Efficient locality-sensitive hashing over high-dimensional data streams

Efficient Locality-Sensitive Hashing Over High-Dimensional Data Streams

Chengcheng Yang[1], Dong Deng[2], Shuo Shang*[3], Ling Shao[4]
[1]*King Abdullah University of Science and Technology*, [2]*Rutgers University*
[3]*University of Electronic Science and Technology of China*
[4]*Inception Institute of Artificial Intelligence*
[1]chengcheng.yang@kaust.edu.sa, [2]d.deng@rutgers.edu, [3]jedi.shang@gmail.com, [4]ling.shao@inceptioniai.org

***Abstract*—Approximate Nearest Neighbor (ANN) search in high-dimensional space is a fundamental task in many applications. Locality-Sensitive Hashing (LSH) is a well-known methodology to solve the ANN problem with theoretical guarantees and empirical performance. We observe that existing LSH-based approaches target at the problem of designing search optimized indexes, which require a number of separate indexes and high index maintenance overhead, and hence impractical for high-dimensional streaming data processing. In this paper, we present PDA-LSH, a novel and practical disk-based LSH index that can offer efficient support for both updates and searches. Experiments on real-world datasets show that our proposal outperforms the state-of-the-art schemes by up to 10× on update performance and up to 2× on search performance.**

I. Introduction

Nearest Neighbor (NN) search in high-dimensional Euclidean space plays an essential role in many applications. Despite there are numerous studies on NN search in multi-dimensional space, it was shown that their performance will sharply degenerate as the dimensionality increases due to the "curse of dimensionality" [9].

Locality-Sensitive Hashing (LSH) is a widely used technique that has shown significant promise in dealing with the curse of high-dimensionality. However, as LSH was originally designed to find objects within a fixed radius, to ensure quality guarantees, it has to build indexes for different radii. In this case, hundreds or thousands of hash tables are constructed, which result in expensive space cost and search cost.

Many LSH based variants have been proposed to alleviate the limitations of LSH [2], [3], [5], [8]. At their core, these approaches aim to build an index with smaller space cost and search cost while preserving an acceptable accuracy. Early work adopts the multi-probe strategy [5] to reduce the number of maintained hash tables. Unfortunately, the space saving trades away the theoretical guarantees on the quality of query results. Recently, it has been proposed to store each LSH projection in separate B-Trees, including C2LSH [2] and QALSH [3]. They can gradually enlarge the query radius by performing B-Tree range searches. However, they suffer from the poor random writes in each B-Tree when processing updates. Other work [8] proposes to projects high-dimensional data objects into a low-dimensional space so that they can be indexed by existing multi-dimensional indexes, such as R-tree. Nevertheless, it cannot scale to more projections since the R-Tree doesn't work well when the dimensionality is more than 6. Therefore, the quality of query result is not stable. In summary, existing methods cannot make a trade-off between update efficiency, search efficiency and quality guarantees, but all of which are crucial factors for efficiently indexing high-dimensional data streams.

In this paper, we propose a new disk-based LSH index that offers fast c-ANN searches with low maintenance cost, while also holding theoretical guarantees on the result quality. First, we follow C2LSH and QALSH to use a set of single LSH functions as base functions, so that we can enjoy the benefits of performing multiple-radii searches within one index. We adopt the LSM-Tree with *cascading update* technique to facilitate efficient updates on each LSH function. Then, different from C2LSH and QALSH that apply Chernoff tail bounds to derive the tail probability of a binomial distribution, which represents the probability of two objects colliding more than a given threshold under the set of base LSH functions, we utilize cumulative distribution function (CDF) of the normal distribution to approximate it more reasonably. As a consequence, the number of LSH functions derived by our method is much smaller than previous studies. To improve the search performance, we combine collision number and precise projection distance information to select the candidates more effectively. In the following, we refer to such a proposal as PDA-LSH (Projection Distance Aware LSH).

II. The PDA-LSH Framework

In this section, we describe the indexing and search methods of PDA-LSH.

