谭宗元, 王洪亚. 基于位置敏感哈希的高效近似最近邻检索研究[J]. 智能计算机与应用,2025,15(9):19-25. DOI:10.20169/j.issn.2095-2163.25041806

基于位置敏感哈希的高效近似最近邻检索研究

谭宗元, 王洪亚
(东华大学 计算机科学与技术学院, 上海 201620)

摘 要: 位置敏感哈希(Locality-Sensitive Hashing,LSH)是一种有效的随机化技术,广泛应用于众多机器学习任务中。然而,哈希计算的开销与数据维度成正比,因此在高维数据和使用大量哈希函数的场景下,往往成为性能瓶颈。本文在 l_2 范数下设计了一种简单而高效的 LSH 方法,称为 FastLSH。该方法通过结合随机采样与随机投影,将哈希时间复杂度从 $O(n)$ 降低至 $O(m)$ (其中 n 为数据维度, $m < n$ 为采样维度数)。更重要的是,FastLSH 保留了可证明的 LSH 性质。论文在多个真实和合成数据集上进行了针对最近邻搜索任务的全面实验。实验结果表明,FastLSH 在查询质量、空间开销和查询效率等方面与当前先进方法表现相当,同时在哈希函数计算上可实现最高达 80 倍的加速。

关键词: 位置敏感哈希; 机器学习; 随机采样; LSH 性质
中图分类号: TP399 **文献标志码**: A **文章编号**: 2095-2163(2025)09-0019-07

Research on efficient approximate nearest neighbor search based on Locality-Sensitive Hashing

TAN Zongyuan, WANG Hongya
(School of Computer Science and Technology, Donghua University, Shanghai 201620, China)

Abstract: Locality-Sensitive Hashing (LSH) is an effective randomized technique and widely used in many machine learning tasks. The cost of hashing is proportional to data dimensions, and thus the performance bottleneck often occurs when dimensionality is high and the number of hash functions involved is large. This paper designs a simple yet efficient LSH scheme, named FastLSH, under l_2 norm. By combining random sampling and random projection, FastLSH reduces the time complexity from $O(n)$ to $O(m)$ ($m < n$), where n is the data dimensionality and m is the number of sampled dimensions. Moreover, FastLSH has provable LSH property. The paper conducts comprehensive experiments over a collection of real and synthetic datasets for the nearest neighbor search task. Experimental results demonstrate that FastLSH is on par with the state-of-the-arts in terms of answer quality, space occupation and query efficiency, while enjoying up to 80× speedup in hash function evaluation.

Key words: Locality-Sensitive Hashing; machine learning; random sampling; LSH property

0 引 言

最近邻(NN)检索是机器学习中的一个重要问题,其在人脸识别、信息检索和重复检测等领域有着广泛的应用。最近邻检索的目的是在数据集 $\boldsymbol{D} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_N\}$ 中找到与给定查询 $\boldsymbol{u}$ 最相似的数据对象(或具有最小距离)。对于低维空间 ($n < 10$),流行的基于树的索引结构如 KD-树[1]、SR-树[2]等提供精确的解并提供令人满意的查询性能。然而,对于高维空间,基于树的索引结构遭受众所周知的维数诅咒,也就是说,得到的查询性能甚至比线性扫描更差[3]。为了解决这个问题,一种可行的方法是通过以准确性换取效率来提供近似结果[4]。

位置敏感哈希(LSH)是解决高维空间中近似最近邻(ANN)检索问题的一种有效的随机化技术。LSH 的基本思想是使用随机哈希函数将高维数据对象映射到低维空间的桶中,通过这种方法,相似对象映射到相同桶的概率比不相似对象的高。针对 l_2 范数的 LSH 机制(E2LSH)最初由文献[6]基于 p-稳定分布提出。由于其亚线性查询时间复杂度和对查询结果的理论保证,E2LSH 可谓是理论和实践中最流行的 ANN 检索算法之一,并且前人已经提出了许多变

作者简介: 谭宗元(1990—),男,博士研究生,主要研究方向:近似最近邻搜索。Email:1041689460@qq.com; 王洪亚(1976—),男,教授,主要研究方向:近似最近邻搜索。

收稿日期: 2025-04-18

体来实现更好的空间占用和查询响应时间[8-17]。

除了用于 ANN 检索外,LSH 还可以用在许多其他机器学习任务中[18-21]。这些 LSH 应用都涉及在整个数据集上计算 LSH 函数。以广泛使用的 E2LSH 为例,计算向量 $\boldsymbol{v}$ 的 k 个哈希值需要 $O(nk)$ 次计算,其中 n 为 $\boldsymbol{v}$ 的维数, k 为 LSH 函数的个数。对于典型的 ANN 检索任务, k 的取值往往在几百到几千之间,并且随着数据集的基数 (N) 不断增长,因为 E2LSH 所需的哈希数是 $O(N^{\rho})$ [6]。因此,在几乎所有基于 LSH 的应用中,哈希是主要的计算和资源瓶颈,尤其当数据以流方式出现和/或 LSH 数据结构必须重复构建时[16, 22]。

