View Reviews

Paper ID
186

Paper Title
Rebuilding Locality-Sensitive Hashing Indices by Exploring Density of Hash Values

Track Name
Research

REVIEWER #1

REVIEW QUESTIONS

1. Overall Evaluation
Weak Reject

2. Originality
Medium

3. Importance
Low

4. Summary of the contribution (in a few sentences)
The paper aims at improving the performance of LSH by considering data distribution. Specifically, LSH often generates unbalanced hash buckets, where there is a skew distribution of items. To solve this problem, layeredLSH is proposed to recursively split hash buckets of large size, until the number of item inside each bucket is smaller than a threshold. Such splitting is able to achieve a better accuracy in query processing.

5. List 3 or more strong points, labelled S1, S2, S3, etc.
S1. LayeredLSH achieves improvements in query accuracy.
S2. LayeredLSH is orthogonal and applicable to many hashing algorithms.
S3. The experimental evaluation is comprehensive.

6. List 3 ore more weak points, labelled W1, W2, W3, etc.
W1. Query efficiency in terms of running time, which is not reported in the paper, could be worse for LayeredLSH.
W2. Baseline algorithms used are relatively inefficient, given that there are so many well-developed hashing algorithms to choose for balanced hashing.
W3. Baseline algorithms may not be tuned to their best performance (w.r.t. parameter l and m in the paper).

7. Detailed evaluation, labelled D1, D2, D3 etc.
D1. The paper presents improvement in terms of accuracy (e.g. # of distance measurements), which is not surprisingly due to the more fine-grained and balanced layered hash index. However, the layered hash index also incurs more computation to query processing and thus increases the average overhead of retrieving a single item to evaluate. Taken accuracy improvement and retrieval degradation together, layered LSH may not ultimately achieve better efficiency. I would like to see more statistics in terms of querying time, which is lacking in this paper.

D2. Although there are limited papers related to balanced hashing in the database community, it is a hot and well studied topic in the machine learning community. Some state-of-the-art algorithms from machine learning community should be used as benchmarks, e.g. ITQ (Iterative Quantization), OPQ (Optimized Product Quantizaton) or kNN Graph (k Nearest Neighbor Graph). All of these algorithms exhibit substantial improvements over LSH.

D3. Parameters l and m are very important to the performance of similarity search. However, both l and m are fixed to 3 without reasons. This paper will be more convincing if it discusses why this setting is reasonable or reports extra experimental results with different settings.

REVIEWER #4

REVIEW QUESTIONS

1. Overall Evaluation
Weak Accept

2. Originality
Medium

3. Importance
High

4. Summary of the contribution (in a few sentences)
This work proposes to rebuild LSH indices by exploring the density of hash values to deal with skewed data. The approach can be applied to other LSH variants whose basic index structure is a hash table. The basic idea is that, after building the LSH hash tables, this method detects overloaded buckets and underloaded buckets, and then recursively split the overloaded buckets. It is also shown that this approach performs better than LSH in distributed computing environment. The experiments conducted on real data demonstrates the effectiveness and efficiency of the proposed algorithm. The paper is well written and organized.

5. List 3 or more strong points, labelled S1, S2, S3, etc.
S1. The paper studies an important problem. How to deal with skewed data is important in many applications.
S2. The proposed method is simple yet effective. The paper is well written and easy to follow.
S3. The experimental study demonstrates the effectiveness and efficiency of the proposed method.

6. List 3 ore more weak points, labelled W1, W2, W3, etc.
W1. It would be better to provide some complexity analysis for the proposed method to make the work more rigorous.
W2. My major concern is that this method is not suitable for dynamic data, such as data streams or data with frequent deletion and insertion.
W3. Some parts of the experiments are not clearly described.

7. Detailed evaluation, labelled D1, D2, D3 etc.
D1. Building LSH hash tables is independent of the data, and we just need to define the hash functions and groups. Since this method rebuilds LSH indices by exploring the density of hash values, it means that the data decides the LayerLSH structure, which is different from DSH as well.

As a result, if the data is dynamic, the overloaded and underloaded buckets keeps changing, and I am not sure if the approach can be easily modified to deal with this problem. In this case, I think DSH maybe a better choice for processing the skewed dynamic data.

D2. I am a bit confused about how figures 5, 6, and 7 are draw. It is said that by tuning the acuracy requirement, multiple LSH indices are created. Does each point on the line in these figures represent a result pair of one LSH index on a query set? How many indices do you create for each query set?

