Similarity Search:
Algorithms for Sets and other High Dimensional Data

Thomas Dybdahl Ahle

Advisor: Rasmus Pagh
Submitted: March 2019

IT UNIVERSITY OF COPENHAGEN

Abstract

We study five fundemental problems in the field of high dimensional algorithms, similarity search and machine learning. We obtain many new results, including:

- Data structures with Las Vegas guarantees for ℓ_1-Approximate Near Neighbours and Braun Blanquet Approximate Set Similarity Search with performance matching state of the art Monte Carlo algorithms up to factors $n^{o(1)}$.
- "Supermajorities:" The first space/time tradeoff for Approximate Set Similarity Search, along with a number of lower bounds suggesting that the algorithms are optimal among all Locality Sensitive Hashing (LSH) data structures.
- The first lower bounds on approximative data structures based on the Orthogonal Vectors Conjecture (OVC).
- Output Sensitive LSH data structures with query time near $t + n^\rho$ for outputting t distinct elements, where n^ρ is the state of the art time to output a single element and tn^ρ was the previous best for t items.
- A Johnson Lindenstrauss embedding with fast multiplication on tensors (a Tensor Sketch), with high probability guarantees for subspace- and near-neighbour-embeddings.

Resumé

Vi kommer ind på fem fundementale problemer inden for højdimensionelle algoritmer, similarity search og machine learning. Vi opdager mange nye resultater, blandt andet:

- Datastrukturer med Las Vegas garantier for ℓ_1-Approksimativ Nær Nabo og Braun Blanquet Approksimativ Mængde-similaritetssøgning der matcher state of the art Monte Carlo algoritmer op til $n^{o(1)}$ faktorer i køretid og plads.
- "Kvalificerede flertal:" De første tid-/plads-trade-offs for Approksimativ Mængde-similaritetssøgning samt et antal nedre grænser der indikerer at algoritmerne er optimale inden for Locality Sensitive Hashing (LSH) datastrukturer.
- De første nedre grænser for approksimative datastrukturer baseret på Orthogonal Vectors Conjecture (OVC).
- En Output Sensitive LSH datastrukturer med forespørgselstid næsten $t + n^\rho$ når den skal returnere t forskellige punkter. Her er n^ρ den tid det tager state of the art algoritmen at returnere et enkelt element og tn^ρ tiden det før tog at returnere t punkter.
- En Johnson Lindenstrauss embedding der tillader hurtig multiplikation på tensorer (en Tensor Sketch). I modsætning til tidligere resultater er den med høj sandsynlighed en subspapce embedding.

Acknowledgements

Starting this thesis four years ago, I thought it would be all about coming up with cool new mathematics. It mostly was, but lately I have I started to suspect that there may be more to it.

I want to start by thanking my superb Ph.D. advisor Rasmus Pagh. I always knew I wanted to learn from his conscientiousness and work-life balance. Working with him I have enjoyed his open office door, always ready to listen to ideas, be they good, bad or half baked. Even once he moved to California, he was still easy to reach and quick to give feedback on manuscripts or job search. I may not have been the easiest of students – always testing the limits – but when I decided to shorten my studies to go co-found a startup, Rasmus was still supportive all the way through.

I would like to thank all my co-authors and collaborators, past, present and future. The papers I have written with you have always been more fun, and better written, than the once I have done on my own. Academia doesn't always feel like a team sport, but perhaps it should.

The entire Copenhagen algorithms society has been an important part of my Ph.D. life. Mikkel Thorup inspired us with memorable quotes such as "Only work on things that have 1% chance of success," and "always celebrate breakthroughs immediately, tomorrow they may not be." Thore Husfeldt made sure we never forgot the ethical ramifications of our work, and Troels Lund reminded us that not everyone understands infinities. At ITU I want to thank Johan Sivertsen, Tobias Christiani and Matteo Dusefante for our great office adventures and the other pagan tribes.

In the first year of my Ph.D. I would sometimes stay very long hours at the office and even slept there once. A year into my Ph.D. I bought a house with 9 other people, in one of the best decisions of my life. Throughout my work, the people at Gejst have been a constant source of support, discussions, sanity and introspection. With dinner waiting at home in the evening I finally had a good reason to leave the office.

In 2017 I visited Eric Price and the algorithms group in Austin, Texas. He and everyone there were extremely hospitable and friendly. I want to thank John Kallaugher, Adrian Trejo Nuñez and everyone else there for making my stay enjoyable and exciting. In general the international community around Theoretical Computer Science has been nothing but friendly and accepting. Particularly the Locality Sensitive Hashing group around MIT and Columbia University has been greatly inspiring, and I want to thank Ilya Razenshteyn for many illuminating discussions.

In Austin I also met Morgan Mingle, who became a beautiful influence on my life. So many of my greatest memories over the last two years focus around her. She is also the only person I have never been able to interest in my research.. [1]

Finally I want to thank my parents, siblings and family for their unending support. Even as I have been absent minded and moved between countries, they have always been there, visiting me and inviting me to occasions small as well as big.

[1] She did help edit this thesis though — like the acknowledgements.

Contents

Chapter 1

Introduction

> Right action is better than knowledge, but in order to do what is right, we must know what is right.
>
> Charlemagne

As the power of computing has increased, so has the responsibility given to it by society. Machine Learning and algorithms are now asked to identify hate speech on Facebook, piracy on YouTube, and risky borrowers at banks. These tasks all require computations to be done on the meaning behind text, video and data, rather than simply the characters, pixels and bits. Oddly the philosophically complex idea of "meaning" is now commonly represented in Computer Science simply as a vector in $\mathbb{R}^d$. However, this still leaves the questions of what computations can be done, how, and using how many resources.

For example, say we want to find duplicate posts in Facebook's reportedly 300 billion comments a year. A classic but efficient way to represent the meaning of documents such as a Facebook post is the "bag of words" approach. We create a vector in $\{0,1\}^d$, indicating for every word in the English Language whether it is present in the document or not. We would then want to find two vectors with a large overlap, however such a task requires computations quadratic in the dataset size (here $\approx 10^{23}$), and is thus completely unfeasible, as is shown in this thesis and work following it [1]!

The topics covered in this thesis may be seen as the "tools" to work with large data, such as the above, or whatever its origin. Our main focus is the search problem in which we are given an object and want to discover other objects that are approximately similar (under some given metric). We will (1) give a new most efficient algorithm in the case of $\{0,1\}^d$ vectors, (2) study the extend to which randomness makes the problem easier, (3) discover better ways to handle large outputs, (4) find better ways to pre-process data, and (5) show that sometimes the problems are just provably unfeasible.

1.1 Overview of Problems and Contributions

This thesis is a collection of articles, lightly edited to fit into a combined document. Each article is devoted to a separate problem, which we briefly describe below. Further details can be found in the later sections of this introduction.

Set Similarity Search The (w_q, w_u, w_1, w_2)-Gap Set Similarity Search problem (GapSS) is to, given some universe U, pre-process n sets $Y \subseteq \binom{U}{w_u|U|}$[1] such that given a query $q \in \binom{U}{w_q|U|}$ if there exists $y \in Y$ with $|y \cap q| \geq w_1|U|$, then we can efficiently return $y' \in Y$ with $|y' \cap q| > w_2|U|$. GapSS generalizes nearly all Set Similarity Search problems, including approximative sub-/superset queries and partial match, a classic problem going back to Rivest's PhD thesis [162].

We give a new data structure for this problem – generalizing and improving previous state of the art algorithms such as MinHash, Chosen Path and Spherical LSH. The algorithm gives a complete space/time trade-off for the problem, something previously only known for data on the sphere [61]. For our technique, which is based on so called "supermajority functions", we give a number of lower bounds, based on p-biased hypercontractive inequalities, showing that supermajorities are the right LSH functions for binary data.

Las Vegas Similarity Search The Locality Sensitive Hashing, LSH, framework for high dimensional similarity search data structures was introduced by Indyk and Motwani in 1998 [95] and gave the first algorithms without an exponential dependency on the data dimension for a large number of problems. The algorithms were randomized, and it was a major open problem if it were possible to match the performance deterministically. To this day, the closest thing we have is deterministic 3-approximation to the nearest neighbour problem by Indyk in [94].

Second best to deterministic algorithms are Las Vegas algorithms. In this version of the nearest neighbour problem, we are allowed to use randomness, but we must always return the correct answer. In this thesis we show that such algorithms can match the state of the art in the Locality Sensitive framework, improving upon a prior algorithm of Rasmus Pagh [148] by a polynomial factor in space and query time.

Later work [185] has shown that similar techniques can handle the even more general framework of "Data Dependent LSH", getting us closer than ever to a full derandomization of Indyk and Motwani.

Output Sensitive Similarity Search With LSH, finding the nearest neighbour takes roughly $\tilde{O}(n^\rho)$ time, for some constant $\rho < 1$. Finding the k nearest neighbours takes time $\approx kn^\rho$. In output sensitive similarity search we would like to reduce this to $\approx k + n^\rho$. For many practical problems, such as k-nearest neighbour classifiers and recommendation systems, this could mean a speedup of more than a factor 100 or 1000.

In [10], together with Rasmus Pagh and Martin Aumüller we show how to achieve query time close to the optimal. Our algorithms have the extra benefit of "parameter

[1] Here $\binom{S}{k}$ denotes all subsets of S with k elements.

freeness", which means that the optimal LSH parameters are chosen dynamically on a per query basis for close to no extra cost. This contrasts with classical hyper parameter tuning of LSH, which optimizes towards the average case query. Techniques inspired by our work was later used for so called "confirmation sampling"[64].

Maximum Inner Product Search After nearest neighbours, the perhaps most important high dimensional search problem is Maximum Inner Product. This has is used in virtually all multi-label machine learning classifiers, and applications ranges all the way to finding large entries in matrix vector multiplication.

It was known since [186] that the problem is computational intractible, but in [12] I showed together with Rasmus Pagh, Francesco Silvestri, and Ilya Razenshteyn that even the approximate version of this problem is hard. This explains the lack of efficient theoretical results. This work was later extended by the seminal work [1], which today lays the foundation for all approximative SETH lower bounds.

Tensor Sketching An Oblivious Subspace Embedding $M \in \mathbb{R}^{k\times d}$ is a matrix such that $|\|Mx\|_2 - \|x\||_2 \le \epsilon$ for all $x \in \mathbb{R}^d$. If M can be applied quickly to vectors on the form $x = x^{(1)} \otimes \cdots \otimes x^{(c)} \in \mathbb{R}^{d^c}$ we call it a Tensor Sketch [147, 155, 37]. These have large number of applications [188], such as guaranteeing the correctness of kernel-linear regression performed directly on the reduced vectors.

In this thesis we construct the first Tensor Sketch that needs only $O(c^2(\epsilon^{-1}k^2 + \epsilon^{-2}k))$ rows, improving over the original $\approx 3^c\epsilon^{-2}k^2$[37] giving better algorithms for many downstream applications. We also show other versions with different rows/application time trade-offs.

See section 6.1.2 for comparisons and some recent developments.

We continue to give an overview of the areas of research touched by this thesis, and how they are related. This contrast the introductions of the individual papers, which are unedited and gives the picture of the litterature at the time of their publication.

1.2 Similarity Search

Similarity Search or is the task of building an index on a dataset such that given a query, we can find the data point with the highest similarity to the query, faster that brute force enumerating the entire dataset.

The most common version is the metric version, known as Near Neighbour Search (NNS). Here points come from some metric space, and the similarity between two points is their negative distance. For example for the ℓ_2 space the problem is: Given a set $X \subseteq \mathbb{R}^d$ and $q \in \mathbb{R}^d$, find $x \in X$ that minimizes $\sqrt{\sum_{i=1}^d (q_i - x_i)^2}$. We can however define the problem for any *any* similarity function, that maps the data point and the query into the real numbers.

The definition of similarity search is thus a strong one, and naturally contains many important problems in areas of computer science including pattern recognition, searching in multimedia data, vector compression, computational statistics, and data

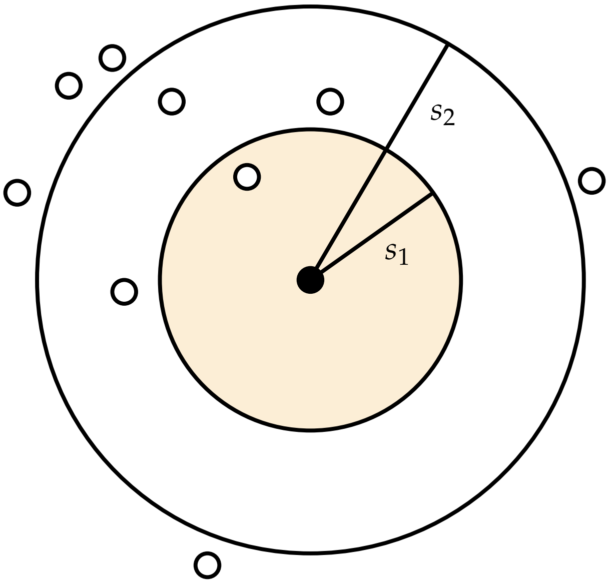


Figure 1.1: The Approximate Similarity Search Problem: The white points are the dataset and the black point is the query. We are guaranteed that there is a point with similarity s_1 (the yellow circle), but is satisfied with any points of similarity more than s_2 (the white circle).

mining [177]. It is also related to many other problems involving distances, such as closest pair, diameter, minimum spanning tree and clustering.

Versions where points come from $\mathbb{R}^d$ have been particularly studied. "Tremendous research effort has been devoted to this problem" – Ryan Williams [56], and for small d many efficient algorithms are known [46]. Yet all known algorithms for $d = \Omega(\log n)$ exhibit a query time of at least $n^{1-O(\epsilon)}$ when the approximation ratio is $1+\epsilon$, approaching the brute-force query time n when ϵ goes to 0. A reason for this was given in [186] as it was shown that k-SAT can be solved by a number of similarity searches on a data structure where the similarity function is inner product [186]. Any data structure that answers queries for this variant in time $n^{1-\epsilon}$ (as well as ℓ_2 and other equivalent distances) for any $\epsilon > 0$, would break the Strong Exponential Time Hypothesis.

The above issue, also known as "The Curse of Dimensionality" is also illustrated in the best known algorithms for Near Neighbour Search, which all have space or time requirements *exponential* in the dimension. There are algorithms with $O(d^{O(1)} \log n)$ query time, but using roughly $n^{O(d)}$ space [65, 128], and in the other end of the scale, there are algorithms using near linear space, but the best data query time bound (on random inputs) is on the form $\min(e^{O(d)}, dn)$ [177].

These issues have lead to the study of *Approximate* Similarity Search, which we discuss next.

1.3 Approximate Similarity Search

The Approximate Similarity Search problem is defined as follows. See also fig. 1.1.

Definition 1 (Approximate Similarity Search or (s_1, s_2)-SS). *Let U be some set and $S : U \times U \to \mathbb{R}$ be some similarity function between pairs of points in U. Given a set $P \subseteq U$, design a data structure that supports the following operation: For any query $q \in U$, if there exists $p \in P$ such that $S(p, q) \geq s_1$, find a point $p' \in P$ such that $S(q, p') \geq s_2$.*

In his thesis [92], Piotr Indyk showed that the above definition is the right one for many of the applications described above. For problems such as furthest neighbour, closest pair, minimum spanning tree, facility location, bottleneck matching and pattern matching, reductions to Approximate Similarity Search yield state of the art approximate algorithms for those problems.

Curiously algorithms for this version of Similarity Search also tends to yield very good algorithms for the exact version of the previous section [64]. A reason for this is the so called "gap phenomenon": Often most of the data points are much further from the query than the nearest neighbour [161].

It is intuitive to think of the dataset as a set of random vectors in $\{0, 1\}^d$ plus a planted point x, which is say distance $d/3$ from our query. If $d = \Omega(\log n)$, the random points will all have distance $d/2 + \tilde{O}(\sqrt{d})$ to the query, so there is a factor 1.5 gap. This is enough to achieve polynomial space and sublinear query time.

In practical datasets most points tend to be similarly uncorrelated with the queries, and the approximate versions is thus sufficient to recover the 'exact' most similar points. The assumption even carries over to a wide range of widely different down stream applications, such as cryptanalysis [114] and training neural networks [172].

Approximate Similarity Search has been particularly successful because of the class of algorithms known as Locality Sensitive Hashing, which we describe in the next section.

1.4 Locality Sensitive Hashing

One intuition for how the "Curse of Dimensionality" appears is the following simple algorithm, which works well for Nearest Neighbour on $\mathbb{R}^d$ in small dimensions: Make a grid over $\mathbb{R}^d$ and store every point in their grid cell. When making a query, the nearest neighbour is bound to fall either in the same cell, or one of the surrounding cells. In two dimensions there are just 8 neighbours, in 3 dimensions there are 26, but in general there are $3^d - 1$. When is logarithmic in the number of points stored, $d = \Omega(\log n)$, the algorithm is no longer faster than comparing the query to the entire dataset. See fig. 1.2.

Locality Sensitive Hashing uses a similar idea, but instead of a fixed grid, it uses multiple random grids (space partitions more generally) and only look in the cell where the query lands. In the most simple version of LSH, the following definition is used:

Definition 2 (Locality Sensitive Hashing, LSH). *A family of hash functions $h : U \to [m]$ is called (s_1, s_2, p_1, p_2)-sensitive (for $s_1 > s_2$ and $p_1 > p_2$) if for any $q, y \in U$:*

1. *if $S(q, y) \geq s_1$, then $\Pr[h(q) = h(y)] \geq p_1$, and*

2. *if $S(q, y) < s_2$, then $\Pr[h(q) = h(y)] \leq p_2$.*

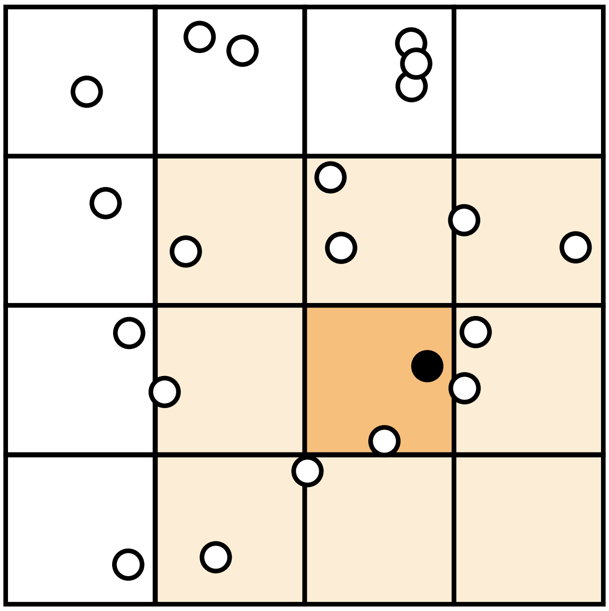


Figure 1.2: A simple points in grid data structure. To ensure we have found the nearest neighbour, we must search all adjacent grid cells.

Given such an h, tailored for our similarity function, S, we can create a hash table in which similar points are likely to collide. Usually we have to tune h by concatenating a number of independent values, and boost our probability of recovering a good point by constructing multiple independent[2] tables.

Doing so, given an (s_1, s_2, p_1, p_2)-sensitive LSH-family, $\mathcal{H}$, and a subset $X \subseteq \mathbb{R}^d$ of size $|X| = n$, there exists a solution to the (s_1, s_2)-Approximate Similarity Search problem using $O(n^{1+\rho} + dn)$ space and with query time dominated by $O(n^\rho \log n)$ evaluations of $\mathcal{H}$, [3] , where we can take $\rho = \frac{\log 1/p_1}{\log 1/p_2}$.

Below are some examples of commonly studied hash functions.

Bit-sampling Indyk at Motwani originally considered the space $U = \{0,1\}^d$ under Hamming distance, corresponding to the similarity function $S(x,y) = 1 - \|x - y\|/d$ where $\|x - y\| = \sum_{i \in [d]} [x_i \neq y_i]$. They took $h(x) = x_i$ where $i \in [d]$ was chosen uniformly at random. Consider $x, y \in U$, then $\Pr[h(x) = h(y)] = 1 - |x - y|/d$, since x and y "collide" under h exactly if the random i falls outside of the $|x - y|$ differing coordinates. By the above construction we then solve the (s_1, s_2)-Similarity Search problem with space $O(n^{1+\rho} + dn)$ and query time $\tilde{O}(n^\rho)$ for $\rho = \frac{\log s_1}{\log s_2}$.

MinHash In [48] Andrei Broder introduced a hash-family for Jaccard similarity $S(x,y) = |x \cap y|/|x \cup y|$ (where $x, y \in \{0,1\}^d$ are considered as subsets of $[d]$). The LSH-family samples a random permutation $\pi : [d] \to [d]$ and let $h(x) = \arg\min_{i \in x} \pi(i)$. Then $\Pr[h(x) = h(y)] = |x \cap y|/|x \cup y|$ since x and y collide exactly if the smallest (under π) element of $x \cup y$ is in $x \cap y$. Again we to solve the (s_1, s_2)-Similarity Search problem with $\rho = \frac{\log s_1}{\log s_2}$.

[2] Two-independence suffices, as has been noted by [62] and others.

[3] If evaluations of $\mathcal{H}$ are slow, it is possible to reduce the number substantially [71] leaving a query time dominated by $\tilde{O}(n^\rho)$ table lookups.

SimHash Moses Charikar suggested [50] using the SDP rounding technique of random hyperplanes to sketch angular similarity, $S(x,y) = \arccos\left(\frac{\langle x,y\rangle}{\|x\|\|y\|}\right)/\pi$. Given $x \in \mathbb{R}^d$ he takes $h(x) = [\langle x, g\rangle]$ where $g \sim \mathcal{N}(0,1)^d$ is sampled as standard Gaussian vector, independent for each h. [4] We again get $\Pr[h(x) = h(y)] = S(x,y)$ and $\rho = \frac{\log s_1}{\log s_2}$.

Based on these examples, it may seem that finding h such that $\Pr[h(x) = h(y)] = S(x,y)$ is a good idea. A similarity function with such a hash function is known by [58] as "LSH feasible". It turns out, however, that this is usually not what we need to get the best possible data structures. The best LSH data structures for angular similarity and Jaccard are [21] and this thesis, both of which beet the direct LSH functions above. For angular similarity [21] gets $\rho = \frac{1+\cos s_1\pi}{1-\cos s_1\pi}\frac{1-\cos s_2\pi}{1+\cos s_2\pi} \ll \frac{\log s_1}{\log s_2}$. Using so called Data Dependent LSH it is possible to do even better when $s_2 > 1/2$.

The first data structure to beat MinHash for Jaccard Similarity was Chosen Path [63]. It used a slightly more general framework than definition 2 introduced by Becker et al. [43] known as Locality Sensitive Filters. We will use the following definition by Christiani [61]:

Definition 3 (Locality Sensirtive Filters, LSF)**.** *Let X be some universe, let $S : X \times X \to \mathbb{R}$ be a similarity function, and let $\mathcal{F}$ be a probability distribution over $\{(Q,U) \mid Q,U \subseteq X\}$. We say that F is $(s_1, s_2, p_1, p_2, p_q, p_u)$-sensitive if for all points $x, y \in X$ and (Q,U) sampled randomly from $\mathcal{F}$ the following holds:*

1. *If $S(x,y) \geq s_1$ then $\Pr[x \in Q, y \in U] \geq p_1$.*
2. *If $S(x,y) \leq s_2$ then $\Pr[x \in Q, y \in U] \leq p_2$.*
3. *$\Pr[x \in Q] \leq p_q$ and $\Pr[x \in U] \leq p_u$.*

We refer to (Q,U) as a filter and to Q as the query filter and U as the update filter.

See also fig. 1.3. The corresponding theorem to the LSH algorithm is the following from [61]:

Theorem 1 (LSF theorem)**.** *Suppose we have access to a family of filters that is $(s_1, s_2,\ p_1, p_2, p_q, p_u)$-sensitive. Then we can construct a fully dynamic data structure that solves the (s_1, s_2)-similarity search problem with query time $dn^{\rho_q + o(1)}$, update time $dn^{\rho_u + o(1)}$, and space usage $dn + n^{1+\rho_u+o(1)}$ where $\rho_q = \log(p_q/p_1)/\log(p_q/p_2)$ and $\rho_u = \log(p_u/p_1)/\log(p_q/p_2)$.*

We must be able to sample, store, and evaluate filters from $\mathcal{F}$ in time $dn^{o(1)}$.

Locality Sensitive Filters were instrumental in getting optimal space/time trade-offs in the Locality Sensitive framework [115, 61, 25]. Previous algorithms by Kapralov and others [103, 125] used the "Multi-probing" framework, in which after searching the $h(x)$ bucket, additional "nearby" buckets were sampled from some distributed and searched. With LSF we can use buckets of different sizes for queries and data points. See fig. 1.3. With this approach it is possible to get $c^2\sqrt{\rho_q} + (c^2-1)\sqrt{\rho_u} \geq \sqrt{2c^2-1}$

[4] Thijs Laarhoven showed in [116] that we can improve ρ by orthogonalizing the hyperplanes.

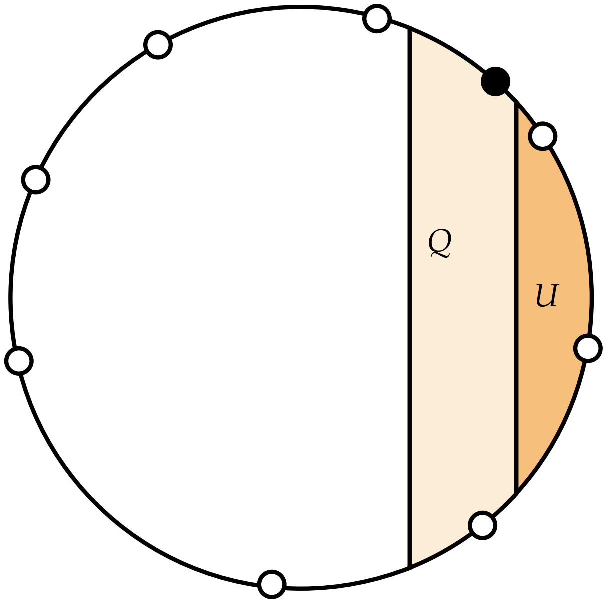


Figure 1.3: LSF with smaller space usage than query time. Here $Q = \{x \in S^{d-1} : \langle x, g\rangle \geq t_1\}$ and $U = \{x \in S^{d-1} : \langle x, g\rangle \geq t_2\}$ for some Gaussian vector g and thresholds t_1, t_2.

for any permitting choice of ρ_q and ρ_u. The authors also show that this trade-off is optimal for any algorithms in the model described.

In this thesis we show that one can also get space/time trade-offs for similarity search of sets – generalizing and improving previous state of the art algorithms such as MinHash, Chosen Path and Spherical LSH.

Indyk, Broder, and Charikar all received the 2012 ACM Paris Kanellakis Theory and Practice Award "for their groundbreaking work on Locality-Sensitive Hashing that has had great impact in many fields of computer science including computer vision, databases, information retrieval, machine learning, and signal processing".

Since then there has been a number of variations and important related problems. We already mentioned time/space trade-offs with LSH. Becker et al. [43] studied LSF for "medium dimension" data, $d = \Theta(\log n)$, and [31] introduced the important idea of "Data Dependent LSH". The later let to the break-through of LSH for general norms [28, 27].

Another direction is the complete or partial derandomization of LSH. We discuss this problem in the next section.

1.5 Las Vegas Similarity Search

A key issue with the data structures described is that they don't actually guarantee finding any near neighbours. To the definition 1 should really read "with probability at least $1 - \delta$" where δ, and the algorithms mentioned should have a factor $\log 1/\delta$ multiplied unto their n^ρ performance. This is a problem for algorithms that need to be transparent and predictable. If a fingerprint database returns "no match" we want to be sure that the suspected criminal is really innocent and not have to wonder if our

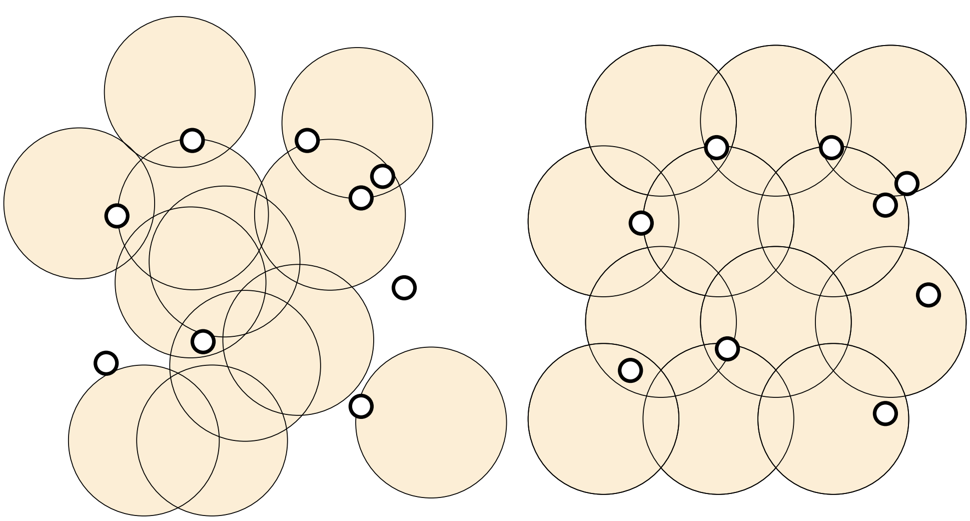


Figure 1.4: The filters on the right guarantee that any two points, sufficiently close, share some filter. The filters on the left only have this guarantee with some probability.

algorithm just predictably failed. Other examples might be blocking people on twitter, paternity tests or prison sentence measuring.

A further issue is that the standard analysis of LSH isn't quite precise about its guarantees on δ. In [180] it was pointed out that it is often "difficult or even impossible to obtain a theoretical bound on failure probability, i.e., the probability of a false negative being produced".

Because of these issues many authors have studied deterministic and Las Vegas versions of LSH. While deterministic algorithms are still either exponential in d or only support relatively large approximation factors [94], Las Vegas algorithms have been more successful.

Already in 2000 [113] it was suggested that one might take δ small enough to be able to union bound over every possible query. This implies that, with constant probability, every query would work without false negatives, rather than having the probabilistic guarantee on queries individually. Unfortunately this only works in $\{0,1\}^d$, and even when it works, it only shows that it is possible in principle to get a Las Vegas data structure. It turns out to be NP-hard to actually check that we indeed have such a data structure [9].

In [148, 32] it was shown how to use combinatoric techniques in addition to the usual probabilistic arguments to create correlated LSH hash tables, which in combination would guarantee that a near neighbour would be found in at least one of the tables. Instead of $\log 1/\delta$ independent repetitions it was shown that some polynomial number in n sufficed. This means, for example, that they needed more repetitions than LSH does to get 0.99 success rate, but fewer than LSH needs for success rate $1 - 2^{-n}$.

In in this thesis (originally [11]), we show that it is even possible for a Las Vegas data structure to match the classically randomized data structures up to a factor $1 + o(1)$ in ρ. This is done using a general approach, rather than one tailored to a specific similarity measure, which allows us to match classical LSH on both Hamming distance and Braun-Blanquet.

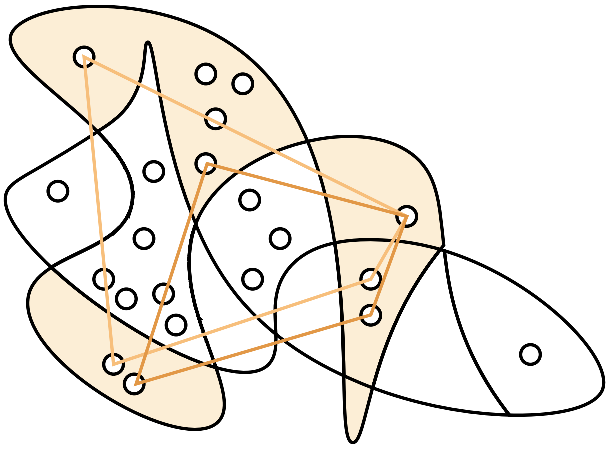


Figure 1.5: A perfect hash-family maps the large universe into buckets of size k^2, such that for any subset, S, of size k there is a h in the family such that $h(S) = |S|$. We use a (k^2, k, r)-Turán design on the buckets, which is guaranteed to contain r of the k buckets. We then take the every combination of points in these buckets as the new r-sets, here catching 20.

The goal is to make a set a locality sensitive filters, such that any two near points are guaranteed to share at least one filter. See fig. 1.4 for a visualization.

The first step is to reduce the dimension the roughly $(\log n)^{1+\epsilon}$. This is done using partitioning ideas from [148, 32] as well as a one-sided-error version of a lemma in [113]. At the same time the union bound idea of [113], mentioned above, we can create a filter family that is likely to work, and if $d = (\log n)^{1-\epsilon}$ we can even check this efficiently. The hard part us thus to "cross the logarithmic barrier" and go from $d = (\log n)^{1-\epsilon}$ to $d = (\log n)^{1+\epsilon}$.

For this we apply the derandomization technique of splitters [136]. These can be viewed as families of partitions of $1, \ldots, 2d$ into pairs of sets $(S_1, \overline{S}_1), (S_2, \overline{S}_2), \ldots$ of size d, such that for any x, y, there is a pair $(S_l, \overline{S}_l)$ for which $\|x_{S_l} - y_{S_l}\| = \|x_{\overline{S}_l} - y_{\overline{S}_l}\| \pm 1$. These families can be taken sufficiently small as to be negligible.

For Braun-Blanquet a particular issue is handling the cases where the dimension (universe size) is large, but the set sizes are small. For this we use the idea of a Turán design [178] named after Paul Turán used in his work on hyper graphs. A (n, k, r)-Turán design, $\mathcal{T}$ is a family of size r sets, such that any size k subset $S \subseteq [n]$, there is $R \in \mathcal{T}$ such that $R \subseteq S$. The splitter techniques suffices to create these of shape roughly $(k^{O(1)}, k, \log n)$ with size near the expectation for random constructions. To get n rather than $k^{O(1)}$ we use perfect hashing.

An (n, k^2, k)-perfect hash family, $\mathcal{H}$, is a family of functions $h : [n] \to [k^2]$ such that for any size k subset $S \subseteq [n]$, there is $h \in \mathcal{H}$ such that $|h(S)| = |S|$. Noga Alon showed in [16] how to deterministically construct such a family of just $k^4 \log n$ functions. In fig. 1.5 we give some intuition for how this perfect hashing is used to increase the size of Turán designs.

Recently Wei [185] showed how to extend the idea of splitters to the sphere S^{d-1} using Count Sketches.[5] Wei also shows how to connect Las Vegas LSH with

[5] It turns out that splitters work on the sphere as well, given we first randomly rotate the data so all coordinates are small. However the Count Sketch approach is much more elegant.

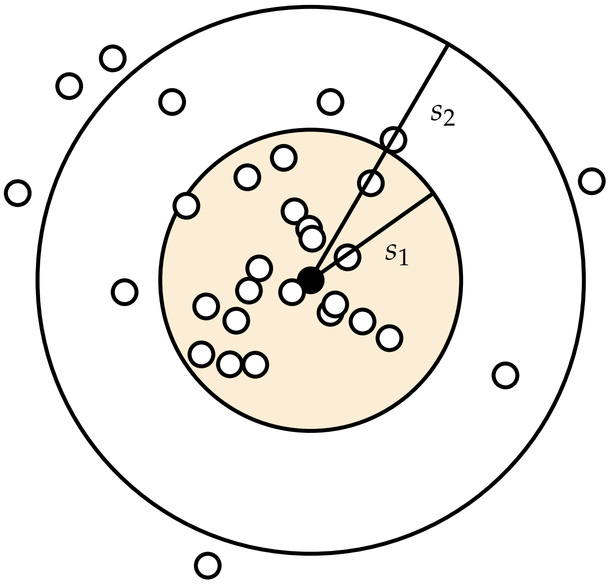


Figure 1.6: An example from [154] of an LSH bottleneck. Compare this to fig. 1.1. The large points in the s_1 circle are likely to collide with the query in many LSH buckets, thus when trying to return the k nearest (distinct) neighbours, we end up spending much time on deduplication.

Data Dependent LSH getting data structures with no false negatives matching the performance of [25].

1.6 Output Sensitive Similarity Search

In many applications of Similarity Search it is necessary to return more than one similar point. Natural examples are recommendation systems, which should give more than one recommendation, and k-nearest neighbour classifiers, which work by labeling a query as the median of the k most similar labeled examples.

While it is possible for traditional LSH to return more than one similar point, it can suffer a large (factor k) penalty when doing so. This is because a single very near neighbour is likely to show up in nearly all of the n^ρ hash tables. See fig. 1.6 for an example.

In [154] a strategy was suggested to detect when many duplicates were happening so one might switch to brute force search, getting a query time of $\min(n, kn^\rho)$. Making LSH actually output sensitive, that is getting the query time from kn^ρ to $k + n^\rho$, is an important practical problem.

In this thesis we suggest a strategy for parameter free search, which is able to use different LSH parameters depending on the number of points close to the query. This has the added benefit of saving the user a difficult tuning step when creating the data structure, which in any case would only be optimal in average, rather than for the query only known later. Our first algorithm has expected query time bounded by $O(k(n/k)^\rho)$, where k is the number of points to report and $\rho \in (0, 1)$ depends on the data distribution and the strength of the LSH family used. We finally present a parameter-free way of using multi-probing, for LSH families that support it, and show that for many such families this approach allows us to get expected query time close to $O(n^\rho + t)$, which is the best we can hope to achieve using LSH.

Another approach follows from using a separate data structure for density estimation, such as [92] or [189]. This has a certain log overhead however, which can be significant for very fast queries.

1.7 Hardness of Similarity Search through Orthogonal Vectors

Lower bounds for LSH like algorithms have traditionally been given either in the very general cell-probe model [152] or the very specific model of boolean functions [132]. One is nice because the bounds hold unconditionally, while the second is nice because we can show tight upper and lower bounds.

In recent years a third way has become popular for showing lower bounds in all of theoretical computer science, that is by reduction to the Orthogonal Vectors Conjecture (or the Strong Exponential Time Hypothesis). The list of "OVC-Hard" problems is long, including central problems in pattern matching and bioinformatics [4, 39], data structures [3] and too many more areas to list here.

It is exciting whether we can show lower bounds for Similarity Search matching our upper bounds. One immediate issue is that Valiant showed $1+\epsilon$ approximate nearest neighbours can be solved as a batch problem in $n^{2-\Omega(\sqrt{\epsilon})}$ time for small ϵ, beating the $n^{2-\Omega(\epsilon)}$ time we get by repeating our best LSH n times. However it would be interesting to at least show that polynomial time is required.

In this thesis we show that this is indeed the case for the similarity measure of Inner Product. We show that for $c = \exp(\sqrt{\log n}/\log\log n)$ the (s_1, cs_1)-similarity search problems requires query time at least $n^{1-\delta}$ for any $\delta > 0$, assuming we want polynomial space.

Previously the only hardness of approximation results known under OVC were problem specific things like distinguishing whether the diameter of a graph on O(n) edges is 2 or at least 3 in truly-sub-quadratic time refutes SETH[163], which implies hardness for $(3/2-\epsilon)$ approximations. In 2017 a major break through was accomplished by [1] who showed a beautiful reduction using probabilistically checkable proofs and hardness for $c = \exp((\log n)^{o(1)})$. Later authors used this "Distributed PCP" to show hardness for Hamming distance [164] and other similarity problems [55].

1.8 Tensor Sketching

The algorithms we have described so far are useful for the problem they are made for: Similarity Search. Often, however, we don't have an algorithm for the exact similarity measure we are interested in, and we need to do some preprocessing to fit it into our algorithms.

In the first section we mentioned the embedding of text into $\{0,1\}^d$. Now a larger inner product meant a large overlap of words. But what if our documents are long and contain nearly all of the words, then we don't get much information from that. An alternative can then be to look all pairs of words, that is mapping into $\{0,1\}^{d^2}$, or even

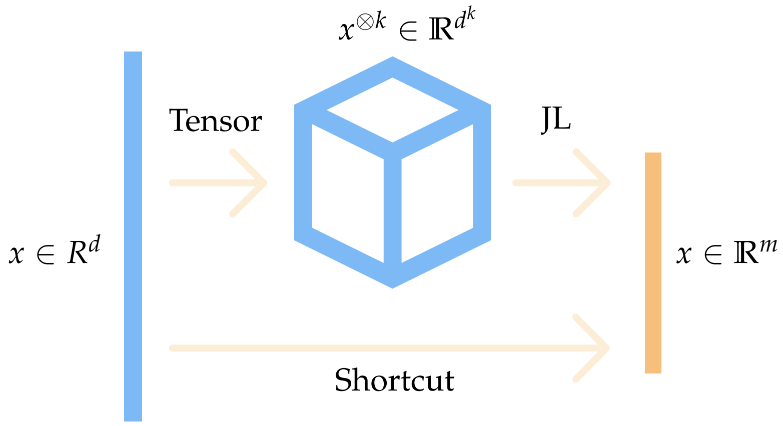


Figure 1.7: Tensor Sketch allows sketching $x^{\otimes k}$ in time and space nearly the same as sketching x.

triples or quadruples. This is called tensoring in the machine learning community. Tensoring works fine with the data structures as described above, however the size of the new space, d^k for some k, may be unmanageable. It is thus natural to want to reduce the dimension, using such ubiquitous tools as the Johnson Lindenstrauss transformation [102, 118], which preserves the inner products while yielding much smaller vectors. Figure 1.7 shows how Tensor Sketching can provide a convenient shortcut for the operations of tensoring, then dimensionality reducing.

The idea of Tensor Sketching goes back to Pagh and Pham [155, 147]. They showed that given two independent count sketches [6] $Cx, C'x \in \mathbb{R}^k$ of a vector $x \in \mathbb{R}^d$, then $\mathcal{F}^{-1}(\mathcal{F}Cx \circ \mathcal{F}C'x)$, where $\mathcal{F}$ is the Fourier transform and $(\circ)$ is the element-wise product, would be a count sketch of $x \otimes x$. This allows computations of count sketches of $x^{\otimes k}$ in time roughly $kd \log d + k \log k$. However, if all we want is an embedding mimicking the tensor product, there are however also other possibilities, such as Fourier features [119, 157] and hyperplane rounding [52].

Tensor Sketches can do more than just embedding for similarity search however. Woodruff et al. showed in [37] that the Tensor Sketch by Pagh and Pham is in fact an Oblivious Subspace Embedding. That is, let M be the Tensor Sketch matrix mentioned, such that $Mx^{\otimes 2} = \mathcal{F}^{-1}(\mathcal{F}Cx \circ \mathcal{F}C'x)$, then for any $y \in \mathbb{R}^{d^2}$ we have $\|My\| = (1 \pm \epsilon)\|y\|$. In [188] Woodruff gives a large number of application of such embeddings, including state of the art algorithms for ℓ_p-regression, low rank approximation, and principal component analysis (PCA) under polynomial kernels.

Since C is a simple dimensionality reduction matrix, and $\mathcal{F}$ is a random rotation, but might consider the sketch $Mx \circ M'x$ where M and M' are fully random Gaussian matrices in $\mathbb{R}^{d \times k}$. We show that this only requires dimension $\epsilon^{-2} \log 1/\delta + \epsilon(\log 1/\delta)^2$.

This sketch takes time dk to calculate however, while the Tensor Sketch by Pagh and Pham only took roughly $k + d \log d$. We show that Fast Johnson Lindenstrauss ala Ailon and Chazelle [13] is a Tensor sketch as well and give the tightest known number of rows needed, though the exact number it requires is still unknown.

In another recent paper Kapralov et al. [105] show similar results. It is however still an open problem how many rows a (fast) Tensor Sketch needs. The only known lower bound is that [118] of the actual Johnson Lindenstrauss transformation.

[6] A count sketch matrix [50] $C \in \mathbb{R}^{k \times d}$ is defined as follows: Given a two-independent hash function $h : [d] \to [k]$, and a four-independent hash function $s : [d] \to \{0, 1\}$, we let $C_{i,j} = [h(j) = i]s(j)$.

Chapter 2

Small Sets Need Supermajorities: Towards Optimal Hashing-based Set Similarity

2.1 Introduction

Sparse boolean vectors arises from the classical representation of documents as "bags of words", where non-zero vector entries correspond to occurrences of words (or shingles). Another example is one-hot encoding of categorical data, such as which movies a user has watched.

Data structures for such data has been constructed for queries such as Superset / Subset / Containment, Partial Match, Jaccard similarity and maximum inner product search (MIPS). Early work goes back to Ronald Rivest's thesis [162] and many later papers have tackled the issue [51, 69]. Unfortunately these problems are equivalent to the Orthogonal Vectors problem [56], which means that we can't do much better than a brute force search through the database.

Hence recent research has focused on approximate versions of the problem, with MinHash (a.k.a. min-wise hashing) by Broder et al.[48, 47] being a landmark result. These problems are usually defined over some "simmilarity measure" like Jaccard similarity or Inner Product, with Chosen Path for Braun Blanquet similarity [63] being a recent break through. It has been observed however, that knowing the size of the sets in the database and queries make all of these equivalent [63], including more than 76 binary similarity (and distance) measures defined in the survey [60]. This method, sometimes known as "norm ranging" is also practical, giving state of the art results at NeurIPS 2018 [192].

We thus define the Gap Similarity Search problem, as the approximate set similarity search problem that is aware of the set weights. Recall the definition from the abstract: The (w_q, w_u, w_1, w_2)-GapSS problem is to, pre-process n sets $Y \subseteq \binom{U}{w_u|U|}$ such that given a query $q \in \binom{U}{w_q|U|}$ we can efficiently return $y' \in Y$ with $|y' \cap q| > w_2|U|$ or determine that there is no $y \in Y$ with $|y \cap q| \geq w_1|U|$. Here U is some universe set, which we can assume to be larger than $\omega(\log n)$ by duplication if necessary.

Note that GapSS includes approximate subset/superset queries by setting $w_1 = w_u$ or $w_1 = w_q$. The classical setting of a planted similar point on a background of random data [162], is included with $w_2 = w_q w_u$.

MinHash, Chosen Path and the extremely studied Spherical LSH [25] all solve the GapSS problem, with different algorithms being more efficient for different ranges of parameters. While the sketching problem for sets have been studied extensively, with faster MinHash algorithms such as [71], the search problem is less well understood. In [63] it was left as an open problem to unify the above methods, ideally finding the optimal LSH algorithm for set data. That is the problem we tackle in this paper.

Approach The proposed data-structure is a simple "list-of-points" data structure: We sample m sets $S_i \subseteq U$ independently and with replacement. We define

$$F_q^{(i)}(q) = \begin{cases} [|q \cap S_i| \geq t_q|S_i|] & \text{if } t_q \geq w_q, \\ [|q \cap S_i| \leq t_q|S_i|] & \text{if } t_q < w_q, \end{cases} \quad \text{and } F_u^{(i)}(y) = \begin{cases} [|y \cap S_i| \geq t_u|S_i|] & \text{if } t_u \geq w_u, \\ [|y \cap S_i| \leq t_u|S_i|] & \text{if } t_u < w_u, \end{cases} \tag{2.1}$$

and store each y from the database in lists indexed each $i \in [m]$ such that $F_u^{(i)}(y) = 1$. When performing a query, q, we search each list indexed by i such that $F_q^{(i)}(q) = 1$ and compare q to each of the ys in those lists, returning the first with $\langle q, y \rangle \geq w_2$. Since t_u (resp. t_q) is usually greater than the expectation of $|y \cap S_i|/|S_i|$ (w_u, resp. w_q) we call these boolean functions supermajorities, taken from social choice theory - "a qualified majority must vote in favour". [1]

While the above algorithm is a simple enough to be described in (roughly) a paragraph, the resulting bounds are complicated, and it is not obvious at first that they would be optimal. Perhaps this is why the scheme hasn't (to our knowledge) been described earlier in the literature. We do however show a number of lower bounds proving that given the right choices of t_u and t_q the scheme is indeed optimal over all choices of functions F_u and F_q for a large range of parameters w_q, w_u, w_1 and w_2. We conjecture that it is optimal over the entire space. For this reason the relative complication is inherent, and researchers as well as practitioners should not shy away from using supermajorities more widely.

A limitation of our result is the assumption that w_x and w_y be constants $\in [0, 1]$. It is common in the literature [71, 63] to assume that the similarities (s_1, s_2) (e.g. Jaccard) are constants, but usually arbitrarily smalls sets are allowed. We believe this is mainly an artefact of the analysis, and something it would be interesting future work to get rid of. In the meantime it also means that we can always hash down to a universe size of $\approx w_2^{-1} \log n$, which removes the need for a factor $|U|$ in the query time.

Intuitively our approach is similar to the Chosen Path algorithm, which fits in the above framework by taking $F_q^{(i)}(q) = [S_i \subseteq q]$ and $F_u^{(i)}(y) = [S_i \subseteq y]$. If $|q|$ is not equal to $|u|$ however (lets say $w_q > w_u$), the queries will have to look in many more lists than each data point is stored in, which gives a non-balanced space/time trade-off. Chosen Path handles this by a certain symmetrization technique (which we will study later), but perhaps the most natural approach is simply to slack the requirement of F_u, including also some lists where S_i is not completely contained in y.

[1] The Chosen Path filters of [63] are similar, but use the 'Consensus' function or 'ALL-function' for both F_u and F_q. The spherical LSF in [25] uses linear threshold functions, but for boolean data it can also use simple majorities (or $1 + o(1)$ fraction majorities.)

In our results we include a comprehensive comparison to a number of other LSH based approaches, which in addition to unifying the space of algorithms also give a lot more intuition for why supermajorities are the right space partition for sparse boolean data.

2.1.1 Related Work

Work on Set Similarity Search has focused on a number of seemingly disparate problems: (1) Super-/Subset queries (2) Partial Match, (3) Jaccard/Braun Blanquet/Cosine similarity queries, and (4) maximum inner product search (MIPS).

The problems all have all traditionally been studied in their exact form:

Super-/Subset queries Pre-process a database D of n points in $\{0,1\}^d$ such that, for all query of the form $q \in \{0,1\}^d$, either report a point $x \in D$ such that $x \subseteq q$ (rasp. $q \subseteq x$) or report that no such x exists.

Partial Match Pre-process a database D of n points in $\{0,1\}^d$ such that, for all query of the form $q \in \{0,1,*\}^d$, either report a point $x \in D$ matching all non-$*$ characters in q or report that no such x exists.

Similarity Search Given a similarity measure $S : \{0,1\}^d \times \{0,1\}^d \to [0,1]$, pre-process a database D of n points in $\{0,1\}^d$ such that, for all query of the form $q \in \{0,1\}^d$ return the point $x \in D$ maximizing $S(q,x)$.

Maximum Inner Product Search Same as Similarity Search, but $S(x,y) = \langle x,y \rangle$.

These problems are all part of an equivalence class of hard problems, known as Orthogonal Vectors [56]. This means that we don't expect the existence of polynomial space data structures that can solve either of these problems faster than a linear scan through the entire database. See also [12, 1, 164].

For this reason people have studied approximate versions of each problem. While the exact definition of the approximation differs in the literature, once we fix the weight of the input vectors, they all become essentially equal to GapSS as defined in this paper. This allows us to compare the best algorithms from each category against each other, as well as against our suggested Supermajorities algorithm. It should be noted that the hardness results mentioned above also holds for approximate variations, so the gap will have to be sufficiently large for any approach to work.

Partial Match The problem is equivalent to the subset query problem by the following well known reductions: (PM → SQ) Replace each $x \in D$ by the set $\{(i, p_i) : i \in [d]\}$. The replace each query q by $\{(i, q_i) : q_i = *\}$. (SQ → PM) Keep the sets in the database as vectors and replace in each query each 0 by an $*$.

The classic approach, studied by Rivest [162], is to split up database strings like *supermajority* and file them under s, u, p etc. Then when given query like *set* we take the intersection of the lists s, e and t. Sometimes this can be done faster than brute force searching each list. He also considered the space heavy solution of storing all subsets, and showed that that when $d \le 2\log n$, the trivial space bound of 2^d can be somewhat improved. Rivest finally studied approaches based on tries and in particular the case where most of the database was random strings. The later case is in some ways similar to the LSH based methods we will describe below.

Indyk, Charikar and Panigrahi [51] also studied the exact version of the problem, and gave algorithms with

1. $n2^{(O(d\log^2 d\sqrt{c/\log n})}$ space and $O(n/2^c)$ time.
2. nd^c space and $O(dn/c)$ query time.

for any $c \in [n]$. Their approach was a mix between the shingling method of Rivest, building a look-up table of size $\approx 2^{\Omega(d)}$, and a brute force search. These bounds manage to be non-trivial for $d = \omega(\log n)$, however only slightly, since otherwise they would break the mentioned OVC lower bounds.

There has also been a large number of practical papers written about Partial Match / Subset queries or the equivalent batch problem of subset joins. [159, 129, 85, 8, 160] Most of these use similar methods to the above, but save time and space in various places by using bloom filters and sketches such as MinHash [48] and HyperLogLog [81].

Maximum Inner Product For exact algorithms, most work has been done in the batch version (n data points, n queries). Here Alman et al. [14] gave an $n^{2-1/\tilde{O}(\sqrt{k})}$ algorithm, when $d = k\log n$.

An approximative version can be defined as: Given $c > 1$, pre-process a database D of n points in $\{0,1\}^d$ such that, for all query of the form $q \in \{0,1\}^d$ return a point $x \in D$ such that $\langle q, x\rangle \geq \frac{1}{c}\max_{x' \in D}\langle q, x'\rangle$. Here [12] gives a data-structure with query time $\approx \tilde{O}(n/c^2)$, and [56] solves the batch problem in time $n^{2-1/O(\log c)}$. (Both when d is $n^{o(1)}$.)

There are a large number of practical papers on this problem as well. Many are based on the Locality Sensitive Hashing framework (discussed below) and have names such as SIMPLE-LSH [138] and L2-ALSH [166]. The main problem for these algorithms is usually that no hash family of functions $h : \{0,1\}^d \times \{0,1\}^d \to [m]$ such that $\Pr[h(q) = h(x)] = \langle q, x\rangle/d$ [12] and various embeddings and asymmetries are suggested as solutions.

The state of the art is a paper from NeurIPS 2018 [192] which suggests partitioning data by the vector norm, such that the inner product can be more easily estimated by LSH-able similarities such as Jaccard. This is curiously very similar to what we suggest in this paper.

We will not discuss these approaches further since, for GapSS, they are all dominated by the three LSH approaches we study next.