本文开发一种简单而高效的 LSH 方案(FastLSH),只需要随机采样和随机投影两个基本操作,并且具有比 E2LSH 更好的时间复杂度。此外,文中还推导 FastLSH 的碰撞概率表达式,并证明 FastLSH 与 E2LSH 渐近等价,这意味着本文提出的方案具有理想的 LSH 性质。为了进一步验证本文的方案性能,对最近邻搜索任务进行全面的实验,其中标准 LSH 由 FastLSH 所替代。实验结果表明,基于 FastLSH 的算法在查询精度和查询效率方面与业界领先的算法相当,同时在索引构建时间方面提供明显的端到端加速。

1 预备知识

令 $\boldsymbol{D}$ 表示欧式空间 $\mathbb{R}^n$ 中大小为 N 的数据集, $\boldsymbol{v} \in \boldsymbol{D}$ 表示数据向量和 $\boldsymbol{u} \in \mathbb{R}^n$ 表示查询向量。文中令 $\phi(x) = 1/\sqrt{2\pi}\exp(-x^2/2)$ 和 $\Phi(x) = \int_{-\infty}^{x} 1/\sqrt{2\pi}\exp(-x^2/2)\mathrm{d}x$ 分别表示标准正态分布 $N(0,1)$ 的概率密度函数(Probability Density Function, PDF)和累积分布函数(Cumulative Distribution Function, CDF)。

1.1 位置敏感哈希

位置敏感哈希(Locality Sensitive Hashing, LSH)是解决近似近邻检索问题的有效方法。在介绍 LSH 之前,首先对近似近邻(Approximate Near Neighbor, ANN)检索问题进行定义,其通常采用的形式如下:

定义 1 (c,R)-NN 检索问题 给定 n 维欧式空间 $\mathbb{R}^n$ 的数据集 $\boldsymbol{D}$ 和参数 $R > 0, \delta > 0$, (c,R)-NN 检索问题是构建一个数据结构使得对于任意的查询 $\boldsymbol{u}$ 以概率 $1-\delta$ 确保如果存在 $\boldsymbol{D}$ 中一个 $\boldsymbol{u}$ 的 R 近邻对象,那么在 $\boldsymbol{D}$ 中返回一些 $\boldsymbol{u}$ 的 cR 近邻对象。

在定义 1 中, $\boldsymbol{u}$ 的 R - 近邻对象为 $\boldsymbol{v}$ 使得 $\|\boldsymbol{v}-\boldsymbol{u}\|_2 \leqslant R$。针对 (c,R) - NN 检索问题,现流行的方法往往基于位置敏感哈希(LSH),这是一组函数,具有良好的特性,即在这些函数的域内,相似度(欧式距离近)高的对象比相似度低的对象在范围空间中碰撞的概率更高。形式化地,考虑 LSH 函数族 H 映射 $\mathbb{R}^n$ 到域 U。

定义 2 位置敏感哈希(LSH) 一个 LSH 函数族 $H = \{h: \mathbb{R}^n \to U\}$ 称为 (R, cR, p_1, p_2) - 敏感的,如果对于任意对象 $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$,

(1) 假设 $\|\boldsymbol{v}-\boldsymbol{u}\|_2 \leqslant R$, 那么 $\Pr_H[h(\boldsymbol{v}) = h(\boldsymbol{u})] \geqslant p_1$;

(2) 假设 $\|\boldsymbol{v}-\boldsymbol{u}\|_2 \geqslant cR$, 那么 $\Pr_H[h(\boldsymbol{v}) = h(\boldsymbol{u})] \leqslant p_2$;

为了使得 LSH 函数族有用,必须满足 $c > 1$ 以及 $p_1 > p_2$。

1.2 针对 l_2 范数的 LSH

文献[6]给出基于 p - 稳定分布的用于相似度为 $l_p(p \in (0,2])$ 范数的 LSH 族。当 $p = 2$ 时,产生众所周知的 l_2 范数 LSH 函数族(E2LSH)。该 LSH 函数定义如下:

$$h_{a,b}(\boldsymbol{v}) = \left\lfloor \frac{\boldsymbol{a}^{\mathrm{T}}\boldsymbol{v} + b}{w} \right\rfloor \tag{1}$$

其中, $\lfloor \cdot \rfloor$ 表示向下取整函数; $\boldsymbol{a} \in \mathbb{R}^n$ 表示一个 n 维的随机投影向量,其每一维的数值独立生成于标准正态分布 $N(0,1)$; b 表示一个实数、均匀地产生于区间 $[0,w]$, 这里桶宽 w 是一个重要的参数,用于调节 E2LSH 的性能。

对于 E2LSH,在 LSH 函数 $h_{a,b}(\cdot)$ 下任意数据对 $(\boldsymbol{v},\boldsymbol{u})$ 的碰撞概率计算如下:

$$p(s) = \Pr[h_{a,b}(\boldsymbol{v}) = h_{a,b}(\boldsymbol{u})] = \int_0^w f_{|sX|}(t)\left(1 - \frac{t}{w}\right)\mathrm{d}t \tag{2}$$

其中 $s = \|\boldsymbol{v}-\boldsymbol{u}\|_2$ 表示数据对 $(\boldsymbol{v},\boldsymbol{u})$ 的欧式距离; $f_{|sX|}(t)$ 表示正态分布 sX 绝对值的 PDF,这里 X 是服从 $N(0,1)$ 的随机变量。给定桶宽 w, 碰撞概率 $p(s)$ 是关于 s 的单调递减函数,这意味着函数 $h_{a,b}(\cdot)$ 满足 LSH 特性。