A. Indexing Strategy

We follow QALSH to use a base of m single *query-aware* LSH functions to enjoy the benefit of multi-radii search and *query-aware* bucket partitions. The indexing procedure essentially projects each d-dimensional object o along m random lines, and stores the projected values in m separate indexes. The difference from QALSH is that PDA-LSH uses the write-friendly LSM-Tree to index the projection pairs $\langle h_i(o), ID(o)\rangle$, where $h_i(o)$ is the i-th projection of object o and $ID(o)$ represents its unique identifier.

*Corresponding author.

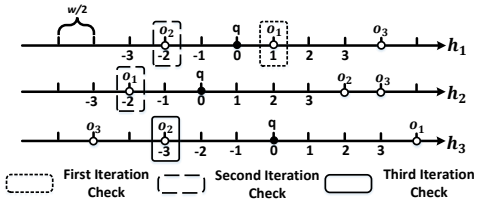


Fig. 1. Searching objects in ascending order of projection distances

Algorithm 1: c-ANN of PDA-LSH

Input: q: the query object; m: the number of LSH functions; l: the collision threshold; thr: the threshold of the squared sum of l projection distances for search radius $R = 1$; w: the initial bucket width for $R = 1$; βn: maximum number of candidates.

Output: the nearest object in the set $\mathcal{C}$ of candidates.

$\mathcal{C} \leftarrow \phi$;
$\mathcal{H}_{\mathcal{M}} \leftarrow$ min-heap with maximum size of βn;
Compute $h_i(q)$ for $i \in \{1, 2, \cdots, m\}$;
Locate each $h_i(q)$ in the corresponding LSM-Tree;
while $\mathcal{C} < \beta n$ **do**
  $o \leftarrow$ pick from all LSM-Trees the next object with smallest projection distance to q;
  $i \leftarrow$ the label of LSM-Tree o comes from;
  $R \leftarrow \frac{|h_i(o)-h_i(q)|}{w/2}$;
  $t \leftarrow thr \times R^2$;
  $Col(o) \leftarrow Col(o) + 1$;
  $Pdist(o) \leftarrow Pdist(o) + (h_i(o) - h_i(q))^2$;
  if *Col(o)* == l **then**
    Push $\langle Pdist(o), ID(o)\rangle$ into the min-heap $\mathcal{H}_{\mathcal{M}}$;
  end
  while $\mathcal{H}_{\mathcal{M}}.root.Pdist \leq t$ **do**
    $o \leftarrow$ pop an object from $\mathcal{H}$;
    $\mathcal{C} \leftarrow \mathcal{C} \cup o$;
    if $|\{o|o \in \mathcal{C} \wedge dist(o, q) \leq c \times R\}| \geq 1$ **then**
      break;
    end
  end
end
return *the nearest object* $o_{min} \in \mathcal{C}$;

B. Approximate Nearest Neighbor Search

The main idea of PDA-LSH is that it uses the precise projection distance to estimate the distance in original space. Moreover, in order to limit the error of estimation, PDA-LSH must collect "abundant" projection distance information to differentiate the "near objects" and "far objects".

To implement the above high level idea, PDA-LSH adopts a threshold based strategy. More specifically, given pre-specified thresholds of collision number l and projection distance t, if a data object o collides with a query object q under at least l LSH functions, and the squared sum of the l projection distances is at most t, then it is a good candidate of being the c-ANN of q. The process can be seen in Algorithm 1. An example of the search procedure is shown in Fig. 1.

c-k-ANN extension. Algorithm 1 can also be easily extended to support the c-k-ANN search by changing the terminating conditions as the following: (1) There exist k objects in $\mathcal{C}$ whose Euclidean distances to q are at most cR; (2) $\beta n + k - 1$ candidates in $\mathcal{C}$ have been found.

III. Parameter Settings

In this section, we illustrate how internal parameters are computed to ensure the correctness of PDA-LSH. Previous study [4] has shown that if objects are accessed in ascending order of their projection distances, the correctness of c-ANN is guaranteed if the algorithm has guarantees on the (R, c)-NN problem. Thus, we only consider the (R, c)-NN problem. Due to space constraints, we leave all the proofs in our extended technical report.

