Theory of Clustering - Course Project

Sushant Agarwal

April 30, 2019

Contents

Abstract

This project explores the notion of a Probabilistic clustering. It presents 2 models for the same, and proposes using Locality Sensitive Hashing (LSH) methods as a tool to achieve it. LSH algorithms perform randomized hashing over a data set endowed with a similarity function S, while giving preference to hashing close by objects to the same bucket.

However, as we shall see, an LSH is not possible for all similarity functions S. It would be useful to find not too restrictive sufficient conditions on the similarity function S such that it yields an LSH.

The report discusses a necessary condition for the similarity function S to yield an LSH. It shows that the above condition is also sufficient for 3 or lesser points to yield an LSH. Unfortunately, we see that this condition is not sufficient for 4 or more points. We discuss how the proposed method, which might be promising for 3 or lesser clusters, does not work for 4 or more clusters, and think of some possible ways to circumvent that.

1 Introduction

1.1 Probabilistic Clustering

In the course, we dealt with deterministic clustering algorithms like k-means, single-linkage etc. which for a given input, give a fixed clustering as an output, regardless of how many times you run the algorithm. (given starting/stopping criteria and other parameters)

This project aims to examine the notion of Probabilistic/Randomized Clustering. This seems to be a relatively unexplored area without too much literature on it.

The notion of Probabilistic/Randomized Clustering might be useful when there is no real 'correct' clustering, and we are kind of conflicted as to which cluster a particular point might lie in. A deterministic clustering would assign the point to a specific clustering, but perhaps that doesn't capture the situation fully. Perhaps there is a reasonable chance that this point belonged to some other cluster. This uncertainty could perhaps be captured in a probabilistic framework. In addition, most commonly used clustering objective functions are NP-hard to optimise, which could be avoided if we use a probabilistic algorithm.

1.2 Possible frameworks

Two frameworks of probabilistic clustering seem quite natural, and are stated below.

Probability Vector: Given input dataset X, number of clusters k, we want to output a vector $p^i = (p^i_1, p^i_2 ... p^i_k)$ of length k for every point $i \in X$, where p^i_j is the probability of point i being in cluster j. Also, we have that for every i, $\sum_{j=1}^{k} p^i_j = 1$.

Clearly, this framework also captures deterministic clustering because for every point i, we could set $p^i_j = 1$ for some j, and $p^i_l = 0$ for every $l \neq j$. If the user is keen to obtain a deterministic clustering, the algorithm could be made to output a particular clustering. 2 methods to do so are as follows:

i) For every point $i \in X$, choose which cluster it falls in my picking the $argmax$ of $p^i = (p^i_1, p^i_2 ... p^i_k)$.

ii) For every point $i \in X$, choose which cluster it falls in by tossing a k-sided coin with appropriate biases. (side i has bias p_i)

Distribution over Clusterings: Given input dataset X, number of clusters k not specified, we want to output a set C of different clustering assignments on X, where $C = \{C_1, C_2 ... C_m\}$, and a probability distribution D over the set C. Let us denote the probability the algorithm assigns to clustering C_i by P_i. We have that $\sum_{j=1}^{m} p_j = 1$.

Clearly, this framework also captures deterministic clustering because the set C could be a singleton. If the user is keen to obtain a deterministic clustering, the algorithm could be made to output a particular clustering. 2 methods to do so are as follows:

i) Choose the argmax j of $P = (P_1, P_2 ... P_m)$. Output C_j.

ii) Toss an m sided coin with appropriate biases. (side i has bias P_i)

2 Locality Sensitive Hashing

U is a collection of objects. According to [3], a locality sensitive hashing (LSH) scheme is a distribution D on a family F of hash functions from $U \to R$, such that for every two objects $x, y \in U$:

$$\mathbb{P}_{h \in F}[h(x) = h(y)] = S(x, y)$$

where $S : U * U \to [0, 1]$ is some similarity function defined on the collection of objects.

We call a similarity function S LSH-able if there exists an LSH scheme for it.

LSH algorithms have found uses in many applications, for eg. in near-duplicate detection and nearest-neighbor search.