1.3 截断的正态分布

如果由于物理原因,需要使用正态分布来描述某一变量的随机变化,但该变量的取值必须严格限制在某一截断区间内、而非 $(-\infty, +\infty)$, 则推荐使用截断正态分布(Truncated Normal Distribution)。截断正态分布是通过对正态随机变量的取值进行下限、上限或双侧限制而得到的概率分布。设截断区

间为 (a_1,a_2)，则其概率密度函数可表示为：

$$\psi(x;\mu,\sigma^2,a_1,a_2)=\begin{cases}0, & x\leqslant a_1\\ \dfrac{\phi(x;\mu,\sigma^2)}{\Phi(a_2;\mu,\sigma^2)-\Phi(a_1;\mu,\sigma^2)}, & a_1<x<a_2\\ 0, & x\geqslant a_2\end{cases}\tag{3}$$

相应地 CDF 定义如下：

$$\Psi(x;\mu,\sigma^2,a_1,a_2)=\begin{cases}0, & x\leqslant a_1\\ \dfrac{\Phi(x;\mu,\sigma^2)-\Phi(a_1;\mu,\sigma^2)}{\Phi(a_2;\mu,\sigma^2)-\Phi(a_1;\mu,\sigma^2)}, & a_1<x<a_2\\ 1, & x\geqslant a_2\end{cases}\tag{4}$$

2　高效的 LSH 函数及 ANN 检索

2.1　高效的 LSH 函数族

本文提出一种新颖的 LSH 函数族，命名为 FastLSH。通过 FastLSH 计算哈希值涉及 2 个简单的步骤，即随机采样和随机投影。

(1)随机采样：在随机采样阶段，首先从 n 个维度中进行随机采样，这里从整数集 $\{1,2,\cdots,n\}$ 中随机采样 m 个样本构成多集 S。对于每个向量 $\boldsymbol{v}=\{v_1,v_2,\cdots,v_n\}$，如果 $i\in S$，那么将所有的 v_i 连接成一个 m 维的向量 $\tilde{\boldsymbol{v}}=\{\tilde{v}_1,\tilde{v}_2,\cdots,\tilde{v}_m\}$。现用一个简单例子展示如何生成 $\tilde{\boldsymbol{v}}$，假设有一个 5 维的向量 $\boldsymbol{v}=\{1,3,5,7,9\}$ 以及多集 $S=\{2,4,2\}$，那么通过 S 得到一个 3 维的向量 $\tilde{\boldsymbol{v}}=\{3,7,3\}$。从这里可以看到 $\boldsymbol{v}$ 中每个元素被选中的概率均为 m/n。接下来，继续重复使用符号 S 并用 $S(\cdot)$ 表示采样算子，那么对于每个向量 $\boldsymbol{v}\in\mathbb{R}^n$，可以得到 $\tilde{\boldsymbol{v}}\in\mathbb{R}^m=S(\boldsymbol{v})$，这里 $m<n$。

(2)随机投影：在随机投影阶段，投影值的计算与式(1)相同，区别在于式(1)使用向量 $\boldsymbol{v}\in\mathbb{R}^n$、而这里使用 $\tilde{\boldsymbol{v}}\in\mathbb{R}^m$，那么整体的哈希函数形式化定义如下：

$$h_{\tilde{\boldsymbol{a}},\tilde{b}}(\boldsymbol{v})=\left\lfloor\frac{\tilde{\boldsymbol{a}}^{\mathrm{T}}S(\boldsymbol{v})+\tilde{b}}{\tilde{w}}\right\rfloor\tag{5}$$

其中，$\tilde{\boldsymbol{a}}\in\mathbb{R}^m$ 表示随机投影向量，其每个元素均从 $N(0,1)$ 中独立生成；$\tilde{w}$ 表示用户预先指定的常量；$\tilde{b}$ 表示从区间 $[0,\tilde{w}]$ 均匀生成的一个实数。该哈希函数 $h_{\tilde{\boldsymbol{a}},\tilde{b}}(\boldsymbol{v})$ 通过 $\tilde{\boldsymbol{a}}$ 映射一个 n 维的向量 $\boldsymbol{v}$ 到整数集。与 E2LSH 相比，FastLSH 将哈希的时间复杂度从 $O(n)$ 降到 $O(m)$。

2.2　FastLSH 用于 ANN 检索

鉴于其方法简单，FastLSH 可以轻松嵌入到任何现有的 LSH 应用中。这里将简要讨论如何将 FastLSH 应用于 ANN 检索任务。对于 ANN 检索，其标准 LSH 索引结构(E2LSH)构建如下：首先使用 k 个独立的 LSH 函数计算哈希码 $H(\boldsymbol{v})=(h_1(\boldsymbol{v}),h_2(\boldsymbol{v}),\cdots,h_k(\boldsymbol{v}))$，这里 $H(\boldsymbol{v})$ 表示 k 个 LSH 码的串联；然后建立一个哈希表将二元组 $(H(\boldsymbol{v}),\mathrm{ID}(\boldsymbol{v}))$ 添加到相应的桶中，这里 $\mathrm{ID}(\boldsymbol{v})$ 表示 $\boldsymbol{v}$ 在数据库 $\boldsymbol{D}$ 的 ID。为了提高精度，从 LSH 族中独立均匀且随机产生 L 组哈希函数 $H_i(\cdot)(i\in\{1,2,\cdots,L\})$ 得到 L 个哈希表。在这样的 ANN 检索框架中使用 FastLSH，仅仅将式(1)定义的 LSH 函数替换为式(5)的 LSH 函数。

