CLSH: Cluster-based Locality-Sensitive Hashing

Xiangyang Xu, Tongwei Ren, Gangshan Wu*
State Key Laboratory for Novel Software Technology
Nanjing University, Nanjing, China
xiangyang.xu@smail.nju.edu.cn, {rentw, gswu}@nju.edu.cn

ABSTRACT

Locality-sensitive hashing (LSH) usually consumes large memory in similarity search, which limits its scalability for large scale applications. In this paper, we propose a novel cluster-based locality-sensitive hashing (CLSH) approach, which extends the conventional LSH framework and aims at indexing and searching large scale high-dimensional datasets. We first utilize a clustering algorithm to partition the raw feature dataset into clusters, and map these clusters to a distributed cluster. Then, LSH method is applied to construct the index for each cluster, and we present two criteria to choose the cluster(s) for further detailed search in order to improve the search quality. This proposed framework comes with following properties. Firstly, CLSH can cope with large scale feature dataset. Secondly, the generated clusters can guide the feature dataset automatical mappings to a distributed cluster. After that, the search time can be reduced a lot by searching on multiple computing nodes. Experiments show that the proposed approach outperforms the existing approaches in terms of efficiency and scalability.

Categories and Subject Descriptors

H.3.1 [**Content Analysis and Indexing**]: Indexing methods; H.3.3 [**Information Search and Retrieval**]: Clustering, Search process

General Terms

Algorithm, Experimentation, Performance

Keywords

Approximate Nearest Neighbor search, clustering, Locality-Sensitive Hashing, distributed cluster, high-dimensional indexing

1. INTRODUCTION

Nearest neighbor search, also known as similarity search, plays an important role in multimedia applications, such as information retrieval, data analysis and object recognition [2, 8, 7, 14]. Tree-based search methods, including k-d tree and R-tree, usually have good performance for nearest neighbor search on low-dimensional features, but they inevitably suffer enormous negative influence on search performance when the feature dimensionality increases. To improve the search performance on high-dimensional features, Approximate Nearest Neighbor (ANN) search was proposed, which aims at balancing search result quality and response time by only providing the approximate results in nearest neighbor search [1]. It has only small difference in search results to the exact nearest neighbor search when the user's quality notion is accurately captured, which is good enough for most practical applications [3]. In ANN search, hashing based methods is dominant for their insensitivity to feature dimensionality, in which Locality-Sensitive Hashing (LSH) [5, 3] is one kind of the pioneering hashing based ANN search and widely used methods. Recently, various extensional works of LSH were presented, such as multi-probe LSH [11], query-adaptive LSH [6], etc., and amounts of learning based hashing methods were proposed, including spectral hashing (SH) [17], semi-supervised hashing (SSH) [16], weakly-supervised hashing [12] and kernelized LSH (KLSH) [9], etc. These methods substantially reduce the searching time while preserving the comparable search quality. However, they are all designed in the centralized settings and their abilities are limited by the memory capacity of single computing node. Therefore, their scalability is severely limited by the scale of feature dataset and the ability of the single computing node.

To overcome the above problem, we propose a novel cluster-based locality-sensitive hashing (CLSH) framework, which aims to extend LSH method for indexing and searching large scale high-dimensional feature datasets. Here, "cluster-based" has two meanings. In one respect, our approach applies clustering on the raw feature dataset, and constructs the index for each cluster. In the other respect, our approach is carried out on a distributed cluster which comprises of multiple computing nodes. Figure 1 shows the overview of our approach. First, we utilize a clustering algorithm to partition the raw feature dataset into clusters. Then, the feature dataset are automatically mapped to different computing nodes with the guide of these clusters. After this, for each cluster, LSH method is applied to construct the index. In the nearest neighbor search phrase, one cluster or several clusters are selected for further detailed search to obtain retrieval results and the search time will be reduced a lot.

Compared to the primary LSH and other hashing based ANN methods, our approach has the following advantages:

- CLSH can cope with larger scale feature dataset by applying LSH method on each cluster instead of the whole feature dataset;

