软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2022,33(4):1200−1217 [doi: 10.13328/j.cnki.jos.006463]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

开放环境多分布特性的局部敏感哈希检索方法*

张 仕[1,2], 赖会霞[1], 肖如良[1,2,3], 潘淼鑫[1], 张路路[1], 陈伟林[1]

[1](福建师范大学 计算机与网络空间安全学院, 福建 福州 350117)

[2](数字福建环境监测物联网实验室(福建师范大学), 福建 福州 350117)

[3](福建省网络安全与密码技术重点实验室(福建师范大学), 福建 福州 350117)

通信作者: 肖如良, E-mail: xiaoruliang@fjnu.edu.cn

摘 要: 基于局部敏感哈希的检索方法能够较好地解决高维大规模数据的近似近邻检索问题. 但在开放环境下针对多种分布特性时, 迄今尚未有令人满意的解决方案. 利用 Laplacian 算子对数据分布剧烈变化敏感的特性, 提出一种具有全局性、适用于开放环境下多种分布特性的基于 Laplacian 算子的局部敏感哈希搜索方法(LPLSH). 该方法把 Laplacian 算子应用于数据投影的概率密度分布, 找到数据投影分布的剧烈变化位置作为超平面的偏移量.从理论上证明了精简维度的哈希函数能够保持局部敏感性及低投影密度区间分割的有效性, 分析了利用 Laplacian 算子计算的二阶导数对超平面偏移量设置的指导意义. 与其他 8 种方法对比, LPLSH 算法的 *F*1 值是其他方法最优值的 0.8 倍–5 倍, 耗费时间也大幅减少. 通过对具有多种分布特性数据集上的实验验证, 结果表明: LPLSH 方法能够同时兼顾效率、精度和召回率, 可满足开放环境下多分布特性的大规模高维检索的鲁棒性需求.

关键词: 开放环境; 近似近邻检索; 数据多分布特性; 局部敏感哈希; 数据检索

中图法分类号: TP311

中文引用格式: 张仕, 赖会霞, 肖如良, 潘淼鑫, 张路路, 陈伟林. 开放环境多分布特性的局部敏感哈希检索方法. 软件学报, 2022, 33(4): 1200–1217. http://www.jos.org.cn/1000-9825/6463.htm

英文引用格式: Zhang S, Lai HX, Xiao RL, Pan MX, Zhang LL, Chen WL. Open Environmental Locality-sensitive Hashing Retrieval for Multiple Distributed Characteristics. Ruan Jian Xue Bao/Journal of Software, 2022, 33(4): 1200–1217 (in Chinese). http://www.jos. org.cn/1000-9825/6463.htm

Open Environmental Locality-sensitive Hashing Retrieval for Multiple Distributed Characteristics

ZHANG Shi[1,2], LAI Hui-Xia[1], XIAO Ru-Liang[1,2,3], PAN Miao-Xin[1], ZHANG Lu-Lu[1], CHEN Wei-Lin[1]

[1](College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350117, China)

[2](Digital Fujian Internet-of-Things Laboratory of Environmental Monitoring (Fujian Normal University), Fuzhou 350117, China)

[3](Fujian Provincial Key Laboratory of Network Security and Cryptology (Fujian Normal University), Fuzhou 350117, China)

Abstract: The retrieval methods based-on locality-sensitive hashing (LSH) provide a feasible solution to the problem of approximate nearest neighbor (ANN) search on high-dimensional, multiple distributed characteristics, and massive data. However, there are still some unresolved problems in open environment, such as poor adaptability to the data with multiple distribution characteristics. Based on the fact that Laplacian operator is sensitive to sharp changes in data, an LSH retrieval method based on Laplacian operator (LPLSH) is proposed, which is suitable for data in open environment with a variety of distributed characteristics, and can segment data on global view. By applying Laplacian operator to the probability density distribution of data projection, the position of the sharp change of distribution will found as the offset of the hyperplane. This study proves theoretically that the reduced dimension can keep the local sensitivity characteristics of the hash function, and the global low projection density interval segmentation is helpful to improve the precision. The guiding significance of using Laplacian operator to obtain the second derivative to set the hyperplane offset is also analyzed. Compared

* 基金项目: 国家自然科学基金(61772004); 福建省科技重大项目(2020H6011); 福建省自然科学基金(2020J01161)
本文由“面向开放场景的鲁棒机器学习”专刊特约编辑陈恩红教授、李宇峰副教授、邹权教授推荐.
收稿时间: 2021-02-18; 修改时间: 2021-07-16; 采用时间: 2021-08-27; jos 在线出版时间: 2021-10-26

with the other 8 methods based on LSH, the $F1$ value of LPLSH is 0.8−5 times of the optimal value of other methods, and it takes less time. Through the analysis of the distribution characteristics of experimental datasets, the experimental results show that LPLSH can take into account the efficiency, accuracy, and recall rate at the same time, can meet the robustness requirements of large-scale high-dimensional retrieval with multi-distribution characteristics in open environment.

Key words: open environment; nearest neighbor search; data multiple distributed characteristics; locality-sensitive hashing; data retrieval

近似近邻(approximate nearest neighbor, ANN)检索被广泛应用于机器学习和数据挖掘相关领域, 如基于KNN 的数据分类[1]、推荐系统[2]和信息检索[3]等. 随着 5G、传感网络和大数据技术的发展, 大规模高维数据的采集和利用成为现实. 这使得开放环境下高维海量数据的 ANN 检索相比于任何其他时候都更是一项具有高度挑战性的工作. 在开放应用环境中, 开放场景的不可预测性(设备、传输故障、微环境变化等)使 ANN 检索出现了新的鲁棒性检索问题, 主要表现在各类噪音数据、异常数据对检索模型具有不可预测的影响, 以及不同维度数据分布的多样性弱化了现有模型适应性. 例如: 在气象环境监测网络中, 其采集的数据种类多样, 各种采集点数据的值域范围、变化规律各不相同, 从而形成各维度具有不同分布特性的高维海量数据环境. 在实现基于相似气象环境的气象预报和灾害预测时, 现有 ANN 检索方法无法提供有效支持. 因此, 构建一个适用于开放环境、具有良好性能的大规模高维数据检索方法, 将对具有高维、多分布特性、海量数据的开放应用有非常重要的意义.

在低维空间中, 近似近邻检索问题已经得到了较好的解决, 如 K-D 树、R 树、SR 树等[4−6]方法提供了一些有效的解决方案. 但是在高维海量数据空间中, 近似近邻查找会导致查询时间和空间消耗呈指数式增长, 使现有低维度数据检索方法无法满足数据检索要求. 针对高维海量数据, Indyk 等学者开创性地提出基于局部敏感哈希(locality-sensitive hashing, LSH)的近似近邻检索方法[7,8]. 该方法利用哈希函数把高维数据转换为二进制序列, 实现近似近邻数据的快速检索. 此后, 许多学者进一步完善, 如 Charikar 提出的基于随机超平面投影的 LSH 检索方法 RHP[9]、Datar 和 Indyk 提出的 E2LSH 增强了方法的可用性[10], 并对后来的研究工作产生了巨大的影响. 这些工作中, 最具有代表性的有 PCAH[11], KLSH[12], SBLSH[13], ITQ[14], DSH[15], OCH[16], GLDH[17]等. 随着深度学习的兴起, 近年来又出现了一些结合深度学习和 LSH 的检索方法[18−21]. 但是, 现有基于局部敏感哈希的检索方法仍然存在不能有效适应开放环境下具有多分布特性等问题, 这些问题具体体现在如下 3 个方面.

(1) 难以适应开放环境下数据分布的多样性: 多数 LSH 相关算法只适用于具有特定分布特性的数据, 适应性弱, 从而制约了基于局部敏感哈希检索方法在开放环境下的应用;

(2) 空间划分难以全局考虑: 部分算法在空间划分上有较大的误差, 例如 PCAH[11]和 RHP[9]方法有较大的切分误差, DSH[15]在一定程度上减少了切分误差, 但其解决方案缺少全局角度的考虑;

(3) 性能与精度需进一步提高: 基于深度学习的 LSH 检索算法较长的预处理时间限制了其进一步应用, 而传统的基于 LSH 的检索算法虽然在性能上占优, 但其精度仍旧参差不齐, 有待进一步提高.

针对以上问题, 本文利用 Laplacian 算子具有对数据分布急剧变化敏感的特性, 提出一种基于 Laplacian 算子的局部敏感哈希数据检索方法(LPLSH), 该方法针对多种数据分布特性进行有效的哈希函数构造, 从数据的全局视角出发, 增强了算法对开放环境多种数据分布的适应性, 进一步提高了算法的精度和效率. 本文主要贡献包含如下 4 点.

(1) 提出一种适应开放环境下多分布特性的 LPLSH, 从全局角度出发, 利用 Laplacian 算子对数据分布变化的敏感性发现数据分布的边缘, 以确定随机超平面的偏移量, 从而形成有效的分割;

(2) 证明了精简维度的哈希函数仍具有局部敏感性及低投影密度区间分割的有效性, 分析了利用 Laplacian 算子求得二阶导数对超平面偏移量设置的指导意义;

(3) 实验验证了本文方法对开放环境应用的适应性: 本文的 LPLSH 方法无需调整参数, 能够在具有不同分布特性数据集上取得高精度和良好召回率的平衡, 在效率上展现出明显优势;

(4) 通过对实验数据的分布特征分析, 结果表明: LPLSH 能够适用于多种分布特征的数据, 显著提高了各维度上分布形式单一且没有明显分类特性数据的检索效果, 增强了开放环境的适应性.

本文第 1 节介绍相关工作. 第 2 节介绍相关的基础知识. 第 3 节详细叙述所提出的适应开放环境多分布特性的基于 Laplacian 算子的局部敏感哈希检索方法. 第 4 节对实验和实验结果进行分析. 最后, 第 5 节总结全文.

1 相关工作

给定具有 n 个元素的数据集, 采用逐个对比的方式查找元素 q 的最近邻元素的时间复杂度为 O(n). 然而, 当所在数据集具有海量、高维特性时, 其查询将会造成很高的时间开销. 退而求其次, 人们利用近似近邻元素(approximate nearest neighbor, ANN)替代最近邻数据元素, 以实现查询准确性与效率的折衷[22].