当来临查询 $\boldsymbol{u}$，先要计算 $\{H_1(\boldsymbol{u}),H_2(\boldsymbol{u}),\cdots,H_L(\boldsymbol{u})\}$，然后检索所有 L 个桶以获得 $\boldsymbol{u}$ 的候选集。事实上，存在 2 种方法对这些候选对象进行处理，即近似的和精确的方法。对于近似的方法，这里只评估候选集中不超过 $3L$ 个对象。LSH 理论确保以恒定的概率返回 (c,R) - NN 对象。但在实际中，往往广泛使用精确的方法，原因在于该方法以评估候选集中所有对象为代价提供更高的查询精度。在精确的方法中，检索时间包含哈希时间与检测候选集所需的时间。

3　理论分析

3.1　碰撞概率的计算

给定的向量对 $(\boldsymbol{v},\boldsymbol{u})$，令 $s=\|\boldsymbol{v}-\boldsymbol{u}\|_2$。对于 n 个元素 $(v_i-u_i)^2\{i=1,2,\cdots,n\}$ 的集合，且均值和方差分别为 $\mu=(\sum_{i=1}^{n}(v_i-u_i)^2)/n$ 和 $\sigma^2=(\sum_{i=1}^{n}((v_i-u_i)^2-\mu)^2)/n$。在执行大小为 m 随机采样算子 $S(\cdot)$ 后，向量 $\boldsymbol{v}$ 和 $\boldsymbol{u}$ 分别变换为 $\tilde{\boldsymbol{v}}=S(\boldsymbol{v})$ 和 $\tilde{\boldsymbol{u}}=S(\boldsymbol{u})$，那么变换之后的向量对 $(\tilde{\boldsymbol{v}},\tilde{\boldsymbol{u}})$ 的平方距离为：

$$\tilde{s}^2=\sum_{i=1}^{m}(\tilde{v}_i-\tilde{u}_i)^2\tag{6}$$

通过中心极限定理，可得以下的引理：

引理 1　如果采样的数量 m 足够大，那么 m 个独立同分布的随机样本 $(\tilde{v}_i-\tilde{u}_i)^2\{i=1,2,\cdots,m\}$ 的和 $\tilde{s}^2$ 渐近收敛于均值为 $m\mu$、方差为 $m\sigma^2$ 的正态分布，即 $\tilde{s}^2\sim N(m\mu,m\sigma^2)$。

回顾下 $\tilde{\boldsymbol{a}}\in\mathbb{R}^m$ 表示随机投影向量，其每个元

素均从 $N(0,1)$ 中独立生成。通过 2-稳定分布,对于任意 2 个向量 $\boldsymbol{v}$ 和 $\boldsymbol{u}$, 向量之间投影差值 ($\boldsymbol{a}^{\mathrm{T}}\boldsymbol{v} - \boldsymbol{a}^{\mathrm{T}}\boldsymbol{u}$) 服从分布为 $\|\boldsymbol{v}-\boldsymbol{u}\|_2 X$, 即 sX, 这里 $X \sim N(0,1)$。值得注意的是,如式(2)所示, sX 的 PDF,即 $f_{sX}(x) = 1/s \cdot \phi(x/s)$, 在计算碰撞概率中是个重要的因素。因此,如果已知 $\tilde{s}X$ 的 PDF,那么在本节提出的 LSH 函数 $h_{\tilde{a},\tilde{b}}(\cdot)$ 下,将很容易推导向量对 ($\boldsymbol{v},\boldsymbol{u}$) 的碰撞概率。令 $f_{|\tilde{s}X|}(t)$ 表示 $\tilde{s}X$ 绝对值的 PDF,通过代替式(2)中的 $f_{|sX|}(t)$, 就可获得 FastLSH 的碰撞概率 $p(s,\sigma)$, 如定理所示。

定理 1 在哈希函数 $h_{\tilde{a},\tilde{b}}(\cdot)$ 下,任意两点发生碰撞的概率为:

$$p(s,\sigma) = \Pr[h_{\tilde{a},\tilde{b}}(\boldsymbol{v}) = h_{\tilde{a},\tilde{b}}(\boldsymbol{u})] = \int_0^{\tilde{w}} f_{|\tilde{s}X|}(t)\left(1-\frac{t}{\tilde{w}}\right)\mathrm{d}t \tag{7}$$

接下来将讨论如何得到 $f_{|sX|}(x)$。值得注意的是,因为 $\tilde{s}^2 \geqslant 0$, 随机变量 $\tilde{s}^2$ 并不确切服从正态分布,然而正态分布定义域为 $(-\infty, +\infty)$。保留正态分布主要特征同时避免负值的数学上站得住的方法涉及截断正态分布,其中定义域为区间的一端或两端。特别地,通过界定定义域范围为 $[0, +\infty)$, $\tilde{s}^2$ 可以使用正态分布 $\tilde{s}^2 \sim N(m\mu, m\sigma^2)$ 得到,即单边的截断正态分布,表示为 $\psi(x;\tilde{\mu},\tilde{\sigma}^2,0,+\infty)$。由于 $\tilde{s} \geqslant 0$, 对于任意常数 $t > 0$, 有 $\Pr[\tilde{s} < t] = \Pr[\tilde{s}^2 < t^2]$, 因此 $\tilde{s}$ 的 CDF,表示为 $F_{\tilde{s}}$, 可以计算如下:

$$F_{\tilde{s}} = \Pr[\tilde{s} < t] = \Pr[\tilde{s}^2 < t^2] = \int_0^{t^2} \psi(x;\tilde{\mu},\tilde{\sigma}^2,0,+\infty)\,\mathrm{d}x \tag{8}$$

