通过迭代最近邻拓展进行快速而精确的哈希

金仲明, 张德兵, 胡尧, Shiding Lin, 蔡登, Member, IEEE, and 何晓飞, Senior Member, IEEE

June 1, 2016

1 摘要

哈希技术广泛使用在近邻检索问题中。这些方法的基本思想是生成二进制编码来表示原数据，并且保留原数据中的相似度关系。给定询问，哈希方法会线性扫描所有与询问点之间汉明距离不超过r_h的点。然而，在寻找精确最近邻时，定位时间与线性扫描时间都与$O\sum_{i=0}^{r_h}\binom{c}{i}$相关，其中c为编码长度。这个复杂度随$r_h$增长而指数级上升。IEH算法通过离线NN图来建立辅助索引，以此来避免r_h过大。这个辅助索引可以很方便地与所有传统哈希方法相结合。

2 引言

当数据维度增大时，传统近邻检索算法的表现甚至比线性扫描更差，因此人们转而选择近似近邻检索。这些算法的关键思想就是生成二进制编码来表示原数据，并且保留原数据中的相似度关系。给定n个d维向量构成的数据集，哈希算法采用c个哈希函数生成长度为c的汉明编码。常用的检索方法是依次计算询问点与所有点直接的汉明距离并将他们进行排名。我们可以利用xor操作来计算汉明距离，不过找到最近邻的时间耗费仍然需要$O(n)$。一种常见的加速方法为使用哈希索引。我们通过把c位的哈希编码放入与其相同的c位哈希桶中，来构建哈希索引。给定一个询问，检索可以按照以下三个阶段进行：1）编码阶段：询问点经过c个哈希函数生成c位编码。2）定位阶段：返回询问点的汉明半径不超过r_h的所有哈希桶中的点3）线性扫描阶段：从返回的点中算出答案。显然，上述操作的总时间复杂度为$O(dc+\sum_{i=0}^{r_h}\binom{c}{i}+\frac{n}{2^c}\sum_{i=0}^{r_h}\binom{c}{i})$。然而，为了找到准确的近邻，现有的哈希算法需要使用很大的r_h值，这会导致时间复杂度指数级上升。

以图1 为例，一个哈希算法产生了l_1, l_2, l_3三个超平面，将二维空间分割为7部分。同一部分中的点拥有相同的哈希码。图中绿色的星形为询问点，红色圆圈为近邻，剩余点为黑色正方形。使用哈希索引时，令汉明半径为2就能达到100%召回率。需要被定位的桶数量为$\sum_{i=0}^{2}\binom{3}{i}=7$，被利用到的点为16个（包括10个错误点）。利用辅助索引我们可以避免使用大的半径。参考图1(b),我们选取与询问点哈希编码相同的3个点a_1, a_2, a_3作为辅助点，通过拓展他们的近邻，我们仍然可以达到100%召回率。在这个方法中，被定位的桶只有一个，且被利用的点只有7个（包含1个错误点）。根据上述分析，与原方法对比，我们的方法可以在保持同样召回率的前提下，使用更少的定位时间和线性扫描时间。

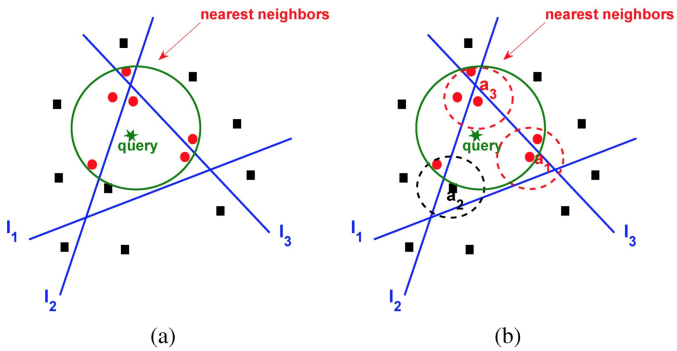

Figure 1: 动机说明

基于这个想法，我们提出了IEH算法，该算法可以在不使用大半径值的情况下实现高召回率。特别地，IEH算法在离线阶段构建了一个K近邻表格，存储了每个数据点的k个近邻。在线阶段中，IEH利用K近邻表格拓展点，这些点来自汉明半径非常小的哈希桶中。通过执行迭代过程，IEH的表现能够获得更好地提升。理论分析与实验结果表明，IEH算法可以极大地改进传统哈希方法的表现。

3 相关工作

3.1 基于哈希的近似近邻检索方法

一般哈希方法可以总结如下。给定n个数据点$X = [x_1, ..., x_n] \in \mathbb{R}^{d*n}$，通过$c$个哈希函数生成$c$位哈希编码：

$$H(x) = [h_1(x), h_2(x), ...h_c(x)]$$

其中$h_l(x) \in 0, 1$是第l个哈希函数。对基于线性投影的哈希，我们有

$$h_l(x) = sgn(F(w_l^T x + t_l))$$

其中w_l是投影向量，t_l是截距。不同哈希函数中各参数不同。

一种广泛使用的哈希算法为局部敏感哈希(LSH),它基于随机投影，即随机生成w_l。在LSH中，F是单位函数，$t_l = 0$。因此我们有：

$$h_l(x) = \begin{cases} 1, \ if \ w_l^T x \geq 0 \\ 0, otherwise \end{cases}$$

其中向量w_l是由“与输入x同维度”的零均值多变量高斯函数$\mathcal{N}(0, I)$生成。基于LSH的大部分拓展算法都尝试降低LSH的空间需求，但这会导致处理询问的时间增加。所有这些算法都是基于随机投影方法，并且不考虑数据分布情况。

与上面这些方法不同，一些基于学习的哈希算法则使用了数据分布。一些算法利用了数据类同矩阵中的谱特性，来生成二进制编码。利用谱特性的方法通常很耗费时间，为了避免高计算花销，Weiss等人提出了谱哈希(SH)方法，该方法中进行了一个强假设：所有数据均匀分布。这个假设引出了一个简单的1-D Laplacians 特征根函数解。然而，该方法忽视了原数据中的几何结构，因此可能导致局部最优的表现。锚图哈希(AGH)算法则克服了这个缺陷。它从数据中生成锚点，并且通过他们表达相似性。在这个方法下，数据的谱分析能够有显著的表现。也有一些方法致力于哈希函数学习中的标记信息杠杆，这引向了监督学习和半监督学习。