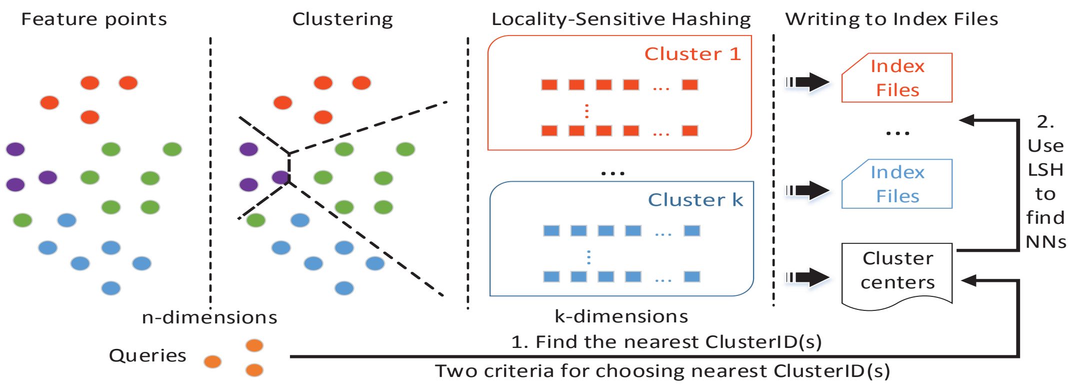


Figure 1: Overview of the proposed CLSH.

- The generated clusters can guide feature dataset automatical mappings to a distributed cluster;
- CLSH can reduce the search time by carrying out searching on multiple computing nodes together instead of searching on a single node.

The rest of this paper is organized as follows. First, we review some preliminaries for our CLSH approach in Section 2. In Section 3, we present a detailed description of the CLSH. Next we demonstrate our experiments, along with discussions in Section 4. Finally, Section 5 brings conclusions and some remarks on the future work.

2. PRELIMINARIES

In the following, we briefly present the preliminaries, including clustering and LSH, which are required to understand the remainders.

Clustering concerns with grouping a set of objects into clusters. Clustering has some desired properties, such as the intra-cluster is homogeneous and the inter-cluster should be separate enough. In other words, objects in the same cluster should be much more similar to each other than those in different clusters. Clustering is widely used in many fields, such as media information retrieval and media analysis.

Generic LSH scheme was proposed in [5] for (R, c)-NN search problem. This scheme ensures that the hashing functions of any LSH families can be used to solve the similarity search problem efficiently. Let S be a d-dimensional feature points collection, $D : S \times S \to R$ is similarity metric, an LSH family is defined as follows: A family $\mathcal{H} = \{h : S \to U\}$ is called (r_1, r_2, p_1, p_2)-*sensitive* under metric D, if for any $\mathbf{p}, \mathbf{q} \in S$:

- If $D(\mathbf{p}, \mathbf{q}) \leq r_1$, then $\Pr_{h\in\mathcal{H}}(h(\mathbf{p}) = h(\mathbf{q})) \geq p_1$;
- If $D(\mathbf{p}, \mathbf{q}) \geq r_2$, then $\Pr_{h\in\mathcal{H}}(h(\mathbf{p}) = h(\mathbf{q})) \leq p_2$.

Where, $r_1 < r_2$ and $p_1 > p_2$. Intuitively, using the hashing function from LSH family, the collision probability of the near points is much higher than the far ones.

The generic LSH scheme works as follows:

1. Index construction: For an integer k, define an LSH function family $\mathcal{G} = \{g : S \to U^k\}$ by concatenating k LSH functions, i.e. $g(\mathbf{p}) = (h_1(\mathbf{p}), \cdots, h_k(\mathbf{p}))$. For an integer L, independently and uniformly choose L functions $g_1(\mathbf{p}), \cdots, g_L(\mathbf{p})$ from $\mathcal{G}$ at random. For every feature point $\mathbf{p} \in S$, store its index key in the buckets $g_1(\mathbf{p}), \cdots, g_L(\mathbf{p})$;
2. Search process: For a query point $\mathbf{q}$, the points $\mathbf{p}_1, \cdots, \mathbf{p}_n$ in the buckets $g_1(\mathbf{q}), \cdots, g_L(\mathbf{q})$ are the query's ANN candidates. For each point $\mathbf{p}_i (i = 1, \cdots, n)$, if $D(\mathbf{q}, \mathbf{p}_i) \leq r$, $\mathbf{p}_i$ is the query's ANN results.

3. APPROACH

For the purpose of scalable high-dimensional indexing and searching, the CLSH for large scale high-dimensional datasets indexing and searching approach is presented in this section. In reality, objects (images, videos, etc.) often have inherent structures and belong to some categories. For instance, images are often characterized into people, landscapes and cars; videos are often characterized into movies, news and TV shows. Therefore, in the CLSH framework, we group the feature dataset into clusters followed by an LSH. Consequently, each cluster may capture a particular aspect of the dataset. However, searching in only one cluster is too harmful to return high quality search results when the query locates near the cluster border. Two methods are presented to choose the cluster(s) for further detailed LSH search in order to facilitate the search quality complementarily.

