基于簇内乘积量化的最近邻检索方法

刘淑伟[1),2)] 陈 威[1),2)] 赵 伟[3)] 陈进才[1),2)] 卢 萍[2),3)]

[1)](华中科技大学武汉光电国家研究中心 武汉 430074)

[2)](华中科技大学信息存储系统教育部重点实验室 武汉 430074)

[3)](华中科技大学计算机科学与技术学院 武汉 430074)

摘 要 本文针对大规模高维数据近邻检索中的瓶颈问题,提出基于向量量化的一种检索方法—簇内乘积量化树方法.该方法运用向量量化和乘积量化的多层树状结构高效表征大规模高维数据集,与现有方法相比降低了索引表空桶率;其次提出基于贪心队列的近邻簇筛选方法减小了计算复杂度,加快了近邻检索速度;最后提出面量化方法用于近似计算候选数据集向量与查询向量间的距离,与点量化和线量化方法相比量化误差更小,提高了近邻查询准确率.本文提出的簇内乘积量化树算法在算子 Sift 和 Gist 描述的大规模高维数据集上与乘积量化树技术相比,首次召回准确率提高了 57.7%,索引表空桶率降低幅度在 50%以上,与局部优化乘积量化技术相比,查全率高达 97%,而查询时间却仅需原来的 1/9.实验结果表明本文提出的基于簇内乘积量化的近邻方法提升了近邻检索性能,为大规模高维数据集近邻检索提供了理论支持.

关键词 向量量化;乘积量化;最近邻检索;面量化;索引结构

中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2020.00303

Nearest Neighbor Search Based on Product Quantization in Clusters

LIU Shu-Wei[1),2)] CHEN Wei[1),2)] ZHAO Wei[3)] CHEN Jin-Cai[1),2)] LU Ping[2),3)]

[1)](*Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan* 430074)

[2)](*Key Laboratory of Information Storage System, Huazhong University of Science and Technology, Wuhan* 430074)

[3)](*School of Computer Science & Technology, Huazhong University of Science and Technology, Wuhan* 430074)

Abstract With the rapid development of Internet and multimedia technologies, the volume and dimension of data increase significantly which lead to the need for effective management of large scale and high dimensional data in many applications. Approximate Nearest Neighbor search is one of the most basic problems. Given a query vector, how to retrieve its nearest neighbor vectors in a large scale dataset quickly and accurately, the study of this problem faces many bottlenecks. Among them, the establishment of an efficient index structure to reconstruct the original dataset, making the output indexing table balanced and effectively fitting the dataset's underlying distribution is one of the most important tasks. In addition, due to the curse of dimension, estimating the distance between high-dimensional vectors effectively is also one of the bottlenecks in the nearest neighbor search. Aiming these difficult problems, we propose a neighbor search method based on product quantization in clusters. First, recombine the hierarchical relationship between vector quantization and product quantization to generate a more compact and balanced index structure which can better fit the original large scale and high dimensional dataset, and the empty bucket rate of the index table is significantly reduced. By adjusting the centroid scale of each layer, the

收稿日期:2018-04-11;在线出版日期:2019-03-18.本课题得到国家自然科学基金(61672246,61272068)、中央高校基本科研业务费专项资金(HUST:2016YXMS018)资助.**刘淑伟**,硕士研究生,主要研究方向为数据挖掘与机器学习.E-mail:amyliushu@foxmail.com.**陈 威**,博士研究生,主要研究方向为计算机视觉与模式识别.**赵 伟**,硕士研究生,主要研究方向为机器学习.**陈进才**(通信作者),博士,教授,博士生导师,中国计算机学会(CCF)高级会员,主要研究领域为信息存储与机器学习.E-mail:jcchen@hust.edu.cn.**卢 萍**,硕士,副教授,主要研究方向为信息存储与机器学习.

intra-cluster quantization tree structure can be flexibly applied to different neighbor search tasks on different large scale datasets. Second, we design a new approach for generating neighbor clusters based on greedy queues. The probability of containing neighboring vectors of the query in its nearest neighbor cluster is low, so a certain number of neighbor clusters are needed to provide enough candidate vectors to apply to the re-ranking part. We propose a greedy queue, which can generate sufficient neighbor clusters for query vector quickly compared with greedy algorithms. Third, our approach also includes an improved re-ranking method called plane quantilization approaching method which is inspired by line quantilization. The calculation of the distance between the candidate set and the query vector is an important part of the re-ranking phase and directly affects the query accuracy. However accurate Euclidean distance calculation between vectors in large-scale and high-dimensional datasets is time consuming and sometimes impossible to achieve under limited physical resources. Therefore, it is necessary to adopt approximate distance calculation approaches. Our method can more efficiently calculate the distance between high dimensional vectors and cause smaller approximation error compared to the state-of-art quantilization approaches, which significantly improved the accuracy of the nearest neighbor retrieval. In the experiments, we use several large scale and high dimensional dataset described by the operator Sift and Gist to measure our method respectively. Compared with the product quantization tree technique, the accuracy of the first recall is increased by 57.7% and the reduction rate of empty bin rate is more than 50%. Compared with local optimized product quantization technology, the recall rate can also be as high as 97%, but the required query time is reduced by 8 times. The experimental results show that our method based on product quantization in clusters can significantly improve the performance of nearest neighbor search, and is an effective solution for nearest neighbor search on large scale and high dimensional datasets.

Keywords vector quantilization; product quantilization; approaching nearest neighbor search; index structure; plane quanlitization

1 引 言

在大数据与云计算环境下,大规模高维度数据的有效管理成为众多应用的重点研究内容[1-3],最近邻(Nearest Neighbor)检索则是其中热点问题之一[4-6].给定一查询向量,如何在大规模高维集合中快速查询该向量的最近邻向量,该问题的研究主要面临两个瓶颈:一是大数据量的环境下,如何建立高效的索引使得索引表均衡紧凑,从而加快检索速度并保障检索准确率;二是如何克服维数灾难[7-8],使得高维向量间的欧氏距离计算复杂度降低.如 KD-tree[9]是多维数据索引算法中最经典的算法之一,但当数据维度达到一定程度的时候,它的时间和空间复杂度急剧增加,性能甚至低于线性扫描[10].为了克服这些难点,近似近邻(Approximate Nearest Neighbor,ANN)的概念被提出[11-13],其主要思想是用"近似"的方式来逼近精确值,从而减小搜索空间,保证一定检索精度的情况下,加快检索速度.目前大量的研究工作均是基于近似近邻的思想来设计检索系统的,这些方法大致可以分为两大类:第一类是基于哈希算法的方法,如局部敏感哈希算法(Locality Sensitivity Hashing)[11],该类方法将数据集依据哈希函数投影到若干桶内,在查询向量所在的桶内执行近邻搜索.这类方法的准确性和检索速度受限于哈希函数的选择[12-14],其应用在大规模高维数据集上的近邻检索效果并不理想[15].第二类是基于向量量化的方法,相比基于哈希的 ANN 算法,其特点是更为直观的对数据集进行空间划分,主要理论依据是向量量化和乘积量化,目前基于此类方法的近邻检索应用于大规模高维数据集上均有较好效果[10,15-17].但这些方法仍存在以下两点问题:(1)索引结构不能很好地拟合数据集概率分布,使得索引表空桶率较高;(2)重排序中距离近似算法引入的误差较大,使得高维向量间的距离估算失真严重.为了解决这些问题,本文提出基于簇内乘积量化近邻检

索方法—簇内乘积量化树方法(Clustered Product Quantilization Tree,C-PQT),主要贡献包括：

(1)设计了簇内乘积量化树结构.首先利用向量量化生成簇,并在簇内构造乘积量化树结构,应用在大规模高维数据集上,可显著降低索引表空桶率.

(2)提出贪心队列近邻簇生成方法.较以往用贪心算法生成近邻簇的方法计算复杂度低,显著加快了查询速度.

(3)提出面量化距离估算方法.该方法较现有点量化和线量化方法量化误差更小,从而有助于提高查询准确率.