3.2 基于树的近似近邻检索方法

KD树方法是近似近邻检索问题中一个最为常用且有效的树方法。传统的KD树算法在低维度上有很好表现，但在高维度上表现迅速退化。Arya等人改造了原始KD树来进行近似匹配，其中使用了优先队列来加速树中的检索。在基于树的算法中，还有随机化KD树、R树等算法。

3.3 基于K近邻图的近似近邻检索方法

正如第一章中所说，IEH算法需要在离线阶段构建一个K近邻表格。这个表格与K近邻图有紧密的联系。一个K近邻图是一个有向图，包含了一个点集X和一个有向边集合E。当x_j是x_i的K近邻之一时，x_i被连接到x_j。IEH算法使用哈希方法来获得初始点，这些点比随机点更加接近询问点。

4 迭代拓展哈希

4.1 标记与符号

令$X = [x_1, ..., x_n]$是大小为N的数据集，$X \in \mathbb{R}^{d*n}$。数据点$x \in \mathbb{R}^d$的K近邻使用$\mathcal{N}_k(x)$。给定询问$q \in \mathbb{R}^d$，我们希望在X中找到它的t近邻$\mathcal{N}_t(q)$。我们使用欧拉距离$\rho(.,.)$来衡量点x与y的距离：$\rho(x, y) = \|x - y\|_2$。

4.2 算法

我们按以下几点说明IEH算法。在离线阶段，IEH需要如下操作：

1. 使用一个现有的哈希方法生成c个哈希函数
2. 使用这些哈希函数生成哈希编码
3. 构建用于在线拓展的最近邻表

在在线阶段，IEH有3个阶段：

1. 编码阶段：生成查询点的c位二进制码
2. 定位阶段：一个非常小的汉明半径中的哈希桶中的候选点会被返回。
3. 拓展阶段：迭代地选择p个最近候选点，利用k近邻表拓展它们的k近邻（迭代次数为s)。

在算法1中我们总结了IEH的在线步骤。

Algorithm 1 迭代拓展哈希（IEH）

Require: p：最近邻数量，用于每次迭代中进行拓展；k：要拓展近邻的数目；s：迭代次数；id：临时索引；M：临时集合；
Ensure: 查询点的最近邻集合

初始化：$id = -1$，$M = \emptyset$
编码阶段：生成查询点的c位二进制码
查询阶段：以一个很小的汉明半径得到一些候选点，放入M中
拓展阶段：
for all iter ¡ s **do**
  在M中找到查询点的最近p个候选点
  for all i ¡ p **do**
    for all j ¡ k **do**
      id = 第i个最近邻候选点的第j个邻居
      if $id \notin M$ **then**
        $M = M \cup \{id\}$
      end if
    end for
  end for
end for
从M中选t个最近邻作为结果

上面的方法与原始的哈希技术不同的地方在于，我们使用拓展阶段替换了一个大的汉明半径。拓展阶段主要取决于最近的候选点和K近邻表。K近邻表在离线阶段被构建，它使用$\mathcal{N}_k(x)$保存数据集X中所有x。因为候选点与询问点很近，所以我们的拓展操作可以获得其他的临近候选点，从而避免一个大的汉明半径。我们也设计了一个迭代步骤，它在每次迭代中拓展前p个最近候选集，来实现询问导向的拓展。

4.3 K近邻表的动态拓展

K近邻表可以被动态地更新。如图2 (a)所示，我们希望在当前的k近邻表中插入新的点x_m。为了动态更新k近邻表，点对间的距离以及它们的近邻需要被记录。例如，点x_1有近邻$x_5, x_7, ...$, x_1与x_7间的距离可以被存储为$\rho(x_1, x_7)(= \rho(x_7, x_1))$。如图2 (b)所示，在当前k近邻表中插入$x_m$的步骤如下所示：

1. 计算x_m与当前k近邻表中所有点的距离。
2. 根据距离值调整表。例如$\rho(x_3, x_9) < \rho(x_3, x_m) < \rho(x_3, x_2)$,$x_m$可以被考虑为$x_3$的近邻，它位于$x_9$和$x_2$之间。因为我们只维护$x_3$的k近邻表，所以$x_7$会从中被删除。在实现中，我们使用链表来构建k近邻图，使它的调整操作能够很快进行。
3. 根据距离值将x_m和它的近邻插入表中。例如，假设有$\rho(x_m, x_4) < \rho(x_m, x_1) < ... < \rho(x_m, x_7) < \rho(x_m, x_20)$，则$x_m$的k近邻为$x_4, x_1, ..., x_7, x_{20}$。

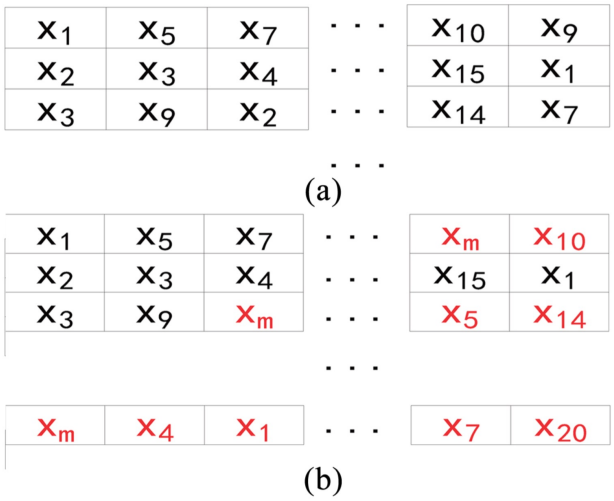

Figure 2: 构建KNN表的离线过程大致说明

4.4 在线阶段的实现

在实现在线阶段时，我们使用了2个关键的数据结构：优先队列和位图。首先，为了在每次迭代中快速地选择前p个最近邻，我们使用优先队列来存储集合$\mathcal{M}$。第二，因为拓展阶段中存在重复的候选点，我们使用位图（位图中每个点对应一个点编号）来避免重复统计。