Similarity Search The problem is usually studied as an approximate problem: Given a similarity measure $S : \{0,1\}^d \times \{0,1\}^d \to [0,1]$ and $s_1 > s_2 \in [0,1]$, pre-process a database D of n points in $\{0,1\}^d$ such that for queries $q \in \{0,1\}^d$ we return a point $x \in D$ with $S(q, x) \geq s_2$ given there is $x' \in D$ with $S(q, x') \geq s_1$.

This naturally generalizes MIPS as defined above. The formulation allows use application of Indyk and Motwani's LSH framework [95]. Here we define a family, $\mathcal{H}$, of functions $h : \{0,1\}^d \times \{0,1\}^d \to [m]$ such that

1. $\Pr_{h\sim\mathcal{H}}[h(q) = h(x)] \geq p_1$ when $S(q, x) \geq s_1$, and
2. $\Pr_{h\sim\mathcal{H}}[h(q) = h(x)] < p_2$ when $S(q, x) < s_2$.

The constructions in [95, 88] then give an algorithm for the approximate similarity search problem with space $n^{1+\rho} + dn$ and query time dominated by n^ρ evaluations of h, where $\rho = \log p_1 / \log p_2$.

If $\mathcal{H}$ exists such that $\Pr_{h\sim\mathcal{H}}[h(q) = h(x)] = S(q, x)$ is achievable (see [59] for a study of when this is the case) then such a family is an obvious choice. An example of this is Broder's MinHash algorithm, which has $\Pr_{h\sim\mathcal{H}}[h(q) = h(x)] = |q \cap x|/|q \cup x|$ where $S(q, x) = |q \cap x|/|q \cup x|$ is the Jaccard similarity.

Choosing $\mathcal{H}$ like this is however not always optimal, as Christiani and Pagh [63] shows by constructing a data structure with $\rho = \frac{\log 2s_1/(1+s_1)}{\log 2s_2/(1+s_2)} < \frac{\log s_1}{\log s_2}$ when the size of sets is equal, $|q| = |x|$. Their approach, known as Chosen Path, is similar to the one presented in this paper, in that it uses the generalized LSH framework known has Locality Sensitive Filters, LSF [43]. In general they get $\rho = \frac{\log b_1}{\log b_2}$ where $b_1 > b_2 \in [0, 1]$ are Blanquet Similarities $B(q, x) = |q \cap x|/\max\{|q|, |x|\}$.

The most studied variant is LSH on the sphere. Here, given $\alpha > \beta \in [-1, 1]$, we pre-process a database D of n points in S^{d-1} and for a query $q \in S^{d-1}$ return $x' \in D$ with $\langle q, x' \rangle \geq \beta$ given the promise that there is $x \in D$ with $\langle q, x \rangle \geq \alpha$. In [21] they show how to get $\rho_{\text{sp}} = \frac{1-\alpha}{1+\alpha}\frac{1+\beta}{1-\beta}$.[2]

While it is clear that both MinHash and Chosen Path can solve GapSS when w_q and w_u is known in advance, using spherical LSH requires that we embed the binary vectors onto the sphere. Multiple ways come to mind, such as mapping $0 \mapsto -1/\sqrt{d}, 1 \mapsto 1/\sqrt{d}$ or $0 \mapsto 0$, $1 \mapsto 1/\sqrt{w_q d}$ (for queries, resp. $1/\sqrt{w_u d}$ for data points). Depending on how we do it, the algorithm of [21] will naturally return different results, however given knowledge of w_q and w_u there is an optimal embedding[3], as we will show in this paper. This gives $\alpha = \frac{w_1 - w_q w_u}{\sqrt{w_q(1-w_q)w_u(1-w_u)}}$ and $\beta = \frac{w_1 - w_q w_u}{\sqrt{w_q(1-w_q)w_u(1-w_u)}}$ which is better than the two previous methods when w_q and w_u are not too small.

Two other classic methods are Bit Sampling [95] and SimHash (Hyperplane rounding) [52], which give $\rho_{\text{bs}} = \frac{\log(1-w_q-w_u+2w_1)}{\log(1-w_q-w_u+2w_2)}$ and $\rho_{\text{hp}} = \frac{\log(1-\arccos(\alpha)/\pi)}{\log(1-\arccos(\beta)/\pi)}$ respectively. (SimHash also works on the sphere, but has the same optimal embedding as spherical LSH.) These ρ-values however turn out to always be larger than ρ_{sp}, so we won't study them as much.

While Chosen Path and Spherical LSH both have proofs of optimality [63, 26, 145, 133] in the LSH model, these optimality proofs consider specific ranges, like when w_q, w_u or w_1 goes to zero. Hence they are not necessarily optimal when used in all the ranges of parameters in which GapSS is interesting. In fact they each have regions of optimality, as was observed in [63] who proposed as an open problem to find an LSF scheme that unified all of the above. This is what we do in this paper, as well as showing matching lower bounds in a wider range of parameters.

[2] For $\beta \to 1$ this approaches $\log \alpha / \log \beta$, which would be like an LSH-able family for inner product on the sphere, but unfortunately this is not achievable with LSH. For the batch problem it was shown possible in [107].

[3] optimal for embeddings on the form $0 \mapsto a, 1 \mapsto b$.

Trade-offs and Data Dependency The above algorithms, based on the LSH framework, all had space usage roughly $n^{1+\rho}$ and query time n^{ρ} for the same constant ρ. This is known as the "balanced regime" or the "LSH regime". Time/space trade-offs are important, since $n^{1+\rho}$ can sometimes be too much space, even for relatively small ρ. Early work on this was done by Panigrahy [151] and Kapralov [104] who gave smooth trade-offs ranging from space $n^{1+o(1)}$ to query time $n^{o(1)}$. A breakthrough was the use of LSF, which allowed time/space trade-offs with sublinear query time even for near linear space and small approximation [115, 61, 26].

Prior to this article, the only way to achieve trade-offs for set data was to embed it into the above spherical algorithms. In this paper we show that it is often possible to do much better, and in some cases get query time less than that of balanced spherical LSH, even with near-linear space.

Arguably the largest break-through in LSH based data-structures was the introduction of data-dependent LSH[22, 31, 29]. It was shown how to reduce the general case of α, β similarity search as described above, to the case $\beta = 0$ (and $\alpha \mapsto \frac{\alpha-\beta}{1-\beta}$), in which many LSH schemes work better. Using those data structures on GapSS with $w_2 > w_q w_u$ will yield often yield better performance than the algorithms described in this paper. However, since the data-dependent methods are equivalent to Spherical LSH for $w_2 = w_x w_y$, we always dominate this case, and it is an exciting open problem to create similar reductions directly for set data, possibly using the space partitioning proposed in this algorithm as a building block.

2.1.2 Results

We split our results in upper bounds, lower bounds and comparisons with other approaches. Regrettably some of our results have to be stated rather indirectly. We provide fig. 2.1 to guide the intuition, as well as some corollaries with results for specific parameters.

Upper bounds

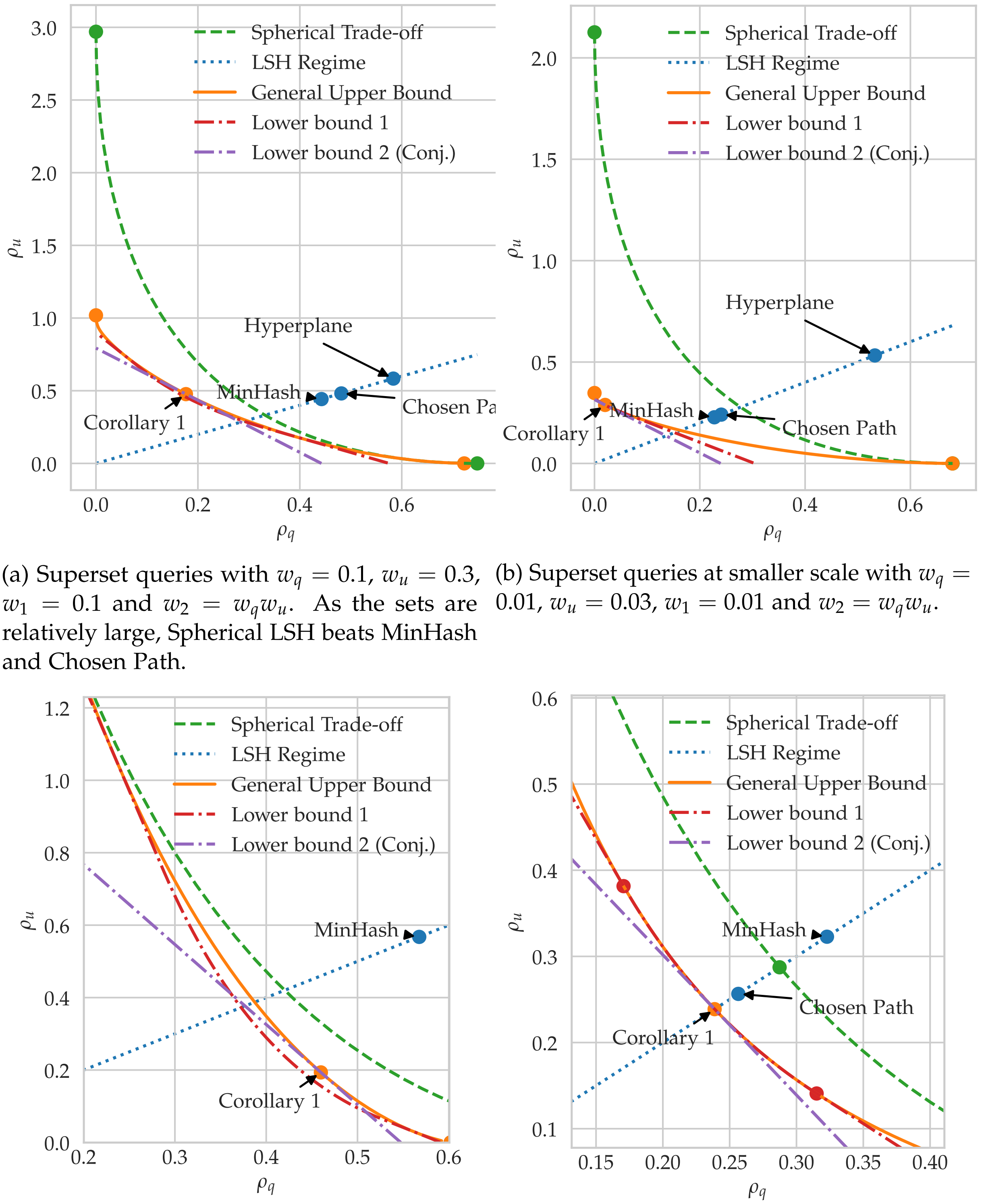
We show the existence of a data-structure for (w_q, w_u, w_1, w_2)-GapSS with space usage $n^{1+\rho_u+o(1)} + O(n\, w_u|U|)$ and query time $n^{\rho_q+o(1)}$ for some ρ_q and ρ_u which are functions of w_q, w_u, w_1, w_2 as well as $t_u, t_q \in [0, 1]$ as described in the introduction.

Our main upper bound, theorem 5, is a bit to intricate to be stated yet, but we note the following corollaries. Each of them follows easily by inserting the giving values into theorem 5.

Corollary 1 (Near balanced). *If we set $t_q = 1 - w_u, t_2 = u - w_q$ we get the near-balanced values:*

$$\rho_q = \frac{H(w_1) - D(w_u, 1 - w_q)}{H(w_2) - D(w_u, 1 - w_q)}, \quad \rho_u = \frac{H(w_1) - D(w_q, 1 - w_u)}{H(w_2) - D(w_u, 1 - w_q)}. \tag{2.2}$$

where $H(w_i) = (1 - w_q - w_u) \log(\frac{1-w_q-w_u+w_i}{w_i})$, and $D(a, b) = a \log \frac{a}{b} + (1 - a) \log \frac{1-a}{1-b}$ is the Kullback–Leibler divergence. (We define $D(0, b) = \log 1/(1 - b)$, $D(1, b) = \log 1/b$ as is standard.)



(a) Superset queries with $w_q = 0.1$, $w_u = 0.3$, $w_1 = 0.1$ and $w_2 = w_q w_u$. As the sets are relatively large, Spherical LSH beats MinHash and Chosen Path.

(b) Superset queries at smaller scale with $w_q = 0.01$, $w_u = 0.03$, $w_1 = 0.01$ and $w_2 = w_q w_u$.

(c) $w_q = 0.4$, $w_u = 0.1$ $w_1 = 0.1$ $w_2 = w_q w_u$. We see that theorem 2 is not tight when $w_q \neq w_u$. However the conjectured Lower bound 2 matches the upper bound exactly.

(d) $w_q = w_u = 0.16$, $w_1 = 0.1$, $w_2 = w_q w_u$. The red dots show the segment where the r, s can be chosen optimally without exceeding $1/2$.

Figure 2.1: Examples of ρ-values obtained from theorem 5 for various parameter settings, compared to that of other algorithms, and to the lower bounds theorem 2 and conjecture 1.

If $w_q = w_u = w$ this simplifies to:

$$\rho_q = \rho_u = \log\left(\frac{w}{w_1}\frac{1-2w+w_1}{1-w}\right) \Big/ \log\left(\frac{w}{w_2}\frac{1-2w+w_2}{1-w}\right) = \frac{\log S(q,x)}{\log S(q,x')}, \tag{2.3}$$

where $S(q,x) = \frac{\langle q,x\rangle}{\|q\|_2\|x\|_2} / \frac{\langle \bar{q},\bar{x}\rangle}{\|\bar{q}\|_2\|\bar{x}\|_2}$ is the cosine similarity divided by the cosine similarity on the complement sets.

In the case of small sets, $w, w_1, w_2 \to 0$, equation (2.3) reduces to $\log(\frac{w}{w_1}) / \log(\frac{w}{w_2})$, which is the ρ-value of Chosen Path, and was shown in [63] to be optimal in this range. In the next section we generalize their lower bound to hold generally for all values w, w_1, w_2 though still only asymptotically sharp for small sets.

Corollary 1 is special, because the optimal values of t_u and t_q depend only on w_u and w_q, while in general it will also depend on w_1 and w_2. We will show (2.3) is optimal for the case $w_2 = w_q^2$ for all choices of w_q and w_1. Conditioned on a conjectured hypercontractive inequality we will even show eq. (2.2) is optimal for all choices of w_q, w_u and w_1 at the particular trade-off.

Corollary 2 (Subset/superset queries). *If $w_1 = \min\{w_u, w_q\}$, $w_2 = w_u w_q$ we can take*

$$\begin{aligned} t_q &= -\frac{w_u(1-w_u)w_q(1-w_q)}{w_q - w_u}\alpha + \frac{w_q(1-w_u)}{w_q - w_u} \\ \textit{and } t_u &= \frac{1}{w_q - w_u}\alpha^{-1} - \frac{w_u(1-w_q)}{w_q - w_u} \\ \textit{for any } \alpha &\in \left[\min\{w_u, w_q\} - w_q w_u, \max\{w_u, w_q\} - w_q w_u\right] \end{aligned}$$

to get data structures with

$$\rho_u = \frac{t_q \log\frac{1-t_u}{1-w_u} - t_u \log\frac{1-t_q}{1-w_q}}{D(t_u, w_u)} \textit{ and } \rho_u = \frac{(1-t_u)\log\frac{t_q}{w_q} - (1-t_q)\log\frac{t_u}{w_u}}{D(t_u, w_u)}.$$

As it turns out, the optimal values for t_u and t_q, when doing superset/subset queries lie on the diagonal hyperbola $t_u w_q(1-w_u) - t_q(1-w_q)w_u = t_u t_q(w_q - w_u)$ with $t_u, t_q = 0$ in one end and $t_q, t_u = 1$ in the other. This means that Chosen Path (without symmetrization) is indeed equivalent to our approach for these problems, when we are interested in near linear space or near constant query time.

Lower bounds

For the lower bounds we will assume $|U| = \omega(\log n)$ (like we reduce to in the upper bounds). This follows all previous LSH-lower bounds, and indeed it is known from [43] that it is possible to do better in the "medium dimension regime" when $|U| = O(\log n)$. In that regime classical data structures such as KD-trees are also competitive, see e.g. [49].

The lower bounds are all in the specific model of Locality Sensitive Filters (definition 4). In other words, the data structure is presumed to be as described in the introduction, and the only part we are allowed to change is how the F functions from equation (2.1) are defined. There is a stronger kind of LSH, in which the filter distribution is allowed to depend on the dataset [22, 31, 25] which does better than data-independent LSF in general. However most of our lower bounds are for the 'random case' $w_2 = w_q w_u$ in which no difference is known between the two approaches.

About the notation in the lower bounds: When we write "$\rho_q \geq A$ and $\rho_u \geq B$" it will mean that it is not possible to make one smaller while not increasing the other. Since there is always $\rho_q = 0$ and $\rho_u = 0$ somewhere on the trade-off, it doesn't make sense to bound one value in isolation.

Theorem 2 (Lower bound 1)**.** *Given $\alpha \geq 0$ and $0 \leq r, s \leq 1/2$, let $u_q = \log \frac{1-w_q}{w_q}$ and $u_u = \log \frac{1-w_u}{w_u}$. If r and s are such that $\frac{\sinh(u_q r)}{\sinh(u_q(1-r))} \frac{\sinh(u_u s)}{\sinh(u_u(1-s))} = \left(\frac{w_1 - w_q w_u}{w_q(1-w_q) w_u(1-w_u)}\right)^2$, then any LSF data structure must have*

$$\rho_q \geq 1 - s - \alpha r \quad \text{and} \quad \rho_u \geq \alpha - s - \alpha r.$$

To get the most out of the lower bound, we will also want r and s such that

$$\frac{u_q \sinh(u_q) \sinh(u_u s) \sinh(u_u(1-s))}{u_u \sinh(u_u) \sinh(u_q r) \sinh(u_q(1-r))} = \alpha,$$

however due to the limitation $r, s \leq 1/2$, this is not always possible.[4]

For $w_u = w_q$ we can take $\alpha = 1$ and $r = s$ to get $\rho_q, \rho_u \geq \log \frac{w_q}{w_1} \frac{1-2w_q+w_1}{1-w_q} / \log \frac{1-w_q}{w_q}$ which exactly matches corollary 1 and shows it is optimal for all w_1 and w_q when $w_u = w_q$, $w_2 = w_q^2$. When $w_q \neq w_u$ the bound is unfortunately not sharp, as can be seen in fig. 2.1.

Theorem 2 is based on the p-biased hypercontractive inequalities of Oleszkiewicz and Krzysztof [143]. Their inequality, while sharp, only handles the case of a single boolean function, and we have expanded it using Cauchy Schwartz to get our more general theorem. This approach, while good enough for sharp space/time trade-offs on the sphere, turns out to be insufficient for sets when $w_q \neq w_u$.

To overcome this, we conjecture a new two-function p-biased hypercontractive inequality, which we unfortunately have not able to prove yet, but for which we have much evidence (see the lower bounds section). This inequality implies the following lower bound:

Conjecture 1 (Lower bound 2)**.** *Let $r = \log \frac{(1-w_q)(1-w_u)}{w_q w_u} / \log \frac{1-w_q-w_u+w}{w}$ and $\alpha \geq 0$ then any LSF data structure must have*

$$\rho_q \geq (\alpha+1)/r - \alpha \quad \text{and} \quad \rho_u \geq (\alpha+1)/r - 1.$$

Setting $\alpha = 1$ this immediately corollary 1 is tight for all w_q, w_u, w_1 and $w_2 = w_q w_u$. We believe it should be possible to extend this further to separate r and s values, as in theorem 2, which would show theorem 5 to be tight for all w_q, w_u, w_1 and the entire time/space trade-offs, but this is work for the future.

Our previous lower bounds have assumed $w_2 = w_q w_u$. To extend to general values of w_2 we show the follow bound:

[4] This is not just an artefact of the proof, since computations of theorem 2 with r, s outside the given range shows that the theorem as stated is indeed false in that case, as it goes above our upper bound. The limitation might be removed by using the more general p-biased inequalities by Wolff [187], but unfortunately those are for assymptotically small sets.

Theorem 3 (Lower bound 3). *If $w_q = w_u$, any LSF data structure that uses the same functions for updates and queries ($F_u^{(i)} = F_q^{(i)}$) must have then any LSF data structure must have*

$$\rho_u, \rho_q \geq \log\left(\frac{w_1 - w_q^2}{w_q(1-w_q)}\right) \Big/ \log\left(\frac{w_2 - w_q^2}{w_q(1-w_q)}\right).$$

Taking $w, w_1, w_2 \to 0$ this recovers Pagh and Christiani's $\rho \geq \log(w_1/w)/\log(w_2/w)$ bound for Braun Blanquet similarity [63].[5] For larger values of w_q, w_1, w_2 the bound is however not tight. Showing any lower bound that holds for $w_2 \neq w_u w_q$ and for large distances is an open problem in the LSH world.

Comparison to previous approaches

Since our lower bounds don't cover the entire range of parameters w_q, w_u, w_1, w_2 (no LSH lower bounds do), we need to compare our ρ values with those achieved by previous methods and show that we get lower values on the entire range.

We show two results towards this: (1) For Spherical LSH we show how to most optimally embed GapSS onto the sphere, and that our ρ values are at least as small as with Spherical LSH in this setting. (2) For MinHash we show dominating family of Chosen Path like algorithms, which it is natural to conjecture is again dominated by supermajorities. The first result is quite interesting on its own right, since many previous algorithms for Maximum Inner Product Search as consisted of various embeddings onto the sphere. The second result is also interesting in that it sheds more light on why MinHash is sometimes faster than Chosen Path, which is a question raised in [63], and shows that small changes to Chosen Path could indeed change this.

Lemma 2.1.1 (Best Binary Embedding). *Let $g, h : \{0,1\}^d \to \mathbb{R}$ be function on the form $g(1) = a_1 x + b_1$ and $h(y) = a_2 y + b_2$. Let $\rho(x, y, y') = f(\alpha(x,y))/f(\alpha(x,y'))$ where $\alpha(x,y) = \langle x, y\rangle / \|x\| \|y\|$ be such that*

$$f(z) \geq 0, \quad \tfrac{d}{dz}\left((\pm 1 - z)\tfrac{d}{dz}\log f(z)\right) \geq 0 \quad \text{and} \quad \tfrac{d^3}{dz^3}\log f(z) \leq 0$$

for all $z \in [-1,1]$. Assume we know that $\|x\|_2^2 = w_q d$, $\|y\|_2^2 = w_u d$, $\langle x, y'\rangle = w_1 d$ and $\langle x, y\rangle = w_2 d$, then

$$\arg\min_{a_1,b_1,a_2,b_2} \rho(g(x), h(y), h(y')) = (1, -w_q, 1, -w_u).$$

See proof in section 2.3.2. Since α, as defined above, scales the vectors down by their norm to make sure they are on the sphere, the lemma indeed says that we should subtract the mean and divide by the standard deviation of our vectors before we use LSH. We show that Spherical LSH and Hyperplane LSH [52] (a.k.a. SimHash) satisfy this lemma, given their ρ values for distinguishing between inner products $\alpha > \beta$:

$$\rho_{\text{hp}} = \frac{\log(1 - \arccos(\alpha)/\pi)}{\log(1 - \arccos(\beta)/\pi)}, \quad \rho_{\text{sp}} = \frac{1-\alpha}{1+\alpha}\frac{1+\beta}{1-\beta}.$$

This implies we should take $\alpha = \frac{w_1 - w_q w_u}{\sqrt{w_q(1-w_q)w_u(1-w_u)}}$ and $\beta = \frac{w_2 - w_q w_u}{\sqrt{w_q(1-w_q)w_u(1-w_u)}}$.

See also fig. 2.1 where we have plotted theorem 5 against Chosen Path, MinHash, Spherical LSF and Hyperplane LSH.

[5] Their bound also implicitly had the same function for queries and updates.

Comparison to MinHash Consider the LSF family, $\mathcal{F}$, formed by one of the functions

$$F_0(x) = [s_0 \in x], F_1(x) \quad = [s_1 \in x \wedge s_0 \notin x], \ldots, F_i(x) = [s_i \in x \wedge s_0 \notin x \wedge \cdots \wedge s_{i-1} \notin x], \ldots$$

where $(s_i \in U)_{i \in \mathbb{N}}$ is a random sequence by sampling elements of U with replacement. Note that while the sequence is infinite, it the functions eventually all become 0 as we get a prefix including all of U, hence we can sample from $\mathcal{F}$ efficiently. Also note that then $h(x) = \min\{i \mid F_i(x) = 1\}$ is the usual MinHash function.

While MinHash is balanced, $\rho_u = \rho_q$, most of the F_i's are on their own not balanced if $w_q \neq w_u$. We can fix this by applying a symmetrization technique implicit in [63]. Using that we get

$$\rho_i = \log \frac{(1 - w_q - w_u + w_1)^i w_1}{\max\{(1 - w_q)^i w_q,\, (1 - w_u)^i w_u\}} \Big/ \log \frac{(1 - w_q - w_u + w_2)^i w_2}{\max\{(1 - w_q)^i w_q,\, (1 - w_u)^i w_u\}}$$

for the LSF data structure using only F_i. Note that $\rho_0 = \log \frac{w_1}{\max\{w_q, w_u\}} / \log \frac{w_2}{\max\{w_q, w_u\}}$ is exactly the same as ρ_{cp} achieved by Chosen Path. This makes sense, since it is exactly the Chosen Path function with the Chosen Path symmetrization technique.

We show that in section 2.3.3 that $\rho_{\text{mh}} = \log \frac{w_1}{w_q + w_u - w_1} / \log \frac{w_2}{w_q + w_u - w_2} \geq \min_{i \geq 0} \rho_i$. In fact we can restrict this to $i \in \{0, \infty, \log(w_q/w_u)/\log((1 - w_q)/(1 - w_u))\}$, where the first gives Chosen Path, the second gives Chosen Path on the complemented sets, and the last gives two concatenated Chosen Path's in a balanced trade-off where $(1 - w_q)^i w_q = (1 - w_u)^i w_u$.

2.2 Preliminaries

The Locality Sensitive Filter approach to similarity search is an extension by Becker et al. [43] to the Locality Sensitive Hashing framework by Indyk and Motwani [95]. We will use the following definition by Christiani [61], which we have slightly extended to support separate universes for query and data points:

Definition 4 (LSF). *Let X and Y be some universes, let $S : X \times Y \to \mathbb{R}$ be a similarity function, and let $\mathcal{F}$ be a probability distribution over $\{(Q, U) \mid Q \subseteq X, U \subseteq Y\}$. We say that F is $(s_1, s_2, p_1, p_2, p_q, p_u)$-sensitive if for all points $x \in X, y \in Y$ and (Q, U) sampled randomly from $\mathcal{F}$ the following holds:*

1. *If $S(x, y) \geq s_1$ then* $\Pr[x \in Q, y \in U] \geq p_1$.

2. *If $S(x, y) \leq s_2$ then* $\Pr[x \in Q, y \in U] \leq p_2$.

3. $\Pr[x \in Q] \leq p_q$ *and* $\Pr[x \in U] \leq p_u$.

We refer to (Q, U) as a filter and to Q as the query filter and U as the update filter.

The main theorem from [61] which we will use for our upper bounds is (paraphrasing):

Theorem 4 (LSF theorem)**.** *Suppose we have access to a family of filters that is* $(s_1, s_2,\ p_1, p_2, p_q, p_u)$*-sensitive. Then we can construct a fully dynamic data structure that solves the* (s_1, s_2)*-similarity search problem with query time* $dn^{\rho_q+o(1)}$*, update time* $dn^{\rho_u+o(1)}$*, and space usage* $dn + n^{1+\rho_u+o(1)}$ *where* $\rho_q = \log(p_q/p_1)/\log(p_q/p_2)$ *and* $\rho_u = \log(p_u/p_1)/\log(p_q/p_2)$.

We must be able to sample, store, and evaluate filters from $\mathcal{F}$ *in time* $dn^{o(1)}$*.*

We will use definition 4 with $S(x, y) = |x \cap y|$. For given values of w_q and w_u, the (s_1, s_2)-similarity search problem then corresponds to the (w_u, w_q, w_1, w_2)-gap similarity search problem.

2.3 Upper bounds

To state our results we first need to define the following functions:

Definition 5 (Entropy Functions)**.** *The relative entropy function (or Kullback–Leibler divergence) is defined for* $a, b \in [0, 1]$ *by:* $D(a, b) = a \log \frac{a}{b} + (1 - a) \log \frac{1-a}{1-b}$ *and* $D(0, b) = \log 1/(1 - b)$*,* $D(1, b) = \log 1/b$*.*

For $x, y \in [0, 1]$*,* $t_1, t_2 \in [0, 1]$ *and* $b \in [0, \min(x, y)]$ *we define the following pair-relative entropy function:*

$$\Lambda(w_q, w_u, w, t_q, t_u) = t_1\lambda_1 + t_2\lambda_2 - \log v \quad \textit{where}$$

$$\lambda_1 = \log\left(\frac{v(1 - t_2) - (1 - x - y + b)}{x - b}\right),\ \lambda_2 = \log\left(\frac{v(1 - t_1) - (1 - x - y + b)}{y - b}\right),$$

$$v = \frac{v_1 + \sqrt{v_1^2 - v_2}}{2b(1 - t_1)(1 - t_2)},$$

and $v_1 = (1 - t_1 - t_2)(b - xy) + b(1 - x - y + b)$*,* $v_2 = 4(1 - t_1)(1 - t_2)(b - xy)b(1 - x - y + b)$*. At* $t_1 = 1$*,* $t_2 = 1$ *or* $b = \min(x, y)$ *it is defined by its limit.*

The goal of this section is to prove the following general upper bound:

Theorem 5 (General Upper Bound)**.** *For any choice of constants* $w_q, w_u \geq w_1 \geq w_2 \geq 0$ *and* $1 \geq t_q, t_u \geq 0$ *we can solve the* (w_q, w_u, w_1, w_2)*-GapSS problem over universe* U *with query time* $n^{\rho_q+o(1)}$ *and space usage* $n^{\rho_u+o(1)} + O(n\, w_u|U|)$*, where*

$$\rho_q = \frac{\Lambda(w_q, w_u, w_1, t_q, t_u) - D(t_q, w_q)}{\Lambda(w_q, w_u, w_2, t_q, t_u) - D(t_q, w_q)}, \quad \rho_u = \frac{\Lambda(w_q, w_u, w_1, t_q, t_u) - D(t_u, w_u)}{\Lambda(w_q, w_u, w_2, t_q, t_u) - D(t_q, w_q)}.$$

The theorem defines the entire space/time trade-off of fig. 2.1 by choices of t_q and t_u. By Lagrangian multipliers we can compute the optimal t_u for any t_q. An easy corollary is

Corollary 3 (Linear space / constant time)**.**

$$\begin{aligned} &\textit{If} \quad t_q\, w_y(1 - w_y) + w_1 w_y = t_u\,(w_1 - w_x w_y) + w_x w_y && \textit{then} \quad \rho_u = 0. \\ &\textit{If} \quad t_u w_x(1 - w_x) + w_1 w_x = t_q(w_1 - w_x w_y) + w_y w_x && \textit{then} \quad \rho_q = 0. \end{aligned}$$

We will use the LSF theorem 4 as the basis for our upper bound with the filter family described in the introduction. Note that we can reduce the universe to $O(\epsilon^{-2}w_2^{-1}\log n)$ by sampling. By a union bound this preserves w_1, w_2, w_q and w_y within a factor $1 \pm \epsilon$. Taking $\epsilon = 1/\log n$ this is absorbed into the $n^{o(1)}$ factor in our bounds. If $|U|$ is too small, we can simply replicate the elements to ensure $|U| = \omega(\log n)$.

We restate the filter family: The functions are constructed by sampling a random subset $S \subseteq U, |S| = \omega(\log n)$ with replacement, and picking two thresholds, $1 \geq t_u \geq 0, 1 \geq t_q \geq 0$. Then

$$F_u(y) = \begin{cases} [|y \cap S| \geq t_u|S|] & \text{if } t_u \geq w_u \\ [|y \cap S| \leq t_u|S|] & \text{if } t_u < w_u \end{cases} \quad \text{and} \quad F_q(x) = \begin{cases} [|x \cap S| \geq t_q|S|] & \text{if } t_q \geq w_q \\ [|x \cap S| \leq t_q|S|] & \text{if } t_q < w_q \end{cases} .$$

To use theorem 4 we then need to compute $p_u = \Pr[F_u(y) = 1]$, $p_q = \Pr[F_q(x) = 1]$ and $p_1 = \Pr[F_u(y) = 1 \wedge F_q(x) = 1]$. The two first follow from the standard Entropy Chernoff bound: $\log p_u = -D(t_u, w_u)|S|(1 + o(1))$ and $\log p_q = -D(t_q, w_q)|S|(1 + o(1))$. where D is from definition 5. Note that this form of the Chernoff bound holds in the $t_u \geq w_u$ case as well as the $t_u < w_u$ case.

The joined probability p_1 (and p_2) is more tricky. Note that we need a bound which is tight up to a factor $1 + o(1)$ in the exponent. We will do this using the Large Deviations theorem by Gartner Ellis (see below) on the sequence $\{X_i\}_{i \in [|S|]} \subseteq \{0,1\}^2$ of outcomes when sampling S, where $X_{i,1} = F_q^{(i)}(x)$ and $X_{i,2} = F_u^{(i)}(x)$. This has joint Bernoulli distribution $\sim \left[\begin{smallmatrix} w_1 & w_u - w_1 \\ w_q - w_1 & 1 - w_q - w_u + w_1 \end{smallmatrix}\right]$ or concretely: $\Pr[X_i = (1,1)] = w_1$, $\Pr[X_i = (1,0)] = w_q - w_1$, $\Pr[X_i = (0,1)] = w_u - w_1$ and $\Pr[X_i = (0,0)] = 1 - w_q - w_u + w_1$.

Using Gartner Ellis we will show the following lemma, from which theorem 5 follows:

Lemma 2.3.1. *there is a $(w_x, w_y, w_1, w_2, p_1, p_2, p_q, p_u)$-sensitive filter, where*

$$\begin{aligned} |S|^{-1}\log 1/p_1 &= \Lambda(w_q, w_u, w_1, t_q, t_u) + o(1), \\ |S|^{-1}\log 1/p_2 &= \Lambda(w_q, w_u, w_2, t_q, t_u) + o(1), \\ |S|^{-1}\log 1/p_q &= D(t_q, w_q) + o(1), \\ |S|^{-1}\log 1/p_u &= D(t_u, w_u) + o(1). \end{aligned}$$

2.3.1 Large Deviations

Theorem 6 (Gartner-Ellis theorem [DZ10, Theorem 2.3.6 and Corollary 6.1.6])**.** *Let $\{X_i\}_{i \in \mathbb{N}} \subseteq \mathbb{R}^k$ be a sequence of iid. random vectors. Let $S_n = \frac{1}{n}\sum_{i=0}^{n} X_i$ be the empirical means. Define the logarithmic generating function $\Lambda(\lambda) = \log E \exp\langle \lambda, X_1 \rangle$, and the rate function $\Lambda^*(z) = \sup_{\lambda \in \mathbb{R}^k}\{\langle \lambda, z \rangle - \Lambda(\lambda)\}$. If $\Lambda(\varepsilon) < \infty$ for all $\varepsilon \in \mathbb{R}^k$ with $\|\epsilon\|_2 < \delta$ for some $\delta > 0$ small enough, then for any set $F \subseteq \mathbb{R}^k$:*

$$\lim_{n \to \infty} \log \Pr\left[S_n \in F\right] = -\inf_{z \in F} \Lambda^*(z).$$

From this we can derive the more simple:

Lemma 2.3.2 (Multi Dimensional Cramer). *Let $X_i \in \mathbb{R}^k$ be a sequence of iid. random variables, and let $t \in \mathbb{R}^k$ be a list of values such that $E[X_1] \le t \le \max X_1$. Let $\Lambda(\lambda) = \log E[\exp\langle X_1, \lambda\rangle]$ be finite for all $\lambda \in \mathbb{R}^k$ then*

$$\frac{1}{n} \log \Pr\left[\frac{1}{n}\sum_{i=1}^{n} X_i \ge t\right] = -\Lambda^*(t) + o(1)$$

where $\Lambda^*(t) = \langle t, \lambda\rangle - \Lambda(\lambda)$ and $\nabla\Lambda(\lambda) = t$.

Proof. We use the Gartner-Ellis theorem. Since we assume $\Lambda(z)$ is finite everywhere, it is also so an epsilon ball around 0. Next note that $\Lambda(\lambda)$ is convex so $\langle \lambda, z\rangle - \Lambda(\lambda)$ is maximized at $\nabla\Lambda(\lambda) = z$.

We need to show $\inf_{z \ge t} \Lambda^*(z) = \Lambda^*(t)$. Let $\mu = E[X_1]$.

Note $\frac{d\Lambda}{d\lambda_i}(0) = \mu_i$ (since Λ is a mgf), thus if $z_i = \mu_i$ then $\lambda_i = 0$ and so $\frac{d\Lambda^*}{dz_i}(\mu_i) = 0$. From this, and the convexity of Λ^* we get $\langle \nabla\Lambda^*(z), z - \mu\rangle \ge 0$, and so for any point in $\{z \ge t\}$ we can always decrease $\Lambda^*(z)$ be moving towards μ, showing that the minimum is achieved at $z = t$. □

See also the details in the appendix and in [75]. Finally we can get the specific version we need:

Lemma 2.3.3. *Let $X_i \in \{0,1\}^2$, $i \in [m]$, have joint probability distribution $P \in [0,1]^{2\times 2}$ such that $\Pr[X_i = (a,b)] = P_{a,b}$. Let $w = P_{1,1}$, $\mu_1 = P_{1,1} + P_{1,2}$ and $\mu_2 = P_{1,1} + P_{2,1}$, if $\mu_1 < t_1 < 1, \mu_2 < t_2 < 1$, then*

$$\frac{1}{m} \log \Pr\left[\tfrac{1}{m}\sum_{i=1}^{n} X_{i,1} \ge t_1 \wedge \tfrac{1}{m}\sum_{i=1}^{n} X_{i,2} \ge t_2\right] = -\Lambda(\mu_1, \mu_2, w, t_1, t_2) + o(1),$$

(where Λ is define in definition 5.) If $t_1 < \mu_1$ then ($\ge$) above is replaced by $\le$ and similarly for $t_2 < \mu_2$.

Proof. This follows directly by lemma 2.3.2, when we plug-in λ_1, λ_2 to check that indeed

$$t = \nabla\Lambda(\lambda) = \frac{1}{be^{\lambda_1+\lambda_2} + (x-b)e^{\lambda_1} + (y-b)e^{\lambda_2} + (1-x-y+b)} \begin{bmatrix} be^{\lambda_1+\lambda_2} + (x-b)e^{\lambda_1} \\ be^{\lambda_1+\lambda_2} + (y-b)e^{\lambda_2} \end{bmatrix}.$$

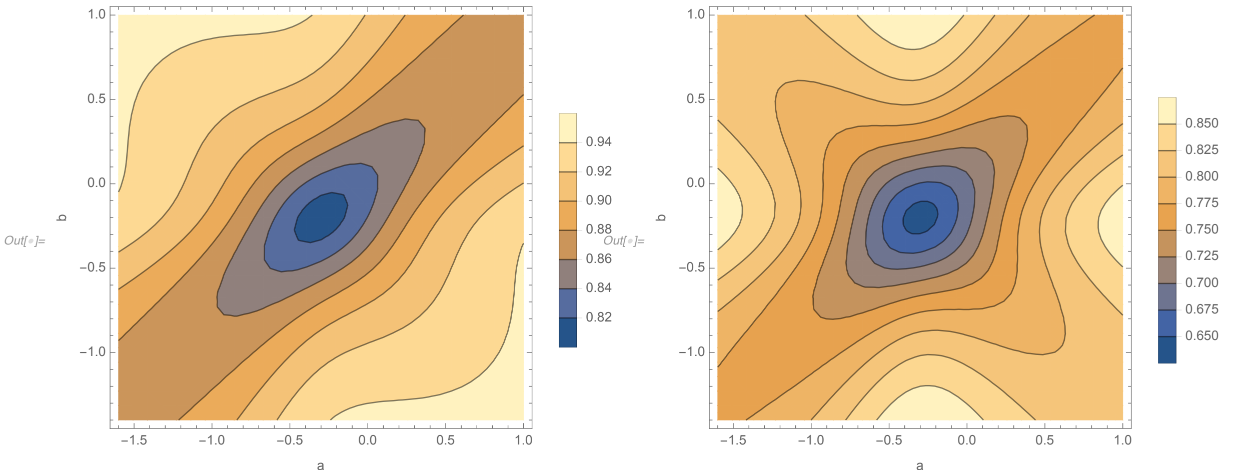
□

This proves theorem 5.

2.3.2 Embedding onto the Sphere

Recall lemma 2.1.1: Let $g, h : \{0,1\}^d \to \mathbb{R}$ be function on the form $g(1) = a_1 x + b_1$ and $h(y) = a_2 y + b_2$. Let $\rho(x, y, y') = f(\alpha(x,y))/f(\alpha(x,y'))$ where $\alpha(x,y) = \langle x, y\rangle / \|x\|\|y\|$ be such that

$$f(z) \ge 0, \quad \tfrac{d}{dz}\Big((\pm 1 - z)\tfrac{d}{dz}\log f(z)\Big) \ge 0 \quad \text{and} \quad \tfrac{d^3}{dz^3}\log f(z) \le 0$$



(a) ρ-hyperplane (b) ρ-spherical

Figure 2.2: ρ-values for hyperplane and spherical LSH under different shifts.

for all $z \in [-1,1]$. Assume we know that $\|x\|_2^2 = w_q d$, $\|y\|_2^2 = w_u d$, $\langle x, y' \rangle = w_1 d$ and $\langle x, y \rangle = w_2 d$, then $\arg\min_{a_1,b_1,a_2,b_2} \rho(g(x), h(y), h(y')) = (1, -w_q, 1, -w_u)$.

In this section we will show that Hyperplane [52] and Spherical [26] LSH both satisfy the requirements of the lemma. Hence we get two algorithms with ρ-values:

$$\rho_{\text{hp}} = \frac{\log(1 - \arccos(\alpha)/\pi)}{\log(1 - \arccos(\beta)/\pi)}, \quad \rho_{\text{sp}} = \frac{1-\alpha}{1+\alpha}\frac{1+\beta}{1-\beta}.$$

where $\alpha = \frac{w_1 - w_x w_y}{\sqrt{w_x(1-w_x)w_y(1-w_y)}}$ and $\beta = \frac{w_2 - w_x w_y}{\sqrt{w_x(1-w_x)w_y(1-w_y)}}$, and space/time trade-offs using the ρ_q, ρ_u values in [61]. [6] Figure 2.2 shows how ρ varies with different translations a, b.

Taking $t_q = w_q(1 + o(1))$ and $t_u = w_u(1 + o(1))$ in theorem 5 recovers ρ_{sp} by standard arguments. This implies that theorem 5 dominates Spherical LSH (for binary data).

Lemma 2.3.4. *The functions $f(z) = (1-z)/(1+z)$ for Spherical LSH and $f(z) = -\log(1 - \arccos(z)/\pi)$ for Hyperplane LSH satisfy lemma 2.1.1.*

Proof. For Spherical LSH we have $f(z) = (1-z)/(1+z)$ and get

$$\frac{d}{dz}\left((\pm 1 - z)\frac{d}{dz}\log f(z)\right) = 2(1 \mp 2z + z^2)/(1-z^2)^2 \geq 0,$$
$$\frac{d^3}{dz^3}\log f(z) = -4(1+3z^2)/(1-z^2)^3 \leq 0.$$

For Hyperplane LSH we have $f(z) = -\log(1 - \arccos(z)/\pi)$ and get

$$\frac{d}{dz}\left((1-z)\frac{d}{dz}\log f(z)\right) = \frac{(\arccos(z) - \sqrt{1-z^2} - \pi)\log(1 - \arccos(z)/\pi) - \sqrt{1-z^2}}{(1+z)\sqrt{1-z^2}(\pi - \arccos(z))^2 \log(1 - \arccos(z)/\pi)^2},$$
$$\frac{d}{dz}\left((-1-z)\frac{d}{dz}\log f(z)\right) = \frac{(\arccos(z) + \sqrt{1-z^2} - \pi)\log(1 - \arccos(z)/\pi) + \sqrt{1-z^2}}{(1-z)\sqrt{1-z^2}(\pi - \arccos(z))^2 \log(1 - \arccos(z)/\pi)^2}.$$

[6] Unfortunately the space/time aren't on a form applicable to lemma 2.1.1. From numerical experiments we however still conjecture that the embedding is optimal for those as well.

In both cases the denominator is positive, and the numerator can be shown to be likewise by applying the inequalities $\sqrt{1-z^2} \le \arccos(z)$, $\sqrt{1-z^2} + \arccos(z) \le \pi$ and $x \le \log(1+x)$.

The $\frac{d^3}{dz^3} \log f(z) \le 0$ requirement is a bit trickier, but a numerical optimization shows that it's in fact less than -1.53. □

Finally we prove the embedding lemma:

Proof of lemma 2.1.1. We have

$$\alpha = \frac{\langle x+a, y+b\rangle}{\|x+a\|\|y+b\|} = \frac{w_1 + w_x b + w_y a + ab}{\sqrt{(w_x(1+a)^2 + (1-w_x)a^2)(w_y(1+b)^2 + (1-w_y)b^2)}}$$

and equivalent with w_2 for β. We'd like to show that $a = -w_x$, $b = -w_y$ is a minimum for $\rho = \frac{1-\alpha}{1+\alpha}\frac{1-\beta}{1+\beta}$.

Unfortunately ρ is not convex, so it is not even clear that there is just one minimum. To proceed, we make the following substitution $a \to (c+d)\sqrt{w_x(1-w_x)} - w_x$, $b \to (c-d)\sqrt{w_y(1-w_y)} - w_y$ to get

$$\alpha(c,d) = \frac{cd + \frac{w_1 - w_x w_y}{\sqrt{w_x(1-w_x)w_y(1-w_y)}}}{\sqrt{(1+c^2)(1+d^2)}}.$$

We can further substitute $cd \mapsto rs$ and $\sqrt{(1+c^2)(1+d^2)} \mapsto r+1$ or $r \ge 0$, $-1 \le s \le 1$, since $1 + cd \le \sqrt{(1+c^2)(1+d^2)}$ by Cauchy Schwartz, and $(cd, \sqrt{(1+c^2)(1+d^2)})$ can take all values in this region.

The goal is now to show that $h = f\left(\frac{rs+x}{r+1}\right) / f\left(\frac{rs+y}{r+1}\right)$, where $1 \ge x \ge y \ge -1$, is increasing in r. This will imply that the optimal value for c and d is 0, which further implies that $a = -w_x$, $b = -w_y$ for the lemma.

We first show that h is quasi-concave in s, so we may limit ourselves to $s = \pm 1$. Note that $\log h = \log f\left(\frac{rs+x}{r+1}\right) - \log f\left(\frac{rs+y}{r+1}\right)$, and that $\frac{d^2}{ds^2} \log f\left(\frac{rs+x}{r+1}\right) = \left(\frac{r}{1+r}\right)^2 \frac{d^2}{dz^2} \log f(z)$ by the chain rule. Hence it follows from the assumptions that h is log-concave, which implies quasi-concavity as needed.

We now consider $s = \pm 1$ to be a constant. We need to show that $\frac{d}{dr} h \ge 0$. Calculating,

$$\frac{d}{dr} f\left(\frac{rs+x}{r+1}\right) / f\left(\frac{rs+y}{r+1}\right) = \frac{(s-x) f\left(\frac{rs+y}{r+1}\right) f'\left(\frac{rs+x}{r+1}\right) - (s-y) f\left(\frac{rs+x}{r+1}\right) f'\left(\frac{rs+y}{r+1}\right)}{(1+r)^2 f\left(\frac{rs+y}{r+1}\right)^2}.$$

Since $f \ge 0$ it suffices to show $\frac{d}{dx}(s-x) f'\left(\frac{rs+x}{r+1}\right) / f\left(\frac{rs+x}{r+1}\right) \ge 0$. If we substitute $z = \frac{rs+x}{r+1}$, $z \in [-1,1]$, we can write the requirement as $\frac{d}{dz}(s-z) f'(z)/f(z) \ge 0$ or $\frac{d}{dz}\left((\pm 1 - z)\frac{d}{dz}\log f(z)\right) \ge 0$. □

2.3.3 A MinHash dominating family

We complete the arguments from section 2.1.2.

We first state the LSF-Symmetrization lemma implicit in [63]:

Lemma 2.3.5 (LSF-Symmetrization)**.** *Given a (p_1, p_2, p_q, p_u)-sensitive LSF-family, we can create a new family that is $(p_1 q/p, p_2 q/p, q, q)$-sensitive, where $p = \max\{p_q, p_u\}$ and $q = \min\{p_q, p_u\}$.*

For some values of p_1, p_2, p_q, p_u this will be better than simply taking $\max(\rho_u, \rho_q)$. In particular when symmetrization may reduce ρ_u by a lot by reducing its denominator.

Proof. W.l.o.g. assume $p_q \geq p_u$. When sampling a query filter, $Q \subseteq U$, pick a random number $\varrho \in [0,1]$. If $\varrho > p_u/p_q$ use $\emptyset$ instead of Q. The new family then has $p'_q = p_q \cdot p_u/p_q$ and so on giving the lemma. □

Using this lemma it is easy to make a version of supermajority LSF that always beats Chosen Path: Simply take $t_q = t_u = 1$ and apply lemma 2.3.5. Then we have exactly the same ρ value as Chosen Path. We do however conjecture that symmetrization is not nessecary for supermajorities, since we have another (presumably more efficient) form of symmetrization via assymetric $t_u \neq t_q$.

Now recall the filter family from the introduction:

$$F_0(x) = [s_0 \in x], F_1(x) \ = [s_1 \in x \wedge s_0 \notin x], \dots, F_i(x) = [s_i \in x \wedge s_0 \notin x \wedge \cdots \wedge s_{i-1} \notin x], \dots$$

where $(s_i \in U)_{i \in \mathbb{N}}$ is a random sequence by sampling elements of U with replacement.

Using just one of these functions, combined with symmetrization, gives the ρ value:

$$\rho_i = \log \frac{(1-w_q-w_u+w_1)^i w_1}{\max\{(1-w_q)^i w_q,\ (1-w_u)^i w_u\}} \Bigg/ \log \frac{(1-w_q-w_u+w_2)^i w_2}{\max\{(1-w_q)^i w_q,\ (1-w_u)^i w_u\}}.$$

We want to show $\rho_{\mathrm{mh}} = \log \frac{w_1}{w_q+w_u-w_1} / \log \frac{w_2}{w_q+w_u-w_2} \geq \min_{i\geq 0} \rho_i$. For this we show the following lemma, which intuitively says that it is never advantageous to combine multiple filter families:

Lemma 2.3.6. *The function $f(x,y,z,t) = \log(\max\{x,y\}/z)/\log(\max\{x,y\}/t)$, defined for $\min\{x,y\} \geq z \geq t > 0$, is quasi-concave.*

This means in particular that

$$\frac{\log(\max\{x+x', y+y'\}/(z+z'))}{\log(\max\{x+x', y+y'\}/(t+t'))} \geq \min\left\{\frac{\log(\max\{x,y\}/z)}{\log(\max\{x,y\}/t)}, \frac{\log(\max\{x',y'\}/z')}{\log(\max\{x',y'\}/t')}\right\},$$

when the variables are in the range of the lemma.

Proof. We need to show that the set

$$\{(x,y,z,t) : \log(\max\{x,y\}/z)/\log(\max\{x,y\}/t) \geq \alpha\} = \{(x,y,z,t) : \max\{x,y\}^{1-\alpha} t^\alpha \geq z\}$$

is convex for all $\alpha \in [0,1]$ (since $z \geq t$ so $f(x,y,z,t) \in [0,1]$). This would follow if $g(x,y,t) = \max\{x,y\}^{1-\alpha} t^\alpha$ would be quasi-concave itself, and the eigenvalues of the Hessian of g are exactly 0, 0 and $-(1-\alpha)\alpha t^{\alpha-2} \max\{x,y\}^{-\alpha-1} \left(\max\{x,y\}^2 + t^2\right)$ so g is even concave! □

We can then show that MinHash is always dominated by one of the filters described, as

$$\rho_{\mathrm{mh}} = \frac{\log \frac{w_1}{w_x+w_y-w_1}}{\log \frac{w_2}{w_x+w_y-w_2}} = \frac{\log \frac{\sum_{i\ge 0}(1-w_x-w_y+w_1)^i w_1}{\max\{\sum_{i\ge 0}(1-w_x)^i w_x, \sum_{i\ge 0}(1-w_y)^i w_y\}}}{\log \frac{\sum_{i\ge 0}(1-w_x-w_y+w_2)^i w_2}{\max\{\sum_{i\ge 0}(1-w_x)^i w_x, \sum_{i\ge 0}(1-w_y)^i w_y\}}} \ge \min_{i\ge 0} \frac{\log \frac{(1-w_x-w_y+w_1)^i w_1}{\max\{(1-w_x)^i w_x, (1-w_y)^i w_y\}}}{\log \frac{(1-w_x-w_y+w_2)^i w_2}{\max\{(1-w_x)^i w_x, (1-w_y)^i w_y\}}},$$

where the right hand side is exactly the symmetrization of the filters $F^{(i)}$. By monotonizity of $(1-w_x)^i w_x$ and $(1-w_y)^i w_y$ we can further argue that it is even possible to limit ourselves to one of $i \in \{0, \infty, \log(w_x/w_y)/\log((1-w_x)/(1-w_y))\}$, where the first gives Chosen Path, the second gives Chosen Path on the complemented sets, and the last gives a balanced trade-off where $(1-w_x)^i w_x = (1-w_y)^i w_y$.

2.4 Lower bounds

In the remainder of this section, we'll assume $d = \omega(\log n)$. We proceed to define the hard distributions for all further lower bounds.

1. A query $x \in \{0,1\}^d$ is created by sampling d random independent bits Bernoulli(w_q) distribution.

2. A dataset $P \subseteq \{0,1\}^d$ is constructed by sampling $n-1$ vectors with random independent bits from such that $y'_i \sim$ Bernoulli(w_2/w_q) if $x_i = 1$ and $y'_i \sim$ Bernoulli($(w_u - w_2)/(1-w_q)$) otherwise, for all $y' \in P$.

3. A 'close point', y, is created by $y_i \sim$ Bernoulli(w_1/w_q) if $x_i = 1$ and $y_i \sim$ Bernoulli($(w_u - w_1)/(1-w_q)$) otherwise. This point is also added to P.