3.1 Index Construction

The CLSH index construction algorithm is shown in Algorithm 1. Before we construct the index, we first employ a clustering algorithm to group the feature points into clusters as Figure 1 demonstrated. In this paper, we adopt k-means, one of the most popular clustering algorithms, in our CLSH framework for further investigation. After the clusters are generated, the cluster center is the guideline for automatically delivering the feature points to different computing nodes, each cluster corresponds to one computing node.

Next, we apply an LSH algorithm to hash each feature point in these clusters. In CLSH framework, we employ p-stable distribution based LSH [3] which is based on the l_p norm metric and is one of the most popular variations of above-mentioned generic LSH scheme. The hashing function of p-stable distribution based LSH families is defined as:

$$h(\mathbf{p}) = \lfloor \frac{\mathbf{a} \cdot \mathbf{p} + b}{w} \rfloor \ , \quad (1)$$

where $\mathbf{a}$ is a d-dimensional vector with elements independently randomized from a p-stable distribution and b is a real number bias uniformly randomized from $[0, w]$. In this LSH scheme, the collision probability of any two feature points $\mathbf{p}_1$ and $\mathbf{p}_2$ is:

$$\Pr(h(\mathbf{p}_1) = h(\mathbf{p}_2)) = p(x) = \int_0^w \frac{1}{x} f_p(\frac{t}{x}) \frac{1-t}{x} dt \ , \quad (2)$$

where $x = \|\mathbf{p}_1 - \mathbf{p}_2\|_p$, $f_p(\cdot)$ is the absolute value of the p-stable distribution's probability density function (PDF).

```
Algorithm 1: CLSH Index Construction Algorithm
  Input  : The number of clusters $nc$,
           The set of feature points $\{\mathbf{p}_i\}$,
           Random vectors $\{\mathbf{a}_i\}$,
           Random bias $\{b_i\}$, Quantization width $w$,
           Concatenate number $k$, Group number $L$
  Output: Indexing files $\{g_{il}\}$
1 begin
2     repeat
3         Dividing $\{\mathbf{p}_i\}$ into $nc$ clusters $\{c_i\}$;
4     until convergence or exceeding max iter.;
5     for $\forall \mathbf{p} \in c_i$ do
6         Utilizing LSH to construct index in each cluster
          (computing node);
          // $\{g_{il}\}$ means $c_i$'s index
7         Writing index to $\{g_{il}\}$;
8     end
9 end
```

In particular, for l_2 norm (Euclidean distance metric) which is used in this paper, elements of $\mathbf{a}$ are randomized from a Gaussian distribution because Gaussian distribution is a 2-stable distribution and its PDF is $\frac{1}{\sqrt{2\pi}}e^{\frac{-x^2}{2}}$.

The built index consists of hashing keys and feature point identities. Before searching, the feature points and the index are loaded to memory locally.

3.2 Nearest Neighbor Searching

In the search phase, we first find which cluster the query point $\mathbf{q}$ belongs to. In the elected cluster(s), we search colliding buckets $g_1(\mathbf{q}), \cdots, g_L(\mathbf{q})$ and get all points $\mathbf{p}_1, \cdots, \mathbf{p}_n$ in these buckets for further detailed distance evaluation. For each $\mathbf{p}_i (i = 0, \cdots, n)$, if $D(\mathbf{q}, \mathbf{p}_i) \leq r$, we treat $\mathbf{p}_i$ as the final ANN to the query $\mathbf{q}$.

Nevertheless, when the query locates near the cluster border, it is insufficient to search only one cluster. Conversely, if we engage LSH in searching the feature points in all clusters, it is very time-consuming to merge the results from different computing nodes. Generally speaking, the nearest several clusters often contain almost all ANN results. Consequently, considering the search efficiency, we search in a few clusters' LSH buckets instead of all. We provide two methods to choose some clusters for further search.

1. Search fixed number s clusters (such as, $s = 2$ clusters) for simplicity;
2. Search the clusters where the distances between their centers and the query satisfy $\frac{d_i}{\min\{d_i\}} \leq T (i = 1, \cdots, k)$ (T is a customizable threshold).

Instinctively, the nearest cluster may contain almost all similar results according to clustering properties and in LSH scheme, the smaller $D(\mathbf{p}_i, \mathbf{p}_j)$, the larger colliding probability. Therefore, search a few clusters may meet the search quality requirements. The dissimilar feature points which collided in LSH scheme are filtered by clustering. As a result, the detailed distance evaluation times are reduced a lot in the refine phase (i.e. less search time in practice).

