清 华 大 学

综 合 论 文 训 练

题目：基于 Spark 的分布式数据量化与索引系统

系　　别：软件学院

专　　业：计算机软件

姓　　名：文庆福

指导教师：王建民教授

2015 年 6 月 10 日

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：学校有权保留学位论文的复印件，允许该论文被查阅和借阅；学校可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

(涉密的学位论文在解密后应遵守此规定)

签　名：_____ 导师签名：_____ 日　期：_____

中文摘要

高维数据的近邻查询是处理多媒体数据的一项重要的技术，在计算机视觉、数据挖掘等前沿研究领域有着丰富的应用。随着数据指数式的增长，如何从海量、高维数据中进行尽可能快速的近邻查询以及大规模的数据如何存储索引，一直以来都是备受研究者关注的问题。

论文的摘要是对论文研究内容和成果的高度概括。摘要应对论文所研究的问题及其研究目的进行描述，对研究方法和过程进行简单介绍，对研究成果和所得结论进行概括。摘要应具有独立性和自明性，其内容应包含与论文全文同等量的主要信息。使读者即使不阅读全文，通过摘要就能了解论文的总体内容和主要成果。

关键词：近邻查询；Spark；向量量化；倒排索引

ABSTRACT

An abstract of a dissertation is a summary and extraction of research work and contributions. Included in an abstract should be description of research topic and research objective, brief introduction to methodology and research process, and summarization of conclusion and contributions of the research. An abstract should be characterized by independence and clarity and carry identical information with the dissertation. It should be such that the general idea and major contributions of the dissertation are conveyed without reading the dissertation.

An abstract should be concise and to the point. It is a misunderstanding to make an abstract an outline of the dissertation and words "the first chapter", "the second chapter" and the like should be avoided in the abstract.

Key words are terms used in a dissertation for indexing, reflecting core information of the dissertation. An abstract may contain a maximum of 5 key words, with semicolons used in between to separate one another.

Keywords: TEX; LATEX; CJK; template; thesis

目 录

主要符号对照表

HPC	高性能计算 (High Performance Computing)
cluster	集群
Itanium	安腾
SMP	对称多处理
API	应用程序编程接口
PI	聚酰亚胺
MPI	聚酰亚胺模型化合物，N-苯基邻苯酰亚胺
PBI	聚苯并咪唑
MPBI	聚苯并咪唑模型化合物，N-苯基苯并咪唑
PY	聚吡咙
PMDA-BDA	均苯四酸二酐与联苯四胺合成的聚吡咙薄膜
ΔG	活化自由能 (Activation Free Energy)
χ	传输系数 (Transmission Coefficient)
E	能量
m	质量
c	光速
P	概率
T	时间
v	速度
劝学	君子曰：学不可以已。青，取之于蓝，而青于蓝；冰，水为之，而寒于水。木直中绳。（车柔）以为轮，其曲中规。虽有槁暴，不复挺者，（车柔）使之然也。故木受绳则直，金就砺则利，君子博学而日参省乎己，则知明而行无过矣。吾尝终日而思矣，不如须臾之所学也；吾尝（足齐）而望矣，不如登高之博见也。登高而招，臂非加长也，而见者远；顺风而呼，声非加疾也，而闻者彰。假舆马者，非利足也，而致千里；假舟楫者，非能水也，而绝江河，君子生非异也，善假于物也。积土成山，风雨兴焉；积水成渊，蛟龙生焉；积善成德，而神明自得，圣心备焉。故不

积跬步，无以至千里；不积小流，无以成江海。骐骥一跃，不能十步；驽马十驾，功在不舍。锲而舍之，朽木不折；锲而不舍，金石可镂。蚓无爪牙之利，筋骨之强，上食埃土，下饮黄泉，用心一也。蟹六跪而二螯，非蛇鳝之穴无可寄托者，用心躁也。—— 荀况

第 1 章　引言

1.1　研究背景

在今天这个数据爆炸的时代，文本、图像、视频以及音频等多媒体数据呈现出指数级的增长。如何快速、准确地从这个海量的互联网数据库中获取我们想要的信息，是我们不得不面对的一个问题。Google[①]、Bing[②]、Baidu[③] 等提供的文本、图像等搜索引擎服务为我们获取信息带来了极大的便利。而在这些搜索引擎背后的都需要用到的一项技术——**近似近邻查询**（Approximate Nearest Neighbor Search）。在大规模数据的应用场景下，精确的近似查询需要耗费时间太长，不具有实际应用价值。近似近邻查询可以大幅度缩短查询时间，同时保证查询结果与精确查询结果近似，因此更具有实用性。除了信息检索以外，近似近邻查询技术被广泛应用于计算机视觉、机器学习、数据挖掘、模式识别等领域。

