Improving the accuracy of Locality Sensitive Hashing

Akarsha Sehwag
2015010, CSE, IIIT Delhi

Abstract

Locality sensitive Hashing is used for finding nearest neighbours for high dimensional data. Finding duplicate points/data might seem simple (we can use a hash table and find it in logn time complexity) but finding similar points w.r.t to the given query might be a complicated problem. The currently used algorithm is probabilistic, and uses randomly picked hyperplanes to seperate the data into clusters which might not give us perfect results. So, we tried to remove that randomized hyperplanes using statistical functions.

Keywords

Generic algorithms, Locality Sensitive Hashing, Big Data, high dimensionality, Mann-whitney U test, Data Analysis.

1 Introduction

The basic concept of LSH algorithm is to generate hashcodes, where similar hashcodes indicate near duplicates. So, for that we divide the entire N-Dimensional space into randomly determined bins where similar items lie in the same bucket atleast once. And the more number of times they lie in the same bucket, the more is the similarity in our datapoints. LSH algorithm is majorly used in the following domains: Finding the movie preferences of similar people; Entity resolution; Plag Checker; Database duplicate removal.

The techniques being used in LSH are:

1. **Shingling**: Used to turn string into sets.
2. **Min-hashing**: Used to turn large sets to small signatures.
3. **Locality Sensitive Hashing**: This focuses on pair of signatures lying in the same bucket, and thus are likely to be similar without considering all the points in the data-set.

2 Problem Statement

Now, to find signatures for each particular point in a n-Dimensional space, we need to create ″K″ number of hyperplanes, randomly selected. And for each hyper-plane, assign a key, ′0′ or ′1′ to each point based on the fact that it lies on the ′left′ or ′right′ of the hyper-plane.
After creating keys for all ′k′ hyperplanes we get a binary-looking key for each data-point of length ′k′. All data-points with same keys lie in the same bucket.

The problem it generates is that the randomly selected hyper-planes sometimes cut the clusters from between. And we assume that every data-set naturally tends to form clusters. When you cut a cluster, you see that the points, close to each other– Less hamming Distance – in a cluster have different keys and the points farther away from each other –Larger Hamming Distance – since they belong to different clusters, have the same key. And thus generating ′False Positives′.

So we want to choose ONLY those hyper-planes which do not pass through any of the clusters. For that purpose, we need to define a test to classify our randomly generated hyperplanes into *Good* and *Bad*. And thus, we can generate the binary keys for each point using only Good hyper-planes so that the similarity in the keys is not due to the false-positives.

3 Our approach

Given the problem statement, we use statistical methods and find the best suited algorithm to remove the false positives in LSH algorithm and thus improve it's accuracy and efficiency.

In this Independent Project, I tried some methods to improve the accuracy of the algorithm as listed:

- **Chi-Square test**: This is used to find out whether or not there is a significant difference between the expected frequencies and the observed frequencies in one or more categories given a degree of freedom. Here we find the expected values and the observed values and the summation squared errors. If it exceeds the significance level, we reject the hyperplane. But did not seem that good-a-test for our problem so shifted to Wilkoxyn test.

- **Wilkoxyn test**: Wilcoxon signed-rank test is a nonparametric test that can be used to determine whether two dependent samples were selected from populations having the same distribution. For each hyperplane I chose around 50 points having different keys and 50 points lying on the same side of the hyperplane. Then, I find the Euclidean distance for each pair of points and creating a dictionary containing the key as ′1′ or ′0′ which signifies whether they lie on the same side of the hyperplane or not. And their values are the Eucledian distances between the two points. Now, creating two clusters, one having keys as ′1′ and other having keys as ′0′. Now check how distinct clusters they form. The problem here was that it assumed the two samples to be dependent. So, we switched to Mann-Whitney U test.

- **Mann Whitney-U test**: This is a nonparametric test used to determine whether two independent clusters were selected from populations having the same distribution. Also, it does not require the assumption of normal distribution.

  Here, we check whether our ′near′ points (points on the same side of hyperplane) and the ′far′ points have a significant difference between their respective means. If yes, the hyperplane should be classified as a good hyperplane.

  So choose such best ′K′ hyperplanes and match the results with the same number of randomly generated hyperplanes.

4 Results and possible reasons

I tried for various ′k′(number of nearest neighbours) and ′p′ (hyperplanes) ranging from 20-50 for ′k′ and 10-40 for ′p′. - For small ′k′, there′s no change in the output.

- For large ’k’ and large ’p’. 50 nearest neighbours were chosen. For LSH, 47 points were correctly classified which converted to 48 with our algorithm. But this happened for just one case.

Reason: We talked about points which are close but have different generated keys which is true only for some query points that lie close to the hyperplane. But since I worked with only 3 query points and 3 sparse datasets, they might not have occurred in those points and thus no change seen in my tests except for one case where I considered large number of nearest neighbours.

PS: I am yet to test on the large dataset which might give me better results. I was not able to do it majorly because of the HPC C++ version problems and due to some time constraints.

5 Conclusion

The modified algorithm for LSH should yeild better efficiency for large dataset size and large number of nearest neighbours being chosen. This can be considered as the future-work for this project.