目前, 现有 ANN 检索方法大多从降低数据维度的角度出发, 通过较短的数据索引编码实现快速近似近邻搜索. 其索引结构的构建有基于哈希的索引结构构建方法[9]和基于乘积量化的索引结构构建方法[23]: 前者以局部敏感哈希算法为代表[7,8], 通过把相似的点映射到相同的索引编码中来实现近似近邻搜索; 后者通过合理地分割数据, 构建基于数据簇的索引结构.

本文主要对基于局部敏感哈希的近似近邻检索方法相关工作进行综述.

1.1 数据无关的局部敏感哈希检索研究

Indyk 等人早在 1998 年提出 LSH 方法[7]后, Gionis 在文献[8]中对其进行完善, 根据 l_1-*norm* 和 l_2- *norm* 距离公式的相似性, 把原始数据转换到 Hamming 空间, 但是 l_1-*norm* 和 Hamming 空间的要求限制了 LSH 方法的进一步应用. 虽然 Indyk 和 Gionis 开创了一种大规模高维数据集上近似近邻元素查找的新方法, 但是该方法通常需要设置几十上百的哈希表才能获得查找的高准确性, 因此难以直接推广使用.

2002 年, Charikar 提出了基于随机超平面投影的 LSH 查找方法(RHP)[9], 其随机投影理论依据来源于 J-L 引理. 该方法随机生成 k 个超平面, 每个超平面把点集分为两部分, 从而确定编码中对应位为 1 或 0. 查询时, 按照相同的方法构造待查询数据的哈希编码, 并根据编码查找对应的哈希桶. RHP 不需要额外的参数调试, 因此在实际应用中表现出了更好的稳定性, 但是该方法的精度需要较长的哈希编码以及众多的哈希表.

2004 年, 文献[10]中提出了基于 p 稳定分布的局部敏感哈希算法 E2LSH. 该方法利用 p-稳定分布性质, 首先生成 k 组 d 维符合正态分布的随机向量 a, 对每个元素 v 计算$[a\cdot v/w]$, 从而得到一组长度为 k 的整数向量. E2LSH 把距离计算方法从 l_1-*norm* 扩展到了 l_2-*norm*, 加快了查询的效率, 且更加容易实现.但是该方法强烈依赖于 w 值的设置, 对空间的划分过于细致, 从而生成了太多的无效空间, 限制了算法效率的提高.

2012 年, Ji 等人提出了超位局部敏感哈希算法 SBLSH[13], 该算法以角度作为相似度度量标准, 对随机投影向量进行分组正交化, 从理论上证明了经过分组正交化的 Hamming 距离的方差比局部敏感哈希的小, 但是该方法所得到的哈希编码不稳定、精确度依赖于编码长度.

1.2 数据依赖的局部敏感哈希检索研究

研究人员做了大量基于 LSH 进行 ANN 检索的改进, 其中之一便是研究数据依赖的局部敏感哈希方法, 这类方法根据数据的特性生成优化的局部敏感哈希函数, 进而得到更好的检索效果.

主成分分析(PCA)不但可以进行降维, 还可以克服图像的旋转问题, 常被用于图像的哈希检索. 结合 PCA 与 LSH, 微软亚洲研究院的 Xin-Jing 提出了基于主成分的局部敏感哈希方法 PCAH[11], 但该方法对各方向利用相同的编码位数, 没有考虑到具体投影方向的数据范围大小; 之后, Spectral Hashing (SPH)[24]给主成分方向具有较大范围的情况赋予更多编码位, 该方法假设数据是均匀分布的, 没有考虑到具体数据的特征; 之后的 ITQ[14]方法通过找到主成分方向, 通过旋转避开数据聚类的中心位置, 以提高算法的精确度. 但是旋转使得超平面偏离主成分方向, 也就弱化了 PCA 的应用, 难以实现在具有较大范围的 PCA 方向上的多位编码实现.

K-means 方法用于数据聚类, 以形成 K 个数据簇. 部分方法利用聚类, 以期构造介于簇间的超平面减少超

平面对密度簇直接分割来提高查询的精度. 例如: Jin 等人提出了基于密度的局部敏感哈希 DSH[15], 该方法利用 *K*-means 聚类形成的簇构造超平面, 并利用熵衡量超平面的优劣; Xiao 等人提出的 GLDH[17]通过利用最小割的方法衡量不同超平面, 不过, 该方法容易导致数据子空间的数据分布不均衡, 从而影响查询精度. DSH 方法只考虑到了超平面分割数据的平衡, 并未从全局角度考虑避免超平面延伸对其他簇中心的切割; GLDH 方法虽然利用割的概念从全局角度避免对数据密集区域的切割, 但是难以应付各维数据分布相似的情况.

核方法常被用于标签数据的分类, 通过对数据的维度变换, 从而形成更多超平面分割选择. Kulis 等人[12]将核方法及 Laplacian 矩阵引入检索结构中, 运用中心极限定理构造多个超平面, 在核空间内对数据进行投影, 再运用余弦相似性度量, 提出了基于核空间的 KLSH. 该方法通过对数据集内部特征的挖掘, 用额外的训练时间换取了检索精度的提高, 同时增强了算法对不同数据分布的适应性. Lechervy 等人[25]则提出利用 Boosting 整合多核函数, 并应用于基于内容的多媒体数据分类和检索中. 不过, 其方法构造过程依赖于已有数据分类. 对于核+LSH 的方法, 其主要问题在于核的选择与数据的适应性上, 且基于核的方法通常需要花费更多时间在数据的预处理和训练上.

此外, Huang 等人提出了 LH (Laplacian hashing)[26], 该方法在计算中利用图拉普拉斯矩阵(graph Laplacian matrix)表示数据的关联关系, 最终通过求得最小 k 个特征值对应的特征向量作为分割超平面. 该方法更适用于带标签数据的分类查找. Zhang 等人提出一种综合性方法(HEHC-LSH)[27], 实现近邻数据的快速查找.

1.3 基于深度学习的局部敏感哈希检索研究

近 3 年来, 基于 LSH 的近似近邻检索领域出现了一些基于深度神经网络的研究工作, 例如: 2017 年, Cao 提出了针对多媒体数据的 HashNet 方法[18]; 2020 年, Dong 提出算法框架 Neural LSH, 通过先聚类再利用神经网络进行近似近邻簇的查找[19]; 2019 年, Chiu 采用神经网络估计近似近邻关系提出的基于近似近邻概率, 而非质心距离的模型排序聚类方法[20]; Sapkota 于 2019 年提出了 DCH 方法[21], 通过深度学习的方式学习图像中的语义信息, 并借助神经网络生成包含语义信息的二维编码.

基于深度神经网络生成哈希编码的方法虽然在检索精度上具有一定的优势, 但其在开放环境下大规模数据集上的训练时间较长, 内存占用大, 对训练数据依赖较强, 在实际开放应用时依然存在一定障碍.

2 基础知识

本节介绍论文的相关基础, 包括核密度估计、Laplacian 算子及基于随机投影的 LSH 数据检索.

2.1 核密度估计

核密度估计(kernel density estimation, KDE)是一种随机变量概率密度函数的非参估计方法, 该方法无需利用数据分布的先验知识, 对数据分布不做任何假定, 是一种从数据样本出发研究数据分布特征的方法.

假定样本数据集 $X=\{x_1,\ldots,x_M\}$包含 M 个元素. 求位置 x 的密度函数值时, 只需要取 x 的邻域$[x-h,x+h]$, 当 $h\to 0$ 时, 把该邻域的密度值当成 x 点的密度函数值, 如公式(1):

$$\hat{f}_N(x)=\frac{1}{2h}\lim_{h\to 0}\frac{N_x}{M} \tag{1}$$

其中, N_x 表示 x 邻域中的样本数量. 由于 h 设置以及有限采样数据的原因, 该方法求出的概率密度不够平滑. 把核函数引入密度估计, 不但可以保证概率密度函数的积分为 1, 同时, 密度函数也变得连续可导, x 处的概率密度值可由公式(2)的核密度估计函数计算[28]:

$$\hat{f}_N(x)=\frac{1}{M}\sum_{i=1}^{M}K_h\left(\frac{|x-x_i|}{h}\right)=\frac{1}{h^dM}\sum_{i=1}^{M}K\left[\frac{|x-x_i|}{h}\right] \tag{2}$$

其中, $K(x)$是核函数, d 是数据的维度, h 是核函数带宽. 核函数带宽表示核密度估计函数的平滑程度, 带宽越大越平滑. 在核密度估计中, 可用的核函数有许多, 其中, 高斯核具有良好的平滑密度估计性质, 是最常用的核函数. 高斯核定义如公式(3):

$$K\left(\frac{|x-x_i|}{h}\right)=\frac{1}{(2\pi)^{d/2}}\exp\left(-\frac{\|x-x_i\|^2}{2h^2}\right) \tag{3}$$

把高斯核带入核密度函数，可以得到高斯核密度函数，如公式(4):

$$\hat{f}_N(x)=\frac{1}{hN}\sum_{i=1}^{M}\frac{1}{(2\pi)^{d/2}}\exp\left(-\frac{\|x-x_i\|^2}{2h^2}\right) \tag{4}$$

2.2 Laplacian算子

Laplacian 算子是 n 维欧几里德空间中的二阶微分算子，其对数据变化具有灵敏的感知能力. 在信号及图像处理的应用中，常被用于进行异常检测、边缘定位、图像锐化等.