在不考虑时间效率的情况下，近似近邻查询问题可以直接通过一种暴力搜索的方式来解决。比如，我们可以直接计算查询数据 q 与数据集合 S 中每一条数据的距离，最终根据距离大小，选取出距离最近的前 n 个数据。但在现实中，由于数据集合 S 的规模非常，这种朴素的方法单词查询的计算时间太长而无法采用。但是，如果从规模比较大的数据集合 S 上删选出一个非常小待选集合 S'，之后再在集合 S' 上进行朴素的暴力搜索选取出前 n 个近邻数据，这时的暴力搜索的时间效率是可以接受的，整个查询过程的时间效率和准确率就取决于删除出待选集合 S' 的过程。待选集合 S' 大小与查询准确率有着密切关系，一般来说，集合 S' 越大，查询准确率越高，但是最终的暴力搜索阶段的时间会变长；集合 S' 越小，则查询效率越低，暴力搜索阶段的时间越短。因此，如何选择一个大小合适、相关性高的待选集合 S' 就成了近似近邻查询问题的关键。

近年来，近似近邻查询问题一直都是研究热点问题之一。目前，这一技术

① https://www.google.com
② http://cn.bing.com
③ https://www.baidu.com

主要面临一下两大挑战：

1. 海量

   随着互联网上的数据越来越多，需要存储的数据量也越来越大。然而，传统的索引结构一般都是基于小规模数据而设计的单机结构。大规模的数据一般无法做到单机存储，更不用说加载到内存当中索引了。这些数据往往存储于分布式系统当中，同时也需要一种分布式的索引结构来支持查询。海量的数据不仅给存储带来了压力，同时也给实时查询带来了巨大挑战。

2. 维度灾难

   在多媒体数据处理过程时，往往都会针对多媒体数据提取特征进行处理。为了更好地刻画数据，一般来说，特征数据的维度越高，刻画的准确性也越高。例如，在图像数据处理过程中，我们常常可能用到的 SIFT、SURF 特征都是 128 维的，GIST 特征有 960 维，而 BOVW（Bag of Visual Words）的维度更是高达成千上万维。如此高维度的数据，仅计算两个向量之间距离的时间消耗就比较长，更不必说在大规模数据上进行查询了。

Spark 是一个基于 MapReduce 的通用的大数据并行计算框架，最初由 UC Berkeley AMP Lab 开发。Spark 的架构是在 Hadoop 基础上的改良，继承了 MapReduce 的优点，它与 Hadoop 最大不同之处就是内存计算，Hadoop 将计算过程的中间数据存储在磁盘上，而 Spark 一般是用内存来存储数据，所有数据操作都在内存中完成。

1.2 主要研究内容

本文的主要的关注点是在高维空间中的近似近邻查询问题，通过研究对比现有的几种最新近似近邻查询方法，了解几种不同方法的优缺点。在 Spark 框架下实现了一种基于向量量化（Vector Quantization）的近似近邻查询方法。最终，通过实验来验证算法的准确性以及测试算法时间效率。

1.3 论文组织结构

第 2 章　索引结构综述

前文已经提到，近邻查询问题最关键的就是如何选取出一个待选集合 S'。为了高效地删选出待选集合，我们通常会先把原始的数据索引起来。依据索引结构的不同，近邻查询的方法可以分为基于树结构的索引和基于哈希的索引。

2.1　基于树结构的索引

2.1.1　KD-树

2.1.2　K-Means 树

2.2　基于哈希的索引

基于哈希的索引方法是将高维的数据压缩成二进制编码的形式来进行近似近邻查询，这类方法在图像、文本、视频等多媒体检索上取得不错的效果。根据哈希函数形式的不同，我们可以简单地将哈希索引分为两类——汉明嵌入和向量量化。