4.5 理论分析

这一章中，我们通过理论分析来说明基于LSH的IEH算法的k近邻拓展可以保证准确的t近邻有很高的概率被返回，当参数k足够大时。

下面的知名引理说明了LSH算法很好地保留了点对间的相似性。

定理 4.1 (Jonson Lindenstrauss 理论) 令$U \in \mathbb{R}^{d*c}$是一个随机矩阵，其中每个$U_{i,j}$在高斯分布$\mathcal{N}(x \mid 0,1)$上独立同分布。令P_u为投影操作，将$\mathbb{R}^d$上的向量投影进U的列向量涵盖的字空间$\mathbb{R}^c$上。对任意两个$\mathbb{R}^d$上的点x, q，给定$\varepsilon \in (0,1)$，我们有$(1-\varepsilon)\|x-q\|_2^2 \leq \|P_u(x-q)\|_2^2 \leq (1+\varepsilon)\|x-q\|_2^2$。

假设X均匀分布。因此，如图3所示，$\mathcal{N}_t(q)$在超平面S_{r1}的半径r_1内随机分布。在定位阶段后，我们得到p个候选点$X' \in \mathbb{R}^{d*p}, X' \subset X$。基于引理1，候选点与询问q之间的期望距离$r_2 = 2^{-\frac{1}{d}}(1+\varepsilon)r_1$。对于候选点$x_i \in X', \mathcal{N}_k(x_i)$在超平面$S_R^{(i)}$上半径不超过$R$的区域内随机分布。下面的定理说明了在半径$R$的条件下，拓展阶段中被包含的点有很高的概率与$\mathcal{N}_t(q)$相符合。

定理 4.2 (Jonson Lindenstrauss 理论) 如果有$R \geq \sqrt{r_1^2 + r_2^2} = \sqrt{4^{-\frac{1}{d}}(1+\varepsilon)^2 + 1} * r1$，那么$S_{r1}$至少有$1-(\frac{1}{2})^p$的概率，能够被 $S_R^{(i)}(i=1,2,..p)$的联合给覆盖。

证明 在图3 中，我们可以清楚地看出，如果$R \geq \sqrt{r_1^2 + r_2^2}$，每个候选点中央的超平面可以覆盖至少一半的超平面$S_{r1}$。在一般的高维数据上，我们可以轻松地检定这个特性仍然保持。基于候选点的随机性，对于给定候选点$x_i, i = 1,2,...,p$，每个超平面S_{r1}上的点被超平面$S_R^{(i)}$覆盖的概率至少为$\frac{1}{2}$。因为有p个独立的超平面，所以S_{r1}至少有$1-(\frac{1}{2})^p$的概率能够被 $S_R^{(i)}(i=1,2,..p)$的联合给覆盖。

注意到R的下边界由ε决定。在IEH算法中，每次迭代中拓展前p个近邻与询问导向的迭代步骤都能降低这个下边界。在拓展阶段中，参数k的值越大，半径R的值越大。因此，使用一个大的k，IEH算法拥有一个非常高的概率找到询问q的精确t近邻。 □

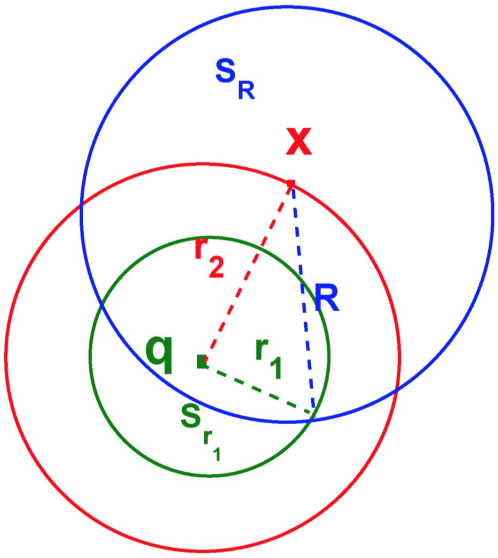

Figure 3: 引理4.1 的说明

4.6 复杂度分析

IEH算法中包含下列三个额外开销。

1. 在离线阶段，IEH需要$O(n^2(d+log_2 n))$的时间来构建k近邻表。在实验中，我们展示了大规模数据上的实际构建时间。
2. 在线阶段，k近邻表需要的额外空间复杂度为$O(kn)$。因为实际使用中，k的取值较小，所以这部分额外花销较低。
3. 在线阶段，算法1说明了IEH仅仅带来了一点额外的时间开销$O(dpks+\sum_{i=1}^{pks} log_2 i)$，用于拓展阶段中的$q$。在这里，$p,k,s$的值都非常小。进一步地，由于存在重复的候选点，所以实际的拓展时间非常少。

5 实验结果

本章节中，我们用高维数据ANN检索问题评估IEH算法。实验中用到的三种基于真实数据的大型数据库分别如下：

1. CIFAR 10：包含60000张图片，每张图片用一个3071维的向量表示。该数据库是公开的，文章10,25,42的作者都使用过。
2. GIST-1M：包含一百万GIST描述子，每个描述子用一个384维的向量表示。该数据库是公开的，文章25,35,以及方法citeSSH都使用过。
3. SIFT-1M：包含一百万SIFT描述子，每个描述子用一个128维的向量表示。该数据库是公开的，文章18,41,42的作者都使用过。

为了减少定位过程中获得候选的数量，针对CIFAR10, GIST-1M, 和SIFT1M，我们分别学出24位、32位、和32位的汉明距离。我们随机抽取1000个数据点作为query，剩下的用作gallery数据库。如果返回点在query点最近k个邻居之内，这个点就被认为是正确的。为了测试不同方面的效果，我们取k=1和k=50两个值。我们的IEH方法有三个参数，我们分别把他们设为p=10，k=50，s=3。参数选择将在5.5章节介绍。所有的代码都是基于c++实现的，并在一台Intel Xeon(R) E7450 CPU（24核），包含256GB内存的电脑上测试的。在离线过程中，我们用了多线程的技术加速建立KNN土的过程。相反的，为了公平起见，在线上测试中，我们没有用多线程技术。