2　相关概念与工作

2.1　向量量化与乘积量化

向量量化总体上讲是一个空间划分的过程,将输入的原始数据集空间通过量化器量化划分成规模较小的码本空间[16].例如,设数据集为 $\boldsymbol{X}\subset\mathbb{R}^D$,输入为向量 $\boldsymbol{x}\in\boldsymbol{X}$,向量量化器可理解为映射函数 q,向量 $\boldsymbol{x}$ 经量化得 $q(\boldsymbol{x})\in\boldsymbol{C}=\{\boldsymbol{c}_0,\boldsymbol{c}_1,\cdots,\boldsymbol{c}_{k-1}\}$,其中 $\boldsymbol{C}$ 是数据集 $\boldsymbol{X}$ 经量化器量化得到的特征值集合又称作码本,这里码本大小为 k,$\boldsymbol{c}_{i=0\cdots k-1}$ 是该码本中的质心.经量化至相同质心的向量集合称 Voronoi Cell[18],下文均简称为簇,式(1)定义由质心 $\boldsymbol{c}_i$ 对应的簇 $\boldsymbol{V}_i$：

$$\boldsymbol{V}_i\overset{\text{def}}{=}\{\boldsymbol{x}\in\boldsymbol{X}:q(\boldsymbol{x})=\boldsymbol{c}_i\}\tag{1}$$

可见,由向量量化生成的 k 个簇是对原始数据集的一次空间划分,被划分至相同簇内的向量都将由同一质心 $\boldsymbol{c}_i$ 表达.衡量一个量化器的好坏由数据集中所有向量与对应的量化质心间的均方误差 $MSE(q)$ 决定,该值又称作量化器的量化误差[19].如式(2)所示,量化器 q 在数据集 $\boldsymbol{X}$ 上引入的量化误差：

$$\begin{aligned}MSE(q)&=E[d(q(\boldsymbol{x}),\boldsymbol{x})^2]\\&=\int p(\boldsymbol{x})d(q(\boldsymbol{x}),\boldsymbol{x})^2\mathrm{d}\boldsymbol{x}\end{aligned}\tag{2}$$

其中,$d(q(\boldsymbol{x}),\boldsymbol{x})=\|\boldsymbol{x}-q(\boldsymbol{x})\|$ 表示向量 $\boldsymbol{x}$ 和质心 $q(\boldsymbol{x})$ 的欧氏距离;$p(\boldsymbol{x})$ 是数据集 $\boldsymbol{X}$ 的概率分布函数,在实际运算中,可遇到任意概率分布的数据集.获取数据集分布 $p(\boldsymbol{x})$ 的计算复杂度高,常采用蒙特卡洛[20]随机采样方法近似数据集的概率分布,再按照式(2)来衡量量化器的优良.

应用向量量化对大规模高维数据集量化时,为了缩小量化误差,需增加码本规模.这时应用 Lloyd 迭代学习算法[21]训练码本计算复杂度会随之增加,况且码本存储随着 k 值线性增大,在有限的物理资源条件下有时根本不可能实现.

乘积量化总体上讲是一个空间转换过程,将原有高维数据集空间转换成较低维度的笛卡尔积空间.具体过程如下:设输入向量 $\boldsymbol{x}\in\mathbb{R}^D$,将它按序拆分成 m 个低维度向量 $\boldsymbol{u}_j(\boldsymbol{x})$,$j=0,\cdots,m-1$,$\boldsymbol{u}_j(\boldsymbol{x})\in\mathbb{R}^{D'}$,其中 $D'=D/m$,需保证 m 能整除 D,则向量 $\boldsymbol{x}$ 被拆分成 m 个子向量：

$$\underbrace{\boldsymbol{x}_1,\cdots,\boldsymbol{x}_{D'}}_{u_0(\boldsymbol{x})},\cdots,\underbrace{\boldsymbol{x}_{D-D'+1},\cdots,\boldsymbol{x}_D}_{u_{m-1}(\boldsymbol{x})}$$

同样原始数据集 $\boldsymbol{X}$ 也将被拆分成 m 个子空间记作 $\boldsymbol{U}_{j=0,\cdots,m-1}\subset\mathbb{R}^{D'}$,接下来对 m 个子空间中的向量按照向量量化处理方法完成簇划分.设输入向量 $\boldsymbol{x}$ 的子向量 $\boldsymbol{u}_j(\boldsymbol{x})\in\boldsymbol{U}_j$,$q_j$ 是子空间 $\boldsymbol{U}_j$ 上的向量量化器,$\boldsymbol{C}_j$ 是该空间的向量量化码本,那么向量 $\boldsymbol{x}$ 经 m 组独立的向量量化后的量化值如下所示：

$$\underbrace{\boldsymbol{x}_1,\cdots,\boldsymbol{x}_{D'}}_{u_0(\boldsymbol{x})},\cdots,\underbrace{\boldsymbol{x}_{D-D'+1},\cdots,\boldsymbol{x}_D}_{u_{m-1}(\boldsymbol{x})}\rightarrow$$
$$q_0(u_0(\boldsymbol{x})),\cdots,q_{m-1}(u_{m-1}(\boldsymbol{x}))$$

该值就是向量 $\boldsymbol{x}$ 的乘积量化值.其中子空间质心 $q_j(\boldsymbol{u}_j(\boldsymbol{x}))\in\boldsymbol{C}_j$,原始数据集 $\boldsymbol{X}$ 经乘积量化得的码本 $\boldsymbol{C}_{\text{product}}$ 是这 m 组独立的向量量化码本的笛卡尔积,如式(3)所示：

$$\boldsymbol{C}_{\text{product}}=\boldsymbol{C}_0\times\cdots\times\boldsymbol{C}_{m-1}\tag{3}$$

设独立向量量化码本 $\boldsymbol{C}_{j=0,\cdots,m-1}$ 的大小均为 k'(为了保证均衡的量化效果,子空间的码本大小均一致[19]),可见 $\boldsymbol{C}_{\text{product}}$ 的码本大小为 $(k')^m$.其中 m 记作乘积量化的划分组数.我们注意到当 $m=1$ 时乘积量化退化成向量量化.接下来给出乘积量化量化误差 $MSE(q_{\text{product}})$ 的计算如式(4)所示：

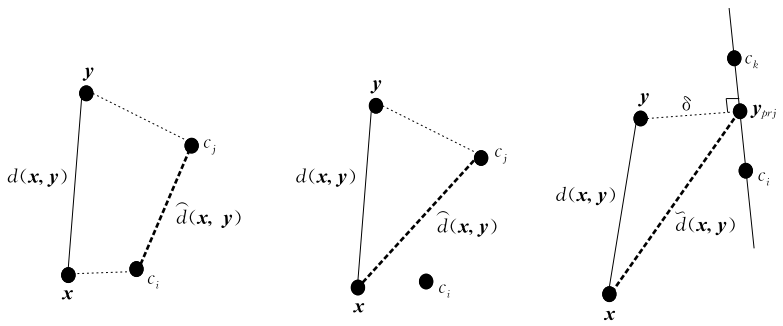
$$MSE(q_{\text{product}})=\sum_j MSE(q_j)\tag{4}$$

其中,$MSE(q_j)$ 是子空间向量量化的量化误差,$MSE(q_{\text{product}})$ 随乘积量化组数 m 和子空间向量维度 D' 变化.其中文献[10]指出,相同码本存储开销时,将原始数据集划分成少量组数,较大码本的子空间相比多组数小码本的子空间,整体得到的量化误差会更小,即在训练中应使 m 较小 k' 较大.

2.2　距离近似方法

在重排序阶段,需要比对查询向量和候选集向量间的距离,获得有序的近邻结果集.但高维向量间精确的欧氏距离计算十分耗时,面向大规模高维数据集有时根本不可能实现,因而在近邻搜索中需采用距离近似求解方案.如文献[10]提出的点量化距离估算方法,它包括对称距离估算和非对称距离估

算,以及文献[17]提出的线量化估算算法,它是现有距离近似的经典方法.其中对称距离估算和非对称距离估算的主要思想是将数据集向量用距离它最近的质心近似表达,然后完成与查询向量的距离计算;线量化估算算法即将数据集向量用它到码本质心构成的最近邻线上的投影点近似表达.如图1所示,其中数据集向量为 $\boldsymbol{y}$,查询向量为 $\boldsymbol{x}$,两者欧氏距离记作 $d(\boldsymbol{x},\boldsymbol{y})$,按照不同近似方法估算的距离记作 $\tilde{d}(\boldsymbol{x},\boldsymbol{y})$.