2.1 LSH for Clustering

LSH algorithms can be viewed as tools for randomized clustering. An LSH scheme can be used to capture the 'Distribution over Clusterings' framework from the above section. A natural goal for probabilistic clustering may be to relate the probability of two points in the domain set X ending up in the same cluster with their input similarity value. Namely, given a similarity function $S : X * X \rightarrow [0, 1]$ on the input set, come up with a probabilistic clustering algorithm that outputs a distribution D on a family F of hash functions from $X \rightarrow R$ such that for all $x, y \in X, \mathbb{P}_{h \sim D}[h(x) = h(y)] = S(x, y)$.

2.2 Example of an LSH scheme

Let us consider a collection of objects U = the set of n-bit vectors. Given $x, y \in U$ we define the similarity function S as follows $S(x, y) = \#\{i : x_i = y_i\}/n$, where x_i is the i'th coordinate of the vector x. For example, disjoint vectors have similarity 0 and $S(x, x) = 1$. If $x = 01001, y = 10011, S(x, y) = 2/5$.

We now construct an LSH scheme for the above S. $F = \{h_1, ..., h_n\}$ is the family of hash functions, where $h_i(x) = x_i$ (The i-th hash function outputs the i-th bit of x). We consider the uniform distribution D over F. It is easy to see that for every $x, y \in U$:

$$\mathbb{P}_{h \in F}[h(x) = h(y)] = \mathbb{P}_i[h_i(x) = h_i(y)] = S(x, y)$$

2.3 Non LSH-able similarities

Below is an example of a similarity function S on 3 points which is not LSH-able. Consider the points x, y, z. The similarity function S is as follows: $S(x, y) = 1, S(y, z) = 1, S(x, z) = 0$.

S is clearly not LSH-able because what we are demanding is that x and y always be hashed to the same value, and y and z always be hashed to the same value. This implies that x and z will always be hashed to the same value. However, the similarity function requires that x and z never be hashed to the same value, which is a contradiction.

Note that even a not so extreme similarity function S as follows in not LSH-able. $S(x, y) = 0.9, S(y, z) = 0.9, S(x, z) = 0.1$. We will prove this later. For now, we just observe the distance function defined as $d = 1 - S$ does not satisfy the triangle inequality.

2.4 Hardness of the LSH feasibility problem

LSH feasibility problem: Given a similarity function S, is there a distribution over hash functions such that it forms an LSH scheme for S? The following result from [1] establishes the intractability of this problem.

Theorem 1. *The LSH-feasibility problem is NP-hard even if S only assumes values in* $\{0, 1/6, 1/3, 2/3, 1\}$

The above theorem implies the following corollary.

Corollary 1. *The LSH-feasibility problem is NP-hard for a general S.*

2.5 Goal of project

It would be useful to find sufficiency conditions on the similarity function S such that it is LSH-able. Ideally, we would like the conditions to be checkable in polynomial time. Note that, due to Corollary 1, we cannot find conditions checkable in polynomial time that are both necessary and sufficient.

A trivial sufficiency condition is requiring the similarity function S to be either the all-0 or all-1 function. These similarity functions are trivially LSH-able, but are clearly not useful in the context of clustering. We should take care to find sufficiency conditions that are not too restrictive and ensure that they make sense in the context of clustering.

We observe that this also falls in line with the 'Notions of Clusterability' discussed in class. We are in a way trying to find niceness properties in data, such that the 'nice' data yields a similarity function S which captures the data well and is also LSH-able.

3 Some Observations

Below I state some observations about LSH-ability that I proved independently.

3.1 Necessary condition for LSH-ability

Theorem 2. *If S is LSH-able, then $d = 1 - S$ satisfies the triangle inequality.*

Proof We are given that S is LSH-able. We have a distribution D on a family F of hash functions, such that for every $x, y \in U$: $\mathrm{P}_{h\sim D}[h(x) = h(y)] = S(x, y)$.