正如我们在前面章节中讨论的那样，IEH带来了一些额外代价。表5.4展示了额外的离线时间代价和线上存储代价。我们可以清楚地看出，额外的代价在大型数据库中是容易处理的。除此之外，我们实验中只用了单机来跑离线实验的预处理过程。对于大型数据库，这个预处理过程可以通过多机并行方式解决。

5.1 与原始哈希方法的比较

既然我们的IEH算法不依赖任何特定的哈希方法，不同的哈希方法将会带来不同的实验结果。我们用了5个最先进的哈希算法。

1. LSH（Locality sensitive hashing），这个方法基本上是基于随机映射的。
2. KLSH（Kernelized locality sensitive hashing），这个方法将LSH方法泛化到内核空间.
3. SH（Spectral hashing），这个方法是基于量化分析特征函数的值，特征函数沿着数据PCA方向计算。
4. AGH（Anchor graph hashing），这个方法建立anchor图以加速波谱分析。
5. SPH（SPherical hashing），这个方法用一个基于哈希函数的超平面，将数据点映射到二进制码。

需要注意的是LSH是一个线性方法，剩下的四种是非线性方法。对于KLSH和AGH，我们用了高斯核，宽度参数θ 是通过随机选择3000个样本进行估计的，θ 是两两距离的平均值。KLSH和AGH都需要选择m个支持样本，我们就用了k-mean产生的这m个样本。整个实验中，参数m固定为100。与文章12,25,26,35,41,和42相似，我们用召回率曲线评估IEH方法的效果，并与原始的哈希方法做对比。为了考虑线上过程的所有运行时间，我们采用展示检索时间作为x轴，而不是抽取样本的数量。召回率曲线中的每个点都对应着一个给定的汉明半径（从0到5）。

图4 展示了1NN和50NN下的有LEH扩展的LSH召回率曲线，与原始LSH方法做对比。LSH$_{IEH}$-1NN表示有LEH扩展的LSH检索ANN。图5, 6, 7, 8 展示了有LEH扩展的其他四种哈希方法的性能。

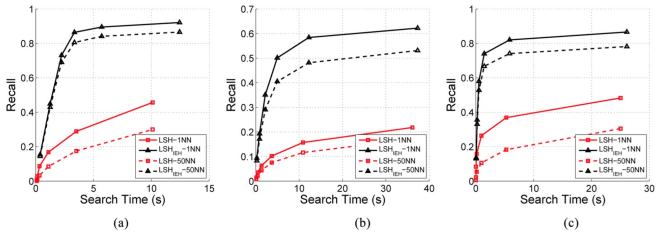

Figure 4: LSH和LSH$_{IEH}$在CIFAR10(a), GIST-1M(b), SIFT-1M(c)上的召回曲线

正如我们在前面章节所说，给定汉明半径，IEH扩展比原始哈希方法消耗了更多时间，只是因为IEH需要迭代KNN过程。然而，消耗同样的时间，IEH扩展获得了明显更高的召回率。更重要的是，在多数情况下IEH能够同时做到更高的召回率和更少的检索时间。例如，在CIFAR10,LSH-1NN用了10.11s获得45.7%的召回率，二LSH$_{LEH}$-1NN只花费2,15s旧大道了73.1%的召回率。这些图片清楚地展示了IEH相对于原始哈希方法的优势。表5.4展示了50NN下原始哈希方法和我们提出IEH方法在三个数据库上的平均正确率均值（mAP）。我们能够清晰地看出IEH扩展同样获得了比原始哈希方法更高的mAP。

5.2 与多表LSH的比较

这部分我们将LSH方法与基于多表LSH哈希方法进行比较。图9和表5.4展示了三个数据库下，50NN的召回率－抽取的样本数量的曲线，多表LSH（LSH$_{MT}$）以及我们提出的IEH算法的mAP。在这些实验中，我们建立一个3张表构成的LSH。我们能够清楚地看到，LEH扩展相比基于多表LSH的方法，同样地取得了更高的召回率和mAP。

5.3 与KD树的比较

在这部分实验中，我们将在数据库上比较KLSH$_{LEH}$（基于KLSH的LEH算法）与KD-Tree of Fast Library for ANN(FLANN)的召回率曲线。FLANN是ANN检索问题方面最受欢迎的库。FLANN的KD树的实现见38,39文。KDT4展示了基于4棵树的KD树方法，KDT8和KDT16也是一样。KD树召回率曲线上的每个点对应一个给定的check number：32,64,128,256,512,1024,2048。check number是KD树的一个参数，对于KLSH$_{IEH}$和KLSH，每个点都对应一个给定的汉明半径（从0到4）。

图10 和11 展示了1NN和50NN下KLSH，KLSH$_{IEH}$和KD树的召回率曲线。我们可以清楚地看出，随着数据维数的增加，IEH比KD树有这更多优势。并且，IEH在检索50NN的时候优势更明显。两个图都能说明，在三个数据库中，IEH都比KD树性能好。表5.4展示了KLSH，KLSH$_{IEH}$和KD树的索引占用内存大小。说明与KD树相比，IEH占用更小空间。考虑到内存使用和速度，KLSH$_{IEH}$在大型ANN检索问题中，比KD树的实现库（FLANN）表现更佳。

5.4 时间复杂度

正如我们在章节3.2中提到的，我们提出的IEH方法与原始哈希方法的主要不同在于展开阶段。表5.4展示了IEH在GIST-1M数据库上的所有运行时间（1000个query）。显然，展开阶段并不好是，尤其是对于较大的汉明半径。