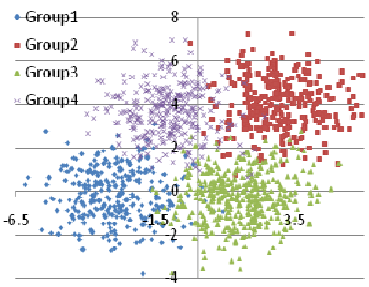
若 f 是二阶可微的实函数，则 f 的 Laplacian 算子定义如公式(5):

$$\Delta f=\nabla^2 f=\nabla\cdot\nabla f \tag{5}$$

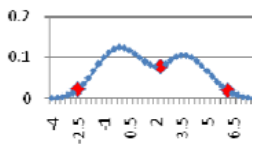
其中,

$$\nabla f(x)=\lim_{\Delta x\to 0}\frac{\Delta y}{\Delta x}=\lim_{\Delta x\to 0}\frac{f(x+\Delta x)-f(x)}{\Delta x} \tag{6}$$

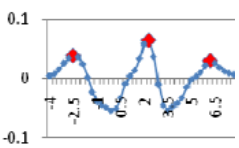
把该算子应用于数据投影的概率密度函数，可以准确发现数据投影密度的变化趋势，感知数据密度剧烈变化位置. 在开放环境下，数据通常是多种分布数据的组合，例如: 图 1(a)是由 4 个无明显界限且满足正态分布的数据簇组成的数据集，该数据集没有明显可见的边界. 把该数据集投影到随机方向 1 上，其高斯核密度如图 1(b)所示; 再把 Laplacian 算子作用于概率密度函数，可得二阶导数值如图 1(c)所示. 从投影的高斯核密度计算结果(如图 1(b)所示)可以看出，Laplacian 算子计算结果的极大值位置也是该投影方向上数据的较为理想的分割位置(如图 1(b)、图 1(c)所示). 对随机方向 2 的投影如图 1(d)所示，此时，数据投影的高斯核密度虽然没有明显的两个波峰及波谷，但通过 Laplacian 算子计算结果，可以看出明显的边缘位置以及两个正态分布的边界(如图 1(e)所示).



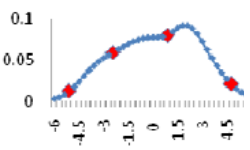
(a) 4 个满足正态分布的数据簇组成的数据集



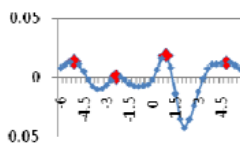
(b) 投影 1 上的概率密度



(c) 投影 1 上的概率密度二阶导数



(d) 投影 2 上的概率密度



(e) 投影 2 上的概率密度二阶导数

图 1　模拟数据及其在两个随机方向上的概率密度估计与二阶导数

这个简单的数据集实验给我们启发: 把 Laplacian 算子应用于数据的投影密度，可以辅助界定对应超平面的偏移量，避开数据集中区域，使得超平面的数据分割具有更好的数据区分能力.

2.3 基于随机投影的LSH方法

定义 1(局部敏感哈希的函数簇). 若 $D(\cdot,\cdot)$是一个计算集合 X 中两元素距离的函数，$r_1<r_2$，$p_1>p_2$，$\forall x,y\in X$,

$\forall h \in H$ 满足如下性质.

1) if $D(x,y) \leqslant r_1$, then $Pr_R[h(x)=h(y)] \geqslant p_1$;

2) if $D(x,y) \geqslant r_2$, then $Pr_R[h(x)=h(y)] \leqslant p_2$.

则 $H=\{h_1,h_2,\ldots\}$是局部敏感哈希的函数簇.

定义 2(LSH 算法). 给定包含 n 个数据的集合 $X \in R^d(|X|=n)$, LSH 算法是指从满足局部敏感哈希的函数簇中选取 k 个函数组成函数集 H, $H=\{h_1,h_2,\ldots,h_k\}$, 并把所有 X 中的数据点 x 利用 H 映射为长度为 k 的哈希编码 $H(x)=\{h_1(x),h_2(x),\ldots,h_k(x)\}$的过程.

定义 3(基于随机投影的哈希函数). 设 $\vec{r}$ 是一个满足高斯分布的随机向量, $\vec{x}$ 表示待编码的数据向量, 那么基于随机投影的哈希函数表示如公式(7):

$$h_{\vec{r}}(\vec{x}) = \begin{cases} 1, & \text{if } \vec{r} \cdot \vec{x} \geqslant 0 \\ 0, & \text{if } \vec{r} \cdot \vec{x} < 0 \end{cases} \tag{7}$$

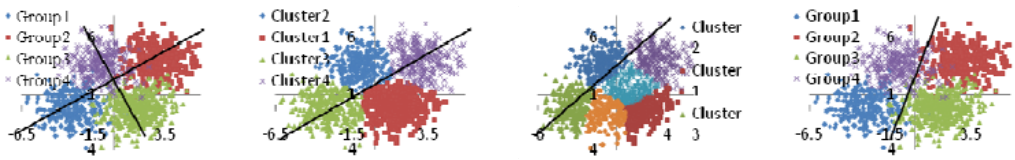
随机投影的方法利用 k 个哈希函数把数据映射到一个子空间中, 在该子空间中的所有点具有相同的哈希编码. 由 LSH 特性可知: 越相似的点, 也就有越大的可能性被投入相同的子空间中.

3 开放环境多分布特性的基于 Laplacian 算子的局部敏感哈希检索方法

海量高维及其各维度呈现不同分布特性, 是开放环境数据的一个基本特征. 本节详细描述了一种基于 Laplacian 算子、可满足开放环境具有多分布特性的海量高维数据的检索方法(LPLSH). 为了能从更加直观的角度理解 LPLSH, 我们先以 PCAH 和 DSH 在 4 个正态分布合成数据集(如图 1(a)所示)上的平面划分为例, 说明 LPLSH 能从更加宏观的角度实现对超平面偏移量的设置, 具有更好的应用鲁棒性.

针对该合成数据, PCAH 方法利用数据主成分方向作为投影矢量, 其所构造的超平面只考虑了方向, 没有考虑如何合理设置偏移量, 从而导致两个簇被从中间分割(如图 2(a)所示). DSH[15]方法以熵为依据, 从聚类间的中垂面中选择划分平面, 但没有考虑超平面延伸对其他聚类的影响; 图 2(b)是 DSH 方法在聚类 k=4 时的情况, 其具有最大熵的中垂面导致了另外两个聚类被切分; 图 2(c)是 DSH 在 k=6 时所生成的聚类, 聚类不能正确反映实际簇, 其所生成的分割平面也无法避免实际簇被分割.

对同样的数据集, 我们把数据投影到一个随机选择的法向量上, 通过 Laplacian 算子计算数据投影概率密度的二阶导, 从而得到超平面的三个可选偏移量, 如图 1(c)所示; 综合考虑密度分布和二阶导数值, 可以得到其分割平面如图 2(d)所示. 虽然该超平面无法完全避免数据簇的误分割, 但是其分割位置能够从总体上最小化对数据簇的划分. 当投影没有明显的波谷时, LPLSH 仍能发现复合分布边界, 从而得到良好的分割位置(如图 1(d)所示). 这也说明了 LPLSH 能够适应多种分布特性数据, 具有良好的鲁棒性.



(a) PCAH 分割实例 (b) DSH(k=4)的分割实例 (c) DSH(k=6)的分割实例 (d) LPLSH 分割实例

图 2 PCAH, DSH 及 LPLSH 在模拟数据集上的分割实例

3.1 开放环境下多分布特性的LPLSH大规模高维数据检索系统框架

开放环境下多分布特性的 LPLSH 检索系统的总体框架如图 3 所示. 从图中可以看出, 该系统一共包含 3 个部分: 哈希函数构造、数据存储和数据查询. 其中,

(1) 哈希函数构造过程包含生成 k 个哈希函数, 每个哈希函数的生成由两个步骤组成: (a) 符合高斯分布的随机向量生成; (b) 把数据投影到随机向量上, 根据投影的高斯核概率密度分布和 Laplacian 算子求得的投影二阶导数确定偏移量;

(2) 数据存储过程中，逐个计算所要存储数据的哈希编码，并把数据 ID 存入对应编码的哈希桶;

(3) 数据查询阶段，用哈希函数计算查询对象的哈希编码，根据哈希编码定位哈希桶得到候选数据集，再逐个计算出候选数据集与查询对象的实际距离，排序并返回查询结果.

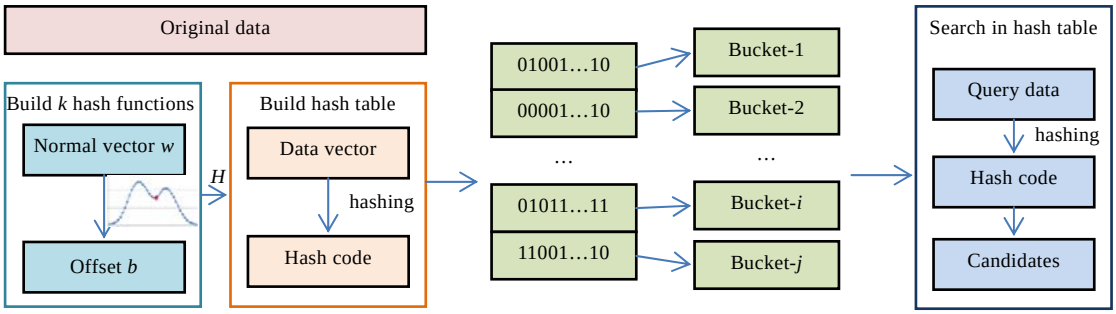


图 3 LPLSH 数据检索总体框架

本节剩余部分将对哈希函数构造进行展开，详细叙述 LPLSH 方法的法向量和偏移量生成过程.

3.2 LPLSH的哈希函数构造

假定所要处理的数据集 $X=\{x_i\in\{R\}^d|i=1,\ldots,N\}\subset\{R\}^d$, $|X|=n$, 且 $x_i=(x_i^{(1)},x_i^{(2)},\ldots,x_i^{(d)})$，数据维度为 d. 在不引起混淆的情况下，X 也表示 $n\times d$ 矩阵，每行 1 个数据元素.