We should mention that the clustering approach and indexing technique are not limited to k-means and p-stable distribution based LSH. Any other clustering algorithms and hashing methods, such as AP [4], AKM [15], Query-adaptive LSH [6] and KLSH [9], can be adopted in our framework.

4. EXPERIMENTS

4.1 Experiment Settings

We validate the proposed approach on INRIA's BIGANN dataset[1] (10 thousand and 1 million 128-dimensional SIFT[10] feature datasets, 1 million 960-dimensional GIST[13] feature dataset), which aims to evaluate the search quality of ANN search algorithm. Our experiments are organized on seven PCs (one master and six slaves as computing nodes connected by a gigabit LAN) with four 64-bit 2.00GHz CPUs and 8GB RAM each. The operating system is Linux with kernel 2.6.32. E2LSH (Exact Euclidean LSH)[2] which is a popular implementation of p-stable distribution based LSH, is rewritten for our distributed settings in C++. The number of cluster nc is set to 6 (one slave handles one cluster) and the parameters k and L in LSH scheme are determined by E2LSH, while the successful probability δ is set to 0.9 and w is 4 throughout our experiments.

As we all know, precision and recall are two key criteria for search quality. Since our method is a kind of filter-and-refine framework, the distances between returned results $\mathbf{p}_i$ and query $\mathbf{q}$ are all less than r (i.e. $D(\mathbf{q}, \mathbf{p}_i) \leq r$). Consequently, we only put our focus on the recall in our experiments. Besides, we will compare the average number of the detailed distance evaluation over 100 queries and present the average search time. As LSH functions are randomly selected, we repeat our experiment 10 times and the average results are recorded.

4.2 Experimental Results and Discussions

Table 1 shows the recall of ANN search. The presented results only comprise the $s \leq 3$ and $T \leq 1.3$ results. When $s > 3$ or $T > 1.3$, the result is almost the same as $s = 3$ or $T = 1.3$ (also in Table 2). Table 2 represents the detailed distance evaluation times in the refine phase. Experimental results demonstrate that CLSH can get comparable recall to E2LSH, but the detailed distance evaluation times decrease a lot. Due to the randomly selected LSH functions, the CLSH's recall is slightly larger than E2LSH's when $s = 3$ and $T = 1.3$ on SIFT10K dataset. This phenomenon also occurs in Table 2 when $s = 3$ and $T = 1.3$ on SIFT1M dataset.

Table 1: Comparison on Recall

Dataset		SIFT10K	SIFT1M	GIST1M
E2LSH		0.9647	0.9494	0.9680
CLSH	$s = 1$	0.8704	0.8926	0.7732
	$s = 2$	0.9667	0.9494	0.9514
	$s = 3$	0.9741	0.9494	0.9647
	$T = 1.1$	0.9518	0.9319	0.8953
	$T = 1.2$	0.9741	0.9494	0.9640
	$T = 1.3$	0.9741	0.9494	0.9647

Table 2: Comparison on the detailed distance evaluation times

Dataset		SIFT10K	SIFT1M	GIST1M
E2LSH		142.6	13,435.3	121,871
CLSH	$s = 1$	95.03	9,854.27	53,021.7
	$s = 2$	124.64	13,318.5	91,421.2
	$s = 3$	134.5	14,639.4	106,805
	$T = 1.1$	108.17	11,078.8	75,891.2
	$T = 1.2$	119.32	12,753.2	93,990
	$T = 1.3$	128.46	13,467	107,738

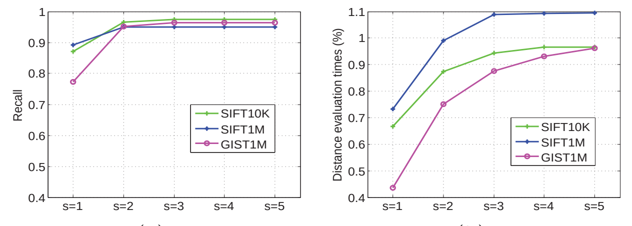
The influence of s over recall and the detailed distance

[1]http://corpus-texmex.irisa.fr/

[2]http://www.mit.edu/~andoni/LSH/

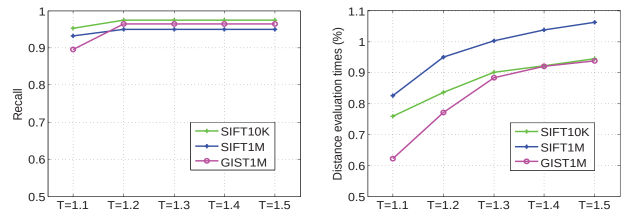
Table 3: Comparison on total search time (s)