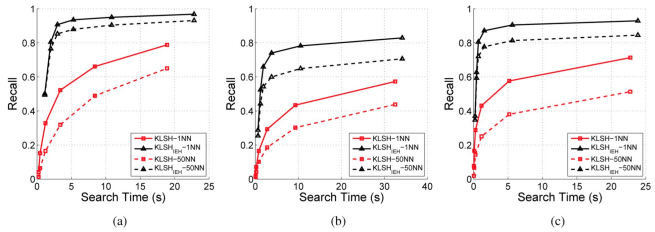

Figure 5: KLSH和KLSH$_{IEH}$在CIFAR10(a), GIST-1M(b), SIFT-1M(c)上的召回曲线

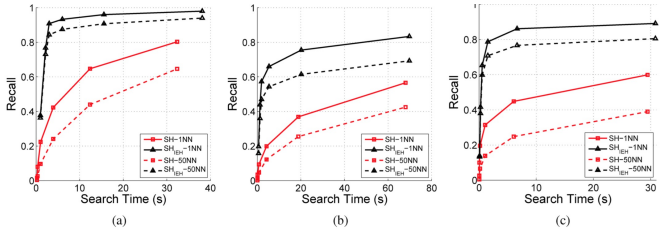

Figure 6: SH和SH$_{IEH}$在CIFAR10(a), GIST-1M(b), SIFT-1M(c)上的召回曲线

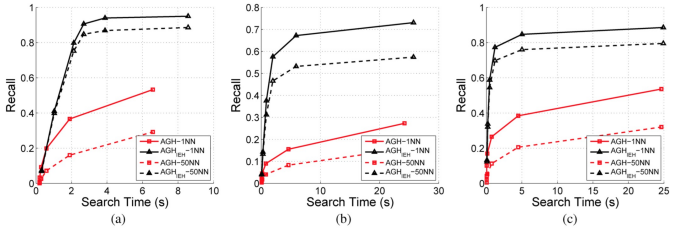

Figure 7: AGH和AGH$_{IEH}$在CIFAR10(a), GIST-1M(b), SIFT-1M(c)上的召回曲线

5.5 参数选择

正如之前讨论的，IEH有三个至关重要的参数：p，k和s。在之前的实验中，我们统一设置p=10,k=50,s=3。在这部分实验中，我们试着检查IEH的性能如何受参数影响。图12, 13, **??** 展示了50NN、汉明半径为2的时候，KLSH$_{IEH}$的召回率随p、k、s变化的曲线。这些图中同时展示了50NN、汉明半径为4的时候的召回率。他们与KLSH的检索时间相同。表5.4, 5.4, 5.4 展示了KLSH$_{IEH}$在50NN、汉明半径为2的时候，检索时间随着p、k、s变化的曲线。可以预料，随着p、k、s的增加，召回率有所上升。当p、k、s较大时，KLSH$_{IEH}$小行更多的时间检索NN。这个特点给用户根据不同应用环境，平衡速度和准确率的方法。我们可以清楚地看出p=10,k=50,s=3是比较合理的参数，同时考虑到高的召回率以及低的检索时间。

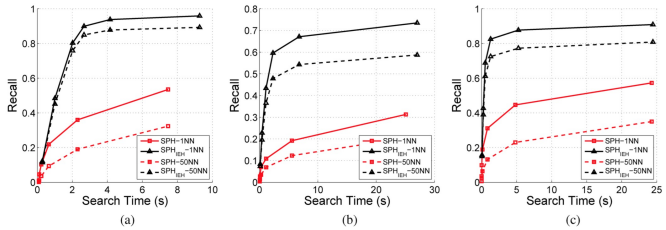

Figure 8: SPH和SPH$_{IEH}$在CIFAR10(a), GIST-1M(b), SIFT-1M(c)上的召回曲线

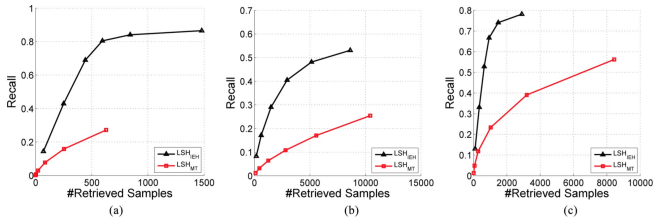

Figure 9: 50-NN LSH$_{MT}$和LSH$_{IEH}$在CIFAR10(a), GIST-1M(b), SIFT-1M(c)上的召回曲线

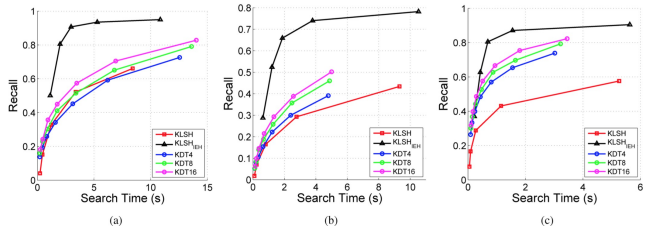

Figure 10: 1NN KLSH, KLSH$_{IEH}$, KD-TREE, 在CIFAR10(a), GIST-1M(b), SIFT-1M(c)上的召回曲线

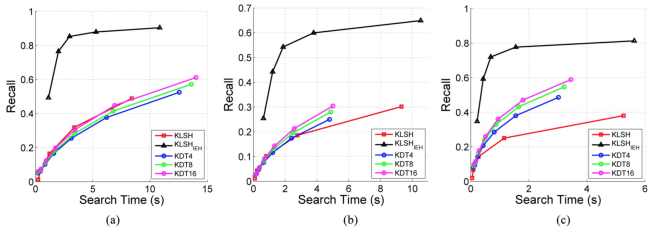

Figure 11: 50-NN KLSH, KLSH$_{IEH}$, KD-TREE在CIFAR10(a), GIST-1M(b), SIFT-1M(c)上的召回曲线

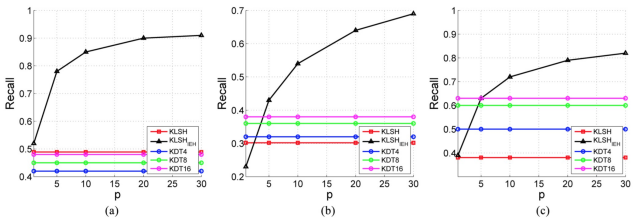

Figure 12: 50-NN KLSH$_{IEH}$相对于汉明半径2，相应参数p的取值在CIFAR10(a), GIST-1M(b), SIFT-1M(c)上的召回曲线

