Intelligent Probing for Locality Sensitive Hashing: Multi-Probe LSH and Beyond

Qin Lv
University of Colorado Boulder
qin.lv@colorado.edu

William Josephson
Solano Labs
wkj@morphisms.net

Zhe Wang
Datrium
zhewang@gmail.com

Moses Charikar
Stanford University
moses@cs.stanford.edu

Kai Li
Princeton University
li@cs.princeton.edu

ABSTRACT

The past decade has been marked by the (continued) explosion of diverse data content and the fast development of intelligent data analytics techniques. One problem we identified in the mid-2000s was similarity search of feature-rich data. The challenge here was achieving both high accuracy and high efficiency in high-dimensional spaces. Locality sensitive hashing (LSH), which uses certain random space partitions and hash table lookups to find approximate nearest neighbors, was a promising approach with theoretical guarantees. But LSH alone was insufficient since a large number of hash tables were required to achieve good search quality. Building on an idea of Panigrahy, our multi-probe LSH method introduced the idea of intelligent probing. Given a query object, we strategically probe its neighboring hash buckets (in a query-dependent fashion) by calculating the statistical probabilities of similar objects falling into each bucket. Such intelligent probing can significantly reduce the number of hash tables while achieving high quality. In this paper, we revisit the problem motivation, the challenges, the key design considerations of multi-probe LSH, as well as discuss recent developments in this space and some questions for further research.

1. LOOKING BACK

In the late 1990s, digital data explosion was already occurring across a broad range of application domains. In addition to text documents, massive amounts of data with diverse feature-rich content, such as images, audio, video, and sensor data, were becoming widely available. The trend motivated us to study similarity search for feature-rich data. Since the features of a data object are often represented as a high-dimensional feature vector, the basic problem for similarity search is to find the (approximate) nearest neighbors of a feature vector in a high-dimensional space.

Proceedings of the VLDB Endowment, Vol. 10, No. 12

Studies had shown that traditional tree-based indexing mechanisms did not work well for feature vectors with more than tens of dimensions, due to the "curse of dimensionality" and exponential growth of the search space. A few other approaches were proposed for problems with low intrinsic dimensionality. Still, there was no good solution for general similarity search in high dimensions. The challenges lie in the high dimensionality of the feature space (typically more than a hundred dimensions) and massive scale (billions or trillions of data objects).

Introduced by Indyk and Motwani in 1998, locality sensitive hashing (LSH) [8] uses a family of locality sensitive hash functions (i.e., certain random space partitions) to hash similar objects into the same bucket of a hash table. With LSH, nearby objects are more likely to be hashed to the same value than objects that are further away. Typically, multiple LSH functions are concatenated for each hash table to reduce false positives, and multiple hash tables are needed to reduce false negatives. Although this basic LSH scheme had shown some promising results for similarity search in high dimensions, one major drawback was that a large number of hash tables, usually on the order of tens or hundreds, were needed to achieve good search quality. The issue was that similar objects may not always fall into the same hash buckets as the query object does, and many hash tables with different sets of hash functions were needed to find that exact match of all hash values. Hence the number of hash tables needed to be very large, limiting the practicality of LSH in real-world applications.

In 2006, Panigrahy proposed an entropy-based LSH scheme and gave a theoretical analysis [13]. The idea was to randomly perturb the query object q to generate multiple objects in its neighborhood, and use those perturbed objects as extra query objects to identify more hash buckets to check for q's nearest neighbors. Intuitively, since the perturbed objects are close to the query object, the buckets they are hashed into may contain candidate objects that may also be close to the query object q. Using the entropy-based LSH scheme, multiple hash buckets can be checked in each hash table, and can trade time for space, which increases the query time but may increase the search quality (finding more or closer neighbors) without increasing the number of hash tables.

The idea of checking multiple hash buckets per table using randomly-perturbed objects was interesting, and was a key motivation for our own work. However, the random

perturbation process of entropy-based LSH was not very efficient. Some buckets (containing good nearest neighbors) could be checked many times, while other buckets (relatively unlikely to contain nearest neighbors) may also be checked, thus incurring significant overhead. Furthermore, for both theoretical analysis and practical use, the perturbation distance R_p (i.e., the distance between the nearest neighbor p and query object q) is needed but often difficult to obtain. Based on our own implementation and evaluation of the entropy-based LSH scheme, it could reduce the number of hash tables by a factor of 2 or 3, but increase the query time by 30% – 210%, in our experiments.

