软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(Suppl.(2)):228−238

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

基于可扩展 LSH 的高维动态数据索引*

胡海苗, 姜 帆

(数字媒体北京市重点实验室(北京航空航天大学),北京 100191)

通讯作者: 胡海苗, E-mail: hu@buaa.edu.cn

摘 要: 提出了一种可扩展的局部敏感哈希索引(SLSH),以解决高维动态数据索引中,由于数据集大小及分布特征无法确定而导致索引效率降低的问题.SLSH 架构于 E^2LSH 之上,继承了其对高维数据索引速度快,并可直接对欧式空间上的数据点进行索引的特点.为了使得哈希索引具有动态的相似性区分能力,SLSH 修改了 E^2LSH 的哈希族,通过哈希桶容量约束自适应调节哈希参数.因此对于分布密度动态变化的数据空间,SLSH 也能够给出鲁棒的划分.

关键词: 相似性检索;近似最近邻搜索;可扩展局部敏感哈希;动态高维数据索引

中文引用格式: 胡海苗,姜帆.基于可扩展 LSH 的高维动态数据索引.软件学报,2015,26(Suppl.(2)):228−238. http://www.jos.org.cn/1000-9825/15033.htm

英文引用格式: Hu HM, Jiang F. Scalable locality sensitive hashing scheme for dynamic high-dimensional data indexing. Ruan Jian Xue Bao/Journal of Software, 2015,26(Suppl.(2)):228−238 (in Chinese). http://www.jos.org.cn/1000-9825/15033.htm

Scalable Locality Sensitive Hashing Scheme for Dynamic High-Dimensional Data Indexing

HU Hai-Miao, JIANG Fan

(Beijing Key Laboratory of Digital Media (BeiHang University), Beijing 100191, China)

Abstract: A scalable locality sensitive hashing (SLSH) scheme is proposed to solve the problem of indexing high-dimensional data for dynamic datasets. The dynamic property destabilizes the size of the dataset, fuzzes up the tendency of data distribution, and conduces to the retrogression of retrieval performance. SLSH inherits two very convenient properties from the novel E2LSH that SLSH can rapidly work on data that is extremely high-dimensional and directly works on Euclidean space. For the purpose of adaptively fit the dynamic data distribution, the original hash family in E2LSH is altered for SLSH. A constraint of hash bucket capacity is applied for the hash parameters adjustment. As a result, SLSH provides robust partitions in the high-dimensional space for the dynamic data.

Key words: similarity search; approximate nearest neighbor search; scalable locality sensitive hashing; high dimensional dynamic data indexing

相似性检索(similarity search)[1−4]的目的在于从指定目标集合(如文档、图像、音频、视频等)中检索出与给定样本相似的目标,其中待检索的目标以及样本均以指定特征空间(feature space)下的高维数据点表示.因此相似性检索则在相应度量空间(metric space)中搜索样本点的最近邻作为检索结果.相似性检索有着广泛的应用,是信息检索、信息压缩、图像/视频数据库、数据挖掘与分析、机器学习、模式识别等领域的研究热点.随着大数据时代的来临,数据呈现爆炸式的增长,一方面,数据量在不断增加,另一方面,数据本身也在不断动态变化.例如,Web 搜索引擎需要在数以万计不断更新的网页中判断与某个给定页面相似的页面;大型的图像检索系统需要对不断更新的图像进行相似性检索.在此背景下,数据的高维特性、大规模特性、动态性给相似性检索带来了极大的挑战.

* 基金项目: 国家自然科学基金(61370121); 国家高技术研究发展计划(863)(2014AA015102)
收稿时间: 2015-05-15; 定稿时间: 2015-10-12

相似性检索的关键在于对待检索的目标建立有效的相似性索引.相似性索引通过对待检索的目标进行预处理,使得相似的目标被预先划分,从而在对给定样本进行检索时,只需要比对相似性索引中给出的可能相似的目标.对于大规模数据集来说,相似性索引的方法极大地减少了相似性检索的比对次数并减少了 I/O,让相似性检索在大规模数据集中的应用成为可能.

为了突破由于维数灾难造成的算法时间复杂度瓶颈,近似最近邻(approximate nearest neighbor)算法受到了越来越多的关注[5−8],并被广泛应用于相似性检索中.其中由 Indyk 等人最先提出的局部敏感哈希(locality sensitive hashing,简称 LSH)方法[9]由于其准确、高效的特点受到了广泛的关注.局部敏感哈希的基本思想是通过一组来自同一个哈希族(Hash family)的哈希函数对目标进行映射,使得相似目标比不相似目标有更大概率发生冲突,从而被映射到同一个哈希桶(Hash bucket)中.由于 Jaccard 系数空间、汉明空间、l_1 与 l_2 范式空间等很多度量空间有其对应的哈希函数,LSH 则可以有效地充当相似性索引完成相似目标预处理.

主流的局部敏感哈希方法主要包括数据依赖(data dependent)与数据无关(data independent)两种[10].传统的局部敏感哈希是数据无关的哈希方法,对原始特征空间作均匀划分.因此,针对有分布趋向性的数据集时,数据无关的哈希方法由于数据分布的密度不均,将会导致高密度区域的哈希桶变得过于臃肿,而低密度的部分变得稀疏,从而降低了索引效率.数据依赖的局部敏感哈希方法[11−14]针对此现象通过学习数据集的分布从而给出较好的划分的哈希函数,以达到针对数据密度的哈希索引.然而基于数据学习的方法破坏了传统局部敏感哈希的数据无关性,使得索引不具备普适性.

对于动态变化的数据集,采用固定哈希编码方法的局部敏感哈希方法对数据的动态性支持有限,无法很好地适应数据集的动态变化.一方面,针对数据集由于动态变化造成的数据分布的改变,数据依赖的局部敏感哈希方法受限与初始数据集的分布特性,无法保证其可持续的有效性.另一方面,数据无关的局部敏感哈希虽然在原理上能够支持数据集的动态变化,但若数据集的大小发生了较大的改变,那么其相应的哈希参数(比如哈希编码长度等)需要随之调整,从而导致需要对整个数据库进行重新索引.