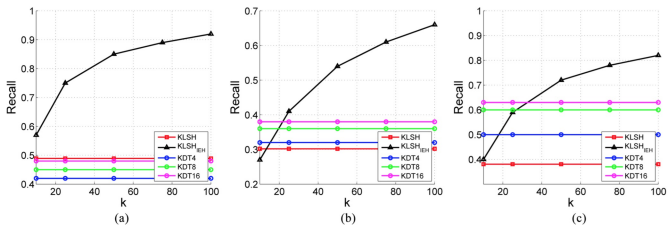

Figure 13: 50-NN KLSH$_{IEH}$相对于汉明半径2，相应参数k的取值在CIFAR10(a), GIST-1M(b), SIFT-1M(c)上的召回曲线

TABLE II

ADDITIONAL COSTS OF IEH

Data Set	Additional Offline Time	Additional Online Storage
CIFAR10	156 seconds	12M
GIST-1M	5.56 hours	200M
SIFT-1M	2.5 hours	200M

TABLE III

50 NN MEAN AVERAGE PRECISION OF ORIGINAL HASHING METHOD AND IEH

DataSet	Hashing Method	Original	IEH
CIFAR10	LSH	0.014	0.079
	KLSH	0.071	0.073
	SH	0.039	0.108
	AGH	0.037	0.076
	SPH	0.031	0.103
GIST-1M	LSH	0.004	0.013
	KLSH	0.014	0.021
	SH	0.010	0.024
	AGH	0.006	0.016
	SPH	0.009	0.022
SIFT-1M	LSH	0.024	0.054
	KLSH	0.042	0.050
	SH	0.051	0.069
	AGH	0.039	0.060
	SPH	0.036	0.063

TABLE IV

50 NN MEAN AVERAGE PRECISION OF LSH$_{MT}$ AND LSH$_{IEH}$

DataSet	LSH$_{MT}$	LSH$_{IEH}$
CIFAR10	0.023	0.079
GIST-1M	0.005	0.013
SIFT-1M	0.038	0.054

TABLE V

STORAGE OF ALL METHODS

Data Set	KLSH	KLSH$_{IEH}$	KDT4	KDT8	KDT16
CIFAR10	10M	22M	14M	28M	57M
GIST-1M	10M	210M	310M	510M	1.01G
SIFT-1M	10M	210M	250M	496M	909M

TABLE VI
PROCESSING TIME (1000 QUERIES) OF IEH ON GIST-1M DATASET

Time	Hashing Method	$r_h = 0$	$r_h = 1$	$r_h = 2$	$r_h = 3$	$r_h = 4$	$r_h = 5$
Coding Time (s)	LSH	0.03	0.03	0.03	0.03	0.03	0.03
	KLSH	0.06	0.06	0.06	0.06	0.06	0.06
	SH	0.08	0.08	0.08	0.08	0.08	0.08
	AGH	0.08	0.08	0.08	0.08	0.08	0.08
	SPH	0.03	0.03	0.03	0.03	0.03	0.03
Locating Time (s)	LSH	0.001	0.001	0.08	0.83	4.97	25.43
	KLSH	0.001	0.01	0.12	0.83	4.78	22.14
	SH	0.001	0.001	0.12	1.09	7.24	31.86
	AGH	0.001	0.001	0.05	0.65	4.31	23.58
	SPH	0.001	0.001	0.12	0.7	4.52	23.04
Linear Scan Time (s)	LSH	0.07	0.38	1.21	2.82	5.88	10.63
	KLSH	0.03	0.17	0.63	1.89	4.62	10.31
	SH	0.001	0.09	0.59	3.05	11.55	36.12
	AGH	0.001	0.001	0.01	0.07	0.23	0.58
	SPH	0.001	0.02	0.1	0.4	0.92	1.99
Expanding Time (s)	LSH	0.21	0.51	1.01	1.54	1.53	1.36
	KLSH	0.73	1.22	1.01	1.17	1.05	0.82
	SH	0.63	1.28	1.43	1.28	1.11	0.9
	AGH	0.04	0.25	0.91	1.41	1.38	1.39
	SPH	0.09	0.35	0.89	1.11	0.98	0.85

TABLE VII
50 NN SEARCH TIME (SECONDS/1000 QUERIES) OF KLSH$_{IEH}$ VERSUS PARAMETER p AT HAMMING RADIUS 2

Data Set	KLSH$_{IEH}$ ($k = 50, s = 3$)					KLSH and KD-Tree
	$p = 1$	$p = 5$	$p = 10$	$p = 20$	$p = 30$	
CIFAR10	1.45	2.15	2.83	3.67	4.45	8.21
GIST-1M	0.91	1.34	1.82	2.65	3.38	9.26
SIFT-1M	0.31	0.48	0.65	0.93	1.19	5.22

TABLE VIII
50 NN SEARCH TIME (SECONDS/1000 QUERIES) OF KLSH$_{IEH}$ VERSUS THE PARAMETER k AT HAMMING RADIUS 2

Data Set	KLSH$_{IEH}$ ($p = 10, s = 3$)					KLSH and KD-Tree
	$k = 10$	$k = 25$	$k = 50$	$k = 75$	$k = 100$	
CIFAR10	1.58	2.09	2.83	3.47	4.11	8.21
GIST-1M	1.01	1.33	1.82	2.26	2.69	9.26
SIFT-1M	0.36	0.48	0.65	0.8	0.95	5.22

TABLE IX
50 NN SEARCH TIME (SECONDS/1000 QUERIES) OF KLSH$_{IEH}$ VERSUS PARAMETER s AT HAMMING RADIUS 2

Data Set	KLSH$_{IEH}$ ($p = 10, k = 50$)					KLSH and KD-Tree
	$s = 1$	$s = 2$	$s = 3$	$s = 4$	$s = 5$	
CIFAR10	2.32	2.77	2.83	2.85	2.86	8.21
GIST-1M	1.3	1.68	1.82	1.87	1.89	9.26
SIFT-1M	0.45	0.58	0.65	0.68	0.69	5.22

6 小结与讨论

本文中，我们提出了一个新颖的算法IEH，不用很大的汉明半径获得了较高的召回率。IEH利用非常小的汉明半径，迭代利用少量最近候选点的展开扩展成kNN。用这个方法，我们同时获得了较高的召回率和较低的检索时间。理论分析和试验结果均表明，该方法能极大地提高传统哈希算法的检索结果。更重要的是，KD树是基于真实数据最近邻检索问题的最先进的解决方法，实验结果也表明我们的方法相比KD树获得了更高的性能。在未来的工作中，我们会用FLANN加速离线预处理过程，建立多表以更好地改进线上阶段的性能。除此之外，我们将考虑设计一个自动调整参数（如p、k、s）的机制。

参考文献

[1] A. Andoni and P. Indyk, “Near-optimal hashing algorithms for approx- imate nearest neighbor in high dimensions,”Commun. ACM, vol. 51, no. 1, pp. 117—122, 2008.