2.2.1　汉明嵌入

汉明嵌入就是要寻找一个映射，对于任意的对象 $x \in S$ 都能映射到二进制串 $b(x) \in 0,1^d$。那么，任意两个对象之间的相似度就可以通过其对应二进制串之间的汉明距离来近似计算：

$$sim(x,y) \approx 1 - \frac{2\delta_{Ham}(b(x),b(y))}{d} \tag{2-1}$$

2.2.2　向量量化

基于向量量化的哈希索引是对原始向量空间进行量化压缩，原始向量 $\mathbf{x} \in \mathbb{R}^D$ 通过量化函数 q 被映射到 $q(\mathbf{x}) \in C = \{\mathbf{c}_i\}$，其中 i 是下标，$\mathbf{c}_i$ 可以称作为码字，而 C 则被称为码本。这种映射可以形式化定义成：$\forall \mathbf{x} \in \mathbb{R}^D$，$\exists \mathbf{c}_i \in C$，$q(\mathbf{x}) = \mathbf{c}_i$。整个量化过程中的量化误差 E 就可以定义为：

$$E = \frac{1}{n}\sum_{\mathbf{x}} \|\mathbf{x} - q(\mathbf{x})\|^2 \tag{2-2}$$

其中 $\|\cdot\|$ 表示欧氏距离，而 n 是数据的总量。对于给定的原始数据集合 S，我们的目标是找到一个码本 C 以及对应量化函数 $q(\mathbf{x})$ 使得量化误差 E 最小。在最小化量化误差过程中，不同的限制条件就对应了不同的量化方法。

2.2.3 K-Means

假设有一个包含 n 个 p 维数据点的集合，$\mathcal{D}\{\mathbf{x}_j\}_{j=1}^{n}$，k-means 算法会将这 n 个数据点聚成 k 类，同时用聚类中心来代表每一个聚类的数据。假如矩阵 $C \in \mathbb{R}^{p\times k}$ 的列向量由 k 个聚类构成，每一列都是一个聚类中心，$C = [\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_k]$。k-means 量化的目标函数如下：

$$l_{\text{k-means}} = \sum_{\mathbf{x}\in\mathcal{D}} \min_i \|\mathbf{x} - \mathbf{c}_i\|_2^2 \tag{2-3}$$

$$= \sum_{\mathbf{x}\in\mathcal{D}} \min_{\mathbf{b}\in\mathcal{H}_{1/k}} \|\mathbf{x} - C\mathbf{b}\|_2^2 \tag{2-4}$$

其中 $\mathcal{H}_{1/k} = \{\mathbf{b}|\mathbf{b} \in \{0,1\}^k \text{ 且 } \|\mathbf{b}\| = 1\}$，$\mathbf{b}$ 是一个 1 对 k 编码的二进制向量（包含 1 个 1，k-1 个 0）。

k-means 量化模型浅显易懂，使用最朴素的近邻查询方法就可以将数据点映射到聚类中心。这种映射过程将每一个原始数据压缩到了 $\log_2 k$ 比特的数据，所需要消耗的存储空间随着 k 的线性增长。

2.2.4 ITQ

ITQ（Iterative Quantization）[1] 的量化方法是在 2011 年的 CVPR 会议上提出，这一方法较之前的哈希方法在准确率上有了显著提升。

ITQ 方法首先是对原始数据集空间进行 PCA 降维，将原始的 $\mathbb{R}^{n\times d}$ 空间降维成 $\mathbb{R}^{n\times c}$ 空间。此时，再考虑将降维后的数据集进行二进制编码。从降维后的空间中取出 $\mathbf{v} \in \mathbb{R}^c$，其对应的编码 $\text{sgn}(\mathbf{v})$ 可以看做事超立方体 $\{-1,1\}^c$ 中的顶点。此时，欲使整体的量化误差最小，就是要使原始数据与编码后的数据的欧氏距离最小，$\min \|\text{sgn}(\mathbf{v}) - \mathbf{v}\|^2$。对于编码前的原始数据集我们用 $X \in \mathbb{R}^{n\times d}$ 来表示，经过 PCA 降维后用 $V \in \mathbb{R}^{n\times c}$ 表示，编码后的数据集用 $B \in \{-1,1\}^{n\times c}$ 来表示。整体量化误差：

$$Q(B,R) = \|B - VR\|_F^2 \tag{2-5}$$

其中 $\|\cdot\|_F$ 是 Frobenius 范式，而 R 是正交矩阵，作用是对 V 进行旋转对齐。为何要引入这个旋转矩阵 R 呢？从下图（2.1）可以看出，左侧是 PCA 降维后的数据集 V，中间是随机旋转后的数据集，右侧是最优化旋转后的数据集 VR。右乘一个旋转矩阵 R，让待编码的数据绕数据中心进行旋转到合适状态，每个数据点就可以用距离其最近的数据顶点来表示。下图中可以用 $(-1,-1),(-1,1),(1,1),(1,-1)$ 这是四个点来表示。此时，所有的数据点到超立方的顶点的距离和最小，也就是量化误差最小。