针对以上问题有学者[15,16]提出结合动态哈希(dynamic hashing)框架[17],将汉明空间上的局部敏感哈希进行动态编码,通过前缀树管理每个哈希编码位,并最终在检索时取出前缀重合最多的数据点作为候选集.该方法虽然很好地解决了局部敏感哈希针对动态数据集的局限性.但是,该方法受限于汉明空间,并且调用的哈希函数次数过多(需要多少位的哈希编码就需要多少个哈希函数)降低了哈希索引的效率.

综合以上对数据维数、数据分布、数据空间以及数据动态变化的分析,本文提出了面向高维动态数据的可扩展局部敏感哈希索引(scalable locality sensitive hashing,简称 SLSH).首先,SLSH 继承了 E^{2}LSH[18,19]方法的优势,能够快速索引高维数据,并可以直接在欧式空间(即 l_2 范数的赋范空间)对 l_2 范数进行索引,避免了汉明嵌入的过程.考虑到欧式空间是当前特征比较中研究最为广泛的空间,直接在欧式空间上进行索引使得度量空间更加直观、运算过程更加简便.

其次,为了能够适应数据数量、数据分布动态变化的高维数据,SLSH 修改了 E^{2}LSH 的哈希族,使得哈希参数受哈希桶容量约束,能动态地调节哈希参数、修改哈希编码长度,从而使得哈希索引具有动态的相似性区分能力.为了节省空间开销,SLSH 不索引空的哈希桶,以树型结构管理哈希键-值对(key-value pair)以平衡索引在检索、管理上的时空开销.实验表明,与同类算法相比,SLSH 能够有效平衡算法速度与准确度之间的关系,在 $O(dn+n^{1+\rho})$的空间复杂度与 $O(dn^{\rho}\log_{1/p_2} n)$ 的时间复杂度内给出鲁棒的索引结果.

本文第 1 节介绍相关工作.第 2 节给出本文算法的理论基础与设计方法.第 3 节给出算法复杂度分析.第 4 节对全文进行总结并讨论当前存在的问题与进一步的研究方向.

1 相关工作

1.1 相关理论与定义

定义为空间 $\Re^d$ 上的 l_p 范数,对于任意的点 $\boldsymbol{v}\in\Re^d$,定义 $\|\vec{\mathbf{v}}\|_p$ 为 $\vec{\mathbf{v}}$ 的 l_p 范数.定义以 $\boldsymbol{X}$ 为元素集合,D 为度量

准则的度量空间为 $\mathcal{M}=(\boldsymbol{X},d)$, $\boldsymbol{v}\in\boldsymbol{X}$.那么,在 $\mathcal{M}$ 中以 $\boldsymbol{v}$ 为中心的球域为 $B(\boldsymbol{v},r)=\{\boldsymbol{q}\in\boldsymbol{X}\mid d(\boldsymbol{v},\boldsymbol{q})\leqslant r\}$.令 $c=1+\varepsilon>1$, $\varepsilon>0$,那么本文针对 (R,c)-近邻问题进行研究.

定义 1[9]. 对于元素域为 S,值域为 U 的局部敏感哈希族为 $\mathcal{H}=\{h:S\to U\}$,对于查询样本 $\boldsymbol{q}$,满足:(1) 若对于任意 $\boldsymbol{v}\in B(\boldsymbol{q},r_1)$,有 $\boldsymbol{q}$ 与 $\boldsymbol{v}$ 发生哈希值冲突的概率 $\Pr_{\mathcal{H}}[h(\boldsymbol{q})=h(\boldsymbol{v})]\geqslant p_1$;(2) 若对于任意 $\boldsymbol{v}\notin B(\boldsymbol{q},r_2)$,有 $\Pr_{\mathcal{H}}[h(\boldsymbol{q})=h(\boldsymbol{v})]\leqslant p_2$,那么称此哈希族 $\mathcal{H}$ 是 (r_1,r_2,p_1,p_2)-敏感的.对于一个有意义的局部敏感哈希方法,则满足需满足 $p_1>p_2$ 且 $r_1<r_2$.

定理 1[9,19]. 对于一个以 D 为度量准则的哈希族 $\mathcal{H}$ 是 (r_1,r_2,p_1,p_2)-敏感的,那么存在一个 (R,c)-近邻算法使得在 D 下,使得算法的空间复杂度为 $O(dn+n^{1+\rho})$,时间复杂度为 $O(dn^{\rho}\log_{1/p_2}n)$,且:

$$\rho=(\ln p_1)/(\ln p_2) \tag{1}$$

定义 2[19]. 对于实数集 $\mathcal{R}$ 上的分布 $\mathcal{D}$,有 $p\geqslant 0$,对于任意实数 $\{v_1,\ldots,v_n\}$ 与取自 $\mathcal{D}$ 的独立随机变量 $\boldsymbol{X}=\{x_1,\ldots,x_n\}$ 满足 $\sum_{i=1}^{n}v_ix_i$ 与 $\left(\sum_{i=1}^{n}|v_i|^p\right)^{1/p}$ 具有相同的分布,则称 $\mathcal{D}$ 是一个 p-稳定分布.其中柯西分布是 $p=1$ 的稳定分布实例,高斯分布是 $p=2$ 的稳定分布实例.

1.2 基于p-稳定分布的局部敏感哈希

基于 p-稳定分布的局部敏感哈希方法是一种数据无关的范式空间上的局部敏感哈希方法,首先由 Datar 于文献[19]中提出,并由 Andoni 于文献[20]进一步归纳完善,称之为 E^{2}LSH.

E^{2}LSH 针对 (R,c)-近邻问题定义,令局部敏感哈希中 $r_1=R$, $r_2=cR$,并根据 p-稳定分布的性质以点积投影 $(\alpha\cdot\mathbf{v})$ 来估计 l_p 范数,故设计哈希函数为 $h_{\alpha,b}(\boldsymbol{v})=\lfloor(\alpha\cdot\boldsymbol{v}+b)/r\rfloor$.其中 α 是来自 p-稳定分布的一组 d 维独立随机变量,r 为定义的一个哈希函数的收缩系数,b 为取自 $[0,r)$ 的一个服从均匀分布的随机变量,$\lfloor\bullet\rfloor$ 为取整函数.那么哈希碰撞的概率:

$$p(c)=\Pr_{\mathcal{H}}[h(\boldsymbol{q})=h(\boldsymbol{v})]=\int_0^r\frac{1}{c}f_p\left(\frac{x}{c}\right)\left(1-\frac{x}{r}\right)\mathrm{d}x \tag{2}$$

其中, f_p 为 p-稳定分布的概率密度函数.故对于 (R,c)-近邻问题有 $p_1=p(1)$, $p_2=p(2)$, $c=r_2/r_1$.