[2] L. Arge, M. Berg, H. Haverkort, and K. Yi, “The priority R-tree: A practically efficient and worst-case optimal R-tree,”in Proc. ACM SIGMOD Int. Conf. Manag. Data, Jun. 2004, pp. 347—358.

[3] S. Arya, D. Mount, N. Netanyahu, R. Silverman, and A. Wu, “An optimal algorithm for approximate nearest neighbor searching in fixed dimensions,”J. ACM, vol. 45, no. 6, pp. 891—923, 1998.

[4] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, “When is ‘near- est neighbor’ meaningful?”in Proc. 7th Int. Conf. Database Theory, Jan. 1999, pp. 217—235.

[5] P. Ciaccia, M. Patella, and P. Zezula, “M-tree: An efficient access method for similarity search in metric spaces,”in Proc. 23rd Int. Conf. Very Large Data Bases, Aug. 1997, pp. 426—435.

[6] S. Dasgupta and Y. Freund, “Random projection trees and low dimen- sional manifolds,”in Proc. 40th Annu. ACM Symp. Theory Comput., May 2008, pp. 537—546.

[7] J. Freidman, J. Bentley, and R. Finkel, “An algorithm for finding best matches in logarithmic expected time,”ACM Trans. Math. Softw., vol. 3, no. 3, pp. 209—226, 1997.

[8] Y. Gao, B. Zheng, G. Chen, Q. Li, and X. Guo, “Continuous visible nearest neighbor query processing in spatial databases,”Very Large Databases J., vol. 20, no. 3, pp. 371—396, 2011.

[9] D. G. Lowe, “Distinctive image features from scale invariant keypoints,”Int. J. Comput. Vision, vol. 60, no. 2, pp. 91—110, 2004.

[10] Y. Gong and S. Lazebnik, “Iterative quantization: A procrustean ap- proach to learning binary codes,”in Proc. IEEE Conf. Comput. Vision Pattern Recogn., Jun. 2011, pp. 817—824.

[11] K. Hajebi, Y. Abbasi-Yadkori, H. Shahbazi, and H. Zhang, “Fast approximate nearest-neighbor search with k-nearest neighbor graph,”in Proc. 22nd Int. Joint Conf. Artif. Intell., Jul. 2011, pp. 1312—1317.

[12] J. He, R. Radhakrishnan, S. F. Chang, and C. Bauer, “Compact hashing with joint optimization of search accuracy and time,”in Proc. IEEE Conf. Comput. Vision Pattern Recognit., Jun. 2011, pp. 753—760.

[13] P. Jain, B. Kulis, and K. Grauman, “Fast similarity search for learned metrics,”IEEE Trans. Pattern Anal. Machine Intell., vol. 31, no. 12, pp. 2143—2157, Dec. 2009.

[14] Z. Jin, Y. Hu, Y. Lin, D. Zhang, S. Lin, D. Cai, et al., “Complmentary projection hashing,”in Proc. IEEE Int. Conf. Comput. Vision, Dec. 2013, pp. 257—264.

[15] Z. Jin, C. Li, Y. Lin, and D. Cai, “Density sensitive hashing,”IEEE Trans. Cybern., vol. PP, no. 99, Oct. 2013.

[16] W. Johnson and J. Lindenstrauss, “Extensions of Lipschitz mappings into a hilbert space,”Contemporary Math., vol. 26, pp. 189—206, Jan. 1984.

[17] A. Joly and O. Buisson, “A posteriori multi-probe locality sen- sitive hashing,”in Proc. 16th Int. Conf. Multimedia, Oct. 2008, pp. 209—218.

[18] A. Joly and O. Buisson, “Random maximum margin hashing,”in Proc. IEEE Conf. Comput. Vision Pattern Recognit., Jun. 2011, pp. 873—880.

[19] B. Kulis and T. Darrell, “Learning to hash with binary reconstructive embeddings,”in Proc. Adv. Neural Inf. Process. Syst., Dec. 2008, pp. 1042—1050.

[20] B. Kulis and K. Grauman, “Kernelized locality-sensitive hashing,”IEEE Trans. Pattern Anal. Mach. Intell., vol. 34, no. 6, pp. 2130—2137, Jun. 2012.

[21] Y. Lifshits and S. Zhang, “Combinatorial algorithms for nearest neigh- bors, near-duplicates and small-world design,”in Proc. ACM-SIAM Symp. Discrete Algorithms, Jan. 2009, pp. 318—326.

[22] Y. Lin, R. Jin, D. Cai, and X. He, “Random projection with filtering for nearly duplicate search,”in Proc. 26th AAAI Conf. Artif. Intell., Jul. 2012, pp. 641—647.

[23] Y. Lin, R. Jin, D. Cai, S. Yan, and X. Li, "Compressed hashing," in Proc. IEEE Conf. Comput. Vision Pattern Recognit., Jun. 2013, pp. 446—451.

[24] T. Liu, A. W. Moore, and A. Gray, "New algorithms for efficient high dimensional non-parametric classification," J. Mach. Learn. Res., vol. 7, pp. 1135—1158, 2006.

[25] W. Liu, J. Wang, R. Ji, Y. Jiang, and S. F. Chang, "Supervised hashing with kernels," in Proc. IEEE Conf. Comput. Vision Pattern Recognit., Jun. 2012, pp. 2074—2081.