向量夹角的余弦值可以利用基于随机投影的技术进行衡量[29]，Charikar 直接把该技术应用于向量余弦相似度的近似近邻搜索[9]，提出了基于随机超平面的 LSH 方法. LPLSH 本质上是随机超平面 LSH 方法的改进，对超平面划分的 LSH 方法，需要生成 k 个哈希函数，$H(x)=\{h_1(x),h_2(x),\ldots,h_k(x)\}$，每个哈希函数表示一个超平面. 构造哈希编码的过程是：把元素投影到超平面法向量上，根据投影在超平面两侧分别赋值 0/1，从而得到哈希编码. 具体如公式(8)、公式(9):

$$HashCode=[h_1(x),h_2(x),\ldots,h_k(x)] \tag{8}$$

$$h_i(x)=\begin{cases}1, & \text{if } w_i^{\mathrm{T}}x-b_i\geqslant 0\\ 0, & \text{if } w_i^{\mathrm{T}}x-b_i<0\end{cases} \tag{9}$$

其中，w_i 是服从高斯分布的随机向量; b 则是平面的偏移量，b 的取值将在第 4.3 节中详细叙述.

在设置法向量时，我们允许利用维度精简方式，即通过随机选择方式确定 w_i 上的 d'个维度，并以服从高斯分布的随机向量值填充，其余设置为 0. 维度精简可保持数据间的距离关系，同时减少 w_i 中的非零数，以减少哈希函数的计算量，从而把哈希编码的计算复杂度从 O(d)降低到 O(d'). 生成哈希函数法向量的算法如算法 1 所示.

算法 1. 哈希函数构造算法(create_hash).

输入: 维度 d, 编码长度 k, 投影维度 d', 数据 X, 采样率 sr=0.1;

输出: $wb=\{(w_i,b_i)|i\in\{0,\ldots,k-1\}\}$.

```
01  X′=randSampling(X,sr);   /*随机采样*/
02  FOR (cutPos=[·], k=0; k<self.d; k++)
03      t_sum+=max(X′[:,k])−min(X′[:,k]); cutPos.append(t_sum));
04  FOR (i=0; i<k;)
05      hits=[0,…,0]; tmp_w=[0,…,0]; candidate_w=random.normal(size=d′); counter=0;
06      WHILE counter<d′:
07          randV=random(0,cutPos[d−1]);
08          FOR (k=0, pos=−1; k<d and tmp≥0; k++)
```

```
09          IF cutPos[k]≥randV
10              pos=k;
11      IF hits[pos]==0
12          tmp_w[pos]=candidate_w[counter]; hits[pos]=1; counter+=1;
13      w[i]=tmp_w;
14      b=Generate_offset(w[i],X′);   /*调用算法 2 计算偏移量*/
15      IF r==True
16          wb=wb∪{(w[i],b)}; i++;
17  RETURN wb
```

算法 1 首先从数据集 X 中随机获取 $N'=N\times sr$ 个样本(第 1 行), $N=|X|$; 之后的 FOR 循环生成每个维度取值范围的叠加(第 2 行、第 3 行), 以便根据维度取值范围指导维度选择. for 循环控制算法逐个生成 k 个投影平面 $(w[i],b)$(第 4–16 行), 其中, WHILE 循环中首先生成 d'个符合正态分布的随机向量, 然后按照 *cutPos* 所记录的每个维度被选中概率为 $w[i]$中具体元素赋值(第 6–12 行). 算法中的数组 *hits* 用于记录 $w[i]$分量已被选中维度, 避免重复选择; 在选定 $w[i]$后, 调用第 4.4 节中的算法 2 进行偏移量的生成.

3.3 LPLSH的偏移量设置

超平面偏移值设置主要包含如下 5 个步骤.

(1) 样本数据在 w_i 所表示的超平面法向量上进行投影;

(2) 确定高斯核带宽 h;

(3) 利用高斯核函数进行概率密度估计;

(4) 把 Laplacian 算子应用于数据投影密度分布;

(5) 选择合理的偏移量值.

本节将针对这 5 个步骤进行详细的展开.

(1) 步骤 1. 在法向量 w_i 上投影.

样本数据集 X'在 w_i 所表示的平面法向量上进行投影, 其操作如公式(10):

$$X'_{proj}=proj(X',w_i)=X'\cdot w_i^T \tag{10}$$

步骤 1 得到 N'维向量, 表示样本数据集在向量 w_i 上的投影, 反映样本数据在投影方向上的分布情况.

(2) 步骤 2. 高斯核带宽的设定.

在高斯核的应用中, 高斯核带宽 h 为计算过程中的关键参数. 对于一个给定的样本数据集投影 X'_{proj}, 若 h 设置太小, 那么最后的概率密度估计结果具有小偏差大方差; 反之, 估计结果则是大偏差小方差. 为此, 文献[30]提出一种快速求带宽 h 的方法, 如公式(11):

$$h=(4\pi)^{-1/10}\left[\left(\frac{3}{8}\right)\pi^{-1/2}\right]^{-1/5}\sigma n^{-1/5}\approx 1.06\sigma n^{-1/5} \tag{11}$$

其中, $n=N'$表示样本元素个数, σ是标准差.

Silverman 在 1986 年提出一种具有更好可用性的标准差替代 A[30], A=min(标准差σ, 四分位数/1.34). 因此, 本文最终应用的求带宽公式为

$$h=1.06An^{-1/5} \tag{12}$$

(3) 步骤 3. 利用高斯核函数进行概率密度估计.

为了能够在投影方向上找到恰当的划分位置/偏移位置, 需要数据投影的概率密度分布做参考. 数据投影后转换到一维空间上, 所以本文采用一维高斯函数作为概率密度估计的核, 高斯核密度函数定义如公式(4).

为了避免投影分布区间过大影响计算精度与效率, 本文把投影区间 $[proj_{\min}=\min(X'_{proj}), proj_{\max}=\max(X'_{proj})]$分为 M 等份, 每等份宽度为 step. 第 k 个位置的高斯核密度值计算如公式(13):

$$\hat{f}_{N',step}(k)=\frac{1}{hN'}\sum_{i=1}^{N'}\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{\|(proj_{\min}+k\times step-x_i\|^2}{2h^2}\right) \tag{13}$$

进一步计算高斯核密度概率分布函数如公式(14):

$$P_{step}[i]=\int_0^{iw}\hat{f}_{N',step}(x)\mathrm{d}x=\sum_0^i w\hat{f}_{N'}(i\times w) \tag{14}$$

核密度方法有许多优点: 首先, 高密度区域的识别与区域形状无关; 其次, 核函数的平滑效应使密度估计对噪声具有较好的鲁棒性, 从而增强算法对开放环境的适应性; 最后, 估计的密度函数能保持数据投影分布的基本形态.

(4) 步骤 4. 利用 Laplacian 算子求高斯核概率密度二阶导数.

超平面偏移量设置的期望位置是 $\hat{f}_N(x)$ 所定义的多个正态分布数据的交界处. 但在开放环境的应用中, 多个中心的数据叠加容易导致密度被平滑, 利用概率密度函数很难分辨出多个数据簇组合的边界位置.

针对高斯核概率密度估计, 利用 Laplacian 算子可以侦测到密度急剧变化区域, 包括数据簇的波谷位置. Laplacian 算子计算高斯核概率密度二阶导数如公式(15):

$$\varphi(x)=\Delta(\hat{f}_{N'}(x))=\sum_{i=1}^{N'}\frac{\|x-x_i\|^2-h^2}{N'h^5\sqrt{2\pi}}\exp\left(-\frac{\|x-x_i\|^2}{2h^2}\right) \tag{15}$$

为了加快运算效率, 按步长 *step* 求 *M* 个位置的概率密度函数的二阶导数, 具体如公式(16):

$$\varphi_{step}(i)=\Delta(\hat{f}_{N',step}(i))=\frac{step^2}{N'h^5\sqrt{2\pi}}\sum_{i=1}^{N'}((proj_{\min}+k\times step-x_i)^2-h^2)\exp\left(-\frac{\|(proj_{\min}+k\times step)-x_i\|^2}{2h^2}\right) \tag{16}$$

(5) 步骤 5. 确定超平面的偏移值 *b*.

综合所求概率密度的二阶导数值及概率密度分布函数, 确定超平面的偏移值 *b*, 就是要找到一个能够反映分布聚类簇数据边缘的位置. 具体见算法 2.

算法 2. 偏移量计算算法(generate_offset).

输入: 随机超平面向量 *w*, 数据样本 *X*′;

输出: (*r*,*b*)分别表示偏移量生成是否成功以及成功时的平面偏移量.

```
01  Proj[0…N′]=X′·W^T;
02  step=(max(Proj)−min(Proj))/M;
03  利用公式(14)求 Pr[0…M];
04  利用公式(16)求 Lap[0…M];
05  proj_min=min(Proj[0…N′]);
06  idxs[·]=extrema(Lap);   /*求 Lap 极值索引*/
07  idxs=Sort(idxs,Lap);   /*idxs 按 Lap 排序*/
08  FOREACH idx IN idxs   /*从大到小遍历*/
09    IF Pr[idx] IN pr_range;
10      RETURN (TRUE,idx*w+proj_min)
11  RETURN (FALSE,None);
```

算法中存在两个常量(M,pr_{range})需要初始设置, 其中, M 表示把数据投影分成多少个区间, 由 M 可以求区间宽度 *step*, 从而支持公式(14)、公式(16)的运算. M 设置太大, 会导致投影的概率密度出现较多极值, 投影的分布变得复杂, 计算效率变低; 若 M 设置太小, 则会掩盖数据的分布特性, M 取值在 80–150 是合理的. pr_{range} 用于确保开放环境中少量离群数据不会对算法产生大的影响, 增强算法的鲁棒性. 依据箱式图的离群数据定义, 以两个具有不同标准差的正态分布复合数据为例, 当标准差比值为 1–10 时, 其离群数据比例为 4.8%–25%. 本文预设离群数据占比为 20%(两端各 10%), 以能覆盖绝大多数复合分布情况, 适应开放应用环境的要求. 本文把 M 设置为 100, pr_{range} 设置为[0.1,0.9].

3.4 算法的复杂性分析

算法 2 中: 投影计算需要的时间是 O($N'd'$), 调用公式(14)求概率分布 Pr 的时间复杂度为 O(MN'), 利用公式(16)求 Laplacian 的时间复杂度为 O(MN'); 对二阶导数组求极大值的索引, 时间复杂度为 O(M); 遍历结果进行排序, 时间复杂度为 O(Mlog(M)); 最后, 对结果进行匹配筛选, 时间复杂度为 O(M). 所以, 算法总的时间复杂度为 O(N'(2M+d')+Mlog$M$+2M). 但是在实际的使用中, d'远小于 d, M 又远小于 N', 二阶导数的极值数量也远小于 M, 所以, 算法 2 的时间复杂度可以简化为 O(N'(M+d')).