由于单个局部敏感哈希函数 $h_{\alpha,b}(\boldsymbol{v})$ 的区分能力较弱,为了令 $h_{\alpha,b}(\boldsymbol{v})$ 具有更高的 p_1 与更低的 p_2,E^{2}LSH 级联了多个哈希函数以生成较长的哈希编码.令 $\mathcal{G}=\{g:S\to U^k\}$ 为级联的哈希函数组,其中 k 为级联的哈希函数个数,那么 $g(\boldsymbol{v})=(h_1(\boldsymbol{v}),\ldots,h_k(\boldsymbol{v}))$, $h_i\in\mathcal{H}$ 为其中的一个级联的二级哈希函数.同时为了弱化单个哈希表的随机性,E^{2}LSH 从 $\mathcal{G}$ 中取 L 个哈希表 $g_1,\ldots,g_L$,将任意 $\boldsymbol{v}\in\boldsymbol{P}$ ($\boldsymbol{P}$ 是大小为 n 的输入数据集)存储到 L 个哈希桶中 $g_1(\boldsymbol{v}),\ldots,g_L(\boldsymbol{v})$.在检索时从 $g_1,\ldots,g_L$ 中取出前 $3L$ 个元素,根据[19,20]中证明,有

$$k=\log_{1/p_2}n \tag{3}$$

$$L=n^{\rho} \tag{4}$$

那么对于检索样本 $\boldsymbol{q}$,存在 $\boldsymbol{v}^*\in B(\boldsymbol{q},r_1)$ 使得 $g_j(\boldsymbol{v}^*)=g_j(\boldsymbol{q})$, $j=1,\ldots,L$ 为恒定概率事件,且在 $\boldsymbol{P}-B(\boldsymbol{q},r_2)$ 中与 $\boldsymbol{q}$ 哈希冲突的元素数量少于 $3L$.

根据上述原理可知,由于 E^{2}LSH 以点积投影 $(\alpha\cdot\boldsymbol{v})$ 来估计 l_p 范数,根据文献[21]介绍可知, $(\alpha\cdot\boldsymbol{v})$ 是可线性组合的,例如 $\alpha\cdot(\boldsymbol{v}_1-\boldsymbol{v}_2)=\alpha\cdot\boldsymbol{v}_1-\alpha\cdot\boldsymbol{v}_2$.因此根据哈希函数 $h_{\alpha,b}(\boldsymbol{v})=\lfloor(\alpha\cdot\boldsymbol{v}+b)/r\rfloor$ 可知,以哈希值 $h_{\alpha,b}(\boldsymbol{v})$ 对原始空间 $\mathfrak{R}^d$ 的划分是一个均匀、线性的划分.因此 E^{2}LSH 通过级联哈希函数的方法是一个组合多个局部敏感的均匀线性分割,以随机过程给出一个局部敏感的不均匀的分割,从而达到有较高区分性的目的.

2 可扩展局部敏感哈希

本文提出的可扩展局部敏感哈希索引针对 E^{2}LSH 中无法动态调整哈希参数的局限性,以哈希索引能否将原始数据集 $\boldsymbol{P}$ 进行均匀分割作为准则,进行动态地建立索引,从而平衡多个哈希编码的长度,使得哈希索引能够动态地适应数据集的分布.

2.1 可扩展局部敏感哈希族

在一个确定参数的 E^{2}LSH 中,其最终性能的好坏取决于算法是否真正地将原始数据集 $\boldsymbol{P}$ 打散.由于哈希表是局部敏感的,同一个哈希桶中的元素在很大概率上是相似的,那么将原始数据集 $\boldsymbol{P}$ 打散则简单地意味着每个哈希桶 $g_j,\ j=1,...,L$ 中保留有较少的元素个数.考虑两个极端的情况,若哈希表完全无法将 $\boldsymbol{P}$ 打散,那么所有数据点都将集中在一个哈希桶中,哈希检索就退化为线性搜索;若哈希表将$\boldsymbol{P}$过分打散,那么每个数据点都将占据一个哈希桶,则哈希检索则会经常返回空候选集.

从概率角度分析,针对确定参数的 E^{2}LSH 与确定的原始数据集 $\boldsymbol{P}$,由于局部敏感哈希的特点,E^{2}LSH 将 $\boldsymbol{P}$ 打散是一个恒定概率的事件;而针对一个动态的数据集 $\boldsymbol{P}$,由于数据的变动是不确定的,因此固定参数的 E^{2}LSH 能否将 $\boldsymbol{P}$ 打散则是一个不确定事件.从而针对动态的数据集,E^{2}LSH 的局限性就变得十分明显.

从数据分布角度分析,E^{2}LSH 的局部敏感哈希函数对原始特征空间作均匀划分,因此针对有分布趋向性的数据集时,由于数据分布的密度不均将会导致高密度区域的哈希桶变得过于臃肿,而低密度区域的哈希桶过于稀疏,从而降低了索引效率.

由对定理 1 可知,ρ 是影响的局部敏感哈希函数的区分能力的评估参数.根据文献[19]中分析,ρ 的大小对 r 的变化并不敏感,因此考虑引入 k 个哈希函数增强哈希表的区分能力.当查询点落在密度较高的区域时,则应当有更小的 c 以返回较小的候选集.根据式(1)、式(2),当 r 固定时,ρ 随着 c 增大而递减,当 c 正向趋近于 1 时,ρ 则趋近于 1.同时由 $k=\log_{1/p_2} n$ 可知,$\rho=-k\cdot\ln p_1/\ln p_2$,因此对于动态变化的数据集,为保持 ρ 不变,随着 n 增加,k 以 $O(\log n)$ 速度增加.同时对于固定的 c 有 k 随着与 r 的增长呈递减趋势,如图 1 所示.