(a) 对称距离估算方法 (b) 非对称距离估算方法 (c) 线量化距离估算方法

图1 点量化与线量化距离近似方法

2.3 相关工作

Hartigan 和 Wong 首次提出并实现应用向量量化做近邻检索[22],即将原始数据集空间量化成有限的码本质心空间.但当面对大规模数据集时,为了控制量化误差变得更小,码本规模也随之增加.体积庞大的码本一是占用较大内存空间,二是严重影响查询速度,且随着数据维度的增加,训练码本的复杂度急剧增加,在有限的物理资源条件下,难以实现;Jégou 等人提出倒排索引技术(IVFADC)[10],该方法首先利用向量量化做初步的粗糙量化,依据量化编码依次将数据集向量插入倒排索引表中,倒排顺序依据向量与量化特征值之间的距离. 该方法的不足之处在于索引表对原始数据集概率分布拟合度低,使得近邻查询时需要与50多万个候选集向量进行距离比对才能使得准确度提升至90%以上;鉴于乘积量化的优势,Babenko 等人提出倒排多目索引近邻检索技术(IMI)[23],该方法应用乘积量化处理原始数据集空间,并采用贪心算法生成近邻簇,相比于上述两种技术,可一定程度降低码本体积庞大对查询速度的影响,但采用贪心算法计算近邻簇的方法计算复杂度和所需的线上存储开销较大,影响近邻检索速度;Ge 等人在乘积量化的基础上提出优化乘积量化技术(OPQ)[24],通过对输入向量执行随机优化旋转从而更准确的拟合数据集空间分布,进一步缩小量化误差,但有时原始数据集分布极不均衡,优化乘积量化方法并不能对原始数据集进行很好的拟合,此时的量化误差增大,近邻查询则不理想;于是 Kalantidis 等人提出局部优化乘积量化方法(LOPQ)[16],在乘积量化各子空间执行优化旋转操作,较优化乘积量化方法更为灵活,使其可获得更小的量化误差,并广泛适用于不同分布的数据集上;Wieschollek 等人提出了乘积量化树方法(PQT)[17],该方法通过堆叠乘积量化和向量量化构造树形索引结构,较前述方法可更有效的缩小量化粒度,使得码本空间和量化误差均显著减小,除此之外在重排序阶段引入线量化距离近似方法来近似表达原始数据集向量,可大大提升重排序阶段的准确性,进而提高查询准确率.此外为了加快近邻簇筛选过程,该方法提出应用固定倾角的扫描器求解近邻桶,相比贪心算法求解策略复杂度减小,一定程度上加快了近邻检索速度.上述方法的不足之处在于索引结构对数据集概率分布拟合度低,使得索引表空桶率较高及重排序中距离近似算法引入的误差较大,使得高维向量间的距离估算失真严重.

本文设计的簇内乘积量化树是基于向量量化和乘积量化构建的三层树状结构.其中第一层应用向量量化将数据集划分为多个簇,在各簇内独立构造第二、三层的乘积量化树.树型结构可以有效对数据进行分类,加速遍历索引的过程,层次越多,数据分类越精细,可以有效的缩小搜索范围.受文献[10]和[17]的启发,本文保留向量量化的粗粒度量化能力,并利用乘积量化树对数据集细粒度的划分优势,重新组合乘积量化和向量量化的层次关系,设计了一种与数据集概率分布拟合度更高的索引结构,簇是由 VQ 产生的,弥补了 PQ 分类准确性较低的缺点,相比经典量化近邻检索算法显著降低了索引表的空桶率,使得索引表更紧凑和均衡;其次提出贪心队列使得近邻簇筛选计算复杂度更小,从而加快了近邻检索的速度;最后受线量化距离近似方法的启发,本文首次提出面量化距离估算方法,进一步降低了距

离近似误差，使得重排序更为准确，进而提高了近邻查询的准确率.

3　C-PQT：簇内乘积量化树检索方法

3.1　簇内乘积量化树结构设计

簇内乘积量化树是基于向量量化和乘积量化构建的三层树状结构，如图 2 所示. 第一层应用向量量化将数据集划分为 k_1 个簇，设该层量化器为 $q^{(1)}$，向量 $\boldsymbol{x}\in\boldsymbol{X}$，$\boldsymbol{X}\subset\mathbb{R}^D$，则向量 $\boldsymbol{x}$ 在该层的最近邻质心为 $q^{(1)}(\boldsymbol{x})$. 第一层向量量化实现了将数据集 $\boldsymbol{X}$ 初步切分成 k_1 个独立的部分即 $\boldsymbol{X}\rightarrow\{\boldsymbol{X}_0,\cdots,\boldsymbol{X}_{k_1-1}\}$，其中 $\boldsymbol{X}_{i=0,\cdots,k_1-1}\subset\mathbb{R}^D$；第二层运用乘积量化技术将数据集在第一层的基础上继续划分，即分别对数据集 $\boldsymbol{X}_i$ 执行乘积量化，第二层是簇间相互独立的，但为了保证相同的划分粒度，簇内乘积量化的参数配置是一致的，如图 2 所示. 设乘积量化的组数为 P（保证 P 可以被数据维度 D 整除），$\boldsymbol{C}_{i,j}(i=0,\cdots,k_1-1;j=0,\cdots,P-1)$ 表示第一层第 i 个簇下的第 j 个子空间的向量量化码本，其大小为 k_2. 设第 i 个簇内的乘积量化器设为 $q_i^{(2)}$，向量 $\boldsymbol{x}\in\boldsymbol{X}_i$，乘积量化过程为

$$q_i^{(2)}(\boldsymbol{x})\rightarrow q_{i,0}^{(2)}(\boldsymbol{u}_0(\boldsymbol{x})),\cdots,q_{i,P-1}^{(2)}(\boldsymbol{u}_{P-1}(\boldsymbol{x})),$$

其中 $q_{i,j}^{(2)}$ 表示第 i 个簇中第 j 子空间的向量量化器，$\boldsymbol{u}_j(\boldsymbol{x})$ 是向量 $\boldsymbol{x}$ 的第 j 个子空间向量，$\boldsymbol{u}_j(\boldsymbol{x})\in\mathbb{R}^{D/P}$，$q_{i,j}^{(2)}(\boldsymbol{u}_j(\boldsymbol{x}))\in\boldsymbol{C}_{i,j}$，第二层作乘积量化得到的码本记为

$$\boldsymbol{C}_{cp}^{(2)}=\sum_{i=0}^{k_1-1}(\boldsymbol{C}_{i,0}\times\cdots\times\boldsymbol{C}_{i,P-1}),$$

其中×代表笛卡尔积，则 $\boldsymbol{C}_{cp}^{(2)}$ 码本大小为 $k_1\cdot(k_2)^P$. 第三层是为了细化划分粒度，在二层子空间质心所表达的簇内继续执行向量量化，设码本大小为 k_3，同样为了均衡划分效果，第三层向量量化的参数配置在各簇中均保持一致. 综上，将该簇内乘积量化树视为量化器 q_{c-pqt}，其码本大小为 $k_1\cdot(k_2\cdot k_3)^P$.

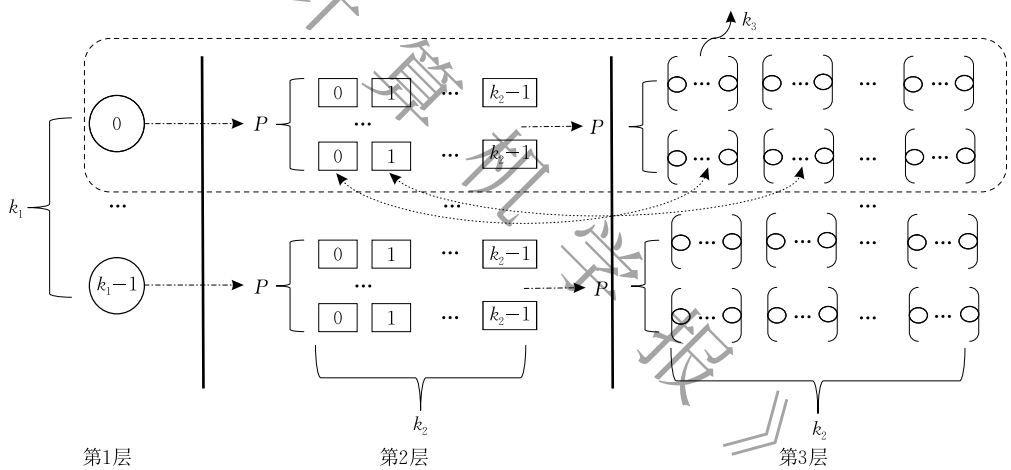


图 2　簇内乘积量化树结构设计

3.2　簇内乘积量化树的剪枝性遍历

簇内乘积量化树的遍历是为了将数据集中的向量映射到最近邻簇中. 设输入是数据集 $\boldsymbol{X}$ 中向量 $\boldsymbol{y}\in\boldsymbol{X}$，$\boldsymbol{X}\subset\mathbb{R}^D$，$w_1$ 为第一层剪枝系数，w_2 为第二层剪枝系数，输出是向量 $\boldsymbol{y}$ 的最近邻簇编码，并以该编码作为索引码将向量插入到索引表中. 向量 $\boldsymbol{y}$ 的遍历过程如下所述分为 5 步：

(1) 计算向量 $\boldsymbol{y}$ 与第 1 层向量码本质心的距离并排序，由近至远取前 w_1（$w_1\leq k_1$）个最近邻质心即 $\{\boldsymbol{c}_{ord_1}^{(1)},\cdots,\boldsymbol{c}_{ordw_1}^{(1)}\}$.