A. PDA-LSH for (R, c)-NN

PDA-LSH can directly solve the (R, c)-NN problem by imposing buckets with width Rw. To ensure the correctness of PDA-LSH, the following two properties, which support the two terminating conditions of PDA-LSH, should hold at the same time with constant probability:

- $\mathcal{P}_1$. $Col(o) \geq l \wedge \sum_{i \in \mathcal{L}(o)} (h_i(o) - h_i(q))^2 \leq thr \times R^2$, given $dist(o, q) \leq R$.
- $\mathcal{P}_2$. $|\{o|Col(o) \geq l \wedge \sum_{i \in \mathcal{L}(o)} (h_i(o) - h_i(q))^2 \leq thr \times R^2 \wedge dist(o, q) > cR\}| < \beta n$.

where $\mathcal{L}(o)$ denotes the set of o's l projections within a base of m projections whose distances are smallest to q. According to the theoretical analysis of LSH based methods [3], the internal parameters of PAD-LSH should be carefully set to ensure: (1) $Pr[\mathcal{P}_1] \geq 1 - \delta$, where δ is the false negative probability; and (2) $Pr[\mathcal{P}_2] \geq \frac{1}{2}$. On this basis, we have the lower bound on the probability of both $\mathcal{P}_1$ and $\mathcal{P}_2$ being true as $\frac{1}{2} - \delta$.

Given the input parameters c, β and δ specified by the user [2], [3], three parameters of PDA-LSH are required to be computed. One is the base cardinality of LSH functions, which is denoted as m, and the others are thresholds of collision number and projection distance, which are denoted by l and thr, respectively.

For ease of reference, given an object o with distance s from the query q, we use $e_1(s, R)$ to denote the event $Col(o) \geq l$ under radius R, and use $e_2(s, R)$ to denote the event $\sum_{i \in \mathcal{L}(o)} (h_i(o) - h_i(q))^2 \leq thr \times R^2$. Accordingly, the assertion of $\mathcal{P}_1$ and $\mathcal{P}_2$ is assured by the following theorem.

Theorem 1. *For any $c > 1$ and $R > 0$, with carefully chosen m, l and thr that satisfy: (1) $Pr[e_1(s, R) \cap e_2(s, R)] \geq 1 - \delta$ if $s \leq R$; (2) $Pr[e_1(s, R) \cap e_2(s, R)] < \frac{\beta}{2}$ if $s > cR$, then the probability that both $\mathcal{P}_1$ and $\mathcal{P}_2$ hold is at least $\frac{1}{2} - \delta$.*

Now our aim is to choose a correct set of m, l and thr that satisfy the requirement of Theorem 1. In addition, the base cardinality m is simply the number of separate indexes in PDA-LSH, and a small number of indexes would lead to small space cost and index access cost. Therefore, it's best to choose a correct set of parameters with the minimum value of m. Given m, l and thr, to verify whether Theorem 1 is satisfied, we need to compute the probability of $e_1(s, R) \cap e_2(s, R)$. By Bayes' theorem, we have $Pr[e_1(s, R) \cap e_2(s, R)] = Pr[e_1(s, R)] \times Pr[e_2(s, R)|e_1(s, R)]$.

We first introduce how to calculate $Pr[e_1(s,R)]$. For any object o with distance s from the query q, let $p(s,R)$ denote their collision probability under a *query-aware* LSH function with bucket width Rw. Due to the stability of standard normal distribution $\mathcal{N}(0,1)$, we have the Lemma 1 as follows:

Lemma 1. *For any $o,q \in \mathbb{R}^d$, $h_i(o) - h_i(q)$ is distributed according to the normal distribution $\mathcal{N}(0, dist^2(o,q))$ [7].*

By Lemma 1, $p(s,R)$ can be computed as:

$$p(s,R) = Pr[|h_i(o) - h_i(q)| \leq R \cdot \frac{w}{2}] = \int_{-\frac{Rw}{2s}}^{\frac{Rw}{2s}} \phi(x)\,dx$$

where $\phi(x)$ is the probability density function (PDF) of $\mathcal{N}(0,1)$. Moreover, as each of the m LSH functions is generated randomly and independently, then we have $Pr[e_1(s,R)] = \sum_{i=l}^{m} \binom{m}{i} p(s,R)^i (1-p(s,R))^{m-i}$.