We thus have $E|x \cap y| = dw_1$ and $E|y| = dw_u$ as well as $E|x \cap y'| = dw_2$ and $E|y'| = dw_u$ for all $y' \in P$ other than the close point. With high probability these are tight up to factors $1 + o(1)$ and by a union bound they hold for all $y' \in P$.

Adding $o(\log n)$ coordinates to x and each y one may ensure equal weight of the query/database sets without changing the inner products. Hence any $(w_q(1 \pm o(1)), w_u(1 \pm o(1)), w_1(1-o(1)), w_2(1+o(1)))$-GapSS data structure must thus be able to find y given x.

Since the dataset is random, we'll assume the data structure is deterministic. In particular we will assume the existence of a single pair of filter functions $f, g : \{0,1\}^d \to \{0,1\}$. We then have for our LSF

$$\begin{aligned}
p_1 &= \Pr_{x,y}[f(x) = 1 \wedge g(y) = 1], \\
p_2 &= \Pr_{x,y'}[f(x) = 1 \wedge g(y') = 1], \\
p_q &= \Pr_{x}[f(x) = 1], \\
p_u &= \Pr_{y}[g(y) = 1].
\end{aligned}$$

These are boolean functions over biased spaces. Hence we will next introduce the p-biased boolean analysis. Our goal will be to lower bound the ρ-values from theorem 4, over all such functions: $\rho_q = \log(p_q/p_1)/\log(p_q/p_2)$ and $\rho_u = \log(p_u/p_1)/\log(p_q/p_2)$.

In the analysis of boolean functions is is common to use $\{-1,1\}^d$ as the function domain. We'll map 1 to -1 (true) and 0 to 1 (false).

2.4.1 p-biased analysis

Given functions $f, g : \{-1,1\}^n \to \{0,1\}$, we write

$$f(x) = \sum_{S\subseteq[n]} \hat{f}(S)\phi_S(x), \qquad g(y) = \sum_{S\subseteq[n]} \hat{g}(S)\gamma_S(y) \tag{2.4}$$

where $\hat{f}, \hat{g} : 2^{[n]} \to \mathbb{R}$ and $\phi(x_i) = \frac{x_i-\mu_x}{\sigma_x}$, $\gamma(y_i) = \frac{y_i-\mu_y}{\sigma_y}$ for $\mu_x = 1-2w_q, \sigma_x = 2\sqrt{w_q(1-w_q)}$ and $\mu_y = 1-2w_u, \sigma_y = 2\sqrt{w_u(1-w_u)}$. Finally $\phi_S, \gamma_S : \{-1,1\}^n \to \mathbb{R}$ are defined $\phi_S(x) = \prod_{i\in S}\phi(x_i)$ and respectively for y.

Any boolean function can be expanded as (2.4), but it is particularly useful in our case. To see why, let π be the probability distribution, with the following probability mass function: $\pi(-1) = w_q, \pi(1) = 1-w_q$, and let $\pi^n : \{-1,1\}^n \to [0,1]$ be the product distribution on $\{-1,1\}^n$. We then have the useful properties:

$$\begin{aligned} p_x = \Pr_{x\sim\pi^n}[f(x)=1] &= E_{x\sim\pi^n}[f(x)] = E_{x\sim\pi^n}[\sum_{S\subseteq[n]} \hat{f}(S)\phi_S(x)] = \hat{f}(\emptyset) \\ &= E_{x\sim\pi^n_{b_x}}[f(x)^2] = E_{x\sim\pi^n_{b_x}}[\sum_{S,T\subseteq[n]} \hat{f}(S)\hat{f}(T)\phi_S(x)\phi_T(x)] = \sum_{S\subseteq[n]} \hat{f}(S)^2. \end{aligned}$$

If we think of f as an LSF-filter, $p_x = \Pr_{x\sim\pi^n}[f(x)=1]$ is the probability that the filter accepts a random point with 'roughly' weight w_q. (Here the weight is the number of -1's, as is common in boolean function analysis.)

Next, we let ψ be the probability distribution, with the following probability mass function:

$$\begin{aligned} \psi(-1,-1) &= w & \psi(-1,1) &= w_q - w \\ \psi(1,-1) &= w_u - w & \psi(1,1) &= 1 - w_q - w_u + w, \end{aligned}$$

then $p_1 = \Pr_{x,y\sim\psi^n}[f(x) = 1 \wedge g(y) = 1]$ is the probability that a random query point and a random data point both get caught by their respective filter. This has the following nice form:

$$\begin{aligned}
p_1 &= E_{x,y\sim\psi_b^n}[f(x)g(y)] \\
&= E_{x,y\sim\psi^n}[\sum_{S,T\subseteq[n]} \hat{f}(S)\hat{g}(T)\phi_S(x)\gamma_T(x)] \\
&= \sum_{S\subseteq[n]} \hat{f}(S)\hat{g}(S)E_{x,y\sim\psi^n}[\phi_S(x)\gamma_T(x)] \\
&= \sum_{S\subseteq[n]} \hat{f}(S)\hat{g}(S)E_{x,y\sim\psi^n}\left[\prod_{i\in S}\frac{x_i-\mu_x}{\sigma_x}\frac{y_i-\mu_y}{\sigma_y}\right] \\
&= \sum_{S\subseteq[n]} \hat{f}(S)\hat{g}(S)\left(\frac{E_{x,y\sim\psi^n}[x_iy_i]-\mu_x\mu_y}{\sigma_x\sigma_y}\right)^{|S|} \\
&= \sum_{S\subseteq[n]} \hat{f}(S)\hat{g}(S)\left(\frac{w_1-w_qw_u}{\sqrt{w_q(1-w_q)w_u(1-w_u)}}\right)^{|S|}. \qquad (2.5)
\end{aligned}$$

Replacing w_1 with w_2 we get the equivalent expression for p_2. Note that if $w_2 = w_qw_u$ equation (2.5) is just $\hat{f}(\emptyset)\hat{g}(\emptyset) = p_xp_y$.

2.4.2 O'Donnel Method $(w_q = w_u, w_2 > w_qw_u)$

The simplest approach is to use the expansion directly. This is what O'Donnel used [145] to prove the first optimal LSH lower bounds for data-independent hashing.

We define the norm $\|f\|_q = (E_{x\sim\pi^n}f(x)^q)^{1/q}$ and equivalently for g with its respective distribution. Note that since f and g are boolean, we have $\|f\|_q = (E_{x\sim\pi^n}f(x))^{1/q} = p_x^{1/q}$. This will turn out to be very useful.

Recall theorem 3. We will prove something slightly stronger:

Lemma 2.4.1. *Assume an stream-LSF scheme uses function $f,g : \{-1,1\}^n \to \{0,1\}$. Assume further that $\sum_{|S|=k}\hat{f}(S)\hat{g}(S) \ge 0$ for all $k \in [n]$, then any LSF data structure must have*

$$\rho_u, \rho_q \ge \log\left(\frac{w_1-w_qw_u}{\sqrt{w_q(1-w_q)w_u(1-w_u)}}\right) \Big/ \log\left(\frac{w_2-w_qw_u}{\sqrt{w_q(1-w_q)w_u(1-w_u)}}\right).$$

In particular this bound holds when $\hat{f} = \hat{g}$, since $\sum_{|S|=k}\hat{f}(S)^2$ is clearly non-negative. If $w_q = w_u$ and $f = g$ we would get that, which is how to get theorem 3.

Proof. Let $\alpha = \frac{w_1-w_qw_u}{\sqrt{w_q(1-w_q)w_u(1-w_u)}}$ and $\beta = \frac{w_2-w_qw_u}{\sqrt{w_q(1-w_q)w_u(1-w_u)}}$, such that $p_1 = \sum_{S\subseteq[n]}\hat{f}(S)\hat{g}(S)\alpha^{|S|}$ and $p_2 = \sum_{S\subseteq[n]}\hat{f}(S)\hat{g}(S)\beta^{|S|}$.

By Hölder's inequality $\sum_S \hat{f}(S)\hat{g}(S) \le \min\{\|f\|_1\|g\|_\infty, \|f\|_\infty\|g\|_1\} = \min\{p_q, p_u\}$. Let $p = \min\{p_u, p_q\}$. By assumption $\sum_{|S|=k}\hat{f}(S)\hat{g}(S) \ge 0$ for all k, so we have that $\sum_k \alpha^k(\sum_{|S|=k}\hat{f}(S)\hat{g}(S))/p$ is a weighted average over the α^k terms. As such we can use the power-means inequality:

$$\begin{aligned}(p_1/p)^{1/\log\alpha} &= \Big(\sum_k (e^k)^{\log\alpha}\sum_{|S|=k}\hat f(S)\hat g(S)/p\Big)^{1/\log\alpha}\\ &\le \Big(\sum_k (e^k)^{\log\beta}\sum_{|S|=k}\hat f(S)\hat g(S)/p\Big)^{1/\log\beta} = (p_2/p)^{1/\log\beta}.\end{aligned}$$

which implies the lower bound on ρ_u, ρ_q:

$$\rho_u, \rho_q \ge \frac{\log p_1/p}{\log p_2/p} \ge \frac{\log\alpha}{\log\beta}.$$

For ρ_q the inequality above follows from $\log(p/p_1)/\log(p/p_2)$ being increasing in p. For ρ_u it is simply increasing the denominator or decreasing the numerator. □

As noted the bound is sharp against our upper bound when w_u, w_q, w_1, w_2 are all small. Also notice that $\log\alpha/\log\beta \le \frac{1-\alpha}{1+\alpha}\frac{1-\beta}{1+\beta}$ is a rather good approximation for α and β close to 1. Here the right hand side is the ρ value of Spherical LSH with the batch-normalization embedding discussed in section 2.3.2.

It would be interesting to try an extend this bound to get rid of the $\sum_{|S|=k}\hat f(S)\hat g(S) \ge 0$ assumption.

Note that the lower bound becomes 0 when $w_2 \to w_q w_u$. In the next section we will find a bound for exactly this case.

2.4.3 Hypercontractive Lower Bound 1 $(w_2 = w_q w_u)$

Recall theorem 2: Given $\alpha \ge 0$ and $0 \le r, s \le 1/2$, let $u_q = \log\frac{1-w_q}{w_q}$ and $u_u = \log\frac{1-w_u}{w_u}$. If r and s are such that $\frac{\sinh(u_q r)}{\sinh(u_q(1-r))}\frac{\sinh(u_u s)}{\sinh(u_u(1-s))} = \left(\frac{w_1 - w_q w_u}{w_q(1-w_q)w_u(1-w_u)}\right)^2$, then any LSF data structure must have

$$\rho_q \ge 1 - s - \alpha r \quad \text{and} \quad \rho_u \ge \alpha - s - \alpha r.$$

Proof. We will prove this theorem using the p-biased version of the hypercontractive inequality, which says:

Theorem 7 ([143] also [141] Theorem 10.18 and [187])**.** *Let (Ω, π) be a finite probability space, $|\Omega| \ge 2$, in which every outcome has probability at least $\lambda < 1/2$. Let $f \in L^2(\Omega, \pi)$. Then for any $q > 2$ and $v = \sqrt{\frac{\sinh(u/q)}{\sinh(u/q')}}$,*

$$\sum_{S\subseteq[n]} v^2 \hat f(S)^2 \le \|f\|_{q'}$$

where $1/q + 1/q' = 1$ and $u = \log\frac{1-\lambda}{\lambda}$.

We can generalize this to two general functions, using Cauchy Schwartz:

$$p_1 = \sum_{S\subseteq[n]} \sqrt{v\sigma}^{|S|}\hat f(S)\hat g(S) \le \sqrt{\sum_{S\subseteq[n]} v^{|S|}\hat f^2(S)\sum_{S\subseteq[n]}\sigma^{|S|}\hat g^2(S)} \le \|f\|_{1/(1-r)}\|g\|_{1/(1-s)} = p_x^{1-r}p_y^{1-s}.$$

where $v = \frac{\sinh(u_x r)}{\sinh(u_x(1-r))}$, $\sigma = \frac{\sinh(u_y s)}{\sinh(u_y(1-s))}$, $u_x = \log\frac{1-w_q}{w_q}$, $u_y = \log\frac{1-w_u}{w_u}$ and $r, s < 1/2$.

In the case $w_2 = w_q w_u$ we have $p_2 = p_u p_q$ by the discussion in section 2.4.1. Using this, we simply lower bound ρ_q and ρ_u in theorem 4 by

$$\begin{aligned}
\rho_q &= \frac{\log p_1/p_q}{\log p_2/p_q} \geq \frac{\log p_q^{1-r} p_u^{1-s}/p_q}{\log p_q p_u/p_q} = 1 - s - \alpha r \quad \text{and} \\
\rho_u &= \frac{\log p_1/p_u}{\log p_2/p_q} \geq \frac{\log p_q^{1-r} p_u^{1-s}/p_u}{\log p_q p_u/p_q} = \alpha - s - \alpha r,
\end{aligned}$$

where $\alpha = \frac{\log 1/p_q}{\log 1/p_u}$. □

If we set $w_q = w_u$, $\alpha = 1$, $r = s$ and $\sigma = v$, then the theorem asks us to set

$$r = \frac{\log \frac{e^{u_x}(1+e^{u_x}v)}{e^{u_x}+v}}{2u_x} = \frac{\log \frac{w_1(1-w_q)^2}{(1-2w_q+w_1)w_q^2}}{2\log\frac{1-w_q}{w_q}} = 1/2 - \frac{\log \frac{1-2w_q+w_1}{w_1}\frac{1-w_q}{w_q}}{2\log\frac{1-w_q}{w_q}},$$

where we note that $r < 1/2$ since $w_1 < 1 - 2w_q + w_1$. Now $\rho_q, \rho_u \geq 1 - r - s$ is exactly $\log\frac{w_1}{1-2w_q+w_1}\frac{1-w_q}{w_q} / \log\frac{1-w_q}{w_q}$ matching corollary 1 as we wanted.

We would like a general way to optimize over r and s, but as with our upper bounds, we don't know how to solve this on a closed form. We can however show the following equation which must be satisfied by optimal s and r:

We have $\frac{dv}{dr} = \frac{u_x \sinh(u_x)}{\sinh(u_x(1-r))^2}$ and likewise for σ and s. By Lagrange multipliers we get the two equations, for some $\lambda \in \mathbb{R}$: $\frac{\lambda\sigma u_x \sinh(u_x)}{\sinh(u_x(1-r))^2} = \alpha$, $\frac{\lambda v u_y \sinh(u_y)}{\sinh(u_y(1-s))^2} = 1$. Dividing through gives the condition:

$$\frac{u_x \sinh(u_x)\sinh(u_y s)\sinh(u_y(1-s))}{u_y \sinh(u_y)\sinh(u_x r)\sinh(u_x(1-r))} = \alpha.$$

Because of the $r, s < 1/2$ condition, this is however not always possible to achieve. Figure 2.1 suggests that the lower bound is tight towards theorem 5 when $w_q = w_u$ and this condition can be met.

Wolff [187] has shown how to extend the p-biased hypercontractive inequality beyond $r, s \leq 1/2$. However his work is only asymptotic. From the plots it is also clear that for $w_q \neq w_u$ theorem 2 is not sharp. It thus seems evident that we need new methods. In the next section we will investigate a new two-function hypercontractive inequality for this purpose.

2.4.4 Hypercontractive Lower Bound 2 ($w_2 = w_q w_u$)

We conjecture a new hypercontractive inequality:

Conjecture 2 (Two-Function p-Biased Hypercontractivity Inequality). *For* $0 < w_q w_u \leq w \leq w_q, w_u < 1$, *Let* $\psi : \{-1,1\}^2 \to [0,1]$ *be the joint probability density function* $\sim \begin{bmatrix} w & w_u - w \\ w_q - w & 1-w_q-w_u+w \end{bmatrix}$.

For any pair of boolean functions $f, g : \{-1,1\}^n \to \{0,1\}$ then

$$E_{x,y\sim\psi}[f(x)g(y)] \le \|f\|_r \|g\|_s$$

where $r = s = \log \frac{(1-w_q)(1-w_u)}{w_q w_u} / \log \frac{1-w_q-w_u+w}{w}$.

If $w \le w_q w_u$ then the inequality goes the other direction.

We reduce it to a simple two-variable inequality, from which conjecture 2 and conjecture 1 would follow. For this we will use the following inductive result by O'Donnell, which we have slightly generalized to support (x, y) from arbitrary shared distributions, rather than just ρ correlated. The proof in O'Donnell [141] goes through without changes.

Theorem 8 (Two-Function Hypercontractivity Induction Theorem [141])**.** *Assume that*

$$\mathop{\mathrm{E}}_{(x,y)\sim\pi}[f(x)g(y)] \le \|f\|_r \|g\|_q$$

holds for every $f, g \in L^2(\Omega, \pi)$. Then the inequality also holds for every $f, g \in L^2(\Omega^n, \pi^n)$.

This means we just have to show a certain 'two point' inequality. That is, we would like the following to be true:

Lemma 2.4.2. *For $0 < w_q w_u \le w \le w_q, w_u < 1$, and any $f_{-1}, f_1, g_{-1}, g_1 \in \mathbb{R}$, then*

$$\begin{aligned} f_{-1}g_{-1}w + f_{-1}g_1(w_q - w) + f_1 g_{-1}(w_u - w) + f_1 g_1(1 - w_q - w_u + w) \\ \le (w_q f_{-1}^r + (1-w_q) f_1^r)^{1/r} (w_u g_{-1}^s + (1-w_u) g_1^s)^{1/s} \end{aligned}$$

for $r = s = \log \frac{(1-w_q)(1-w_u)}{w_q w_u} / \log \frac{1-w_q-w_u+w}{w}$.

If $w \le w_q w_u$ then the inequality goes the other direction.

Unfortunately we don't have a proof of this. However computer optimization suggests that it is true at least up to an error of 10^{-14}. Equality is achieved when $f_{-1}/f_1 = g_{-1}/g_1$ are either 1 or $(1 - w_q - w_u + w)/w$, and in these points the gradient match, which suggests the choice of r, s is sharp.

Dividing through, we may assume that $f(1)$ and $g(1)$ are both either 0 or 1. (If the values are negative, we can bound lsh by the positive versions. Rhs doesn't care.) If $g(1) = 0$ we just have to show

$$f(-1)g(-1)w + g(-1)(w_u - w) \le w_q^{1/r} f(-1) w_u^{1/s} g(-1)$$

which follows from the one-function Hyper Contractive Inequality.

Otherwise we can focus on proving

$$\begin{aligned} xyw + x(w_q - w) + y(w_u - w) + (1 - w_q - w_u + w) \\ \le (w_q x^r + 1 - w_q)^{1/r} (w_u y^r + 1 - w_u)^{1/r} \end{aligned}$$

where we defined $x = f(-1)/f(1), y = g(-1)/g(1)$.

Assuming now conjecture 2 we continue to prove Lower bound 2 (conjecture 1). Following the same approach as in Lower bound 1, we get

$$\rho_q = \frac{\log p_1/p_q}{\log p_2/p_q} \geq \frac{\log p_q^{1/r} p_u^{1/s}/p_q}{\log p_q p_u/p_q} = (1/r-1)\alpha + 1/s \quad \text{and}$$
$$\rho_u = \frac{\log p_1/p_u}{\log p_2/p_q} \geq \frac{\log p_q^{1/r} p_u^{1/s}/p_u}{\log p_q p_u/p_q} = \alpha/r + (1/s-1).$$

Which is what we wanted.

Setting $\alpha = \frac{D(w_u,1-w_q)}{D(w_q,1-w_u)}$ gives

$$\rho_q \geq \frac{(1-w_q-w_u)\log(\frac{1-w_q-w_u+w_1}{w_1}) - D(w_u,1-w_q)}{D(w_q,1-w_u)},$$
$$\rho_u \geq \frac{(1-w_q-w_u)\log(\frac{1-w_q-w_u+w_1}{w_1}) - D(w_q,1-w_u)}{D(w_q,1-w_u)},$$

matching exactly corollary 1 for $w_2 = w_q w_u$ and all w_q, w_u, w_1.

Besides actually proving lemma 2.4.2, it would be nice to extend it to a complete spectrum of r, s values. The fact that we get a match in this specific point suggests that this may indeed be a fruitful path to showing optimality of the entire theorem 5.

2.5 Conclusion

We show new matching upper and lower bounds for Set Similarity in the symmetric setting, $w_q = w_u$. We also show strong evidence that our upper bound is optimal in the asymmetric setting, $w_q \neq w_u$, as well as in the time-space trade-offs. If the lower bounds can be extended, this would unify the approaches between sparse and vectors on the sphere, and close this important area of LSH, which has always seemed a lot more mystical, with its ad hoc feeling MinHash approach.

Th

2.5.1 Open problems

More closed forms In particular theorem 5, but also the lower bounds, suffer from only being indirectly stated. It would be useful to have a closed form for how to set t_u and t_q for all values of w_q, w_u, w_1, w_2 - both for practical purposes and for showing properties about the trade-off.

More lower bounds Besides proving conjecture 1 it would be useful to extend it to the entire space/time trade-off. This would seemingly require new hypercontractive inequalities, something that may also be useful in other parts of boolean function analysis.

Handle small sets We currently assume that $w_q, w_u, w_1, w_2 \in [0,1]$ are constants independent of $|U|$. For the purposes of finding the optimal space partition for GapSS this is not a big deal, but for practical applications of set similarity, supporting small sets would make supermajorities a lot more useful.

Algorithms for low dimension We know that LSF can break the LSH lower bounds when $d = O(\log n)$ [43]. It would be nice to have something similar for sets, even though universes that small will be pretty rare.

Data dependent As mentioned, the biggest break through in LSH over the last decade is probably data-dependent LSH. Naturally we will want to know how this can be extended to set data.

Sparse, non-binary data We now know that threshold functions do well on binary data and on the sphere. It is an exciting open problem to analyse how they do on sparse data on the sphere. This may be the most common type of data in practice.

2.6 Appendix

Lemma 2.6.1. *Let $X \in \mathbb{R}^k$ be a random variable, such that the cumulant generating function $\Lambda(\lambda) = \log E[\exp\langle\lambda, X\rangle]$ is finite for $\lambda \in \mathbb{R}^k$. Define $\Lambda^*(z) = \sup_{\lambda\in\mathbb{R}^k}\{\langle\lambda, z\rangle - \Lambda(\lambda)\}$. Then $\mathcal{L}(z) = (\nabla_\lambda\Lambda)^{-1}(z)$ is well defined for $z \in (\min X, \max X)$, and*

$$\Lambda^*(z) = \langle z, \mathcal{L}(z)\rangle - \Lambda(\mathcal{L}(z)), \tag{2.6}$$

$$\nabla_z\Lambda^*(z) = \mathcal{L}(z), \tag{2.7}$$

$$H_z\Lambda^*(z) = (H_\lambda\Lambda)^{-1}(z). \tag{2.8}$$

Further, $\Lambda(\lambda)$ is convex, $\Lambda^*(z)$ is concave, and $\Lambda^*(EX) = 0$.

Proof of lemma 2.6.1. Finite moment-generating functions are log of a sum of analytic, log-convex functions. Hence $\Lambda(\lambda)$ is analytic and convex. This implies that $\langle\lambda, z\rangle - \Lambda(\lambda)$ has a unique supremum at $z = \nabla\Lambda(\lambda)$, so we can define the inverse $\mathcal{L}(z) = (\nabla\Lambda)^{-1}(z)$ such that $\Lambda^*(z) = \Lambda(\mathcal{L}(z))$ as needed for eq. (2.6).

For eq. (2.7) we compute using the chain-rule:

$$\nabla_z\Lambda^*(z) = \nabla_z\left[\langle z, \mathcal{L}(z)\rangle - \Lambda(\mathcal{L}(z))\right] = z^T J_z\mathcal{L}(z) + \mathcal{L}(z) - (\nabla\Lambda(\lambda))^T J_z\mathcal{L}(z) = \mathcal{L}(z),$$

where $J_z\mathcal{L}$ is the Jacobian of $\mathcal{L}$.

For eq. (2.8) let $f(\lambda) = \nabla\Lambda(\lambda) : \mathbb{R}^d \to \mathbb{R}^d$, then $\mathcal{L}(z) = f^{-1}(z)$ and we can compute using the inverse function theorem:

$$H_z\Lambda^*(z) = J_z\mathcal{L}(z) = (J_\lambda f(\lambda))^{-1} = (J_\lambda\nabla\Lambda(\lambda))^{-1} = (H_\lambda\Lambda)^{-1}.$$

Since Λ is convex, the Hessian is positive-semidefinite, and so its inverse is negative-semidefinite. This implies that Λ^* is concave.

Finally, since Λ is the cumulant generating function, we have $\Lambda(\lambda) = \langle\lambda, EX\rangle + \lambda^T\Sigma\lambda/2 + \dots$ (where Σ is the covariance matrix), and so $\nabla\Lambda(0) = EX$, which gives the last identity: $\mathcal{L}(EX) = 0$. □

Chapter 3

Optimal Las Vegas Locality Sensitive Data Structures

Originally published in: Annual Symposium on Foundations of Computer Science, FOCS 2017

3.1 Introduction

Locality Sensitive Hashing has been a leading approach to high dimensional similarity search (nearest neighbour search) data structures for the last twenty years. Intense research [95, 84, 113, 92, 93, 52, 73, 125, 151, 20, 22, 25, 43, 10, 36] has applied the concept of space partitioning to many different problems and similarity spaces. These data structures are popular in particular because of their ability to overcome the 'curse of dimensionality' and conditional lower bounds by [186], and give sub-linear query time on worst case instances. They achieve this by being approximate and Monte Carlo, meaning they may return a point that is slightly further away than the nearest, and with a small probability they may completely fail to return any nearby point.

Definition 6 ((c, r)-Approximate Near Neighbour)**.** *Given a set P of n data points in a metric space (X, dist), build a data structure, such that given any $q \in X$, for which there is an $x \in P$ with $\mathrm{dist}(q, x) \le r$, we return a $x' \in P$ with $\mathrm{dist}(q, x') \le cr$.*

A classic problem in high dimensional geometry has been whether data structures existed for (c, r)-Approximate Near Neighbour with Las Vegas guarantees, and performance matching that of Locality Sensitive Hashing. That is, whether we could guarantee that a query will always return an approximate near neighbour, if a near neighbour exists; or simply, if we could rule out false negatives? The problem has seen practical importance as well as theoretical. There is in general no way of verifying that an LSH algorithm is correct when it says 'no near neighbours' - other than iterating over every point in the set, in which case the data structure is entirely pointless. This means LSH algorithms can't be used for many critical applications, such as finger print data bases. Even more applied, it has been observed that tuning the error probability parameter is hard to do well, when implementing LSH [84, 34]. A Las Vegas data structure entirely removes this problem. Different authors have described the problem with different names, such as 'Las Vegas' [91], 'Have no false negatives' [86, 148], 'Have total recall' [156], 'Are exact' [32] and 'Are explicit' [108].

Recent years have shown serious progress towards finally solving the problem. In particular [148] showed that the problem in Hamming space admits a Las Vegas algorithm with query time $dn^{1.38/c+o(1)}$, matching the $dn^{1/c}$ data structure of [95] up to a constant factor in the exponent. In this paper we give an algorithm in the Locality Sensitive Filter framework [43, 61], which not only removes the factor 1.38, but improves to $dn^{1/(2c-1)+o(1)}$ in the case $cr \approx d/2$, matching the algorithms of [21] for Hamming space.

We would like to find an approach to Las Vegas LSH that generalizes to the many different situations where LSH is useful. Towards that goal, we present as second algorithm for the approximate similarity search problem under Braun-Blanquet similarity, which is defined for sets $x, y \subseteq [d]$ as $\text{sim}(x,y) = |x \cap y| / \max(|x|, |y|)$. We refer to the following problem definition:

Definition 7 (Approximate similarity search). *Let $P \subseteq \mathcal{P}([d])$ be a set of $|P| = n$ subsets of $[d]$; (here $\mathcal{P}(X)$ denotes the powerset of X.) let $\text{sim} : \mathcal{P}([d]) \times \mathcal{P}([d]) \to [0,1]$ be a similarity measure. For given $s_1, s_2 \in [0,1]$, $s_1 > s_2$, a solution to the "(s_1, s_2)-similarity search problem under sim" is a data structure that supports the following query operation: on input $q \subseteq [d]$, for which there exists a set $x \in P$ with $\text{sim}(x,q) \geq s_1$, return $x' \in P$ with $\text{sim}(x',q) > s_2$.*

The problem has traditionally been solved using the Min-Hash LSH [48, 47], which combined with the results of Indyk and Motwani [95] gives a data structure with query time dn^ρ and space $dn^{1+\rho}$ for $\rho = \log s_1 / \log s_2$. Recently it was shown by [63] that this could be improved for vectors of equal weight to $\rho = \log \frac{2s_1}{1+s_1} / \log \frac{2s_2}{1+s_2}$. We show that it is possible to achieve this recent result with a data structure that has no false negatives.

3.1.1 Summary of Contributions

We present the first Las Vegas algorithm for approximate near neighbour search, which gives sub-linear query time for any approximation factor $c > 1$. This solves a long standing open question from [91] and [148]. In particular we get the following two theorems:

Theorem 9. *Let $X = \{0,1\}^d$ be the Hamming space with metric $\text{dist}(x,y) = \|x \oplus y\| \in [0,d]$ where $\oplus$ is "xor" or addition in $\mathbb{Z}_2$. For every choice of $0 < r$, $1 < c$ and $cr \leq d/2$, we can solve the (c,r)-approximate near neighbour problem in Hamming space with query time dn^ρ and space usage $dn + n^{1+\rho}$ where $\rho = 1/c + \hat{O}((\log n)^{-1/4})$.*

Note: $\hat{O}$ hides $\log\log n$ factors.

Corollary 4. *When $r/d = \Omega((\log n)^{-1/6})$, we get the improved exponent $\rho = \frac{1-cr/d}{c(1-r/d)} + \hat{O}((\log n)^{-1/3} d/r)$.*

This improves upon theorem 9 when r/d is constant (or slightly sub-constant), including in the important "random case", when $r/d = 1/(2c)$ where we get $\rho = 1/(2c-1) + o(1)$.

Theorem 10. *Let sim be the Braun-Blanquet similarity $\text{sim}(x,y) = |x \cap y| / \max(|x|, |y|)$. For every choice of constants $0 < s_2 < s_1 < 1$, we can solve the (s_1, s_2)-similarity problem over sim with query time dn^ρ and space usage $dn + n^{1+\rho}$ where $\rho = \log s_1 / \log s_2 + \hat{O}((\log n)^{-1/2})$.*

For sets of fixed size w, the dn terms above can be improved to wn. It is also possible to let s_1 and s_2 depend on n with some more work.

The first result matches the lower bounds by [142] for "data independent" LSH data structures for Hamming distance and improves upon [148] by a factor of $\log 4 > 1.38$ in the exponent. By deterministic reductions from ℓ_2 to ℓ_1 [94] and ℓ_1 to hamming (appendix 3.6.1), this also gives the best currently known Las Vegas data structures for ℓ_1 and ℓ_2 in $\mathbb{R}^d$. The second result matches the corresponding lower bounds by [63] for Braun-Blanquet similarity and, by reduction, Jaccard similarity. See table 3.1 for more comparisons.

Detaching the data structures from our constructions, we give the first explicit constructions of large Turán Systems [169], which are families $\mathcal{T}$ of k-subsets of $[n]$, such that any r-subset of $[n]$ is contained in a set in $\mathcal{T}$. Lemma 3.4.1 constructs (n, k, r)-Turán Systems using $(n/k)^r e^\chi$ sets, where $\chi = O(\sqrt{r}\log r + \log k + \log\log n)$. For small values of k this is sharp with the lower bound of $\binom{n}{r}/\binom{k}{r}$, and our systems can be efficiently decoded, which is likely to have other algorithmic applications.

3.1.2 Background and Related Work

The arguably most successful technique for similarity search in high dimensions is Locality-Sensitive Hashing (LSH), introduced in 1998 by [95, 88]. The idea is to make a random space partition in which similar points are likely to be stored in the same region, thus allowing the search space to be pruned substantially. The granularity of the space partition (the size/number of regions) is chosen to balance the expected number of points searched against keeping a (reasonably) small probability of pruning away the actual nearest point. To ensure a high probability of success (good recall) one repeats the above construction, independently at random, a small polynomial (in n) number of times.

In [148, 32] it was shown that one could change the above algorithm to not do the repetitions independently. (Eliminating the error probability of an algorithm by independent repetitions, of course, takes an infinite number of repetitions.) By making correlated repetitions, it was shown possible to reach zero false negatives much faster, after only polynomially many repetitions. This means, for example, that they needed more repetitions than LSH does to get 0.99 success rate, but fewer than LSH needs for success rate $1 - 2^{-n}$.

An alternative to LSH was introduced by [43, 77]. It is referred to as Locality Sensitive Filters, or LSF. While it achieves the same bounds as LSH, LSF has the advantage of giving more control to the algorithm designer for balancing different performance metrics. For example, it typically allows better results for low dimensional data, $d = O(\log n)$, and space/time trade-offs [25]. The idea is to sample a large number of random sections of the space. In contrast to LSH these sections are not necessarily partitions and may overlap heavily. For example, for points on the sphere S^{d-1} the sections may be defined by balls around the points of a spherical code. One issue compared to LSH is that the number of sections in LSF is very large. This means we need to impose some structure so we can efficiently find all sections containing a particular point. With LSH the space partitioning automatically provided such an algorithm, but for LSF it is common to use a kind of random product code. (An interesting alternative is [63], which uses a random branching processes.) LSF is similar to LSH in that it only approaches 100% success rate as the number of sections goes to infinity.

The work in this paper can be viewed as way of constructing correlated, efficiently decodable filters for Hamming space and Braun-Blanquet similarity. That is, our filters guarantee that any two close points are contained in a shared section, without having an infinite number of sections. Indeed the number of sections needed is equal to that needed by random constructions for achieving constant success probability, up to $n^{o(1)}$ factors. It is not crucial that our algorithms are in the LSF framework rather than LSH. Our techniques can make correlated LSH space partitions of optimal size as well as filters. However the more general LSF framework allows for us to better show of the strength of the techniques.

One very important line of LSH/LSF research, that we don't touch upon in this paper, is that of data dependency. In the seminal papers [22, 31, 25] it was shown that the performance of space partition based data structures can be improved, even in the worst case, by considering the layout of the points in the data base. Using clustering, certain bad cases for LSH/LSF can be removed, leaving only the case of "near random" points to be considered, on which LSH works very well. It seems possible to make Las Vegas versions of these algorithms as well, since our approach gives the optimal performance in these near random cases. However one would need to find a way to derandomize the randomized clustering step used in their approach.

There is of course also a literature of deterministic and Las Vegas data structures not using LSH. As a baseline, we note that the "brute force" algorithm that stores every data point in a hash table, and given a query, $q \in \{0,1\}^d$, looks up every $\sum_{k=1}^{r} \binom{d}{k}$ point of Hamming distance most r. This requires $r \log(d/r) < \log n$ to be sub-linear, so for a typical example of $d = (\log n)^2$ and $r = d/10$ it won't be practical. In [69] this was somewhat improved to yield $n(\log n)^r$ time, but it still requires $r = O(\frac{\log n}{\log \log n})$ for queries to be sub-linear. We can also imagine storing the nearest neighbour for every point in $\{0,1\}^d$. Such an approach would give fast (constant time) queries, but the space required would be exponential in r.

In Euclidean space (ℓ_2 metric) the classical K-d tree algorithm [46] is of course deterministic, but it has query time $n^{1-1/d}$, so we need $d = O(1)$ for it to be strongly sub-linear. Allowing approximation, but still deterministic, [34] found a $(\frac{d}{c-1})^d$ algorithm for $c > 1$ approximation. They thus get sublinear queries for $d = O(\frac{\log n}{\log \log n})$.

For large approximation factors [88] gave a deterministic data structure with query time $O(d \log n)$, but space and preprocessing more than $n \cdot O(1/(c-1))^d$. In a different line of work, [91] gave a deterministic $(d\epsilon^{-1} \log n)^{O(1)}$ query time, fully deterministic algorithm with space usage $n^{O(1/\epsilon^6)}$ for a $3 + \epsilon$ approximation.

See Table 3.1 for an easier comparison of the different results and spaces.

3.1.3 Techniques

Our main new technique is a combination of 'splitters' as defined by [136, 16], and 'tensoring' which is a common technique in the LSH literature.

Reference	**Space**	**Exponent, search time**	**Comments**
[46]	ℓ_2	$1 - 1/d$	Exact algorithm, Fully deterministic.
[69]	Hamming	$r\frac{\log\log n}{\log n}$	Sub-linear for $r < \frac{\log n}{\log\log n}$. Exact.
[34]	ℓ_2	$d^{\frac{\log(d/(c-1))}{\log n}}$	Sub-linear for $d < \frac{\log n}{\log\log n}$.
[88]	Hamming	$o(1)$	c-approximation, Fully deterministic, $(1/(c-1))^d$ space.
[91]	Hamming	$o(1)$	$(3+\epsilon)$-approximation, Fully deterministic, $n^{\Omega(1/\epsilon^6)}$ space.
[32]	Hamming	$\approx 3/c$	The paper makes no theoretical claims on the exponent.
[148]	Hamming	$1.38/c$	Exponent $1/c$ when $r = o(\log n)$ or $(\log n)/(cr) \in \mathbb{N}$.
[146]	ℓ_p	$O(d^{1-1/p}/c)$	Sub-linear for ℓ_2 when $c = \omega(\sqrt{d})$.
This paper	Hamming, ℓ_1, ℓ_2	$1/c$	Actual exponent is $\frac{1-cr/d}{c(1-r/d)}$ which improves to $1/(2c-1)$ for $cr \approx d/2$.
[148]	Braun-Blanquet	$1.38\,\frac{1-b_1}{1-b_2}$	Via reduction to Hamming. Requires sets of equal weight.
This paper	Braun-Blanquet	$\frac{\log 1/b_1}{\log 1/b_2}$	See [63] figure 2 for a comparison with [148].

Table 3.1: Comparison of Las Vegas algorithms for high dimensional near neighbour problems. The exponent is the value ρ, such that the data structure has query time $n^{\rho+o(1)}$. All listed algorithms, except for [91] use less than n^2 space. All algorithms give c-approximations, except for the first two, and for [91], which is a $(3+\epsilon)$-approximation.

Tensoring means constructing a large space partition $P \subseteq \mathcal{P}(X)$ by taking multiple smaller random partitions $P_1, P_2, \dots$ and taking all the intersections $P = \{p_1 \cap p_2, \cdots \mid p_1 \in P_1, p_2 \in P_2, \dots\}$. Often the implicit partition P is nearly as good as a fully random partition of equal size, while it is cheaper to store in memory and allows much faster lookups of which section covers a given point. In this paper we are particularly interested in P_i's that partition different small sub-spaces, such that P is used to increase the dimension of a small, explicit, good partition.

Unfortunately tensoring doesn't seem to be directly applicable for deterministic constructions, since deterministic space partitions tend to have some overhead that gets amplified by the product construction. This is the reason why [148] constructs hash functions directly using algebraic methods, rather than starting with a small hash function and 'amplifying' as is common for LSH. Algebraic methods are great when they exist, but they tend to be hard to find, and it would be a tough order to find them for every similarity measure we would like to make a data structure for.

It turns out we can use splitters to help make tensoring work deterministically. Roughly, these are generalizations of perfect hash functions. However, where a (d, m, k)-perfect hash family guarantees that for any set $S \subseteq [d]$ of size k, there is a function $\pi : [d] \to [m]$ such that $|\pi(S)| = k$, a (d, m)-splitter instead guarantees that the is some π such that $|S \cap \pi^{-1}(i)| = d/m$ for each $i = 1, \dots, m$; or as close as possible if m does not divide d. That is, for any S there is some π that 'splits' S evenly between m buckets.

Using splitters with tensoring, we greatly limit the number of combinations of smaller space partitions that are needed to guarantee covering. We use this to amplify partitions found probabilistically and verified deterministically. The random aspect is however only for convenience, since the greedy set cover algorithm would suffice as well, as is done in [16]. We don't quite get a general reduction from Monte Carlo to Las Vegas LSH data structures, but we show how two state of the art algorithms may be converted at a negligible overhead.

A final technique to make everything come together is the use of dimensionality reductions. We can't quite use the standard bit-sampling and Johnson–Lindenstrauss lemmas, since those may (though unlikely) increase the distance between originally near points. Instead we use two dimensionality reduction lemmas based on partitioning. Similarly to [148] and others, we fix a random permutation. Then given a vector $x \in \{0, 1\}^d$ we permute the coordinates and partition into blocks $x_1, \dots, x_{d/B}$ of size B. For some linear distance function, $\text{dist}(x, y) = \text{dist}(x_1, y_1) + \cdots + \text{dist}(x_{d/B}, y_{d/B})$, which implies that for some i we must have $\text{dist}(x_i, y_i) \le \text{dist}(x, y)B/d$. Running the algorithm separately for each set of blocks guarantee that we no pair gets mapped too far away from each other, while the randomness of the permutation lets us apply standard Chernoff bounds on how close the remaining points get.

Partitioning, however, doesn't work well if distances are very small, $cr << d$. This is because we need $B = \frac{d}{cr}\epsilon^{-2} \log n$ to get the said Chernoff bounds on distances for points at distance cr. We solve this problem by hashing coordinates into buckets of $\approx cr/\epsilon$ and taking the xor of each bucket. This has the effect of increasing distances and thereby allowing us to partition into blocks of size $\approx \epsilon^{-3} \log n$. A similar technique was used for dimensionality reduction in [113], but without deterministic guarantees. The problem is tackled fully deterministically in [91] using codes, but with the slightly worse bound of $\epsilon^{-4} \log n$.

For the second problem of Braun-Blanquet similarity we also need a way to reduce the dimension to a manageble size. Using randomized reductions (for example partitioning), we can reduce to $|x \cap y| \sim \log n$ without introducing too many false positives. However we could easily have e.g. universe size $d = (\log n)^{100}$ and $|x| = |y| = (\log n)^2$, which is much too high a dimension for our splitter technique to work. There is probably no hope of actually reducing d, since increasing $|x|/d$ and $|y|/d$ makes the problem we are trying to solve easier, and such a reduction would thus break LSH lower bounds.

Instead we introduce tensoring technique based on perfect hash functions, which allows us to create Turán Systems with very large universe sizes for very little overhead.

In the process of showing our results, we show a useful bound on the ratio between two binomial coefficients, which may be of separate interest.

3.1.4 Notation

We use $[d] = \{1, \dots, d\}$ as convenient notation sets of a given size. Somewhat overloading notation, for a predicate P, we also use the Iversonian notation $[P]$ for a value that is 1 if P is true and 0 otherwise.

For a set $x \subseteq [d]$, we will sometimes think of it as a subset of the universe $[d]$, and at other times as a vector $x \in \{0,1\}^d$, where $x_i = 1$ indicates that $i \in x$. This correspondence goes further, and we may refer to the set size $|x|$ or the vector norm $\|x\|$, which is always the Hamming norm, $\|x\| = \sum_{i=1}^d x_i$. Similarly for two sets or points $x, y \in \{0,1\}^d$, we may refer to the inner product $\langle x, y \rangle = \sum_{i=1}^d x_i y_i$ or to the size of their intersection $|x \cap y|$.

We use $S \times T = \{(s,t) : s \in S, t \in T\}$ for the cross product, and $x \oplus y$ for symmetric difference (or 'xor'). $\mathcal{P}(X)$ is the power set of X, such that $x \subseteq X \equiv x \in \mathcal{P}(X)$. $\binom{X}{k}$ denotes all subsets of X of size k.

For at set $S \subseteq [d]$ and a vector $x \in \{0,1\}^d$, we let x_S be the projection of x onto S. This is an $|S|$-dimensional vector, consisting of the coordinates $x_S = \langle x_i : i \in S \rangle$ in the natural order of i. For a function $f : [a] \to [b]$ we let $f^{-1} : \mathcal{P}([b]) \to \mathcal{P}([a])$ be the 'pullback' of f, such that $f^{-1}(S) = \{i \in [a] \mid f(i) \in S\}$. For example, for $x \in \{0,1\}^a$, we may write $x_{f^{-1}(1)}$ to be the vector x projected onto the coordinates of $f^{-1}(\{1\})$.

Sometimes when a variable is $\omega(1)$ we may assume it is integral, when this is achievable easily by rounding that only perturbs the result by an insignificant $o(1)$ amount.

The functional $\text{poly}(a, b, \dots)$ means any polynomial combination of the arguments, essentially the same set as $(a \cdot b \dots)^{\pm O(1)}$.

3.1.5 Organization

We start by laying out the general framework shared between our algorithms. We use a relatively common approach to modern near neighbour data structures, but the overview also helps establish some notation used in the later sections.

The second part of section 3.2 describes the main ideas and intuition on how we achieve our results. In particular it defines the concept of 'splitters' and how they may be used to create list-decodable codes for various measures. The section finally touches upon the issues we encounter on dimensionality reduction, which we can use to an extent, but which is restricted by our requirement of '1-sided' errors.

In sections 3.3 and 3.4 we prove the main theorems from the introduction. The sections follow a similar pattern: First we introduce a filter family and prove its existence, then we show a dimensionality reduction lemma and analyze the resulting algorithm.

3.2 Overview

Both algorithms in this paper follow the structure of the Locality Sensitive Filter framework, which is as follows: For a given universe U, we define a family $\mathcal{F}$ of 'filters' equipped with a (possibly random) function $F : U \to \mathcal{P}(\mathcal{F})$, which assigns every point a set of filters.

Typically, $\mathcal{F}$ will be a generous covering of U, and $F(x)$ will be the sets that cover the point x. Critically, any pair x, y that is close/similar enough in U must share a filter, such that $F(X) \cap F(Y) \neq \emptyset$. Further we will want that pairs x, y that are sufficiently far/dissimilar only rarely share a filter, such that $E[|F(x) \cap F(Y)|]$ is tiny.

To construct the data structure, we are given a set of data points $P \subseteq U$. We compute $F(x)$ for every $x \in P$ and store the points in a (hash) map $T : \mathcal{F} \to \mathcal{P}(P)$. For any point $x \in P$ and filter $f \in F(x)$, we store $x \in T[f]$. Note that the same x may be stored in multiple different buckets.

To query the data structure with a point $x \in U$, we compute the distance/similarity between x and every point $y \in \bigcup_{f \in F(x)} T[f]$, returning the first suitable candidate, if any.

There are many possible variations of the scheme, such as sampling $\mathcal{F}$ from a distribution of filter families. In case we want a data structure with space/time tradeoffs, we can use different $\mathcal{F}$ functions for data points and query points. However in this article we will not include these extensions.

We note that while it is easy to delete and insert new points in the data structure after creation, we are going to choose $\mathcal{F}$ parametrized on the total number of points, $|P|$. This makes our data structure essentially static, but luckily [144] have found general, deterministic reductions from dynamic to static data structures.

3.2.1 Intuition

The main challenge in this paper will be the construction of filter families $\mathcal{F}$ which are: (i) not too large; (ii) have a $F(\cdot)$ function that is efficient to evaluate; and most importantly, (iii) guarantee that all sufficiently close/similar points always share a filter. The last requirement is what makes our algorithm different from previous results, which only had this guarantee probabilistically.

For concreteness, let us consider the Hamming space problem. Observe that for very low dimensional spaces, $d = (1 + o(1)) \log n$, we can afford to spend exponential time designing a filter family. In particular we can formulate a set cover problem, in which we wish to cover each pair of points at distance $\leq r$ with Hamming balls of radius s. This gives a family that is not much larger than what can be achieved probabilistically, and which is guaranteed to work. Furthermore, this family has sublinear size ($n^{o(1)}$), making $F(x)$ efficient to evaluate, since we can simply enumerate all of the Hamming balls and check if x is contained.

The challenge is to scale this approach up to general d.

Using a standard approach of randomly partitioning the coordinates, we can reduce the dimension to $(\log n)^{1+\epsilon}$. This is basically dimensionality reduction by bit sampling, but it produces $d/\log n$ different subspaces, such that for any pair x, y there is at least one subspace in which their distance is not increased. We are left with a gap from $(\log n)^{1+\epsilon}$ down to $\log n$. Bridging this gap turns out to require a lot more work. Intuitively we cannot hope to simply use a stronger dimensionality reduction, since $\log n$ dimensions only just fit n points in Hamming space and would surely make too many non-similar points collide to be effective.

A natural idea is to construct higher-dimensional filter families by combining multiple smaller families. This is a common technique from the first list decodable error correcting codes, for example [78]: Given a code $\mathcal{C} \subseteq \{0,1\}^d$ with covering radius r, we can create a new code $\mathcal{C}_2 \subseteq \{0,1\}^{2d}$ of size $|\mathcal{C}|^2$ with covering radius $2r$ by taking every pair of code words and concatenating them. Then for a given point $x \in \{0,1\}^{2d}$ we can decode the first and last d coordinates of $x = x_1 x_2$ separately in $\mathcal{C}$. This returns two code words c_1, c_2 such that $\text{dist}(x_1, c_1) \le r$ and $\text{dist}(x_2, c_2) \le r$. By construction $c_1 c_2$ is in $\mathcal{C}_2$ and $\text{dist}(x_1 x_2, c_1 c_2) \le 2r$.

This combination idea is nice when it applies. When used with high quality inner codes, the combined code is close to optimal as well. In most cases the properties of $\mathcal{C}$ that we are interested in won't decompose as nicely. With the example of our Hamming ball filter family, consider $x, y \in \{0,1\}^{2d}$ with distance $\text{dist}(x, y) = r$. If we split $x = x_1 x_2$ an $y = y_1 y_2$ we could decode the smaller vectors individually in a smaller family, however we don't have any guarantee on $\text{dist}(x_1, y_1)$ and $\text{dist}(x_2, y_2)$ individually, so the inner code might fail to return anything at all.

To solve this problem, we use a classic tool for creating combinatorial objects, such as our filter families, called 'splitters'. Originally introduced by [126, 136] they are defined as follows:

Definition 8 (Splitter). *A (B, l)-splitter H is a family of functions from $\{1, \ldots, B\}$ to $\{1, \ldots, l\}$ such that for all $S \subseteq \{1, \ldots, B\}$, there is a $h \in H$ that splits S perfectly, i.e., into equal-sized parts $h^{-1}(j) \cap S$, $j = 1, 2, \ldots, l$. (or as equal as possible, if l does not divide $|S|$).*

The size of H is at most B^l, and using either a generalization by [16] or a simple combinatorial argument, it is possible to ensure that the size of each part $|h^{-1}(j)|$ equals B/l (or as close as possible).

We now explain how splitters help us combine filter families. Let H be a splitter from $\{1, \ldots, 2d\}$ to $\{1, 2\}$. For any $x, y \in \{0,1\}^{2d}$ we can let S be the set of coordinates on which x and y differ. Then there is a function $h \in H$ such that $|h^{-1}(1) \cap S| = |h^{-1}(2) \cap S| = |S|/2$. (Or as close as possible if $|S|$ is odd.) If we repeat the failed product combination from above for every $h \in H$ we get a way to scale our family from d to $2d$ dimensions, taking the size from $|\mathcal{F}|$ to $(2d)^2 |\mathcal{F}|^2$. That is, we only suffer a small polynomial loss. In the end it turns out that the loss suffered from creating filter families using this divide and conquer approach can be contained, thus solving our problem.

An issue that comes up, is that the 'property' we are splitting (such as distance) can often be a lot smaller than the dimensionality d of the points. In particular this original dimensionality reduction may suffer an overhead factor $d/|S|$, which could make it nearly useless if $|S|$ is close to 1. To solve this problem, both of our algorithms employ special half-deterministic dimensionality reductions, which ensures that the interesting properties get 'boosted' and end up taking a reasonable amount of 'space'. These reductions are themselves not complicated, but they are somewhat non-standard, since they can only have a one sided error. For example for Hamming distance we need that the mapped distance is never larger than its expected value, since otherwise we could get false negatives.

For Hamming distance our dimension reduction works by hashing the coordinates randomly from [d] to [m] taking the xor of the coordinates in each bucket. This is related to the β-test in [113]. The idea is that if x and y are different in only a few coordinates, then taking a small random group of coordinates, it is likely to contain at most one where they differ. If no coordinates differ, then after taking the xor the result will still be the same, but if exactly one (or an odd number) of coordinates differ, the resulting coordinate will be different.

For set similarity things are a bit more hairy. There is no data independent dimensionality reduction that can reduce the size of the domain. In fact this would break the lower bounds of e.g. [63]. Instead we come up with a new construction based on perfect hash functions, which greatly increases the number of filters needed, but only as much as we can afford given the easier sub-problems.

The idea can be motivated as follows: Suppose you have to make a family of sets $\mathcal{T} \subseteq \mathcal{P}([n])$ of size r, such that for each each set $K \subseteq [n]$ of size $|K| = k$ there is an $R \in \mathcal{T}$ such that $R \subseteq K$. Then you might try to extend this to the domain $[2n]$ as follows: For each $R \in \mathcal{T}$ and each $b \in \{0,1\}^r$, make a new set $R' = \{i + nb_i : i \in R\}$ (where b_i is padded appropriately). This creates $2^r|\mathcal{T}|$ new sets, which can be shown to have the property, that for any set $K \subseteq [2n]$ of size $|K| = k$, there is an R' such that $R' \subseteq K$. That is as long as $K \cap (K - n) = \emptyset$, since then we can consider $R \in \mathcal{T}$ such that $R \subseteq (K \bmod n)$. That is R is a subset of K 'folded' into the first $[n]$ elements, and one of the R' will be a subset of K.

Because of the requirement that $|K \bmod n| = k$ we need to use perfect hashing as a part of the construction. However for non-Las Vegas algorithms, a similar approach may be useful, simply using random hashing.

3.3 Hamming Space Data Structure

We will give an efficient filter family for LSF in Hamming space. Afterwards we will analyze it and choose the most optimal parameters, including dimensionality reduction.