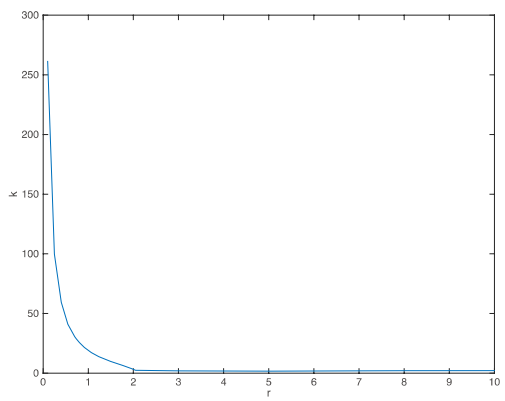


图 1 k 与 r 的关系曲线

因此一个 r 值可扩展伸缩的 E^{2}LSH 哈希索引,则可适应动态的数据集不确定的数据分布.对于取自 E^{2}LSH 的局部敏感哈希族$\mathcal{H}$,记$\mathcal{H}$ (r)为其可扩展局部敏感哈希族,则对于任意 $\boldsymbol{v}\in S$,有$\mathcal{H}(\boldsymbol{r})=\{h(\boldsymbol{v}|\boldsymbol{r}):S\to U\}$.由于 r 值的不同,因此 $h(\boldsymbol{v}|r_i)\in\mathcal{H}$ $(\boldsymbol{r})$具有不同的区分能力.

2.2 哈希桶容量约束

由于过于臃肿的哈希桶将导致哈希检索退化为线性搜索.在有确定的检索样本、哈希索引与原始数据集的情况下,哈希检索的时间复杂度为 $O(d+Md)$.其中 $O(d)$ 为计算哈希值的耗时,M 为哈希检索的返回的候选集的大小(即被找到的哈希桶中元素的数量),$O(Md)$ 则为在候选集中进行线性搜索的耗时.因此影响哈希索引检索速度的主要因素是哈希桶的大小.

可扩展局部敏感哈希索引通过引入参数 m 来约束哈希桶容量的最大上限.结合 $\mathcal{H}(\boldsymbol{r})$ 的设计,若哈希桶冲突元素超过 m,则从 $\mathcal{H}(\boldsymbol{r})$ 中取出一个有更强区分能力的下级哈希函数,对桶中元素进行再哈希.因此索引查询运算的时间复杂度为 $O(dL+md)$, $m<M$.其中 L 为取出的哈希函数的个数.

为了使下一级哈希函数比现有哈希函数有更强的区分能力,那么则应该增加哈希编码的长度,即增大 k 的值.根据图 1 中的性质,r 则应为一个元素值递减的向量,满足 $r_{i+1}=f(r_i),1\leqslant i\leqslant K-1$.例如可以取 $f(r_i)=r_i/2$.

同时为了避免索引无限制向下递增,可以设置 $r_{\max}$ 或者 $K_{\max}$,约束哈希索引至有限层.根据[19]中的发现,在 c 值固定的情况下, ρ 随着 r 增长到某个固定点后趋于稳定.因此对于任意固定的 c,求得其初始稳定点.由于 ρ 与 r 的关系表达式(1)、式(2)较为复杂,往往无法求得显式解,因此可借助 MATLAB 等工具计算其近似值.详细的算法时空复杂度分析见附录 A.

2.3 可扩展索引结构

根据第 2.1 节中的可扩展局部敏感哈希族,设计哈希表 $\mathcal{T}(\boldsymbol{r})=\{t(\boldsymbol{r}):S\to U^k\}$.其中对于独立的哈希表 $t(\boldsymbol{r})=\{h_1(\boldsymbol{v}\,|\,r_1),...,h_K(\boldsymbol{v}\,|\,r_K)\}$,且有 $r_i<r_j$.对于 $t(\boldsymbol{r})$ 中哈希函数 $h(\boldsymbol{v}\,|\,r_i)\in\mathcal{H}(\boldsymbol{r})$ 的组织形式与 E^{2}LSH 不同,不通过二级哈希函数进行再次哈希映射,而将 K 个哈希函数进行分层组织(如图 2 所示),仅对超过容量约束的哈希桶进行再次哈希,其分割效果如图 3 所示.

在可扩展局部敏感哈希索引的树状结构中,叶子节点为元素被打散的哈希桶;中间节点为元素未被打散的哈希桶,我们称之为超哈希桶,且仅超哈希桶具有下级哈希函数,映射到区分能力更强的哈希桶或超哈希桶上.

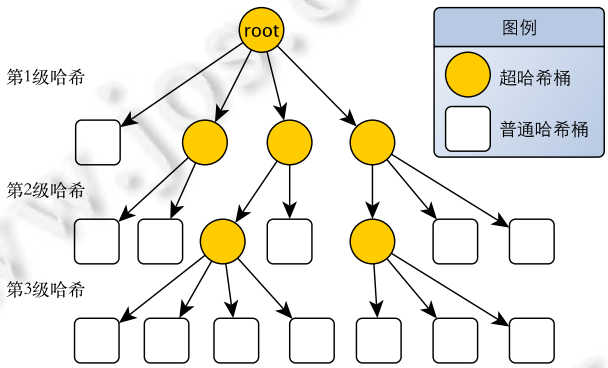
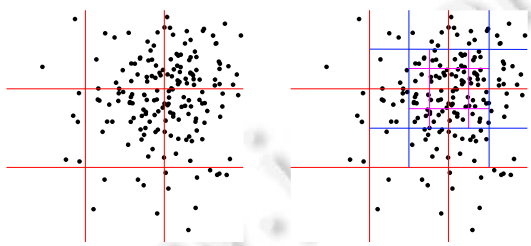


图 2 可扩展局部敏感哈希索引结构示意图



(a) LSH 分割 (b) SLSH 分割

图 3 扩展局部敏感哈希分割效果示意图

定义 $\mathcal{B}(\boldsymbol{q},i)$ 为对于元素 $\boldsymbol{q}\in S$,哈希函数值为 $\{h_1(\boldsymbol{q}\,|\,r_1),...,h_i(\boldsymbol{q}\,|\,r_i)\}$ 的哈希桶或超哈希桶,$|\mathcal{B}(\boldsymbol{q},i)|$ 为映射到 $\mathcal{B}(\boldsymbol{q},i)$ 上的元素个数,那么 SLSH 的查询过程则通过不断递增 i 直到遇到一个不为超哈希桶的 $\mathcal{B}(\boldsymbol{q},i)$,并将其中元素全部作为候选集返回.插入、删除的过程,则首先通过查询过程找到普通哈希桶 $\mathcal{B}(\boldsymbol{q},i)$,向其中插入新元素或者删除所需元素,并在修改后从 $\mathcal{B}(\boldsymbol{q},i-1)$ 位置调整哈希索引.哈希索引的调整算法如下.