算法 1 调用了算法 2, 实现了哈希函数的构造. 算法 1 的每一次 WHILE 循环都是随机找到一组 $w[i]$之后调用算法 2, 以确定依据该组 $w[i]$的哈希函数偏移量, WHILE 循环将会生成 k 个 $w[i]$; 综合上述对算法 2 的分析, 算法 1 的总时间复杂度为 O(kN'(M+d')). 在少数情况下, 可能有部分 $w[i]$无法确定其偏移量, 需要多于 k 次循环的情况, 但是并不会影响总体时间复杂度.

从最终时间复杂度可以看出: 首先, 我们用采样数据代替全体数据集, 即 N'代替 N, 使其运行花费时间大大减少; 其次, 哈希函数不需要考虑所有维度上的信息, 其维度从 d 降低到了 d', 这不但加快了哈希函数的构造, 也使得后继建立哈希结构和查询更加快速; 最后, 在计算概率密度和利用 Laplacian 算子计算二阶导数时规范了数据范围为[0…M), 这一处理不但可以减少计算量, 同时也对实际数据投影的概率密度和二阶导数值进行了平滑, 避免了用少量采样数据代替全部数据的影响.

3.5 相关性质分析

本节将对 LPLSH 方法相关理论进行证明和分析, 主要包括: (1) 证明精简维度的哈希函数仍然满足局部敏感性; (2) 证明在投影的低密度区域进行超平面分割具有更小的最大误分割元素占比; (3) 分析 Laplacian 算子求得的二阶导数对超平面偏移量设置的指导意义.

定理 1. 若数据集 $P\subset R^d$, $x\in P$, 如下哈希函数定义能够满足局部敏感哈希特性:

$$h_w(x)=\begin{cases}1, & \text{if } w\cdot x\geqslant 0\\ 0, & \text{if } w\cdot x<0\end{cases} \tag{17}$$

其中, w=[w_0,w_1,…,w_{d-1}], w 是由 d'($d'\leqslant d$)个符合正态分布的随机值及 d–d'个 0 构成.

证明: 不失一般性, 假设数据集中 d 个维度的数据是相互独立的, w 的前 d'项是非零值, 即 w=[w_0,w_1,…, w_{d-1},0,…,0]. 向量 v[i…j]表示向量[v_i,…,v_j], v'=v[0…d'–1], 表示向量 v 的前 d'维组成的向量, v''=v[d'…d–1], 表示向量 v 的第 d'+1 到 d 维组成的向量. 由文献[9]的基于随机超平面哈希函数相似性可知: 对 u,$v\in P$, 有如下公式成立:

$$\Pr[h_{w'}(u')=h_{w'}(v')]=1-\frac{\theta(u',v')}{\pi} \tag{18}$$

由于 w''=w[d'...d–1]=[0,...,0], 所以:

$$\Pr[h_{w''}(u'')=h_{w''}(v'')]=1 \tag{19}$$

综合公式(18)、公式(19), 可以得公式(20):

$$\Pr[h_{w'}(u')=h_{w'}(v')\wedge h_{w''}(u'')=h_{w''}(v'')]=\Pr[h_w(u)=h_w(v)]=1-\frac{\theta(u',v')}{\pi} \tag{20}$$

由于各维度数据的相互独立性, 结合公式(20)可知: u, v 距离越大, 则其 $h_w(\cdot)$相似的概率越小. 由此说明了从 d 维度中随机选择 d'个维度, 并给向量 w 对应维度设置为符合正态分布的随机值, 而其余设置为 0, 基于该向量 w 的哈希函数满足局部敏感特性. 证毕. □

由定理 1 可知: 算法 1 通过精简 d 维投影向量到随机 d'维投影向量的方法能保持哈希函数的局部敏感性, 符合随机超平面局部敏感哈希方法的基本要求.

定义 4(最大误分割元素占比). 随机法向量 w 及偏移量 b 确定的超平面, 所对应的哈希函数为

$$h_{w,b}(x)=\begin{cases}1, & \text{if } w\cdot x\geqslant b\\ 0, & \text{if } w\cdot x<b\end{cases};$$

$f(x)$为数据投影的概率密度函数; 最大误分割元素占比是指所有查询中, 与查询点投影的δ距离内可能被设置为不同的哈希值(即误分割到超平面两侧)的元素占所有数据元素的比例, 定义如公式(21):

$$WR_{w,b}=\int_{b-\delta}^{b+\delta} f(x)\mathrm{d}x \tag{21}$$

对查询点 q, 当$|q-b|\geqslant\delta$时, 该查询被误分割元素为 0; 当$|q-b|<\delta$时, 其误分割元素为与 q 距离为δ范围内, 且被分割到超平面另外一侧的元素. 最大误分割元素占比反映了在所有查询中, 可能被误分割到随机超平面另外一侧的元素占所有元素的比例的最坏情况, 其值越小越好.

定理 2. 假设 w 为随机超平面的法向量, 数据集 P 在 w 上投影的概率密度函数为 $f(x)$, 那么随机超平面的偏离值 b 对应的 $f(b)$越小, 基于该超平面的最大误分割元素占比越小.

证明: 我们不妨假设在一维空间考虑该问题, 若扩展到高维空间, 需加入哈希函数的局部敏感性质, 建立数据与其投影之间的关系.

假设有两个不同的偏离值 $b1$, $b2$, 所对应位置的数据投影概率密度分别为发 $f(b1)$, $f(b2)$, 且 $f(b1)>f(b2)$. 他们的最大误分割元素占比分别为 $WR_{w,b1}$, $WR_{w,b2}$, 当$\delta\to0$ 时, 由最大误分割元素占比定义可知:

$$WR_{w,b}=\int_{b-\delta}^{b+\delta} f(x)\mathrm{d}x\approx 2\delta f(b).$$

由于 $f(b1)>f(b2)$, 所以 $WR_{w,b1}>WR_{w,b2}$. 证毕. □

从定理 2 可知: 在投影的低密度区域进行超平面分割, 具有更小的最大误分割元素占比. 在设置随机超平面的偏移量时, 找到概率密度越小的位置, 则越说明该超平面所穿过的是数据稀疏区域, 那么基于该超平面分割的数据查询也自然具有更高的查询精度.

与此同时, 我们注意到: 如果利用设置偏移量让所有数据位于超平面的同一侧, 则此时该位置数据的概率密度为 0, 具有理论上的最小最大误分割元素占比. 但是这导致了超平面分割方法的完全失效, 会使基于超平面分割的 LSH 检索方法退化为暴力搜索. 为此, 本文通过利用 Laplacian 算子计算概率密度函数的二阶导数, 以期望能够找到二者之间的折衷.

从全局角度确定超平面对数据空间的分割位置可以有多种方法, 例如: 直观上, 可以通过法向量上的投影密度分布找到密度函数波峰之间的波谷位置, 但是在很多情况下, 数据投影的概率密度可能形成多种不同的形态. 以两个符合正态分布概率密度为例(实际情况可能更加复杂), 其关系可以是: (1) 两个正态分布均值相近, 即 $u1\approx u2$, 无法形成两个波峰; (2) 两个正态分布 $u1$ 与 $u2$ 距离较远, 形成明显的波峰和波谷; (3) 两个正态分布的 $u1$ 和 $u2$ 较为接近, 无法形成波谷, 其复合概率密度又与单个正态分布明显不同. 对于上述情况, Laplacian 算子均可以准确感知正态分布剧烈变化位置.

从上面的分析可知: 通过利用 Laplacian 算子对投影的概率密度求二阶导数的方法, 可以量化的方式发现数据剧烈变化的位置, 通过发现这些位置, 并利用其作为哈希函数的偏移量, 可以提供一种有效的随机超平面分割方法, 从而提高基于超平面分割的局部敏感哈希近似近邻元素的检索效果, 增强算法对开放环境的适应性.

4 实验与结果

实验针对具有不同分布特性的 5 个数据集展开; 第 4.2 节中, 对 8 个算法及 LPLSH 的 3 种不同设置在 100NN 上的检索精度和效率进行对比与分析; 第 4.3 节中对 LPLSH 方法的不同 d'设置进行对比; 第 4.4 节中将详细分析各数据集数据的分布特性对算法检索效果的影响, 说明 LPLSH 的高适应性.

为了观察维度精简对整个检索方法的影响, 针对精简维度 d'对 LPLSH 做两种设置. 在 $d'=d$ 时, LPLSH 标记为 LPLSH-A; 在 $d'=ln(n)$时, LPLSH 标记为 LPLSH-R. LPLSH-E 则表示以 Epanechnikov 核(如公式(22))替代高斯核的算法实现, 以考察不同核函数对算法的影响:

$$K_{Epane}\left(\frac{x}{h}\right)=\begin{cases}\dfrac{3}{4h}\left(1-\left(\dfrac{x}{h}\right)^2\right), & \left|\dfrac{x}{h}\right|\leqslant 1\\ 0, & \left|\dfrac{x}{h}\right|>1\end{cases} \tag{22}$$

4.1 实验准备

4.1.1 参数设置

对于每个数据集, 随机抽取其中 2‰数据作为查询对象, 并采用欧式距离作为相似度度量标准, 本文检索 100NN 并进行实验分析, 以考察 LPLSH 与其他方法的对比.

4.1.2 数据集

本文挑选了不同领域中的 5 个大规模高维数据集进行对比实验, 数据集见表 1.