[26] W. Liu, J. Wang, S. Kumar, and S. F. Chang, "Hashing with graphs," in Proc. 28th Int. Conf. Mach. Learn., Jun./Jul. 2011, pp. 1—8.

[27] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, "Multi-probe LSH: Efficient indexing for high-dimensional similarity search," in Proc. 33rd Int. Conf. Very Large Data Bases, Sep. 2007, pp. 950—961.

[28] Y. Mu, J. Shen, and S. Yan, "Weakly-supervised hashing in kernel space," in Proc. IEEE Conf. Comput. Vision Pattern Recognit., Jun. 2010, pp. 3344—3351.

[29] M. Muja and D. G. Lowe, "Fast approxiamte nearest neighbors with automatic algorithm configuration," in Proc. Int. Conf. Vision Theory Applicat., Feb. 2009, pp. 331—340.

[30] M. Muja and D. Lowe, FLANN: Fast Library for Approximate Nearest Neighbors User Manual, [Online]. Available: http://www.cs.ubc.ca/ research/flann/uploads/FLANN/flann manual-1.8.4.pdf, 2009.

[31] M. Norouzi and D. J. Fleet, "Minimal loss hashing for compact binary codes," in Proc. 28th Int. Conf. Mach. Learn., Jun./Jul. 2011, pp. 353—360.

[32] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," Int. J. Comput. Vision, vol. 42, no. 3, pp. 145—175, 2001.

[33] R. Panigrahy, "Entropy based nearest neighbor search in high dimen- sions," in Proc. 17th Annu. ACM-SIAM Symp. Discrete Algorithms, Jan. 2006, pp. 1186—1195.

[34] R. Paredes and E. Chvez, "Using the k-nearest neighbor graph for proximity searching in metric spaces," in Proc. 12th Int. Conf. String Process. Inform. Retrieval, , Nov. 2005, pp. 127—138.

[35] J. P. Heo, Y. Lee, J. He, S. F. Chang, and S. E. Yoon, "Spherical hashing," in Proc. IEEE Conf. Comput. Vision Pattern Recogn., Jun. 2012, pp. 2957—2964.

[36] M. Raginsky and S. Lazebnik, "Locality sensitive binary codes from shift-invariant kernels," in Proc. Adv. Neural Inform. Process. Syst., Dec. 2009, pp. 1509—1517.

[37] S. Ray, S. Bandyopadhyay, and S. Pal, "Dynamic range-based distance measure for microarray expressions and a fast gene-ordering algorithm," IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 37, no. 3, pp. 742—749, Jun. 2007.

[38] J. S. Beis and D. G. Lowe, "Shape indexing using approximate nearest- neighbour search in high-dimensional spaces," in Proc. IEEE Conf. Comput. Vision Pattern Recognit., Jun. 1997, pp. 1000—1006.

[39] C. Silpa-Anan and R. Hartley, "Optimised KD-trees for fast image descriptor matching," in Proc. IEEE Conf. Comput. Vision Pattern Recognit., Jun. 2008, pp. 1—8.

[40] T. Trzcinski, V. Lepetit, and P. Fua, "Thick boundaries in binary space and their influence on nearest-neighbor search," Pattern Recognit. Lett., vol. 33, no. 16, pp. 2173—2180, 2012.

[41] J. Wang, S. Kumar, and S.-F. Chang, "Sequential projection learning for hashing with compact codes," in Proc. 27th Int. Conf. Mach. Learn., Jun. 2010, pp. 1127—1134.

[42] J. Wang, S. Kumar, and S. F. Chang, "Semi-supervised hashing for large scale search," IEEE Trans. Pattern Anal. Mach. Intell., vol. 34, no. 12, pp. 2074—2081, Dec. 2012.

[43] J.-L. Wang and C.-Y. Chang, "Fast retrieval of electronic messages that contain mistyped words or spelling errors," IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 27, no. 3, pp. 441—451, Jun. 1997.

[44] X. Wang, D. Shasha, and K. Zhang, "Metricmap: An embedding technique for processing distance-based queries in metric spaces," IEEE Trans. Systems, Man, Cybern. B, Cybern., vol. 35, no. 5, pp. 973—987, Oct. 2005.

[45] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in Proc. Adv. Neural Inform. Process. Syst., Dec. 2008, pp. 1753—1760.

[46] C. Wu, J. Zhu, D. Cai, C. Chen, and J. Bu, "Semi-supervised nonlinear hashing using bootstrap sequential projection learning," IEEE Trans. Knowl. Data Eng., vol. 25, no. 6, pp. 1380—1393, Jun. 2013.

[47] B. Xu, J. Bu, Y. Lin, C. Chen, X. He, and D. Cai, "Harmonious hashing," in Proc. 23rd Int. Joint Conf. Artif. Intell., Aug. 2013, pp. 1820—1826.

[48] H. Xu, J. Wang, Z. Li, G. Zeng, S. Li, and N. Yu, "Complementary hashing for approximate nearest neighbor search," in Proc. IEEE Int. Conf. Comput. Vision, Nov. 2011, pp. 1631—1638.

[49] P. Xu, C.-H. Chang, and A. Paplinski, "Self-organizing topological tree for online vector quantization and data clustering," IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 35, no. 3, pp. 515—526, Jun. 2005.

[50] M.-S. Yang and C.-H. Chen, "On the edited fuzzy k-nearest neighbor rule," IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 28, no. 3, pp. 461—466, Jun. 1998.

[51] Z. Yu, D. Cai, and X. He, "Error-correcting output hashing in fast similarity search," in Proc. 2nd Int. Conf. Internet Multimedia Comput. Service, Dec. 2010, pp. 7—10.

[52] D. Zhang, G. Yang, Y. Hu, Z. Jin, D. Cai, and X. He, "A unified ap- proximate nearest neighbor search scheme by combining data structure and hashing," in Proc. 23rd Int. Joint Conf. Artif. Intell., Aug. 2013, pp. 681—687.

[53] S. Zhang, H.-S. Wong, Z. Yu, and H. Ip, "Hybrid associative retrieval of three-dimensional models," IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 40, no. 6, pp. 1582—1595, Dec. 2010.