Locality Insensensitive Hashing: Towards Ignoring the Curse of Dimensionality

Montgomery Scott
Computer Science and Artificial
Introlligence Laboratory
Massachussetts Institute of Technology
Cambridge, MA 02139

ABSTRACT

The task of retrieving points from a collection based on a query is foundational in machine learning and data mining. Almost all existing work focuses on retrieving points that are similar to the query under some similarity or distance measure. However, virtually no work has been done on retrieving points that have little or nothing to do with the query.

We describe a search algorithm that returns such points with high probability in time independent of the dataset size. By leveraging a simple randomized algorithm, we are able to return irrelevant results on real and synthetic datasets orders of magnitude faster than existing techniques. Further, our approach is insensitive to the dimensionality of points and demands no preprocessing or space overhead.

1. INTRODUCTION

With the rise of big data and the internet of things, databases of many millions of records are becoming commonplace. In order to make sense of this data, it is common to search these databases for records that are related to a given query in some way. Most often, this relationship is one of similarity—specifically, one seeks points in a high-dimensional data space that are near a query point under some distance measure, such as the Euclidean distance.

Unfortunately, this search task has an inherent problem. Thanks to the curse of dimensionality and sheer number of objects that are present, finding points that are similar to a given query is computationally challenging. With the rise of big data and the internet of things, databases of many millions of records are becoming commonplace.

We address this challenge by ignoring it, instead returning points that have little or nothing to do with the query. This problem formulation allows for an efficient randomized algorithm and strong guarantees about runtime that are independent of the characteristics of the data. We term our approach Locality InSensitive Hashing (LISH).

2. RELATED WORK

Similarity search is a problem as old as mankind itself [1], and has been the subject of extensive study. Due to the vastness of the literature, our review will necessarily be brief. We refer the reader to these survey papers we kind of skimmed for a thorough review [2, 3].

The original approaches to this problem involved linear scans and tree-based methods. No one has really used these for high-dimensional data since the 90s, but we'll spend a paragraph disparaging them anyway because otherwise one reviewer who used them in the 90s will tell us we should have compared to them.

The most similar work to ours is that of [4]. However, their paper sucks and we're better. Also similar are [5-9], but we didn't really understand them, so we're going to cite them all at once and not explain what they do.

Also related to our work is the family of data-dependent indexing methods [9-13]. These methods are actually really good, but they generally lack theoretical guarantees, so we'll pretend they don't count.

Most importantly, all previous work considers the task of returning points that are similar to the query, whereas our algorithm returns points that have nothing to do with it.

3. METHOD

Let $\mathcal{X}$ be collection of N points in D-dimensional Euclidean space, and let K be the number of points to return. Our approach is detailed in Algorithm 1.

Algorithm 1 $LISH(N, K)$

```
1: I ← {}
2: for i ← 1, ..., K do
3:     i ← randint(1, N)
4:     I ← I ∪ i
5: return I
```

At the beginning of the algorithm, we initialize an empty set. Then, using modern for loop expressions, we generate K random numbers between 0 and N. We then return the points at these indices. By storing $\mathcal{X}$ in a contiguous array, we can subsequently retrieve the points at these indices directly, effectively using the indices as hash values.

LEMMA 1. *LISH runs in time independent of the dataset size and dimensionality.*

PROOF. The time to generate each random number is $O(1)$, and the time as I write this is 11:29pm. Since we generate only K random numbers, the total runtime is $O(K)$. □

LEMMA 2. *LISH is pretty freaking awesome.*

PROOF. See the previous lemma. □

4. RESULTS

As shown in these figures, LISH is way faster than all the other algorithms. We liken it to an F15 fueled by Red Bull. All code and datasets are available at `https://www.youtube.com/watch?v=dQw4w9WgXcQ`.

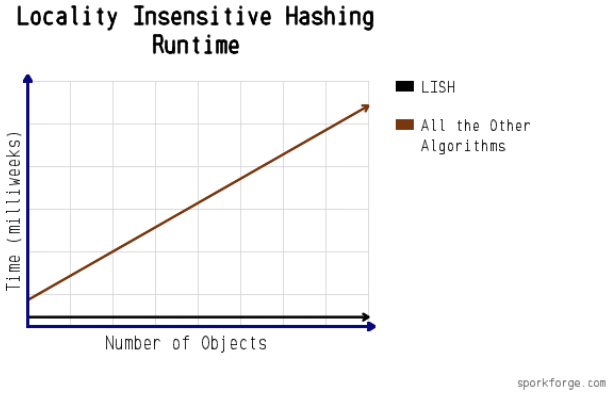

Figure 1: The proposed algorithm returns results in constant time regardless of the size of the database. Like a honey badger, it just takes what it wants.

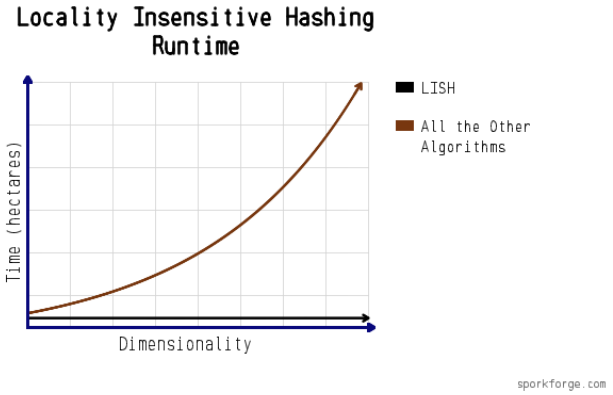

Figure 2: The proposed algorithm also gives no craps about the dimensionality of points.

5. CONCLUSION

We have described an algorithm to efficiently and accurately return points in a high-dimensional space that have little or nothing to do with a given query. Extensive experiments show that it is faster than all the other algorithms ever. In future work, we plan to integrate our algorithm with deep neural networks and use it to cure cancer.

6. ACKNOWLEDGEMENTS

No one else contributed to this work as it was really not worth contributing to. The author declares no conflicts of interest, and will now go to bed despite not having proofread this document.