After $Pr[e_1(s,R)]$ is available, the biggest challenge, however, is that there is no closed form expression for the conditional probability $Pr[e_2(s,R)|e_1(s,R)]$, and compute it exactly is very expensive if not impossible. Consequently, it is computationally prohibitive to find the optimal parameters. Alternatively, we resort to a heuristic and practical method, which finds relatively small values of m and l that lead to satisfactory search performance without abandoning quality guarantees. Next, we present how to compute them.

B. Computing Internal Parameters

The first step in our parameter setting is to choose m properly, so that we have "enough" projection information to make a distinction between the "near objects" and "far objects". To ensure that there always exists at least one correct setting, we propose to choose a conservative value of m. Similar to QALSH, we first quantize the projection distances into collision number to enjoy a closed form expression of the probability analysis. Then, by choosing an appropriate m, we aim to make sure that there must exist a set of m, l and $thr = +\infty$ which satisfy Theorem 1 (PDA-LSH is actually equivalent to QALSH in this case).

Obviously, given the search radius R and two objects o, q with distance s, their collision number under m hash functions follows the binomial distribution $\mathcal{B}(m, p(s,R))$. Thus, the probability of two objects colliding more than a given threshold is exactly the right-tail probability of a binomial distribution. Different from QALSH that applies Chernoff tail bounds to derive the tail probability, we propose to approximate $\mathcal{B}(m, p(s,R))$ by the normal distribution $\mathcal{N}(m \cdot p(s,R), m \cdot p(s,R) \cdot (1-p(s,R)))$, and use its CDF to get a more accurate estimation of the right-tail probability. Suppose that m is given, we aim to find a value of l that ensures: (1) the right-tail probability of $\mathcal{N}(m \cdot p_1, m \cdot p_1 \cdot (1-p_1))$ associated with l is at least $1-\delta$, where $p_1 = p(R,R)$; (2) the right-tail probability of $\mathcal{N}(m \cdot p_2, m \cdot p_2 \cdot (1-p_2))$ associated with l is at most $\frac{\beta}{2}$, where $p_2 = p(cR,R)$. After applying the standard normal distribution transformation, we have

$$l \leq mp_1 + \Phi^{-1}(\delta)\sqrt{mp_1(1-p_1)} = l_{upper}$$

$$l \geq mp_2 + \Phi^{-1}(1-\frac{\beta}{2})\sqrt{mp_2(1-p_2)} = l_{lower}.$$

On the base of above bounds, the minimum m that satisfies $l_{upper} \geq l_{lower}$ can be computed as $m_{min} = \lceil (\frac{\Phi^{-1}(1-\frac{\beta}{2})\sqrt{p_2(1-p_2)} - \Phi^{-1}(\delta)\sqrt{p_1(1-p_1)}}{p_1 - p_2})^2 \rceil$.

Once m is determined, our goal is to choose a correct set of l and thr where l is minimized. The smaller l is, the earlier the candidates can be found, and the less index access cost is required. However, for each specified l, recall that the conditional probability $Pr[e_2(s,R)|e_1(s,R)]$ has no closed-form expression. Alternatively, we use the Monte Carlo method [1] to estimate it. Before jumping into details of setting l and thr, we first introduce two useful Lemmas that will be used to simplify our parameter setting procedure.

Lemma 2. *For any $R > 0$ and $o_1, o_2, q \in \mathbb{R}^d$, let $s_1 = dist(o_1, q)$ and $s_2 = dist(o_2, q)$. If $0 < s_1 < s_2$, we have $Pr[e_1(s_1,R)] > Pr[e_1(s_2,R)]$, $Pr[e_2(s_1,R)|e_1(s_1,R)] > Pr[e_2(s_2,R)|e_1(s_2,R)]$ and $Pr[e_1(s_1,R) \cap e_2(s_1,R)] > Pr[e_1(s_2,R) \cap e_2(s_2,R)]$.*

Lemma 3. *For any $c,s,R > 0$, $Pr[e_1(s,R) \cap e_2(s,R)] = Pr[e_1(cs,cR) \cap e_2(cs,cR)]$.*