算法 1. 从 $\mathcal{B}(\boldsymbol{v},i)$ 调整哈希索引.

IF $\mathcal{B}(\boldsymbol{v},i)$ 为超哈希桶

  IF $i>1$ 且 $|\mathcal{B}(\boldsymbol{v},i)|<m$

    删除下级哈希,将 $\mathcal{B}(\boldsymbol{v},i)$ 替换为普通哈希桶

  END

ELSE
IF $|\mathcal{B}(\boldsymbol{v},i)|>m$
对于所有 $\boldsymbol{u}\in|\mathcal{B}(\boldsymbol{v},i)|$,计算 $h(\boldsymbol{u}\,|\,r_{i+1})$
将 $\boldsymbol{u}$ 置入相应哈希桶 $\mathcal{B}(\boldsymbol{u},i+1)$ 中
END
END

3 实验与评估

为了更好地评估 SLSH 在不同类型数据集的效率,文本通过模拟了 3 种类型的数据集用于检索,分别为均匀分布数据集、高斯分布数据集以及动态变化数据集.3 个模拟数据集中的点均为维数 $d=500$ 维的高维向量,每个维度上的数据的取值范围在 $R=[-50,50]$ 的一个宽度的 100 的值域内.其中均匀分布数据集中,每个点取 R 上的均匀分布.高斯分布数据集,首先在 R 上随机生成一个 500 维的高维向量作为分布中心,围绕该中心生成随机偏差服从高斯分布 $norm(0,50)$ 的分布数据.动态数据集的生成方式与高斯分布数据集类似,区别在于:高斯分布数据集、均匀分布数据集在检索时取服从原始分布的 100 个 d 维向量作为检索数据,而动态数据集则取自新的高斯分布中心的随即向量作为检索数据.动态数据集.在实际应用中,随着数据集的动态变化,新的检索数据的特性可能与建立索引时的原始数据集不同,故借此模拟数据集的动态变化.

同时,为了衡量 SLSH 在真实的特征数据集上的检索表现,本文在法国国家计算机自动化研究所 IRISA/INRIA 提供的 ANN_SIFT1M[10]数据集上进行了检索准确性的实验.其中 ANN_SIFT1M 包含 100 万条 128 维 SIFT 特征,以及用于检索的 1 万条查询特征.ANN_SIFT1M 被广泛用于评估大规模特征数据检索方法的准确性及其性能.本文以前 N 个返回结果中包含正确结果的条数的百分比作为精确度作为衡量指标.

为了比较 SLSH 索引与其他索引在以上 4 种数据集上的表现,本文选取了几种基于不同理论实现的有代表性的近邻索引作为文本的对照组:

KD 树[22]:经典的精确近邻索引.

E^2LSH[18]:应用最广泛的数据无关 LSH 索引,为比较单个哈希表的数据区分能力,实验仅设置单个哈希表,每个哈希表 30 个哈希函数,并令 $r=100$.

DSH[12]:数据依赖 LSH 索引,令最长哈希编码长度为 32(即计算机 32 位整形数值的最大长度).

LSH 森林[15]:汉明空间上的动态哈希索引,设置单棵哈希前缀树,令前缀树最大深度为 1 000.

此外,实验还对比了线性搜索在上述情况中的效果,作为高维索引效率的评判基线.图 4(原始数据见附录 B)给出了不同索引算法在不同数据量的服从上述 3 种分布情况的数据集下的建立索引耗时、平均检索耗时,以及进行返回的前 20 个近邻中与检索样本的平均平方误差(MSE).其中数据量以 10 的次方递增.

在算法的时间开销上,根据附录 A 的分析,算法的时间复杂度会随着数据量的增长呈线性约束.与非哈希的方法相比,由于 KD 树索引与线性搜索受维数灾难影响,SLSH 的在数据检索上的时间开销有明显的提升;与基于哈希的方法相比,SLSH 没有 DSH 的学习过程,并且哈希函数运算的次数远小于 LSH 森林,建立索引的速度接近与仅通过树形结构对插入元素进行调整的 KD 树.由于 LSH 森林所需的哈希函数众多,因此在检索时,其效率较低,并不明显由于线性检索,反而与 KD 树接近.SLSH 在数据检索时间大于 E^2LSH.然而这是因为 E^2LSH 对参数十分敏感,由于实验对于不同大小的数据集均采用固定参数的 E^2LSH,因此 E^2LSH 经常无法给出有效的候选集(见附录 B 中的表 4、表 8、表 12),从而节省了对候选集进行再次筛选的过程.

在算法的准确性上,根据图 4(7)~图 4(9)以及图 5(原始数据见附录 C),线性搜索、KD 树等精确检索无疑给出了最小的 MSE 以及最高的检索精确度,针对数据集进行学习的 DSH 方法也同样给出了较好的检索结果.由于通过前缀树管理哈希编码,在不限制编码长度的情况下,LSH 森林的准确率非常接近于线性搜索.SLSH 则平衡了索引的准确性与时间复杂度,取了折衷的效果.