Dataset	E2LSH	CLSH						Max$\{T_{c_i}\}$
		$s=1$	$s=2$	$s=3$	$T=1.1$	$T=1.2$	$T=1.3$	
SIFT10K	0.00031	0.00021	0.00022	0.00024	0.00022	0.00024	0.00025	0.00022
SIFT1M	0.01531	0.00813	0.00907	0.00994	0.00915	0.00983	0.01011	0.00813
GIST1M	0.59721	0.25116	0.25832	0.26014	0.25883	0.26001	0.26797	0.25271



(a) (b)

Figure 2: Recall and the detailed distance evaluation times vs. s. Figure 2(a): Influence of s on recall. Figure 2(b): Influence of s on the detailed distance evaluation times.



(a) (b)

Figure 3: Recall and the detailed distance evaluation times vs. T. Figure 3(a): Influence of T on recall. Figure 3(b): Influence of T on the detailed distance evaluation times.

evaluation times (normalized by E2LSH's, same in Figure 3) is demonstrated in Figure 2. Figure 3 illustrates the influence of T.

Table 3 shows the average total search time in our distributed settings. Total search time consists of search time T_s, network transmission time T_t, and results merging time T_m. The last column T_{c_i} is maximum total search time of searching each cluster. We can infer from Table 3 that although T_s remains almost the same because of parallel searching different computing nodes, T_t and T_m are not negligible to total search time. When CLSH gets access to several clusters, total search time is much larger than T_{c_i} especially in SIFT1M and GIST1M as the larger dataset and higher dimensionality. Therefore, searching within a few clusters could be much more effective, and the total search time is significantly reduced eventually as the Table 3 indicates.

5. CONCLUSIONS

In this paper, we present a distributed scalable framework, cluster-based locality-sensitive hashing, for large scale high-dimensional datasets indexing and searching. The clusters are treated as a guideline to automatically deliver the feature dataset to different computing nodes. Besides, in the search phase, two criteria are applied to choose pick several clusters for further retrieval instead of all. Experimental results indicate that CLSH reduces the detailed distance evaluation times a lot while preserving the search quality, and the query search time is significantly reduced. The proposed framework is more efficient, scalable and practical, and the clustering approaches and indexing techniques can be extended to other methods besides k-means and LSH (as aforementioned in Section 3). Future work includes investigating data-adaptive hashing functions in each cluster and extending the proposed approach to further applications, such as image retrieval and object recognition.

Acknowledgements. This work is supported by the National Science Foundation of China under Grant No.61321491 and No.61202320, Research Project of Excellent State Key Laboratory (No.61223003), Natural Science Foundation of Jiangsu Province (No.BK2012304) and National Special Fund (No.2011ZX05035-004-004HZ).

6. REFERENCES

[1] C. Böhm, S. Berchtold, and D. A. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Comput. Surv.*, 33(3):322–373, Sept. 2001.

[2] X. Chen, Y. Mu, S. Yan, and T.-S. Chua. Efficient large-scale image annotation by probabilistic collaborative multi-label propagation. In *MM*, pages 35–44. ACM, 2010.

[3] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SoCG*, pages 253–262. ACM, 2004.

[4] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, 2007.

[5] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC*, pages 604–613. ACM, 1998.

[6] H. Jégou, L. Amsaleg, C. Schmid, and P. Gros. Query adaptative locality sensitive hashing. In *ICASSP*, pages 825–828. IEEE, 2008.

[7] Z. Ji, P. Jing, Y. Su, and Y. Pang. Rank canonical correlation analysis and its application in visual search reranking. *Signal Processing*, 93(8):2352–2360, 2013.

[8] A. Joly and O. Buisson. Random maximum margin hashing. In *CVPR*, pages 873–880. IEEE, 2011.

[9] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing. *TPAMI*, 34(6):1092–1104, 2012.

[10] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[11] 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. VLDB Endowment, 2007.

[12] Y. Mu, J. Shen, and S. Yan. Weakly-supervised hashing in kernel space. In *CVPR*, pages 3344–3351. IEEE, 2010.

[13] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 42(3):145–175, 2001.

[14] Y. Pang, Z. Ji, P. Jing, and X. Li. Ranking graph embedding for learning to rerank. *TNNLS*, 24(8):1292–1303, Aug 2013.

[15] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, pages 1–8. IEEE, 2007.

[16] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for scalable image retrieval. In *CVPR*, pages 3424–3431. IEEE, 2010.

[17] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, pages 1753–1760. MIT Press, 2008.