Lemma 2 indicates that, when setting parameters to satisfy Theorem 1, we only need to bound the probability $Pr[e_1(s,R) \cap e_2(s,R)]$ for two critical distances, i.e., $Pr[e_1(R,R) \cap e_2(R,R)] \geq 1-\delta$ and $Pr[e_1(cR,R) \cap e_2(cR,R)] \leq \frac{\beta}{2}$. For this purpose, we first introduce how to estimate them with respect to the specific l and thr. Moreover, based on Lemma 3, we can normalize the problem of estimating $Pr[e_1(R,R) \cap e_2(R,R)]$ and $Pr[e_1(cR,R) \cap e_2(cR,R)]$ to estimating $Pr[e_1(1,1) \cap e_2(1,1)]$ and $Pr[e_1(c,1) \cap e_2(c,1)]$. Next, we propose a Monte Carlo method to estimate them.

Take $Pr[e_1(1,1) \cap e_2(1,1)]$ as an example, we first randomly sample N points on a $(d-1)$-sphere with radius 1 [6], where d is the dimensionality of dataset. Note that the center of the sphere does not affect the estimation because the projection distance is only related to the distance in original space (see Lemma 1), thus we assume that the default center is the origin. For specific l and thr, we then find N' points, each of which meets the following limits: (1) there are at least l projections falling in the interval $[-\frac{w}{2}, \frac{w}{2}]$; (2) there exists a set of l projections whose square sum is at most thr. By the law of large numbers, $Pr[e_1(1,1) \cap e_2(1,1)]$ can be approximated by $\frac{N'}{N}$ if N is large enough. The estimation of $Pr[e_1(c,1) \cap e_2(c,1)]$ can be performed in a similar way, the only difference is that we randomly sample points on a $(d-1)$-sphere with radius c.

With the aforementioned methods of probability estimation, we inspect the values of l in ascending order, and select the minimum l, which ensures that there exists a correct thr satisfying $Pr[e_1(1,1) \cap e_2(1,1)] \geq 1-\delta \wedge Pr[e_1(c,1) \cap e_2(c,1)] \leq \frac{\beta}{2}$ as the result.

IV. Experimental Results

In this section, we evaluate the efficiency and accuracy of our proposed method. We compare PDA-LSH with QALSH

TABLE I
INTERNAL PARAMETERS AND INDEX SIZE OF PDA-LSH, QALSH AND SRS

Datasets	d	n	PDA-LSH			QALSH			SRS		
			m	l (l')	Index Size	m	l	Index Size	m	l	Index Size
Sift	128	1,000,000	45	33 (36)	346.09 MB	83	63	905.92 MB	6	-	38.7 MB
Msong	420	994,020	45	33 (36)	343.32 MB	83	63	886.54 MB	6	-	37.6 MB

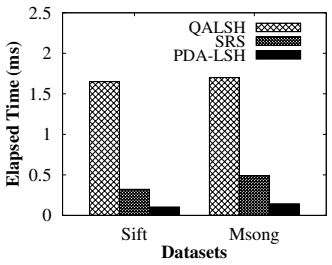
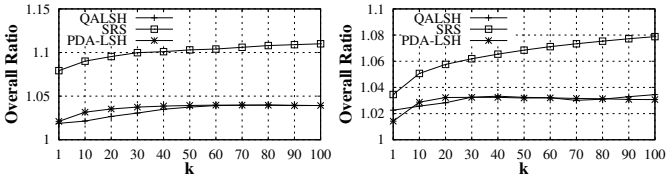
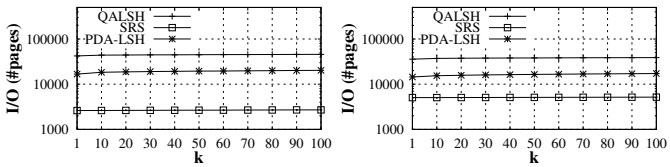


Fig. 2. Update performance comparison.



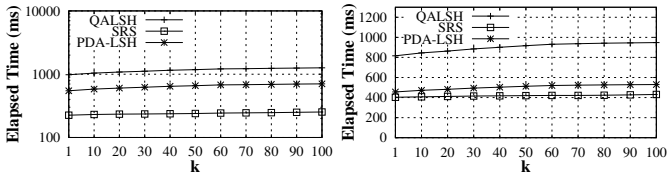
(a) Sift (b) Msong

Fig. 3. Overall ratio comparison varying k.