Consider 3 points $x, y, z \in U$. Consider an arbitrary function $f \in F$. Let indicator functions $X_1(f), X_2(f), X_3(f)$ be such they are equal to 0 if $f(x) = f(y)$, $f(z) = f(y)$ and $f(x) = f(z)$ respectively and 1 if $f(x) \neq f(y)$, $f(z) \neq f(y)$ and $f(x) \neq f(z)$ respectively. Note that $d(x, y) = \mathbb{E}_{h\sim D}[X_1(h)]$, $d(y, z) = \mathbb{E}_{h\sim D}[X_2(h)]$ and $d(x, z) = \mathbb{E}_{h\sim D}[X_3(h)]$.

Consider an arbitrary $h \in F$. There are four cases. If $X_1(h) = 0$ and $X_2(h) = 0$ then $X_3(h)$ is forced to be 0. If $X_1(h) = 0$ and $X_2(h) = 1$ then $X_3(h)$ is forced to be 1. If $X_1(h) = 1$ and $X_2(h) = 0$ then $X_3(h)$ is forced to be 1. If $X_1(h) = 1$ and $X_2(h) = 1$ then $X_3(h)$ can be either 0 or 1. In each case, $X_1(h) + X_2(h) \geq X_3(h)$. This is true for every $h \in F$. Therefore, $\mathbb{E}_{h\sim D}[X_1(h)] + \mathbb{E}_{h\sim D}[X_2(h)] \geq \mathbb{E}_{h\sim D}[X_3(h)]$. This implies that $d(x, y) + d(y, z) \geq d(x, z)$.

Similarly, we can show that $X_3(h) + X_2(h) \geq X_1(h)$ and $X_1(h) + X_3(h) \geq X_2$ for every $h \in F$. Therefore, $d(x, z) + d(y, z) \geq d(x, y)$ and $d(x, y) + d(x, z) \geq d(y, z)$. This is true for every $x, y, z \in U$. Therefore, d satisfies the triangle inequality.

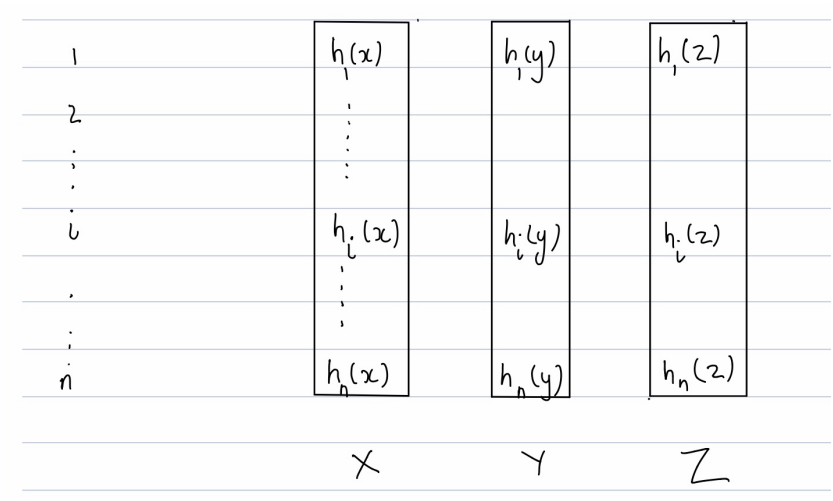
3.2 Necessary and Sufficient condition for 3 points

Below is a sufficient condition on the similarity function S to be LSH-able for a data set X consisting of 3 points. It is reasonable to assume that $d = 1 - S$ always satisfies symmetry.

Theorem 3. *Given $|X| = 3$. If $d = 1 - S$ is satisfies the triangle inequality, S is LSH-able.*