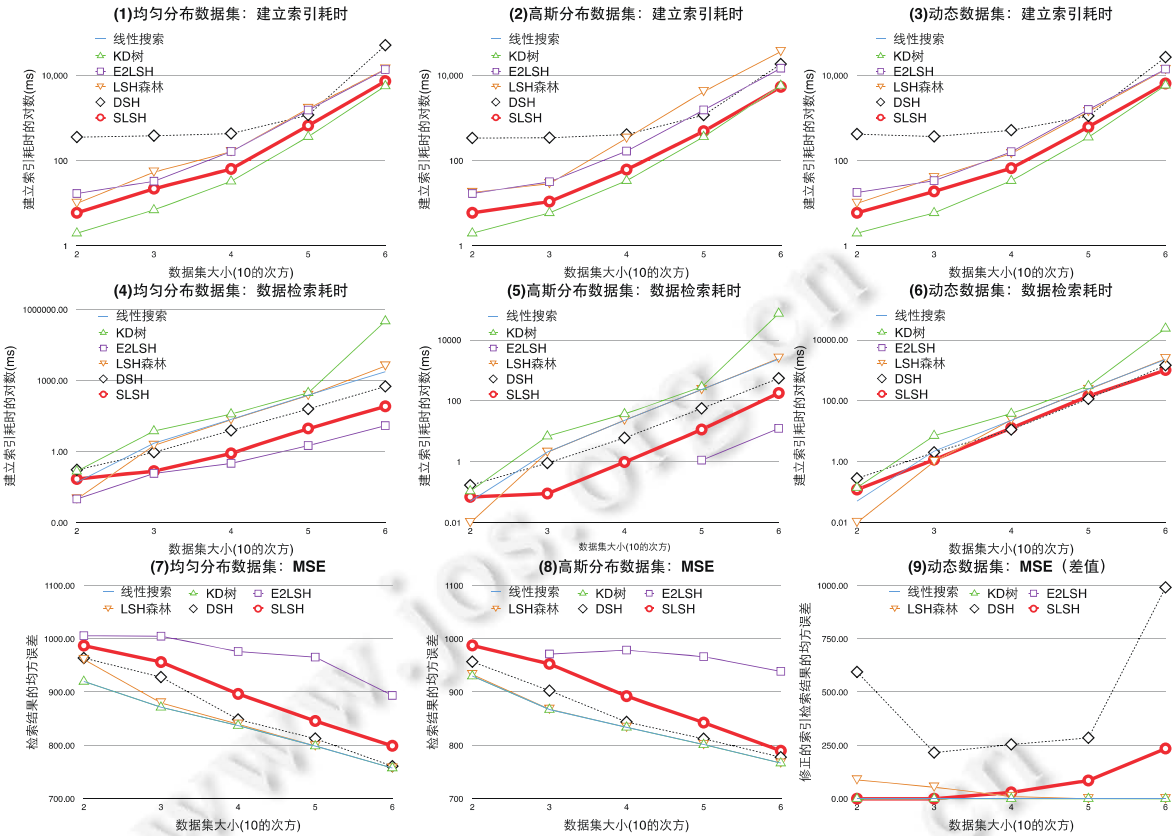


图 4　不同数据集上 3 种索引效率的比较

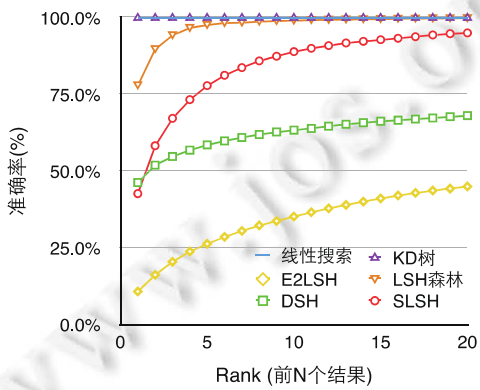


图 5　ANN_SIFT1M 数据集上检索准确性的比较

针对实验中的 3 个模拟数据集与真实数据集,固定参数的 E^{2}LSH 只在高密度的均匀分布数据集上有较好的表现,在有分布趋向性的高斯分布数据集以及动态数据集上,往往无法返回有效的候选集(导致图上若干测试下 E^{2}LSH 没有数据点).在动态数据集情况下,受影响最大的莫过于诸如 DSH 等基于学习的哈希方法.由于检索数据的分布与原始数据大相庭径,并且是随机产生,因此 DSH 在动态数据集上的准确性则变得无法估计.由于检索数据与原始数据的相似性差异较大,因此带来了较大的 MSE,为了更好地区分各个索引在动态数据集上的精确度,图 4(9)中以线性检索的 MSE 作为标准,将其与各个算法的 MSE 的差值作图.由图 4(9)以及图 5 可知,LSH 森林与 SLSH 均有较稳定的表现,然而查看 LSH 森林的检索耗时不难发现,LSH 森林在时间复杂度上已然退化

为线性搜索.同样地,由于检索数据与原始数据的差异性,在计算哈希前缀的过程中很早就遇到了哈希值的差异,生成了较短的哈希前缀.短哈希前缀使得大量匹配上的元素都被索引当作候选集,导致索引效率急剧下降.

4 总结与讨论

为了针对动态的高维数据进行有效索引,本文基于 E^{2}LSH 框架提出可扩展的局部敏感哈希索引 SLSH,为继承了 E^{2}LSH 直接针对欧式空间索引以及对高维数据能够快速准确索引的优点,通过改进原始局部敏感哈希族,使得参数能够动态适应当前数据集的特征,从而避免了 E^{2}LSH 受参数影响较大的缺点.通过证明,SLSH 在增大局部敏感哈希族的数据区分能力的同时,依然保持时空复杂性与数据集大小呈次线性关系,保持时间复杂度为 $O(dn^{\rho}\log_{1/p_2} n)$,空间复杂度为 $O(dn+n^{1+\rho})$.实验结果表明 SLSH 很好地平衡了索引的准确性与时间开销,并且对于复杂的动态数据集效果鲁棒.

由于式(1)、式(2)较为复杂,从而能得到关于 r 与 k 的显式解,两者关系仅有一个趋向性的描述,因此对于 SLSH 的动态调参不利.下一步的研究则针对该问题,探寻适合 SLSH 的调参策略,使得哈希索引的速度、准确率进一步提升.

References:

[1] Novak D, Batko M, Zezula P. Metric index: An efficient and scalable solution for precise and approximate similarity search. Information Systems, 2011,36(4):721−733.

[2] Jégou H, Schmid C, Harzallah H, Verbeek J. Accurate image search using the contextual dissimilarity measure. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2010,32(1):2−11.

[3] Cai JJ, Liu Q, Chen F, Joshi D, Tian Q. Scalable image search with multiple index tables. In: Proc. of the Int'l Conf. on Multimedia Retrieval. 2014. 407.