表 1 实验数据集

数据集名称	元素数量	维度	其他信息
FMA (https://github.com/mdeff/fma)	106 574	518	音频采样
MNIST (http://yann.lecun.com/exdb/mnist/)	60 000	784	手写数字图像
GLoVe (https://nlp.stanford.edu/projects/glove/)	400 000	300	Wiki2014+Giga5 的文本词频特征
NUS_WIDE (https://lms.comp.nus.edu.sg/wp-content/uploads/2019/research/nuswide/NUS-WIDE.html)	269 648	4 096	数字图像
SIFT1M (http://corpus-texmex.irisa.fr/)	1 000 000	128	SIFT 特征集

4.1.3 比较算法

本文挑选两种数据无关的 LSH 近似近邻检索方法(RHP, SBLSH)及 6 种数据依赖的 LSH 近似近邻检索方法(PCAH, DSH, ITQ, KLSH, LH, GLDH), 本文算法及所用对比方法的参数设置见表 2.

表 2 比较算法列表

对比算法名称	参数设置与其他
PCAH[11]	基于主成分分析的哈希算法
DSH[15]	为提高效率, 使用 MiniBatchKmeans 替代原文的 K-means 算法, 迭代次数 20, *batchSize*=0.1*n*, 通过多次实验, 验证其不会对结果造成明显影响. *r* 设置太小影响精度, 设置太大影响效率. 本文通过多次实验, 最终确定设 *r*=4, 聚类数则根据数据元素数量和码长进行设置, 范围在 15–50 递增
ITQ[14]	基于迭代量化的局部敏感哈希算法, 按原文说明, 采样率设置为 2.5%, 迭代次数 50; 继续增加采样数据并不会明显影响算法精度, 却会导致算法效率明显变差
RHP[9]	基于随机超平面的局部敏感哈希算法
SBLSH[13]	超位局部敏感哈希算法
KLSH[12]	参数 *p* 缺省值为 300, 其构造超平面复杂度为 O(p^3), 查询时间复杂度为 O(*p*). 为提高准确率, 本文按数据集大小设置 *p*=300, 600, 900
LH[26]	哈希函数构造需要进行较多矩阵运算, 与本文算法相同, 通过采样 10%数据进行分析, SIFT 数据集上则采样 5%
GLDH[17]	由于算法效率问题, 文献[17]中分析和检索数据为 10^4, 本文也按照 10^4 设置 采样数据进行哈希函数构造, 设置$\sigma=k^{1/2}n^{-1/5}$, 聚类数为 20, *r*=6, *T*=80, β=75
LPLSH-A	基于 Laplacian 算子的局部敏感哈希, $d'=d$
LPLSH-R	基于 Laplacian 算子的局部敏感哈希, $d'=\ln(n)$
LPLSH-E	利用二次核(Epanechnikov)实现的基于 Laplacian 算子的局部敏感哈希, $d'=d$

4.1.4 性能指标

实验主要关注 3 个性能指标: 精度、*F*1 值、时间消耗, 其中,

- 精度: 反映存储到同一哈希桶的元素与查询元素的相似程度, 其值越大, 则说明桶中元素互为近邻元素的概率越大. 通常的精度定义以哈希桶中包含的最近邻 *K* 个元素为分母, 该定义经常会导致哈希桶元素中元素越多其值越优, 不能很好地反映哈希结构的质量; 本文采用公式(23)定义, 该 *precision* 以整个哈希桶元素数量为分母, 能够更准确地反映所构造哈希结构的质量:

$$precision(q)=\frac{100\text{NN}\cap bucket(h(q))}{|bucket(h(q))|} \tag{23}$$

- $F1$ 值: 该指标可以综合反映算法的检索效果, 其计算如公式(24), $F1$ 值越高, 则算法的综合搜索能力越强:

$$F1=\frac{2\times recall\times precision}{recall+precision} \tag{24}$$

单纯依赖 $F1$ 并不能全面反映方法的检索效果, 特别是在哈希桶中命中近邻元素较少时, 通常会由于少数元素不同导致各衡量指标的显著差异. 为此, 本文同时展示检索精度和 $F1$ 值, 并综合分析哈希函数的检索质量;

- 算法时间是指算法运行的整体时间, 包含哈希函数的构造、数据的存储和随机抽取总数的 0.2%的元素进行近邻查询花费的时间.

4.2 实验结果与分析

本文在实验中设置不同的编码长度, 利用对应的方法构造哈希表. 在查询过程中, 只查询二进制哈希编码对应的哈希桶, 并获取桶中的数据元素进行计算查询 *precision* 和 $F1$. 实验结果如图 4 所示.

4.2.1 与现有算法的比较分析

首先, 我们把本文算法产生的结果与 RHP 进行对比. 在所有 5 个数据集中, LPLSH 算法的 $F1$ 值均明显优于 RHP 算法, 其中, 在 SIFT 数计集上表现最为接近. 同时也可以看到: 除了 FMA 数据集, RHP 方法在其他几个数计集上都显示出较高的精度. 结合 $F1$ 指标可以知道: RHP 方法的随机划分容易把数据划分得过分细致, 以至于每一个桶中的数据都很少(低召回率), 导致具有较高精度的同时, $F1$ 却不如其他算法. 从这里也可以看出: 采用完全随机的方式无法有针对性地进行数据划分, 在均匀分布, 或者各维度数据分布满足具有相似期望的情况下, 常常具有较好的效果(如 SIFT). 从与 RHP 对比中可以看出: 本文所采用的方法虽然是基于超平面划分的概念, 但是其改进是明显有效的.

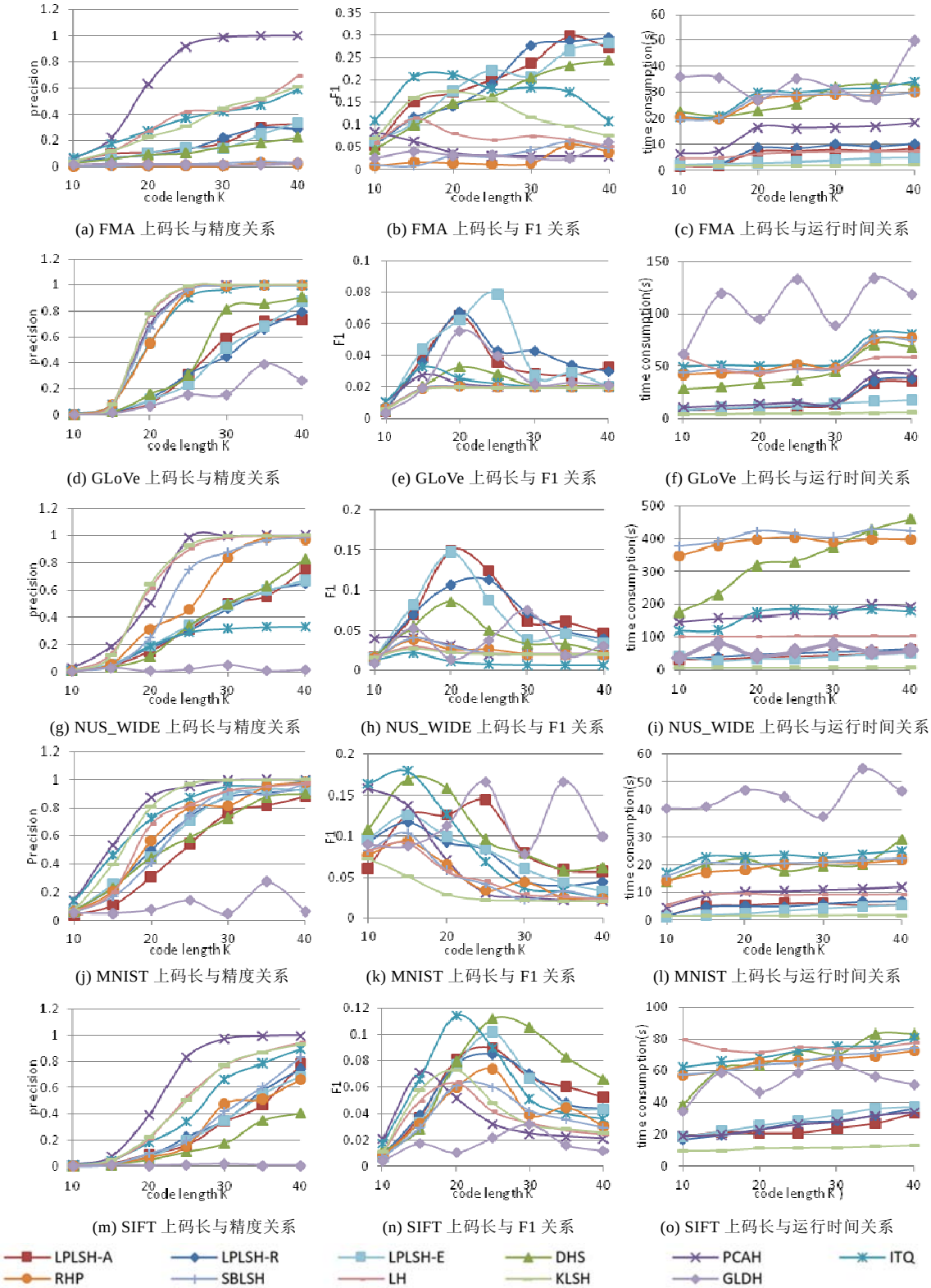
SBLSH 与 RHP 都属于数据无关的 LSH 检索方法, 他们具有非常接近的曲线形态. 在所有情况下, LPLSH 的均明显优于 SBLSH, 这也说明了数据依赖的 LSH 对数据分布信息利用的有效性.

与 DSH 对比, 在数据集 FMA 中, LPLSH 方法在 $F1$ 指标上优于 DSH 方法, 同时也具有较高的精度, 说明了 LPLSH 相比 DSH 在该数据集上能够更好地协调精度和召回率. 在数据集 GLoVe 和 NUS_WIDE 中, LPLSH 的 $F1$ 明显优于 DSH; DSH 在精度上基本与 LPLSH 相当, 这充分说明了 LPLSH 在这两个数据集上划分的有效性. 在 MNIST 上, DSH 和 LPLSH-A 在不同的 K 位置展示出不同的最高 $F1$ 值, DSH 在 $F1$ 上则优于 LPLSH-R; 在精度上, LPLSH 与 DSH 相当. 最后, 在 SIFT 数据集上, DSH 在 $F1$ 上具有较为明显的优势; 但是在精度上, DSH 则明显不如 LPLSH, 说明了 DSH 分割得到的每一个哈希桶上元素较多, 从而具有较高的召回率.