(2) 计算向量 $\boldsymbol{y}$ 与上述 w_1 个簇内对应的第二层乘积量化码本质心间的距离，并在各子空间进行排序，各子空间保留前 w_2（$w_2\leq w_1\cdot k_2$）个最近邻质心，则第二层筛选出的质心记作矩阵

$$\begin{Bmatrix}\boldsymbol{c}_{1,ord_1}^{(2)} & \cdots & \boldsymbol{c}_{1,ordw_2}^{(2)}\\ \cdots & \cdots & \cdots\\ \boldsymbol{c}_{P,ord_1}^{(2)} & \cdots & \boldsymbol{c}_{P,ordw_2}^{(2)}\end{Bmatrix}_{P\times w_2}.$$

(3) 计算向量 $\boldsymbol{y}$ 与上述簇矩阵中的第三层向量码本质心间的距离，即各子空间内需要计算 $W=w_2\cdot k_3$ 次，然后分组进行排序，得到各子空间第三层质心的有序排列矩阵 $\boldsymbol{MatC}$ 为

$$\begin{Bmatrix}\boldsymbol{c}_{1,ord_1}^{(3)} & \cdots & \boldsymbol{c}_{1,ord_M}^{(3)}\\ \cdots & \cdots & \cdots\\ \boldsymbol{c}_{P,ord_1}^{(3)} & \cdots & \boldsymbol{c}_{P,ord_M}^{(3)}\end{Bmatrix}_{P\times W}.$$

(4) 向量 $\boldsymbol{y}$ 的最近邻簇为矩阵 $\boldsymbol{MatC}$ 的第一列元素的集合，其最近邻簇的编码方法如式(5)所示：

$$code_{\text{best}}=\sum_{i=1}^{P}\phi_i\cdot base^i \tag{5}$$

其中 $base=k_1\cdot k_2\cdot k_3$ 是基底即第一层生成簇下包含的第三层质心的数量，ϕ_i 是 ***MatC*** 的第一列各质心的编号，如质心 $\boldsymbol{c}^{(3)}_{1,ord_1}$ 对应编号为 ϕ_1. 如图 2 所示已给出第一层和第二层质心对应的簇编号，在此基础上计算 ϕ_i 为：$\phi_i=\phi'_i\cdot k_2+\phi''_i\cdot k_3$，其中 $i=1,\cdots,p$，$\phi'_i\in\{0,\cdots,k_1-1\}$ 是 ϕ_i 所属的第一层簇编号，$\phi''_i\in\{0,\cdots,k_2-1\}$ 是 ϕ_i 所属的第二层簇编号.

(5) 将向量 $\boldsymbol{y}$ 的指针插入到索引表编码为 $code_{\text{best}}$ 的桶内.

可见，对簇内乘积量化树的剪枝遍历过程，就是分层次剪枝的过程，目的是在空间中找到向量的最近邻簇，尽可能的缩小量化的误差. 在实际的实现中可以通过调整参数 w_1 和 w_2 控制精确距离计算的次数，为了确保较高的准确度，应满足 $w_1\geq 1/4k_1$，$w_2\geq 1/8k_2$. 且簇内乘积量化树结构的灵活性可以通过调整参数 k_1,k_2,k_3 实现，当 $k_1=1$ 簇内乘积量化树退化成乘积量化树结构，当 $k_2=k_3=1$ 簇内乘积量化树退化成倒排索引结构.

3.3 近邻簇筛选

对于查询向量 $\boldsymbol{x}$，通过簇内乘积量化树的剪枝遍历可获取其最近邻簇，那么索引表中这一项中包含的所有数据集的向量均成为查询向量 $\boldsymbol{x}$ 的近邻候选向量. 然而大量实验研究表明最近邻簇内包含查询向量最近邻结果向量的概率并不高，为了提高查询准确率需要簇筛选过程来获取足够的近邻簇，从而为重排序阶段提供足够的候选集向量，如在大规模数据集上做近邻查询，一般需要 10 000～20 000 个候选向量. 若索引表中桶的平均大小为 100 左右，则需要生成 100～200 个近邻簇.

簇筛选是近邻查询技术的重要环节，文献[9-10]将近邻簇的筛选过程抽象成一个图的最短路径问题，运用贪心算法和局部最优策略依次由近至远地生成近邻簇. 这种方法做到近邻簇严格按照欧氏距离由近至远的标准筛选，但是为了给贪心算法做准备，需要在线计算所有可能的簇与查询向量的距离，查询时会占用较多的线上计算资源. 本文采用贪心队列去近似求解近邻簇. 设查询向量 $\boldsymbol{x}\subset\mathbb{R}^D$，在簇内乘积量化树遍历的第三步我们初步得到向量 $\boldsymbol{x}$ 的第三层码本质心的有序矩阵 ***MatC***，下面接着分析 ***MatC*** 如何按照贪心队列去近似生成查询向量 $\boldsymbol{x}$ 的近邻簇.

设第二层乘积量化的划分组数为 P，剪枝系数为 w_2，第三层向量量化的码本大小参数为 k_3，则贪心队列的生成如下：

(1) 确定贪心队列的大小，由于乘积量化组数为 P，所以贪心队列每行容纳 P 个元素，各元素均有 $w_2\cdot k_3$ 种可能，则 P 个元素的组合种类共计 $(w_2\cdot k_3)^P$，所以贪心队列的行数共计 $(w_2\cdot k_3)^P$，贪心队列可表达为 $P\times(w_2\cdot k_3)^P$ 的矩阵.

(2) 填充贪心队列的元素，将行号 $row=0,\cdots,(w_2\cdot k_3)^P-1$ 转化成以 $w_2\cdot k_3$ 基底的进制数，依次从右至左的填充到该行中，如 $row=1$ 转换成 $w_2\cdot k_3$ 进制数为

$$\underbrace{0,\cdots,0,1}_{P\text{ bits}},$$

第 1 行的贪心队列元素依次为

$$1,\underbrace{0,\cdots,0}_{P-1\text{个}0}.$$

(3) 重排贪心队列的行，计算各行元素的平方和：

$$s_i=\sum_{j=1}^{P}a_{i,j}^2$$

($a_{i,j}$ 表示原贪心队列第 i 行第 j 列元素)，以其升序重排各行，得到重排后贪心队列矩阵为

$$(a'_{i,j})_{(w_2\cdot k_3)^P\times P}.$$

(4) 重构矩阵 ***MatC*** 列向量，调整方法为：依照贪心队列的第 i 行元素 $(a'_{i,1},\cdots,a'_{i,P})$ 值重排矩阵 ***MatC*** 第 i 列元素，则该列向量重排后为 $(\boldsymbol{c}^{(3)}_{1,a'_{i,1}},\cdots,\boldsymbol{c}^{(3)}_{P,a'_{i,P}})^{\mathrm{T}}$，依照这种规则重构矩阵 ***MatC*** 各列，取调整后的矩阵 ***MatC*** 的前 M 列作为查询向量 $\boldsymbol{x}$ 的前 M 个近邻簇.

3.4 面量化距离估算方法

重排序过程是用来解决在候选集合内快速定位查询向量的最近邻向量的问题. 候选集合向量和查询向量间的距离计算是重排序阶段的重要环节，直接影响查询准确率. 但面对高维数据集，向量间精确的欧式距离计算十分耗时，有时在有限的物理资源条件下根本不可能实现，故需采取近似距离的计算策略. 文献[10]提出点量化距离近似算法，文献[17]提出了线量化距离近似算法，这两种方法的确大大提高了重排序的效率，但距离近似误差仍对查询准确率带来较大的影响. 在上述两种距离近似算法的基础上，本文首次提出面量化距离近似方法，即将数据集向量由它到由码本质心构成的最近邻面上的投影点来近似表达，该方法较点量化和线量化量化误差更小，实验表明相同情况下，重排序过程中应用面量化方法，可获得更高的查询准确率.