Motivated by the potential and limitations of basic LSH and entropy-based LSH, we investigated an intelligent probing technique for LSH-based indexing. The focus was on "intelligent probing", and we developed two important ideas during our investigation.

- The first was direct and deterministic probing of the hash buckets, as opposed to the random perturbation of objects in entropy-based LSH. As such, all the hash buckets we check are unique, and we can control the specific order of buckets to check as needed. Furthermore, the buckets we check should be "neighboring" buckets whose hash values are close to that of the query object's bucket.
- The second was query-dependent probing. Since we have the raw feature vector of the query object, we know not only the hash values and the bucket it is hashed into, but also its exact position in the hash space (e.g., distances to the left and right boundaries of the hash slot). This allows us to determine the likelihood of the nearest neighbors being hashed to the left or right side. This asymmetry between the information available about the query and the indexed data is also implicit in Panigrahy's algorithm and analysis.

Our investigation led us to the design of multi-probe LSH [11], an efficient similarity search method that strategically probes a query objects neighboring hash buckets by calculating the statistical probabilities of similar objects falling into each bucket. By intelligently probing multiple buckets in each hash table, based on their likelihood of containing the nearest neighbors, multi-probe LSH can reduce the number of hash tables by an order of magnitude while achieving high search quality.

2. PROGRESS SINCE THEN

Adaptations

Since its original publication at VLDB 2007, multi-probe LSH has been deployed in various systems to achieve a good balance between search speed and good results. In [7], multi-probe LSH was extended in the peer-to-peer Chord-style overlay network to search for nearest neighbors. Bahmani *et al.* [2] built a distributed search system using multi-probe as the first layer of hashing, to achieve decreased network cost while maintaining load balance between machines. Kalantidis *et al.* [10] built multi-probe LSH index to speed up search for clothing in an automatic product suggestion system. Rublee *et al.* [15] deployed a real-world smartphone application for object detection and patch-tracking on top of multi-probe LSH index. Yu *et al.* [18] combined multi-probe LSH with order statistics based LSH to build a scalable audio content retrieval system. Fusco *et al.* [5] built a system for on-the-fly compression, archiving and indexing of streaming network traffic. Its on-the-fly LSH index was inspired by multi-probe LSH. In an effort to index and search 100 million images, Moise *et al.* [12] deployed a cluster with the help of multi-probe LSH to make a scalable system. On the theory side, Andoni *et al.* [1] showed that a practical LSH family for angular distance called cross-polytope LSH achieves optimal parameters and also developed a multi-probe version of this hash family. This is the basis for the FALCONN similarity search library [14].

Data dependent improvement

Joly and Buisson [9] extended our multi-probe LSH method by estimating the success probabilities of buckets in a data dependent fashion. Additionally, they allow perturbations of magnitude more than 1 in each coordinate of the hash table address. Their method uses a prior on the data distribution estimated from a training set of queries (sampled at random from the data set) and their explicitly computed nearest neighbors. Exploiting this knowledge of the data distribution allows better probability estimation for selecting more probable buckets and also to decide the number of hash tables to achieve desired accuracy. They called this method a posteriori multi-probe LSH.

Choosing parameters

Although LSH is a promising approach, in practice it is not easy to use partly because its search quality is sensitive to several parameters that are quite data dependent. Dong *et al.* [4] looked into this problem to provide guidance on how these parameters should be chosen, and how to tune parameters for a given dataset. It presents a statistical performance model of multi-probe LSH which can accurately predict the average search quality and latency given a small sample dataset. It solves the problem where we can automatically tune the parameter with different dataset and growing dataset. Also it uses the model to devise an adaptive LSH search algorithm to determine the probing parameter dynamically for each query. Such adaptive probing method addresses the problem where the variance of search performance can be extremely high even when the average performance is tuned for optimal.