PCAH 算法的实验结果表明: 该算法总是具有较高的查询精度, 在 FMA 数据集中, 甚至两倍于其他算法的精度. 但是综合衡量指标 $F1$ 值却没有像精度一样优于其他算法, 这也说明了 PCAH 的划分导致了很低的召回率, 未能协调好查全与查准的关系. 从总体上看, PCAH 方法的 $F1$ 值在 K 较小时与其他方法差别不大, 但是随着 K 的增大, $F1$ 快速降低, 总体检索表现明显不如 LPLSH.

ITQ 算法也具有较高的查询精度, 精度上仅次于 PCAH; 多数情况下, ITQ 在 K 值较小时(K=10)的 $F1$ 表现较好, 但是随着 K 的增加, 其 $F1$ 先明显增加, 之后快速下降. 在数据集 FMA, MNIST 和 SIFT 中, ITQ 的 $F1$ 值先随着 K 增大快速增加, 之后快速下降. 从 $F1$ 最高值上看, ITQ 在这 3 个数据集上均明显不如 LPLSH. 在数据集 GLoVe 和 NUS_WIDE 上, LPLSH 算法的 $F1$ 则全面明显优于 ITQ 算法, 这也说明 LPLSH 具有良好的适应性.

LH 算法展现出较高的精度值. 在各个数据集中, 其 $F1$ 均不突出, 从具体的数据集上看, LH 算法在 FMA, MNIST 和 SIFT 这 3 个数据集上的 $F1$ 表现不如本文的 LPLSH 算法, 在 GLoVe 和 NUS_WIDE 上则远远落后于 LPLSH 的表现.



(a) FMA 上码长与精度关系 (b) FMA 上码长与 F1 关系 (c) FMA 上码长与运行时间关系

(d) GLoVe 上码长与精度关系 (e) GLoVe 上码长与 F1 关系 (f) GLoVe 上码长与运行时间关系

(g) NUS_WIDE 上码长与精度关系 (h) NUS_WIDE 上码长与 F1 关系 (i) NUS_WIDE 上码长与运行时间关系

(j) MNIST 上码长与精度关系 (k) MNIST 上码长与 F1 关系 (l) MNIST 上码长与运行时间关系

(m) SIFT 上码长与精度关系 (n) SIFT 上码长与 F1 关系 (o) SIFT 上码长与运行时间关系

图 4 在 5 个数据集上, 100NN 检索的编码长度 K 与精度、$F1$ 和运行时间的关系曲线

KLSH 算法的表现与 LH 算法非常类似, 都展现出较高的精度值, 且二者精度值非常接近; 在各个数据集中的 $F1$ 均明显不如 LPLSH, 这也说明了 LPLSH 能够实现精度与召回率之间的良好平衡.

GLDH 算法在 5 个数据集上都表现出非常低的精度值, 说明了该方法构造的哈希桶中数据过多, 分布不均衡, 从而导致精度值低. 在 FMA, SIFT 和 NUS_WIDE 数据集上, GLDH 的 $F1$ 值明显低于 LPLSH; 在 GLoVe 数据集上, GLDH 的 $F1$ 值接近 LPLSH 算法的 $F1$ 值; 而在 MNIST 数据集上, 当编码长度为 35 时, 其 $F1$ 值则高于其他算法. 从总体上看: GLDH 在精度、综合指标 $F1$ 值上的表现均不如 LPLSH; 此外, 该算法的检索效果受具体数据集的影响较大.

从上面的分析中可以看出: 本文算法在具有不同分布特性的 5 个数据上都有良好的表现, 说明其具有良好的查询效果以及对开放环境卓越的适应性; 稳定的精度表现和 $F1$, 说明算法能够平衡准确率和召回率. 特别的, 在 FMA, GLoVe 和 NUS_WIDE 这 3 个数据集上, 更是全面展现出最好的查询效果.

4.2.2 算法效率分析

从算法的时间消耗上看, KLSH 具有很高的效率. 其原因在于, 该方法所用样本数量较小(本文实验设置 KLSH 的样本数量为缺省数量的 2 –3 倍, 其绝对值并不大). 除了 KLSH, LPLSH 方法在所有其他数据集上都展现出高效的一面. 具体来看: 大多数算法的时间主要花费在数据预处理过程中, 例如聚类、PCA 分析、特征值和特征向量的计算等. 至于查询时间方面, 单个查询都能在较短时间完成数据的查询操作, 具体时间还取决于哈希桶中元素的数量. GLDH 算法采用固定样本数, 其所用时间则显得更加稳定.

同时也可以注意到, 所有算法的运行时间大都随着编码长度 K 的增大而增大.

- 一方面是由于随着编码长度的增加, 需要花费更多的时间在数据预处理和构造哈希函数上; 此外, 随着编码长度的增加, 需要更多的时间计算数据的哈希编码, 从而导致了更长的处理时间;
- 在另外一方面, 编码长度 k 的增加也导致了每一个哈希桶中元素数量的减少, 从而减少了同一个桶中元素的排序时间. 但是这一时间上的消耗非常少, 所以总体体现出时间消耗随编码长度的增加而增加.

4.3 LPLSH-A, LPLSH-R和LPLSH-E的对比

LPLSH-A 对应的哈希函数考虑了所有的维度信息, 即 $d'=d$; 而 LPLSH-R 则只考虑了部分的维度信息, 即 $d'=ln(n)$. 图 4 中包含了两种情况下的算法对比. 在 FMA 和 SIFT 两个数据集上, LPLSH-A 和 LPLSH-R 的 $F1$ 指标基本没有差别. 对于 GLoVe 数据集, 当 K=30 时, LPLSH-A 的 $F1$ 略差于 LPLSH-R; 而在其他 K 取值的情况, 他们的 $F1$ 也几乎没有差别. 在数据集 NUS_WIDE, 当 K=20 时, LPLSH-R 的 $F1$ 不如 LPLSH-A; 而在其他 K 取值的情况, 他们的 $F1$ 几乎没有差别. 而在上述 4 个数据集中, 二者的精度也几乎一致. 在数据集 MNIST 中, LPLSH-A 的 $F1$ 指标要明显优于 LPLSH-R; 与之相对应的是, LPLSH-A 的精度略不如 LPLSH-R, 这也说明了 LPLSH-R 的召回率不如 LPLSH-A.

在所用时间的对比上, 总体上二者较为接近, 却不能看出 LPLSH-R 更加节省时间, 其主要原因在于: 当我们在确定平面偏移量时, 若不能确保分割位置在[0.1,0.9]之间, 则需重新随机生成法向量并再次计算. 使用的维度数量越少, 则越容易出现这种情况. 所以 LPLSH-R 需生成更多的随机向量并分析, 从而导致其构造时间增加. 而查询时间耗费都在毫秒级, 所以 LPLSH-R 在少量查询中并没有体现出总时间的节省.

从上述结果也可以看出: 通过维度的精简, 不会导致 LPLSH 方法精度和召回率退化. 对大多数数据集上的应用而言, 精简后虽然需要耗费较多的哈希函数构造时间, 但是在后期的查询中, LPLSH-R 会逐步展现出其性能优势.

利用二次核(epanechnikov kernel)替换高斯核后, 从图 4 中可以看出: LPLSH-E 在数据集 GLoVe 和 SIFT 上表现出略高的 $F1$, 在 MNIST 上又略低于 LPLSH-A, 而在 FMA 和 NUS_WIDE 上两者相当. 因此从总体上来说, 二次核替换高斯核并没有导致检查精度、和时间上的大幅提高或者降低, 说明只要能够选择合适的带宽, 他们都能有良好的表现. 在效率方面, 除了 SIFT 数据集, 在其余 4 各数据集上, LPLSH-E 的效率均略微高于 LPLSH-A. 其主要原因在于: 高斯核随着 x 值远离中心, 其值趋近于 0(不等于 0); 而二次核在$|x-x_0|>h$ 时,

其值为零, 自然也就在一定程度上减少了计算量, 提高了效率.

4.4 数据分布对实验结果的影响

从第 4.2 节的对比分析可以看出: 在所有对比方法中, 没有一种方法能够在所有数据集上形成绝对的优势, 这也说明了不同的方法都有其适用范围. 例如: LPLSH 方法在 GLoVe 和 NUS_WIDE 数据集上明显优于其他所有方法, 在 FMA 上总体优于现有各种方法, 在 MNIST 和 SIFT 数据集上则优于多数方法. 本节将从数据集的分布特征出发, 分析 LPLSH 在各种分布特征数据集上的适用性, 说明其对开放环境的高适应性.

(1) 数据集 MNIST 与 SIFT

- MNIST 的数据特点: 每个维度中的数据主要分布在黑和白两端, 且白色端占了主要部分, 中间数据极少;
- SIFT 的数据特点: 各个维度中的数据以 0 为占比最高, 之后快速下降形成长尾; 少部分维度在尾部较远处形成另外一个小波峰.

对于 MNIST 和 SIFT, 通过利用满足正态分布的随机向量进行投影, 其数据投影特征也分为两类: 其一是形成单峰效果, 之后快速下降, 形成长尾; 其二是能够形成双峰甚至多峰的分布特性, 且主波峰占比绝大部分, 其余峰值虽然出现, 但峰值小. 按照本文的方法, 以数据分布变化最为剧烈的位置为超平面偏移量, 其偏移位置通常位于主波峰底部, 未能在两个波峰中间空白区进行划分; 数据划分后, 后半部分的元素间距离平均值较大, 从而导致了 LPLSH 方法在这两个数据集上未能大幅超越其他算法.

在该数据集中, LPLSH-A 要总体上略优于 LPLSH-R. 其原因在于: LPLSH-A 的哈希函数考虑了更多维度信息, 也就使得投影所形成的概率密度具有更加明显的多正态分布形态, 从而 Laplacian 算子计算的二阶导数最大值更加靠近双峰中间位置; 而 LPLSH-R 考虑的维度较少, 其二阶导数最大值更可能靠近其中一边.