Lemma 3.3.1. *For every choice of parameters $B, b \in \mathbb{N}$, $b \le B$, $0 < r < B/2$ and $s^2 = O(B/\sqrt{b})$, there exists a code $\mathcal{C} \subseteq \{0,1\}^B$ of size $|\mathcal{C}| = \operatorname{poly}(B^{B/b}) \exp(\frac{s^2}{2(1-r/d)})$ with the following properties:*

1. *Given $x \in \{0,1\}^B$ we can find a subset $C(x) \subseteq \{c \in \mathcal{C} : \operatorname{dist}(x,c) \le B/2 - s\sqrt{B}/2\}$ in time $|C(x)| + \operatorname{poly}(B^{B/b}, e^{s^2 b/B})$.*

2. *For all pairs $x, y \in \{0,1\}^B$ with $\operatorname{dist}(x,y) \le r$ there is some common nearby code word $c \in C(x) \cap C(y)$.*

3. *The code requires $4^b \operatorname{poly}(B^{B/b}, e^{s^2 b/B})$ time for preprocessing and $\operatorname{poly}(B^{B/b}, e^{s^2 b/B})$ space.*

Note that we don't actually guarantee that our 'list-decoding' function $C(x)$ returns *all* nearby code words, just that it returns enough for property (2) which is what we need for the data structure. This is however not intrinsic to the methods and using a decoding algorithm similar to [43] would make it a 'true' list-decoding.

Proof. We first show how to construct a family for $\{0,1\}^b$, then how to enlarge it for $\{0,1\}^B$. We then show that it has the covering property and finally the decoding properties. In order for our probabilistic arguments to go through, we need the following lemma, which follows from Stirling's Approximation:

Lemma 3.3.2. *For $t = \frac{d}{2} - \frac{s\sqrt{d}}{2}$, $1 \le s \le d^{1/4}/2$ and $r < d/2$, Let $x, y \in \{0,1\}^d$ be two points at distance $\operatorname{dist}(x,y) = r$, and let $I = |\{z \in \{0,1\}^d : \operatorname{dist}(z,x) \le t, \operatorname{dist}(z,y) \le t\}|$ be the size of the intersection of two hamming balls around x and y of radius t, then*

$$\frac{7}{8d} \exp\left(\tfrac{-s^2}{2(1-r/d)}\right) \le I\,2^{-d} \le \exp\left(\tfrac{-s^2}{2(1-r/d)}\right)$$

Proof is in the appendix.

Let $s' = s\sqrt{b/B}$. Consider any two points $x, y \in \{0,1\}^b$ with distance $\le (r/d)b$. If we choose $a \in \{0,1\}^b$ uniformly at random, by lemma 3.3.2 we have probability $p = \operatorname{poly}(b)\exp(\frac{-s'^2}{2(1-r/d)})$ that both x and y have distance at most $t = b/2 - s'\sqrt{b/4}$ with c. By the union bound over pairs in $\{0,1\}^b$, if we sample $p^{-1} b \log 2$ independent as, we get constant probability that some a works for every pair. We can verify that a set A of such filters indeed works for every pair in time $4^b|A|$. By repeatedly sampling sets A and verifying them, we get a working A in expected $O(1)$ tries.[1]

Next we define $\mathcal{C}$. We build a splitter, that is a set Π of functions $\pi : [B] \to [B/b]$, such that for every set $I \subseteq [B]$ there is a $\pi \in \Pi$ such that $\lfloor |I|b/B \rfloor \le |\pi^{-1}(j) \cap I| \le \lceil |I|b/B \rceil$ for $j = 1, \dots, B/b$. By the discussion after definition 8, such a set of size $\operatorname{poly}(B^{B/b})$ exists and can be constructed in time and space proportional to its size. Implicitly we can then define $\mathcal{C}$ by making one code word $c \in \{0,1\}^B$ for every $\pi \in \Pi$ and $1 \le j_1, \dots, j_{B/b} \le |A|$, satisfying the requirement that $c_{\pi^{-1}(j_k)} = A_{j_k}$ for $k = 1 \dots B/b$. That is, for a given set of rows of A and a split of $[B]$, we combine the rows into one row c such that each row occupies a separate part of the split. Note that this is possible, since splitter has all partitions of equal size, b. The created family then has size $|\mathcal{C}| = |\Pi||A|^{B/b} = \operatorname{poly}(B^{B/b})\exp(\frac{-s^2}{2(1-r/d)})$ as promised. Since the only explicit work we had to do was finding A, we have property (3) of the lemma.

We define the decoding function $C(x) \in \mathcal{C}$ for $x \in \{0,1\}^B$ with the following algorithm: For each $\pi \in \Pi$ compute the inner decodings $A_j = \{a \in A : \operatorname{dist}(x_{\pi^{-1}(j)}, a) \le b/2 - s\sqrt{b}/2\}$ for $j = 1, \dots, B/b$. Return the set of all concatenations in the product of the A_j's: $C(x) = \{a_1 \| a_2 \| \dots \| a_{B/b} : a_1 \in A_1, \dots\}$. Computing the A_j's take time $(B/b)|A|$, while computing and concatenating the product takes linear time in the size of the output. This shows property (1).

Finally for property (2), consider a pair $x, y \in \{0,1\}^B$ of distance $\le r$. Let I be the set of coordinates on which x and y differ. Then there is a function $\pi \in$

[1] The randomness is not essential, and we could as well formulate a set cover instance and solve it using the greedy algorithm, which matches the probabilistic construction up to a log factor in size and time.

Π such that x and y differ in at most $|I|b/B = rb/B$ coordinates in each subset $\pi^{-1}(1), \dots, \pi^{-1}(B/b) \subseteq [B]$. Now for each pair of projected vectors $x_{\pi^{-1}(1)}, y_{\pi^{-1}(1)}, \dots$ (let's call them $x_1, y_1, \dots$) there is an $a_j \in A$ such that $\operatorname{dist}(a_j, x_j) \leq b/2 - s'\sqrt{b}/2$ and $\operatorname{dist}(a_j, y_j) \leq b/2 - s'\sqrt{b}/2$. This means that x and y must both have distance at most $(b/2 - s'/2)B/b = B/2 - s\sqrt{B}/2$ to that $c \in \mathcal{C}$ which has $c_{\pi^{-1}(j)} = a_j$ for $j = 1 \dots B/b$. By the same reasoning, this c will be present in both $C(x)$ and $C(y)$, which proves the lemma. □

Returning to the problem of near neighbour search in $\{0,1\}^d$, it is clear from the $4^b \operatorname{poly}(B^{B/b})$ construction time of the above family, that it will not be efficient for dimension $B = (\log n)^{\omega(1)}$. For this reason we will apply the following dimensionality reduction lemma:

Lemma 3.3.3. *Given $d \geq cr > r \geq 1$ and $\epsilon, \delta > 0$, define $B = 27\epsilon^{-3} \log 1/\delta$ and $m = 3cr/\epsilon$ and assume $\delta^{-1} \geq m$, then there is a random set F of at most $S = m/B$ functions $f : \{0,1\}^d \to \{0,1\}^B$ with the following properties for every $x, y \in \{0,1\}^d$:*

1. *With probability 1, there is at least one $f \in F$ st.:*

$$\operatorname{dist}(f(x), f(y)) \leq \operatorname{dist}(x,y)/S.$$

2. *If $\operatorname{dist}(x,y) \geq cr$ then for every $f \in F$ with probability at least $1-\delta$:*

$$\operatorname{dist}(f(x), f(y)) \geq (1-\epsilon)cr/S.$$

The idea is to randomly throw the d coordinates in $m = 3cr/\epsilon$ buckets, (xor-ing the value if more than two coordinates land in the same group.) For two points with $\leq cr$ differences, this has the effect of rarely colliding two coordinates at which the points disagree, thus preserving distances. It further has the effect of changing the relative distances from arbitarily low r/d to something around ϵ, which allows partitioning the coordinates into groups of around $\epsilon^{-3} \log 1/\delta$ coordinates using Chernoff bounds.

Proof. To prove lemma 3.3.3 first notice that if $B \geq d$ we can use the identity function and we are done. If $B \geq m$, then we can duplicate the vector $\lceil m/B \rceil = O(\epsilon^{-2} \log 1/\delta)$ times. Also, by adjusting ϵ by a constant factor, we can assume that B divides m.

For the construction, pick a random function $h : [d] \to [m]$. Define $g : \{0,1\}^d \to \{0,1\}^m$ by 'xor'ing the contents of each bucket, $g(x)_i = \bigoplus_{j \in h^{-1}(i)} x_j$, and let $f_i(x) = g(x)_{(iB,(i+1)B]}$ for $i = 0 \dots m/B$ be the set of functions in the lemma. We first show that this set has property (1) and then property (2).

Observe that g never increases distances, since for any $x, y \in \{0,1\}^d$ the distance

$$\operatorname{dist}(g(x), g(y)) = \sum_{i=1}^{m} \left[\bigoplus_{j \in h^{-1}(i)} x_j \neq \bigoplus_{j \in h^{-1}(i)} y_j \right]$$

is just $\sum_{i=1}^{m} \left(\sum_{j \in h^{-1}(i)} [x_j \neq y_j] \bmod 2 \right)$ which is upper bounded by the number of coordinates at which x and y differ. By averaging, there must be one f_i such that $\operatorname{dist}(f_i(x), f_i(x)) \leq \operatorname{dist}(g(x), g(y))B/m \leq \operatorname{dist}(x,y)/S$.

For the second property, let $R = \text{dist}(x, y) \geq cr$ and let $X_1, \ldots, X_m$ be the random number of differences between x and y in each bucket under h. Let $Y_1, \ldots, Y_m$ be iid. Poisson distributed variables with mean $\lambda = \mathrm{E}\, X_1 = R/m \geq \epsilon/3$. We use the the Poisson trick from [131] theorem 5.7:

$$\Pr[\sum_{i=1}^{B}(X_i \bmod 2) < x] \leq e\sqrt{m}\Pr[\sum_{i=1}^{B}(Y_i \bmod 2) < x].$$

The probability $\Pr[Y \bmod 2 \equiv 1]$ that a Poisson random variable is odd is $(G(1) - G(-1))/2$ where $G(z) = \sum_i \Pr[Y = i]z^i = e^{\lambda(z-1)}$. This gives us the bound $\Pr[Y_i \bmod 2 \equiv 1] = (1 - e^{-2\lambda})/2 \geq (1 - e^{-2\epsilon/3})/2 \geq (1 - \epsilon/3)\epsilon/3$. We can then bound the probability of an f_i decreasing distances too much, using a Chernoff bound ($\Pr[Z \leq x] \leq \exp(-D[x/B \mid p]B)$):

$$\begin{aligned}
&\Pr[\text{dist}(f_i(x), f_i(y)) \leq (1-\epsilon)cr/S] \\
&\quad \leq e\sqrt{m}\exp(-D[(1-\epsilon)\epsilon/3 \mid (1-\epsilon/3)\epsilon/3]B) \\
&\quad \leq e\sqrt{m}\exp(-2\epsilon^3 B/27).
\end{aligned}$$

Since $cr/S = crB/m = B\epsilon/3$. Here $D[\alpha \mid \beta] = \alpha \log\frac{\alpha}{\beta} + (1-\alpha)\log\frac{1-\alpha}{1-\beta} \geq (\alpha - \beta)^2/(2\beta)$ is the Kullback–Leibler divergence. For our choice of B the error probability is then $e\sqrt{m}\delta^2$ which is less than δ by our assumptions. This proves the lemma. □

Using lemma 3.3.3 we can make at most $3cr/\epsilon = O(d/\epsilon)$ data structures, as described below, and be guaranteed that in one of them, we will find a near neighbour at distance $r' = r/S = \epsilon/(3c)B$. In each data structure we will have to reduce the distance cr', at which we expect far points to appear, to $cr'(1-\epsilon)$. This ensures we see at most a constant number of false positives in each data structure, which we can easily filter out. For $\epsilon = o(1)$ this change be swallowed by the approximation factor c, and won't significantly impair our performance.

When using the filter family of lemma 3.3.1 for LSF, the time usage for queries and inserting points is dominated by two parts: 1) The complexity of evaluating $C(x)$, and 2) The expected number of points at distance larger than $cr'(1-\epsilon)$ that falls in the same filter as x.

By randomly permuting and flipping the coordinates, we can assume that x is a random point in $\{0,1\}^d$. The expected time to decode $C(x)$ is then

$$\begin{aligned}
&\mathrm{E}\,|C(x)| + \text{poly}(B^{B/b}, e^{s^2b/B}) \\
&= |\mathcal{C}| \Pr_x[0 \in C(x)] + \text{poly}(B^{B/b}, e^{s^2b/B}) \\
&\leq \text{poly}(B^{B/b}, e^{s^2b/B}) \exp\left(\tfrac{s^2}{2(1-r'/B)} - \tfrac{s^2}{2}\right).
\end{aligned}$$

For estimating collisions with far points, we can similarly assume that x and y are random points in $\{0,1\}^d$ with fixed distance $cr'(1-\epsilon)$:

$$\begin{aligned}
&\mathrm{E}\,|\{y \in P : C(x) \cap C(y) \neq \emptyset\}| \\
&\leq n\,|\mathcal{C}| \Pr_{x,y}[0 \in C(x), 0 \in C(y)] \\
&\leq B^{O(B/b)} \exp\left(\tfrac{s^2}{2(1-r'/B)} - \tfrac{s^2}{2(1-c(1-\epsilon)r'/B)} + \log n\right) \\
&= B^{O(B/b)} \exp\left(\tfrac{s^2}{2}\left(\tfrac{1}{1-r'/B} - \tfrac{1}{1-cr'/B} + O(\epsilon)\right) + \log n\right).
\end{aligned}$$

Finally we should recall that constructing the data structures takes time $4^b \operatorname{poly}(e^{s^2 b/B})$ plus n inserts.

We now choose the parameters:

$$\begin{aligned} s^2/2 &= \tfrac{1-cr'/B}{cr'/B} \log n, & B &= 27\epsilon^{-3} \log n, \\ b &= \log_4 n, & \epsilon &= (\log n)^{-1/4}. \end{aligned}$$

This makes the code construction time $n^{1+o(1)}$ while evaluating $C(x)$ and looking at far points takes expected time at most $n^{1/c+\tilde{O}(\log n)^{-1/4}}$. To use lemma 3.3.1 we have to check that $s^2 = O(B/\sqrt{b}) = O((\log n)^{5/4})$, but $s^2/2 = \frac{1-\epsilon/3}{\epsilon/3} \log n = (\log n)^{5/4}(1 - o(1))$ so everything works out. This shows theorem 9.

To get the result of corollary 4, we just need to substitute the dimensionality reduction lemma 3.3.3 for a simple partitioning approach. (Lemma 3.3.4 below.) The idea is that of [148] which is to randomly partition the d coordinates in B parts and run the algorithm on those. The important point is that in this case $r'/B = r/d$, so the relative distance is not decreased. We choose parameters

$$\begin{aligned} s^2/2 &= \tfrac{1-cr/d}{cr/d} \log n & B &= O(\epsilon^{-2}(cr/d)^{-1} \log n), \\ b &= \log_4 n, & \epsilon &= (\log n)^{-1/3}. \end{aligned}$$

This again makes this makes the code construction time $n^{1+o(1)}$ while evaluating $C(x)$ and looking at far points takes time $n^{\frac{1-c\delta}{c(1-\delta)}+\tilde{O}(\log n)^{-1/3} d/r}$ as in the corollary. Again we need need to check that $s^2 = O(B/\sqrt{b}) = O((\log n)^{7/6})$. This works as long as $r/d = \Omega((\log n)^{-1/6})$, which is the assumption of the corollary.

Lemma 3.3.4. *For any $d \geq r \geq 1$ and $\epsilon > 0$ there is a set F of d/B functions, $f : \{0,1\}^d \to \{0,1\}^B$, where $B = 2\epsilon^{-2} d/(cr) \log n$, such that:*

1. *With probability 1, there is at least one $f \in F$ st.:*

$$\operatorname{dist}(f(x), f(y)) \leq \operatorname{dist}(x,y)\, B/d.$$

2. *If $\operatorname{dist}(x,y) \geq cr$ then for every $f \in F$ with probability at least $1 - 1/n$:*

$$\operatorname{dist}(f(x), f(y)) \geq (1-\epsilon) cr\, B/d.$$

Proof. Fix a random permutation. Given $x \in \{0,1\}^d$, shuffle the coordinates using the permutation. Let $f_1(x)$ be the first B coordinates of x, $f_2(x)$ the next B coordinates and so forth. For any $y \in \{0,1\}^d$, after shuffling, the expected number of differences in some block of B coordinates is $\operatorname{dist}(x,y)B/d$. Then the first property follows because $\sum_i \operatorname{dist}(f_i(x), f_i(y)) = \operatorname{dist}(x,y)$ so not all distances can be below the expectation. The second property follows from the Chernoff/Hoeffding bound [90]. □

3.4 Set Similarity Data Structure

We explore the generality of our methods, by making a Las Vegas version of another very common LSH data structure. Recall the theorem we are trying to prove, from the introduction:

Theorem 11. *Given a set P of n subsets of $[d]$, define the Braun-Blanquet similarity* $\mathrm{sim}(x, y) = |x \cap y| / \max(|x|, |y|)$ *on the elements of P. For every choice of* $0 < b_2 < b_1 < 1$ *there is a data structure on P that supports the following query operation:*

On input $q \subseteq [d]$, for which there exists a set $x \in P$ with $\mathrm{sim}(x, q) \geq b_1$*, return an $x' \in P$ with* $\mathrm{sim}(x', q) > b_2$*. The data structure uses expected time dn^ρ per query, can be constructed in expected time $dn^{1+\rho}$, and takes expected space $n^{1+\rho} + dn$ where* $\rho = \frac{\log 1/b_1}{\log 1/b_2} + \hat{O}(1/\sqrt{\log n})$.

By known reductions [63] we can focus on the case where all sets have the same weight, w, which is known to the algorithm. This works by grouping sets in data structures with sets of similar weight and uses no randomization. The price is only a $(\log n)^{O(1)}$ factor in time and space, which is dominated by the $n^{\hat{O}(1/\sqrt{\log n})}$ factor in the theorem.

When two sets have equal weight w, being b-close in Braun-Blanquet similarity coresponds exactly to having an intersection of size bw. Hence for the data structure, when trying to solve the (b_1, b_2)-approximate similarity search problem, we may assume that the intersections between the query and the 'close' sets we are interested in are at least $b_1 w$, while the intersections between the query and the 'far' sets we are not interested in are at most $b_2 w$.

The structure of the algorithm follows the LSF framework as in the previous section. A good filter family for set similarity turns out to be the r-element blocks of a Turán system. This choice is inspired by [63] who sampled subsets with replacement.

Definition 9 ([178, 68])**.** *A Turán (n, k, r)-system is a collection of r-element subsets, called 'blocks', of an n element set X such that every k element subset of X contains at least one of the blocks.*

Turán systems are well studied on their own, however all known general constructions are either only existential or of suboptimal size. The following lemma provides the first construction to tick both boxes, and with the added benefit of being efficiently decodable.

An interesting difference between this section and the last, is that we don't know how to do a dimensionality reduction like we did for hamming distance. Instead we are (luckily) able to make an efficiently decodable filter family even for very large dimensional data points.

Lemma 3.4.1. *For every n, k, r, where $n > k > r^{3/2}$, there exists a Turán (n, k, r)-system, $\mathcal{T}$, of size $|\mathcal{T}| \leq (n/k)^r e^\chi$ where $\chi = O(\sqrt{r} \log r + \log k + \log \log n)$ with the following properties:*

1. *Correctness: For every set $K \subseteq [n]$ of size at least k, there is at least one block $R \in \mathcal{T}$ such that $R \subseteq K$.*

2. *Efficient decoding: Given a set $S \subseteq [n]$, we can find all the blocks it contains $T(S) = \{R \in \mathcal{T} : R \subseteq S\}$ in time $|S||T(S)| + e^{\chi}$. Furthermore, $T(S)$ has expected size $\leq (|S|/k)^r e^{\chi}$.*

3. *Efficient construction: The system is constructible, implicitly, in $e^{r(1+o(1))}$ time and space.*

Notes: A simple volume lower bound shows that an (n, k, r)-system must have at least $\binom{n}{r}/\binom{k}{r} \geq (n/k)^r$ blocks, making our construction optimal up the factor e^{χ}. Using the sharper bound $\binom{n}{r}/\binom{k}{r} \approx (n/k)^r \exp(\frac{r^2}{2k})$ from lemma 3.6.1, we get that the factor $\exp(\Omega(\sqrt{r}))$ is indeed tight for $k = O(r^{3/2})$.

The size of the system is in 'expectation', which is sufficient for our purposes, but is in fact fairly easy to fix. On the other hand, the 'expectation' in the size of sets $T(S)$ seems harder to get rid of, which is the reason why the data strcuture is Las Vegas and not deterministic.

3.4.1 The algorithm

We continue to describe the algorithm and proving theorem 10 using the lemma. The proof of the lemma is at the end and will be the main difficulty of the section.

As discussed, by the reduction of [63] we can assume that all sets have weight w, intersections with close sets have size $\geq b_1 w$ and intersections with far sets have size $\leq b_2 w$. The data structure follows the LSF scheme as in the previous section. For filters we use a Turán $(d, b_1 w, \frac{\log n}{\log 1/b_2})$ design, constructed by lemma 3.4.1. Note that if $b_1 w < (\frac{\log n}{\log 1/b_2})^{3/2}$ ($k < r^{3/2}$ in the terms of the lemma), we can simply concatenate the vectors with themselves $O(\log n)$ times. If $b_1 w \leq \frac{\log n}{\log 1/b_2}$ we can simply use all the $\binom{d}{b_1 w}$ sets of size $b_1 w$ as a Turán $(d, b_1 w, b_1 w)$ system and get a fast deterministic data structure.

As in the previous section, given a dataset P of n subsets of $[d]$, we build the data structure by decoding each set in our filter system $\mathcal{T}$. We store pointers from each set $R \in \mathcal{T}$ to the elements of P in which they are a contained. By the lemma, this takes time $n(w(w/k)^r + 1)e^{\chi} = wn(1/b_1)^{\frac{\log n}{\log 1/b_2}} e^{\chi} \leq dn^{\rho}$, while expected space usage is $n(w/k)^r e^{\chi} + e^{r(1+o(1))} + dn = n^{\rho} + dn$ as in the theorem. Building $\mathcal{T}$ takes time $e^{r(1+o(1))} = n^{(1+o(1))/\log 1/b_2} = n^{1+o(1)}$.

Queries are likewise done by decoding the query set x in $\mathcal{T}$ and inspecting each point $y \in P$ for which there exists $R \in \mathcal{T}$ with $R \subseteq y$, until a point y' with $\text{sim}(x, y') > b_2$ is found. Let's call this the candidate set of x. The expected number of false positive points in the candidates is thus

$$\sum_{y \in P} \mathrm{E}[|\{R \in \mathcal{T} : R \subseteq x \cap y\}|] = \sum_{y \in P} \mathrm{E}[|T(x \cap y)|] \leq n(b_2 w/(b_1 w))^{\frac{\log n}{\log 1/b_2}} e^{\chi} = n^{\rho}.$$

Computing the actual similarity takes time w, so this step takes time at most $wn^{\rho} \leq dn^{\rho}$. We also have to pay for actually decoding $T(x)$, but that takes time $w(w/(b_1 w))^{\frac{\log n}{\log 1/b_2}} e^{\chi} + e^{\chi} \leq dn^{\rho}$ as well.

Finally, to see that the algorithm is correct, if $\mathrm{sim}(x,y) \geq b_1$ we have $|x \cap y| \geq b_1 w$, and so by the Turán property of $\mathcal{T}$ there is $R \in \mathcal{T}$ such that $R \subseteq x \cap y$ which implies $R \subseteq x$ and $R \subseteq y$. This shows that there will always be at least one true high-similarity set in the candidates, which proves theorem 10.

3.4.2 The proof of lemma 3.4.1

Proof. We first prove the lemma in four parts, starting with a small design and making it larger step by step. To more easily describe the small designs, define $a = kr^{-3/2}\log(r^{3/2})$ and $b = \sqrt{r}$. The steps are then

1. Making a $(k^2/(a^2b), k/(ab), r/b)$ using brute force methods.
2. Use splitters to scale it to $((k/a)^2, k/a, r)$.
3. Use perfect hashing to make it an $(n/a, k/a, r)$ design.
4. Use partitioning to finally make an (n, k, r) design.

We first prove the lemma without worrying about the above values being intergers. Then we'll show that each value is close enough to some integer that we can hide any excess due to approximation in the loss term.

The four steps can also be seen as proofs of the four inequalities:

$$\begin{aligned} T(n,k,r) &\leq \tbinom{n}{r}/\tbinom{k}{r}\left(1+\log\tbinom{n}{k}\right), \\ T(cn,ck,cr) &\leq \tbinom{cn}{c}\,T(n,k,r)^c, \\ T(ck^2,k,r) &\leq (k^4\log ck^2)c^r\,T(k^2,k,r), \\ T(cn,ck,r) &\leq c\,T(n,k,r). \end{aligned}$$

where the c are arbitrary integer constants > 0.

1. For convenience, define $n' = k^2/(a^2b)$, $k' = k/(ab)$, $r' = r/b$ and assume they are all intergers. Probabilistically we can build a Turán (n', k', r')-system, $\mathcal{T}^{(n')}$, by sampling

$$\ell = \tbinom{n'}{r'}/\tbinom{k'}{r'}(1+\log\tbinom{n'}{k'}) \leq (n'/k')^{r'}e^{r'^2/k'}(1+k'\log(en'/k')) = (n'/k')^{r'}r^{5/2}(1+o(1))$$

independent size r'-sets. (Here we used the bound on $\tbinom{n'}{r'}/\tbinom{k'}{r'}$ from lemma 3.6.1 in the appendix.) For any size k' subset, K, the probability that it contains none of the r'-sets is $\left(1-\tbinom{k'}{r'}/\tbinom{n'}{r'}\right)^{\ell} \leq e^{-1}/\tbinom{n'}{k'}$. Hence by the union bound over all $\tbinom{n'}{k'}$ K-sets, there is constant probability that they all contain an r'-set, making our $\mathcal{T}^{(n')}$ a valid system.

We can verify the correctness of a sampled system, naiively, by trying iterating over all $\tbinom{n'}{k'}$ K-sets, and for each one check if any of the R-sets is contained. This takes time bounded by

$$\begin{aligned} \tbinom{n'}{k'}\ell &\leq (en'/k')^{k'}(n'/k')^{r'}r^{5/2}(1+o(1)) \\ &= \left(\frac{er^{3/2}}{\log(r^{3/2})}\right)^{\frac{r}{\log(r^{3/2})}}\left(\frac{r^2}{\log(r^{3/2})}\right)^{\sqrt{r}} r^{O(1)} \\ &= e^{r+O(r/\log r)} \end{aligned}$$

as in the preprocessing time promised by the lemma. Since we had constant success probability, we can repeat the above steps an expected constant number of times to get a correct system.

Notice that the system has a simple decoding algorithm of brute-force iterating through all ℓ sets in $\mathcal{T}^{(n')}$.

2. To scale up the system, we proceed as in the previous section, by taking a splitter, Π, that is a set of functions $\pi : [bn'] \to [b]$ such that for any set $I \subseteq [bn']$ there is a $\pi \in \Pi$ such that

$$\lfloor |I|/b \rfloor \leq |\pi^{-1}(j) \cap I| \leq \lceil |I|/b \rceil \quad \text{for } j = 1, \dots, b.$$

In other words, each π partitions $[bn']$ in b sets $[bn'] = \pi^{-1}(1) \cup \dots \pi^{-1}(b)$ and for any subset $I \subseteq [bn']$ there is a partition which splits it as close to evenly as possible. We discuss the constructions of such sets of functions in the appendix.

For each $\pi \in \Pi$, and distinct $i_1, \dots, i_b \in [|\mathcal{T}^{(n')}|]$, we make a br'-set, $R \subseteq [bn']$, which we think of as an indicator vector $\in \{0,1\}^{bn'}$, such that $R_{\pi^{-1}(j)} = \mathcal{T}^{(n')}_{i_j}$ for $j = 1 \dots b$. That is, the new block, restricted to $\pi^{-1}(1), \pi^{-1}(2), \dots$, will be equal to the i_1th, i_2th, … blocks of $\mathcal{T}^{(n')}$. Another way to think of this is that we take the i_1th, i_2th, … blocks of $\mathcal{T}^{(n')}$ considered as binary vectors in $\{0,1\}^{n'}$ and combine them to a bn' block 'spreading' them using π.

The new blocks taken together forms a family $\mathcal{T}^{(bn')}$ of size

$$|\mathcal{T}^{(bn')}| = |\Pi||\mathcal{T}^{(n')}|^b = \tbinom{bn'}{b}[(n'/k')^{r'} r^{O(1)}]^b \leq (en')^b (n'/k')^{br'} r^{O(b)} = (n'/k')^{br'} r^{O(b)},$$

where we used only the trivial bound $\binom{n}{k} \leq (en/k)^k$ and the fact that $n' = r^{O(1)}$.

We now show correctness of $\mathcal{T}^{(bn')}$. For this, consider any bk'-set $K \subseteq [bn']$. We need to show that there is some br'-set $R \in \mathcal{T}^{(bn')}$ such that $R \subseteq K$. By construction of Π, there is some $\pi \in \Pi$ such that $|\pi^{-1}(j) \cap K| = k'$ for all $j = 1, \dots, b$. Considering K as an indicator vector, we look up $K_{\pi^{-1}(1)}, \dots, K_{\pi^{-1}(b)}$ in $\mathcal{T}^{(n')}$, which gives us b disjoint r'-sets, $R'_1, \dots, R'_b$. By construction of $\mathcal{T}^{(bn')}$ there is a single $R \in \mathcal{T}^{(bn')}$ such that $R_{\pi^{-1}(j)} = R'_j$ for all j. Now, since $R'_j \subseteq K_{\pi^{-1}(j)}$ for all j, we get $R \subseteq K$ and so we have proven $\mathcal{T}^{(bn')}$ to be a correct (bn', bk', br')-system.

Decoding $\mathcal{T}$ is fast, since we only have to do $|\Pi| \cdot b$ lookups in (enumerations of) $\mathcal{T}^{(n')}$. When evaluating $T^{(bn')}(K)$ we make sure we return every br'-set in K. Hence we return the entire "product" of unions:

$$T^{(bn')}(K) = \bigcup_{\pi \in \Pi} \{R_1 \cup \dots \cup R_b : R_1 \in T^{(n')}(K_{\pi^{-1}(1)}), R_2 \in \dots\}.$$

In total this takes time $|T^{(bn')}(K)|$ for the union product plus an overhead of $|\Pi| b |\mathcal{T}^{(n')}| \leq (en')^b r^{O(r'+b)} = r^{O(\sqrt{r})}$ for the individual decodings.

3. Next we convert our $((k/a)^2, k/a, r)$ design, $\mathcal{T}^{(bn')}$ (note that $bn' = (k/a)^2$), to a $(n/a, k/a, r)$ design, call it $\mathcal{T}^{(n/a)}$.

Let $\mathcal{H}$ be a perfect hash family of functions $h : [n/a] \to [(k/a)^2]$, such that for every k/a-set, $S \subseteq [n/a]$, there is an $h \in \mathcal{H}$ such that $|h(S)| = k/a$. That is, no element of S gets mapped to the same value. By lemma 3 in [16], we can efficiently construct such a family of $(k/a)^4 \log(n/a)$ functions.

We will first describe the decoding function $T^{(n/a)} : \mathcal{P}([n/a]) \to \binom{[n/a]}{k/a}$, and then let $\mathcal{T}^{(n/a)} = T^{(n/a)}([n/a])$. For any set $S \subseteq [n/a]$ to be decoded, for each $h \in \mathcal{H}$, we evaluate $T^{(bn')}(h(S))$ to get all r-sets $R \in \mathcal{T}^{(bn')}$ where $R \subseteq h(S)$. For each such set, we return each set in

$$(h^{-1}(R_1) \cap S) \times (h^{-1}(R_2) \cap S) \times \cdots \times (h^{-1}(R_r) \cap S),$$

where R_i is the ith element of R when considered as a $[bn']^r$ vector.

This takes time equal to the size of the above product (the number of results, $|T(S)|$) plus an overhead of $|\mathcal{H}|$ times the time to decode in $\mathcal{T}^{(bn')}$ which is $|\mathcal{H}|r^{O(\sqrt{r})} = e^{\mathcal{X}}$ by the previous part. The other part of the decoding time, the actual size $|T^{bn'}(h(S))|$, is dominated by the size of the product. To prove the the 'efficient decoding' part of the lemma we thus have to show that the expected size of $T(S)$ is $\le (|S|a/k)^r e^{\mathcal{X}}$ for any $S \subseteq [n/a]$. (Note: this is for a set $S \subseteq [n/a]$, part four of the proof will extend to sets $S \subseteq [n]$ and that factor a in the bound will disappear.)

At this point we will add a random permutation, $\pi : [(k/a)^2] \to [(k/a)^2]$, to the preprocessing part of the lemma. This bit of randomness will allow us to consider the perfect hash-family as 'black box' without worrying that it might conspire in a worst case way with our fixed family $\mathcal{T}^{(bn')}$. We apply this permutation to each function of $\mathcal{H}$, so we are actually returning

$$T^{(n/a)}(S) = \bigcup \left\{ \begin{matrix} (h^{-1}(\pi^{-1}R_1) \cap S) \times (h^{-1}(\pi^{-1}R_2) \cap S) \times \cdots \times (h^{-1}(\pi^{-1}R_r) \cap S) \\ \hfill : \text{for all } R \in T^{(bn')}(\pi h(S)) \text{ and } h \in \mathcal{H}. \end{matrix} \right\}$$

We can then show for any $S \subseteq [n/a]$:

$$\begin{aligned}
E_\pi[|T^{(n/a)}(S)|] &= E_\pi \left[\sum_{h \in \mathcal{H},\, R \in T^{(bn')}(\pi h(S))} \left| (h^{-1}(\pi^{-1}R_1) \cap S) \times \cdots \times (h^{-1}(\pi^{-1}R_r) \cap S) \right| \right] \\
&= \sum_{h \in \mathcal{H},\, R \in \mathcal{T}^{(bn')}} E_\pi \left[|(h^{-1}(\pi^{-1}R_1) \cap S)| \cdots |(h^{-1}(\pi^{-1}R_r) \cap S)| \cdot [R \subseteq \pi h(S)] \right] \\
&= |\mathcal{T}^{(bn')}| \sum_{h \in \mathcal{H}} E_\pi \left[|(h^{-1}(\pi^{-1}R_1) \cap S)| \cdots |(h^{-1}(\pi^{-1}R_r) \cap S)| \right] && (3.1) \\
&\le |\mathcal{T}^{(bn')}| \sum_{h \in \mathcal{H}} E_\pi \left[|h^{-1}(\pi_1) \cap S| \right]^r && (3.2) \\
&= |\mathcal{T}^{(bn')}|\, |\mathcal{H}|\, (|S|/(k/a)^2)^r \\
&= (|S|a/k)^r e^{\mathcal{X}}.
\end{aligned}$$

For eq. (3.1) we used that

$$[R \subseteq h(S)] = [\forall_{r \in R} r \in h(S)] = [\forall_{r \in R} h^{-1}(r) \cap S \ne \emptyset] \qquad (3.3)$$

and so the value in the expectation was already 0 exactly when the Iversonian bracket was zero.

For eq. (3.2) we used the Maclaurin's Inequality [44] which says that $E(X_1 X_2 \dots X_r) \le (EX_1)^r$ when the X_is are values sampled identically, uniformely at random without replacement from som set of non-negative values. In our case those values were sizes of partitions $h^{-1}(1) \cap S, \dots, h^{-1}(bn') \cap S$, which allowed us to bound the expectation as if h had been a random function.

We need to show that $T^{(n/a)}$ is a correct decoding function, that is $T^{(n/a)}(S) = \{R \in \mathcal{T}^{(n/a)} : R \subseteq S\}$, and the correctness of $\mathcal{T}^{(n/a)}$, that is $|S| \ge k/a$ implies $T^{(n/a)}(S) \ne \emptyset$.

For this, first notice that T is monotone, that is if $S \subseteq U$ then $T(S) \subseteq T(U)$. This follows because $R \subseteq \pi h(S) \implies R \subseteq \pi h(U)$ and that each term $h^{-1}(R_i) \cap S$ is monotone in the size of S. This means we just need to show that $T(K)$ returns something for every $|K| = k$, since then $\mathcal{T} = T([n/a]) = T(\bigcup_K K) \supseteq \bigcup T(K)$ will return all these things.

Hence, consider a k-set, $K \subseteq [n/a]$. By the property of $\mathcal{H}$, there must be some $h \in \approx H$ such that $|h(K)| = k$, and by correctness of $\mathcal{T}^{(bn')}$ we know there is some r-set, $R \in T^{(bn')}(h(K))$. Now, for these h and R, since $R \subseteq h(K)$ and using eq. (3.3) we have that $(h^{-1}(R_1) \cap K) \times \dots$ is non-empty, which is what we wanted. Conversely, consider some $R \in \mathcal{T}^{(n/a)} = T^{(n/a)}([n/a])$ such that $R \subseteq K$, then $R \in h^{-1}(R'_1) \times h^{-1}(R'_2) \dots$ for some $R' \in \mathcal{T}^{(bn')}$ and $h(R) \subseteq h(K)$. However $h(R)$ is exactly R', since $R_i \in h^{-1}(R'_i) \implies h(R_i) = R'_i$, which shows that $T^{(n/a)}(K)$ returns all the set we want.

4. Finally we convert our $(n/a, k/a, r)$ design, $\mathcal{T}^{(n/a)}$ to an (n,k,r) design, call it $\mathcal{T}$. We do this by choosing a random permutation $\pi : [n] \to [n]$ and given any set $S \subseteq [n]$ we decode it as

$$T(S) = T^{(n/a)}(\pi S \cap \{1, \dots, n/a\}) \cup \dots \cup T^{(n/a)}(\pi S \cap \{n - n/a + 1, \dots, n\}).$$

To see that this is indded an (n,k,r) design, consider any set $K \subseteq [n]$ of size $|K| = k$, there must be one partition $K \cap \{1 \dots, n/a\}, \dots$ that has at least the average size k/a. Since $\mathcal{T}^{(n/a)}$ is a $(n/a, k/a, r)$ design, it will contain a set $R \subseteq K \cap \{in - n/a + 1, \dots, in\} \subseteq K$ which we return.

It remains to analyze the size of $T(S)$, which may of course get large if we are so unlucky that π places much more than the expected number of elements in one of the partitions. In expectation this turns out to not be a problem, as we can see as follows:

$$\begin{aligned}
E_\pi|T(S)| &= \sum_i E_\pi \left[\left|T^{(n/a)}(\pi S \cap p_i)\right|\right] \\
&= a\sum_s E\left[\left|T^{(n/a)}(\pi S \cap p_1)\right| \,\middle|\, |\pi S \cap p_1| = s\right] \Pr[|\pi S \cap p_1| = s] \\
&= a\sum_s (sa/k)^r e^\chi \binom{|S|}{s}\binom{n-|S|}{n/a-s} \Big/ \binom{n}{n/a} \\
&\le e^\chi \sum_s \frac{\binom{s}{r}}{\binom{k/a}{r}} \frac{\binom{|S|}{s}\binom{n-|S|}{n/a-s}}{\binom{n}{n/a}} \\
&= e^\chi \frac{\binom{|S|}{r}}{\binom{k/a}{r}\binom{n}{n/a}} \sum_s \binom{|S|-r}{s-r}\binom{n-|S|}{n/a-s} \\
&= e^\chi \frac{\binom{|S|}{r}\binom{n-r}{n/a-r}}{\binom{k/a}{r}\binom{n}{n/a}} = e^\chi \frac{\binom{|S|}{r}\binom{n/a}{r}}{\binom{k/a}{r}\binom{n}{r}} \\
&\le e^\chi (|S|a/k)^r e^{r^2/(k/a)} a^{-r} = (|S|/k)^r e^\chi.
\end{aligned}$$

Here we used Vandermonde convolution to complete the sum over s, and then eventually lemma 3.6.1 to bound the binomial ratios. This completes the proof of lemma 3.4.1. □

3.4.3 Integrality considerations

In the proof, we needed the following quantities to be integral: $b = r/b = \sqrt{r}$, $a = kr^{-3/2}\log(r^{3/2})$, $k^2/(a^2b) = k/a = r^{3/2}/\log(r^{3/2})$, $k/(ab) = r/\log(r^{3/2})$, n/a.

It suffices to have $\sqrt{r}$ and $r/\log(r^{3/2})$ integral, and that the later divides k.

It is obviously ridiculous to require that r is a square number. Or is it? You can actually make a number square by just changing it by a factor $1 + 2/\sqrt{r}$. That would only end up giving us an $e^{O(\sqrt{r})}$, so maybe not so bad?

To make $r/\log(r^{3/2})$ integral, we can multiply with a constant. Since it didn't matter that we divided by a log, surely it doesn't matter that we multiply with a constant.

To make $r/\log(r^{3/2})$ divide k, we need k to have some divisors. We can't just round k to say, a power of two, since that could potentially change it by a constant factor, which would come out of $(n/k)^r$. We can change k with at most $1 + 1/\sqrt{r}$. So $1 + 1/\sqrt{k}$ would be just fine. Of course we can change it by an additive $r/\log(r^{3/2})$. That corresponds to a factor about $1 + r/k$. Since $k > r^{3/2}$ that is fine! Or maybe we'll subtract that, because then it is still a valid (n, k, r) design. In the same way, if we round r up to the nearest square root, we don't have to make the changes in the later calculations, but they can be kept intirely inside the lemma.

3.5 Conclusion and Open Problems

We have seen that, perhaps surprisingly, there exists a relatively general way of creating efficient Las Vegas versions of state-of-the art high-dimensional search data structures.

As bi-products we found efficient, explicit constructions of large Turán systems and covering codes for pairs. We also showed an efficient way to do dimensionality reduction in hamming space without false negatives.

The work leaves a number of open questions for further research:

1) Can we make a completely deterministic high-dimensional data structure for the proposed problems? Cutting the number of random bits used for Las Vegas guarantees would likewise be interesting. The presented algorithms both use $O(d \log d)$ bits, but perhaps limited independence could be used to show that $O(\log d)$ suffice?

2) Does there exist Las Vegas data structures with performance matching that of data-dependent LSH data structures? This might connect to the previous question, since a completely deterministic data structure would likely have to be data-dependent. However the most direct approach would be to repeat [25], but find Las Vegas constructions for the remaining uses of Monte Carlo randomness, such as clustering.

3) By reductions, our results extend to ℓ_2 and ℓ_1 with exponent $n^{1/c}$. This is optimal for ℓ_1, but for ℓ_2 we would hope to get n^{1/c^2}. Can our techniques be applied to yield such a data structure? Are there other interesting LSH algorithms that can be made Las Vegas using our techniques? The author conjectures that a space/time trade-off version of the presented algorithm should follow easily following the approach of [25, 115, 61].

4) In the most general version, we we get the overhead term $(\log n)^{-1/4}$ in our ρ value. Some previous known LSH data structures also had large terms, such as [20], which had a $(\log n)^{-1/3}$ term and [25], which has $(\log \log n)^{-\Theta(1)}$, but in general most algorithms have at most a $(\log n)^{-1/2}$ term.

   Can we improve the overhead of the approach given in this paper? Alternatively, is there a completely different approach, that has a smaller overhead?

3.5.1 Acknowledgements

The author would like to thank Rasmus Pagh, Tobias Christiani and the members of the Scalable Similarity Search project for many rewarding discussions on derandomization and set similarity data structures. Further thanks to Omri Weinstein, Rasmus Pagh, Martin Aumüller, Johan Sivertsen and the anonymous reviewers for useful comments on the manuscript; and to the people at University of Texas, Austin, for hosting me while doing much of this work. An extra thanks to Piotr Wygocki for pointing out the need for a deterministic reduction from ℓ_1 to Hamming space.

The research leading to these results has received funding from the European Research Council under the European Union's 7th Framework Programme (FP7/2007-2013) / ERC grant agreement no. 614331.

3.6 Appendix

3.6.1 Explicit reduction from ℓ_1 to Hamming

Theorem 12. *For $d, r, c \geq 1$ and a set of points $P \subseteq \mathbb{R}^d$ of size $|P| = n$, there is a function $f : \mathbb{R}^d \to \{0,1\}^b$ where $b = 2d^2\epsilon^{-3}\log n$, such that for any two points $x \in \mathbb{R}^d, y \in P$,*

1. *if $\|x - y\|_1 \leq r$ then $\|f(x) - f(y)\|_1 \leq (1+\epsilon)S$,*
2. *if $\|x - y\|_1 \geq cr$ then $\|f(x) - f(y)\|_1 \geq (1-\epsilon)cS$,*

and the scale factor $S = b/(2dcr) = (d\log n)/(cr\epsilon^3)$.

Proof. First notice that we can assume all coordinates are at most rn. This can be assured by imposing a grid of side length $2rn$ over the points of P, such that no point is closer than distance r from a cell boundary. Since points $x, y \in \mathbb{R}^d$ in different cells must now be more than distance r from each other, we can set the embedded distance to cS by prefixing points from different cells with a different code word. The grid can be easily found in time $O(dn)$ by sweeping over each dimension seperately.

Notice that for actual data structure purposes, we can even just process each cell seperately and don't have to worry about separating them.

By splitting up each coordinate into positive and negative parts, we can further assume each coordinate of each vector is positive. This costs a factor of 2 in d.

Next, if we have an $2\epsilon r/d$ grid, then there is always a grid point within ℓ_1-distance ϵr. That means if we multiply each coordinate by $d/(2\epsilon r)$ and round the coordinates to nearest integer, we get distances are changed by at most ϵr.

We are now ready for the main trick of the reduction. Let M be the largest coordinate, which we can assume is at most dn/ϵ, and $R = dc/(2\epsilon)$ be the value of cr after scaling and rounding. For each coordinate we map $[M] \to [\lfloor M/R \rfloor]^R$ by $h(c) := \langle \lfloor \frac{x}{R} \rfloor, \lfloor \frac{x+1}{R} \rfloor, \dots, \lfloor \frac{x+R-1}{R} \rfloor \rangle$. The point of this mapping is that for every $c_1, c_2 \in [M]$, $\text{dist}(h(c_1), h(c_2)) = \min(|c_1 - c_2|, R)$, where dist is hamming distance.

100	12	12	12	12	13	13	13	13
105	13	13	13	13	13	13	13	14
	*	*	*	*				*

Figure 3.1: Mapping 100 and 105 to $[\lfloor 100/8 \rfloor]^8$ preserving ℓ_1 distance in Hamming distance.

All that's left is to use a code with good minimum and maximum distance to map down into $\{0,1\}$. A random code with bit length $k = 4\epsilon^{-2}(\log 4n)$ suffices. To see this, let X be a binomial random variable, $X \sim B(k, 1/2)$. Then

$$\Pr[(1-\epsilon)k/2 \leq C \leq (1+\epsilon)k/2] \leq 2e^{-\epsilon^2 k/2} \leq 1/(8n^2)$$

so by union bound over all $\binom{M/R}{2} \leq 2n^2$ pairs of values, we have constant probability that the code works. For a given code, we can check this property deterministically in time kn^2, so we can use rejection sampling and generate the code in time $\approx O(n^2)$. Of course, n^2 time may be too much. Luckily there are also explicit codes with the property, such as those by Naor and Naor [135].

The complete construction follows by concatenating the result of h on all coordinates. □

See [94] for an explicit reduction from ℓ_2 to ℓ_1.

3.6.2 The Ratio of Two Binomial Coefficients

Classical bounds for the binomial coefficient: $(n/k)^k \leq \binom{n}{k} \leq (en/k)^k$ give us simple bounds for binomial ratios, when $n \geq m$: $(n/em)^k \leq \binom{n}{k}/\binom{m}{k} \leq (en/m)^k$. The factor e on both sides can often be a nuisance.

Luckily tighter analysis show, that they can nearly always be either removed or reduced. Using the fact that $\frac{n-i}{m-i}$ is increasing in i for $n \geq m$, we can show $\binom{n}{k}/\binom{m}{k} = \prod_{i=0}^{k-1} \frac{n-i}{m-i} \geq \prod_{i=0}^{k-1} \frac{n}{m} = \left(\frac{n}{m}\right)^k$. This is often sharp enough, but on the upper bound side, we need to work harder to get results.

Let $\mathrm{H}(x) = x \log 1/x + (1-x)\log 1/(1-x)$ be the binary entropy function,

Lemma 3.6.1. *For $n \geq m \geq k \geq 0$ we have the following bounds:*

$$\left(\frac{n}{m}\right)^k \leq \left(\frac{n}{m}\right)^k \exp\left(\frac{n-m}{nm}\frac{k(k-1)}{2}\right) \leq \binom{n}{k}\bigg/\binom{m}{k} \leq \frac{\exp\left(n\,\mathrm{H}(k/n)\right)}{\exp\left(m\,\mathrm{H}(k/m)\right)} \leq \left(\frac{n}{m}\right)^k e^{k^2/m}$$

If $m \geq n$ we can simply flip the inequalities and swap n for m. Note that $(n/em)^k \leq (n/m)^k$ and $e^{k^2/m} \leq e^k$, so the bounds strictly tighten the simple bounds states above.

Especially the entropy bound is quite sharp, since we can also show: $\binom{n}{k}/\binom{m}{k} \geq \frac{\exp((n+1)\,\mathrm{H}(k/(n+1)))}{\exp((m+1)\,\mathrm{H}(k/(m+1)))}$, though for very small values of k, the lower bound in the theorem is actually even better. We can also get a feeling for the sharpness of the bounds, by considering the series expansion of the entropy bound at $k/m \to 0$: $\frac{\exp(n\,\mathrm{H}(k/n))}{\exp(m\,\mathrm{H}(k/m))} = \left(\frac{n}{m}\right)^k \exp(\frac{n-m}{nm}\frac{k^2}{2} + O(k^3/m^2))$.

For the proofs, we'll use some inequalities on the logarithmic function from [176]:

$$\log(1+x) \geq x/(1+x) \tag{3.4}$$

$$\log(1+x) \geq 2x/(2+x) \text{ for } x \geq 0 \tag{3.5}$$

$$\log(1+x) \leq x(2+x)/(2+2x) \text{ for } x \geq 0. \tag{3.6}$$

In particular eq. (3.5) and eq. (3.6) imply the following bounds for the entropy function:

$$\mathrm{H}(x) \leq x \log 1/x + x(2-x)/2 \tag{3.7}$$

$$\mathrm{H}(x) \geq x \log 1/x + 2x(1-x)/(2-x), \tag{3.8}$$

which are quite good for small x.

We'll prove theorem 3.6.1 one inequality at a time, starting from the left most:

Proof. The first inequality follows simply from $\frac{n-m}{nm}\frac{k(k-1)}{2} \geq 0$, which is clear from the conditions on $n \geq m \geq k$.

The second inequality we prove by using eq. (3.4), which implies $1 + x \geq \exp(x/(1+x))$, to turn the product into a sum:

$$\begin{aligned}
\binom{n}{k}\Big/\binom{m}{k} &= \prod_{i=0}^{k-1} \frac{n-i}{m-i} \\
&= \left(\frac{n}{m}\right)^k \prod_{i=0}^{k-1} \frac{1-i/n}{1-i/m} \\
&= \left(\frac{n}{m}\right)^k \prod_{i=0}^{k-1} \left(1 + \frac{i/m - i/n}{1-i/m}\right) \\
&\geq \left(\frac{n}{m}\right)^k \exp\left(\sum_{i=0}^{k-1} \frac{i(n-m)}{(n-i)m}\right) \\
&\geq \left(\frac{n}{m}\right)^k \exp\left(\sum_{i=0}^{k-1} i\frac{n-m}{nm}\right) \\
&= \left(\frac{n}{m}\right)^k \exp\left(\frac{k(k-1)}{2}\frac{n-m}{nm}\right).
\end{aligned}$$

For the entropy upper bound we will use an integration bound, integrating $\log(n-i)/(m-i)$ by parts:

$$\begin{aligned}
\binom{n}{k}\Big/\binom{m}{k} &= \prod_{i=0}^{k-1} \frac{n-i}{m-i} \\
&= \exp\left(\sum_{i=0}^{k-1} \log \frac{n-i}{m-i}\right) \\
&\leq \exp\left(\int_0^k 1 \log \frac{n-x}{m-x} dx\right) \\
&= \exp\left(x \log \frac{n-x}{m-x}\bigg|_0^k - \int_0^k x\left(\frac{1}{m-x} - \frac{1}{n-x}\right) dx\right) \\
&= \exp\left(k \log \frac{n-k}{m-k} + \int_0^k \left(\frac{m}{m-x} - \frac{n}{n-x}\right) dx\right) \\
&= \exp\left(k \log \frac{n-k}{m-k} - \bigg| m \log \frac{1}{m-x} - n \log \frac{1}{n-x}\bigg|_0^k\right) \\
&= \exp\left(n\,\mathrm{H}(k/n) - m\,\mathrm{H}(k/m)\right).
\end{aligned}$$

The integral bound holds because $\log\frac{n-i}{m-i}$ is increasing in i, and so $\log\frac{n-i}{m-i} \leq \int_i^{i+1} \log\frac{n-x}{m-x}dx$. We see that $\frac{n-i}{m-i}$ is increasing by observing $\frac{n-i}{m-i} = \frac{n}{m} + \frac{in/m-i}{m-i}$ where the numerator and denominator of the last fraction are both positive. The entropy lower bound, mentioned in the discussion after the theorem, follows similarly from integration, using $\log\frac{n-i}{m-i} \geq \int_{i-1}^{i} \log\frac{n-x}{m-x}dx$.

For the final upper bound, we use the bounds eq. (3.7) and eq. (3.8) on $\mathrm{H}(k/n)$ and $\mathrm{H}(k/m)$ respectively:

$$\frac{\exp\left(n\,\mathrm{H}(k/n)\right)}{\exp\left(m\,\mathrm{H}(k/m)\right)} \leq \left(\frac{n}{m}\right)^k \exp\left(\frac{k^2}{2}\left(\frac{1}{m-k/2}-\frac{1}{n}\right)\right) \leq \left(\frac{n}{m}\right)^k \exp\left(\frac{k^2}{m}\right).$$

□

3.6.3 Proof of lemma 3.3.2

	$\overbrace{\qquad\qquad}^{d-r}$	$\overbrace{\qquad\qquad}^{r}$
x	0	0
y	0	1
z	j	i

Figure 3.2: To calculate how many points are within distance t from two points x and y, we consider without loss of generality $x = 0\ldots0$. For a point, z, lying in the desired region, we let i specify the number of 1's where x and y differ, and j the number of 1's where they are equal. With this notation we get $d(x,z) = i+j$ and $d(y,z) = j+r-i$.