[4] Zhu CZ, Satoh S. Large vocabulary quantization for searching instances from videos. In: Proc. of the 2nd ACM Int'l Conf. on Multimedia Retrieval. 2012. 52.

[5] Gionis A, Indyk P, Motwani R. Similarity search in high dimensions via hashing. VLDB, 1999,99:518−529.

[6] Indyk P. Nearest neighbors in high-dimensional spaces. 2004.

[7] Xia H, Wu PC, Hoi SCH, Jin R. Boosting multi-kernel locality-sensitive hashing for scalable image retrieval. In: Proc. of the 35th Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval. 2012. 55−64.

[8] Nolen M, Lin KI. Approximate high-dimensional nearest neighbor queries using R-forests. In: Proc. of the 17th Int'l Database Engineering & Applications Symp. 2013. 48−57.

[9] Indyk P, Motwani R. Approximate nearest neighbors: Towards removing the curse of dimensionality. In: Proc. of the 30th Annual ACM Symp. on Theory of Computing. 1998. 604−613.

[10] Paulevé L, Jégou H, Amsaleg L. Locality sensitive hashing: A comparison of hash function types and querying mechanisms. Pattern Recognition Letters, 2010,31(11):1348−1358.

[11] Deng C, Deng HR, Liu XL, Yuan Y. Adaptive multi-bit quantization for hashing. Neurocomputing, 2015,151:319−326.

[12] Jin ZM, Li C, Lin Y, Cai D. Density sensitive hashing. IEEE Trans. on Cybernetics, 2014,44(8):1362−1371.

[13] Gao JY, Jagadish HV, Lu W, Ooi BC. DSH: Data sensitive hashing for high-dimensional *k*-nnsearch. In: Proc. of the 2014 ACM SIGMOD Int'l Conf. on Management of Data. 2014. 1127−1138.

[14] Weiss Y, Torralba A, Fergus R. Spectral hashing. In: Advances in Neural Information Processing Systems. 2009. 1753−1760.

[15] Bawa M, Condie T, Ganesan P. LSH forest: Self-Tuning indexes for similarity search. In: Proc. of the 14th Int'l Conf. on World Wide Web. 2005. 651−660.

[16] Jagadish HV, Ooi BC, Vu QH. Baton: A balanced tree structure for peer-to-peer networks. In: Proc. of the 31st Int'l Conf. on Very Large Data Bases. 2005. 661−672.

[17] Larson P. Dynamic hashing. BIT Numerical Mathematics, 1978,18(2):184−201.

[18] Andoni A, Indyk P. Near-Optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: Proc. of the 47th IEEE Symp. on Foundations of Computer Science. 2006. 459−468.

[19] Datar M, Immorlica N, Indyk P, Mirrokni VS. Locality-Sensitive hashing scheme based on p-stable distributions. In: Proc. of the 20th Annual Symp. on Computational Geometry. 2004. 253−262.

[20] Andoni A. Nearest neighbor search: The old, the new, and the impossible. Massachusetts Institute of Technology, 2009.

[21] Indyk P. Stable distributions, pseudorandom generators, embeddings and data stream computation. Journal of the ACM, 2006,53(3): 307−323.

[22] Robinson JT. The KDB-tree: A search structure for large multidimensional dynamic indexes. In: Proc. of the 1981 ACM SIGMOD Int'l Conf. on Management of Data. 1981. 10−18.

附录 A. SLSH 时间、空间复杂度推导

SLSH 检索最长哈希编码分支的时间复杂度满足,其中 K 为 SLSH 中哈希函数个数上限;检索最短哈希编码分支(长度为 1)的时间为 $\Omega(d)$.故 SLSH 满足定理 1 中对局部敏感哈希最近邻索引的时间复杂度要求.

证明:由式(1)、式(3)可知, $O(dK)=O(d\cdot\log_{1/p_2}n)$,

由于 SLSH 仅考虑使用单张哈希表,若与 E^{2}LSH 一样考虑多张哈希表,那么根据式(4)有

$$O(dKL)=O(d\cdot\log_{1/p_2}n\cdot n^{\rho}).$$ □

SLSH 的空间复杂度包括数据存储与索引结构存储两个部分.考虑空间利用的最坏情况,所有普通哈希桶被排在第 K 层节点上,且每个元素独占一个哈希桶,那么,此时消耗的空间为 $\Theta(dn+Kn)$;考虑空间利用最小的情况,所有数据被安排在第 1 层哈希桶内,此时的空间消耗为 $O(dn+O(1))$.因此对于一个有确定 K 值的 SLSH,其空间复杂度为 $O(dn+n)$.故 SLSH 满足定理 1 中对局部敏感哈希最近邻索引的空间复杂度要求.

证明:由于 SLSH 仅考虑使用单张哈希表,若与 E^{2}LSH 一样考虑多张哈希表,那么根据式(4)有

$$O(dn+nL)=O(dn+n\cdot n^{\rho})=O(dn+n^{1+\rho}).$$ □

表 1 均匀分布数据集:建立索引耗时 (ms)

数据量	10^2	10^3	10^4	10^5	10^6
线性搜索	–	–	–	–	–
KD 树	2	7	33	367	5 679
E^{2}LSH	17	33	164	1 540	13 753
LSH 森林	10	54	163	1 680	14 122
DLSH	358	386	433.7	1 179	51 211
SLSH	6	22	64	666	7 346

表 2 均匀分布数据集:数据检索耗时 (ms)

数据量	10^2	10^3	10^4	10^5	10^6
线性搜索	0.05	2.26	22.87	243.67	2 359
KD 树	0.15	7.46	38.44	306.91	32 7612
E^{2}LSH	0.01	0.12	0.32	1.8	12.5
LSH 森林	0.01	1.84	22.38	235.61	4 138.5
DLSH	0.17	0.95	7.86	63.17	569
SLSH	0.07	0.15	0.84	9.49	81.5

表 3 均匀分布数据集:均方误差 (mse)

数据量	10^2	10^3	10^4	10^5	10^6
线性搜索	920.1	871.7	837.3	798.8	757.56
KD 树	920.1	871.7	837.3	798.8	757.56
E^{2}LSH	1006	1005	976.3	965.7	893.8
LSH 森林	961.3	879.8	839.8	799.1	757.56
DLSH	964	928.3	848.7	812.8	761.05
SLSH	987.3	956.5	896.7	845.8	799.21

