Learning to Find Good Hash Functions for Embeddings

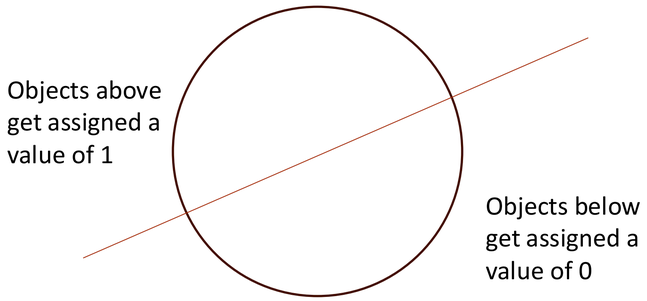
Ben Claydon, Richard Connor, Alan Dearle

University of St Andrews

Introduction

In this paper, we study the approximate near neighbour problem over L2 normalised spaces using binary locality sensitive hash functions.

These functions use a **randomly generated** hyperplane to assign values of 0 or 1 to objects depending on whether they are above or below this plane.



When a query is presented, we retrieve only those points in a dataset which have the **same hash value** (i.e. lie on the **same side** of the hyperplane) as the query.

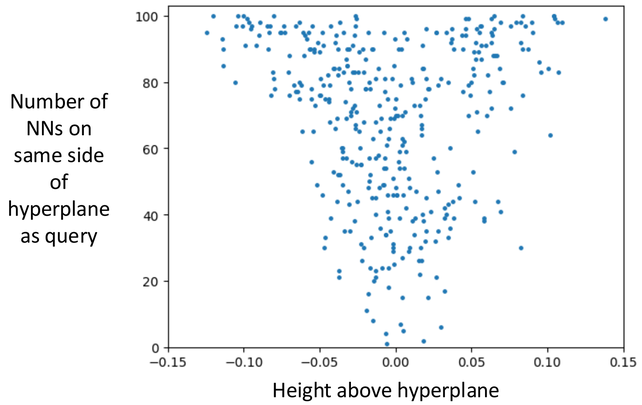
University of St Andrews

Problem

The **best-case scenario** for such a function is when the query is far from the plane. This is because is highly likely that its neighbours are also far from the plane and on the same side as the query.

Conversely, the **worst-case** is the instance where a query sits **directly on** the hyperplane. In this case, we expect a **50%** chance of each individual near neighbour hashing to the same value as the query.

However, we have observed that binary hash functions do not perform this way when applied to real datasets. The figure below shows measurements over the MirFlickr image dataset comparing the number of near neighbours retrieved by many random hyperplane hash functions (represented by each blue dot) vs their height above the hyperplane:



We show that:

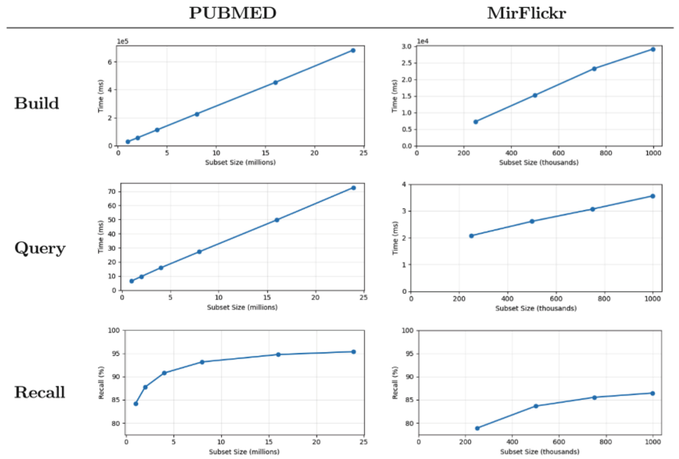
1. There exist many hash functions which retrieve **almost all** the near neighbours of a given query.
2. There also exist many hash functions which **assign the query to a different hash value to nearly all its near neighbours.**
3. We also observe that **query-hyperplane height and hash function performance is not strongly correlated.**

The second property is particularly troubling. Often, many independent hash functions are concatenated; only those items in the dataset whose **output matches the query for all individual hash functions** are retrieved**.** If **any** individual function retrieves a low number of near neighbours, the compound hash function it belongs to will similarly retrieve a low number.

Solution

In response to these issues, we demonstrate that:

- These instances of pathological hash functions are **easily detectable** via means of a **simple neural network**.
- By selecting only those functions which are expected to perform well, we can **improve** the performance of binary compound hash functions.
- We use this idea to build a mechanism. We show that it possesses fast linear build and query times, and **recall improves as data size increases.**



Contact Information

bc89@st-Andrews.ac.uk | Ben Claydon
rchc@st-Andrews.ac.uk | Richard Connor
al@st-Andrews.ac.uk | Alan Dearle