设数据集向量 $\boldsymbol{y}\in\mathbb{R}^D$，如图 3 所示，$[\boldsymbol{C}_i]_p$ 表示乘积量化第 P 个子空间的第 i 个码本质心，则在

第二层乘积量化第 P 个子空间中，设向量 $\boldsymbol{y}$ 在该部分的子向量 $\boldsymbol{u}_p(\boldsymbol{y})$，执行点量化将 $\boldsymbol{u}_p(\boldsymbol{y})$ 量化成距离它最近的子空间码本质心 $[\boldsymbol{C}_i]_p$，该点又可记作 $q_{\text{point}}(\boldsymbol{u}_p(\boldsymbol{y}))$，量化误差为两点间的距离 Δ_i；固定点 $[\boldsymbol{C}_i]_p$，寻找另一码本质心，使得 $\boldsymbol{u}_p(\boldsymbol{y})$ 到线 $[\boldsymbol{C}_i]_p-[\boldsymbol{C}_j]_p$ 的距离最短，执行线量化得到投影点 $q_{\text{line}}(\boldsymbol{u}_p(\boldsymbol{y}))$ 又可记作 S_1，投影点位置通过求解 $\bar{\lambda}_p$ 确定，线量化误差为 ε_i 是点 $\boldsymbol{u}_p(\boldsymbol{y})$ 至投影点 $q_{\text{line}}(\boldsymbol{u}_p(\boldsymbol{y}))$ 的距离；再寻找一子空间码本质心 $[\boldsymbol{C}_k]_p$，使得由这三个质心构成的面距离向量点 $\boldsymbol{u}_p(\boldsymbol{y})$ 最近，这个面就是向量 $\boldsymbol{y}$ 在乘积量化空间中第 P 子空间的最近邻面，即距离 $\boldsymbol{u}_p(\boldsymbol{y})$ 的最近邻面记作 S_p. 面量化距离近似方法就是用 $\boldsymbol{u}_p(\boldsymbol{y})$ 到 S_p 上的投影点 $q_{\text{line}}(\boldsymbol{u}_p(\boldsymbol{y}))$ 近似表达它，量化误差为 δ_i 是点 $\boldsymbol{u}_p(\boldsymbol{y})$ 到点 $q_{\text{line}}(\boldsymbol{u}_p(\boldsymbol{y}))$ 的距离. 在乘积量化各子空间找到向量 $\boldsymbol{y}$ 的子空间向量的最近邻面的投影点 $q_{\text{plane}}(\boldsymbol{u}_{p=1,\cdots,P}(\boldsymbol{y}))$，即完成数据集向量 $\boldsymbol{y}$ 的近似表达.

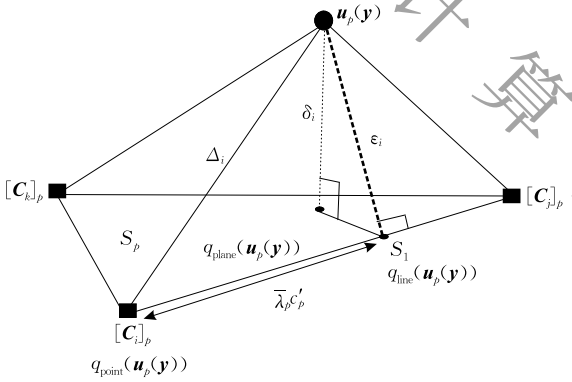


图 3　面量化示意图

直接计算面投影点 $q_{\text{line}}(\boldsymbol{u}_p(\boldsymbol{y}))$ 复杂度高，本文用多次线投影近似求解该点，如图 4 所示. 首先连接点 S_1 和 $[\boldsymbol{C}_k]_p$，两点间距离为 i'_p，应用线量化将点 $\boldsymbol{u}_p(\boldsymbol{y})$ 投影至线 $[\boldsymbol{C}_j]_p-S_1$ 上，投影点为 S_2，投影点位置通过求解 $\tilde{\lambda}_p$ 确定，两点间直线距离为 δ' 又可理解为点 S_2 与点 $\boldsymbol{u}_p(\boldsymbol{y})$ 之间的误差距离；最后连接点 $[\boldsymbol{C}_j]_p$、S_2 并延长与线 $[\boldsymbol{C}_i]_p-[\boldsymbol{C}_k]_p$ 的交点为 $\boldsymbol{C}'_p$，将 $[\boldsymbol{Y}_i]_p$ 投影至线 $[\boldsymbol{C}_j]_p-\boldsymbol{C}'_p$ 上（两点间距离为 c'_p），投影点为 S_3. 同理 $\boldsymbol{C}'_p$ 可由 λ_p 确定具体位置，S_3 与点 $\boldsymbol{u}_p(\boldsymbol{y})$ 的误差距离为 δ''. 由于直角三角形斜边最大，即在直角三角形 $\triangle(\boldsymbol{u}_p(\boldsymbol{y}),S_2,S_1)$ 中 $\delta'<\varepsilon_i$，在直角三角形 $\triangle(\boldsymbol{u}_p(\boldsymbol{y}),S_2,S_3)$ 中 $\delta''<\delta'$，推得 $\delta''\leq\delta'\leq\varepsilon_i$. 用投影点 S_3 去近似 $\boldsymbol{u}_p(\boldsymbol{y})$ 的面投影点 $q_{\text{plane}}(\boldsymbol{u}_p(\boldsymbol{y}))$ 会带来一定的误差，因为 $\delta_i\leq\delta''$. 理论上我们可以持续将 $\boldsymbol{u}_p(\boldsymbol{y})$ 进行多次线投影，使得与投影点间的误差持续变小，显然代价是算法的时间和空间复杂度会增大，权衡两者，我们在实验中只进行上述三次投影去近似确定面投影点. 由于只进行三次投影，即使三个质心相同，但是计算顺序不同，得到的近似投影点也会不同，所以计算顺序对查询准确度有影响. 我们选取所有三个质心的组合，计算出每个组合的 δ''，选择 δ'' 最小的一组作为最终的三个质心点.

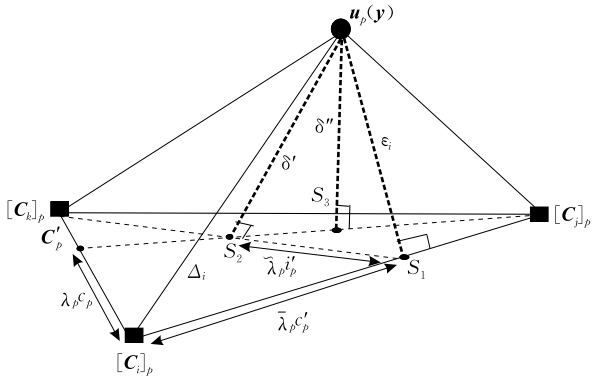


图 4　面投影点的近似求解

接着给出本文提出的面量化距离估算方法与点量化和线量化在量化误差上的比对，在直角三角形 $\triangle(\boldsymbol{u}_p(\boldsymbol{y}),[\boldsymbol{C}_i]_p,S_1)$ 中 $\Delta_i>\varepsilon_i$，综上所得 $\Delta_i>\varepsilon_i>\delta_i$. 本文提出的面量化距离估算方法，其量化误差经证明小于点量化和线量化方法. 在构建数据库阶段面量化比线量化需要花费更多的时间，查询阶段的距离计算部分也比线量化多了一步，但是这个计算时间很短，对整个查询时间影响很小.

完成数据集向量的近似表达，至此可以探讨基于面量化，查询向量与候选向量间的距离估算. 设查询向量 $\boldsymbol{x}$，候选向量集合为 L_C，向量 $\boldsymbol{y}\in L_C$ 两者之间的距离由式(6)得：

$$d(\boldsymbol{x},\boldsymbol{y})^2=\sum_{p=1}^{P}d(\boldsymbol{u}_p(\boldsymbol{x}),\boldsymbol{u}_p(\boldsymbol{y}))^2 \tag{6}$$

向量 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 的距离可分解成第二层乘积量化空间中各子向量间的距离和. 如图 5 所示，在第二层乘

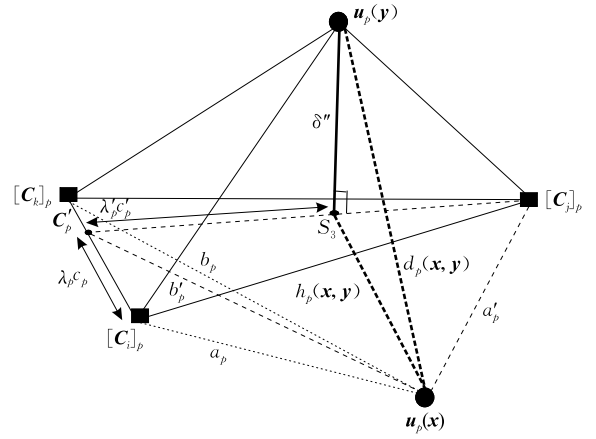


图 5　向量间距离近似求解示意

积量化第 p 个子空间中，查询向量 $\boldsymbol{x}$ 的子向量 $\boldsymbol{u}_p(\boldsymbol{x})$ 与候选集向量 $\boldsymbol{y}$ 的子向量 $\boldsymbol{u}_p(\boldsymbol{y})$ 的距离可定义为 $d_p(\boldsymbol{x},\boldsymbol{y})\overset{\text{def}}{=}d(\boldsymbol{u}_p(\boldsymbol{x}),\boldsymbol{u}_p(\boldsymbol{y}))$.

