分类性能进行了验证。实验结果表明，该分类模型对网络流量的最好整体分类准确率达到了超过98%，对于WWW、MAIL、SERVICES等类型的网络流量分类效果非常出色，同时在空间开销上降低了95%的数据存储空间，基于哈希编码的网络流量分类模型整体上达到了设计要求。

然而，本文所做的工作还存在许多不足和需要完善之处，最明显的不足是在样本不平衡的情况下对部分类型的网络流量分类性能不够理想，分类精度和召回率都较低。对于分类的时间开销，和传统KNN算法相比没有体现出明显优势。此外，由于本文所使用的Moore数据集所采集的网络流量数据为基于TCP协议的网络流量，因此本文对基于UDP协议的网络流量分类效果有待进一步验证。

实际上，网络流量分类的研究仍有巨大的改进空间，网络流量分类方法往往在某特定网络环境中采集的某个特定的数据集上进行研究，缺少时空普适性，分类结果并不能完全反映在其他网络或其他数据集中的分类性能。随着网络技术的发展，网络规模不断增长，新型网络协议和应用层出不穷，可扩展性、自适性和可靠性良好的网络分类方法和模型依然难得。因为本文作者水平有限，本文提出的方法对于许多局限未能突破，更多的问题还留待后来的研究者解决。

# 参考文献

# 致 谢

感谢我的指导老师齐恒副教授在毕业设计的选题、实施和论文撰写各个阶段对我的细心指导。在老师的无私帮助下，我对科学研究有了一定的了解，我今后的学习和工作将从老师所要求的严谨认真的治学态度中获益匪浅。