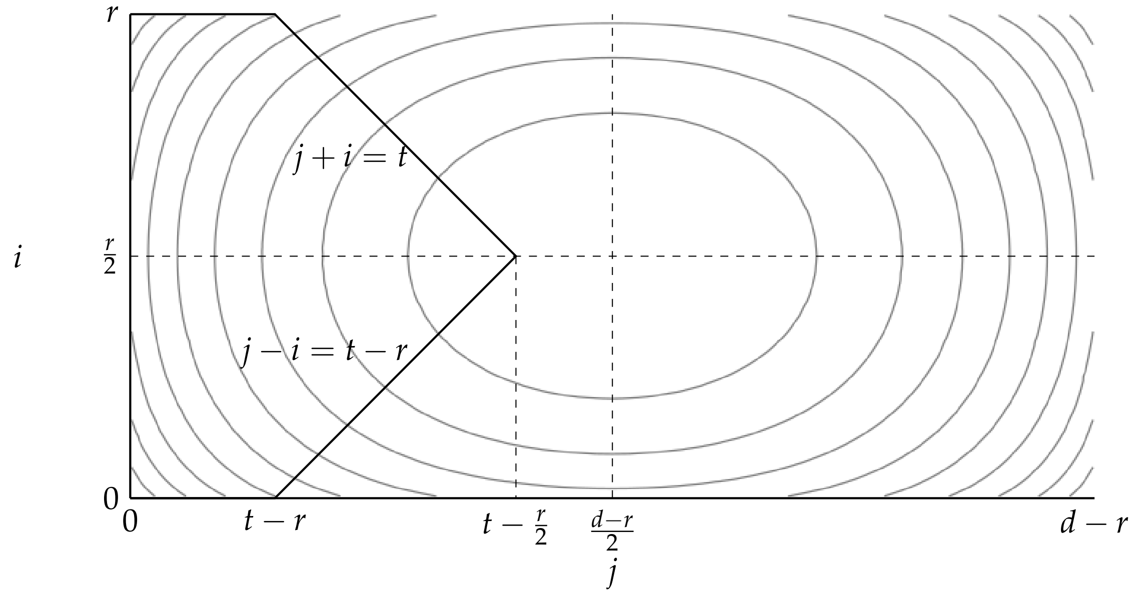


Figure 3.3: A contour plot over the two dimensional binomial. The pentagon on the left marks the region over which we want to sum. For the upper bound we sum i from 0 to r and j from 0 to $t-r/2$.

Proof. From figure 3.2 we have that $I = \sum_{\substack{i+j\leq t\\ j-i\leq t-r}} \binom{r}{i}\binom{d-r}{j}$, and from monotonicity (and figure 3.3) it is clear that $\binom{r}{r/2}\binom{d-r}{t-r/2} \leq I \leq \sum_{\substack{0\leq i\leq r\\ 0\leq j\leq t-r/2}} \binom{r}{i}\binom{d-r}{j}$.

We expand the binomials using Stirling's approximation: $\frac{\exp(nH(k/n))}{\sqrt{8(1-k/n)k}} \leq \binom{n}{k} \leq \sum_{i \leq k} \binom{n}{i} \leq \exp(nH(k/n))$ where $H(x) = x \log \frac{1}{x} + (1-x) \log \frac{1}{1-x}$ is the binary entropy function, which we bound as $\log 2 - 2(\frac{1}{2} - x)^2 - 4(\frac{1}{2} - x)^4 \leq H(x) \leq \log 2 - 2(\frac{1}{2} - x)^2$. We then have for the upper bound:

$$I2^{-d} \leq 2^{r-d} \exp\left[(d-r)H\left(\tfrac{t-r/2}{d-r}\right)\right] \leq \exp\left[-\tfrac{s^2}{2(1-r/d)}\right]$$

And for the lower bound:

$$\begin{aligned} I2^{-d} \geq 2^{-d}\binom{r}{r/2}\binom{d-r}{t-r/2} &\geq \frac{2^{r-d}}{\sqrt{2r}} \frac{\exp\left[(d-r)H\left(\tfrac{t-r/2}{d-r} - \log 2\right)\right]}{\sqrt{8(1-\tfrac{t-r/2}{d-r})(t-r/2)}} \\ &\geq \exp[-\tfrac{s^2}{2(1-r/d)}] \frac{\exp[-\tfrac{s^4}{4(1-r/d)^3 d}]}{\sqrt{4r(d-r)(1-\tfrac{ds^2}{(d-r)^2})}} \\ &\geq \exp[-\tfrac{s^2}{2(1-r/d)}] \frac{1}{d} \frac{1-2s^4/d}{\sqrt{1-4s^2/d}}, \end{aligned}$$

where for the last inequality we used the bound $e^x \geq 1 + x$. The last factor is monotone in s and we see that for $s \leq d^{1/4}/2$ it is $\geq \frac{7}{8}\left(1 - 1/\sqrt{d}\right)^{-1/2} \geq \frac{7}{8}$, which gives the theorem. □

The factor of $1/d$ can be sharpened a bit, e.g. by using the two dimensional Berry-Essen theorem from [45].

Chapter 4

Parameter-free Locality Sensitive Hashing for Spherical Range Reporting

Originally published in: Proceedings of Symposium on Discrete Algorithms, SODA 2017
Joint work with: Rasmus Pagh, Martin Aumüller

4.1 Introduction

Range search is a central problem in computational geometry, see e.g. the survey [7]. Given a set S of n points in $\mathbb{R}^d$, the task is to build a data structure that answers queries of the following type: For a region R (from a predefined class of regions), *count* or *report* all points from S that belong to R. Examples for such classes of regions are simplices [127], halfspaces [54], and spheres [33].

In this paper we study the *spherical range reporting (SRR) problem*: Given a distance parameter r and a point set S, build a data structure that supports the following queries: Given a point q, report all points in S within distance r from q. This problem is closely related to *spherical range counting* ("return the number of points") and *spherical range emptiness* ("decide whether there is a point at distance at most r"). Solving spherical range searching problems in time that is truly sublinear in the point set size $n = |S|$ seems to require space exponential in the dimensionality of the point set S. This phenomenon is an instance of the *curse of dimensionality*, and is supported by popular algorithmic hardness conjectures (see [15, 186]).

For this reason, most algorithms for range searching problems involve approximation of distances: For some approximation parameter $c > 1$ we allow the data structure to only distinguish between distance $\leq r$ and $> cr$, while points at distance in between can either be reported or not. We refer to this relaxation as c-approximate SRR. Approximate range reporting and counting problems were considered by Arya et al. in [33], by Indyk in his Ph.D. thesis [92] as "enumerating/counting point locations in equal balls" and by Andoni in his Ph.D. thesis [17] as "randomized R-near neighbor reporting". In low dimensions, tree-based approaches allow us to build efficient data structures with space usage $\tilde{O}(n\gamma^{d-1}(1 + (c-1)\gamma^2))$ and query time $\tilde{O}(1/((c-1)\gamma)^{d-1})$ for an approximation factor $1 < c \leq 2$ and trade-off parameter $\gamma \in [1, 1/(c-1)]$, see [33]. The exponential dependency of time and/or space on the dimension makes these algorithms inefficient in high dimensions.

Our approach uses the *locality-sensitive hashing* (LSH) framework [96], which hashes points into some smaller space such that close points are more likely to hash to the same value than distant points. We will introduce in Section 4.2. Using this technique to solve SRR is not new: Both Indyk [92] and Andoni [17] described extensions of the general LSH framework to solve this problem. As we will show, their approaches yield running times of $\Omega(tn^\rho)$, where t is the number of elements at distance at most cr from the query, and $\rho \in (0,1)$ is a parameter that depends on the distance r, the approximation factor c, and the LSH family used to build the data structure. When the output size t is large this leads to running time $\Omega(n^{1+\rho})$, which is worse than a linear scan! Indyk [92] also describes a reduction from spherical range counting to the (c,r)-approximate near neighbor problem that asks to report a *single* point from the result set of c-approximate SRR. The reduction uses $O(\log^2 n/(c-1)^3)$ queries of independently built (c,r)-ANN data structures, giving a running time of $O(n^\rho \log^2 n/(c-1)^3)$. Building upon Indyk's technique, Chazelle et al. [54] proposed a data structure that solves approximate halfspace range queries on the unit sphere by applying a dimension reduction technique to Hamming space. All of these algorithms use a standard LSH index data structure in a black-box fashion. We propose a data structure that is almost similar to a standard LSH data structure, but query it in an adaptive way. Our guarantees are probabilistic in the sense that each close point is present in the output with constant probability.

Using LSH-based indexes for range reporting means that we report each point closer than distance r with a certain probability, as well as some fraction of the points with distance in the range (r, cr). When c is large, this can have negative consequences for performance: a query could report nearly every point in the data set, and any performance gained from approximation is lost. When the approximation factor c is chosen close to 1, data structures working in high dimensions usually need many independent repetitions to find points at distance r. This is another issue with such indexes that makes range reporting hard: very close points show up in every repetition, and we need to remove these duplicates.

A natural approach to overcome the difficulties mentioned above is to choose the approximation factor c such that the cost of duplicated points roughly equals the cost of dealing with far points. For LSH-based algorithms, many papers explain an offline approach of finding the "optimal" value of c for a data set [19, 41, 76, 171] which usually involves sampling, or making assumptions on the data distribution. However, the best value of c depends not only on the data set, but also on the *query*. This situation is depicted in Figure 4.1. In this paper, we provide a query algorithm that adapts to the input and finds a near-optimal c at *query time*. We manage to do this in time proportional to the number of points eventually returned for the optimal parameters, making the search essentially free.

Output-sensitivity. To illustrate the improvement over standard fixed parameter LSH, we propose hard data sets for spherical range reporting. In these data sets, we pick $t-1$ very close points that show up in almost every repetition, one point at distance r, and the remaining points close to distance cr. In this case LSH would need $\Theta(n^\rho)$ repetitions to retrieve the point at distance r with constant probability,

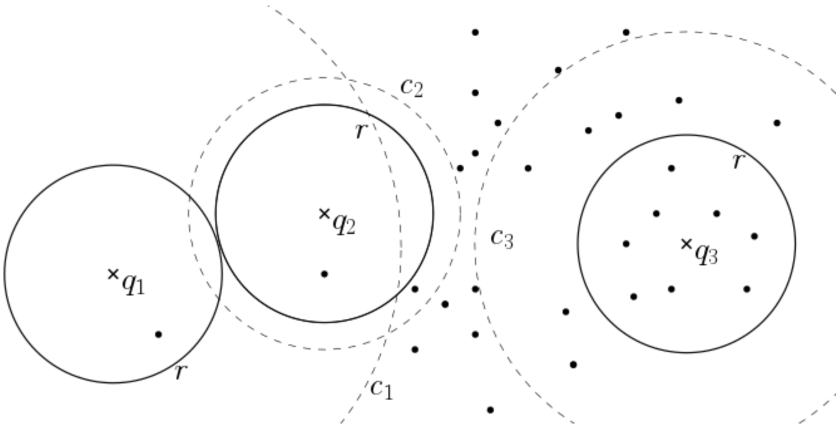


Figure 4.1: Three queries at radius r centered around points q_1, q_2, and q_3. The dashed circles around the queries contain about twice as many points as the inner solid circles. The ratios between the two radii are called c_1, c_2, and c_3, respectively. Note that the ratios are different across queries on the same dataset. This means that the running time to answer these queries can differ because larger c values allow faster query times.

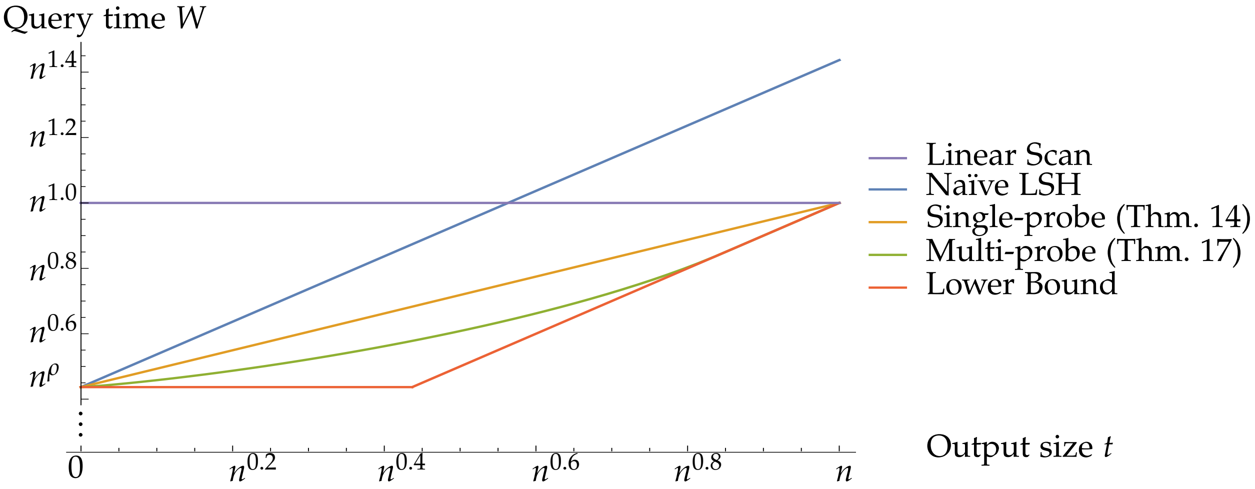


Figure 4.2: Overview of the running time guarantees of the proposed algorithms "Adaptive Single-probe" and "Adaptive Multi-probe" for collision probabilities $p_1 = 0.8$ and $p_2 = 0.6$ in d-dimensional Hamming space such that $\rho \approx 0.436$. The x-axis shows the output size t as a function of n, the y-axis shows the expected work W, i.e., the expected number of points the algorithm retrieves from the hash tables. For comparison, we plotted the target time of $O(n^{\rho} + t)$, the running time $O(tn^{\rho})$ of a naive LSH approach, and the running time $O(n)$ of a linear scan.

where e.g. $\rho = 1/c$ in Hamming space [96] and $\rho = 1/c^2$ in Euclidean space [20]. This means that the algorithm considers $\Theta(tn^{\rho})$ candidate points, which could be as large as $\Theta(n^{1+\rho})$ for large t. In Section 4.5 we describe and analyze two algorithms *Adaptive Single-probe* and *Adaptive Multi-probe* that mitigate this problem. The basic idea is that these algorithms "notice" the presence of many close points, and respond by choosing c more lenient, allowing for t far points being reported per repetition in addition to the t close points. This in turn allows us to do only $\Theta((n/t)^{\rho})$ repetitions, for a total candidate set of size $\Theta(t(n/t)^{\rho})$, which is never larger than n. In general, the number of points between distance r and cr have a linear influence on these running times. This is made precise in Section 4.4.

Multi-probing. When we stick to the LSH framework, the best running time we could hope for is $\Theta(n^\rho + t)$, giving the optimal output sensitive running time achievable by (data independent) LSH. In order to get closer to this bound, we analyze the *multi-probing approach* for LSH data structures, introduced in [151] and further developed in [125]. The idea is that LSH partitions the space in many buckets, but usually only examines the exact bucket in which the query point falls in each repetition. Multi-probing considers all buckets “sufficiently correlated” with the query bucket to increase the likelihood of finding close points. To our knowledge, multi-probing has always been applied in order to save space by allowing a smaller number of repetitions to be made and trading this for an increase in query time. Our motivation is different: We want to take advantage of the fact that each of the very close points can only be in one bucket per repetition. Hence by probing multiple buckets in each repetition, *we not only save memory, but also gain a large improvement in the dependency on t in our running time*. We do this by generalizing the adaptive single-probe algorithm to not only find the optimal c for a query, but also the optimal number of buckets to probe. As we show in Section 4.5, we are able to do this in time negligible compared to the size of the final candidate set, making it practically free. The algorithm works for any probing sequence supplied by the user, but in Section 4.6 we provide a novel probing sequence for Hamming space and show that it strictly improves the query time compared to the non-multi-probing variant. For large values of t, we show that the running time matches the target time $O(n^\rho + t)$. An overview of the exact running time statements of the algorithms proposed here with a comparison to standard LSH, a linear scan, and the optimal running time for LSH-based algorithms is depicted in Figure 4.2.

Techniques. The proposed data structure is very similar to a standard LSH data structure as described in [96]. In such a data structure, k locality-sensitive hash functions are concatenated to increase the gap between the collision probability of “close” points and “far away” points. A certain concatenation length k is fixed according to the number of points in the data set, the approximation factor c, and the strength of the hash family at hand. From the value k and the hash family one can compute how many repetitions (using independent hash functions) have to be made to guarantee that a close point is found with, say, constant probability. We give a detailed review of this approach in Section 4.2. Instead of building the data structure for only one particular k, we build a multi-level variant that encompasses all lengths $1, \ldots, k$ at the same time. At query time, we do an efficient search over the parameter space to find the provably best level. The algorithm then retrieves only the candidates from this level and filters far away points and duplicates. The reason we are able to do an efficient search over the parameter space is that certain parts of the output size can be estimated very quickly when storing the size of the hash table buckets in the LSH data structure. For example, when considering very large c, though the output may be large, there are only few repetitions to check. Gradually decreasing c, which technically means increasing the length k in the LSH data structure, we will eventually have to check so many repetitions that the mere task of iterating through them would be more work than scanning through the smallest candidate set found so far. Since the number of repetitions for each value of c grows geometrically, it ends up being bounded by the last check, which has size not larger than the returned candidate set. For multi-probing it turns out that a similar strategy works, but the search problem is now two-dimensional.

Additional Related Work. Our approach to query-sensitivity generalizes and extends the recent work of Har-Peled and Mahabadi [89] which considers approximate near neighbors. Our method applies to every space and metric supported by the LSH framework while [89] is presented for Hamming space.

The proposed single-probing algorithm can be thought of as an adaptive query algorithm on the trie-based LSH forest introduced by Bawa et al. in [41] for the related approximate k-nearest neighbor problem. The authors of [41] make significant assumptions on the distance distribution of approximate nearest neighbors. The algorithm proposed in [41] always looks at all $n^{f(c)}$ repetitions where $f(c)$ depends on the largest distance r supported by the algorithm and the approximation factor. It collects points traversing tries synchronously in a bottom-up fashion. By looking closer at the guarantees of LSH functions, we show that one can gradually increase the number of repetitions to look at and find the best level to query directly. We hope that the insights provided here will shed new light on solving approximate nearest neighbors beyond using standard reductions as in [88]. We note that very recent work of Andoni et al. [30] provides new guarantees for LSH forest in worst case settings — it would be interesting to see if this could lead to improvements of our results.

Combining results of very recent papers [115, 61, 24] on space/time-tradeoffs makes it possible to achieve running times that are asymptotically similar to our results with respect to multi-probing. We give a detailed exposition in Appendix 4.8.5.

4.2 Preliminaries

Our data structures can be implemented in a standard model of computation that allows unit cost retrieval of a memory word. For simplicity we will assume that hash function evaluation as well as distance computation can also be done in unit time — results in more general settings follows by reduction.

Let (X, dist) be a metric space over X with distance function $\mathrm{dist}\colon X^2 \to \mathbb{R}$. In this paper, the space usually does not matter; only the multi-probing sequence in Section 4.6 is tied to Hamming space.

Definition 10 (Spherical Range Reporting). *Given a set of points $S \subseteq X$ and a number $r \geq 0$, construct a data structure that supports the following queries: Given a point $q \in X$, report each point $p \in S$ with $dist(p, q) \leq r$ with constant probability.*

Note that we consider the exact version of SRR, where distances are not approximated, but allow point-wise probabilistic guarantees. Of course, repeating an algorithm that solves SRR $\Theta(\log |S|)$ times yields an algorithm that outputs each close point with high probability.

Definition 11 (LSH Family, [52]). *A* locality-sensitive hash (LSH) family $\mathcal{H}$ *is family of functions $h\colon X \to R$, such that for each pair $x, y \in X$ and a random $h \in \mathcal{H}$, for arbitrary $q \in X$, whenever $dist(q, x) \leq dist(q, y)$ we have* $\Pr[h(q) = h(x)] \geq \Pr[h(q) = h(y)]$.

Usually the set R is small, like the set $\{0, 1\}$. Often we will concatenate multiple independent hash functions from a family, getting functions $h_k\colon X \to R^k$. We call this a *hash function at level k*.

Having access to an LSH family $\mathcal{H}$ allows us to build a data structure with the following properties.

Theorem 13 ([88, Theorem 3.4]). *Suppose that for some metric space (X, dist) and some factor $c > 1$, there exists an LSH family such that* $\Pr[h(q) = h(x)] \geq p_1$ *when $dist(q, x) \leq r$ and* $\Pr[h(q) = h(x)] \leq p_2$ *when $dist(q, x) \geq cr$ with $p_1 > p_2$. Then there exists a data structure such that for a given query q, it returns with constant probability a point within distance cr, if there exists a point within distance r. The algorithm uses $O(dn + n^{1+\rho})$ space and evaluates $O(n^\rho)$ hash functions per query, where $\rho = \frac{\log(1/p_1)}{\log(1/p_2)}$.*

It is essential for understanding our algorithms to know how the above data structure works. For the convenience of the reader we provide a description of the proof next.

Proof. Given access to an LSH family $\mathcal{H}$ with the properties stated in the theorem and two parameters L and k (to be specified below), repeat the following process independently for each i in $\{1, \ldots, L\}$: Choose k hash functions $g_{i,1}, \ldots, g_{i,k}$ independently at random from $\mathcal{H}$. For each point $p \in S$, we view the sequence $h_i(p) = (g_{i,1}(p), \ldots, g_{i,k}(p)) \in R^k$ as the hash code of p, identify this hash code with a bucket in a table, and store a reference to p in bucket $h_i(p)$. To avoid storing empty buckets from R^k, we resort to hashing and build a hash table T_i to store the non-empty buckets for S and h_i.

Given a query $q \in X$, we retrieve all points from the buckets $h_1(q), \ldots, h_L(q)$ in tables $T_1, \ldots, T_L$, respectively, and report a close point in distance at most cr as soon as we find such a point. Note that the algorithm stops and reports that no close points exists after retrieving more than $3L$ points, which is crucial to guarantee query time $O(n^\rho)$.

The parameters k and L are set according to the following reasoning. First, set k such that it is expected that at most one distant point at distance at least cr collides with the query in one of the repetitions. This means that we require $np_2^k \leq 1$ and hence we define $k = \lceil \frac{\log n}{\log(1/p_2)} \rceil$. To find a close point at distance at most r with probability at least $1 - \delta$, the number of repetitions L must satisfy $\delta \leq (1 - p_1^k)^L \leq \exp(-p_1^k \cdot L)$. This means that L should be at least $p_1^{-k} \ln \delta$ and simplifying yields $L = O(n^\rho)$. Note that these parameters are set to work even in a worst-case scenario where there is exactly one point at distance p and all other points have distance slightly larger than cr. □

For ease of presentation, we assume that the collision probability p_1 is $\Theta(1)$ throughout the paper.

The LSH framework can easily be extended to solve SRR. We just report all the points that are in distance at most r from the query point in the whole candidate set retrieved from all tables $T_1, \ldots, T_L$ [17]. For the remainder of this paper, we will denote the number of points retrieved in this way by W ("work"). It is easy to see that this change to the query algorithm would already solve SRR with the guarantees stated in the problem definition. However, we will see in Section 4.4 that its running time might be as large as $O(n^{1+\rho})$, worse than a linear scan over the data set.

4.3 Data Structure

We extend a standard LSH data structure in the following way.

Assume we are given a set $S \subseteq X$ of n points, two parameters r and L, and access to an LSH family $\mathcal{H}$ that maps from X to R. Let $\text{reps}(k) = \lceil p_1^{-k} \rceil$ where p_1 is a lower bound on the probability that points at distance r collide under random choice of $h \in \mathcal{H}$. Let K be the largest integer such that $\text{reps}(K) \leq L$. A *Multi-level LSH data structure* for S is set up in the following way: For each $k \in \{0, \ldots, K\}$ choose functions $g_{k,i}$ for $1 \leq i \leq \text{reps}(k)$ from $\mathcal{H}$ independently at random. Then, for each $k \in \{0, \ldots, K\}$, build $\text{reps}(k)$ hash tables $T_{k,i}$ with $1 \leq i \leq \text{reps}(k)$. For a fixed pair $k \in \{0, \ldots, K\}$ and $i \in \{1, \ldots, \text{reps}(k)\}$, and each $x \in X$, concatenate hash values $(g_{1,i}(x), \ldots, g_{k,i}(x)) \in R^k$ to obtain the hash code $h_{k,i}(x)$. Store references to all points in S in table $T_{k,i}$ by applying $h_{k,i}(x)$. For a point $x \in X$, and for integers $0 \leq k \leq K$ and $1 \leq i \leq \text{reps}(k)$, we let $|T_{k,i}(x)|$ be the number of points in bucket $h_{k,i}(x)$ in table $T_{k,i}$. We store this value so it can be retrieved in constant time. In contrast to a standard LSH data structure, we only accept the number of repetitions, i.e., the allowed space to build the data structure, as an additional parameter. The value K is chosen such that the number of repetitions available suffices to obtain a close point at distance r with constant probability, cf. the proof of Theorem 13. This is ensured by the repetition count for all levels $0, \ldots, K$. The space usage of our data structure is $O(n \sum_{0 \leq k \leq K} p_1^{-k}) = O(np_1^{-K}) = O(nL)$. Hence multiple levels only add a constant overhead to the space consumption compared to a standard LSH data structure for level K. Figure 4.3 provides a visualization of the data structure.

We describe an alternative tree-based data structure that trades query time for space consumption in Appendix 4.8.1.

4.4 Standard LSH, Local Expansion, and Probing the Right Level

In this section we show that using a standard LSH approach can yield running time $\Omega(tn^\rho)$ when standard parameter settings such as the ones in the proof of Theorem 13 are used to solve SRR. Then, we define a measure for the difficulty of a query. Finally, we show that if the output size and this measure is known, inspecting a certain level in the multi-level LSH data structure gives output- and query-sensitive expected running times.

Suppose we want to solve SRR using an LSH family $\mathcal{H}$. Assume that the query point $q \in X$ is fixed. Given n, t with $1 \leq t \leq n$, and $c > 1$, we generate a data set S by picking

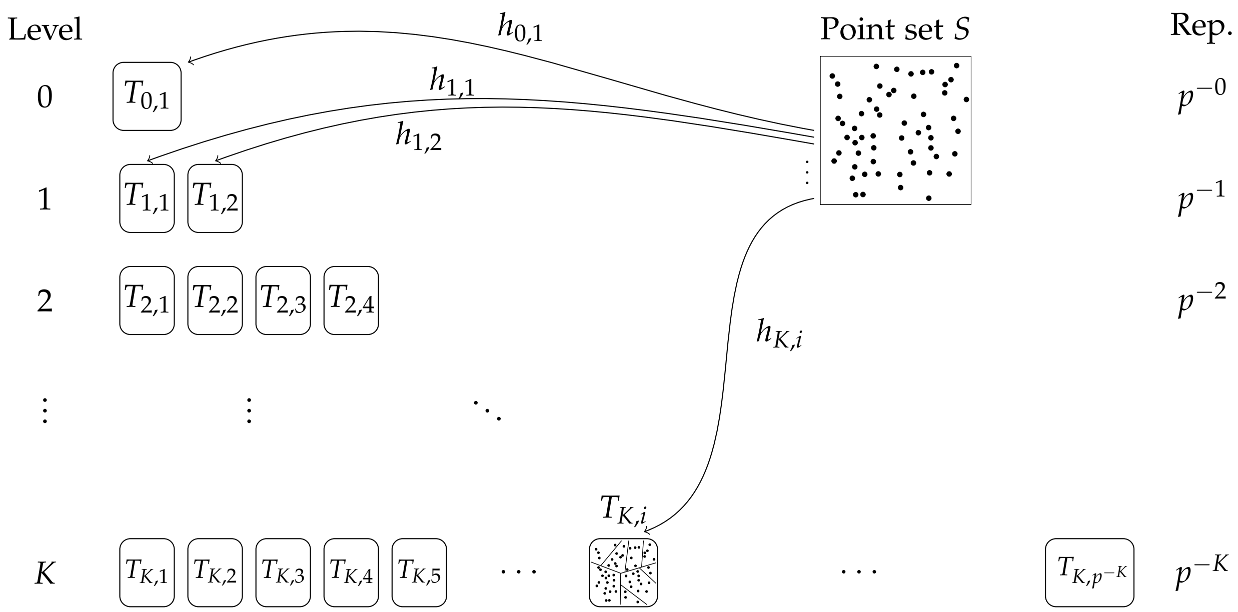


Figure 4.3: Overview of the multi-level LSH data structure with tables $T_{k,i}$ and hash functions $h_{k,i}$ splitting a data set S. The data structure is set up for levels $0, ..., K$ with repetition count "Rep." for collision probability $p = 1/2$. Example for a space partition of S induced by hash function $h_{K,i}$ is explicitly depicted as the content of table $T_{K,i}$ where each class is a bucket.

- $t - 1$ points at distance ϵ from q, for ϵ small enough that even concatenating $\lceil \frac{\log n}{\log 1/p_2} \rceil$ hash functions from $\mathcal{H}$, we still have collision probability higher than 0.01,
- one point $x \in X$ with $\text{dist}(q, x) = r$,
- the remaining $n - t$ points at distance cr.

We call a set S that is generated by the process described above a *t-heavy input for SRR on q*. By definition, a t-heavy input has expansion c at query point q. We argue that the standard LSH approach is unnecessarily slow on such inputs.

Observation 1. *Fix two values n and $t \leq n$, and let $q \in X$ be a fixed query point. Let S with $|S| = n$ be a t-heavy input generated by the process above. Suppose we want to solve SRR in (X, dist) using the LSH data structure from the proof of Theorem 13 with LSH family $\mathcal{H}$ build for S. Then the expected number of points retrieved from the hash tables on query q in the LSH data structure is $\Theta(tn^\rho)$.*

Proof. The standard LSH data structure is set up with $k = \lceil \frac{\log n}{\log 1/p_2} \rceil$ and $L = O(n^\rho)$. L repetitions are necessary to find the close point at distance r with constant probability. By the construction of S, each repetition will contribute at least $\Theta(t)$ very close points in expectation. So, we expect to retrieve $O(tn^\rho)$ close points from the hash tables in total. □

The process described above assumes that the space allows us to pick sufficiently many points at a certain distance. This is for example true in $\mathbb{R}^d$ with Euclidean distance. In Hamming space $\{0,1\}^d$ we would change the above process to enumerate the points from distance $1, 2, \ldots$ and distance $cr+1, cr+2, \ldots$. If d and r are sufficiently large, the same observation as above also holds for inputs generated according to this process.

For a set S of points, a point q, and a number $r > 0$, let $N_r(q)$ be the number of points in S at distance at most r from q. We next define the *expansion* at a query point q for a given distance. The expansion measures how far we can increase the radius of an r-sphere around the query point before the number of points before the number of points covered more than doubles. This dimensionality measure is central in the running time analysis of our proposed algorithms.

Definition 12 (Expansion). *Let $r > 0$, $q \in X$ and $S \subseteq X$ be a set of points. The* expansion *$c^*_{q,r}$ at point q is the largest number c such that $N_{cr}(q) \leq 2N_r(q)$, where $c^*_{q,r}$ is ∞ if $N_r(q) \geq n/2$.*

We will often simply write c^*_q, when r is known in the context. A visualization for the expansion around a query is shown in Figure 4.1.

4.4.1 Query Algorithms If t and c^*_q are Known

In the following, we state the parameter ρ as used in Theorem 13 as a function $\rho(r,c)$ such that $\rho(r,c) = \frac{\log(1/p(r))}{\log(1/p(cr))}$, where $p(\Delta)$ is the probability that two points at distance Δ collide. (The probability is over the random choice of the LSH function.) We omit the parameters when their value is clear from the context. The following statements use the Multi-Level LSH data structure introduced in Section 4.2.

Theorem 14. *Let $r > 0$ and $c \geq 1$. Let S be a set of n points and let DS be the Multi-level LSH data structure for S using $L = \Omega(n^{\rho(r,c)})$. Given a query point q, let $t = N_r(q)$ and c^*_q be the expansion around q in S. Then there exists a query algorithm on DS to solve SRR with the following properties:*

(i) If $c^*_q \geq c$, the algorithm has expected running time $O(t(n/t)^{\rho(r,c^*_q)})$.

(ii) Otherwise, the algorithm has expected running time $O(t(n/t)^{\rho(r,c)} + N_{cr}(q))$.

For $t = 0$, the running time is the same as $t = 1$.

Proof. Let p_1 and p_2 be the probabilities that the query point collides with points at distance r and $c^*_q r$, respectively, given the LSH family used. We consider statement (i) first. Set $k = \lceil \frac{\log(n/t)}{\log(1/p_2)} \rceil$ and note that $p_2^k \leq t/n$ and $p_1^{-k} = \Theta((n/t)^{\rho(r,c^*_q)})$. The algorithm works by inspecting all points in the query buckets in tables $T_{k,1}, \ldots, T_{k,p_1^{-k}}$. This repetition count guarantees constant probability of finding each close point and since $c^*_q \geq c$, we can assume $p_1^{-k} \leq L$. In each repetition, we expect collisions with not more than t close points, $N_{c^*_q r}(q)p_1^k$ points at distance at most $c^*_q r$ and np_2^k far points. By linearity of expectation, the expected number of collisions in all buckets is then not more than $p_1^{-k}(t + N_{c^*_q r}(q)p_1^k + np_2^k)$. By the choice of k, $np_2^k \leq t$ and so this is $O(tp_1^{-k} + N_{c^*_q r}(q))$. By the definition of c^*_q, $N_{c^*_q r}(q) = O(t)$ which means that this term is dominated by the former. Finally looking at every bucket takes time $O(p_1^{-k})$, but this is likewise dominated if $t \geq 1$. Statement (ii) follows by the same line of reasoning, simply using c instead of c_q^*. Since this value of c does not have the expansion property, the term $N_{cr}(q)$ is present in the running time. □

The running time bounds depend on the number of points at distance at most cr. This is inherent to our approach when the expansion around the query is smaller than the c value that can be read off from the number of repetitions and the LSH hash family at hand. The influence of these points is however only linear in their number.

Theorem 14 basically shows that there exists a single level of the multi-level LSH data structure that we want to probe when the number of close points t and the expansion around the query are known.

4.5 Adaptive Query Algorithms

In this section we describe a query algorithm that (with small overhead) obtains the results from Theorem 14 without knowing t or the expansion around the query. It turns out that we can get something even better, namely, a running time equivalent to the best over all choices for k and the number of repetitions if we were to know the entire distribution of distances $\mathrm{dist}(q, x)$ from the query point q to data points x in the data set.

We work on the multi-level LSH data structure DS set up for $S \subseteq X$ with tables $T_{k,i}$ as introduced in Section 4.3. DS is assumed to have been built with L repetitions and K levels.

Suppose a query algorithm inspects the buckets at level k. Let W_k denote the sum of the number of points retrieved from these buckets and the number of buckets inspected. By linearity of expectation, the expected work $\mathrm{E}[W_k]$ is

$$\mathrm{E}[W_k] = p_1^{-k}(1 + \sum_{x \in S} \Pr[h_k(q) = h_k(x)]),$$

because the algorithm considers p_1^{-k} repetitions (we omit ceilings for ease of presentation) and the expected number of collision within one repetition is $\sum_{x \in S} \Pr[h_k(q) = h_k(x)]$.

The function $\mathrm{E}[W_k]$ over k will have a minimum in $[0, K]$. We denote this minimum by W_{single}, i.e., we set

$$W_{\text{single}} = \min_{0 \leq k \leq K} \mathrm{E}[W_k]. \tag{4.1}$$

W_{single} denotes the expected work on the "optimal level" of the data structure for the query point, in the sense that we expect to check the fewest points while guaranteeing to find each close point with constant probability. We refer to it with the subscript "single" to emphasize that only a single bucket is checked in each repetition. Our algorithmic goal is to describe an algorithm that finds this level with only small overhead. Since W_{single} describes the minimum expected work knowing the distance distribution *from the query point*, we note that this quantity is always upper bounded by running times stated in Theorem 14. However, in many important cases it can be much smaller than that. To make this precise, we calculate W_{single} for different data distributions, including “locally growth-restricted” as considered in [73]. In this case it turns out that $W_{\text{single}} = O(\log n)$, an exponential improvement over the “standard” query time $O(n^\rho)$. Details are provided in Appendix 4.8.2.

The query algorithm is given as Algorithm 1 and works as follows: For each level $0 \leq k \leq K$, calculate the work of doing $\text{reps}(k) = \lceil 2p_1^{-k} \log 2k \rceil$ repetitions by summing up all bucket sizes. (The $2 \log 2k$ factor is a technical detail explained in the proof below.) Terminate as soon as the optimal level has been provably found, which may be one that we have considered in the past, and report all close points in the candidate set. The decision whether to consider a new level is based on whether the number of buckets to look at is larger than the smallest candidate set found so far.

Algorithm 1 Adaptive-Single-Probe(q, p_1, T)

```
1: $k \leftarrow 1, k_{\text{best}} \leftarrow 0, w_{\text{best}} \leftarrow n$;
2: while $\text{reps}(k) \leq \min(L, w_{\text{best}})$ do
3:     $w_k \leftarrow \sum_{i=1}^{\text{reps}(k)} (1 + |T_{k,i}(q)|)$;
4:     if $w_k < w_{k_{\text{best}}}$ then
5:         $k_{\text{best}} \leftarrow k; w_{\text{best}} \leftarrow w_k$;
6:     $k \leftarrow k + 1$;
7: return $\bigcup_{i=1}^{\text{reps}(k_{\text{best}})} \{x \in T_{k_{\text{best}},i}(q) \mid \text{dist}(x, q) \leq r\}$
```

Adaptive query algorithm on the Multi-level LSH data structure from Section 4.3 set up with hash tables $T_{k,i}$ with $0 \leq k \leq K$ and $1 \leq i \leq \text{reps}(k) = \lceil 2p_1^{-k} \log 2k \rceil$. We denote with $|T_{k,i}(q)|$ the size of the bucket the query point q is hashed to in the i-th repetition on level k.

Theorem 15. *Let r and $S \subseteq X$ with $|S| = n$ be given. Then Algorithm 1 solves SRR with point-wise constant probability. The expected running time of the **while**-loop in lines (2)–(6) and the expected number of distance computations in line (7) is $O(W_{single} \log \log W_{single})$, where*

$$W_{\text{single}} = \min_{0 \leq k \leq K} \left[p_1^{-k} (1 + \sum_{x \in S} \Pr[h_k(q) = h_k(x)]) \right].$$

This theorem says that Algorithm 1 finds the best level for the query point at hand with only small overhead compared to an algorithm that would have been told by some oracle which level of the data structure has minimum expected work for the query.

Proof. First we show that the algorithm works correctly, then we argue about its running time.

For correctness, let $y \in S$ be a point with $\text{dist}(y, q) \leq r$. We want to show that with constant probability, y is present in one of the buckets the algorithm inspects in line 7. Since y is present in any particular repetition with probability p_1^k, intuitively making

$1/p_1^k$ repetitions should suffice for correctness. However, our case is a bit more delicate. The probability of y being present in $\bigcup_i T_{k,i}$ is p_1^k if we do not know anything about the data. However, for the choice of k used by the algorithm, $\bigcup_i T_{k,i}$ is particularly small. The expected size of $\bigcup_i T_{k,i}$ is larger conditioned on y being present, and so equivalently if we choose the k that minimizes $\bigcup_i T_{k,i}$ we also decrease the chance of y finding there. It is seems difficult to handle this correlation directly, so we take a different approach here. We prove that y is present for every k we may choose with sufficiently high probability before the algorithm inspects bucket sizes. This requires us to choose a slightly higher number of repetitions, in particular $\text{reps}(k)$ is such that the probability of finding y is at least $1 - (1 - p_1^k)^{\text{reps}(k)} \geq 1 - 1/(2k)^2$. By a union bound over the K levels of the data structure, y can be found on every level with probability at least $1 - \sum_{k=1}^{\infty} 1/(2k)^2 \geq 1/2$, which shows correctness.

Now we consider the running time. The work inside the loop is dominated by line (3) which takes time $O(\text{reps}(k))$, using constant time access to the size of the buckets. Say the last value k before the loop terminates is k^*, then the loop takes time $\sum_{k=1}^{k^*} O(\log k \cdot p_1^{-k}) \leq \log k^* \cdot p_1^{-k^*} \sum_{k=0}^{\infty} O(p_1^k) = O(\log k^* \cdot p_1^{-k^*}) = O(w_{\text{best}})$, where the last equality is by the loop condition, $\text{reps}(k^*) \leq w_{\text{best}}$.

In line 7, the algorithm looks at w_{best} points and buckets. Hence the total expected work is

$$\begin{aligned} \mathrm{E}[w_{\text{best}}] &= \mathrm{E}\left[\min_{0 \leq k \leq K} w_k\right] \\ &\leq \min_{0 \leq k \leq K} \mathrm{E}[w_k] \\ &= O(\min_{0 \leq k \leq K} \log k \cdot \mathrm{E}[W_k]) \\ &= O(\log k' \cdot \mathrm{E}[W_{k'}]) \\ &= O(W_{\text{single}} \log\log W_{\text{single}}). \end{aligned} \tag{4.2}$$

Here the first inequality follows by Jensen's inequality using the concavity of the min function. We bound the minimum over k by choosing a concrete value $k' = \arg\min_{0 \leq k \leq K} \mathrm{E}[W_k]$. Finally, we bound k' by $p_1^{-k'} \leq W_{\text{single}}$. □

4.5.1 A Multi-probing Version of Algorithm 1

A common technique when working with LSH is multi-probing [125, 151, 76, 21, 103]. The idea is that often the exact bucket $h_k(q)$ does not have a much higher collision probability with close points than some "nearby" bucket $\sigma(h_k(q))$. To be more precise, for a hash function $h_k \colon X \to R^k$, we define a *probing sequence* $\sigma = (\sigma_{k,\ell})_{\ell \geq 1}$ as a sequence of functions $R^k \to R^k$. Now when we would probe bucket $T_{k,i}(h_k(q))$, we instead probe $T_{k,i}(\sigma_{k,1}(h_k(q))), T_{k,i}(\sigma_{k,2}(h_k(q))), \ldots$. (Where $\sigma_{k,1}$ will usually be the identity function.)

For a point y at distance r from q, we will be interested in the event $[\sigma_{k,\ell}(h_k(q)) = h_k(y)]$. The probability that this event occurs is donated by $p_{k,\ell}$. We say that a probing sequence σ is *reasonable*, if (i) for each k we have $p_{k,1} \geq p_{k,2} \geq \ldots$ and (ii) for each ℓ

we have $p_{k,\ell} \geq p_{k+1,\ell}$.[1] The intuition is that we probe buckets in order of probability of collision with the query point. In particular, by disjointness of the events, the probability of a collision within the first ℓ probes is exactly

$$P_{k,\ell} = p_{k,1} + \cdots + p_{k,\ell}. \tag{4.3}$$

This means that with ℓ probes per repetition, it suffices to repeat $1/P_{k,\ell}$ times to obtain constant probability of finding y.

To state the complexity of our algorithm, we generalize the quantities W_k and W_{single} from eq. (4.1) in the natural way to include multi-probing. We let $W_{k,\ell}$ denote the sum of the number points retrieved from the buckets plus the number of buckets inspected for each parameter pair (k, ℓ). By linearity of expectation, the expected work for (k, ℓ) is then

$$\mathrm{E}[W_{k,\ell}] = \frac{1}{P_{k,\ell}} \left(\ell + \sum_{\substack{x \in S \\ 1 \leq i \leq \ell}} \Pr[\sigma_{k,i}(h(q)) = h(x)] \right)$$

Analogously to the single-probing approach, we let W_{multi} denote the minimal work one would expect to need for an LSH based approach that knows the optimal values of k and ℓ:

$$W_{\text{multi}} = \min_{\substack{0 \leq k \leq K \\ \ell \geq 1}} \mathrm{E}[W_{k,\ell}] \tag{4.4}$$

Incorporating multi-probing into Algorithm 1 is done by carefully searching through the now two-dimensional, infinite space $[0, K] \times [1, \infty]$ of parameters. The algorithm is given as Algorithm 2 and works as follows: For parameter pairs (k, ℓ) we define two functions

$$\begin{align} \operatorname{reps}(k, \ell) &= \lceil 2 \log(2\ell k) / P_{k,\ell} \rceil \quad \text{and} \tag{4.5} \\ \operatorname{cost}(k, \ell) &= \ell \cdot \operatorname{reps}(k, \ell), \tag{4.6} \end{align}$$

where $\operatorname{reps}(k, \ell)$ is the repetition count analogous to the first algorithm, and $\operatorname{cost}(k, \ell)$ represents the number of buckets that need to be inspected for the parameter pair. The parameter space is explored in order of this cost.

For each considered parameter pair (k, ℓ), the actual work $w_{k,\ell}$ of inspecting the candidate set $\bigcup_{i=1}^{\operatorname{reps}(k,\ell)} \bigcup_{j=1}^{\ell} T_{k,i}(\sigma_{k,j}(h_k(q)))$ is calculated by summing up the bucket sizes plus the number of buckets. The algorithm keeps track of the size of the smallest work load seen so far, and it terminates as soon as this size is smaller than the smallest cost value of all parameter pairs not considered so far. We note that to obtain good query time, we have to compute the work loads $w_{k,\ell}$ in line 9 of Algorithm 2 slightly differently. The exact replacement is discussed in the proof below.

[1] As long as the collision probabilities $p_{k,\ell}$ are known or can be estimated accurately enough, an arbitrary probing sequence can be made reasonable by sorting it. All probing sequences we know of from the literature follow this approach, see [21] for an example.

Algorithm 2 Adaptive-Multi-probe(q, σ, T)

```
$w_{\text{best}} \leftarrow n$; $k_{\text{best}} \leftarrow 0$; $\ell_{\text{best}} \leftarrow 1$
PQ $\leftarrow$ empty priority queue        ▷ Manages pairs $(k, \ell)$ with priority
    $\text{cost}(k, \ell) = \ell \cdot \text{reps}(k, \ell)$.
PQ.insert($(1, 1)$)
while PQ.min() $< w_{\text{best}}$ do
    $(k, \ell) \leftarrow$ PQ.extractMin()
    if $k < K$ and $\ell = 1$ then
        PQ.insert($(k + 1, 1)$)
    PQ.insert($(k, \ell + 1)$)
    $w_{k,\ell} \leftarrow \sum_{i=1}^{\text{reps}(k,\ell)} \sum_{j=1}^{\ell} \left(1 + |T_{k,i,j}(q)|\right)$
    if $w_{k,\ell} < w_{\text{best}}$ then
        $k_{\text{best}} \leftarrow k$;   $\ell_{\text{best}} \leftarrow \ell$;   $w_{\text{best}} \leftarrow w_{k,\ell}$
return $\bigcup_{i=1}^{\text{reps}(k_{\text{best}},\ell_{\text{best}})} \bigcup_{j=1}^{\ell_{\text{best}}} \{x \in T_{k_{\text{best}},i,j}(q) \mid \text{dist}(x, q) \leq r\}$
```

Adaptive multi-probing query algorithm on the Multi-Level LSH data structure from Section 4.3 with K levels and $\text{reps}(k, \ell) = \lceil 2 \log(2\ell k)/P_{k,\ell} \rceil$. For clarity, we write $T_{k,i,j}(q) = T_{k,i}(\sigma_{k,j}(h_{k,i}(q)))$ for the jth bucket in the sequence at level k, repeititon i. The algorithm scans the parameter space $[0, K] \times [1, \infty)$ in order of $cost(k, \ell)$ starting at $(0, 1)$. The PQ.min function returns the smallest priority in the priority queue.

Theorem 16. *Let $S \subseteq X$ and r be given. Let (k, ℓ) be a pair that minimizes the right-hand side of eq.* (4.4)*. Given a reasonable probing sequence σ, Algorithm 2 on DS solves SRR with point-wise constant probability. If DS supports at least reps(k, ℓ) repetitions, the expected running time is $O(W_{multi} \log^3 W_{multi})$ and the expected number of distance computations is $O(W_{multi} \log W_{multi})$, where*

$$W_{multi} = \min_{\substack{0 \leq k \leq K \\ \ell \geq 1}} \left[\frac{1}{P_{k,\ell}} \left(\ell + \sum_{\substack{x \in S \\ 1 \leq i \leq \ell}} \Pr[\sigma_{k,i}(h(q)) = h(x)] \right) \right]$$

Proof. We first show correctness and then bound the running time of the algorithm.

To show the correctness of the algorithm, let $y \in S$ be an arbitrary point at distance at most r from the query point q. Similarly to the proof of Theorem 15, we show that the algorithm is correct for all parameter pairs simultaneously. For each pair (k, ℓ) considered by the algorithm, point y is found within the first ℓ probed buckets in $T_{k,i}$ with probability at least $P_{k,\ell}$ for $1 \leq i \leq \text{reps}(k, \ell)$ by eq. (4.3). With $\text{reps}(k, \ell)$ repetitions, the probability of finding y is at least $1 - (1 - P_{k,\ell})^{\text{reps}(k,\ell)} \geq 1 - (2\ell k)^{-2}$ using the definition from eq. (4.5). A union bound over the whole parameter space then yields $\sum_{k=1}^{\infty} \sum_{\ell=1}^{\infty} (2\ell k)^{-2} < 7/10$, and so y can be found in one of the probed buckets for every parameter choice with constant probability, which shows correctness.

To analyze the running time, we consider the value $w_{\text{best}} = w_{k_{\text{best}},\ell_{\text{best}}}$ found in the loop. This value bounds the number of distance computations and we will show that the number of operations done in the while-loop can similarly be bounded by $O(w_{\text{best}} \log^2 w_{\text{best}})$. We show that w_{best} is the smallest among all $w_{k,\ell}$ in the parameter space, which will allow us to bound $\mathrm{E}(w_{\text{best}}) = O(W_{\text{multi}} \log W_{\text{multi}})$ as in the theorem.

Starting from the end, consider any pair (k, ℓ) in the parameter space. If the algorithm actually inspected this pair, we must have $w_{k,\ell} \geq w_{\text{best}}$, since we always keep the smallest value seen. If the pair was not inspected by the algorithm, let (k', ℓ') be the pair of smallest cost left in the priority queue when the loop breaks. Then we will show

$$w_{\text{best}} \leq \operatorname{cost}(k', \ell') \leq \operatorname{cost}(k, \ell) \leq w_{k,\ell}$$

proving minimality. Recall that $\operatorname{cost}(k, \ell)$ from eq. (4.6) is used as the priority of a parameter pair in the priority queue. The loop condition thus directly gives us the first inequality. For the third inequality note that $\operatorname{cost}(k, \ell)$ denotes the number of buckets associated with a parameter pair (k, ℓ). Since $w_{k,\ell}$ is the number of these buckets plus the points points inside them, we get the inequality.

Finally for the second inequality, we will show that $\operatorname{cost}(k, \ell)$ is monotone in k as well as in ℓ. Since (k, ℓ) was not considered by the algorithm, it must have either higher k or ℓ than some pair in the priority queue while the other parameter is the same, and so by monotonicity its cost is higher, giving the inequality.

The function $\operatorname{cost}(k, \ell)$ is monotone in k because $p_{k,\ell} \geq p_{k+1,\ell}$ for reasonable probing sequences as defined. This implies $P_{k+1,\ell} \leq P_{k,\ell}$ and so $1/P_{k,\ell}$, $\operatorname{reps}(k, \ell)$ and $\operatorname{cost}(k, \ell)$ are all monotonically increasing in k.

For ℓ we have to be a bit more careful, since $1/P_{k,\ell}$ is decreasing in ℓ. This is of course because doing more multiprobing increases our chances of finding y, and so we have to do fewer independent repetitions. Luckily $\ell/P_{k,\ell}$ is monotonically increasing in ℓ, and so $\operatorname{cost}(k, \ell)$ is as well. The proof for this is given in Appendix 4.8.3.

Next we bound the running time of the loop. The way line 9 is written in the figure, the loop actually takes far too much time. When computing a new value $w_{k,\ell+1}$, we instead take advantage of the work $w_{k,\ell}$ already discovered, and only consider the number of buckets that are new or no longer needed. Specifically, we may compute

$$\begin{aligned} w_{k,\ell+1} = w_{k,\ell} + &\sum_{i=1}^{\operatorname{reps}(k,\ell+1)} |T_{k,i,\ell+1}(q)| \\ &- \sum_{j=1}^{\ell} \sum_{i=1+\operatorname{reps}(k,\ell)}^{\operatorname{reps}(k,\ell+1)} |T_{k,i,j}(q)|. \end{aligned}$$

Using this calculation, we consider each bucket size at most twice (once when added, and once when subtracted). Thus, computing the values $w_{k,1}, \ldots, w_{k,\ell}$ takes time at most $2(\operatorname{reps}(k, 1) + \cdots + \operatorname{reps}(k, \ell))$.

Let k^* be the largest value such that a parameter pair (k^*, ℓ) was visited. Similarly, for each k let ℓ_k be the largest value such that the pair (k, ℓ_k) was visited. Then the total time spent is no more than

$$\begin{aligned}\sum_{k=1}^{k^*} 2 \sum_{\ell=1}^{\ell_k} \mathrm{reps}(k,\ell) &\leq 4 \sum_{k=1}^{k^*} \log(2k\ell_k) \sum_{\ell=1}^{\ell_k} 1/P_{k,\ell_k} \\ &= \sum_{k=1}^{k^*} (\log k\ell_k)\, O(\ell_k(\log \ell_k)/P_{k,\ell_k}) \\ &= \sum_{k=1}^{k^*} (\log \ell_k)\, O(\mathrm{cost}(k,\ell_k)), \end{aligned} \tag{4.7}$$

where we used Lemma 4.8.1 to bound the sum over $1/P_{k,\ell_k}$. Now, observe that by the loop condition we know that $\mathrm{cost}(k,\ell_k) \leq w_{\mathrm{best}}$. Moreover, the value k^* is bounded by $k^* = O(\log w_{\mathrm{best}})$ because the algorithm considered $(k^*, 1)$ and we know that $p_1^{-k^*} \leq \mathrm{cost}(k^*, 1) \leq w_{\mathrm{best}}$. Finally, we bound ℓ_k by $\mathrm{cost}(k,\ell_k) \leq w_{\mathrm{best}}$, so $\log \ell_k = O(\log w_{\mathrm{best}})$. This allows us to bound eq. (4.7) by $O(w_{\mathrm{best}} \log^2 w_{\mathrm{best}})$. Having these bounds on k^* and ℓ_k, we can now see that the loop is iterated $O(w_{\mathrm{best}} \log w_{\mathrm{best}})$ times and each priority queue operation takes time $O(\log\log w_{\mathrm{best}})$, because there are at most k^* elements managed at the same time. So, all priority queue operations take time $O(w_{\mathrm{best}} \log w_{\mathrm{best}} \log\log w_{\mathrm{best}})$ and are dominated by the work load computations.