Stanley *et al.* [16] continued to work on this problem, and suggested that the distance profile is a sufficient statistic to optimize LSH. The algorithm only need to know two probability distribution functions: the distances between the query point and 1) its nearest neighbor or 2) any random member of the data set. Given a desired performance level (chance of finding the true nearest neighbor) and data set size, the algorithm returns the LSH parameters that allow an LSH index to meet the performance goal and have the minimum computational cost.

3. LIMITATIONS AND OPEN QUESTIONS

Choosing parameters - theoretical analysis

While considerable effort has been expended on analyzing locality sensitive hashing schemes, much of the effort has been focused on analyzing classic LSH. What this amounts to is analyzing the success probability of one probe into the

hash table constructed by LSH. In order to analyze multi-probe LSH, we need to analyze the probability of success from probing multiple hash buckets of the hash table.

As mentioned before, our work was motivated by the algorithm and analysis of Panigrahy [13]. His algorithm randomly perturbs the query point and queries the bucket that the perturbed query falls into, repeating this process multiple times. In order to analyze this scheme, we need to bound the number of queries needed to find a target point (for a given failure probability). Panigrahy gave an elegant entropy argument to bound the number of such queries needed. The key quantity of interest is the entropy of the hash value of a randomly perturbed point p, given the query q and hash function h. If the entropy is high, a lot of probing is needed to find the target point and vice versa.

Our multi-probe LSH scheme directly probes hash buckets by computing/estimating the probabilities that the target falls into them and picking the buckets in decreasing order of these estimates. If we computed these probabilities exactly, for any fixed failure probability, the number of buckets probed by our scheme is upper bounded by the number of buckets probed by Panigrahy's algorithm. Consequently, his entropy based analysis gives an upper bound on the number of queries needed by multi-probe LSH. But this bound could be overly pessimistic.

An interesting open question here is to give a tight bound on the query complexity of multi-probe LSH for various locality sensitive hash families. A closely related question is the interaction of the number of hash tables with the success probability of the scheme: If one has additional space available, is it better to double the number of hash tables and probe only half as many hash buckets in each table? The answer is not obvious a priori: additional probing in an existing table has the advantage that it is guaranteed to return data items that were not seen before, while probing a new hash table could generate duplicates of items already seen. Both these questions we raise can be answered by a tighter understanding of how the success probability of multi-probe LSH (in a single hash table) increases as a function of the number of buckets probed. This would be very valuable for choosing parameters in using multi-probe LSH in practice, but is also an interesting theoretical question that has not been investigated.

Distance function, learning to hash

Depending on the specific distance function used for the feature vectors, different families of LSH functions may be used for similarity search. Our original multi-probe LSH paper focused on Euclidean distance and used the Gaussian distribution (2-stable) as the p-stable distribution. The multi-probe concept can also be applied to other kinds of LSH works with different distance measures. For example, Andoni *et al.* [1] investigated the LSH family for the angular distance, evaluated the multi-probe version of the algorithm, and showed 10× faster performance than hyperplane LSH. Gorisse *et al.* [6] presented a new LSH scheme adapted to Chi-squared distance, and its extension to multi-probe LSH which reduced the memory usage while preserving accuracy.

Classic LSH algorithms are data independent since they use random space partitions to identify nearby objects in high-dimensional spaces. More recently, researchers have investigated data-dependent hashing or learning to hash techniques [1, 17]. The general idea is to construct dynamic hash functions based on the specific distributions/characteristics of the dataset. Many different methods have been developed and good results have been reported. This line of research is somewhat orthogonal to our multi-probe LSH work, and interested readers are encouraged to read the survey paper above and other related papers. We note here that the multi-probe concept may still be applied in such settings, as demonstrated in [1].

Large candidate set size

One limitation of the LSH method is its relatively large candidate set. This is because each probed bucket contains many other data items that are not the k-nearest neighbors. In general, the higher the accuracy we would like to achieve, the larger the candidate set size. In [4], Dong *et al.* pointed out that up to a certain point, probing a large number of buckets only increases its candidate set size without improving its accuracy. To achieve accuracy of about 95%, the candidate set size can be as large as 5-10% of the entire dataset. Since the LSH approach requires linearly scanning the entire candidate set, a large candidate set can be time consuming.