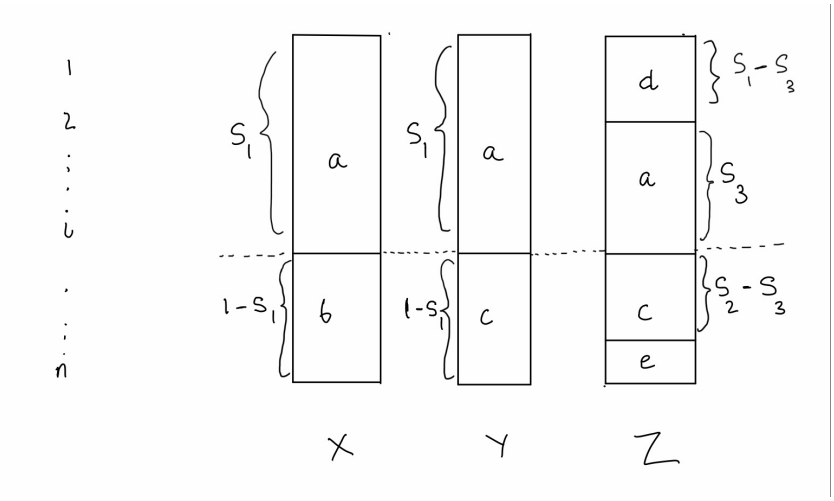
Proof Consider 3 points x, y and z with similarity function S as above. Let us denote $S(x, y)$, $S(y, z)$ and $S(x, z)$ by S_1, S_2 and S_3 respectively. Without loss of generality, we can assume that $S_1 \geq S_2 \geq S_3$. We will now construct an LSH scheme. We need to give a class of functions F and a distribution D over F.

The columns below are the values the functions in F give to x, y and z. Row i depicts function h_i.



We choose D to be the uniform distribution. So for simplicity, can normalise the length of each column to 1. Therefore, columns for x and y must overlap (have the same value on) on exactly S_1 length, columns for y and z must overlap on exactly S_2 length and columns for x and z must overlap on exactly S_3 length.

We can construct F as given in the figure. Each segment is labelled with an alphabet, which represents the value each row in that segment is mapped to. For example, in the figure below, X and Y are mapped to a for the first i rows. The total number of rows is n, and we have that $i/n = S_1$.



Such a construction makes sense only if $S_2 - S_3 \leq 1 - S_1$, which is satisfied because $1 - S$ satisfies the triangle inequality. It is easy to see that in this construction, the properties needed to be an LSH are satisfied.

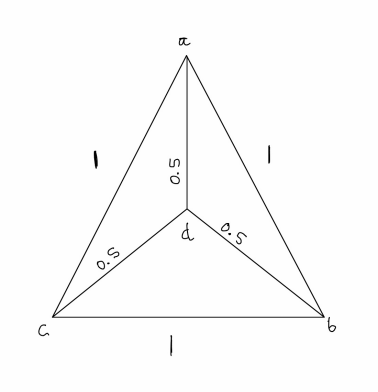
The above theorem, along with Theorem 2 leads to the following corollary.

Corollary 2. *Given* $|X| = 3$. $d = 1 - S$ *satisfies the triangle inequality* $\Leftrightarrow$ S *is LSH-able.*

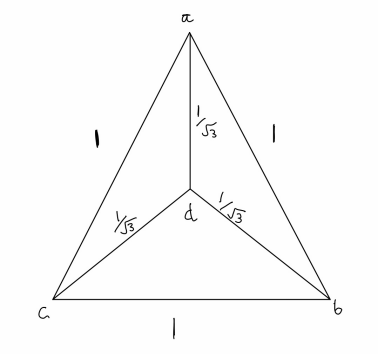
3.3 Necessary condition is not sufficient for 4 points

Below is an example of 4 points with the similarity function S such that $d = 1 - S$ satisfies the triangle inequality. However, S is not LSH-able.

Consider the points a, b, c, d. The sides are labeled with the distances $d = 1 - S$ is as shown in picture below. Note that the triangle inequality is satisfied.



S is not LSH-able because what it is demanding is that a, b and c always be hashed to different values, and d overlap on functions with exactly 0.5 probability mass with each of them. However, with one particular function, d can only agree with at most one out of a, b, c. This means that the total probability mass of the functions would be at least $0.5 * 3 = 1.5$ which is not possible.