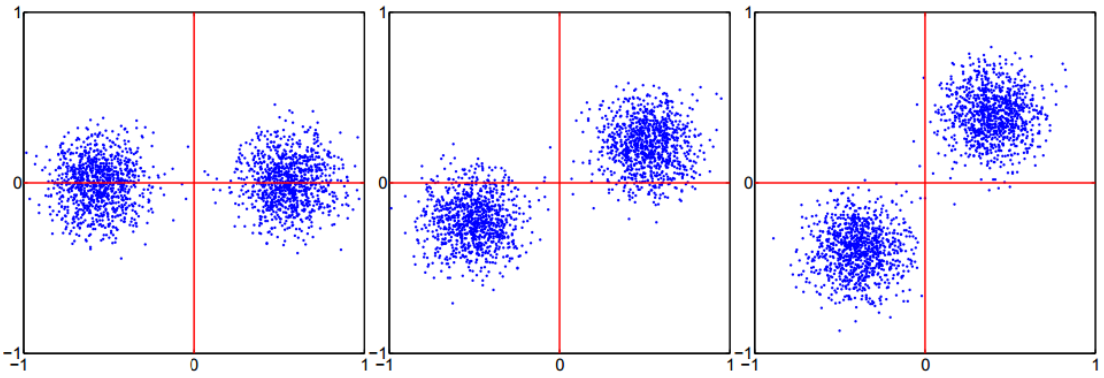

图 2.1　ITQ 旋转过程，图像来源[1]

这样，整个问题的目标函数就变成了 $\min\|B-VR\|_F^2$。这个公式中有两个未知量，编码后的矩阵 B 和旋转矩阵 R。所以，这个问题的求解要考虑采用交替迭代的方法。先对一个随机生成的矩阵进行 SVD 分解得到一个正交矩阵作为 R 的初始值，此时 R 已知，就可以用 $B=\mathrm{sgn}(VR)$ 来求解 B；当 B 已知后，可以对 B^TV 进行 SVD 分解求解 R。既然 R 求出，又可以重新一次迭代，固定 R 求 B，如此交替迭代就可以求解出该问题。

2.2.5　Cartesian K-Means

第 3 章　Spark 弹性分布式数据集与 MLlib 综述

3.1　弹性分布式数据集

弹性分布式数据集（Resilient Distributed Datasets，下文简称 RDD）[2] 是 Spark 中的分布式内存的抽象。相比于 Hadoop 中的计算过程，RDD 可以被缓存在内存当中，每一次的计算产生的结果都可以保留在内存当中。对于迭代计算，这样多次迭代的计算每次可以将结果保存在内存中，下一次迭代又可以直接从内存中读取数据计算，从而避免了大量的磁盘读写操作，大大节省了计算时间。

一般来说，RDD 的创建是通过 SparkContext 来实现，主要包含有两种创建来源：一是从支持的文件系统（或支持的数据库）读取创建；二是从内存数据集合生成。不同于 Hadoop 中仅有 Map 和 Reduce 操作，RDD 还支持其他类型的操作，主要分为转换操作、控制操作和行为操作三类。转换操作顾名思义，就是将一个 RDD 操作之后转换为另一个 RDD，包括 map、flatMap、filter 等操作。控制操作主要用来将 RDD 缓存到内存中或者磁盘上，比如 cache、persist、checkpoint 等操作。行为操作主要分为两类：一类是变成集合或标量的操作，另一类是将 RDD 外部文件系统或数据库的操作。Spark 中的所有对 RDD 的操作，只有当执行行为操作时，才会执行之前的转换或控制操作。例如，我们先对 RDD 执行 map 操作，然后执行 reduce 操作，在 map 操作时，Spark 并不会真正执行，只是记录，只有执行 reduce 操作时才会真正一起计算。这一特性称为惰性计算（lazy computing）。图（3.1）展示的 RDD 的一个操作流程。