Finally, a calculation analogous to eq. (4.2) shows that $w_{\mathrm{best}} = O(W_{\mathrm{multi}} \log W_{\mathrm{multi}})$ which proves the theorem. □

4.5.2 Summary of Results

We stress that the proposed algorithms work at least as well as standard LSH for each data set and query, given the space restrictions w.r.t. the number of repetitions provided by the user. In particular, these quantities are always at most as large as the expected running times stated in Theorem 14 and Theorem 17 (to be presented in the next section), given the number of repetitions is as large as stated in these theorems. This comes at the cost of making slightly more repetitions per level. Specifically, $O(\log\log W_{\mathrm{single}}) = O(\log\log n)$ more repetitions are needed in Theorem 15 and $O(\log^3 W_{\mathrm{multi}}) = O(\log^3 n)$ more repetitions are needed in Theorem 16.

Most importantly, Theorem 15 and Theorem 16 show that we are never more than log factors away from the ideal query time of a tuned LSH data structure across all possible parameters given the space constraints of the data structure. These quantities are query specific parameters, so we cannot assume that offline tuning achieves these candidate set sizes for all query points simultaneously.

4.6 A Probing Sequence in Hamming Space

In this section we analyze bit sampling LSH in Hamming space [88, Section 3.2.1] using a novel, simple probing sequence. We consider the static setting as in Section 4.4.1, where the number of points to report and the expansion around the query is known. We then show the existence of a certain (optimal) level and probing length parameters, and prove that using those give a good expected running time. The adaptive query algorithm from Section 4.5 would find parameters at least as good as those, and thus

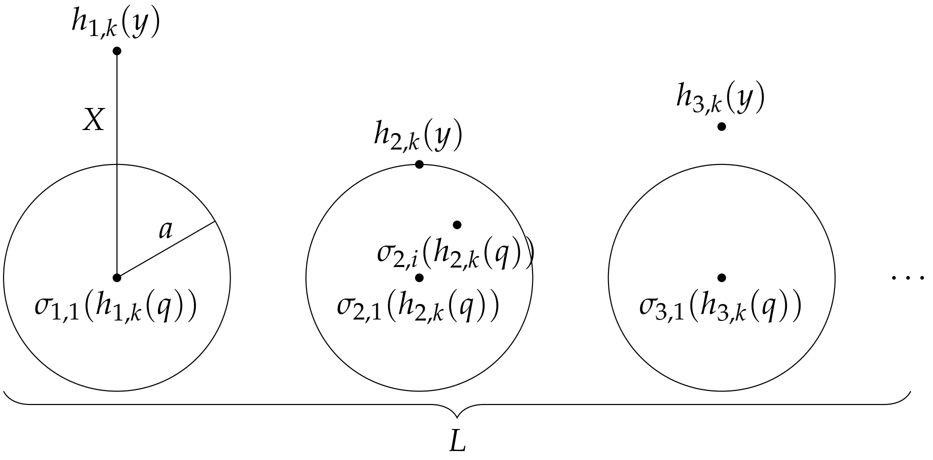

$h_{1,k}(y)$
X
a
$\sigma_{1,1}(h_{1,k}(q))$
$h_{2,k}(y)$
$\sigma_{2,i}(h_{2,k}(q))$
$\sigma_{2,1}(h_{2,k}(q))$
$h_{3,k}(y)$
$\sigma_{3,1}(h_{3,k}(q))$
$\cdots$
L

Figure 4.4: At each of the L repetitions we query the closest ℓ positions. Since the projected distance X to our target point y is distributed as $\text{Bin}(k, \text{dist}(q,y)/d)$, we find y with constant probability by setting $L = O(\Pr[X \le a]^{-1})$.

has a running time at least as good as what we show here (asymptotically within logarithmic factors).

Our scheme uses hash functions $h_k : \{0,1\}^d \to \{0,1\}^k$ that sample k positions at random with repetition. For a fixed query point $q \in \{0,1\}^d$ and $k \ge 1$, the probing sequence $\sigma_{k,\ell}$ maps $h_k(q)$ to the ℓ-th closest point in $\{0,1\}^k$, where ties are broken arbitrarily. This sequence can be generated efficiently, see [109].

Fix a target close point $y \in \{0,1\}^d$ at distance r, let p be the probability that q and y collide, and let $p_{k,\ell}$ be the probability that y lands in the ℓ-th bucket that we check. Furthermore, let $V(a) = \sum_{i=0}^{a} \binom{k}{i}$ be the volume of the radius a Hamming ball. If $\sigma_{k,\ell} h_k(q)$ is at distance a to $h(q)$, we have a collision if q and y differ in exactly a out of the k coordinates chosen by h_k. Hence, $p_{k,\ell} = p^{k-a}(1-p)^a$ for the a satisfying $V(a-1) < \ell \le V(a)$. Thus, the sequence is reasonable. Figure 4.4 illustrates our approach.

The best approximations to sizes of hamming balls are based on the entropy function. Hence, for the purpose of stating the theorem, we introduce the following notation. For $\alpha \in [0,1]$ and $\beta \in [0,1]$, we let

$$\begin{aligned} \mathrm{H}(\alpha) &= \alpha \log 1/\alpha + (1-\alpha)\log 1/(1-\alpha) \text{ and} \\ \mathrm{D}\left(\alpha \mid \beta\right) &= \alpha \log(\alpha/\beta) + (1-\alpha)\log((1-\alpha)/(1-\beta)) \end{aligned}$$

denote the binary entropy of α and the relative entropy between α and β, respectively. Moreover, let $\rho(r,c) = \frac{\log t}{\log n}\left(1 + \frac{\mathrm{D}(\alpha | 1-p(r))}{\mathrm{H}(\alpha)}\right)$ where α is defined implicitly from $\frac{\log t}{\log n}\left(1 + \frac{\mathrm{D}(\alpha | 1-p(cr))}{\mathrm{H}(\alpha)}\right) = 1$.

Theorem 17. *Let $r > 0$ and $c \ge 1$. Let $S \subseteq \{0,1\}^d$ be a set of n points and let DS be the Multi-level LSH data structure obtained from preprocessing S with $L = \Omega(n^{\rho(r,c)})$. Given a query point q, let $t = N_r(q) + 1$ and let c_q^* be the expansion around q in S. If $c_q^* \ge c$, there exists a query algorithm on DS to solve SRR with running time $O(n^{\rho(r,c_q^*)})$, otherwise the running time is $O(n^{\rho(r,c)} + N_{cr}(q))$.*

We do not know of a simple closed form for $\rho(r, c)$, but Figure 4.2 on Page 71 shows a numerical evaluation for comparison with the running time obtained for single-probing and other approaches.

The figure suggests that we always get better exponents than the single-probe approach, and that we get optimal query time for large t and asymptotically optimal for $t = n^{o(1)}$. Corollary 5 confirms this:

Corollary 5. *Let $\rho = (\log p_1)/(\log p_2) < 1/c$ be the usual exponent for bit sampling, then:*

If $t \geq n^{1-1/(p_2 \log p_1 + (1-p_2)\log 1/(1-p_1))}$, the expected query time is $O(n^{\rho(r,c)}) = O(n^\rho + t)$.

If $t = n^{o(1)}$, the expected query time is $O(n^{\rho(r,c)}) = n^\rho t^{O\left(1/(\frac{\log n}{\log t})\right)} = n^\rho t^{o(1)}$.

Note that this is the first algorithm to beat $n^\rho t^{\Omega(1)}$, even if only for certain ranges of t. The proof gives the exact asymptotic behavior.

Proof of Theorem 17. We will now show that the smaller number of repetitions needed by multi-probing leads to fewer collisions with the t very close points in hard instances of SRR. To see this, we bound the value of W_{multi} from eq. (4.4) as follows:

$$\begin{aligned}
&W_{\text{multi}} = \\
&\quad \min_{k,\ell} \left[\frac{\ell + \sum_{x \in S, 1 \leq \ell} \Pr[\sigma_{k,\ell}(h_k(q)) = h_k(x)]}{\sum_{1 \leq \ell} \Pr[\sigma_{k,\ell}(h_k(q)) = h_k(y)]} \right] \\
&\leq \min_{k,a} \left[\frac{V_k(a) + \sum_{x \in S} \Pr[\text{dist}(h_k(q), h_k(x)) \leq a]}{\Pr[\text{dist}(h_k(q), h_k(y)) \leq a]} \right] \\
&\leq \min_{k,a} \left[\frac{V_k(a) + t + t' \Pr[\mathrm{X}_1 \leq a] + n \Pr[\mathrm{X}_2 \leq a]}{\Pr[\mathrm{X}_1 \leq a]} \right], \\
&\text{where } X_1 \sim \text{Bin}(k, 1-p_1) \text{ and } X_2 \sim \text{Bin}(k, 1-p_2).
\end{aligned}$$

The first inequality holds by restricting ℓ to only take values that are the volume of a k-dimensional hamming ball; in the second inequality we upper bounded the collision probabilities for points in ranges $[0, r)$, $[r, cr)$ and $[cr, d]$.

The next step is to minimize this bound over the choice of k and a. We focus on $t' = O(t)$ and so we want

$$V_k(a) = t = n \Pr[\mathrm{X}_2 \leq a]. \tag{4.8}$$

For simplicity we write $\alpha = a/k$ for the normalized radius. We use the following tight bound [153] on the tail of the binomial distribution, for $\alpha \in (0, 1/2)$:

$$\begin{aligned}
\Pr[\text{Bin}(k, p) \leq \alpha k] &= \exp(-k \, \mathrm{D}\,(\alpha \mid p))\Theta(1/\sqrt{k}) \\
V_k(\alpha k) &= \exp(k H(\alpha))\Theta(1/\sqrt{k}).
\end{aligned}$$

With those, our equation eq. (4.8) can be written as

$$\begin{aligned}
k\,\mathrm{H}(\alpha) &= \log t = \log n - k\,\mathrm{D}\,(\alpha \mid 1-p_2) \text{ suggesting} \\
k &= \frac{\log t}{\mathrm{H}(\alpha)} = \frac{\log n}{\mathrm{H}(\alpha) + \mathrm{D}\,(\alpha \mid 1-p_2)} \text{ and} \\
\frac{\log t}{\log n} &= \frac{\mathrm{D}\,(\alpha \mid 1-p_2)}{\mathrm{H}(\alpha)} + 1.
\end{aligned}$$

We can then plug k into the bound on W_{multi}:

$$\begin{aligned} W_{\text{multi}} &\leq \frac{3t + t' \Pr[X_1 \leq a]}{\Pr[X_1 \leq a]} \\ &= \frac{t}{\exp(-k \,\mathrm{D}\,(\alpha \mid 1 - p_1))\Theta(1/\sqrt{k})} + t' \\ &= O\left(n^{\frac{\log t}{\log n}\left(\frac{\mathrm{D}(\alpha|1-p_1)}{\mathrm{H}(\alpha)}+1\right)}\right) + t' \end{aligned} \tag{4.9}$$

which are exactly the values stated in Theorem 17. □

Proof sketch of Corollary 5. For the first statement observe that if α is as large as $1 - p_1$, then $\Pr[\mathrm{Bin}(k, 1 - p_1) \leq \alpha k]$ is constant. The second factor in the minimization has all terms being within a constant of t, and so the whole thing becomes $O(t)$. We can check that $\alpha \geq 1 - p_1$ happens exactly when $\frac{\log t}{\log n} \geq \frac{\mathrm{H}(1-p_2)}{\mathrm{H}(1-p_2)+\mathrm{D}(1-p_2|1-p_1)}$. In this range $t \geq n^\rho$, so $O(t) = O(n^\rho + t)$.

For the second part of the corollary, we solve the equation implied by Theorem 17, asymptotically as $\tau = \frac{\log t}{\log n} \to 0$. Details can be found in Appendix 4.8.4, but the idea is as follows: We first define $f_p(\alpha) = 1 + \frac{\mathrm{D}(\alpha|p)}{\mathrm{H}(\alpha)}$, and show $f_{p_1}(\alpha) = (\rho + \psi\alpha/\log\frac{1}{p_2} + O(\alpha^2))f_{p_2}(\alpha)$ for ψ being the constant defined in Corollary 5. Using bootstrapping, we show the inversion $\alpha = f_{p_2}^{-1}(1/\tau) = \frac{\log 1/p_2}{\alpha \log 1/\alpha} + O(1/\log\frac{1}{\alpha})$. Plugging this into eq. (4.9) proves the corollary. □

4.7 Conclusion

In this article we proposed two adaptive LSH-based algorithms for Spherical Range Reporting that are never worse than a static LSH data structure knowing optimal parameters for the query in advance, and much better on many input distributions where the output is large or the query is easy.

The main open problem remaining is to achieve target time $O(n^\rho + t)$ for all inputs and n and $t \leq n$. One approach might be a data-dependent data structure as described in [25]. In the light of our multi-probing results, this bound might even be obtained using data-independent methods as well. Here, it would be interesting to analyze other probing sequences. It would be interesting to see whether one can describe adaptive query algorithms that make use of the output-sensitive space/time-tradeoff data structures we described in Appendix 4.8.5. Finally, it would be natural to extend our methods to give better LSH data structures for the approximate k-nearest neighbor problem.

Acknowledgements

The authors would like to thank the anonymous reviewers for their useful suggestions, which helped to improve the presentation of the paper. In addition they want to thank Ninh Pham for early discussions about adaptive LSH and output sensitivity. Finally

they thank the entire Scalable Similarity Search group at ITU Copenhagen for reviews and interesting comments on earlier versions of this paper. Old Appendix

4.8 Appendix

4.8.1 Trie-based Version of the Data Structure

In this section we discuss an alternative representation of our data structure. This is meant as a replacement for the Multi-level LSH data structure described in the main paper.It offers better space consumption while being slower to query.

As in the LSH forest data structure proposed by Bawa et al. [41], we do not store references to data points in hash tables. Instead we use a sorted array with a trie as a navigation structure on the array. The technical description follows.

First, choose $K \cdot L$ functions $g_{i,j}$ for $1 \leq i \leq L$ and $1 \leq k \leq K$ from $\mathcal{H}$ independently at random. For each $i \in \{1, \ldots, L\}$, we store a sorted array A_i with references to all data points in S ordered lexicographically by there bucket code over R^K. To navigate this array quickly, we build a trie over the bucket codes of all keys in S of depth at most K. Each vertex of the trie has two attributes `leftIndex` and `rightIndex`. If the path from the root of the trie to vertex v is labeled $L(v)$, then `leftIndex` and `rightIndex` point to the left-most and right-most elements in A_i whose bucket code starts with $L(v)$. We fix some more notation. For each point $q \in X$, we let $v_{i,k'}(q)$ be the vertex in trie $\mathcal{T}_i$ that is reached by searching for the bucket code of q on level at most k'. Furthermore, we let $\mathcal{T}_{i,k}(q)$ denote the set of keys that share the same length k prefix with q in trie $\mathcal{T}_i$. We can compute $|\mathcal{T}_{i,k}(q)|$ by subtracting $v_{i,k}(q)$.`leftIndex` from $v_{i,k}(q)$.`rightIndex` $+ 1$.

4.8.2 Examples For Calculating W_{single} for Certain Input Distributions

In this section we discuss two examples to get a sense for quantity eq. (4.1) defined on Page 78.

Example 1 (Random Points in Hamming Space) Fix a query point $q \in \{0,1\}^d$ and assume that our data set S consists of n uniform random points from $\{0,1\}^d$. Then the distance X from our query point is binomially distributed $\sim \text{Bin}(n, 1/2)$. If we choose the bitsampling hash function as in [96], $\sum_{x \in S} \Pr[h_k(q) = h_k(x)]$ is just

$$n\,\mathrm{E}[(1 - X/d)^k] = n\,\mathrm{E}[(X/d)^k]$$

by symmetry. This coresponds to finding the kth moment of a binomial random variable, which we can calculate by writing $X = d/2 + Z_d\sqrt{d/4}$ where Z_d has some distribution with $\mathrm{E}(Z_d) = 0$ and where $Z_d \to Z$ converges to a standard normal. Then

$$\begin{aligned} n\,\mathrm{E}[(X/d)^k] &= n2^{-k}\,\mathrm{E}(1 + Z_d/\sqrt{d})^k \\ &= n2^{-k}(1 + k\,\mathrm{E}(Z_d)/\sqrt{d} + O(k^2/d)) \\ &= n2^{-k}(1 + O(k^2/d)). \end{aligned}$$

For dimension $d = \Omega(\log n)^2$ our algorithm would find the ideal $k \approx \log_2 n$ to get $\sum_{x \in S} \Pr[h_k(q) = h_k(x)] = O(1)$ and $W = n^{\frac{\log 1/p_1}{\log 2}}$.

This work is of course exactly what we would expect for LSH with bitsampling and far points at distance $cr = d/2$. However normally the user would have had to specify this cr value, instead of the algorithm simply finding it for us.

Example 2 (Locally Growth-Restricted Data) Another interesting setting to consider is when the data is locally growth-restricted, as considered by Datar et al. [73, Appendix A]. This means that the number of points within distance r of q, for any $r > 0$, is at most r^c for some small constant c. In [73], the LSH framework is changed by providing the parameter k to the hash function. However, if we fix $r = k$, our algorithm will find a candidate set of size $W = O(\log n)$. So, our algorithm takes advantage of restricted growth and adapts automatically on such inputs.

Formally we can reuse the proof from [73], since they also inspect all colliding points. It is easy to see that theis integral $\int_1^{r/\sqrt{2}} e^{-Bc} c^b \, \mathrm{d}c$ is still bounded by $2^{O(b)}$ when we start at $c = 0$ instead of $c = 1$, since the integrand is less than 1 in this interval.

4.8.3 Lemma 4.8.1

Lemma 4.8.1. *Let $x_1 \geq x_2 \geq \ldots$ be a non-increasing series of real numbers, and let $X_n = \sum_{k=1}^n x_k$ be the nth prefix sum. Then it holds:*

$$n/X_n \leq (n+1)/X_{n+1} \tag{4.10}$$

$$\sum_{k=1}^n 1/X_k = O(n \log n / X_n). \tag{4.11}$$

Proof. Since the values x_k are non-increasing, we have $X_n \geq n x_n \geq n x_{n+1}$ and so

$$(n+1)X_n \geq nX_n + nx_{n+1} = nX_{n+1}$$

which is what we want for eq. (4.10). For the second inequality, we use eq. (4.10) inductively, we get $a/X_a \leq b/X_b$ whenever $a \leq b$. Hence we can bound eq. (4.11) term-wise as

$$\begin{aligned} \sum_{k=1}^n \frac{1}{X_k} &\overset{eq.\ (4.10)}{\leq} \sum_{k=1}^n \frac{n}{kX_n} \\ &= \frac{n}{X_n} \sum_{k=1}^n \frac{1}{k} \\ &= \frac{n}{X_n} H_n \\ &= O(n \log n / X_n). \end{aligned}$$

Here $H_n = 1 + 1/2 + \cdots + 1/n = \log n + O(1)$ is the nth harmonic number with the asymptotics by Euler [79]. □

We may notice that the bound is tight for $x_1 = x_2 = \cdots = x_n$. Say $x_k = 1$ for all k, then $X_k = k$ and $\sum_{k=1}^{n} 1/X_k = H_n = \Omega(n \log n / X_n)$. It is however common for the x_is to be strictly decreasing. In such cases we get closer to the other extreme, in which the $\log n$ factor disappears as $\sum_{k=1}^{n} 1/X_k = n/X_n$, which is sharp when $x_1 = 1$ and $x_k = 0$ for all other $k \geq 2$.

4.8.4 Proof of Corollary 5, second part

Recall that our algorithm runs in time $n^{\rho(r,c)}$ where $\rho(r,c) = \frac{\log t}{\log n} f(\alpha, p_1)$. Here $f(\alpha, p) = 1 + \frac{\mathrm{D}(\alpha|1-p)}{\mathrm{H}(\alpha)}$, and α is define implicitly from $\frac{\log t}{\log n} f(\alpha, p_2) = 1$.

When t is small compared to n, the multiprobing radius α can be made quite small as well. Working towards results for $t = n^{o(1)}$ we thus consider the regime $\alpha = o(1)$:

$$\begin{aligned} f(\alpha, p_1) &= 1 + \frac{\mathrm{D}\left(\alpha \mid 1 - p_1\right)}{\mathrm{H}(\alpha)} \\ &= \frac{\mathrm{H}(\alpha) + \mathrm{D}\left(\alpha \mid 1 - p_1\right)}{\mathrm{H}(\alpha) + \mathrm{D}\left(\alpha \mid 1 - p_2\right)} f(\alpha, p_2) \\ &= \frac{\log \frac{1}{p_1} + \alpha \log \frac{p_1}{1-p_1}}{\log \frac{1}{p_2} + \alpha \log \frac{p_2}{1-p_2}} f(\alpha, p_2) \\ &= \left(\rho + \frac{\psi}{\log 1/p_2} \alpha + O(\alpha^2)\right) f(\alpha, p_2), \end{aligned} \tag{4.12}$$

for small α and constants

$$\begin{aligned} \rho &= \frac{\log 1/p_1}{\log 1/p_2} \\ \psi &= \frac{\log \frac{p_1}{1-p_1} \log \frac{1}{p_2} - \log \frac{1}{p_1} \log \frac{p_2}{1-p_2}}{\log 1/p_2} \leq \log \frac{1}{1-p_1} \end{aligned}$$

depending on p_1 and p_2. This already shows that we get running time $n^{\frac{\log t}{\log n} \rho f(\alpha, p_2)(1+O(\alpha))} = n^{\rho(1+O(\alpha))}$, which is optimal up to lower order terms, if indeed $\alpha = o(1)$. We thus direct our attention to how fast α goes to 0 as a function of t and n. For that we first expand the following asymptotics:

$$\begin{aligned} \mathrm{H}(\alpha) + \mathrm{D}\left(\alpha \mid 1 - p\right) &= \alpha \log \tfrac{1}{1-p} + (1 - \alpha) \log \tfrac{1}{p} \\ &= \log \tfrac{1}{p} + O(\alpha) \end{aligned}$$

$$\begin{aligned} \mathrm{H}(\alpha) &= \alpha \log \tfrac{1}{\alpha} + (1 - \alpha) \log \tfrac{1}{1-\alpha} \\ &= \alpha \log \tfrac{1}{\alpha} + (1 - \alpha)(\alpha - O(\alpha^2)) \\ &= \alpha(\log \tfrac{1}{\alpha} + 1) + O(\alpha^2) \end{aligned}$$

$$\begin{aligned} f(\alpha, p) &= \frac{\mathrm{H}(\alpha) + \mathrm{D}\left(\alpha \mid 1-p\right)}{\mathrm{H}(\alpha)} \\ &= \frac{\log\frac{1}{p} + O(\alpha)}{\alpha(\log\frac{1}{\alpha} + 1) + O(\alpha^2)} \\ &= \frac{\log\frac{1}{p} + O(\alpha)}{\alpha(\log\frac{1}{\alpha} + 1)} \end{aligned} \tag{4.13}$$

We would like to solve eq. (4.13) for α, and plug that into eq. (4.12). To this end, we let $y = f(\alpha, p)/\log\frac{1}{p}$ and note the asymptotic bound $1/y^2 < \alpha < 1/y$. That gives us $\alpha = O(1/y)$ and $\log 1/\alpha = O(\log y)$, and we can use these estimates to "bootstrap" an inversion:

$$\begin{aligned} \alpha &= \frac{1 + O(\alpha)}{y(\log\frac{1}{\alpha} + 1)} \\ &= \frac{1 + O\left(\frac{1+O(\alpha)}{y(\log\frac{1}{\alpha}+1)}\right)}{y\left(\log\frac{1}{\frac{1+O(\alpha)}{y(\log\frac{1}{\alpha}+1)}} + 1\right)} \\ &= \frac{1 + O\left(\frac{1}{y\log y}\right)}{y\left(\log\left[y(\log\frac{1}{\alpha} + 1)\right] + \log\left[\frac{1}{1+O(1/y)}\right] + 1\right)} \\ &= \frac{1 + O\left(\frac{1}{y\log y}\right)}{y\log y + O(y\log\log y)} \\ &= \frac{1 + o(1)}{y\log y} \end{aligned} \tag{4.14}$$

Plugging the result back into eq. (4.12) we finally get:

$$\begin{aligned} \log n^{\rho(r,c)} &= (\log t) f(\alpha, p_1) \\ &= \log t\left(\rho + \frac{\psi}{\log 1/p_2}\alpha + O(\alpha^2)\right) f(\alpha, p_2) \\ &= \log t\left(\rho + \frac{\psi}{\log 1/p_2}\frac{1 + o(1)}{y\log y}\right) f(\alpha, p_2) \\ &= \log t\left(\rho f(\alpha, p_2) + \frac{\psi(1 + o(1))}{\log f(\alpha, p_2)}\right) \\ &= \rho\log n + \psi\frac{1 + o(1)}{\log\frac{\log n}{\log t}}\log t \\ &= \rho\log n + o(\log t), \end{aligned}$$

as $\frac{\log n}{\log t}$ goes to ∞, i.e. when $t = n^{o(1)}$.

We note again that this improves upon all other known methods, which get $\log \mathrm{E}(W_k) = \rho\log n + \Omega(\log t)$ for the same range of parameters.

4.8.5 A Different Approach to Solving SRR

We reconsider the approach to solve SRR presented in Indyk's Ph.D. thesis [92, Page 12] under the name "enumerative PLEB". While his method does not yield good running times directly, it is possible to combine a number of very recent results, to get running times similar to the ones achieved by our methods. We give a short overview of this approach next. As in Section 4.4.1, we assume that the number of points t to report is known. At the end of this section we describe a counting argument that is also contained in Indyk's Ph.D. thesis [92] that allows to solve the c-approximate spherical range counting problem in an output-sensitive way.

Indyk describes a black-box reduction to solve SRR using a standard dynamic data structure for the (c, r)-near neighbor problem. It works by repeatedly querying an (c, r)-near point data structure (time $O(n^{\rho_q})$) and then deleting the point found (time $O(n^{\rho_u})$), where ρ_q and ρ_u are the query- and update-parameters. (For a standard LSH approach, we have $\rho_q = \rho_u$.) This is done until the data structure no longer reports any points within distance r. Due to the guarantees of an (c, r)-near neighbor data structure, in the worst case the algorithm recovers all points within distance cr, giving a total running time of $t'(n^{\rho_q} + n^{\rho_u})$, where t' is the number of points within distance cr which might yield a running time of $\Omega(n^{1+\rho})$ as noticed in Section 4.4.

Of course, we can never guarantee sublinear query time when t' is large, but we can use a space/time-tradeoff-aware to improve the factor of t, when the number of returned points is large.

We will assume the (c, r)-near neighbor data structure used in the reduction is based on LSH. In [24], Andoni et al. describe a general data structure comprising loosely "all hashing-based frameworks we are aware of":

Definition 13 (List-of-points data structure)**.**

- *Fix sets $A_i \subseteq \mathcal{R}^d$, for $i = 1 \ldots m$; with each possible query point $q \in \mathcal{R}^d$, we associate a set of indices $I(q) \subseteq [m]$ such that $i \in I(q) \Leftrightarrow q \in A_i$;*
- *For a given dataset S, the data structure maintains m lists of points $L_1, L_2, \ldots, L_m$, where $L_i = S \cap A_i$.*

Having such a data structure, we perform queries as follows: For a query point q, we scan through each list L_i for $i \in I(q)$ and check whether there exists some $p \in L_i$ with $\|p - q\| \leq cr$. If it exists, return p.

Data structures on this form naturally allow insertions of new points, and we notice that if "Lists" are replaced by "Sets" we can also efficiently perform updates.

To solve spherical range reporting, we propose the following query algorithm for a point q:

1. For each $i \in I(q)$ look at every point x in L_i.
2. If $\|x - q\| \leq r$, remove the point from all lists, L_j, where it is present.

This approach allows for a very natural space/time-tradeoff. Assuming that querying the data structure takes expected time $O(n^{\rho_q})$ and updates take expected time $O(n^{\rho_u})$, the expected running time of the query is $O(n^{\rho_q} + tn^{\rho_u})$. This asymmetry can be exploited with a time/space tradeoff. In very recent papers [115, 61, 24] it was shown how to obtain such tradeoffs in Euclidean space for approximation factor $c \geq 1$, for any pair (ρ_q, ρ_u) that satisifies

$$c^2\sqrt{\rho_q} + (c^2-1)\sqrt{\rho_u} = \sqrt{2c^2-1}.$$

To minimize running time, we may take exponents balancing $T = n^{\rho_q} = tn^{\rho_u}$ and obtain

$$\begin{aligned}\frac{\log T}{\log n} &= \frac{1}{2c^2-1} + \frac{c^2-1}{2c^2-1}\tau \\ &\quad + \frac{c^2\left(c^2-1\right)}{2c^2-1}\left(2-\tau-2\sqrt{1-\tau}\right) \\ &\overset{(*)}{\leq} \rho + (1-c^4\rho^2)\tau,\end{aligned}$$

where $\tau = \frac{\log t}{\log n}$ and $\rho = 1/(2c^2-1)$. Here (*) holds for $t \leq \frac{2c^2-1}{c^4}$, and $T = O(t)$ otherwise. Note that this approach requires knowledge of t. A visualization of the running time guarantees of this approach is shown in Figure 4.5. Note that it requires knowledge of t and does not adapt to the expansion around the query point. It would be interesting to see whether our adaptive methods could be used to obtain a variant that is query-sensitive. Next, we discuss an algorithm for the spherical range counting problem that can be used to obtain an approximation of the value t sufficient for building the data structure presented here.

4.8.6 Solving c-approximate Spherical Range Counting

In [92, Chapter 3.6], Indyk shows that by performing $O((\log n)^2/\alpha^3)$ queries to independently built (c,r)-near neighbor data structures, there is an algorithm that returns for a query q a number C such that $(1-\alpha)N_r(q) \leq C \leq (1+\alpha)N_{cr}(q)$ with constant probability. The running time of the black-box reduction is $O(n^\rho(\log n)^2/\alpha^3)$. We show in this section that we can solve the problem in time $O((n/t)^\rho \log n/\alpha^3)$.

At the heart of the algorithm of [92] is a subroutine that has the following output behavior for fixed C:

1. If $N_{cr}(q) \leq C(1-\alpha)$, it will answer SMALLER

2. If $N_r(q) \geq C$, it will answer GREATER

The subroutine uses $O(\log n/\alpha^2)$ queries of independently build (c,r)-near neighbor data structures, each built by sampling n/C points from the data set.

We can use the above subroutine to solve the spherical range counting problem in time $O((n/t)^\rho \log n/\alpha^3)$ time as follows. Half the size of α, and perform a geometrical search for the values $t = n, (1-\alpha)n, (1-\alpha)^2 n, \ldots$. Assuming that a query on a data structure that contains n points takes expected time $O(n^\rho)$ and stopping as soon as

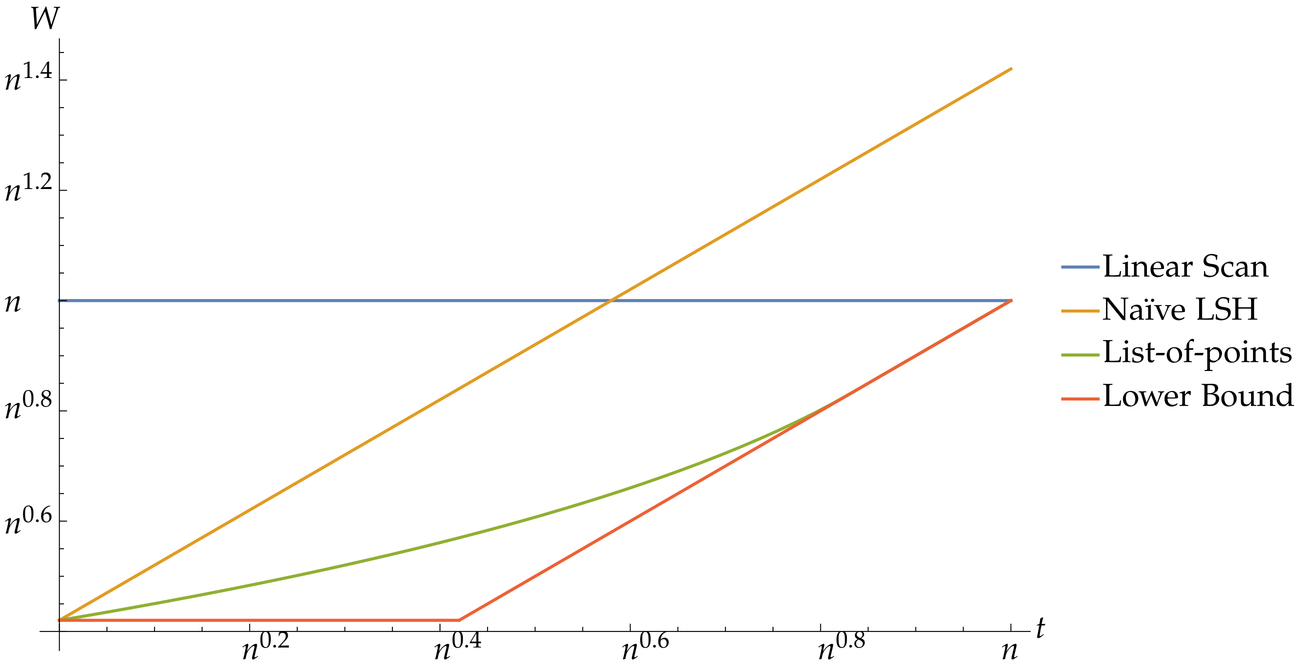


Figure 4.5: Visualization of the running time guarantees of the space/time-tradeoff list-of-points data structure for $c = 1.3$ in Euclidean space. The x-axis shows the value of t compared to n, the y-axis shows the expected work W. For comparison, we plotted the lower bound of $O(n^\rho + t)$, the running time $O(tn^\rho)$ of the naïve LSH approach, and the running time $O(n)$ of a linear scan.

the algorithm answers "Greater" for the first time, we obtain a running time (without considering the $O(\log n/\alpha^2)$ repetitions for each t value) of

$$\begin{aligned}
&\left(\frac{n}{n}\right)^\rho + \left(\frac{n}{n(1-\alpha)}\right)^\rho + \left(\frac{n}{n(1-\alpha)^2}\right)^\rho + \cdots + \left(\frac{n}{t}\right)^\rho \\
&\le \left(\frac{n}{t}\right)^\rho + \left(\frac{n(1-\alpha)}{t}\right)^\rho + \left(\frac{n(1-\alpha)^2}{t}\right)^\rho + \ldots \\
&= \left(\frac{n}{t}\right)^\rho \frac{1}{1-(1-\alpha)^\rho} \\
&\le \left(\frac{n}{t}\right)^\rho \frac{1}{\alpha\rho} \qquad \text{as } (1-\alpha)^\rho \le 1-\alpha\rho \text{ for } 0 \le \rho \le 1,
\end{aligned}$$

which results in a total running time of $O((n/t)^\rho \log n/\alpha^3)$.

Chapter 5

On the Complexity of Inner Product Similarity Join

Originally published in: Symposium on Principles of Database Systems, PODS 2016

Joint work with: Rasmus Pagh, Ilya Razenshteyn, Francesco Silvestri

5.1 Introduction

This paper is concerned with *inner product similarity join* (*IPS join*) where, given two sets $P, Q \subseteq \mathbb{R}^d$, the task is to find for each point $q \in Q$ at least one pair[1] $(p, q) \in P \times Q$ where the inner product (or its absolute value) is larger than a given threshold s. Our results apply also to the problem where for each $q \in Q$ we seek the vector $p \in P$ that maximizes the inner product, a search problem known in literature as *maximum inner product search* (MIPS) [158, 167].

Motivation

Similarity joins have been widely studied in the database and information retrieval communities as a mechanism for linking noisy or incomplete data. Considerable progress, in theory and practice, has been made to address metric spaces where the triangle inequality can be used to prune the search space (see e.g. [20, 194]). In particular, it is now known that in many cases it is possible to improve upon the quadratic time complexity of a naive algorithm that explicitly considers all pairs of tuples. The most prominent technique used to achieve provably subquadratic running time is locality-sensitive hashing (LSH) [88, 84]. In the database community the similarity join problem was originally motivated by applications in data cleaning [53, 32]. However, since then it has become clear that similarity join is relevant for a range of other data processing applications such as clustering, semi-supervised learning, query refinement, and collaborative filtering (see e.g. [165] for references and further examples). We refer to the recent book by Augsten and Böhlen [35] for more background on similarity join algorithms in database systems.

Inner product is an important measure of similarity between real vectors, particularly in information retrieval and machine learning contexts [111, 173], but not captured by techniques for metric similarity joins such as [98, 194]. Teflioudi et al. [175] studied

[1] Since our focus is on lower bounds, we do not consider the more general problem of finding all such pairs. Also note that from an upper bound side, it is common to limit the number of occurrences of each tuple in a join result to a given number k.

the IPS join problem motivated by applications in recommender systems based on latent-factor models. In this setting, a user and the available items are represented as vectors and the preference of a user for an item is given by the inner product of the two associated vectors. Other examples of applications for IPS join are object detection [80] and multi-class prediction [74, 100]. IPS join also captures the so-called *maximum kernel search*, a general machine learning approach with applications such as image matching and finding similar protein/DNA sequences [70].

Challenges of IPS join

Large inner products do not correspond to close vectors in any metric on the vector space, so metric space techniques cannot directly be used. In fact, there are reasons to believe that inner product similarity may be inherently more difficult than other kinds of similarity search: Williams [186, 15] has shown that a truly subquadratic *exact* algorithm for IPS join would contradict the Strong Exponential Time Hypothesis, an important conjecture in computational complexity. On the upper bound side new reductions of (special cases of) *approximate* IPS join to fast matrix multiplication have appeared [179, 106], resulting in truly subquadratic algorithms even with approximation factors asymptotically close to 1. However, the approach of reducing to fast matrix multiplication does not seem to lead to practical algorithms, since fast matrix multiplication algorithms are currently not competitive on realistic input sizes. From a theoretical viewpoint it is of interest to determine how far this kind of technique might take us by extending lower bounds for exact IPS join to the approximate case.

Another approach to IPS join would be to use LSH, which has shown its utility in practice. The difficulty is that inner products do not admit locality-sensitive hashing as defined by Indyk and Motwani [167, Theorem 1]. Recently there has been progress on *asymmetric* LSH methods for inner products, resulting in subquadratic IPS join algorithms in many settings. The idea is to consider collisions between two *different* hash functions, using one hash function for query vectors and another hash function for data vectors [167, 168, 139]. However, existing ALSH methods give very weak guarantees in situations where inner products are small relative to the lengths of vectors. It is therefore highly relevant to determine the possibilities and limitations of this approach.

Problem definitions

We are interested in two variants of IPS join that slightly differ in the formulation of the objective function. For notational simplicity, we omit the term IPS and we simply refer to IPS join as join. Let $s > 0$ be a given value. The first variant is the *signed* join, where the goal is to find at least one pair $(p, q) \in P \times Q$ for each point $q \in Q$ with $p^T q \geq s$. The second variant is the *unsigned* join which finds, for each point $q \in Q$, at least one pair $(p, q) \in P \times Q$ where $|p^T q| \geq s$. We observe that the unsigned version can be solved with the signed one by computing the join between P and Q and between P and $-Q$, and then returning only pairs where the absolute inner products are larger than s. Signed join is of interest when searching for similar or preferred items with a positive correlation, like in recommender systems. On the other hand, unsigned join can be used when studying relations among phenomena where even a large negative correlation is of interest. We note that previous works do not make the distinction between the signed and unsigned versions since they focus on settings where there are no negative dot products.

Our focus is on *approximate* algorithms for signed and unsigned joins. Indeed, approximate algorithms allow us to overcome, at least in some cases, the curse of dimensionality without significantly affecting the final results. Approximate signed joins are defined as follows.

Definition 14 (Approximate signed join)**.** *Given two point sets P, Q and values* $0 < c < 1$ *and* $s > 0$*, the* signed (cs, s) join *returns, for each $q \in Q$, at least one pair $(p, q) \in P \times Q$ with $p^T q \geq cs$ if there exists $p' \in P$ such that $p'^T q \geq s$. No guarantee is provided for $q \in Q$ where there is no $p' \in P$ with $p'^T q \geq s$.*

The unsigned (cs, s) join is defined analogously by taking the absolute value of dot products. Indexing versions of signed/unsigned exact/approximate joins can be defined in a similar way. For example, the signed (cs, s) search is defined as follows: given a set $P \subset \mathbb{R}^d$ of n vectors, construct a data structure that efficiently returns a vector $p \in P$ such that $p^T q > cs$ for any given query vector $q \in \mathbb{R}^d$, under the promise that there is a point $p' \in P$ such that $p^T q \geq s$ (a similar definition holds for the unsigned case).

As already mentioned, LSH is often used for solving similarity joins. In this paper, we use the following definition of asymmetric LSH based on the definition in [167].

Definition 15 (Asymmetric LSH)**.** *Let $\mathcal{U}_p$ denote the data domain and $\mathcal{U}_q$ the query domain. Consider a family $\mathcal{H}$ of pairs of hash functions $h = (h_p(\cdot), h_q(\cdot))$. Then $\mathcal{H}$ is said (s, cs, P_1, P_2)-* asymmetric LSH *for a similarity function sim if for any $p \in \mathcal{U}_p$ and $q \in \mathcal{U}_q$ we have:*

1. *if $sim(p, q) \geq s$ then* $\Pr_{\mathcal{H}}[h_p(p) = h_q(q)] \geq P_1$*;*
2. *if $sim(p, q) < cs$ then* $\Pr_{\mathcal{H}}[h_p(p) = h_q(q)] \leq P_2$*.*

When $h_p(\cdot) = h_q(\cdot)$, we get the traditional (symmetric) LSH definition. The ρ value of an (asymmetric) LSH is defined as usual with $\rho = \log P_1 / \log P_2$ [20]. Two vectors $p \in \mathcal{U}_p$ and $q \in \mathcal{U}_q$ are said to collide under a hash function from $\mathcal{H}$ if $h_p(p) = h_q(q)$.

5.1.1 Overview of results

Hardness results

The first results of the paper are conditional lower bounds for approximate signed and unsigned IPS join that rely on a conjecture about the *Orthogonal Vectors Problem* (OVP). This problem consists in determining if two sets $A, B \subseteq \{0, 1\}^d$, each one with n vectors, contain $x \in A$ and $y \in B$ such that $x^T y = 0$. It is conjectured that there cannot exist an algorithm that solves OVP in $O\left(n^{2-\epsilon}\right)$ time as soon as $d = \omega\left(\log n\right)$, for any given constant $\epsilon > 0$. Indeed, such an algorithm would imply that the Strong Exponential Time Hypothesis (SETH) is true [186].

Many recent interesting hardness results rely on reductions from OVP, however we believe ours is the first example of using the conjecture to show the conditional hardness for an approximate problem. In particular we show the following result:

Theorem 18. *Let $\alpha > 0$ be given and consider sets of vectors P, Q with $|Q| = n$, $|P| = n^\alpha$. Suppose there exists a constant $\epsilon > 0$ and an algorithm with running time at most $d^{O(1)} n^{1+\alpha-\epsilon}$, when d and n are sufficiently large and for all $s > 0$, for at least one of the following IPS join problems:*

1. *Signed (cs, s) join of $P, Q \subseteq \{-1, 1\}^d$ where $c > 0$.*
2. *Unsigned (cs, s) join of $P, Q \subseteq \{-1, 1\}^d$ where*
$$c = e^{-o(\sqrt{\log n}/\log\log n)} \ .$$
3. *Unsigned (cs, s) join of $P, Q \subseteq \{0, 1\}^d$ where*
$$c = 1 - o(1) \ .$$

Then the OVP conjecture is false.

Discussion. For the search problem, theorem 18 implies that, assuming the OVP conjecture, there does not exist a data structure for signed/unsigned (cs, s) inner product search with $(nd)^{O(1)}$ construction time and $n^{1-\epsilon} d^{O(1)}$ query time, for constant $\epsilon > 0$. This follows by considering a join instance with α constant small enough that we can build the data structure on P in time $o(nd)$. We can then query over all the points of Q in time $n^{1+\alpha(1-\epsilon)} d^{O(1)}$, contradicting the OVP conjecture. Our result can be seen as an explanation of why all LSH schemes for IPS have failed to provide sub-linear query times for small s. As the theorem however does not cover the case where c is very small, e.g. $n^{-\delta}$ for unsigned $\{-1, 1\}^d$, we show in section 4 that for such approximation requirements, there are indeed useful data structures.

We stress that the hardness result holds for algorithms solving signed/unsigned (cs, s) joins for *any* c in the specified range and *all* $s > 0$. It is possible to show complete relations between hard values of c, s and d, but for the sake of clearnes, we have prefered to optimize the largest range of hard c's. For intuition we can say, that the exact instances of (cs, s) joins that are hard, turn out to be the ones where s/d and cs/d are very small, that is when we have to distinguish nearly orthogonal vectors from very nearly orthogonal vectors. If we inspect the proofs of Theorem 18 and Lemma 5.2.3, we see that for unsigned join in $\{-1, 1\}^d$, the hard case has cs/d around $n^{1/\log\log n}$. Similarly for $\{0, 1\}^d$ join, cs ends up at just barely $\omega(1)$, while the d is as high as $n^{o(1)}$. It is interesting to note for $\{0, 1\}$ that if cs had been slightly lower, at $O(1)$, we could have solved the OVP problem exact in subquadratic time using an $n\binom{n^{o(1)}}{O(1)} = n^{1+o(1)}$ algorithm.

It is interesting to compare our conditional lower bound to the recent upper bounds by Karppa et al. [106], who get sub-quadratic running time for unsigned join of normalized vectors in $\{-1, 1\}^d$, when $\log(s/d)/\log(cs/d)$ is a constant smaller than 1.[2] Our next Theorem 19 shows that we cannot hope to do much better than this, though it does not completely close the gap. A hardness result for $\log(s/d)/\log(cs/d) = 1 - o(1)$ is still an interesting open problem. However, while the algorithm of Karppa et al. works

[2] More precisely they need $\log(s/d)/\log(cs/d) < 2/\omega$, where ω is the matrix multiplication constant. Note that the d term is due to normalization.

even for dimension $n^{1/3}$, our bound only requires the dimension to be slightly larger than polylog, so it may well be that their algorithm is optimal, while another algorithm with a higher dependency on the dimension matches our bound from above.

Theorem 19. *Let $\alpha > 0$ be given and consider sets of vectors P, Q with $|Q| = n$, $|P| = n^\alpha$. Suppose there exists a constant $\epsilon > 0$ and an algorithm with running time at most $d^{O(1)} n^{1+\alpha-\epsilon}$, when d and n are sufficiently large and for all $s > 0$, for at least one of the following IPS join problems:*

1. *Unsigned (cs, s) join of $P, Q \subseteq \{-1, 1\}^d$ where*

$$\frac{\log(s/d)}{\log(cs/d)} = 1 - o(1/\sqrt{\log n}) \ .$$

2. *Unsigned (cs, s) join of $P, Q \subseteq \{0, 1\}^d$ where*

$$\frac{\log(s/d)}{\log(cs/d)} = 1 - o(1/\log n) \ .$$

Then the OVP conjecture is false.

The $\{-1, 1\}^d$ case seems to be harder than the $\{0, 1\}^d$ case. In fact Valiant [179] reduces the general case of $P, Q \subseteq \mathbb{R}^d$ to the case $P, Q \subseteq \{-1, 1\}^d$ using the Charikar hyperplane LSH [52]. Another piece of evidence is that we can achieve runtime $n^{1+\frac{\log(s/d)}{\log(cs/d)}}$ using LSH for $\{0, 1\}^d$, but it is not known to be possible for $\{-1, 1\}^d$. Furthermore there appears to be some hope for even better data dependent LSH, as we show in section 5.4.2. The $\{0, 1\}^d$ case is particularly interesting, as it is occurs often in practice, for example when the vectors represent sets. A better understanding of the upper and lower bounds for this case is a nice open problem. For an elaborate comparison of the different upper and lower bounds, see Table 5.1.

Techniques. From a technical point of view, the proof uses a number of different algebraic techniques to expand the gap between orthogonal and non-orthogonal vectors from the OVP problem. For the $\{-1, 1\}$ we use an enhanced, deterministic version of the "Chebyshev embedding" [179], while for the interesting $\{0, 1\}$ part, we initiate a study of embeddings for restricted alphabets.

Inner product LSH lower bounds

In the second part of the paper we focus on LSH functions for signed and unsigned IPS. We investigate the gap between the collision probability P_1 of vectors with inner product (or absolute inner product) larger than s and the collision probability P_2 of vectors with inner product (or absolute inner product) smaller than cs. As a special case, we get the impossibility result in [139, 167], that there cannot exist an asymmetric LSH for unbounded query vectors. Specifically we get the following theorem:

Theorem 20. *Consider an (s, cs, P_1, P_2)-asymmetric LSH for signed IPS when data and query domains are d-dimensional balls with unit radius and radius U respectively. Then, the following upper bounds on $P_1 - P_2$ apply:*

(cs, s) **join problem**	**Ref.**	**Hard apx.**	**Possible**	**Hard apx.**	**Possible apx.**
Signed, $\{-1,1\}^d$	*new*	$c > 0$		$\frac{\log(s/d)}{\log(cs/d)} > 0$	
Unsigned, $\{-1,1\}^d$	[106]		$c < n^{-\epsilon}$	$\frac{\log(s/d)}{\log(cs/d)} \geq 1 - o\left(\frac{1}{\log n}\right)$	$\frac{\log(s/d)}{\log(cs/d)} = 1 - \epsilon$
	new	$c \geq e^{-o\left(\frac{\sqrt{\log n}}{\log\log n}\right)}$	$c < n^{-\epsilon}$	$\frac{\log(s/d)}{\log(cs/d)} \geq 1 - o\left(\frac{1}{\sqrt{\log n}}\right)$	$\frac{\log(s/d)}{\log(cs/d)} = 1/2 - \epsilon$
Unsigned, $\{0,1\}^d$	[168]				$\frac{\log(s/d)}{\log(cs/d)} = 1 - \epsilon$
	new	$c \geq 1 - o(1)$	$c < n^{-\epsilon}$	$\frac{\log(s/d)}{\log(cs/d)} \geq 1 - o\left(\frac{1}{\log n}\right)$	

Table 5.1: The table describes the ranges of approximations for (cs, s) joins that are hard and possible to do in subquadratic time, when parametrized in terms of c (second and third column) or $\log(s/d)/\log(cs/d)$ ratio (fourth and fifth column). Any algorithm for subquadratic join, which overlap with these ranges, would contradict the OVP. The permissible approximations are those ranges for which truly subquadratic algorithms are known. The upper bounds cited to [106] use fast matrix multiplication, whereas the rest don't and are usable as data structures. The bounds not cited elsewhere are new in this paper, though we are aware that other people have noted the hardness of signed $\{-1,1\}$ join and the data structure for $\{0,1\}$ join.

1. *if $d \geq 1$ and $s \leq \min\{cU, U/(4\sqrt{d}\}$, we have $P_1 - P_2 = O\left(1/\log(d \log_{1/c}(U/s))\right)$ for signed and unsigned IPS;*
2. *if $d \geq 2$ and $s \leq U/(2d)$, we have $P_1 - P_2 = O\left(1/\log(dU/(s(1-c)))\right)$ for signed IPS;*
3. *if $d > \Theta\left(U^5/(c^2 s^5)\right)$ and $s \leq U/8$, we have $P_1 - P_2 = O\left(\sqrt{s/U}\right)$ for signed and unsigned IPS.*

It follows that, for any given dimension d, there cannot exist an asymmetric LSH when the query domain is unbounded.

Discussion. The upper bounds for $P_1 - P_2$ translate into lower bounds for the ρ factor, as soon as P_2 is fixed. To the best of our knowledge, this is the first lower bound on ρ that holds for asymmetric LSH. Indeed, previous results [132, 142] have investigated lower bounds for symmetric LSH and it is not clear if they can be extended to the asymmetric case.

Techniques. The starting point of our proof is the same as in [139]: Use a collision matrix given by two sequences of data and query vectors that force the gap to be small. The proof in [139] then applies an asymptotic analysis of the margin complexity of this matrix [174], and it shows that for any given value of $P_1 - P_2$ there are sufficiently large data and query domains for which the gap must be smaller. Unfortunately, due to their analysis, an upper bound on the gap for a given radius U of the query domain is not possible, and so the result does not rule out very large gaps for small domains. Our method also highlights a dependency of the gap on the dimension, which is missing in [139]. In addition, our proof holds for $d = 1$ and only uses purely combinatorial arguments.

IPS upper bounds

In the third part we provide some insights on the upper bound side. We first show that it is possible to improve the asymmetric LSH in [139, 168] by just plugging the best known data structure for Approximate Near Neighbor for ℓ_2 on a sphere [31] into the reduction in [139, 38]. With data/query points in the unit ball, this LSH reaches $\rho = (1-s)/(1+(1-2c)s)$. In the $\{0,1\}$ domain, this LSH improves upon the state of the art [168] for some ranges of c and s.

Then we show how to circumvent the impossibility results in [139, 167] by showing that there exists a symmetric LSH when the data and query space coincide by allowing the bounds on collision probability to not hold when the data and query vectors are identical.

We conclude by describing a data structure based on the linear sketches for ℓ_p in [18] for unsigned (cs, s) search: for any given $0 < \kappa \leq 1/2$, the data structure yields a $c = 1/n^\kappa$ approximation with $\tilde{O}\left(dn^{2-2/\kappa}\right)$ construction time and $\tilde{O}\left(dn^{1-2/\kappa}\right)$ query time. Theorem 18 suggests that we cannot substantially improve the approximation with similar performance.

The last data structure allows us to reach truly subquadratic time for $c = 1/n^\kappa$ for the unsigned version in the $\{0,1\}$ and $\{-1,1\}$ domains for all value s. We note that the result in [106] also reaches subquadtratic time for the $\{-1,1\}$ case. However, it exploits fast matrix multiplication, whereas our data structure does not.

5.1.2 Previous work

Similarity join

Similarity join problems have been extensively studied in the database literature (e.g. [53, 57, 66, 98, 99, 121, 124, 170, 181, 182, 190]), as well as in information retrieval (e.g. [42, 72, 191]), and knowledge discovery (e.g. [6, 40, 183, 193, 195]). Most of the literature considers algorithms for particular metrics (where the task is to join tuples that are near according to the metric), or particular application areas (e.g. near-duplicate detection). A distinction is made between methods that approximate distances in the sense that we only care about distances up to some factor $c > 1$, and methods that consider exact distances. Known exact methods do not guarantee subquadratic running time. It was recently shown how approximate LSH-based similarity join can be made I/O-efficient [149].