D3. Some typos, e.g.,
In "Datasets and Queryies", these three datasets -> these six datasets
Title of section VI.B, Overal Performance -> Overall Performance

8. Required changes for a revision, if applicable. Labelled R1, R2, R3, etc. (Please mark the requests clearly.)
R1. Please describe how to deal with the dynamic data.
R2. More explanations on figures 5, 6, and 7.
R3. Fix the typos.

REVIEWER #3

REVIEW QUESTIONS

1. Overall Evaluation
Weak Accept

2. Originality
High

3. Importance
Medium

4. Summary of the contribution (in a few sentences)
In this paper, authors address the performance issue of building LSH indexes on skewed datasets. By exploring the density information of hashed values, they propose a LayerLSH index that reconstructs overloaded/underloaded buckets to improve precision/recall. For each overloaded bucket, all contained data are recursively rehashed to multiple groups of smaller buckets, forming a multi-layered index structure. To ensure proper precision and recall, algorithms are provided to choose parameters for new rehashing LSHs. The proposed idea can also be extended to other LSH variants (such as LSB and DSH), and to distributed environments which can benefits more from balanced bucket sizes. The experiments show that LayerLSH can reach the same search quality as LSH but only has 5%-20% query cost.

5. List 3 or more strong points, labelled S1, S2, S3, etc.
S1. The proposed index maintains a small number of hash tables in level 0 for all data points, and only adds more hash tables for dense areas (i.e., overloaded buckets). In this manner, index structures can be more effectively utilized to answer queries with different densities.

S2. The layered hash tables can be applied to other LSH variants to improve efficiency for skewed data, which are also suitable for distributed processing.

S3. The experiments are conducted on many real datasets, showing that the proposed method is robust and can improve performance in most cases.

6. List 3 ore more weak points, labelled W1, W2, W3, etc.
W1. LayerLSH use a fixed w parameter for hash functions in all levels and dense areas. It might be not efficient when densities differ a lot, as we might need large m and l. (D2)

W2. It is not clearly discussed how indexes are constructed on disk. Is it first built in memory for the entire structure and dump into disk, or directly built on disk? (D3)

W3. The design of primary choices during query processing is not evaluated in experiments. (D6)

7. Detailed evaluation, labelled D1, D2, D3 etc.
D1. In Figure 2(b), I did not clearly observe higher accuracy for sparse queries and low accuracy for dense queries as described. The average error ratios are almost same for all density, but with increasing variance for higher density.

D2. In the proposed method, a pre-defined w is fixed for all hash functions in different levels, which means all buckets are of similar radius and collide probabilities are only controlled by m and l. Since in skewed dataset, the kNN radius can differ a lot, if the radius is not suitable, we might need to have large m and l to reach acceptable precision/recall, which is expensive. It should be more efficient to have smaller w for hash tables in higher levels, so that the radius could be closer to the real kNN distance. In this manner, only small m and l are needed.

D3. During the construction of a hash table layer, we cannot know in advance which buckets are overloaded/underloaded before loading all data points. Hence, we cannot organize similar buckets in continuous file blocks in an online manner (unless data can be entirely kept in memory), and overloaded buckets will be removed entirely in the end. For my understanding, to achieve data locality discussed, data compaction need be conducted to re-organize file blocks after construction. Such process is not mentioned neither in Algorithm 1 nor in Section

III.D.

D4. How to organize 'nearby' buckets (defined in m-dimensions) into continuous file blocks (in 1 dimension) in Section III.D? Is any space filling curve used?

D5. When rehashing an overloaded bucket, as data points are less and clustered, most buckets in sub-tables might be empty, which waste space for empty bucket pointers.

D6. In experiments, there is no comparison for different primary choices (i.e., R, P, none) as discussed in Section III.C. It is interesting to show how these choices affect query performance.

Minor Points:
M1. Legends in Figure 2 can be shortened.
M2. (Page3) "They are hashed to different buckets in two hash tables in Figure 3." I only see one hash table in Level 0.

---

8. Required changes for a revision, if applicable. Labelled R1, R2, R3, etc. (Please mark the requests clearly.)

R1. Need to discuss why w must be fixed in this index. Will using different w improve performance? If it is not easy to change the w, authors should at least mention the challenges.(D2)

R2. Need to evaluate the influence of primary choices. (D6)