这里, $\tilde{\mu} = m\mu, \tilde{\sigma}^2 = m\sigma^2$。由于 PDF 是 CDF 的导数,为此 $\tilde{s}$ 的 PDF,表示为 $f_{\tilde{s}}$, 可以由以下公式得到:

$$f_{\tilde{s}}(t) = \frac{\mathrm{d}}{\mathrm{d}t}[F_{\tilde{s}}(t)] = 2t\psi(t^2;\tilde{\mu},\tilde{\sigma}^2,0,+\infty) \tag{9}$$

虽然已知 $\tilde{s}$ 和 X 的分布函数,但是直接计算两者的乘积 $\tilde{s}X$ 并不容易。然而以下引理给出随机变量 $W = XY$ 的特征函数,其中 X 和 Y 是 2 个独立的随机变量,一个服从标准正态分布 $N(0,1)$, 另外一个服从均值为 μ、方差为 σ^2 的分布。

引理 2 2 个独立的随机变量的乘积的特征函数为:

$$\varphi_W(x) = E_Y\{\exp(-\frac{x^2Y^2}{2})\}$$

其中, x 表示标准的正态随机变量, Y 表示均值为 μ、方差为 σ^2 的独立随机变量。

由于 X 服从标准的正态分布,因此可以通过引理 2 得到 $\tilde{s}X$ 的特征函数。

引理 3 $\tilde{s}X$ 的特征函数表示为:

$$\varphi_{\tilde{s}X}(x) = \frac{1}{2(1-\Phi(\frac{-\tilde{\mu}}{\tilde{\sigma}}))}\exp(\frac{1}{8}x^4\tilde{\sigma}^2 - \frac{1}{2}\tilde{\mu}x^2)\operatorname{erfc}(\frac{\frac{1}{2}x^2\tilde{\sigma}^2-\tilde{\mu}}{\sqrt{2}\tilde{\sigma}}) \quad -\infty < x < +\infty$$

其中, $\operatorname{erfc}(t) = \frac{2}{\sqrt{\pi}}\int_t^{+\infty}\exp(-x^2)\,\mathrm{d}x\,(-\infty < t < +\infty)$ 表示补余误差函数。

根据引理 3 得到特征函数,那么 $\tilde{s}X$ 的 PDF,表示为 $f_{\tilde{s}X}(t)$, 可以通过逆傅里叶变换得到,其计算如下所示:

$$f_{\tilde{s}X}(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty}\exp(-\mathrm{i}tx)\varphi_{\tilde{s}X}(x)\,\mathrm{d}x \tag{10}$$

其中,符号 $\mathrm{i} = \sqrt{-1}$ 表示虚数单元。

3.2 FastLSH 的渐近行为

根据式(7)可以看到,任意 2 个向量 $\boldsymbol{v}$ 和 $\boldsymbol{u}$ 在 FastLSH 下发生碰撞的概率 $p(s,\sigma)$ 取决于距离 s 和标准差 σ。从这个角度来看, FastLSH 可以视为 E2LSH 的一般版本,其增加了一个额外的影响因子,即向量对 ($\boldsymbol{v},\boldsymbol{u}$) 每个维度距离平方的变化,使得证明 FastLSH 的性质变得更加困难。尽管如此,接下来给出 FastLSH 渐近等价于 E2LSH。

通过式(2)和式(7),可以看到 E2LSH 和 FastLSH 碰撞概率的表达式及其相似。事实上,如果 $f_{\tilde{s}X}(t)$ 服从正态分布 $N(0,ms^2/n)$, 那么根据以下的事实,通过调节桶宽为 $\tilde{w} = mw/n$, 总能得到 $p_w(s) = p_{\tilde{w}}(s,\sigma)$。

依据事实是:对于 E2LSH, $f_{sX}(t)$ 服从正态分布 $N(0,s^2)$ 且在桶宽 w 下的碰撞概率 $p(s)$ 等于在桶宽 αw 下的碰撞概率 $p(\alpha s)$, 即 $p(s) = p(\alpha s)$, 这里 $\alpha > 0$。直接证明 $f_{\tilde{s}X}(t)$ 与 $N(0,ms^2/n)$ 等价并不是容易,为此间接比较两者之间的特征函数。以下的定理给出 $\tilde{s}X$ 特征函数的渐近行为。

定理 2 当 $m \to +\infty$ 时, $\tilde{s}X$ 的特征函数与 $N(0, ms^2/n)$ 渐近等价,即:

$$\lim_{m\to+\infty}\frac{\varphi_{\tilde{s}X}(x)}{\exp(-\frac{ms^2x^2}{2n})} = 1 \tag{11}$$

其中，$\exp(-ms^2x^2/2n)$ 表示正态分布 $N(0, ms^2/n)$ 的特征函数。

值得注意的是，$\exp(-ms^2x^2/2n)$ 表示正态分布 $N(0,ms^2/n)$ 的特征函数。因此，定理 2 意味着 $f_{\tilde{s}X}(t)$ 渐近等价于 $N(0,ms^2/n)$，这是因为概率分布由其特征函数唯一确定，为此可得以下的推论。