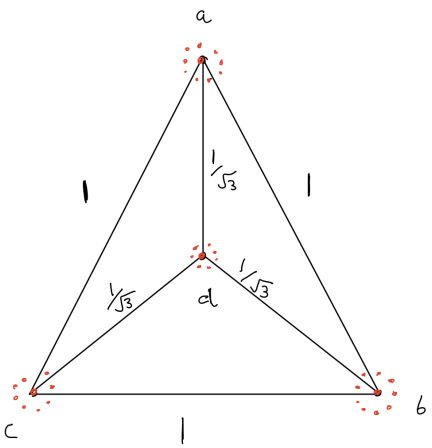


Below is an example of 4 points with the distance function $d = 1 - S$ such that the points are embeddable in Euclidean space. This is a stronger requirement than the triangle inequality. Even so, we can show that S is not LSH-able using the same argument as above.

4 Conclusion and Future Work

I was hoping that the following method for clustering would be promising: Given points in Euclidean space, compute the pairwise distances d and scale them such that the maximum distance between two points is 1. Take the similarity function to be $S = 1 - d$ and try and find a LSH scheme for this S. Use this LSH scheme to assign a clustering to the points.

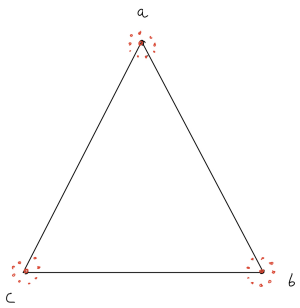
However, we saw in section 3.3 that there exist 4 points a, b, c, d with the distance function $d = 1 - S$ such that the points are embeddable in Euclidean space and that S is not LSH-able. This implies that any clusters somewhat centred around a, b, c, d as below with $S = 1 - d$ are not LSH-able.



However, these clusters are quite 'nice'. They are roughly of the same size and are well separated. We see that such a 'nice' data set is not clusterable(LSH-able) if we choose $S = 1 - d$, and this approach fails for 4 or more clusters.

However, I feel that this approach might still hold some promise for ≤ 3 clusters. We saw that any 3 points embeddable in Euclidean space are LSH-able. I have a feeling that if we have 3 'nice' clusters as below, that are well separated and roughly centred around 3 points a, b, c in Euclidean space, it might always be possible to construct an LSH for $S = 1 - d$ if the clusters are 'nice' enough, for some notion of niceness. I

played with some small toy examples and was always able to come up with an LSH, but could not show that there always exists one.



Because the previous approach of constructing similarity function by taking $S = 1 - d$ does not really work for more than 3 clusters, we could try and alter it. Given some data points, we could try and look at other ways of constructing the similarity function S such that it is LSH-able. Some possibilities include $S = 1/(1 + d)$ and e^{-d}. We could also use some angular similarity measures, as used in [2]. We could try and find sufficiency conditions for similarity matrices to be LSH-able, and investigate how to reach those matrix given some points and pairwise distances in Euclidean space such that the similarity matrix captures the situation meaningfully.

Also, Motwani and others have talked about another notion of LSH in [5], given below.

Let (X, d) be a distance space and let D be a distribution over functions from $X \to R$. We say D is an (r_1, r_2, p_1, p_2)-sensitive LSH if for two objects $x, y \in X$ and $h \sim D$:

- If $d(x, y) \leq r_1$ then $\mathbb{P}_{h \sim D}[h(x) = h(y)] \geq p_1$
- If $d(x, y) \geq r_2$ then $\mathbb{P}_{h \sim D}[h(x) = h(y)] \leq p_2$

This notion is more relaxed and might be interesting to look into. [4] gives an LSH scheme (for this relaxed notion) for points in the Eucildean space under the l_p norm for $p \in (0, 2]$.

The notion of locality preserving hashing, as in [6] might also be interesting to look at.

Acknowledgements

I would like to thank Professor Shai Ben-David and Shrinu Kushagra for useful discussions.

References

[1] Locality Sensitive Hashing-Ravi Kumar.

[2] Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. Practical and optimal lsh for angular distance. In *Advances in Neural Information Processing Systems*, pages 1225–1233, 2015.

[3] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388. ACM, 2002.

[4] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM, 2004.

[5] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

[6] Piotr Indyk, Rajeev Motwani, Prabhakar Raghavan, and Santosh Vempala. Locality-preserving hashing in multidimensional spaces. Citeseer, 1997.