我们将点 $\boldsymbol{u}_p(\boldsymbol{y})$ 由点 S_3 近似表达，则距离 $d_p(\boldsymbol{x},\boldsymbol{y})$ 可由 $\boldsymbol{u}_p(\boldsymbol{x})$ 和投影点 S_3 的距离 $h_p(\boldsymbol{x},\boldsymbol{y})\overset{\text{def}}{=}d(\boldsymbol{u}_p(\boldsymbol{x}),s_3)$ 逼近. 经三角运算推得：

$$h_p(\boldsymbol{x},\boldsymbol{y})=b_p'^2+\lambda_p'^2\cdot c_p'^2+\lambda_p'(a_p'^2-b_p'^2-c_p'^2)$$

其中：

$$b_p'^2=a_p^2+\lambda_p^2\cdot c_p^2-\lambda_p(a_p^2-b_p^2+c_p^2),$$

$$c_p'^2=a_{1p}^2+\lambda_p^2\cdot c_p^2-\lambda_p(a_{1p}^2-b_{1p}^2+c_p^2),$$

$$\lambda_p=\frac{\tilde{\lambda}_p}{1-\bar{\lambda}_p+\bar{\lambda}_p\tilde{\lambda}_p},$$

则式(6)可进一步得式(7)：

$$\begin{aligned}d(\boldsymbol{x},\boldsymbol{y})^2&\approx\sum_{p=1}^{P}h_p(\boldsymbol{x},\boldsymbol{y})^2\\&=\sum_{p=1}^{P}(b_p'^2+\lambda_p'^2\cdot c_p'^2+\lambda_p'(a_p'^2-b'^2-c_p'^2))\end{aligned}\tag{7}$$

由于数据集中的任意向量均可能成为查询的候选集向量，为了加快查询的线上速度，我们将式(7)中出现的 5 个参数组分别提前计算并保存至数据库中，即需为每一个数据集向量保存一个 $5\cdot P$ 大小的重排信息元组：

$$(\lambda_1,\cdots,\lambda_p;\lambda_1',\cdots,\lambda_p';i_1,\cdots,i_p;j_1,\cdots,j_p;k_1,\cdots,k_p).$$

4 簇内乘积量化树的最近邻查询

根据第 3 节中对簇内乘积量化树的各个关键结构的说明，本节给出应用簇内乘积量化树做最近邻检索的一般步骤，设查询向量 $\boldsymbol{x}\in\mathbb{R}^D$，其前 N 个最近邻结果向量集合可按照如下分 6 个步骤获取(其中步骤(1)～(3)为根据设计的簇内乘积量化树结构，进行剪枝性遍历)：

(1) 计算查询向量 $\boldsymbol{x}$ 与第一层向量码本质心欧式距离并依照距离排序，由近至远取前 w_1 个最近邻簇即 $\{\boldsymbol{C}_{1,ord_1},\cdots,\boldsymbol{C}_{1,ordw_1}\}$.

(2) 计算查询向量 $\boldsymbol{x}$ 与上述筛选出来的 w_1 簇内的第二层乘积量化码本质心的欧式距离，并依据该距离分组进行排序，每组保留前 w_2 个最近簇，第二层有序质心矩阵为

$$\begin{Bmatrix}\boldsymbol{C}_{1,ord_1}^{(2)} & \cdots & \boldsymbol{C}_{1,ordw_2}^{(2)}\\ \cdots & \cdots & \cdots\\ \boldsymbol{C}_{P,ord_1}^{(2)} & \cdots & \boldsymbol{C}_{P,ordw_2}^{(2)}\end{Bmatrix}_{P\times w_2}.$$

(3) 计算查询向量 $\boldsymbol{x}$ 与上述矩阵中第三层量化码本质心间的距离，即共需要计算 $W=w_2\cdot k_3$ 次. 然后依据该距离分组排序，得到第三层各子空间有序的质心排列矩阵 $\boldsymbol{MatC}$ 为

$$\begin{Bmatrix}\boldsymbol{C}_{1,ord_1}^{(3)} & \cdots & \boldsymbol{C}_{1,ord_M}^{(3)}\\ \cdots & \cdots & \cdots\\ \boldsymbol{C}_{P,ord_1}^{(3)} & \cdots & \boldsymbol{C}_{P,ord_M}^{(3)}\end{Bmatrix}_{P\times W}.$$

(4) 对查询向量 $\boldsymbol{x}$ 的近邻簇筛选，即将矩阵 $\boldsymbol{MatC}$ 按照贪心队列调整. 并按照式(5)求得前 M 个近邻簇的索引码.

(5) 获取查询向量 $\boldsymbol{x}$ 的近邻向量候选集合 L_C，即依照步骤 4 中获取的前 M 个近邻簇编码在索引表中查询，将 M 个簇内的所有数据集向量放入候选集合 L_C 中.

(6) 重排序，按照式(7)依次计算查询向量 $\boldsymbol{x}$ 与候选集合 L_C 中的每一个向量的面量化近似距离，并依据该距离对候选集合 L_C 中的向量重新排序. 重排序结束返回前 N 个最近邻结果向量.

可见查询的过程与遍历的过程基本对称，在数据集向量的遍历过程中将数据集的向量依次放入索引表，并计算重排信息元组存入数据库中，方便查询向量在重排序阶段读取，此外重排序阶段可以选择对筛选出来的前 N 个最近邻向量进行精确距离的排序再输出，消耗是 N 次 D 维的欧式距离计算. 当 N 较小的时候，可以进一步精确排序，较大则不推荐.

5 实验测评与分析

5.1 实验环境

本节通过实验评估簇内乘积量化树的最近邻检索性能. 实验采用的服务器平台搭载两个 Intel(R) Xeon(R) E5-2620v4，8 核 2.10GHz 处理器，32GB 的内存，操作系统为 Ubuntu 16.04，实现采用 c++ 单线程模式，指令集为 SSE2. 用到的工具包有 c++ Eigen、Timer 和标准容器库等. 选取的公开测试数据集为 ANN_SIFT10K、ANN_SIFT1M 和 ANN_GIST1M，数据集详细描述如表 1 所示. 每个数据集均包括三个部分，分别为训练数据集合、基础数据集合和查询数据集合. 其中查询数据集合附着谜底文件(groundtruth file)，标识数据集合中每个向量的前 K 个真实最近邻向量，对于上述三个数据集，K 值为 100. 训练数据集合是用于训练簇内乘积量化树的索引结构，即簇内乘积量化树的各层质心；基础数据集合用于构建索引表和数据库，完成数据插入和预计算必要的重排信息；查询数据集合借助谜底文件做近邻检索结果的比对，将查询返回的结果向量与谜底文件比对统计查询准确率.

表 1　实验数据集简述

数据集	描述子	维度	训练数据集大小	基础数据集大小	查询集数据集大小
ANN_SIFT10k	Sift	128	25 000	10 000	100
ANN_SIFT1M	Sift	128	100 000	1 000 000	10 000
ANN_GIST1M	Gist	960	500 000	1 000 000	1000

5.2　实验结果分析

5.2.1　查询准确率与查询时间

在信息检索领域，查询准确率、查全率和查询时间是衡量检索技术的重要指标，用更少的时间获取更高的检索准确率是几乎所有近邻检索算法的目标. 其中查询准确率定义为

$$\text{查询准确率}=\frac{\text{命中的近邻数据数目}}{\text{查询返回的数据数目}}.$$

本文查询准确度用 $R@X$ 表示，代表在重排序阶段输出的前 X 个返回结果中与谜底文件比对命中的最近邻向量占比. 查全率定义为

$$\text{查全率}=\frac{\text{命中的近邻数据数目}}{\text{谜底文件中近邻数据总数目}}.$$

在数据集 ANN_SIFT10K、ANN_SIFT1M 和 ANN_GIST1M 中，谜底文件中提供前 100 个最近邻向量的结果，所以 $R@100$ 表示查全率指标；查询时间顾名思义用于衡量查询响应速度.