1. [] 第41次中国互联网络发展状况统计报告[R].北京，中国互联网络信息中心，2018 [↑](#endnote-ref-2)
2. [] AW Moore, D Zuev. Discrimination for use in flow-based classification[R]. Intel Research, Cambridge. 2005. [↑](#endnote-ref-3)
3. [] <https://www.iana.org/assignments/service-names-port-numbers/service-names->port-numbers.xml [↑](#endnote-ref-4)
4. [] Li W, Canini M, Moore A W, et al. Efficient application identification and the temporal and spatial stability of classification schema[J]. Computer Networks, 2009, 53(6):790-809. [↑](#endnote-ref-5)
5. [] Zuev D, Moore A W. Traffic Classification Using a Statistical Approach[C]// International Conference on Passive and Active Network Measurement. Springer-Verlag, 2005:321-324. [↑](#endnote-ref-6)
6. [] 付文亮, 嵩天, 周舟. RocketTC:一个基于FPGA的高性能网络流量分类架构[J]. 计算机学报, 2014, 37(2):414-422. [↑](#endnote-ref-7)
7. [] 陈传通. 基于正则表达式匹配的网络流量识别系统的研究与实现[D]. 山东大学, 2013. [↑](#endnote-ref-8)
8. [] Caballero J, Yin H, Liang Z, et al. Polyglot:automatic extraction of protocol message format using dynamic binary analysis[C]// ACM Conference on Computer and and Communications Security. 2007:317-329. [↑](#endnote-ref-9)
9. [] Hu Y, Chiu D M, Lui J C S. Profiling and identification of P2P traffic[M]. Elsevier North-Holland, Inc. 2009. [↑](#endnote-ref-10)
10. [] Bonfiglio D, Mellia M, Meo M, et al. Revealing skype traffic:when randomness plays with you[C]// Conference on Applications. ACM, 2007:37-48. [↑](#endnote-ref-11)
11. [] 杨季, 石亮山, 陈波,等. 基于子图模式的网络流量分类方法研究[J]. 计算机应用研究, 2014, 31(6):1816-1819. [↑](#endnote-ref-12)
12. [] Bacquet C. AN INVESTIGATION OF A MULTI-OBJECTIVE GENETIC ALGORITHM APPLIED TO ENCRYPTED TRAFFIC IDENTIFICATION[J]. Network Traffic Identification, 2010. [↑](#endnote-ref-13)
13. [] 于孝关, 陈贞翔, 彭立志. 基于决策树的网络流量分类方法[J]. 济南大学学报(自然科学版), 2012, 26(3):291-295. [↑](#endnote-ref-14)
14. [] Alshammari R, Zincir-Heywood A N. Identification of VoIP encrypted traffic using a machine learning approach[J]. Journal of King Saud University - Computer and Information Sciences, 2015, 27(1):77-92. [↑](#endnote-ref-15)
15. [] Pan W, Cheng G, Tang Y. WENC: HTTPS Encrypted Traffic Classification Using Weighted Ensemble Learning and Markov Chain[C]// Trustcom/bigdatase/icess. IEEE, 2017:50-57. [↑](#endnote-ref-16)
16. [] Jin Y, Duffield N, Erman J, et al. A Modular Machine Learning System for Flow-Level Traffic Classification in Large Networks[J]. Acm Transactions on Knowledge Discovery from Data, 2012, 6(1):1-34. [↑](#endnote-ref-17)
17. [] Erman J, Mahanti A, Arlitt M, et al. Identifying and discriminating between web and peer-to-peer traffic in the network core[C]// International Conference on World Wide Web. ACM, 2007:883-892. [↑](#endnote-ref-18)
18. [] Li Z, Yuan R, Guan X. Accurate Classification of the Internet Traffic Based on the SVM Method[C]// IEEE International Conference on Communications. IEEE, 2007:1373-1378. [↑](#endnote-ref-19)
19. [] 邓河, 阳爱民, 刘永定. 一种基于SVM的P2P网络流量分类方法[J]. 计算机工程与应用, 2008, 44(14):122-126. [↑](#endnote-ref-20)
20. [] 周文刚, 陈雷霆, Lubomir,等. 基于半监督的网络流量分类识别算法[J]. 电子测量与仪器学报, 2014, 28(4):381-386. [↑](#endnote-ref-21)
21. [] Bentley J L. Multidimensional Binary Search Trees Used for Associative Searching.” Communications of the ACM, 18(9), 509-517[J]. 1975, 18(9):509-517. [↑](#endnote-ref-22)
22. [] Hubo E, Mertens T, Haber T, et al. The Quantized kd-Tree: Efficient Ray Tracing of Compressed Point Clouds[C]// Interactive Ray Tracing 2006, IEEE Symposium on. IEEE, 2006:105-113. [↑](#endnote-ref-23)
23. [] Procopiuc O, Agarwal P K, Arge L, et al. Bkd-Tree: A Dynamic Scalable kd-Tree[J]. Lecture Notes in Computer Science, 2003, 2750:46-65. [↑](#endnote-ref-24)
24. [] Bernardina P D. PCA-tree: uma proposta para indexação multidimensional[J]. Biblioteca Digital De Teses E Dissertações Da Usp, 2007. [↑](#endnote-ref-25)
25. [] Dasgupta S, Freund Y. Random projection trees for vector quantization[J]. IEEE Transactions on Information Theory, 2009, 55(7):3229-3242. [↑](#endnote-ref-26)
26. [] <http://image.baidu.com/?fr=shitu> [↑](#endnote-ref-27)