表 4 均匀分布数据集:检索返回空候选集次数

数据量	10^2	10^3	10^4	10^5	10^6
线性搜索	0	0	0	0	0
KD 树	0	0	0	0	0
E^{2}LSH	99	94	81	53	35
LSH 森林	0	0	0	0	0
DLSH	4	12	37	46	50
SLSH	0	0	0	0	0

表 5 高斯分布数据集:建立索引耗时 (ms)

数据量	10^2	10^3	10^4	10^5	10^6
线性搜索	–	–	–	–	–
KD 树	2	6	34	365	5 802
E^{2}LSH	17	32	168	1 551	14 772
LSH 森林	18	29	337	4 186	36 391
DLSH	341	347	413.26	1 173	18 513
SLSH	6	11	62	492	5 393

表 6 高斯分布数据集:数据检索耗时 (ms)

数据量	10^2	10^3	10^4	10^5	10^6
线性搜索	0.05	2.08	22.9	237.11	2 348.5
KD 树	0.11	6.96	37.23	288.79	74 895
E^{2}LSH	–	–	–	1.13	12.5
LSH 森林	0.01	2.06	23.29	235.08	2 509
DLSH	0.17	0.91	6.11	56.57	552
SLSH	0.07	0.09	0.98	11.45	182

表 7 高斯分布数据集:均方误差 (MSE)

数据量	10^2	10^3	10^4	10^5	10^6
线性搜索	930.24	867.35	833.96	801.58	766.86
KD 树	930.24	867.35	833.96	801.58	766.86
E^2LSH	–	971.52	978.89	966.6	938.94
LSH 森林	932.98	867.86	834.05	801.58	766.86
DLSH	957.08	902.89	844.02	812.47	777.98
SLSH	987.6	952.95	892.5	842.75	790.06

表 8 均匀分布数据集:检索返回空候选集次数

数据量	10^2	10^3	10^4	10^5	10^6
线性搜索	0	0	0	0	0
KD 树	0	0	0	0	0
E^2LSH	100	92	77	51	16
LSH 森林	0	0	0	0	0
DLSH	3	35	45	50	23
SLSH	0	0	0	0	0

表 9 均匀分布数据集:建立索引耗时 (ms)

数据量	10^2	10^3	10^4	10^5	10^6
线性搜索	–	–	–	–	–
KD 树	2	6	34	356	5 888
E^2LSH	18	34	162	1 589	14 056
LSH 森林	10	40	147	1 423	13 725
DLSH	421	371	518	1 119	26 911
SLSH	6	19	67	619	6 480

表 10 动态数据集:数据检索耗时 (ms)

数据量	10^2	10^3	10^4	10^5	10^6
线性搜索	0.05	2.18	22.34	240.94	2 232.34
KD 树	0.14	7.09	37.7	304.02	23 624.56
E^2LSH	–	–	–	–	–
LSH 森林	0.01	1	23.44	232.31	2 380.97
DLSH	0.28	2.01	11.19	114.62	1 459.28
SLSH	0.12	1.15	12.61	141.69	1 009.92

表 11 均匀分布数据集:均方误差 (MSE)

数据量	10^2	10^3	10^4	10^5	10^6
线性搜索	0	0	0	0	0
KD 树	0	0	0	0	0
E^2LSH	–	–	–	–	–
LSH 森林	88.37	54.37	10.44	0	0
DLSH	594.63	216.5	254.62	285.89	991.43
SLSH	0	0	29.97	85.72	235.34

表 12 均匀分布数据集:检索返回空候选集次数

数据量	10^2	10^3	10^4	10^5	10^6
线性搜索	0	0	0	0	0
KD 树	0	0	0	0	0
E^2LSH	100	100	100	100	100
LSH 森林	0	0	0	0	0
DLSH	0	0	0	0	0
SLSH	0	0	0	0	0

表 13 ANN_SIFT1M 数据集:检索准确度

Rank	线性搜索 (%)	KD 树 (%)	E^2LSH (%)	LSH 森林 (%)	DSH (%)	SLSH (%)
1	100	100	10.77	77.99	46.31	42.69
2	100	100	16.22	89.80	51.99	58.30
3	100	100	20.47	94.35	54.80	67.24
4	100	100	23.82	96.69	56.88	73.34
5	100	100	26.32	97.69	58.59	77.90
6	100	100	28.60	98.33	59.88	81.28
7	100	100	30.52	98.44	60.99	83.77
8	100	100	32.31	98.78	61.96	85.97
9	100	100	33.78	98.98	62.80	87.48
10	100	100	35.23	99.12	63.40	88.99
11	100	100	36.64	99.28	63.95	90.07
12	100	100	37.87	99.39	64.69	90.95
13	100	100	39.02	99.48	65.32	91.81
14	100	100	40.10	99.54	65.82	92.34
15	100	100	41.14	99.60	66.30	92.87
16	100	100	42.10	99.71	66.63	93.35
17	100	100	42.94	99.77	67.00	93.87
18	100	100	43.70	99.83	67.35	94.42
19	100	100	44.31	99.86	67.79	94.83
20	100	100	45.04	99.87	68.13	95.10

表 14 ANN_SIFT1M 数据集:其他指标

性能指标	线性搜索	KD 树	E^2LSH	LSH 森林	DSH	SLSH
建立索引耗时 (ms)	4 118	6 728	4 873	8 876	17 674	4 916
数据检索耗时 (ms)	1 120.55	2 632.42	4.59	657.92	228.58	58.05
均方误差	990 158.61	990 158.61	1 384 659.29	1 006 922.24	1 062 838.44	1 134 420.73
返回空集次数	0	0	294	0	582	0

表中建立索引耗时包括读取 ANN_SIFT1M 数据集的耗时.由于线性搜索仅有读取数据集的时间开销,可以线性搜索为基准判断其他索引耗时.

胡海苗(1983－),男,安徽绩溪人,博士,CCF 会员,主要研究领域为视频图像分析处理.

姜帆(1991－),男,硕士,主要研究领域为视频图像分析处理.