(a) Sift (b) Msong

Fig. 4. I/O cost comparison varying k.



(a) Sift (b) Msong

Fig. 5. Running time comparison varying k.

and SRS, which are two state-of-the-art I/O efficient algorithms with theoretical guarantees on the result quality.

A. Experimental Setup

The experiments were ran on a workstation powered by Intel Xeon Gold-6148 CPU on Linux (Ubuntu 16.04), having a 15K RPM disk. All the experiments were conducted using the direct I/O mode. We experimented on two real-world datasets. The characteristics of the datasets are summarized in Table I. We set the page size to 8 KB. We use 4-byte integers to store the object IDs and 4-byte floats to store the hash projections. To be fair, the success probability of all algorithms is set to $\frac{1}{2} - \frac{1}{e}$. The default approximate ratio c is set to 2. Both of PDA-LSH and QALSH use $w = 2.719$ so that the gap between p_1 and p_2 is maximized.

B. Experimental Results

Space consumption. Table I shows the index size of the three algorithms. SRS has the smallest size because it uses much fewer projections than other methods. The index size of PDA-LSH is 2.6× smaller than QALSH. There are two primary reasons: one is that PDA-LSH uses fewer separate indexes. Another reason is that the LSM-Tree is more space efficient than the B-Tree when handling updates, because it has no split/merge operations.

Update performance. From Fig. 2, we can see that PDA-LSH always achieves the best performance.

Search performance. We follow the previous studies [3], [8] and adopt the following three metrics in our search performance evaluations: overall ratio, I/O cost and running time. The results are shown in Fig. 3-5. We observe that both PDA-LSH and QALSH offer much higher accuracies than SRS. SRS has the smallest I/O cost due to its small index size. However, this is often at the expense of sacrificing result accuracy. We also observe that the I/O cost of PDA-LSH is 2.3-2.6× less than QALSH while offering comparable result accuracies. When the data dimensionality is large, PDA-LSH has comparable performance than SRS although its I/O cost is higher. The main reason is that PDA-LSH benefits from the fast sequential I/Os brought by LSM-Tree.

V. CONCLUSION

In this paper, we addressed the problem of indexing high-dimensional data over streams, in which case the update efficiency, search efficiency and quality guarantees are all crucial. We proposed an efficient LSH index, namely PDA-LSH, for high-dimensional streaming data processing. The experimental results demonstrate the efficiency and accuracy of our proposal.

REFERENCES

[1] K. Binder. Monte carlo simulations in statistical physics. In *Encyclopedia of Complexity and Systems Science*, pages 5667–5677. 2009.

[2] J. Gan, J. Feng, Q. Fang, and W. Ng. Locality-sensitive hashing scheme based on dynamic collision counting. In *SIGMOD*, pages 541–552, 2012.

[3] Q. Huang, J. Feng, Q. Fang, W. Ng, and W. Wang. Query-aware locality-sensitive hashing scheme for l_p norm. *VLDB J.*, 26(5):683–708, 2017.

[4] W. Liu, H. Wang, Y. Zhang, W. Wang, and L. Qin. I-LSH: I/O efficient c-approximate nearest neighbor search in high-dimensional space. In *ICDE*, pages 1670–1673, 2019.

[5] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe LSH: efficient indexing for high-dimensional similarity search. In *VLDB*, pages 950–961, 2007.

[6] M. E. Muller. A note on a method for generating points uniformly on n-dimensional spheres. *Commun. ACM*, 2(4):19–20, 1959.

[7] R. Panigrahy. Entropy based nearest neighbor search in high dimensions. In *SODA*, pages 1186–1195, 2006.

[8] Y. Sun, W. Wang, J. Qin, Y. Zhang, and X. Lin. SRS: solving c-approximate nearest neighbor queries in high dimensional euclidean space with a tiny index. *PVLDB*, 8(1):1–12, 2014.

[9] R. Weber, H. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, pages 194–205, 1998.