推论 1　$m \to +\infty$，$f_{\tilde{s}X}(t)$ 渐近等价于 $N(0, ms^2/n)$ 的 PDF，即 $f_{\tilde{s}X}(t) \sim f_{sX}(t)$。

通过推论 1 以及前述事实，如果 $\tilde{w} = mw/n$，那么碰撞概率渐近相等，即 $p(s) = p(s,\sigma)$，在该情况下，意味着标准差 σ 并不影响碰撞概率，使得 FastLSH 等价于 E2LSH。

4　实验结果

（1）数据集：本节使用不同类型的 11 个公开的高维真实数据集和 1 个合成的数据[23]进行实验，且每个数据集的查询个数均为 200，见表 1。Sun（http://groups. csail. mit. edu/vision/SUN/）数据集包含大约 8 万个 512 维的 GIST 特征的图像。Cifar（http://www. cs. toronto. edu/ kriz/cifar. html）数据集是从 TinyImage 中提取的 5 万个 512 维 GIST 特征向量的集合。Audio（http://www. cs. princeton. edu/cass/audio. tar. gz）是 192 维的数据集，由来自 DARPA TIMIT 音频速度数据集的约 5 万个音频特征向量组成。Trevi（http://phototour. cs. washington. edu/patches/default. htm）数据集包含大约 10 万张位图图像，且每张图像表示为 4 096 维的特征向量。Notre（http://phototour. cs. washington. edu/datasets/）数据集包含一组 Flickr 图像以及重建的大约 30 万个 128 维的特征。Sift（http://corpus-texmex. irisa. fr）数据集分别包含 1 百万个 128 维的 SIFT 向量。GIST（https://github. com/aaalgo/kgraph）数据集包含 1 百万个 960 维度的 GIST 特征向量。Deep（https://yadi. sk/d/I yaFVqchJmoc）数据集包含 256 维的 1 百万个数据对象，每个对象是由卷积神经网络激活后获得的自然图像的深度神经编码。Ukbench（http://vis. uky. edu/stewe/ukbench/）数据集包含大约 1 百万个 128 维特征的图像。Glove（http://nlp. stanford. edu/projects/glove/）数据集包含大约 1.2 百万个从 Tweets 提取的图像。ImageNet（http://cloudcv. org/objdetect/）数据集包含大约 2.4 百万个数据对象组成的 150 维密集 SIFT 特征。Random 数据集包括 1 百万个从单位超球随机选择 100 维向量。

表 1　12 个真实数据集的统计信息

Table 1　Statistics of 12 real datasets

Datasets	# of Points	# of Queries	Dimension
Sun	79 106	200	512
Cifar	50 000	200	512
Audio	53 387	200	192
Trevi	9 990	200	4 096
Notre	332 668	200	128
Sift	1 000 000	200	128
Gist	1 000 000	200	960
Deep	1 000 000	200	256
Ukbench	1 097 907	200	128
Glove	1 192 514	200	100
ImageNet	2 340 373	200	150
Random	1 000 000	200	100

（2）基准方法：文中将 FastLSH 与 2 种常见的 LSH 算法进行对比实验，分别是 E2LSH[6] 和 ACHash[24]。其中，E2LSH 是一种经典的基础型 LSH 方法，能够在保持近似最近邻检索效果的同时，实现子线性的查询时间。ACHash 则通过引入 Hadamard 变换和稀疏随机投影，有效提升了哈希函数的计算效率。

（3）评估指标：这里使用召回率（Recall）、平均查询时间（Average Query Time）、哈希代价（Hashing Cost）以及索引代价（Indexing Cost）这 4 个常用的标准作为评估指标以评估 E2LSH、FastLSH 和 ACHash 的性能，其中哈希代价表示评估一个哈希函数所需的时间消耗，而索引代价表示算法除检索外的端到端时间开销。

（4）参数设置：在相同数据集和目标召回率的设定下，为保证实验的公平性，对 3 种算法采用相同的参数配置：k 表示每个哈希表中的哈希函数数量，L 表示哈希表的数量。由此，3 种算法在空间占用方面保持一致。在所有实验中，FastLSH 的参数 m 固定为 30。为获得最佳性能，ACHash 的采样比例采用默认值 0.25。研究在所有数据集上报告 3 种算法的 R10@10 指标结果。

（5）实验结果与讨论：在 E2LSH 算法框架内，本节旨在证明 FastLSH 可以显著减少索引构建时间，同时实现与其他基于 LSH 的算法几乎相同的查询精度和查询时间。值得注意的是，FastLSH 的目的并非降低查询时间，因为查询处理中的时间开销主要由查询及其候选对象之间距离的精确评估所决定。在查询阶段，FastLSH、ACHash 与 E2LSH 同样需要计算查询

与其候选对象的欧式距离以确定 ANN 检索结果,如果检测的候选对象越多,所需的时间开销则越大。下面给出 3 个算法的性能对比结果。

这里比较 E2LSH、ACHash 和 FastLSH 在 0.9 目标精度时的性能。图 1 给出了精度、平均查询时间、LSH 计算时间和索引构建时间的结果。从图 1 中(a)、(b)、(e)和(f)中可以看到,FastLSH 和 E2LSH 实现基本相当的查询性能和查询响应时间。另外对于不同的数据集,精度和查询响应时间均做多次取平均并绘画误差棒,在对应的柱状图顶部展示。由于所获得的精度相差较小,误差棒需要放大才能显示清晰,说明 3 种算法均得到稳定的查询性能;然而对于 ACHash,查询响应时间的误差棒波动较大,意味着其查询效率稳定性较差。主要原因在于 ACHash 缺乏理论上的保证,在大多数情况下,其查询效率略低于 FastLSH 和 E2LSH。