首先将簇内乘积量化树方法与其他方法进行查询准确度和查询时间的对比. 如表 2～表 4 分别是簇内乘积量化树方法在数据集 ANN_SIFT10K、ANN_SIFT1M 和 ANN_GIST1M 上和其他相关量化方法的比对结果.

表 2　各近邻检索方法在数据集 ANN_SIFT10K 上的查询准确率与查询时间对比

方法	$R@1$	$R@10$	$R@100$	查询时间/ms
PQT_0	0.59	0.88	0.89	0.33
$C\text{-}PQT_0$	0.70	0.87	0.87	0.21
$C\text{-}PQT_2$	0.71	0.96	0.97	0.52

表 3　各近邻检索方法在数据集 ANN_SIFT1M 上的查询准确率与查询时间对比

方法	$R@1$	$R@10$	$R@100$	查询时间/ms
IVFADC	0.28	0.70	0.93	11.2
LOPQ	0.51	0.93	0.97	51.1
PQT_0	0.52	0.84	0.91	3.9
$C\text{-}PQT_0$	0.62	0.84	0.85	1.4
$C\text{-}PQT_1$	0.82	0.91	0.91	3.7
$C\text{-}PQT_2$	0.71	0.96	0.97	5.7

表 4　各近邻检索方法在数据集 ANN_GIST1M 上的查询准确率与查询时间对比

方法	$R@1$	$R@10$	查询时间/ms
PQT_1	0.306	0.533	15.3
$C\text{-}PQT_3$	0.578	0.724	16.7
$C\text{-}PQT_4$	0.665	0.905	34.6

如表 2 和表 3 所示，乘积量化树[17]案例 PQT_0 的配置为 $k_1=64, k_2=8, P=2, w_1=8$，即该树形结构第一层运用乘积量化，组别数目为 2，每组划分成 64 个子空间，剪枝参数为 8，每个子空间在第二层中运用向量量化细分，码本大小为 8. 选取前 500 个最近邻簇参与重排序即 $M=500$，并从这些候选向量中由近至远选不超过 20 000 个向量参与重排序与查询向量比对，即 L_C 的大小小于等于 20 000. 簇内乘积量化结构案例 $C\text{-}PQT_0$ 的参数配置为 $k_1=4, k_2=32, P=2, k_3=8, w_1=2, w_2=16$，近邻簇筛选出的大小为 $M=500$，L_C 的大小小于等于 20 000；$C\text{-}PQT_1$ 的配置同 $C\text{-}PQT_0$，但最近邻簇筛选出的大小为 $M=1000$；$C\text{-}PQT_2$ 的配置为 $k_1=8, k_2=32, P=2, k_3=1, w_1=1, w_2=4$，近邻簇大小和 L_C 的大小同 $C\text{-}PQT_0$. 从表 2 和表 3 得出，本文提出的簇内乘积量化树技术在 Sift 算子描述的数据集上获得了优越的近邻检索性能，与前人提出的经典算法相比，相同查询时间内可获得更高的查询准确率. 其中首次召回准确率 $R@1$ 相比乘积量化树技术提高了 57.7%，同局部优化乘积量化技术相比，查全率也可高达 97%，查询时间却仅需其 1/9.

表 4 给出的是不同方法在数据集 ANN_GIST1M 上的实验比对. 其中乘积量化树案例 PQT_1 的配置为 $k_1=8, k_2=4, P=2, w_1=4$，近邻簇大小和 L_C 的大小不变；其中簇内乘积量化结构案例 $C\text{-}PQT_3$ 的近邻簇和 L_C 的大小不变，结构参数配置为 $k_1=4, k_2=128, P=2, k_3=1, w_1=3, w_2=128$；$C\text{-}PQT_4$ 的结构参数配置同 $C\text{-}PQT_3$，近邻簇大小 $M=1000$，L_C 的大小小于等于 20 000. 可见在 Gist 算子描述下，本文提出的簇内乘积量化树技术同样具备更好的近邻检索性能.

纵向对比表 2～表 4 可得，在对查询时间和查询首次召回准确率要求高的情况下，可适量增大 k_3，即细化第 3 层向量量化的划分粒度；若对查询查全率指标要求高的情况下，可适当弱化或直接舍弃第 3 层的划分. 综上，本文提出的簇内乘积量化树技术与前人经典方法相比大大提高了近邻检索性能，并具有更为广泛的应用潜力. 虽然训练码本，建立索引表和数据库属于离线任务，不影响查询时间，

但是为了说明该方法的训练复杂程度，测试了在SIFT1M数据集下码训练本时间约为45 s，索引表和数据库构建时间约为286 s.

接下来为了说明簇内乘积量化树的索引结构的优越性，与其他方法进行空桶率对比.

5.2.2 空桶率比对

索引表空桶率可用于衡量索引结构的高效性，本文与乘积量化树技术作对比，用实验数据说明簇内乘积量化树结构设计的优越性. 如表5和表6分别是各近邻方法在数据集ANN_SIFT1M和ANN_GIST1M上生成索引表空桶率对比. 其中簇内乘积量化树结构下码本大小为 $k=k_1\cdot(k_2\cdot k_3)^P$；PQT码本大小为 $k=(k_2\cdot k_1)^P$. 实验结果说明簇内乘积量化树作为索引结构可更有效的拟合数据集的概率分布，从而生成更均衡的索引表. 与PQT相比索引表空桶率降低幅度在50%以上，而索引表的均衡性关系着近邻簇内是否包含足够的数据集向量参与重排序，所以均衡的索引表有利于提高查询准确率.

表5 ANN_SIFT1M上空桶率对比

方法	码本总数	非空桶数	空桶率/%
PQT_0	262144	92390	64.8
$C\text{-}PQT_0$	262144	161525	38.3
$C\text{-}PQT_2$	8192	8167	3.1

表6 ANN_GIST1M上空桶率对比

方法	码本总数	非空桶数	空桶率/%
PQT_1	1024	453	63.4
$C\text{-}PQT_3$	65536	40108	38.8

5.2.3 面量化粒度对近邻检索的影响

实验中我们发现，重排序过程中执行面量化距离近似方法时，可将子空间进一步划分，然后在更低维度的子空间进行面量化近似求解. 该细化的粒度称作面量化粒度，设为 P_{plane}，P_{plane} 对查询准确率和查询时间均有极大影响，本文通过实验说明面量化粒度带来的影响. 如图6所示，数据集选取的是ANN_SIFT1M，簇内乘积量化树的配置为 $k_1=4$，$k_2=32$，$P=2$，$k_3=8$，$w_1=2$，$w_2=16$，近邻簇大小为500，候选集合规模小于等于20000，可见查询准确度会随着 P_{plane} 的增大而上升，查全率不受量化粒度的影响. 如表7所示是不同量化粒度下量化器引入的平均量化误差以及对应的查询时间比对，显然量化粒度增大，平均量化量化误差减小但计算复杂度增加，查询时间随之增大，如何在查询准确度和查询速度之间做一个合适的权衡，需要根据具体应用要求进行取舍，选择 $P_{\text{plane}}=16$ 可在一定的精确度损失范围内适当减少计算和存储的开销.

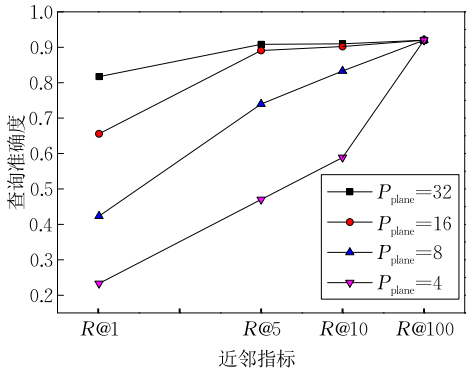


图6 查询准确度与量化粒度的关系

表7 量化粒度与量化误差及查询时间的关系

P_{plane}	最小量化误差	最大量化误差	平均量化误差	查询时间/ms
4	1297.441	137157.11	24667.010	1.90
8	833.257	90986.01	11713.250	2.10
16	350.316	47129.23	2744.910	2.57
32	21.110	7798.33	473.386	3.73

5.2.4 距离近似引入的量化误差对比

如表8所示，将本文提出的面量化近似距离估算方法与点量化中非对称距离估算方法和线量化估算方法作比对，表中所用的簇内乘积量化案例是 $C\text{-}PQT_3$，数据集选取的是ANN_SIFT1M. 可见同样场景下，应用面量化距离估似方法引入的平均量化误差仅为点量化的1/59，线量化的1/6. 可见面量化大大减小了向量近似引入的量化误差，提高了重排准确性，为近邻查询准确率提供了保障.