IPS join

The inner product similarity for the case of *normalized* vectors is known as "cosine similarity" and it is well understood [52, 120, 158]. While the general case where vectors may have any length appears theoretically challenging, *practically* efficient indexes for unsigned search were proposed in [158, 110], based on tree data structures combined with a branch-and-bound space partitioning technique similar to k-d trees, and in [38] based on principal component axes trees. For document term vectors Low and Zheng [123] showed that unsigned search can be sped up using matrix compression ideas. However, as many similarity search problems, the exact version considered in these papers suffers from the curse of dimensionality [184].

The efficiency of approximate IPS approaches based on LSH is studied in [167, 139]. These papers show that a traditional LSH does exist when the data domain is the unit ball and the query domain is the unit sphere, while it does not exist when both domains are the unit ball (the claim automatically applies to any radius by suitably normalizing vectors). On the other hand an asymmetric LSH exists in this case, but it cannot be extended to the unbounded domain $\mathbb{R}^d$. An asymmetric LSH for binary inner product is proposed in [168]. The unsigned version is equivalent to the signed one when the vectors are non-negative.

Algebraic techniques

Finally, recent breakthroughs have been made on the (unsigned) join problem in the approximate case as well as the exact. Valiant [179] showed how to reduce the problem to matrix multiplication, when $cs \approx O(\sqrt{n})$ and $s \approx O(n)$, significantly improving on the asymptotic time complexity of approaches based on LSH. Recently this technique was improved by Karppa et al. [106], who also generalized the sub-quadratic running time to the case when $\log(s)/\log(cs)$ is small. In another surprising development Alman and Williams [15] showed that for $d = O(\log n)$ dimensions, truly subquadratic algorithms for the *exact* IPS join problem on binary vectors is possible. Their algorithm is based on an algebraic technique (probabilistic polynomials) and tools from circuit complexity.

5.2 Hardness of IPS join

We first provide an overview of OVP and of the associated conjecture in next Section 5.2.1. Then, in Section 5.2.2, we prove Theorem 18 by describing some reductions from the OVP to signed/unsigned joins.

5.2.1 Preliminaries

The Orthogonal Vectors Problem (OVP) is defined as follows:

Definition 16 (OVP). *Given two sets P and Q, each one containing n vectors in $\{0,1\}^d$, detect if there exist vectors $p \in P$ and $q \in Q$ such that $p^T q = 0$.*

OVP derives its hardness from the Strong Exponential Time Hypothesis (Williams [186]), but could potentially be true even if SETH is not. We will therefore assume the following plausible conjecture:[3]

Conjecture 3 (OVP, [186]). *For every constant $\epsilon > 0$, there is no algorithm for OVP with $|P| = |Q| = n$ and dimension $d = \omega(\log n)$ running in $O(n^{2-\epsilon})$ time.*

The conjecture does not hold for $d = O(\log n)$: recently Abboud et al. [2] have proposed an algorithm for OVP running in time $n^{2-1/O(\gamma \log^2 \gamma)}$, when $d = \gamma \log n$. Thus, in order to disprove OVP, an algorithm must be strongly subquadratic when $d = \gamma \log n$ for *all* constant $\gamma > 0$.

[3] We will use the name, OVP, for the problem as well as the conjecture. Sorry about that.

The OVP conjecture, as usually stated, concerns the case where the two sets have equal size. However in order to eventually show hardness for data structures, we consider the following generalization of OVP, which follows directly from the original:

Lemma 5.2.1 (Generalized OVP). *Suppose that there exist constants $\epsilon > 0$ and $\alpha > 0$, and an algorithm such that for $d = \omega(\log n)$ the algorithm solves OVP for $P, Q \subseteq \{0,1\}^d$ where $|P| = n^\alpha$ and $|Q| = n$ in time $O(n^{1+\alpha-\epsilon})$. Then OVP is false.*

Proof. Without loss of generality assume $\alpha \leq 1$ (otherwise is enough to invert the role of P and Q). Suppose we have an algorithm running in time $O(n^{1+\alpha-\epsilon})$ for some $\epsilon > 0$. Take a normal OVP instance with $|P| = |Q| = n$. Split P into chunks P_i of size n^α and run the OVP algorithm on all pairs (P_i, Q). By our assumption this takes time $n^{1-\alpha}O(n^{1+\alpha-\epsilon}) = O(n^{2-\epsilon})$, contradicting OVP. □

5.2.2 Reductions from OVP

In this section we prove Theorem 18, about hardness of approximate joins. We will do this by showing the existence of certain efficient 'gap embeddings' that make orthogonality discoverable with joins. We need the following definition:

Definition 17 (Gap Embedding). *An unsigned (d_1, d_2, cs, s)-gap embedding into the domain $\mathcal{A}$ is a pair of functions $(f, g) : \{0,1\}^{d_1} \to \mathcal{A}^{d_2'}$, where $d_2' \leq d_2$, $\mathcal{A} \subseteq \mathbb{R}$, and for any $x, y \in \{0,1\}^{d_1}$:*

$$|f(x)^T g(y)| \geq s \quad when \quad x^T y = 0$$
$$|f(x)^T g(y)| \leq cs \quad when \quad x^T y \geq 1$$

A 'signed embedding' is analogous, but without the absolute value symbols. We further require that the functions f and g can be evaluated in time polynomial to d_2.

Gap embeddings connect to the join problem, by the following technical lemma:

Lemma 5.2.2. *Suppose there exist a join algorithm for (un)signed (cs, s)-join over $\mathcal{A}$ and a family of (un)signed $(d, 2^{o(d)}, cs, s)$-gap embeddings into $\mathcal{A}$, for all d large enough.*

- *For given constants $\alpha \geq 0$, and $\epsilon > 0$, the algorithm has running time $d^{O(1)} n^{1+\alpha-\epsilon}$ when $|Q| = n$ and $|P| = n^\alpha$ for all n and d large enough.*
- *The embedding has can be evaluated in time $d_2^{O(1)}$.*

Then OVP can be solved in $n^{1+\alpha-\epsilon}$ time, and the conjecture is false.

Proof. First notice that for any function $d_2 = 2^{o(d)}$ we can take $d = \omega(\log n)$ growing slowly enough that $d_2 = n^{o(1)}$.

To see this, assume $d_2(d) = 2^{f(d)}$ where $f(d) = o(d)$. Then we have $f(d(n)) = o(d(n)) = o(1)d(n)$ that is $f(d(n)) = f'(n)d(n)$ for some $f'(n) = o(1)$. Now take $d(n) = \frac{\log n}{\sqrt{f'(n)}} = \omega(\log n)$ and we get $d_2(d(n)) = 2^{f'(n)d(n)} = 2^{\sqrt{f'(n)}\log n} = n^{o(1)}$ as desired.

Hence, there is a family of $(d(n), d_2(n), cs, s)$-gap embeddings for all n, where $d(n) = \omega(\log n)$ and $d_2(n) = n^{o(1)}$. By the generalized OVP lemma, for large enough n, we can thus take a hard OVP instance with $|Q| = n$, $|P| = n^{\alpha}$ and dimension $d(n)$. Apply the coresponding gap embedding, (f, g), to the instance, such that the maximum inner product between $f(P)$, $g(Q)$ is at least s if the OVP instance has an orthogonal pair and $\leq cs$ otherwise. Now run the algorithm for (un)signed (cs, s) join on $(f(P), g(Q))$, which produces the orthogonal pair, if it exists.

It remains to show that the running time of the above procedure is $O(n^{1+\alpha-\epsilon'})$ for some $\epsilon' > 0$. But this is easy, since by assumption, performing the embedding takes time $n^{1+o(1)}$, and nunning the algorithm on vectors of dimension $n^{o(1)}$ takes time $n^{1+\alpha-\epsilon+o(1)}$. So letting $\epsilon' = \epsilon/2$ suffices. □

The last ingredient we need to show Theorem 18 is a suitable family of embeddings to use with Lemma 5.2.2:

Lemma 5.2.3. *We can construct the following gap embeddings:*

1. *A signed $(d, 4d-4, 0, 4)$-embedding into $\{-1, 1\}$.*
2. *An unsigned $(d, (9d)^q, (2d)^q, (2d)^q e^{q/\sqrt{d}}/2)$-embedding into $\{-1, 1\}$, for any $q \in \mathbb{N}_+$, $d > 1$.*
3. *An unsigned $(d, k2^{d/k}, k-1, k)$-embedding into $\{0, 1\}$, for any integer $1 \leq k \leq d$.*

Proof. We will use the following notation in our constructions: Let $x \boxplus y$ be the concatenation of vectors x and y;[4] Let x^n mean x concatenated with itself n times;[5] And let $x \boxtimes y$ mean the vectorial representation of the outer product xy^T. Tensoring is interesting because of the following folklore property: $(x_1 \boxtimes x_2)^T (y_1 \boxtimes y_2) = \text{trace } (x_1 x_2^T)^T (y_1 y_2^T) = \text{trace } x_2 (x_1^T y_1) y_2^T = (x_1^T y_1)(x_2^T y_2)$.

(Embedding 1) The signed embedding is a simple coordinate wise construction:

$$\hat{f}(0) := (1, -1, -1) \qquad \hat{g}(0) := (1, 1, -1)$$
$$\hat{f}(1) := (1, 1, 1) \qquad \hat{g}(1) := (-1, -1, -1)$$

such that $\hat{f}(1)^T \hat{g}(1) = -3$ and $\hat{f}(0)^T \hat{g}(1) = \hat{f}(1)^T \hat{g}(0) = \hat{f}(0)^T \hat{g}(0) = 1$. This, on its own, gives a $(d, 3d, d-4, d)$ embedding, as non orthogonal vectors need to have at least one (1,1) at some position.

We can then translate all the inner products by $-(d-4)$:

$$f(x) := \hat{f}(x_1) \boxplus \cdots \boxplus \hat{f}(x_n) \boxplus 1^{d-4}$$
$$g(x) := \hat{g}(x_1) \boxplus \cdots \boxplus \hat{g}(x_n) \boxplus (-1)^{d-4}$$

which gives the $(d, 4d-4, 0, 4)$ embedding we wanted. Note that the magnitudes of non orthogonal vectors may be large ($-4d+4$), but we do not care about those for signed embeddings.

[4] $\boxplus$ for concatenation and $\boxtimes$ for tensoring stresses their dual relationship with + and × on the inner products in the embedded space. We note however that in general, it is only safe to commute $\boxplus$'es and $\boxtimes$'es in an embedding (f, g), when both f and g are commuted equally.

[5] If we wanted to further stress the duality between construction and embedding, we could define $\vec{n}$ to be the all 1 vector of length n. Then $\vec{n} \boxtimes x$ would stand for repeating x n times.

(Embedding 2) We recall the recursive definition of the q-th order Chebyshev polynomial of first kind, with $q \geq 0$ (see, e.g., [5] page 782):

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_q(x) &= 2xT_{q-1}(x) - T_{q-2}(x) \end{aligned}$$

The polynomials have the following properties [179]:

$$\begin{aligned} |T_q(x)| &\leq 1 \quad \text{when} \quad |x| \leq 1 \\ |T_q(1+\epsilon)| &\geq e^{q\sqrt{\epsilon}} \quad \text{when} \quad 0 < \epsilon < 1/2 \end{aligned}$$

We use the same coordinate wise transformation as in the signed embedding, but instead of translating by a negative value, we translate by adding $d+2$ ones, giving a $(d, 4d+2, 2d-2, 2d+2)$ unsigned embedding. Let the vectors created this way be called x and y.

On top of that, we would like to construct an embedding for the polynomial $T_q(u/2d)$, where T_q is the qth order Chebyshev polynomial of the first kind. However since this will not in general be interger, there is no hope for constructing it using $\{-1, 1\}$.

Luckily it turns out we can construct an embedding for $b^q T_q(u/b)$ for any integers b and q. Let (f_q, g_q) be the qth embedding of this type, defined by:

$$\begin{aligned} f_0(x),\ g_0(y) &:= 1,\ 1 \\ f_1(x),\ g_1(y) &:= x,\ y \\ f_q(x) &:= (x \boxtimes f_{q-1}(x))^2 \boxplus f_{q-2}(x)^{(2d)^2} \\ g_q(y) &:= (y \boxtimes g_{q-1}(y))^2 \boxplus (-g_{q-2}(y))^{(2d)^2} \end{aligned}$$

We make the following observations:

- If x and y are $\{-1, 1\}$ vectors, then so are $f_q(x)$ and $g_q(y)$.
- The inner product of the embedded vectors, $f_q(x)^T g_q(x)$ is a function of the original inner product:

  $$\begin{aligned} f_0(x)^T g_0(y) &= 1 \\ f_1(x)^T g_1(y) &= x^T y \\ f_q(x)^T g_q(y) &= 2x^T y\ f_{q-1}(x)^T g_{q-1}(y) \\ &\quad - (2d)^2 f_{q-2}(x)^T g_{q-2}(y) \end{aligned}$$

  Indeed it may be verified from the recursive definition of T_q that $f_q(x)^T g_q(y) = (2d)^n T_q(x^T y/2d)$ as wanted.

- Let d_q be the dimension of $f_q(x)$ and $g_q(y)$. Then we have:

  $$\begin{aligned} d_0 &= 1 \\ d_1 &= 4d - 4 \\ d_q &= 2(4d-4)d_{q-1} + (2d)^2 d_{q-2} \end{aligned}$$

It can be verified that $d_q \leq (9d)^q$ for any $q \geq 0$ and $d \geq 8$. Interestingly the $(2d)^2$ concatenations don't increase d_q significantly, while $d^{2+\epsilon}$ for any $\epsilon > 0$ would have killed the simple exponential dependency.

- Finally, with dynamic programming, we can compute the embeddings in linear time in the output dimension. This follows from induction over q.

Putting the above observations together, we have for any integer $q \geq 0$ a $(d, (9d)^q, (2d)^q, (2d)^q T_q(1 + 1/d))$ embedding. By the aforementioned properties of the Chebyshev polynomials, we have the desired embedding. We note that the Chebyshev embedding proposed by Valiant [179] can provide similar results; however, our construction is deterministic, while Valiant's is randomized.

(Embedding 3) The third embedding maps into $\{0, 1\}$. The difficulty here is that without -1, we cannot express subtraction as in the previous argument. It turns out however, that we can construct the following polynomial:

$$(1 - x_1 y_1)(1 - x_2 y_2) \cdots (1 - x_d y_d)$$

since $\{0, 1\}$ is closed under tensoring and

$$1 - x_i y_i = (1 - x_i, 1)^T (y_i, 1 - y_i)$$

where $1 - x_i$, y_i and $1 - y_i$ are both in $\{0, 1\}$. The polynomial has the property of being 1 exactly when the two vectors are orthogonal and 0 otherwise.

However we cannot use it directly with Lemma 5.2.2, as it blows up the dimension too much, $d_2 = 2^{d_1}$. Instead we "chop up" the polynomial in k chunks and take their sum:

$$\sum_{i=0}^{k-1} \prod_{j=1}^{d/k} (1 - x_{ik/d+j} y_{ik/d+j})$$

This uses just $d_2 = k 2^{d/k}$ dimensions, which is more manageble. If k does not divide d, we can let the last "chop" of the polynomial be shorter than d/k, which only has the effect of making the output dimension slightly smaller.

Finally we get the gap $s = k$ and $cs = k - 1$. The later follows because for non orthogonal vectors, at least one chunk has a $(1 - x_i y_i)$ terms which evaluates to zero. We thus have a $(d, k2^{d/k}, k - 1, k)$-embedding into $\{0, 1\}$. The explicit construction is thus:

$$f(x) := \mathop{\boxplus}\limits_{i=0}^{k-1} \mathop{\boxtimes}\limits_{j=1}^{d/k} (1 - x_{ik/d+j}, 1)$$

$$g(x) := \mathop{\boxplus}\limits_{i=0}^{k-1} \mathop{\boxtimes}\limits_{j=1}^{d/k} (y_{ik/d+j}, 1 - y_{ik/d+j})$$

And the running time is linear in the output dimension. □

Finally we parametize and prove Theorem 18.

Proof. (Theorem 18) We first prove the bounds parametrized by c. To get the strongest possible results, we want to get c as small as possible, while keeping the dimension bounded by n^δ for some $\delta > 0$.

1. The first embedding is already on the form $(d, 2^{o(d)}, 0, 4)$, showing theorem 1 for signed $(0, 4)$ join, and thus any $c > 0$.
2. In the second embedding we can take q to be any function in $o\left(\frac{d}{\log d}\right)$, giving us a family of $(d,\ 2^{o(d)},\ 2^{o(d)},\ 2^{o(d)}e^{o\left(\frac{\sqrt{d}}{\log d}\right)})$. Thus by lemma 5.2.2, and for a small enough $d = \omega(\log n)$ in OVP, we get that even $c \le e^{-o\left(\frac{\sqrt{\log n}}{\log\log n}\right)} \le 1/\text{polylog(n)}$ is hard.
3. Finally for the third embedding, we can pick any $k = \omega(1)$ and less than d, to get a family of $(d, 2^{o(d)}, k-1, k)$ embeddings. Again picking $d = \omega(\log n)$ small enough in OVP, we have by lemma 5.2.2 that $c \le (k-1)/k = 1 - 1/k = 1 - o(1)$ is hard. Notice that this means any c not bounded away from 1 is hard.

□

For the bounds parametrized by $\log(s)/\log(cs)$, we need to tweak our families slightly differently. This in turn allows for hardness for shorter vectors than used in the previous results.

Proof. (Theorem 19)

For embedding 1 the result follows directly as $\log(s/d)/\log(cs/d) \to 0$ as $c \to 0$.

We have to remember that the results in theorem 18 are stated in terms of normalized s. For embedding 2 we calculate:

$$\begin{aligned}\frac{\log(s/d_2)}{\log(cs/d_2)} &= \frac{q\log(2/9) + q/\sqrt{d} - \log 2}{q\log(2/9)} \\ &= 1 - \frac{1}{\log(9/2)\sqrt{d}} + \frac{\log 2}{q\log(9/2)} \\ &= 1 - o\left(1/\sqrt{\log n}\right)\end{aligned}$$

Where in the last step we have taken $q = \sqrt{d}$ and $d = \omega(\log n)$ as by the OVP conjecture.

It is important to notice that we could have taken q much larger, and still satisfied lemma 5.2.2. However that wouldn't have improved the result, except by more quickly vanishing second order asymptitic terms. What we instead gain from having $d_2 = (9d)^{\sqrt{d}}$ is that, as one can verify going through the lemma, we show hardness for any join algorithm running in time $d^{o(\frac{\log d}{\log\log^2 d})} n^{1+\alpha-\epsilon}$. That is, the hardness holds even for algorithms with a much higher dependency on the dimension than polynomial.

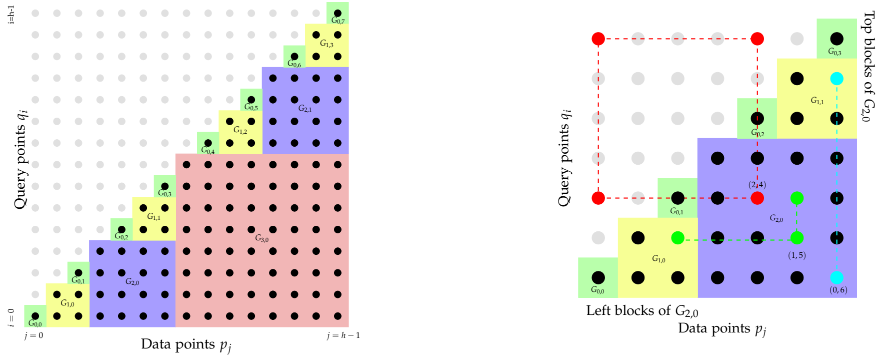


Figure 5.1: On the left, a 15×15 grid: black nodes are P_1-nodes, gray nodes are P_2-nodes; the colored blocks denote the partitioning of the lower triangle into squares. On the right, a zoom of the $G_{2,0}$ square and of its left and top squares: the red nodes collide under a $(2,4)$-shared function; the green nodes collide under a $(1,5)$-partially shared function; the cyan node collide under a $(0,6)$-proper function (specifically, row proper).

Similarly, we calculate for embedding 3:

$$\begin{aligned}\frac{\log(s/d_2)}{\log(cs/d_2)} &= \frac{\log \frac{k}{k2^{d/k}}}{\log \frac{k-1}{k2^{d/k}}}\\ &= 1 - \frac{k\log(1+1/(k-1))}{d+k\log(1+1/(k-1))}\\ &= 1 - 1/d + O(1/(kd))\\ &= 1 - o(1/\log n)\end{aligned}$$

Where we have taken $k = d$ and $d = \omega(\log n)$ as by the OVP conjecture.

Taking $k = d$ means that d_2 is only $2d$.

□

5.3 Limitations of LSH for IPS

We provide an upper bound on the gap between P_1 and P_2 for an (s, cs, P_1, P_2)-asymmetric LSH for signed/unsigned IPS. For the sake of simplicity we assume the data and query domains to be the d-dimensional balls of radius 1 and $U \geq 1$, respectively. The bound holds for a fixed set of data vectors, so it applies also to data dependent LSH [31]. A consequence of our result is that there cannot exist an asymmetric LSH for any dimension $d \geq 1$ when the set of query vectors is unbounded, getting a result similar to that of [139], which however requires even the data space to be unbounded and $d \geq 2$.

We firsts show in Lemma 5.3.1 that the gap $P_1 - P_2$ can be expressed as a function of the length h of two sequences of query and data vectors with suitable collision properties. Then we provide the proof of the aforementioned Theorem 20, where we derive some of such sequences and then apply the lemma.

Lemma 5.3.1. *Suppose that there exists a sequence of data vectors $P = \{p_0, \dots, p_{n-1}\}$ and a sequence of query vectors $Q = \{q_0, \dots, q_{n-1}\}$ such that $q_i^T p_j \geq s$ if $j \geq i$ and $q_i^T p_j \leq cs$ otherwise (resp., $|q_i^T p_j| \geq s$ if $j \geq i$ and $|q_i^T p_j| \leq cs$ otherwise) . Then any (s, cs, P_1, P_2)-asymmetric LSH for signed IPS (resp., unsigned IPS) must satisfy $P_1 - P_2 \leq 1/(8 \log n)$.*

Proof. For the sake of simplicity we assume that $n = 2^\ell - 1$ for some $\ell \geq 1$; the assumption can be removed by introducing floor and ceiling operations in the proof. Let $\mathcal{H}$ denote an (s, cs, P_1, P_2)-asymmetric LSH family of hash functions, and let h be a function in $\mathcal{H}$. The following argument works for signed and unsigned IPS.

Consider the $n \times n$ grid representing the collisions between $Q \times P$, that is, a node (i, j) denotes the query-data vectors q_i and p_j. We say that a node (i, j), with $0 \leq i, j < n$, collides under h if vectors q_i and p_j collide under h. By definition of asymmetric LSH, all nodes with $j \geq i$ must collide with probability at least P_1, while the remaining nodes collide with probability at most P_2. We use *lower triangle* to refer to the part of the grid with $j \geq i$ and P_1*-nodes* to refer to the nodes within it; we refer to the remaining nodes as P_2*-nodes*.

We partition the lower triangle into squares of exponentially increasing side as shown in Figure 5.1. Specifically, we split the lower triangle into *squares* $G_{r,s}$ for every r and s with $0 \leq r < \log(n+1) = \ell$ and $0 \leq s < (n+1)/2^{r+1} = 2^{\ell-r-1}$, where $G_{r,s}$ includes all nodes in the square of side 2^r and top-left node $((2s+1)2^r - 1, (2s + 1)2^r - 1)$. For a given square $G_{r,s}$, we define the *left squares* (resp., *top squares*) to be the set of squares that are on the left (resp., top) of $G_{r,s}$. We note that the left squares (resp., top squares) contain 2^{r-i-1} squares of side 2^i for any $0 \leq i < r$ and all P_1-nodes with $s2^{r+1} \leq i, j < (2s+1)2^r - 1$ (resp., $(2s+1)2^r - 1 < i, j \leq (s+1)2^{r+1} - 2$) .

We define the *mass* $m_{i,j}$ of a node (i, j) to be the collision probability, under $\mathcal{H}$, of q_i and p_j. We split the mass of a P_1-node into three contributions called shared mass, partially shared mass, and proper mass, all defined below. Consider each P_1-node (i, j) and each function $h \in \mathcal{H}$ where (i, j) collides. Let $G_{r,s}$ be the square containing (i, j) and let $K_{h,i,j}$ denote the set of P_1-nodes (i', j') on the left side of the same row or on the top of the same column of (i, j) (i.e., $i' = i$ and $i \leq j' < j$, or $j' = j$ and $i < i' \leq j$) and with the same hash value of (i, j) under h (i.e., $h(i) = h(j) = h(i') = h(j')$). Clearly all nodes in $K_{h,i,j}$ collide under h. For the given node (i, j), we classify h as follows (see Figure 5.1 for an example):

- (i, j)*-shared function.* $K_{h,i,j}$ contains at least a node (i, j') in a left square, and at least a node (i', j) in a top square.

- (i, j)*-partially shared function.* Function h is not in case 1 and $K_{h,i,j}$ contains at least a node node (i, j') with $j' < j$, and at least a node (i', j) with $i' > i$. That is, $K_{h,i,j}$ contains only nodes in $G_{r,s}$ and in the left blocks, or only nodes in $G_{r,s}$ and in the top blocks.

- (i, j)*-proper function.* $K_{h,i,j}$ contains no points (i, j') for any $i \leq j' < j$ or contains no points (i', j) for any $i < i' \leq j$. That is, $K_{h,i,j}$ cannot contain at the same time a point in a left square and a point in a top square. Function h is said row (resp., column) proper if there are no nodes in the same row (resp., column). We break ties arbitrary but consistently if $K_{h,i,j}$ is empty.

The *shared mass* $m^s_{i,j}$ is the sum of probabilities of all (i,j)-shared functions. The *partially shared mass* $m^{ps}_{i,j}$ is the sum of probabilities of all (i,j)-partially shared functions. The *proper mass* $m^p_{i,j}$ is the sum of probabilities of all (i,j)-proper functions (the row/column proper mass includes only row/column proper functions). We have $m_{i,j} = m^p_{i,j} + m^{ps}_{i,j} + m^s_{i,j}$. The *mass* $M_{r,s}$ of a square $G_{r,s}$ is the sum of the masses of all its nodes, while the *proper mass* $M^p_{r,s}$ is the sum of proper masses of all its nodes. The sum of row proper masses of all nodes in a row is at most one since a function h is row proper for at most one node in a row. Similarly, the sum of column proper masses of all nodes in a column is at most one. Therefore, we have that $\sum_{r,s} M^p_{r,s} \leq 2n$.

We now show that $\sum_{(i,j)\in G_{r,s}} m^s_{i,j} \leq 2^{2r}P_2$ for every $G_{r,s}$. Consider a node (i,j) in a given $G_{r,s}$. For each (i,j)-shared function h there is a P_2-node colliding under h: indeed, $K_{h,i,j}$ contains nodes (i,j') in the left blocks and (i',j) in the top blocks with $h(i) = h(j) = h(i') = h(j')$ (i.e., $s2^{r+1} \leq j' < (2s+1)2^r - 1$ and $(2s+1)2^r - 1 < i' \leq (s+1)2^{r+1} - 2$); then node (i',j') is a P_2-node since $i' > j'$ and collides under h. By considering all nodes in $G_{r,s}$, we get that all the P_2-nodes that collide in a shared function are in the square of side 2^{r-1} and bottom-right node in $((2s+1)2^r, (2s+1)2^r - 2)$. Since these P_2-nodes have total mass at most $2^{2r}P_2$, the claim follows.

We now prove that $\sum_{(i,j)\in G_{r,s}} m^{ps}_{i,j} \leq 2^{r+1}M^p_{r,s}$. A (i,j)-partially shared function is (i',j) or (i,j')-proper for some $i' < i$ and $j' > j$, otherwise there would be a node in left blocks and a node in top blocks that collide with (i,j) under h, implying that h cannot be partially shared. Since an (i,j)-proper function is partially shared for at most 2^{r+1} nodes in $G_{r,s}$, we get

$$\sum_{(i,j)\in G_{r,s}} m^{ps}_{i,j} \leq 2^{r+1} \sum_{(i,j)\in G_{r,s}} m^p_{i,j} = 2^{r+1}M^p_{r,s}.$$

By the above two bounds, we get

$$M_{r,s} \leq \sum_{(i,j)\in G_{r,s}} m^p_{i,j} + m^{ps}_{i,j} + m^s_{i,j} \leq (2^{r+1}+1)M^p_{r,s} + 2^{2r}P_2.$$

Since $M_{r,s} \geq 2^{2r}P_1$ we get $M^p_{r,s} \geq (2^{r-1}-1)(P_1 - P_2)$. By summing among all squares, we get

$$2n \geq \sum_{r=0}^{\ell-1} \sum_{s=0}^{2^{\ell-r-1}-1} M^p_{r,s} > (P_1 - P_2)\frac{n \log n}{4}$$

from which the claim follows. □

We are now ready to prove Theorem 20.

Proof. (Theorem 20) The upper bounds to $P_1 - P_2$ in the different cases follow by applying Lemma 5.3.1 with different sequences of query and data vectors. We anticipate that in all three cases the gap $P_1 - P_2$ becomes 0 if the query ball is unbounded (i.e., $U = +\infty$), and hence there cannot exist an asymmetric LSH with $P_1 > P_2$.

First case. We now show that there exist data and query sequences of length $n = \Theta(md)$, with $m = \Theta\left(\log_{1/c}(U/s)\right)$, for signed and unsigned IPS in $d \geq 1$ dimensions if $s = O\left(U/\sqrt{d}\right)$. Note that $m \geq 1$ since we assume $s \leq cU$. As a warm-up, we start with $d = 1$. Let $Q = \{q_i, \forall\ 0 \leq i < n\}$ and $P = \{p_j, \forall\ 0 \leq j < n\}$ with

$$q_i = Uc^i, \qquad p_j = s/(Uc^j). \tag{5.1}$$

Let $p_j \in P$ and $q_i \in Q$. We get $p_j^T q_i = c^{i-j}s$: if $j \geq i$ then $p_j^T q_i \geq s$ and $p_j^T q_i \leq cs$ otherwise. Data and query vectors are respectively contained in the unit ball and in the ball of radius U since $i, j < \Theta\left(\log_{1/c}(U/s)\right)$. Being the sequences P and Q of length m, the claim follows.

Let now $d \geq 2$ and assume for the sake of simplicity $d = 2d'$ (the general case just requires some more tedious computations). Consider the following sequences $Q_k = \{q_{i,k}, \forall\ 0 \leq i < m\}$ and $P_k = \{p_{j,k}, \forall\ 0 \leq j < m\}$ for each $0 \leq k < d'$, where $q_{i,k}$ and $p_{j,k}$ are d-dimensional vectors defined as follows. Denote with $q_{i,k}[t]$ the t-th coordinate of $q_{i,k}$, for $0 \leq t < d$ (similarly for $p_{j,k}$). For vector $q_{i,k}$ we have: $q_{i,k}[2k] = Uc^i$; $q_{i,k}[2t+1] = 2s$ for each $k \leq t < d'$; remaining positions are set to 0. For vector $p_{i,k}$ we have: $p_{i,k}[2k] = s/(Uc^i)$; $p_{i,k}[2k-1] = 1/2$ (only when $k > 0$); remaining positions are set to 0. Intuitively, these data and query sequences follow by constructing the 1-dimensional sequences in Equation 5.1 on d' orthogonal dimensions and then by suitably translating each sequence. As an example, for $d = 6$ we get:

$$\begin{aligned}
q_{i,0} &= (Uc^i, 2s, 0, 2s, 0, 2s) & p_{j,0} &= (s/(Uc^j), 0, 0, 0, 0, 0);\\
q_{i,1} &= (0, 0, Uc^i, 2s, 0, 2s) & p_{j,1} &= (0, 1/2, s/(Uc^j), 0, 0, 0);\\
q_{i,2} &= (0, 0, 0, 0, Uc^i, 2s) & p_{j,2} &= (0, 0, 0, 1/2, s/(Uc^j), 0).
\end{aligned}$$

The query and data sequences $Q = \{Q_0, \ldots, Q_{d'-1}\}$ and $P = \{P_0, \ldots, P_{d'-1}\}$ satisfy the hypothesis of Lemma 5.3.1. Indeed, it can be verified that: $p_{j,\ell}^T q_{i,\ell} = sc^{i-j}$ and thus $p_{j,\ell}^T q_{i,\ell} \geq s$ if $j \geq i$ and $p_{j,\ell}^T q_{i,\ell} \leq cs$ otherwise; $p_{j,\ell'}^T q_{i,\ell} = 0$ if $\ell' < \ell$; $p_{j,\ell'}^T q_{i,\ell} \geq s$ if $\ell' > \ell$. Further, when $s \leq U/(2\sqrt{d})$ data and query vectors are contained in balls with radius 1 and U respectively, with the exception of vectors $q_{i,k}$ for $0 \leq i < 1/(2\log(1/c))$ and $0 \leq k < d'$ which are contained in a ball of radius $2U$. However, these query vectors and the respective data vectors can be removed from the above sequences without affecting the asymptotic length. We thus get two sequences of length $n = (m - 1/(2\log(1/c)))d' = \Theta\left(d \log_{1/c}(U/s)\right)$, and the claim follows. Since all inner products are non negative, the upper bound on $P_1 - P_2$ holds for signed and unsigned IPS.

Second case. Longer query and data sequences, with length $n = \Theta(md)$ for $m = \Theta\left(\sqrt{U/(s(1-c))}\right)$, can be constructed for signed IPS when $d \geq 2$. We start considering the case $d = 2$. Let $Q = \{q_i, \forall\ 0 \leq i < m\}$ and $P = \{p_j, \forall\ 0 \leq j < m\}$ with

$$\begin{aligned}
q_i &= \left(\sqrt{sU}(1-(1-c)i), \sqrt{sU(1-c)}\right),\\
p_j &= \left(\sqrt{\frac{s}{U}}, j\sqrt{\frac{s(1-c)}{U}}\right).
\end{aligned} \tag{5.2}$$

We observe that these sequences are similar to the one used in [139]. We have $p_j^T q_i = s(1-c)(j-i) + s$: then, $p_j^T q_i \geq s$ if $j \geq i$ and $p_j^T q_i \leq cs$ otherwise. If $s \leq U/2$, data and query vectors are within balls of radius respectively 1 and U.

Let now $d \geq 2$ and assume for the sake of simplicity $d = 2d'$ (the general case just requires some more tedious computations). Consider the following sequences $Q_k = \{q_{i,k}, \forall\ 0 \leq i < m\}$ and $P_k = \{p_{j,k}, \forall\ 0 \leq j < m\}$ for each $0 \leq k < d'$, where $q_{i,k}$ and $p_{j,k}$ are d-dimensional vectors defined as follows. For vector $q_{i,k}$ we have: $q_{i,k}[2k] = \sqrt{sU}(1-(1-c)i)$; $q_{i,k}[2k+1] = \sqrt{sU(1-c)}$; $q_{i,k}[2t] = \sqrt{Us}$ for each $k < t < d'$; remaining positions are set to 0. For vector $p_{i,k}$ we have: $p_{i,k}[2k] = \sqrt{s/U}$; $p_{i,k}[2k] = j\sqrt{s(1-c)/U}$; remaining positions are set to 0. We observe that the two sequences follow by constructing the 2-dimensional sequences in Equation 5.2 on d' orthogonal planes and then suitably translate. Then, it follows that data and query sequences $P = \{P_0, \dots P_{d'-1}\}$ and $Q = \{Q_0, \dots Q_{d'-1}\}$ satisfy the hypothesis of Lemma 5.3.1 and they are respectively contained in balls or radius one and U respectively if $s \leq U/(2d)$. Being $m = nd' = O\left(d\sqrt{U/(s(1-c))}\right)$ and the claim follows. We observe that the above sequences may generate large negative inner products and then they cannot be used for unsigned IPS.

Third case. Finally, we provide an upper bound on $P_1 - P_2$ for signed and unsigned IPS that holds for $d \geq \Theta\left(U^5/(c^2 s^5)\right)$ by providing data and query sequences of length $n = 2^{\sqrt{U/(8s)}}$. Suppose there exists a family $\mathcal{Z}$ of $2n-1$ vectors such that $|z_i^T z_j| \leq \epsilon$ and $(1-\epsilon) \leq z_i^T z_i \leq (1+\epsilon)$ for any $z_i \neq z_j$, for $\epsilon = c/(2\log^2 n)$. It can be shown with the Johnson-Lindenstrauss lemma that such a family exists when $d = \Omega\left(\epsilon^{-2}\log n\right)$ (for an analysis see e.g. [150]). For notational convenience, we denote the vectors in $\mathcal{Z}$ as follows: $z_{b_0}, z_{b_0,b_1}, \dots, z_{b_0,b_1,\dots,b_{\log n-1}}$ for each possible value $b_0, \dots, b_{\log n-1} \in \{0,1\}$. Let $b_{i,\ell}$ denote the ℓ-th bit of the binary representation of i and with $\bar{b}_{i,\ell}$ its negation, where we assume $\ell = 0$ to be the most significant bit. Let $Q = \{q_i, \forall\ 0 \leq i < n\}$ and $P = \{p_j, \forall\ 0 \leq j < n\}$ with

$$q_i = \sqrt{2sU} \sum_{\ell=0}^{\log n-1} \bar{b}_{i,\ell} z_{b_{i,0},\dots b_{i,\ell-1},\bar{b}_{i,\ell}}$$

$$p_j = \sqrt{2s/U} \sum_{\ell=0}^{\log n-1} b_{j,\ell} z_{b_{j,0},\dots b_{j,\ell-1},b_{j,\ell}}$$

Since the inner product of two distinct vectors in $\mathcal{Z}$ is in the range $[-\epsilon, \epsilon]$, we have that $p_j^T q_i$ can be upper bounded as

$$\begin{aligned} p_j^T q_i \leq & \epsilon 2s(\log^2 n - \log n) + \\ & + 2s \sum_{\ell=0}^{\log n-1} b_{j,\ell}\bar{b}_{i,\ell} z_{b_{j,0},\dots b_{j,\ell-1},b_{j,\ell}} z_{b_{i,0},\dots b_{i,\ell-1},\bar{b}_{i,\ell}} \end{aligned}$$

Suppose $i > j$. Then there exists a bit position ℓ' such that $b_{i,\ell'} = 1$, $b_{j,\ell'} = 0$ and $b_{i,\ell} = b_{j,\ell}$ for all $\ell < \ell'$. We get $b_{j,\ell}\bar{b}_{i,\ell} = 0$ for all $\ell \leq \ell'$ and $z_{b_{j,0},\dots b_{j,\ell-1},b_{j,\ell}} \neq z_{b_{i,0},\dots b_{i,\ell},\bar{b}_{i,\ell}}$ for all $\ell > \ell'$. It then follows that $p_j^T q_i < \epsilon 2s \log^2 n$ when $i > j$. On the other hand we get that $p_j^T q_i$ can be lower bounded as

$$\begin{aligned} p_j^T q_i \geq & -\epsilon 2s(\log^2 n - \log n) + \\ & + 2s \sum_{\ell=0}^{\log n-1} b_{j,\ell}\bar{b}_{i,\ell} z_{b_{j,0},\dots b_{j,\ell-1},b_{j,\ell}} z_{b_{i,0},\dots b_{i,\ell-1},\bar{b}_{i,\ell}} \end{aligned}$$

Suppose $i \leq j$. Then there exists an index ℓ' such that $b_{j,\ell'} = 1$, $b_{i,\ell'} = 0$ and $b_{j,\ell} = b_{i,\ell}$ for all $\ell < \ell'$. We get $b_{j,\ell'}\bar{b}_{i,\ell'} = 1$ and $z_{b_{j,0},\dots b_{j,\ell'-1},b_{j,\ell'}} = z_{b_{i,0},\dots b_{i,\ell'-1},\bar{b}_{i,\ell'}}$. It follows that $p_j^T q_i \geq -\epsilon 2s \log^2 n + 2s$. When $d = \Omega\left((\log^5 n)/c^2\right)$, we can set $\epsilon = c/(2\log^2 n)$. From the above lower and upper bounds, it then follows that $p_j^T q_i \leq cs$ if $j < i$ and $p_j^T q_i \geq s$ if $j \geq i$ and hence the sequences Q and P satisfy the hypothesis of Lemmma 5.3.1. The data and query vectors in P and Q are respectively contained in balls of radius 1 and U: each q_i (resp., p_j) is given by the sum of at most $\log n$ vectors whose norms are not larger than $\sqrt{2sU}(1+\epsilon)$ (resp., $\sqrt{2s/U}(1+\epsilon)$); being $n = 2^{\sqrt{U/(8s)}}$ the claim follows. □

5.4 Upper bounds

This section contains three observations with implications for IPS join and its indexing version. We first notice in Section 5.4.1 that by plugging the best known LSH for ℓ_2 distance on a sphere [31] into a reduction presented in [38, 139], we get a data structure based on LSH for signed IPS with search time exponent $\rho = (1-s)/(1+(1-2c)s)$.

Then, in Section 5.4.2, we show how to circumvent the results in [139, 167] showing that symmetric LSH is not possible when the data and query domains coincide (while an asymmetric LSH does exist). We use an slightly modified definition of LSH that disregards the collision probability of 1 for pairs of identical vectors, and assume that vectors are represented with finite precision. The LSH construction uses explicit incoherent matrices built using Reed-Solomon codes [137] to implement a symmetric version of the reduction in [38, 139].

Finally, in Section 5.4.3, we solve unsigned (cs, s) join using linear sketches for ℓ_p norms from [18]. Given $\kappa \geq 2$ we obtain approximation $c = 1/n^{1/\kappa}$ in $\tilde{O}\left(dn^{2-2/\kappa}\right)$ time. Although this trade-off is not that strong, it is not far from the conditional lower bound in Theorem 18.

5.4.1 Asymmetric LSH for signed IPS

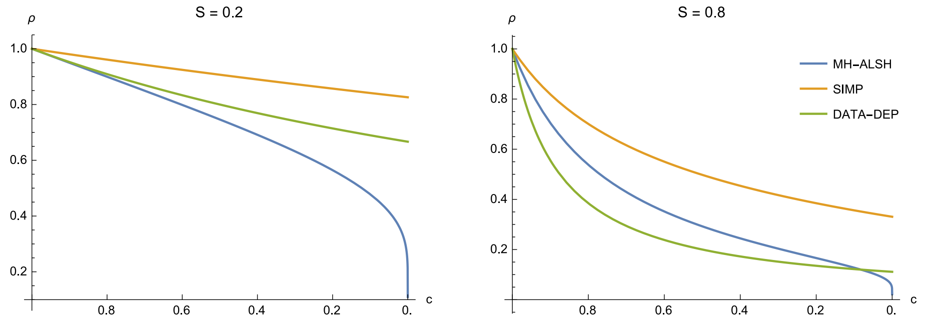


Figure 5.2: Our ρ value (DATA-DEP) compared to that of [139] (SIMP) and the binary data only of [168] (MH-ALSH).

We assume the data and query domains to be d-dimensional balls with respective radius 1 and U. Vectors are embedded into a $(d+2)$-dimensional unit sphere using the asymmetric map as in [139]: a data vector p is mapped to $(p, \sqrt{1-||p||^2}, 0)$, while a query q is mapped to $(q/U, 0, \sqrt{1-||q||^2/U^2})$. This transformation just scales the inner product by a factor U, and hence signed inner product search can be seen as an instance of ANN in ℓ_2 with distance threshold $r = \sqrt{2(1-s/U)}$ and approximation $c' = \sqrt{(1-cs/U)/(1-s/U)}$. The latter can be solved in space $O(n^{1+\rho} + dn)$ and query time $O(n^\rho)$ using the LSH construction of [31]. We get the following ρ value (for the LSH gap as well as for the exponent of the running time):

$$\rho = \frac{1}{2c'^2 - 1} = \frac{1 - s/U}{1 + (1-2c)s/U} \ . \tag{5.3}$$

In Figure 5.2, we plot the ρ values of three LSH constructions: the one proposed here (with $U = 1$), the one from [139], and the one from [168]. The latter works only for binary vectors. We point out that our bound is always stronger than the one from [139] and sometimes stronger than the one from [168], despite that the latter is tailored for binary vectors. The latter conclusion is somewhat surprising, since the data structure we obtain works for non-binary vectors as well.

In practice, one may want to use a recent LSH family from [21] that—both in theory and in practice—is superior to the hyperplane LSH from [52] used in [139].

5.4.2 Symmetric LSH for almost all vectors

Neyshabur and Srebro [139] show that an asymmetric view on LSH for signed IPS is required. Indeed they show that a symmetric LSH for signed IPS does not exist when data and query domains are balls of the same radius, while an asymmetric LSH does exist. (On the other hand, when the data domain is a ball of given radius U and the query domain is a sphere of same radius, a symmetric LSH does exist.) In this section we show that even when data and query spaces coincide a nontrivial symmetric LSH does exist if we disregard the trivial collision probability of 1 when data and query vectors are identical.

We first show how to reduce signed IPS to the case where data and query vectors lie on a unit sphere. The reduction is deterministic and maintains inner products up to an additive error ε for all vectors x, y with $x \neq y$. We then plug in any Euclidean LSH for ANN on the sphere, for example the one from [31]. This reduction treats data and query vectors identically, unlike the one from [139], and thus we are able to obtain a symmetric LSH.

Assume that vector coordinates are encoded as k-bit numbers, and that data and query vectors are in the unit ball. The idea is the following. There are at most $N = 2^{O(dk)}$ possible data vectors and queries. Imagine a collection of N unit vectors $v_1, \ldots, v_N$ such that for every $i \neq j$ one has $|v_i^T v_j| \leq \varepsilon$. Then, it is easy to check that a map of a vector p to $f(p) = (p, \sqrt{1-\|p\|^2} \cdot v_p)$ maps a vector from a unit ball to a unit sphere and, moreover, for $p \neq q$ one has $|f(p)^T f(q) - p^T q| \leq \varepsilon$.

What remains is to construct such a collection of vectors v_i. Moreover, our collection of vectors must be explicit in a strong sense: we should be able to compute v_u given a vector u (after interpreting it as an dk-bit string). Such a constructions are well known, e.g., in [137] it is shown how to build such vectors using Reed-Solomon codes. The resulting dimension is $O\left(\varepsilon^{-2} \log N\right) = O\left(kd/\varepsilon^2\right)$ [117, 137].

After performing such a reduction we can apply any state-of-the-art LSH (or data structure for ANN) for ℓ_2 norm on a sphere, e.g. from [31, 21], with distance threshold $r^2 = 2(1 - s + \epsilon)$, approximation factor $c'^2 = (1 - cs - \epsilon)/r^2$. If ε is sufficiently small we get a ρ value close to the one in (5.3). The final result is therefore a symmetric LSH for symmetric domains that does not provide any collision bound for all pairs (q, p) with $q = p$ since the guarantees on the inner product fail for these pairs. This LSH can used for solving signed (cs, s) IPS as a traditional LSH [20], although it is required an initial step that verifies whether a query vector is in the input set and, if this is the case, returns the vector q itself if $q^T q \geq s$.

5.4.3 Unsigned IPS via linear sketches

In this section we propose a linear sketch for *unsigned c-MIPS*, that can be used for solving unsigned (cs, s) join. The unsigned c-MIPS is defined as follows: given a set $P \subset \mathbb{R}^d$ of n vectors, construct a data structure that efficiently returns, for a given query vector q, a vector $p \in P$ where $|p^T q| \geq c(p'^T q)$, where p' is the vector in P with maximum absolute inner product with q. The unsigned (cs, s) join between sets P and Q can be computed by constructing a data structure for unsigned c-MIPS for vectors in P and then performs a query for each vector in Q.

Of independent interest, we notice that unsigned c-MIPS can be solved by a data structure for unsigned (cs, s) search. Let $\mathcal{D}$ be a data structure for unsigned (cs, s) search on the data set P, and suppose we are given a query q and the promise that there exists $p' \in P$ such that $p'^T q > \gamma$. Then, unsigned c-MIPS can be solved by performing on $\mathcal{D}$ the queries q/c^i for any $0 \leq i \leq \lceil \log_{1/c}(s/\gamma) \rceil$. Intuitively, we are scaling up the query q until the largest inner product becomes larger than the threshold s. We notice that γ can be also considered as the smallest inner product that can be stored according to the numerical precision of the machine.

Our data structure requires $\tilde{O}\left(dn^{2-2/\kappa}\right)$ construction time and $\tilde{O}\left(dn^{1-2/\kappa}\right)$ query time and provide a $c = 1/n^{1/\kappa}$ approximation with high probability, for any $\kappa \geq 2$. This gives an algorithm for unsigned (cs, s) join on two sets of size n requiring time $\tilde{O}\left(dn^{2-2/\kappa}\right)$. As shown in Theorem 18, we are unlikely to significant improve further the approximation factor if the OVP conjecture is true.

First, suppose we are only interested in approximating the value of $\max_p |q^t p|$ and not to find the corresponding vector. Then, the problem is equivalent to estimating $\|Aq\|_\infty$, where A is an $n \times d$ matrix, whose rows are data vectors. This problem can be tackled using linear sketches (for an overview see [188, 23]). More specifically, we use the following result from [18]: for every $2 \leq \kappa \leq \infty$ there exists a distribution over $\widetilde{O}(n^{1-2/\kappa}) \times n$ matrices Π such that for every $x \in \mathbb{R}^n$ one has:

$$\Pr_{\Pi}\left[(1-c)\|x\|_\kappa \leq \|\Pi x\|_\infty \leq (1+c)\|x\|_\kappa\right] \geq 0.99$$

for a suitable constant $0 < c < 1$. Thus, to build a data structure for computing $\|Aq\|_\infty$, we sample a matrix Π according to the aforementioned result in [18] and compute the $\tilde{O}\left(n^{1-2/\kappa}\right) \times d$ matrix $A_s = \Pi A$. Then, for every query q, we compute $\|A_s q\|_\infty$ in time $\tilde{O}\left(d \cdot n^{1-2/\kappa}\right)$, which is a $O\left(n^{1/\kappa}\right)$-approximation to $\|Aq\|_\infty$ with probability at least 0.99. Note that we can reduce the probability of error from 0.01 to $\delta > 0$ as usual, by building $O(\log(1/\delta))$ independent copies of the above data structure and reporting the median estimate.

We now consider the recovery of the vector that almost maximizes $|p^t q|$. We recover the index of the desired vector bit by bit. That is, for every bit index $0 \leq i < \log n$, we consider every binary sequence b of length i and build a data structure for the dataset containing only the vectors in $\mathcal{P}$ for which the binary representations of their indexes have prefix b. Although the number of data structures is n, the total required space is still $\tilde{O}\left(dn^{1-2/\kappa}\right)$ since each vector appears in only $\log n$ data structures. The claim stated at the beginning follows.

5.5 Conclusion

This paper has investigated different aspects of the complexity of *approximate* similarity join with inner product. In particular, we have related the hardness of this problem to the OVP conjecture. Under some assumptions on c and s, the proposed conditional lower bounds rule out algorithms for signed/unsigned (cs, s) IPS join running in $n^{2-\epsilon}$ time, for a constant $\epsilon > 0$, unless the OVP conjecture is false. Nevertheless, the data structures in section 5.4 show that it still possible to reach weak subquadratic time, and even truly subquadratic time for small values of the approximation factor.

The hardness of signed/unsigned IPS holds even for weak approximation factors when the vector domain is $\{-1, 1\}^d$. Indeed, the result holds if $c \geq 0$ for signed join, and if $c \geq e^{-o(\sqrt{\log n}/\log\log n)}$ for unsigned join. When $c < 1/n^{\Omega(1)}$, the data structure for unsigned IPS in section 5.4.3 reaches strongly subquadratic time and this gives evidence that the constraint on c of the hardness result cannot be significantly relaxed. On the other hand, when vectors are in the $\{0, 1\}^d$ domain, a stronger assumption is required on the approximation factor. In this case, the conditional lower bound holds for $c = 1 - o(1)$ and hence it does not rule out truly subquadratic algorithms for constant approximation. We believe that a different approach is required to show the hardness of IPS for constant approximation: indeed, the proposed reduction from OVP to IPS in the $\{0, 1\}^d$ domain strongly relies on the ability to distinguish inner products smaller than $k - 1$ and larger than k for some $k = \omega(1)$, implying $c \geq 1 - o(1)$. From an upper bound point of view, the LSH proposed in section 5.4.1 improves upon the state of the art [168] based on minwise hashing for different values (e.g., when $s \geq d/3$ and $c \geq 0.83$); however, it still gives weak subquadratic algorithm for signed/unsigned IPS with constant c. On the other hand, if $c = 1/n^{\Omega(1)}$, then the data structure based on linear search in section 5.4.3 reaches truly subquadratic time. An interesting open question is therefore to assess if strongly subquadratic time is possible when the approximation is constant in the $\{0, 1\}^d$ domain.

Chapter 6

High Probability Tensor Sketch

Joint work with: Jakob Bæk Tejs Knudsen

6.1 Introduction

The polynomial method has recently found many great applications in algorithm design, such as finding orthogonal vectors [14] and gap amplification in nearest neighbour [179]. Consider the polynomial $P(x,y) = \sum_{i<j<k}(x_i + y_i - x_iy_i)(x_j + y_j - x_jy_j)(x_k + y_k - x_ky_k)$ which counts the number of triangles in the union between two graphs, $x, y \in \{0,1\}^{\binom{n}{2}}$, expressed as binary vectors over the edges. Splitting P into monomials, one may construct functions $f, g : \{0,1\}^{\binom{n}{2}} \to \mathbb{R}^m$ such that $P(x,y) = \langle f(x), g(y)\rangle$, and use these embeddings to solve the 'most triangles in union' problem on a database of graphs, using an off the shelf nearest neighbours algorithm. (See our Applications for more on this.)

Other examples are kernel functions in statistics, such as $P(x,y) = \exp(-\|x - y\|_2^2) = \sum_{k\ge 0}(-1)^k\|x-y\|^{2k}/k!$, the Gaussian Radial Basis Function. The celebrated 'kernel trick' has been used in linear methods such as kernel PCA kernel nearest neighbour or kernel regression to allow detection of nonlinear dependencies between data without explicitly constructing feature vectors in high dimensional spaces. However the 'trick' requires the computation of the inner product between all pairs of points, while explicit embeddings scale linearly in the number of points, and have thus recently experienced a comeback.