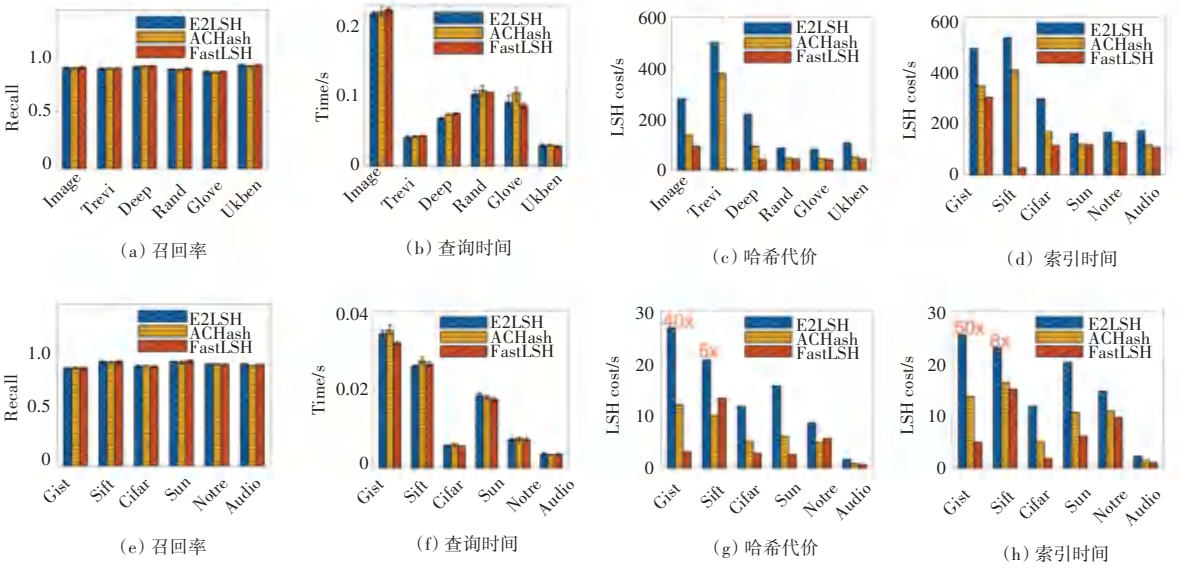

图 1 召回率、平均查询时间、LSH 计算时间和索引构建时间的比较

Fig. 1 Comparison of recall, average query time, LSH calculation time and index cost

当涉及到 LSH 计算和索引开销时,E2LSH、ACHash 和 FastLSH 的性能相差很大,如图 1 中(c)、(d)、(g)和(h)所示。从图 1(c)和图 1(g)中发现 FastLSH 的 LSH 计算时间明显优于 E2LSH 和 ACHash,例如针对高维数据集 Trevi、相比于 E2LSH,FastLSH 获得大约 80 倍的加速,然而对于 ACHash 则有大约 60 倍的加速。针对 Gist,FastLSH 相比 E2LSH 和 ACHash 分别获得 24 倍和 11 倍的加速。然而对于低维度的数据集、例如 Random 和 Glove,尽管此时 ACHash 的采样维度为 25,低于 FastLSH 使用固定的采样维度 $m = 30$,但是 FastLSH 仍然获得较快的 LSH 计算加速,并比 E2LSH 速度提高至少 2 倍以上。主要原因在于 FastLSH 的 LSH 计算的时间复杂度为 $O(m)$。并非 E2LSH 的 $O(n)$,然而对于 ACHash,由于需要进行 Hadamard 变换以及使用固定的采样比例,使得 LSH 计算开销多于 FastLSH。

此外,索引构建时间的端到端加速如图 1 中(d)和(h)所示。由于 FastLSH 哈希代价的显著下降,其构建索引所花费的时间开销比 E2LSH 与 ACHash 分别最多减少了大约 20 倍与 15 倍。除了哈希之外,索引构建过程还包括哈希表初始化和链表维护等其他操作,这些操作不能加速。因此,索引构建中的端到端时间开销减少的程度不如哈希代价减少的程度。尽管如此,对于低维的数据集,FastLSH 的端到端所需的时间均比 E2LSH 与 ACHash 少,说明 LSH 算法中哈希代价在端到端时间开销中占主要消耗,特别是数据维度非常高时。

5 结束语

本文基于随机采样提出一种高效的 LSH 函数 FastLSH,用于提高哈希函数的计算效率。FastLSH 将哈希函数的计算复杂度从 $O(n)$ 降为 $O(m)$,这里 $m < n$。通过中心极限定理,FastLSH 给出针对检索结果具有可证明的概率保证,并严格证明 FastLSH 的碰撞概率渐近等价于经典的 E2LSH。在 ANN 检索任务上进行了大量的实验,结果表明通过调节合适的 m, FastLSH 实现与 E2LSH 基本相等的

查询性能并有效提高 LSH 函数的计算效率。

参考文献

[1] BENTLEY J L. Multidimensional binary search trees used for associative searching [J]. Communications of the ACM, 1975, 18(9): 509-517.