27. [] Piotr Indyk, Rajeev Motwani. Approximate Nearest Neighbor: Towards Removing the Curse of Dimensionality[J]. Theory of Computing, 2012, 604-613(11):604-613. [↑](#endnote-ref-28)
28. [] Gionis, Aristides, Indyk, Piotr, Motwani, Rajeev. Similarity Search in High Dimensions via Hashing[J]. 1999, 8(2):518--529. [↑](#endnote-ref-29)
29. [] Charikar M S. Similarity estimation techniques from rounding algorithms[J]. 2002:380-388. [↑](#endnote-ref-30)
30. [] Datar M, Immorlica N, Indyk P, et al. Locality-sensitive hashing scheme based on p-stable distributions[C]// Twentieth Symposium on Computational Geometry. ACM, 2004:253-262. [↑](#endnote-ref-31)
31. [] Chum O, Philbin J, Isard M, et al. Scalable near identical image and shot detection[C]// Proceedings of the 6th ACM international conference on Image and video retrieval. ACM, 2007:549-556. [↑](#endnote-ref-32)
32. [] Kulis B, Grauman K. Kernelized locality-sensitive hashing[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2012, 34(6):1092. [↑](#endnote-ref-33)
33. [] He J, Liu W, Chang S F. Scalable similarity search with optimized kernel hashing[C]// ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, Dc, Usa, July. DBLP, 2010:1129-1138. [↑](#endnote-ref-34)
34. [] 林朝晖, 于俊清, 何云峰,等. 高维分布式局部敏感哈希索引方法[J]. 计算机科学与探索, 2013, 7(9):811-818. [↑](#endnote-ref-35)
35. [] 林悦. 基于哈希算法的高维数据的最近邻检索[D]. 浙江大学, 2013. [↑](#endnote-ref-36)
36. [] Matsushita Y, Wada T. Principal Component Hashing: An Accelerated Approximate Nearest Neighbor Search[C]// Pacific Rim Symposium on Advances in Image and Video Technology. Springer-Verlag, 2009:374-385. [↑](#endnote-ref-37)
37. [] Weiss Y, Torralba A, Fergus R. Spectral hashing[C]// International Conference on Neural Information Processing Systems. Curran Associates Inc. 2008:1753-1760. [↑](#endnote-ref-38)
38. [] Shao J, Wu F, Ouyang C, et al. Sparse spectral hashing[J]. Pattern Recognition Letters, 2012, 33(3):271-277. [↑](#endnote-ref-39)
39. [] Liu Q, Liu G, Li L, et al. Reversed Spectral Hashing.[J]. IEEE Transactions on Neural Networks & Learning Systems, 2017, PP(99):1-9. [↑](#endnote-ref-40)
40. [] Gong Y, Lazebnik S, Gordo A, et al. Iterative quantization: a Procrustean approach to learning binary codes for large-scale image retrieval.[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2013, 35(12):2916-2929. [↑](#endnote-ref-41)
41. [] Amsaleg L. Locality sensitive hashing: A comparison of hash function types and querying mechanisms[J]. Pattern Recognition Letters, 2010, 31(11):1348-1358. [↑](#endnote-ref-42)
42. [] Liu W，Wang J，Kumar S，et a1．Hashing with graphs[C]／／International Conference on Machine Learning．20 1 1． [↑](#endnote-ref-43)
43. [] Heo J P, Lee Y, He J, et al. Spherical hashing[C]// Computer Vision and Pattern Recognition. IEEE, 2012:2957-2964. [↑](#endnote-ref-44)
44. [] Ge T, He K, Ke Q, et al. Optimized Product Quantization[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2014, 36(4):744-55. [↑](#endnote-ref-45)
45. [] Kalantidis Y, Avrithis Y. Locally Optimized Product Quantization for Approximate Nearest Neighbor Search[C]// IEEE Conference on Computer Vision and Pattern Recognition. IEEE Computer Society, 2014:2329-2336. [↑](#endnote-ref-46)
46. [] Li L, Hu Q, Han Y, et al. Distribution Sensitive Product Quantization[J]. IEEE Transactions on Circuits & Systems for Video Technology, 2017, PP(99):1-1. [↑](#endnote-ref-47)
47. [] <http://www.cl.cam.ac.uk/research/srg/netos/projects/archive/nprobe/data/papers/sigmetrics/> index.html [↑](#endnote-ref-48)
48. [] Moore A W, Zuev D. Internet traffic classification using bayesian analysis techniques[J]. ACM SIGMETRICS Performance Evaluation Review, 2005, 33(1):50-60. [↑](#endnote-ref-49)
49. [] Zuev D, Moore A W. Traffic Classification Using a Statistical Approach[C]// International Conference on Passive and Active Network Measurement. Springer-Verlag, 2005:321-324. [↑](#endnote-ref-50)
50. [] Andrew Moore, James Hall, Christian Kreibich, Euan Harris, and Ian Pratt. Architecture of a Network Monitor[J].Passive & Active Measurement Workshop 2003 (PAM2003), April 2003. [↑](#endnote-ref-51)
51. [] https://www.cs.waikato.ac.nz/ml/weka/ [↑](#endnote-ref-52)