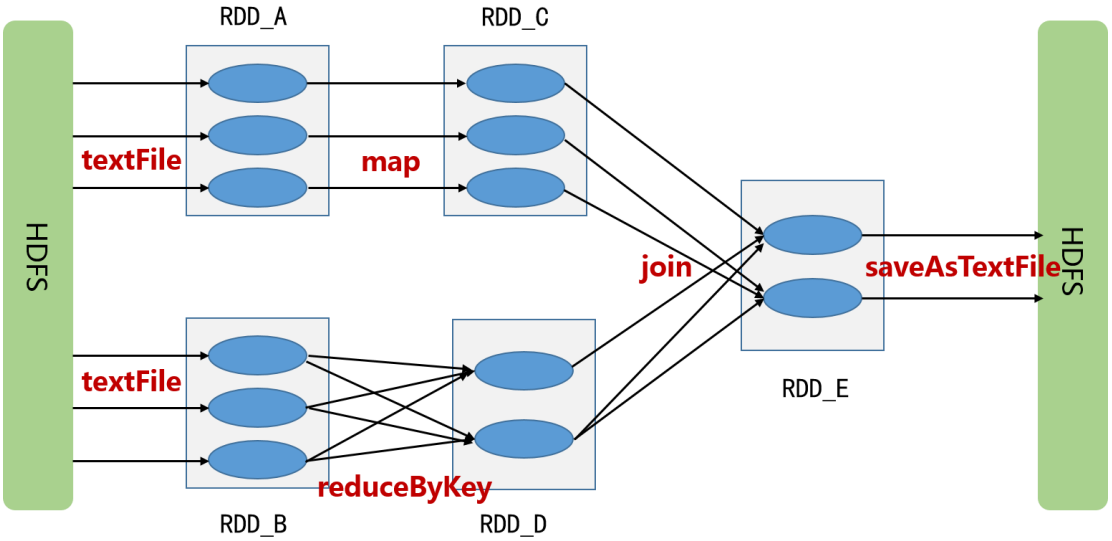

图 3.1　RDD 操作流程示例

RDD 的持久化是 Spark 中 RDD 的一个重要特性。通过 RDD 的持久化，我们可以将 RDD 缓存到内存或者磁盘上。默认地，RDD 并不会进行缓存，每次计算需要用到 RDD 时，都需要重新计算获取 RDD，这样是非常低效的，特别是对于一些迭代的计算。因此，我们需要考虑在计算过程中，将一些重复用到的 RDD 进行缓存操作，这样我们我们只要第一次计算出 RDD，以后每次用到相同的 RDD 时就不用在重复计算，从缓存中直接读取就可以了。RDD 的 cache() 和 persist() 函数就是用于缓存的，其中 cache() 函数是指将 RDD 缓存在内存当中，而 persist() 根据参数可以将 RDD 进行不同级别的缓存。RDD 本身自带有容错机制，通过记录 RDD 的演变过程以便在任务执行失败的时候能够恢复出原有的数据，而不需要通过备份的形式实现容错。例如，当前的 RDD 的因任务执行失败而数据丢失，系统则会根据记录的 RDD 演变关系回溯到未丢失的祖先 RDD，重新根据转换操作计算出一个新的 RDD 进行恢复。

3.2　MLlib

随着数据量的不断增大，传统的单机式的机器学习算法实现已经无法满足大数据的要求，分布式机器学习算法是解决大数据机器学习的提供了可能。MLlib 是基于 Spark 构建的一个常用分布式机器学习算法和工具库，目前支持的算法包括分类算法、回归算法、聚类算法、协同过滤算法、降维算法等。MLlib

中的分布式机器学习算法，相比于以往的算法，在时间效率有了明显提升。下图（3.2）是逻辑回归算法在 Hadoop 和 Spark 上的执行时间对比。

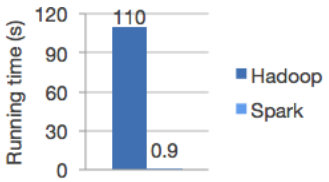

图 3.2　Hadoop 和 Spark 上逻辑回归算法效率对比，图像来源①

MLlib 中已经实现的一些常用机器学习算法可以供使用者调用，在 Spark 实现分布式地实现大规模数据的机器学习的任务正是一个不错的选择。

① http://spark.apache.org/

第 4 章　基于 Spark 的分布式近似近邻查询系统

4.1　乘积量化

与之前提到的 K-Means 的量化方法，乘积量化（Product Quantization）[3] 也是向量量化的一种。假设我们需要量化压缩 128 维的向量到 64 比特，采用 K-Means 的量化方法的话，需要有 2^{64} 个聚类中心，这样不管是从 K-Means 聚类所需要的时间还是从存储聚类中心所占的空间来看，都是不可行的。