[2] KATAYAMA N, SATOH S I. The SR-tree: An index structure for high-dimensional nearest neighbor queries [J]. ACM Sigmod Record, 1997, 26(2): 369-380.

[3] SAMET H. Foundations of multidimensional and metric data structures [M]. San Francisco:Morgan Kaufmann, 2006.

[4] KUSHILEVITZ E, OSTROVSKY R, RABANI Y. Efficient search for approximate nearest neighbor in high dimensional spaces [C]//Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing. New York:ACM, 1998: 614-623.

[5] ANDONI A, INDYK P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions [J]. Communications of the ACM, 2008, 51(1): 117-122.

[6] DATAR M, IMMORLICA N, INDYK P, et al. Locality-sensitive hashing scheme based on p-stable distributions[C]// Proceedings of the Twentieth Annual Symposium on Computational Geometry. New York:ACM, 2004: 253-262.

[7] INDYK P, MOTWANI R. Approximate nearest neighbors: Towards removing the curse of dimensionality[C]//Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing. Piscataway, NJ:IEEE, 1998: 604-613.

[8] GAN Junhao, FENG Jianlin, FANG Qiong, et al. Locality-sensitive hashing scheme based on dynamic collision counting [C]//Proceedings of 2012 ACM SIGMOD International Conference on Management of Data. New York: ACM, 2012: 541-552.

[9] HUANG Qiang, FENG Jianlin, ZHANG Yikai, et al. Query-aware locality-sensitive hashing for approximate nearest neighbor search [J]. Proceedings of the VLDB Endowment, 2015, 9(1): 1-12.

[10] LIU Wanqi, WANG Hanchen, ZHANG Ying, et al. I-LSH: I/O efficient c-approximate nearest neighbor search in high-dimensional space [C]//Proceedings of 2019 IEEE 35th International Conference on Data Engineering (ICDE). Piscataway,NJ:IEEE, 2019: 1670-1673.

[11] LU Kejing, KUDO M. R2LSH: A nearest neighbor search scheme based on two-dimensional projected spaces [C]// Proceedings of 2020 IEEE 36th International Conference on Data Engineering (ICDE). Piscataway, NJ: IEEE, 2020: 104510-104556.

[12] LU Kejing, WANG Hongya, WANG Wei, et al. VHP: Approximate nearest neighbor search via virtual hypersphere partitioning [J]. Proceedings of the VLDB Endowment, 2020, 13(9): 1443-1455.

[13] LV Qin, JOSEPHSON W, WANG Zhe, et al. Multi-probe LSH: Efficient indexing for high-dimensional similarity search [C]//Proceedings of the 33rd International Conference on Very Large Data Bases. Vienna, Austria: Yahoo Research, 2007: 950-961.

[14] SUN Yifang, WANG Wei, QIN Jianbin, et al. SRS: Solving c-approximate nearest neighbor queries in high dimensional euclidean space with a tiny index [J]. Proceedings of the VLDB Endowment, 2014,8(1):1-12.

[15] TIAN Yao, ZHAO Xi, ZHOU Xiaofang. DB-LSH 2.0: Locality-sensitive hashing with query-based dynamic bucketing [J]. IEEE Transactions on Knowledge and Data Engineering, 2023, 36(3): 1000-1015.

[16] YANG Chengcheng, DENG Dong, SHANG Shuo, et al. Efficient locality-sensitive hashing over high-dimensional data streams [C]//Proceedings of 2020 IEEE 36th International Conference on Data Engineering (ICDE). Piscataway,NJ:IEEE, 2020: 1986-1989.

[17] ZHENG Bolong, ZHAO Xi, WENG Lianggui, et al. PM-LSH: A fast and accurate LSH framework for high-dimensional approximate NN search [J]. Proceedings of the VLDB endowment, 2020, 13(5): 643-55.

[18] KOGA H, ISHIBASHI T, WATANABE T. Fast agglomerative hierarchical clustering algorithm using Locality-Sensitive Hashing [J]. Knowledge and Information Systems, 2007, 12: 25-53.

[19] CHEN C, HORNG S J, HUANG C P. Locality sensitive hashing for sampling-based algorithms in association rule mining [J]. Expert Systems with Applications, 2011, 38(10): 12388-12397.

[20] LUO Chen, SHRIVASTAVA A. Arrays of (locality-sensitive) count estimators (ace) anomaly detection on the edge [C]// Proceedings of 2018 World Wide Web Conference. New York: ACM, 2018: 1439-1448.

[21] CHEN B, MEDINI T, FARWELL J, et al. Slide: In defense of smart algorithms over hardware acceleration for large-scale deep learning systems [J]. Proceedings of Machine Learning and Systems, 2020, 2: 291-306.

[22] SUNDARAM N, TURMUKHAMETOVA A, SATISH N, et al. Streaming similarity search over one billion tweets using parallel locality-sensitive hashing [J]. Proceedings of the VLDB Endowment, 2013, 6(14): 1930-1941.

[23] LI Wen, ZHANG Ying, SUN Yifang, et al. Approximate nearest neighbor search on high dimensional data: experiments, analyses, and improvement [J]. IEEE Transactions on Knowledge and Data Engineering, 2019, 32(8): 1475-1488.

[24] DASGUPTA A, KUMAR R, SARLÓS T. Fast locality-sensitive hashing[C]//Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York:ACM, 2011: 1073-1081.