(2) 数据集 GLoVe 与 NUS_WIDE

GLoVe 和 NUS_WIDE 两个数据集在每一个维度上的数据分布都大致符合正态分布, 各个维度上正态分布的数学期望基本相同, 方差有所不同. 对于这样的数据集, 当投影到随机生成的符合正态分布向量上时, 其基本形态还是保持为正态分布, 不会形成多峰效果.

对于一般的方法, 其所生成的超平面大都从正态分布的中间穿过, 所以导致算法退化为随机超平面方法, 效果不好. 而本文的 LPLSH 方法则能够利用 Laplacian 算子对数据变化的敏感性, 有效定位到数据投影正态分布剧烈上升的开始位置, 或者急剧下降到平缓的过度位置, 从而正确设置哈希函数的偏移量. 因此, 在这两个数据集上, 本文 LPLSH 方法有明显优于其他所有算法的表现.

(3) 数据集 FMA

从单个维度上看, FMA 数据具有较高的集中度, 且基本满足单个正态分布形态; 但各个维度上数据的数学期望和方差的差异性极大, 数学期望的差异达上千倍, 方差达数十倍. 对于这样的数据集, 当投影到随机生成的符合正态分布的向量上时, 其多个维度复合形成的概率密度形态大都近似于单个偏态分布, 且具有较大的方差, 这也符合数据长条形分布的特点(方差差异明显). 因此, 当利用 Laplacian 算子求数据投影剧烈变化位置时, 哈希函数偏移量总是能够被准确定位在偏态分布的剧增起始/剧减结束位置, 从而展现出良好的分割效果.

综上, 本文所提出的 LPLSH 方法在高维大规模数据集上的 ANN 检索具有较高的精度和效率, 对多种分布特性数据的 ANN 检索具有良好的适应性, 能够适应开放环境下多分布特性的 ANN 检索需求.

5 总 结

开放环境对大规模高维数据的近似近邻检索提出了更高的要求, 构建一个适用于开放环境且具有良好性能的大规模高维数据检索方法是一项非常重要的工作. 本文提出了一种适应多种分布特性的基于 Laplacian 算子的 LSH 近似近邻数据检索方法 LPLSH, 该方法通过随机选择满足正态分布的投影向量, 对数据投影进行高斯核密度估计, 再利用 Laplacian 算子计算投影概率密度的二阶导数, 确定超平面的偏移量, 由此生成的哈希

函数簇能够从全局的角度有效感知数据分布的剧烈变化位置, 以提高基于超平面分割的局部敏感哈希方法的有效性. 通过多种分布特性数据的实验, 体现了本文所提出算法在开放环境下具有很强的鲁棒性, 且实现了数据查询精度和效率上的有效均衡.

基于超平面分割的局部敏感哈希方法需要有恰当的法向量以及合适的偏移量才能发挥最好的效果. 本文通过随机产生的法向量具有较大的随机性, 下一步将结合数据的分布特性进行更加有针对性的法向量选择, 以提高基于数据检索的精度. 此外, 还将从利用用户检索数据特征出发探索有效的哈希函数构造方法, 实现向应用为中心的转变.

References:

[1] Zhang SC, Li XL, Zong M, *et al*. Learning *k* for *k*NN classification. ACM Trans. on Intelligent Systems and Technology, 2017, 8(3): 43:1−43:19. [doi: 10.1145/2990508].

[2] Arampatzis A, Kalamatianos G. Suggesting points-of-interest via content-based, collaborative, and hybrid fusion methods in mobile devices. ACM Trans. on Information Systems, 2017, 36(3): 23:1−23:28. [doi: 10.1145/3125620]

[3] Zhang J, Tang J, Ma C, *et al*. Fast and flexible top-*k* similarity search on large networks. ACM Trans. on Information Systems, 2017, 36(2): 13:1−13:30. [doi: 10.1145/3086695]

[4] BentleyJL. Multidimensional binary search trees used for associative searching. Communications of ACM, 1975, 18(9): 509−517.

[5] Beckmann N, Kriegel H, Schneider R, *et al*. The R*-tree: An efficient and robust access method for points and rectangles. SIGMOD Rec., 1990, 19(2): 322−331. [doi: 10.1145/93605.98741]

[6] Katayama N, Satoh S. The SR-tree: An index structure for high-dimensional nearest neighbor queries. SIGMOD Record, 1997, 26(2): 369−380.

[7] Indyk P, Motwani R. Approximate nearest neighbors: Towards removing the curse of dimensionality. In: Vitter J, ed. Proc. of the 30th Annual ACM Symp. on the Theory of Computing. New York: ACM, 1998. 604−613. [doi: 10.1145/276698.276876]

[8] Gionis A, Indyk P, Motwani R. Similarity search in high dimensions via hashing. In: Atkinson MP, Orlowska ME, Valduriez P, Zdonik SB, Brodie ML, eds. Proc. of the 25th Int'l Conf. on Very Large Data Bases. San Francisco: Morgan Kaufmann Publishers, 1999. 518−529.

[9] Charikar MS. Similarity estimation techniques from rounding algorithms. In: Reif J, ed. Proc. of the 34th Annual ACM Symp. on Theory of Computing. New York: ACM, 2002. 380−388.

[10] Datar M, Immorlica N, Indyk P, *et al*. Locality-sensitive hashing scheme based on *p*-stable distributions. In: Snoeyink J, Boissonnat JD, eds. Proc. of the 20th Annual Symp. on Computational Geometry. New York: ACM, 2004. 253−262. [doi: 10.1145/997817.997857].

[11] Wang XJ, Zhang L, Jing F, *et al*. Annosearch: Image auto-annotation by search. In: Proc. of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition. IEEE, 2006. 1483−1490.

[12] Kulis B, Grauman K. Kernelized locality-sensitive hashing for scalable image search. In: Proc. of the 12th Int'l Conf. on Computer Vision. IEEE CS, 2009. 2130−2137.

[13] Ji JQ, Li JM, Yan SC, *et al*. Super-bit locality-sensitive hashing. In: Pereira F, Burges CJC, Bottou L, eds. Proc. of the 25th Int'l Conf. on Neural Information Processing Systems. Curran Associates, 2012. 108−116.

[14] Gong Y, Lazebnik S. Iterative quantization: A procrustean approach to learning binary codes. In: Proc. of the 2011 IEEE Conf. on Computer Vision and Pattern Recognition. IEEE CS, 2011. 817−824.

[15] Jin ZM, Li C, Lin Y, *et al*. Density sensitive hashing. IEEE Trans. on Systems, Man, and Cybernetics, 2014, 44(8): 1362−1371.

[16] Liu H, Ji RR, Wang JD, *et al*. Ordinal constraint binary coding for approximate nearest neighbor search. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2019, 41(4): 941−955.

[17] Li YQ, Xiao RL, Wei X, *et al*. GLDH: Toward more efficient global low-density locality-sensitive hashing for high dimensions. Information Science, 2020, 533: 43−59. [doi: 10.1016/j.ins.2020.04.046]

[18] Cao ZJ, Long MS, Wang JM, *et al*. HashNet: Deep learning to hash by continuation. In: Proc. of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition. IEEE, 2017. 5609−5618.

[19] Dong Y, Indyk P, Razenshteyn I, *et al*. Learning space partitions for nearest neighbor search. arXiv: 1901.08544, 2019.

[20] Chiu CY, Prayoonwong A, Liao YC. Learning to index for nearest neighbor search. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2019, 42(8): 1942−1956.

[21] Sapkota M, Shi XS, Xing FY, *et al*. Deep convolutional hashing for low-dimensional binary embedding of histopathological images. IEEE Journal of Biomedical and Health Informatics, 2019, 23(2): 805−816.

[22] Arya S, Mount DM. Approximate nearest neighbor queries in fixed dimensions. In: Ramachandran V, ed. Proc. of the 4th Annual ACM/SIGACT-SIAM Symp. on Discrete Algorithms. PA: Society for Industrial and Applied Mathematics, 1993. 271−280.

[23] Jégou H, Douze M, Schmid C. Product quantization for nearest neighbor search. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2011, 33(1): 117−128.

[24] Weiss Y, Torralba A, Fergus R. Spectral hashing. In: Koller D, Schuurmans D, Bengio Y, Bottou L, eds. Proc. of the 21st Int'l Conf. on Neural Information Processing Systems. Curran Associates, 2008. 1753−1760.

[25] Lechervy A, Gosselin PH, Precioso F. Boosting kernel combination for multi-class image categorization. In: Proc. of the 19th IEEE Int'l Conf. on Image Processing. IEEE, 2012. 1893−1896.

[26] Huang Y, Guan Y. Laplacian hashing for fast large-scale image retrieval and its applications for midway processing in a cascaded face detection structure. Multimedia Tools and Applications, 2016, 75(23): 16315−16332.

[27] Zhang S, Huang J, Xiao RL, *et al*. Toward more efficient locality-sensitive hashing via constructing novel hash function cluster. Concurrency and Computation Practice and Experience. 2021, e6355:1−21. [doi: 10.1002/cpe.6355]

[28] Ghosh S. Kernel Smoothing Principles, Methods and Applications. Wiley, 2018. 19−21.

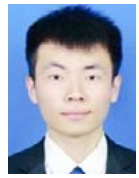
[29] Gilbert AC, Kotidis Y, Muthukrishnan S, *et al*. QuickSAND: Quick summary and analysis of network data. DIMACS Technical Report, 2001-43, AT&T Labs-Research, 2001.

[30] Qi Li, Jeffrey SR. Nonparametric Econometrics: Theory and Practice. Princeton University Press, 2006. 14−15.

张仕(1977－), 男, 博士, 副教授, CCF 高级会员, 主要研究领域为机器学习, 流计算, 大数据算法研究.

潘淼鑫(1987－), 女, 讲师, 主要研究领域为机器学习, 大数据检索.

赖会霞(1975－), 女, 讲师, 主要研究领域为机器学习, 大数据检索.



张路路(1996－), 男, 硕士生, 主要研究领域为人工智能技术及应用.

肖如良(1966－), 男, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为系统安全工程, 机器学习, 计算智能, 大数据技术.

陈伟林(1996－), 男, 硕士生, 主要研究领域为机器学习, 大数据检索.