表8 各类量化方法量化误差比对

量化方法	最小误差	最大误差	平均误差
点量化[10]	1845.09	142857.00	27884.700
线量化[17]	64.20	31378.80	2759.040
面量化	6.52	9799.33	473.386

5.2.5 投影次数对查询准确度的影响

如表9所示，使用不同的投影次数去近似确定面投影点，对比它们的查询准确度. 数据集选取的是ANN_SIFT1M，簇内乘积量化树的配置均为 $k_1=4$，$k_2=16$，$P=2$，$k_3=8$，$w_1=1$，$w_2=4$，近邻簇大小为500，候选集合规模小于等于20000. 从表中可以看出投影次数越多，查询准确度越高. 由于不管投影多少次，查询时只会选择最后两次的投影结果用于计算距离，所以增加投影次数对查询时间没有影响，但是更多的投影次数可以降低投影误差，所以准确度有所提升.

表 9　投影次数对查询准确度影响

投影次数	$R@1$	$R@10$	查询时间/ms
3	0.710	0.846	5.8
4	0.711	0.846	5.8

6　结　论

作为大规模高维数据管理的关键技术，近似最近邻检索技术面临诸多难点.例如建立高效索引结构完成对原始大规模数据集的降维处理时，如何均衡有效地建立索引表以拟合原始数据集概率分布；面对维数灾难时，如何高效计算高维向量间的距离等.本文针对这些现有问题，提出簇内乘积量化树方法，该方法运用基于向量量化和乘积量化的多层树状检索结构高效表征高维数据集.实验表明该方法应用在大规模数据集上可生成更均衡的索引表，空桶率得到显著降低；为了加快近邻簇生成速度，本文提出贪心队列近邻簇筛选方法，可快速为查询向量筛选出足够的近邻簇；为了有效估算高维向量间距离，本文设计了一种新的距离近似方法—面量化距离估算方法，经实验表明该方法具备更低的量化误差，从而更准确的重构高维数据，提高查询准确率.面对更大规模的数据，可以通过 GPU 实现并行，由于各个基础向量的建立索引和量化的过程都是相互独立的，各个查询向量的查询过程也是相互独立的，所以该方法很适合并行.

参　考　文　献

[1] Sivic J, Zisserman A. Video Google: A text retrieval approach to object matching in videos//Proceedings of the 9th International Conference on Computer Vision. Nice, France, 2003: 1470-1478

[2] Jégou H, Tavenard R, Douze M, et al. Searching in one billion vectors: Re-rank with source coding//Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing. Prague, Czech Republic, 2011: 861-864

[3] Zhao Qi-Lu, Li Zong-Min. Cross-modal social image clustering. Chinese Journal of Computers, 2018, 41(1): 98-111(in Chinese)
(赵其鲁，李宗民. 跨模态社交图像聚类. 计算机学报，2018，41(1)：98-111)

[4] Shakhnarovich G, Darrell T, Indyk P. Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing). Cambridge, USA: The MIT Press, 2006

[5] Muja M, Lowe D G. Scalable nearest neighbor algorithms for high dimensional data. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2014, 36(11): 2227-2240

[6] Zhu Li, Qiu Yuan-Yuan, Yu Shuai, Yuan Sheng. A fast kNN-based MST outlier detection method. Chinese Journal of Computers, 2017, 40(12): 2856-2870(in Chinese)
(朱利，邱媛媛，于帅，原盛. 一种基于快速 k-近邻的最小生成树离群检测方法. 计算机学报，2017，40(12)：2856-2870)

[7] Beyer K, Goldstein J, Ramakrishnan R, et al. When is "nearest neighbor" meaningful//Proceedings of the International Conference on Database Theory. Jerusalem, Israel, 1999: 217-235

[8] Böhm C, Berchtold S, Keim D A. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. ACM Computing Surveys, 2001, 33(3): 322-373

[9] Friedman J H, Bentley J L, Finkel R A. An algorithm for finding best matches in logarithmic expected time. ACM Transactions on Mathematical Software, 1977, 3(3): 209-226

[10] Jégou H, Douze M, Schmid C. Product quantization for nearest neighbor search. IEEE Transactions on Pattern Analysis and Machine Inteligence, 2011, 33(1): 117-128

[11] Datar M, Immorlica N, Indyk P, et al. Locality-sensitive hashing scheme based on p-stable distributions//Proceedings of the 20th Annual Symposium on Computational Geometry. Brooklyn, USA, 2004: 253-262

[12] Andoni A, Indyk P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions//Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science. Berkeley, USA, 2006: 459-468

[13] Babenko A, Lempitsky V. Tree quantization for large-scale similarity search and classification//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Boston, USA, 2015: 4240-4248

[14] Muja M, Lowe D G. Scalable nearest neighbor algorithms for high dimensional data. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2014, 36(11): 2227-2240

[15] Babenko A, Lempitsky V. Additive quantization for extreme vector compression//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Columbus, USA, 2014: 931-938

[16] Kalantidis Y, Avrithis Y. Locally optimized product quantization for approximate nearest neighbor search//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Columbus, USA, 2014: 2321-2328

[17] Wieschollek P, Wang O, Sorkine-Hornung A, et al. Efficient large-scale approximate nearest neighbor search on the GPU//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas, USA, 2016: 2027-2035

[18] Linde Y, Buzo A, Gray R. An algorithm for vector quantizer design. IEEE Transactions on Communications, 1980, 28(1): 84-95

[19] Gray R M, Neuhoff D L. Quantization. IEEE Transactions on Information Theory, 1998, 44(6): 2325-2383
[20] Mertens S. The easiest hard problem: Number partitioning. Computational Complexity and Statistical Physics, 2006, 125(2): 125-139
[21] Lloyd S. Least squares quantization in PCM. IEEE Transactions on Information Theory, 1982, 28(2): 129-137
[22] Hartigan J A, Wong M A. Algorithm AS 136: A *k*-means clustering algorithm. Journal of the Royal Statistical Society, Series C(Applied Statistics), 1979, 28(1): 100-108
[23] Babenko A, Lempitsky V. The inverted multi-index// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Providence, USA, 2012: 3069-3076
[24] Ge T, He K, Ke Q, et al. Optimized product quantization for approximate nearest neighbor search//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Portland, USA, 2013: 2946-2953

LIU Shu-Wei, M. S. candidate. Her current research interests include data mining and machine learning.

CHEN Wei, Ph. D. candidate. His research interests include computer vision and pattern recognition.

ZHAO Wei, M. S. candidate. His research interest is machine learning.

CHEN Jin-Cai, Ph. D., professor, Ph. D. supervisor. His research interests include data storage and machine learning.

LU Ping, M. S., associate professor. Her research interests include data storage and machine learning.

Background

The effective management of large-scale and high-dimensional data has become a key research for many applications. Nearest Neighbor Search is one of the hot issues in it.

Existing methods can be divided into two categories. The first category is hash-based approaches, such as Locality Sensitivity Hashing. The accuracy and search speed of this kind of methods depend on the choice of the hash function. Due to the difficulty in hash function design, these methods are not optimistic on large-scale and high-dimensional datasets. The second category is based on the vector and product quantization. Compared with the hash-based ANN algorithm, it can be more intuitive to spatially partition the dataset. Current classical nearest neighbor search methods based on vector quantilization have good performance on large scale and high dimensional datasets, but these methods still have the following two problems: (1) the index structure can not fit the underlying distribution of the dataset well, which makes the empty bucket rate of index table higher; (2) the quantization error introduced by the re-ranking method is relatively large, which makes the distance estimation between the high dimensional vectors greatly distorted. In order to solve these problems, we present an efficient approach extending the idea of Vector Quantilization and Product Quantilization. We propose a three level indexing structure that can immensely reduce the times of exact distance calculations between high dimension vectors and produce a more compact and balanced index table. Our approach also includes an improved re-ranking method called plane quantilization approaching algorithm inlighted by line quantilization. This Clustered Product Quantilization Tree approach significantly outperforms recent state of the art methods for high dimensional nearest neighbor queries on standard reference datasets like ANN_SIFT1M and ANN_GIST1M.

This work is supported by the National Natural Science Foundation of China under Grant Nos. 61672246, 61272068, and the Fundamental Research Funds for Central Universities under Grant No. HUST:2016YXMS018.