对于上面同样的问题，在乘积量化的算法中，我们首先将原始的数据空间划分为 m 个不相交的子空间，也就是将 128 维的向量切成 m 个长度为 $\frac{128}{m}$ 的子向量。在每个子空间里，分别对子空间中的子向量集合进行 K-Means 聚类，聚类中心数量为 h。这样我们就可以用 $1\cdots h$ 这些编号来对子向量进行编码，m 个子向量的编码组合在一起就构成了原始向量的编码。这样，原始空间中一个 128 维的向量就可以压缩到一个 $m\log_2 h$ 比特编码表示，从而大大节省了空间。

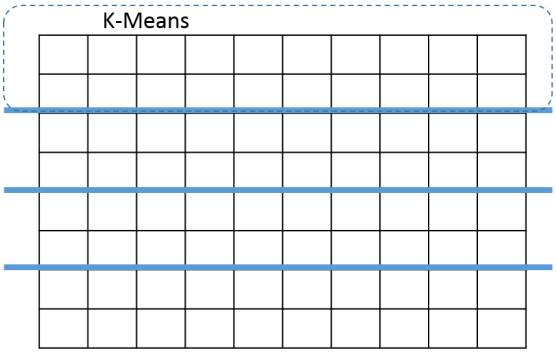

图 4.1　PQ 算法中子空间的划分

更为一般来说，我们可以得到乘积量化方法的目标函数如下：

$$l_{\mathrm{PQ}} = \sum_{i=1}^{n} \min \left\| \mathbf{x}_i - \begin{bmatrix} C^1\mathbf{b}_i^1 \\ \vdots \\ C^m\mathbf{b}_i^m \end{bmatrix} \right\|_2^2 \tag{4-1}$$

其中 $\mathbf{b}_i^j \in \{0,1\}^h$ 且 $\|\mathbf{b}_i^j\| = 1$，$j \in \{1,\cdots,m\}$。在上面式子中，$C^j(j \in 1,\cdots,m)$ 就

是我们需要求解的矩阵，在编码过程中，我们称其作为码本（codebook）。求解过程其实比较简单，正如前文提到，在每个子空间中做 K-Means 聚类就可以求解出码本，这样我们就可以利用码本对每个子空间中的子向量进行编码，从而对原始向量进行编码表示。

4.2 利用乘积量化的近似近邻查询

前文所提的乘积量化只是介绍了如何进行数据的压缩编码，那么乘积量化到底如何应用到近似近邻查询当中呢？

4.3 Spark 上的分布式实现

4.3.1 RDD 的划分

4.3.2 训练阶段

4.3.3 编码阶段

4.3.4 查询阶段

第 5 章　实验与分析

5.1　实验环境

5.2　实验数据集

5.3　实验结果与分析

第 6 章　总结

6.1　本文工作总结

6.2　未来工作展望

插图索引

表格索引

公式索引

参考文献

[1] Gong Y, Lazebnik S. Iterative quantization: A procrustean approach to learning binary codes. Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition, Washington, DC, USA: IEEE Computer Society, 2011. 817–824

[2] Zaharia M, Chowdhury M, Das T, et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12), San Jose, CA: USENIX, 2012. 15–28

[3] Jégou H, Douze M, Schmid C. Product quantization for nearest neighbor search. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2011, 33(1):117–128

致　谢

感谢我的导师王建民教授，王老师严谨治学、勤恳工作的态度深深影响了我。

感谢龙明盛学长对我毕设的耐心指导。同时，也要感谢研究组里的各位学长和学姐，没有他们的答疑解惑，我的毕设工作不可能顺利完成。

感谢大学四年言传身教的各位老师，他们不仅教给我文化知识，而且也教会我为人处世之道。

感谢软件学院一字班里的每一位同学，四年来我们共同学习、互帮互助、共同成长。

感谢林梓佳辅导员四年来的悉心指导和照顾，在我最困难的时候，给予我鼓励和帮助，谢谢林导！

最后，还要感谢在背后一直默默支持我的家人和朋友，谢谢你们！

声　明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：_____ 日　期：_____

附录 A　外文资料的调研阅读报告

As one of the most widely used techniques in operations research, *mathematical programming* is defined as a means of maximizing a quantity known as *objective function*, subject to a set of constraints represented by equations and inequalities. Some known subtopics of mathematical programming are linear programming, nonlinear programming, multiobjective programming, goal programming, dynamic programming, and multilevel programming[1].

It is impossible to cover in a single chapter every concept of mathematical programming. This chapter introduces only the basic concepts and techniques of mathematical programming such that readers gain an understanding of them throughout the book[2,3].