To overcome this limitation and the limitation that LSH works only with certain similarity measures or distance functions, another approach to search for k-nearest neighbors in the high-dimensional space is to construct a k-nearest neighbor graph that allows generic similarity measures. An example is the nearest-neighbor decent (NN-decent) algorithm [3], an efficient approximation algorithm based on the principle that a neighbor of a neighbor is likely to be a neighbor. This approach can efficiently construct an approximate k-nearest neighbor graph and can substantially reduce the number of data items to be examined, comparing with the LSH approach. However, it has the restriction that one can only query the nearest neighbors of data items in the graph.

4. REFERENCES

[1] A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, and L. Schmidt. Practical and optimal lsh for angular distance. In *NIPS '2015: Proceedings of the 28th International Conference on Neural Information Processing Systems*, pages 1225–1233, 2015.

[2] B. Bahmani, A. Goel, and R. Shinde. Efficient distributed locality sensitive hashing. In *CIKM '2012: Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2174–2178. ACM, 2012.

[3] W. Dong, C. Moses, and K. Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *WWW '2011: Proceedings of the 20th international conference on World wide web*, pages 577–586. ACM, 2011.

[4] W. Dong, Z. Wang, W. Josephson, M. Charikar, and K. Li. Modeling LSH for performance tuning. In *CIKM '2008: Proceedings of the 17th ACM conference on Information and knowledge management*, pages 669–678. ACM, 2008.

[5] F. Fusco, M. P. Stoecklin, and M. Vlachos. Net-fli: on-the-fly compression, archiving and indexing of streaming network traffic. *Proceedings of the VLDB Endowment*, 3(1-2):1382–1393, 2010.

[6] D. Gorisse, M. Cord, and F. Precioso. Locality-sensitive hashing for Chi2 distance. *IEEE*

Transactions on pattern analysis and machine intelligence, 34(2):402–409, 2012.

[7] P. Haghani, S. Michel, P. Cudré-Mauroux, and K. Aberer. LSH at large-distributed KNN search in high dimensions. In *WebDB*, 2008.

[8] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC '1998: Proceedings of the 30th ACM Symposium on Theory of Computing*, pages 604–613. ACM, 1998.

[9] A. Joly and O. Buisson. A posteriori multi-probe locality sensitive hashing. In *Proceedings of the 16th ACM international conference on Multimedia*, pages 209–218. ACM, 2008.

[10] Y. Kalantidis, L. Kennedy, and L.-J. Li. Getting the look: clothing recognition and segmentation for automatic product suggestions in everyday photos. In *Proceedings of the 3rd ACM conference on International conference on multimedia retrieval*, pages 105–112. ACM, 2013.

[11] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe LSH: Efficient indexing for high-dimensional similarity search. In *VLDB '2007: Proceedings of the 33rd international conference on Very large data bases*, pages 950–961, Sep. 2007.

[12] D. Moise, D. Shestakov, G. Gudmundsson, and L. Amsaleg. Indexing and searching 100m images with map-reduce. In *Proceedings of the 3rd ACM International conference on multimedia retrieval*, pages 17–24. ACM, 2013.

[13] R. Panigrahy. Entropy based nearest neighbor search in high dimensions. In *SODA '2006: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1186–1195. Society for Industrial and Applied Mathematics, 2006.

[14] I. Razenshteyn and L. Schmidt. FALCONN (FAst Lookups of Cosine and Other Nearest Neighbors) library. `https://falconn-lib.org`.

[15] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *ICCV '2011: Proceedings of the 2011 IEEE international conference on Computer vision*, pages 2564–2571. IEEE, 2011.

[16] M. Slaney, Y. Lifshits, and J. He. Optimal parameters for locality-sensitive hashing. *Proceedings of the IEEE*, 100(9):2604–2623, 2012.

[17] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen. A survey on learning to hash. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9), 2017.

[18] Y. Yu, M. Crucianu, V. Oria, and E. Damiani. Combining multi-probe histogram and order-statistics based lsh for scalable audio content retrieval. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 381–390. ACM, 2010.