While these polynomial expansions often produce prohibitively large vectors, they can often be reduced by some means, such as the Johnson-Lindenstrauss transform [102]. This in turn creates a strange phenomenon where we first blow up the dimension only to later squash it back down. It is tempting to look for shortcut to go straight to the final dimension.

This was the idea of Pagh and Pham [147, 155] with Tensor Sketch. The observation was that $P(x,y) = \langle x,y\rangle^2 = \langle x \otimes x, y \otimes y\rangle$, where $x \otimes x$ is the tensor product (Kronecker) of x with itself. They further observed that if C and C' are independent Count Sketch matrices, then $\langle Cx * C'x, Cy * C'y\rangle \approx \langle x,y\rangle^2$ while the dimension of $Cx * C'x$ is much smaller than of $x \otimes x$. Since the convolution $(\cdot * \cdot)$ can be computed in near linear time using the fast Fourier transformation, they could sketch $x \otimes x$ in basically the time required to sketch x. For a higher order polynomial kernel, replacing

$f(x) = x^{\otimes c}$ with $f(x) = C^{(1)}x * \cdots * C^{(c)}x$ thus takes the sketching time from d^c to cd. A huge improvement!

In this paper we improve on the main shortcomings of Tensor Sketch: To preserve the norm of vectors up to $1 \pm \epsilon$ with probability $1 - \delta$, it requires embedding into dimension roughly $3^c\epsilon^{-2}\delta^{-1}$. The exponential dependency on c greatly limits the degree of polynomials that can be embedded, and the linear dependency on δ^{-1} means we can't use a standard union bound trick to get e.g. a near neighbour preserving embedding [97], as could be achieved with the Johnson Lindenstrauss transform, which embeds into only $\epsilon^{-2}\log 1/\delta$ dimensions. We overcome both of these obstacles, by analyzing a scheme, that with the same embedding time, requires only $c^2\epsilon^{-2}(\log 1/\delta)^3$ dimensions.

A hugely important idea was introduced by Avron et al. [37]: They proved that a Tensor Sketch with sufficiently many rows is a Subspace embedding. This allowed many application that previously were only applied heuristically, such as solving a regression problem $\arg\min_x \|Ax - y\|_2$ directly in the reduced space while guaranteeing correct results. Using the subspace embeddings, they obtained the fastest known algorithms for computing an computing an approximate kernel PCA and many other problems.

However, the weaknesses of Tensor Sketch remained: The exponential dependency on c meant that the method could only be applied with relatively low degree polynomials. In this paper we show that our High Probability Tensor Sketch is also a subspace embedding, solving this major roadblock. We also show a second version of our sketch, which improves upon [37] by allowing an embedding dimension linear in the subspace dimension, rather than quadratic. In many uses of the subspace method the embedding dimension becomes larger than the number of points, which means we can get a quadratic improvement on these applications.

Our approach is to analyze fast family of Johnson Lindenstrauss matrices M with the further property that $M(x \otimes y) = M'x \circ M''y$ where $\circ$ is the Hadamard (or element-wise) vector product. We also analyze the case where M' and M'' are fully independent Gaussian matrices, and show that we are within a single factor $\log 1/\delta$ in embedding dimension while supporting much faster matrix-vector multiplication. The direct application of this method, $M^{(1)}x^{(1)} \circ \cdots \circ M^{(c)}$, would result in an exponential dependency on c, but by instead combining vectors as $M^{(1)}x^{(1)} \circ M'^{(1)}(M^{(2)}x^{(2)} \circ M'^{(2)}(\ldots$ we show that this dependency can be reduced to c^2. See also the Technical Overview below.

6.1.1 Overview

Our main contribution is to answer the questions "Does Tensor Sketch work with high probability?" and "Does there exist subspace embeddings for higher order polynomial embeddings?" For both of those questions, the answer is yes!

Theorem 21 (Construction A). *There is a distribution $\mathcal{M}$ over matrices $M \in \mathbb{R}^{m \times d^c}$ where*

1. $|\|Mx\|_2 - \|x\|_2| \le \epsilon$ *with probability* $\ge 1 - \delta$ *for any* $x \in \mathbb{R}^{d^c}$.

2. *M can be applied to tensors* $x^{(1)} \otimes \cdots \otimes x^{(c)} \in \mathbb{R}^{d^c}$ *in time* $O(c\,(d\log d + m\log m))$.

3. *m can be taken to be $O(c^2\epsilon^{-2}(\log 1/\delta)(\log 1/\epsilon\delta)^2)$.*

4. *$\mathcal{M}$ can compute fast approximate matrix multiplication:* $\Pr[|\|A^TM^TMB - A^TB\|_F| > \epsilon\|A\|_F\|B\|_F] < \delta$.

5. *There is an $m = O(c^2\epsilon^{-2}\lambda^2(\log 1/\delta)(\log 1/\epsilon\delta)^2)$ such that $\mathcal{M}$ is an (ϵ, δ)-subspace embedding. (See definition 24.)*

The result matches, up to a single factor $\log 1/\delta$ the embedding dimension needed for fully independent Gaussian matrices M and M', for $\|Mx \circ M'y\|_2$ to approximate $\|x \otimes y\|_2$. (See Appendix, Theorem 25.) However, suffering slightly in the embedding time, we can go all the way down to one:

Theorem 22 (Construction B)**.** *There is a distribution $\mathcal{M}$ over matrices $M \in \mathbb{R}^{m \times d^c}$ where*

1. *$|\|Mx\|_2 - \|x\|_2| \le \epsilon$ with probability $\ge 1 - \delta$ for any $x \in \mathbb{R}^{d^c}$.*

2. *Matrices $M \sim \mathcal{M}$ can be applied to tensors $x^{(1)} \otimes \cdots \otimes x^{(c)} \in \mathbb{R}^{d^c}$ in time $O(c\, m \min(d, m))$.*

3. *m can be taken to be $O(c^2\epsilon^{-2} \log 1/\delta \log^2(c\epsilon^{-1} \log 1/\delta))$.*

4. *There is an $m = O(c^2\epsilon^{-2}(\lambda + \log 1/\delta) \log^2(c\epsilon^{-1}\lambda \log 1/\delta))$ such that $\mathcal{M}$ is an (ϵ, δ)-subspace embedding. (See definition 24.)*

While the first theorem requires an intricate analysis of the combination of two Fast-JL matrices, the second one follows nearly directly from our general recursive construction theorem below:

Theorem 23. *Let $c > 0$ be a positive integer, and $Q^{(1)} \in \mathbb{R}^{m \times d}$ and $Q^{(i)} \in \mathbb{R}^{m \times md}$ be independent random matrices for every $i \in [c] \setminus \{1\}$. Define $M^{(k)} = Q^{(k)}(M^{(k-1)} \otimes I_d) \in \mathbb{R}^{m \times d^k}$ for $k \in [c]$, where $M^{(0)} = 1 \in \mathbb{R}$. Let $t > 0$ be a positive integer, and let $k_i \in [c]$ for every $i \in [t]$. Then the matrix*

$$M = \bigoplus_{i \in [t]} M^{(k_i)} \in \mathbb{R}^{tm \times \Sigma_{i \in [t]} d^{k_i}}$$

has the following properties.

1. *Let $\varepsilon \in (0, 1)$ and $\delta > 0$. If $Q^{(i)}$ has $(\varepsilon/2c, \delta/c)$-JL property for every $i \in [c]$, then M has (ε, δ)-JL property.*

2. *If $Q^{(i)}x$ can be evaluated in time T, where $x \in \mathbb{R}^{md}$, for every $i \in [c] \setminus \{1\}$, and $Q^{(1)}y$ can be evaluated in time T', where $y \in \mathbb{R}^d$, then $M(\bigoplus_{i \in [t]} \bigotimes_{j \in [k_i]} x^{(i,j)})$ can be evaluated in time $O(T't + T\sum_{i \in [t]} k_i)$, where $x^{(i,j)} \in \mathbb{R}^d$ for every $i \in [t], j \in [k_i]$.*

Now the difference between Construction A and Construction B is simply which matrices $Q^{(1)}, \ldots, Q^{(c))}$ that are used as basis for the construction.

Paper Structure The paper is structured as follows: After the comparison to related work and preliminaries we give a Technical overview of the sketch. We find it is useful to have some established notation before this section.

The technical part is split in three: We first show Theorem 23. This gives a recursive construction, which can be applied to tensors using any of the shelf Johnson-Lindenstrauss matrix. Combined with the fastest analysis of Fast-JL [112] this gives our theorem 22.

We proceed to analyze a small change in the construction of Fast-JL matrices, which allow for very fast application to tensor products. Specifically we show that the random diagonal matrix can be replaced by the Kronecker product of two smaller diagonal matrices without losing the JL-property, if the number of rows is increased slightly. This gives our theorem 21.

Finally in the last section, we show some algorithmic applications of our constructions. For example how to use it to find the two graphs in a database whose union has the most triangles.

6.1.2 Related work

Work related to sketching of tensors and explicit kernel embeddings is found in fields ranging from pure mathematics to physics and machine learning. Hence we only try to compare ourselves with the four most common types we have found.

We focus particularly on the work on subspace embeddings [155, 37], since it is most directly comparable to ours. An extra entry in this category is [105], which is currently in review, and which we were made aware of while writing this paper. That work is in double blind review, but by the time of the final version of this paper, we should be able to cite it properly.

Subspace embeddings For most applications [37], the subspace dimension, λ, will be much larger than the input dimension, d, but smaller than the implicit dimension d^c. Hence the size of the sketch, m, will also be assumed to satisfy $d << m << d^c$ for the purposes of stating the results. We will hide constant factors, and $\log 1/\epsilon$, $\log d$, $\log m$, $\log c$, $\log \lambda$ factors.

Note that we can always go from m down to $\approx \epsilon^{-2}(\lambda + \log 1/\delta)$ by applying a fast-JL transformation after embedding. This works because the product of two subspace embeddings is also a subspace embedding, and because fast-JL is a subspace embedding by the net-argument (see lemma 6.2.3). The embedding dimensions in the table should thus mainly be seen as a space dependency, rather than the actual embedding dimension for applications.

Reference	Embedding dimension, m	Embedding time	Note
[155, 37]	$\tilde{O}(3^c\, d\, \lambda^2\, \delta^{-1}\, \epsilon^{-2})$	$\tilde{O}(c\,(d+m))$	
Theorem 21	$\tilde{O}(c^2\, \lambda^2\, (\log 1/\delta)^3\, \epsilon^{-2})$	$\tilde{O}(c\,(d+m))$	
Theorem 22	$\tilde{O}(c^2\, (\lambda + \log 1/\delta)\, \epsilon^{-2})$	$\tilde{O}(c\, d\, m)$	
[105], Theorem 1	$\tilde{O}(c\, \lambda^2\, \delta^{-1}\, \epsilon^{-2})$	$\tilde{O}(c\,(d+m))$	Independent work.
[105], Theorem 2	$\tilde{O}(c^6\, \lambda\, (\log 1/\delta)^5\, \epsilon^{-2})$	$\tilde{O}(c\,(d+m))$	Independent work

Some of the results, in particular [155, 37] and [105] Theorem 1 can be applied faster when the input is sparse. Our results, as well as [105], Theorem 2 can similarly be optimized for sparse inputs, by preprocessing vectors with an implementation of Sparse JL [67].

In comparison to the previous result [155, 37] we are clearly better with an exponential improvement in c as well as δ.

Compared to the new work of [105], all four bounds have some region of superiority. Their first bound of has the best dependency on c, but has an exponential dependency on $\log 1/\delta$. Their second bound has an only linear dependency on $d + \lambda$, but has large polynomial dependencies on c and $\log 1/\delta$.

Technically the methods of all five bounds are similar, but some details and much of the analysis differ. Our results as well as the results of [105] use recursive constructions to avoid exponential dependency on c, however the shape of the recursion differs. We show all of our results using the p-moment method, while [105] Theorem 1 and [155, 37] are shown using 2nd-moment analysis. This explains much of why their dependency on δ is worse.

Approximate Kernel Expansions A classic result by Rahimi and Rect [157] shows how to compute an embedding for any shift-invariant kernel function $k(\|x-y\|_2)$ in time $O(dm)$. In [119] this is improved to any kernel on the form $k(\langle x, y\rangle)$ and time $O((m+d)\log d)$. This is basically optimal in terms of time and space, however the method does not handle kernel functions that can't be specified as a function of the inner product, and it doesn't provide subspace embeddings. See also [134] for more approaches along the same line.

Tensor Sparsification There is also a literature of tensor sparsification based on sampling [140], however unless the vectors tensored are already very smooth (such as ±1 vectors), the sampling has to be weighted by the data. This means that these methods in aren't applicable in general to the types of problems we consider, where the tensor usually isn't known when the sketching function is sampled.

Hyper-plane rounding An alternative approach is to use hyper-plane rounding to get vectors on the form ±1. Let $\rho = \frac{\langle x,y\rangle}{\|x\|\|y\|}$, then we have $\langle \operatorname{sign}(Mx), \operatorname{sign}(My)\rangle = \sum_i \operatorname{sign}(M_i x)\operatorname{sign}(M_i y) = \sum_i X_i$, where X_i are independent Rademachers with $\mu/m = E[X_i] = 1 - \frac{2}{\pi}\arccos\rho = \frac{2}{\pi}\rho + O(\rho^3)$. By tail bounds then $\Pr[|\langle \operatorname{sign}(Mx), \operatorname{sign}(My)\rangle - \mu| > \epsilon\mu] \le 2\exp(-\min(\frac{\epsilon^2\mu^2}{2\sigma^2}, \frac{3\epsilon\mu}{2}))$. Taking $m = O(\rho^{-2}\epsilon^{-2}\log 1/\delta)$ then suffices with high probability. After this we can simply sample from the tensor product using simple sample bounds.

The sign-sketch was first brought into the field of data-analysis by [52] and [179] was the first, in our knowledge, to use it with tensoring. The main issue with this approach is that it isn't a linear sketch, which hinders some applications, like subspace embeddings. It also takes dm time to calculate Mx and My. In general we would like fast-matrix-multiplication type results.

6.2 Preliminaries

We will use the following notation

k, i, j	Indicies
c	Tensor order
d	Original dimension (Assumed to be a power of 2.)
d^c	Implicit dimension
m	Sketch dimension
λ	Subspace dimension
Λ	Subspace of R^{d^c}
M	Sketching matrix
$\mathcal{M}$	Distribution of sketching matrices

We say $f(x) \lesssim g(x)$ if $f(x) = O(g(x))$. For $p \geq 1$ and random variables $X \in R$, we write $\|X\|_p = (E|X|^p)^{1/p}$. Note that $\|X + Y\|_p \leq \|X\|_p + \|Y\|_p$ by the Minkowski Inequality.

Definition 18 (Direct sum)**.** *We define the direct sum of two vectors as*

$$x \oplus y = \begin{bmatrix} x \\ y \end{bmatrix},$$

and the direct sum between two matrices as

$$A \oplus B = \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}.$$

Definition 19 (Kronecker (tensor) product). *We define the tensor-product (or Kronecker) of two matrices as:*

$$A \otimes B = \begin{bmatrix} a_{1,1}B & \cdots & a_{1,n}B \\ \vdots & \ddots & \vdots \\ a_{m,1}B & \cdots & a_{m,n}B \end{bmatrix},$$

and in particular of two vectors: $x \otimes y = [x_1y_1, x_1y_2, \dots, x_ny_n]^T$. Taking the tensor-product of a vector with itself, we get the tensor-powers:

$$x^{\otimes k} = \underbrace{x \otimes \cdots \otimes x}_{k \text{ times}}.$$

The Kronecker product has the useful mixed-pruduct property when the sizes match up:

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$$

We note in particular the vector variants $(I \otimes B)(x \otimes y) = x \otimes By$ and $\langle x \otimes y, z \otimes t\rangle = \langle x, y\rangle\langle z, t\rangle$.

Definition 20 (Hadamard product). *Also sometimes known as the 'element-wise product':*

$$x \circ y = [x_1y_1, x_2y_2, \dots, x_ny_n]^T.$$

Taking the Hadamard product with itself gives the Hadamard-power:

$$x^{\circ k} = \underbrace{x \circ \cdots \circ x}_{k \text{ times}} = [x_1^k, x_2^k, \dots, x_n^k]^T.$$

Definitions

Definition 21 (JL-moment property). *We say a distribution over random matrices $M \in \mathbb{R}^{m \times d}$ has the (ϵ, δ)-JL-moment property, when*

$$\| \|Mx\|_2^2 - 1\|_p \le \epsilon\delta^{1/p}$$

for all $p > 1$ and $x \in \mathbb{R}^d$, $\|x\| = 1$.

Note that by Markov's inequality, the JL-moment-property implies $E\|Mx\|_2 = \|x\|_2$ and that taking $m = O(\epsilon^{-2} \log 1/\delta)$ suffices to have $\Pr[|\|Mx\|_2 - \|x\|_2| > \epsilon] < \delta$ for any $x \in \mathbb{R}^d$. (This is sometimes known as the Distributional-JL property.)

Definition 22 ((ϵ, δ)-Approximate Matrix Multiplication). *We say a distribution over random matrices $M \in \mathbb{R}^{k \times d}$ has the (ϵ, δ)-Approximate Matrix Multiplication property if for any matrices A, B with proper dimensions,*

$$\begin{aligned} &\| \|A^T M^T M B - A^T B\|_F \|_p \\ &\qquad \le \epsilon \delta^{1/p} \|A\|_F \|B\|_F. \end{aligned}$$

Lemma 6.2.1 (Shown in [188]). *Any distribution that has the (ϵ, δ)-JL-moment-property has the $(3\epsilon, \delta)$-Approximate Matrix Multiplication property.*

We note that the factor of 3 on ϵ can be removed by combining the analysis in [188] with Appendix Lemma 6.7.3.

Definition 23 (ϵ-Subspace embedding). *$M \in \mathbb{R}^{k \times D}$ is a subspace embedding for $\Lambda \subseteq \mathbb{R}^D$ if for any $x \in \Lambda$,*

$$|\|Mx\|_2 - \|x\|_2| \le \epsilon.$$

Definition 24 ((λ, ϵ)-Oblivious Subspace Embedding). *A distribution, $\mathcal{M}$, over $R^{m \times D}$ matrices is a (D, λ)-Oblivious Subspace Embedding if for any linear subspace, $\Lambda \subseteq \mathbb{R}^D$, of dimension λ, $M \sim \mathcal{M}$ is an ϵ-subspace embedding for Λ with probability at least $1 - \delta$.*

Lemma 6.2.2. *Any distribution that has the $(\epsilon/(3\lambda), \delta)$-JL-moment-property is a (λ, ϵ)-oblivious subspace embedding.*

Proof. Let $U \in \mathbb{R}^{\lambda \times m}$ be orthonormal such that $U^T U = I$, it then suffices (by [188]) to show $\|U^T M^T M U - I\| \le \epsilon$.

From lemma 6.2.1 we have that $\|U^T M^T M U - I\| \le 3\epsilon\delta^{1/p}\|U\|_F^2 = 3\epsilon\delta^{1/p}\lambda$. □

Lemma 6.2.3. *There is a $C > 0$, such that any distribution that has the $(\epsilon, \delta e^{C\lambda})$-JL-moment-property is a (λ, ϵ)-oblivious subspace embedding.*

Proof. For any λ-dimensional subspace, Λ, there exists an ϵ-net $T \subseteq \Lambda \cap S^{d-1}$ of size C^d such that if M preserves the norm of every $x \in T$ then M preserves all of Λ up to $1 + \epsilon$. See [188] for details. □

Lemma 6.2.4 (Khintchine's inequality [87]). *Let $p \ge 1$, $x \in R^d$, and $\sigma \mathbb{R}^d$ be independent Rademacher ± 1 random variables. Then*

$$\|\sum_{i=1}^{d} \sigma_i x_i\|_p \lesssim \sqrt{p}\|x\|_2.$$

6.3 Technical Overview

The main component of any tensor sketch is a matrix $M : \mathbb{R}^{m \times d_1 d_2}$ such that $\|Mx\|_2 \approx \|x\|_2$ and which an be applied efficiently (faster than md_1d_2) to simple tensors $x = x^{(1)} \otimes x^{(2)}$, where $x^{(1)} \in \mathbb{R}^{d_1}, x^{(2)} \in \mathbb{R}^{d_2}$.

If $x^{(1)}$ and $x^{(2)}$ are ±1 vectors, sampling from x works well and can be done without actually constructing x. For this reason a natural general sketch is $S(M^{(1)}x^{(1)} \otimes M^{(2)}x^{(2)})$, where $M^{(1)}$ and $M^{(2)}$ are random rotations.

The original Tensor Sketch did $\mathcal{F}^{-1}(\mathcal{F}C^{(1)}x \circ \mathcal{F}C^{(2)}x)$, where $C^{(1)}$ and $C^{(2)}$ are Count Sketch matrices. At first sight this may look somewhat different, but we can ignore the orthonormal $\mathcal{F}^{-1}$, and then we have $M^{(1)}x^{(1)} \circ M^{(2)}x^{(2)}$ which is just sampling the diagonal of $M^{(1)}x^{(1)} \otimes M^{(2)}x^{(2)}$. Since $M^{(1)}$ and $M^{(2)}$ are independent, sampling the diagonal works as well as any other subset of the same size.

Since Tensor Sketch only used 2nd moment analysis, the natural technical question is "how well does $M^{(1)}x^{(1)} \circ M^{(2)}x^{(2)}$ really work?" when $M^{(1)}$ and $M^{(2)}$

can be anything. In Theorem 25 we show that an embedding dimension of $m = \Theta(\epsilon^{-2} \log 1/\delta + \epsilon^{-1}(\log 1/\delta)^2$ is both sufficient and necessary for (sub)-gaussian matrices, which we conjecture is optimal across all distributions.

Sub-gaussian matrices however still take md time to evaluate, so our tensor sketch would still take $m(d_1 + d_2)$ time in total. We really want $M^{(1)}$ and $M^{(2)}$ to have fast matrix-vector multiplication. It is thus natural to analyze the above scheme where $M^{(1)}$ and $M^{(2)}$ are Fast Johnson Lindenstrauss matrices ala [13, 112]. We do this in Section 6.5 and show that $m = \epsilon^{-2}(\log 1/\delta)(\log 1/\epsilon\delta)^2$ suffices. For ϵ not too small, this matches our suggested optimum by one $\log 1/\delta$ factor.

The final challenge is to scale up to larger tensors than order 2. Our Lemma 6.5.1 shows and exponential dependency: $m = \epsilon^{-2}(\log 1/\delta)(\log 1/\epsilon\delta)^c$, which would be rather unfortunate. Luckily it turns out, that by continuously 'squashing' the dimension back down to $\epsilon^{-2}(\log 1/\delta)(\log 1/\epsilon\delta)^2$, we can avoid this explosion.

While we usually think of applying our sketching matrix to simple tensors, we always analyze everything assuming the input has full rank. This adds some extra difficulty to the analysis, but it is worth it, since by showing that our matrix has the so called JL-moment-property, we get that it is also a subspace embedding for free, by Lemma 6.2.2 and Lemma 6.2.3.

6.4 The High Probability Tensor Sketch

In this section we will prove Theorem 23 which is the backbone of our theorems. Theorem 22 will follow as an easy corollary, while Theorem 21 is completed in the next section.

Before we show the full theorem we will consider a slightly easier construction. Given independent random matrices $Q^{(2)}, \dots, Q^{(c)} \in \mathbb{R}^{m \times dm}$, from a distribution to be discussed later, and $Q^{(1)} \in \mathbb{R}^{m \times d}$, we define $M^{(0)} = 1 \in \mathbb{R}$ and recursively $M^{(k)} = Q^{(k)}(M^{(k-1)} \otimes I_d)$ for $k \in [c]$. The goal of this section is to show that M^k has JL- and related properties when the $Q^{(i)}$s have, and that $M^{(k)}$ can be evaluated efficiently on simple tensors, $x^{(1)} \otimes \dots \otimes x^{(k)} \in \mathbb{R}^{d^k}$, for $k \in [c]$.

First we show a rather simple fact which will prove to be quite powerful.

Lemma 6.4.1. *Let $\varepsilon \in (0,1)$ and $\delta > 0$. If $P \in \mathbb{R}^{m_1 \times d_1}$ and $Q \in \mathbb{R}^{m_2 \times d_2}$ are two matrices with (ϵ, δ)-JL moment property, then $P \oplus Q \in \mathbb{R}^{(m_1+m_2)\times(d_1+d_2)}$ has (ϵ, δ)-JL moment property.*

Proof. Let $x \in \mathbb{R}^{d_1+d_2}$ and choose $y \in \mathbb{R}^{d_1}$ and $z \in \mathbb{R}^{d_2}$ such that $x = y \oplus z$. Now using the triangle inequality and JL moment property, we get that

$$\begin{aligned}\left\| \|(P \oplus Q)x\|_2^2 - \|x\|_2^2 \right\|_p &\le \left\| \|Py\|_2^2 - \|y\|_2^2 \right\|_p + \left\| \|Qz\|_2^2 - \|z\|_2^2 \right\|_p \\ &\le \varepsilon\delta^{1/p} \|y\|_2^2 + \varepsilon\delta^{1/p} \|z\|_2^2 \\ &= \varepsilon\delta^{1/p} \|x\|_2^2,\end{aligned}$$

since $\|y\|^2 + \|z\|_2 = \|y \oplus z\|_2$ by disjointness. □

An easy consequence of this lemma is that for any matrix T, $I_\ell \otimes T$ has (ε, δ)-JL moment property when T has (ε, δ)-JL moment property, since $I_\ell \otimes Q = \underbrace{Q \oplus Q \oplus \ldots \oplus Q}_{\ell \text{ times}}$.

Similarly, $Q \otimes I_\ell$ has (ε, δ)-JL moment property, since you can obtain $Q \otimes I_\ell$ by reordering the rows of $I_\ell \otimes Q$, which trivially does not change the JL moment property.

It is now easy to show that $M^{(k)}$ has JL-property when $Q^{(1)}, \ldots, Q^{(k)}$ has JL-property for $k \in [c]$.

Lemma 6.4.2. *Let $\varepsilon \in (0,1)$ and $\delta > 0$. If $Q^{(1)}, \ldots, Q^{(c)}$ has the $(\varepsilon/2c, \delta/c)$-JL property, then $M^{(k)}$ has $(k/c\varepsilon, k/c\delta)$-jl property for every $k \in [c]$.*

Proof. Let $k \in [c]$ be fixed. We note that an alternative way of expressing $M^{(k)}$ is as follows:

$$M^{(k)} = Q^{(k)}(Q^{(k-1)} \otimes I_d)(Q^{(k-2)} \otimes I_{d^2}) \ldots (Q^{(1)} \otimes I_{d^{k-1}})$$

Let $x \in \mathbb{R}^{d^k}$ be any vector. Define $x^{(i)} = (Q^{(i)} \otimes I_{d^{k-i}})x^{(i-1)}$ for $i \in [k]$ and $x^{(0)} = x$. Since $Q^{(i)}$ has $(\varepsilon/2k, \delta/k)$-JL property then $Q^{(i)} \otimes I_{d^i}$ has $(\varepsilon/2c, \delta/c)$-JL property by the previous discussion, hence $\Pr\left[\left|\|x^{(i)}\|_2^2 - \|x^{(i)}\|_2^2\right| \geq \varepsilon/2c\|x^{(i)}\|_2^2\right] \leq \delta/c$. Now a simple union bound give us that

$$1 - k/c\varepsilon \leq (1 - \varepsilon/2c)^k \leq \left|\|Mx\|_2^2 - \|x\|_2^2\right| \leq (1 + \varepsilon/2c)^k \leq 1 + k/c\varepsilon$$

with probability at least $1 - k/c\delta$, which finishes the proof. □

Corollary 6. *Let $\varepsilon \in (0,1)$. If $Q^{(1)}, \ldots, Q^{(c)}$ has the $(\epsilon/(2\lambda c), \delta/c)$-JL property, then $M^{(k)}$ is a λ-subspace embedding.*

Proof. This follows from lemma 6.2.2. □

Note that if $Q^{(i)}x$, where $x \in \mathbb{R}^{dk}$, can be evaluated in time T for every $i \in [c] \setminus \{1\}$, and $Q^{(1)}y$, where $y \in \mathbb{R}^d$, also can be evaluated in time T', then $M^{(k)}z$, where $z \in \mathbb{R}^{d^k}$, can be evaluated in time $T'd^{k-1} + T\sum_{i=0}^{k-2} d^i = T'd^{k-1} + T(d^{k-1} - 1)/(d-1) = \Theta(T'd^{k-1} + Td^{k-2})$. Meanwhile, if $x \in \mathbb{R}^{d^k}$ is on the form $x^{(1)} \otimes \ldots \otimes x^{(k)}$, we have $M^{(k)}x = Q^{(k)}(M^{(k-1)}(x^{(1)} \otimes \ldots x^{(k-1)}) \otimes x^{(k)})$. Now an easy induction argument shows that this allows evaluation in time $O(T' + Tk)$, which is exponentially faster.

Using this construction it now becomes easy to sketch polynomials. More precisely, let $t \in \mathbb{Z}_{>0}$, $k_i \in [c]$ for every $i \in [t]$, then the matrix $M = \bigoplus_{i \in [t]} M^{(k_i)}$ has (ε, δ)-JL property and can be evaluated at the vector $x = \bigoplus_{i \in [t]} \bigotimes_{j \in [k_i]} x^{(i,j)}$ in time $O(T't + T\sum_{i \in [t]} k_i)$, where $x^{(i,j)} \in \mathbb{R}^d$ for every $i \in [t], j \in [k_i]$.

This discussion proves Theorem 23. Note that if we apply a Fast Johnson Lindenstrauss Transform between every direct sum we can obtain an output dimension of $O(m)$.

Example 1. *Often it is possible to get an even faster evaluation time if the input has even more structure. For example consider the matrix $M = \bigoplus_{i \in [c]} M^{(i)}$ and the vector $z = \bigoplus_{i \in [c]} \bigotimes_{j \in [i]} x^{(j)}$, where $x^{(j)} \in \mathbb{R}^d$ for every $j \in [c]$. Then Mz can be evaluated in time $O(T' + cT)$ by exploiting the fact that*

$$M^{(k)}(\bigotimes_{j \in [k]} x^{(j)}) = Q^{(k)}(M^{(k-1)}(\bigotimes_{j \in [k-1]} x^{(j)}) \otimes x^{(k)}) \ ,$$

so we can use the previous calculations.

As promised we now get the proof of Theorem 22 by choosing $Q^{(1)}, \dots, Q^{(c)}$ to be Fast Johnson Lindenstrauss Matrices. Using the analysis from Krahmer et al. [112] they can be evaluated in time $O(md \log md)$ and if we set $m = \tilde{O}(c^2 \log(1/\delta)/\varepsilon^2)$ then $Q^{(1)}, \dots, Q^{(c)}$ has $(\varepsilon/2c, \delta/c)$-JL property. Now Theorem 23 give us the result.

6.5 Fast Constructions

The purpose of this section is to show the following lemma:

Lemma 6.5.1. *Let $c \in \mathbb{Z}_{>0}$, and $(D^{(i)})_{i\in[c]} \in \prod_{i\in[c]} \mathbb{R}^{d_i \times d_i}$ be independent diagonal matrices with independent Rademacher variables. Define $d = \prod_{i\in[c]} d_i$ and $D = \bigotimes_{i\in[c]} D_i \in \mathbb{R}^{d\times d}$. Let $S \in \mathbb{R}^{m\times d}$ be an independent sampling matrix which samples exactly one coordinate per row. Let $x \in \mathbb{R}^d$ be any vector and $p \geq 1$, then*

$$\left\| \tfrac{1}{m} \|SHDx\|_2^2 - \|x\|_2^2 \right\|_p \lesssim \sqrt{p}\,(p + \log m)^{c/2} \|x\|_2^2 / \sqrt{m} + p\,(p + \log m)^c \|x\|_2^2 / m.$$

Setting $m = O(\epsilon^{-2} \log 1/\delta (\log 1/\epsilon\delta)^c)$ thus suffices for SHD to have the (ϵ, δ)-JL-moment-property. This then gives (by lemma 6.2.1 and 6.2.2) that SHD is a subspace embedding.

We note that SHD can be applied efficiently to simple tensors by the relation:

$$\begin{aligned} SH_{d_1 d_2}(D^{(1)} \otimes D^{(2)})(x \otimes y) &= (S^{(1)} \otimes S^{(2)})(H_{d_1} \otimes H_{d_2})(D^{(1)} \otimes D^{(2)})(x \otimes y) \\ &= S^{(1)} H_{d_1} D^{(1)} x \circ S^{(2)} H_{d_2} D^{(2)} y, \end{aligned}$$

where H_n is the size n Hadamard matrix and $S^{(1)}$ and $S^{(2)}$ are independent sampling matrices. Combining this fact with the construction in the previous section gives Theorem 21.

The rest of this section is devoted to proving Lemma 6.5.1. We first show two technical lemmas, which seem like they could be useful for many other things.

Lemma 6.5.2. *Let $p \geq 1$, $c \in Z_{>0}$, and $(\sigma^{(i)})_{i\in[c]} \in \prod_{i\in[c]} \mathbb{R}^{d_i}$ be independent Rademacher vectors. Let $a_{i_0,\dots,i_{c-1}} \in \mathbb{R}$ for every $i_j \in [d_j]$ and every $j \in [c]$, then*

$$\Big\| \sum_{i_1\in[d_1],\dots,i_c\in[d_c]} \prod_{j\in[c]} \sigma_{i_j}^{(j)} a_{i_0,\dots,i_{c-1}} \Big\|_p \lesssim p^{c/2} \Big(\sum_{i_1\in[d_1],\dots,i_c\in[d_c]} a_{i_0,\dots,i_{c-1}}^2 \Big)^{1/2} = p^{c/2} \|a\|_{HS}.$$

Proof. The proof will be by induction on c. For $c = 1$ then the result is just Khintchine's inequality (Lemma 6.2.4). So assume that the result is true for every value up to c.

Using the induction hypothesis we get that

$$\begin{aligned}
\Big\| \sum_{\substack{i_1\in[d_1],\\ \dots,i_c\in[d_c]}} \prod_{j\in[c]} \sigma^{(j)}_{i_j} a_{i_1,\dots,i_c} \Big\|_p &= \Big\| \sum_{\substack{i_1\in[d_1],\\ \dots,i_{c-1}\in[d_{c-1}]}} \prod_{j\in[c-1]} \sigma^{(j)}_{i_j} \Big(\sum_{i_c\in[d_c]} \sigma^{(c)}_{i_c} a_{i_1,\dots,i_c} \Big) \Big\|_p \\
&\lesssim p^{(c-1)/2} \Big\| \Big(\sum_{\substack{i_1\in[d_1],\\ \dots,i_{c-1}\in[d_{c-1}]}} \Big(\sum_{i_c\in[d_c]} \sigma^{(c)}_{i_c} a_{i_1,\dots,i_c} \Big)^2 \Big)^{1/2} \Big\|_p && \text{(I.H.)} \\
&= p^{(c-1)/2} \Big\| \sum_{\substack{i_1\in[d_1],\\ \dots,i_{c-1}\in[d_{c-1}]}} \Big(\sum_{i_c\in[d_c]} \sigma^{(c)}_{i_c} a_{i_1,\dots,i_c} \Big)^2 \Big\|_{p/2}^{1/2} \\
&\le p^{(c-1)/2} \Big(\sum_{\substack{i_1\in[d_1],\\ \dots,i_{c-1}\in[d_{c-1}]}} \Big\| \Big(\sum_{i_c\in[d_c]} \sigma^{(c)}_{i_c} a_{i_1,\dots,i_c} \Big)^2 \Big\|_{p/2} \Big)^{1/2} && \text{(Triangle)} \\
&= p^{(c-1)/2} \Big(\sum_{\substack{i_1\in[d_1],\\ \dots,i_{c-1}\in[d_{c-1}]}} \Big\| \sum_{i_c\in[d_c]} \sigma^{(c)}_{i_c} a_{i_1,\dots,i_c} \Big\|_p^2 \Big)^{1/2} \\
&\lesssim p^{c/2} \Big(\sum_{i_1\in[d_1],\dots,i_c\in[d_c]} a^2_{i_1,\dots,i_c} \Big)^{1/2} && \text{(Khintchine)}
\end{aligned}$$

where the last inequality is by using Khintchine's inequality. Plugging this into the previous inequality finishes the induction step and hence the proof. □

The next lemma we nee is a type of Chernoff bound for pth moments.

Lemma 6.5.3. *Let $p \ge 2$ and $X_0, \dots, X_{k-1}$ be independent non-negative random variables with p-moment, then*

$$\Big\| \sum_{i\in[k]} (X_i - E[X_i]) \Big\|_p \lesssim \sqrt{p} \sqrt{\sum_{i\in[k]} E[X_i]} \Big\| \max_{i\in[k]} X_i \Big\|_p^{1/2} + p \Big\| \max_{i\in[k]} X_i \Big\|_p$$

Proof.

$$\begin{aligned}
\Big\| \sum_{i\in[k]} (X_i - \mathrm{E}[X_i]) \Big\|_p &\lesssim \Big\| \sum_{i\in[k]} \sigma_i X_i \Big\|_p && \text{(Symmetrization)} \\
&\lesssim \sqrt{p} \Big\| \sqrt{\sum_{i\in[k]} X_i^2} \Big\|_p && \text{(Khintchine's inequality)} \\
&= \sqrt{p} \Big\| \sum_{i\in[k]} X_i^2 \Big\|_{p/2}^{1/2} \\
&\le \sqrt{p} \Big\| \max X_i \Big\|_p^{1/2} \Big\| \sum_{i\in[k]} X_i \Big\|_p^{1/2} && \text{(Hölder's inequality)} \\
&\le \sqrt{p} \Big\| \max X_i \Big\|_p^{1/2} \sqrt{\sum_{i\in[k]} \mathrm{E}[X_i]} \\
&\quad + \sqrt{p} \Big\| \max X_i \Big\|_p^{1/2} \Big\| \sum_{i\in[k]} (X_i - \mathrm{E}[X_i]) \Big\|_p^{1/2} && \text{(Triangle inequality)}
\end{aligned}$$

Now let $C = \| \sum_{i\in[k]}(X_i - \mathrm{E}[X_i]) \|_p^{1/2}$, $B = \sqrt{\sum_{i\in[k]} \mathrm{E}[X_i]}$, and $A = \sqrt{p}\| \max X_i \|_p^{1/2}$. then we have shown $C^2 \leq A(B+C)$. That implies C is smaller than the largest of the roots of the quadratic. Solving this quadratic inequality gives $C^2 \lesssim AB + A^2$ which is the result. □

We can finally go ahead and prove Lemma 6.5.1.

Proof. For every $i \in [m]$ we let S_i be the random variable that says which coordinate the i'th row of S samples, and we define the random variable $Z_i = M_i x_i = H_{S_i} D x_i$. We note that since the variables $(S_i)_{i\in[m]}$ are independent then the variables $(Z_i)_{i\in[m]}$ are conditionally independent given D, that is, if we fix D then $(Z_i)_{i\in[m]}$ are independent.

Using Lemma 6.5.3 we get that

$$\| \frac{1}{m} \sum_{i\in[m]} Z_i^2 - \|x\|_2^2 \|_p \lesssim \sqrt{p}\, \big(\frac{1}{m} \sum_{i\in[m]} \mathrm{E}\Big[Z_i^2 \;\Big|\; D\Big] \,\big)^{1/2} \| \max_{i\in[m]} \frac{1}{m} Z_i^2 \|_p^{1/2} + p \, \| \max_{i\in[m]} \frac{1}{m} Z_i^2 \|_p \tag{6.1}$$

It follows easily that $\mathrm{E}[Z_i^2 \mid D] = \|x\|_2^2$ from the fact that $\|HDx\|_2^2 = d\|x\|_2^2$, hence $\big(\frac{1}{m}\sum_{i\in[m]} \mathrm{E}[Z_i^2 \mid D]\,\big)^{1/2} = \|x\|_2$. Now we just need to bound $\| \max_{i\in[m]} \frac{1}{m} Z_i^2 \|_p = \frac{1}{m}\| \max_{i\in[m]} Z_i^2 \|_p$. First we note that

$$\|Z_i^2\|_p = \|(H_{S_i} D x_i)^2\|_p \lesssim p^c \|x\|_2^2$$

by Khintchine's inequality. Let $q = \max\{p, \log m\}$, then we get that

$$\| \max_{i\in[m]} Z_i^2 \|_p \leq \| \max_{i\in[m]} Z_i^2 \|_q \leq \big(\sum_{i\in[m]} \|Z_i^2\|_q^q \big)^{1/q} \leq m^{1/q} q^c \|x\|_2^2$$

Now since $q \geq \log m$ then $m^{1/q} \leq 2$ so $\| \max_{i\in[m]} Z_i^2 \|_p \lesssim q^c \|x\|_2^2 \leq (p + \log m)^c \|x\|_2^2$. Plugging this into 6.1 finishes the proof. □

6.6 Applications

The classic application of Tensor Sketching is compact bilinear pooling (or multilinear). This simply corresponds to expanding x ans $x^{\otimes 2}$ (bilinear pooling) and then hashing back to a smaller size (compact). First discussed in [83] which showed how to do back-propagration through a tensor-sketch layer. Then applied to all many applications such as visual convolutional models [122], question answering [82], visual reasoning [101], video classification [130].

These results are usually given without any particular guarantees, but we can also use polynomial embeddings for concrete algorithms using the following lemma:

6.6.1 Sketching Polynomials

Theorem 24. *Given any degree c polynomial, $P(z) = \sum_{i=0}^{c} a_i z^i$, there are a pair of embeddings $f, g : \mathbb{R}^d \to \mathbb{R}^m$, such that for any $x, y \in \mathbb{R}^d$, the inner product*

$$\langle f(x), g(y)\rangle = (1 \pm \epsilon) P(\langle x, y\rangle)$$

with probability at least $1 - \delta$. Using Construction A we set $m = O(c^2\varepsilon^{-2}(\log 1/\delta)(\log 1/\varepsilon\delta)^2)$, and $f(x)$ and $g(y)$ can be computed in $O(c(d\log d + m\log m))$ time. Using Construction B we set $m = \tilde{O}(c^2\varepsilon^{-2}(\log 1/\delta))$, and $f(x)$ and $g(y)$ can be computed in $O(cm\min(d,m))$ time.

Proof. We note that

$$P(\langle x,\ y\rangle) = \sum_{i=0}^{c} a_i \langle x,\ y\rangle^i = \sum_{i=0}^{c} \left\langle a_i x^{\otimes i},\ y^{\otimes i}\right\rangle = \left\langle \bigoplus_{i=0}^{c} a_i x^{\otimes i},\ \bigoplus_{i=0}^{c} y^{\otimes i}\right\rangle.$$

So using Theorem 23 together with Construction A and Construction B give the result. □

We note that the output dimension m in the theorem can be improved by applying a Fast Johnson Lindenstrauss Transform in the end.

Example 2 (Explicit Gaussian Kernel). *Say we want $\langle f(x), g(y)\rangle \approx \exp(-\langle x,y\rangle^2) = \sum_{k=0}^{c}(-1)^k\langle x,y\rangle^{2k}/k! + O(\langle x,y\rangle^{2c+1}/c!)$. Then using Theorem 24 we can obtain $\langle f(x), g(y)\rangle = (1 \pm \varepsilon)\sum_{k=0}^{c}(-1)^k\langle x,y\rangle^{2k}/k!$, hence get an approximation of $\varepsilon + O(\langle x,y\rangle^{2c+1}/c!)$ with probability $1-\delta$, using $O(c(d\log d + m\log m))$ time where $m = O(c^3\varepsilon^{-2}(\log 1/\delta)(\log 1/\varepsilon\delta)^2)$.*

6.6.2 Embeddings

Lemma 6.6.1 (Symmetric Polynomials). *Given any degree polynomial, $P(x_1, \ldots, x_d, y_1, \ldots, y_d)$ with k monomials we can make an embedding $f, g : R^d \to R^m$ such that*

$$E\langle f(x), g(y)\rangle = \sum_{\pi} P(x_{\pi(1)}, \ldots, x_{\pi(d)}, y_{\pi(1)}, \ldots, y_{\pi(d)}).$$

$f(x)$ and $g(y)$ can be computed in time $k\,4^{\Delta}$ sketching operations, where Δ is the maximum combined degree of a monomial. (E.g. 4 for $x_1^2 y_1 y_2$.)

For x and y boolean, we get $\|f(x)\|_2^2 \le k(\|x\|_2^2 + \kappa - 1)!/(\|x\|_2^2 - 1)! \le k(\|x\|_2^2 + \kappa - 1)^{\kappa}$. where κ is the numer of different indicies in a monomial. E.g. for $x_1 y_1 x_2$ it is 2.

Proof. See the Appendix Section 6.7.2. □

Example 3 (Triangle counting). *Say you have a database of graphs, $\mathcal{G}$, seen as binary vectors in $\{0,1\}^e$. Given a query graph, G, you want to find $G' \in \mathcal{G}$ such that the number of triangles in $G \cup G'$ is maximized.*

We construct the polynomial $P(x,y) = (x_1 + y_1 - x_1y_1)(x_2 + y_2 - x_2y_2)(x_3 + y_3 - x_3y_3)$ which is 1 exactly when $x \cup u$ has a triangle on edges 1, 2, 3. The maximum number of *different indicies is $\kappa = 3$. The maximum number of triangles (with ordering) is $6\binom{d,3}{\approx}d^3$. We have $\|f(x)\|_2\|g(x)\| \approx (d+3)^3$, so to get precision within 1% of the maximum number, we need to set $\epsilon < 0.01d^3/(d+3)^3$.*

Since our approximation works with high probability, we can take a union bound and plug the embedded vectors into the standard data structure of [25] or others.

6.6.3 Oblivious Subspace Embeddings

In [188] the authors show a number of applications of polynomial kernels in oblivious subspace embeddings. They also show that the original tensor sketch [155] is an oblivious subspace embedding when the sketch matches the size described in the introduction. It is shown how each of:

1. Approximate Kernel PCA and Low Rank Approximation,

2. Regularizing Learning With the Polynomial Kernel,

3. Approximate Kernel Principal Component Regression,

4. Approximate Kernel Canonical Correlation Analysis,

can be computed with state of the art performance.

However, each of the applications encounter an exponential dependency on c. They also inherit the tensor-sketch linear dependency on the inverse error probability. Our sketch improves each of these aspects directly black box.

6.7 Appendix

6.7.1 Subgaussian construction

Before stating the theorem, we not the following matrix product:

Definition 25 (Face-splitting product)**.** *Defined between to matrices as the Kronecker-product between pairs of rows:*

$$C \bullet D = \begin{bmatrix} C_1 \otimes D_1 \\ C_2 \otimes D_2 \\ \dots \\ C_n \otimes D_n \end{bmatrix}.$$

Face-splitting product has the relation $(A \bullet B)(x \otimes y) = Ax \circ By$.

Theorem 25 (Subgaussian)**.** *Let $T, M \in \mathbb{R}^{m\times d}$ have iid. sub-gaussian entries, then $\|\|\frac{1}{\sqrt{m}}(T \bullet M)x\|_2^2 - \|x\|_2^2\|_p \leq \sqrt{p/m} + pq/m$, where $q = \max(p, m)$.*

This immediately implies that for $m = \Omega(\epsilon^{-2}\log 1/\delta + \epsilon^{-1}(\log 1/\delta)(\log 1/\epsilon\delta))$, $T \bullet M$ has the JL-moment property.

We note that the analysis is basically optimal. Assume M and T were iid. Gaussian matrices and $x = e_1'^{\otimes 2}$ were a simple tensor with a single 1 entry. Then $|\|(M \bullet T)x\|_2^2 - \|x\|_2^2| = |\|Mx' \circ Tx'\|_2^2 - 1| \sim |(gg')^2 - 1|$ for $g, g' \in R$ iid. Gaussians. Now $\Pr[(gg')^2 > (1+\epsilon)] \approx \exp(-\min(\epsilon, \sqrt{\epsilon}))$, thus requiring $m = \Omega(\epsilon^{-2}\log 1/\delta + \epsilon^{-1}(\log 1/\delta)^2)$ matching our bound up to a $\log 1/\epsilon$.

Proof. Let $Q = T \bullet M \in \mathbb{R}^{n \times ab}$.

$$\begin{aligned}
\|(T \bullet M)x\|_2^2 &= \sum_k ((T \bullet M)_k x)^2 \\
&= \sum_k (TU_{(i)})_{k,k}^2
\end{aligned}$$

where $U_{(i)} = (XM^T)_{(i)} = (M_iX)^T$, when x is seen as a $d \times d$ matrix.

We then have sub-gaussians:

$$\begin{aligned}
E\|U_{(i)}\|_2^2 = E\|M_iX\|_2^2 = \sum_k E(M_iX_{(k)})^2 = \sum_k \|X_{(k)}\|^2 = \|X\|_F^2 = 1 \\
\|\|U_{(i)}\|_2^2\|_p \le \|\|M_iX^T\|_2^2 - 1\| + 1 \le \sqrt{p}\|X^TX\|_F + p\|X^TX\| + 1 \le p \\
\|\sum_k \|U_{(k)}\|_2^4\|_p \le \sqrt{pk} + k + pq^2,
\end{aligned}$$

The last bound followed from independence and bounded variance of the $\|U_{(k)}\|$s. It is possible to go without this assumption though, suffering a small loss in the final dimension.

We bound

$$\|(T \bullet M)x\|_2^2/k - \|x\|_2^2\|_p \le \|\sum_k (T_kU_{(k)})^2 - \|U\|_F^2\|_p/k \tag{6.2}$$

$$+ \|\|U\|_F^2/k - \|x\|_2^2\|_p. \tag{6.3}$$

Bounding eq. (6.3) follows simply from the JL property of M. Bounding eq. (6.2) is a bit trickier, and we use the Hanson-Wright inequality:

$$\begin{aligned}
\|\sum_k (T_kU_{(k)})^2 - \|U\|_F^2\|_p &\le \big\|\sqrt{p}\big(\sum_k \|U_{(k)}U_{(k)}^T\|_F^2\big)^{1/2} + p\max_k \|U_{(k)}U_{(k)}^T\|\big\|_p \\
&\le \big\|\sqrt{p}\big(\sum_k \|U_{(k)}\|_2^4\big)^{1/2} + p\max_k \|U_{(k)}\|_2^2\big\|_p \\
&\le \sqrt{p}\|\sum_k \|U_{(k)}\|_2^4\|_{p/2}^{1/2} + pq. \qquad (6.4)
\end{aligned}$$

Here we used the maximum trick from the next section to bound the max term.

A short aside: It would be sweet to split

$$\begin{aligned}
\|\sum_k \|U_{(k)}\|_2^4\|_{p/2} &\le \|\max \|U_{(k)}\|_2^2 \sum \|U_{(k)}\|_2^2\|_{p/2} \\
&\le \|\max \|U_{(k)}\|_2^2\|_p \, \|\sum \|U_{(k)}\|_2^2\|_p \\
&\le p\,\|\|U\|_F^2\|_p,
\end{aligned}$$

but unfortunately the second factor is[1] $\sqrt{pk} + p$, which means we end up with $p(pk)^{1/4}$ term in eq. (6.4), which is too much. ($p^{3/4}k^{1/4}$ would have been tolerable.) We'll show how to shave this factor p on the $\sqrt{pk}$ term.

We instead have to work directly on the fourth powers. We use triangle and Bernstein

$$\begin{aligned}\|\sum_k \|U_{(k)}\|_2^4\|_{p/2} &\le \Big\|\sum_k \|U_{(k)}\|_2^4 - E\|U_{(k)}\|_2^4\Big\|_{p/2} + \sum_k E\|U_{(k)}\|_2^4 \\ &\le \sqrt{p}\Big(\sum_k (E\|U_{(k)}\|_2^4)^2\Big)^{1/2} + p\|\max\|U_{(k)}\|_2^4\|_{p/2} + \sum_k \|\|U_{(k)}\|_2^2\|_2^2 \\ &\le \sqrt{p}\Big(\sum_k \|\|U_{(k)}\|_2^2\|_4^4\Big)^{1/2} + p\|\max\|U_{(k)}\|_2^2\|_p^2 + 4k \\ &\le \sqrt{p}\Big(\sum_k 4^4\Big)^{1/2} + p\max\|\|U_{(k)}\|_2^2\|_q^2 + k \\ &\le \sqrt{pk} + pq^2 + k.\end{aligned}$$

Now plugging into eq. (6.2) and eq. (6.4) we get

$$\begin{aligned}\|\|(T \bullet M)x\|_2^2/k - \|x\|_2^2\|_p &\le (\sqrt{p}\sqrt{\sqrt{pk} + pq^2 + k} + pq)/k + \sqrt{p/k} + p/k \\ &\le \sqrt{p/k} + pq/k.\end{aligned}$$

Finally we can normalize and plug into the power-Markov inequality:

$$\Pr[\|(T \bullet M)x\|_2^2/k - \|x\|_2^2\|_p \ge \epsilon] \le \max(\sqrt{p/k}, pq/k)^p \epsilon^{-p},$$

which gives that we must take

$$k = \epsilon^{-2}\log 1/\delta + \epsilon^{-1}(\log 1/\delta)^2.$$

□

Proofs of the lemmas:

Lemma 6.7.1 (Max trick). *Let* $q = \max(p, \log k)$*, then*

$$\|\max X_i\|_p \le e\max\|EX_i\|_q.$$

Proof.

$$\begin{aligned}\|\max X_i\|_p &\le \|\max X_i\|_q \\ &= (E\max X_i^q)^{1/q} \\ &\le (\sum EX_i^q)^{1/q} \\ &\le (k\max EX_i^q)^{1/q} \\ &\le e(\max EX_i^q)^{1/q} \\ &= e\max\|EX_i\|_q.\end{aligned}$$

□

[1] $\|\|U\|_F^2\|_p = \|\sum_j M_j X^T X M_j^T\|_p \le \sqrt{pk}\|X\|_F^2 + p\|X\|^2$.

Lemma 6.7.2 (p-norm Bernstein)**.** *For independent variables X_i,*

$$\|\sum_i X_i - E\sum_i X_i\|_p \le \sqrt{p}\,\big(\sum_i EX_i^2\big)^{1/2} + p\,\|\max_i X_i\|_p.$$

Proof.

$$\begin{aligned}
\|\sum_i X_i - E\sum_i X_i\|_p &\le \|\sum_i g_i X_i\|_p && \text{(See notes)}\\
&\le \sqrt{p}\,\|\sum_i X_i^2\|_{p/2}^{1/2}\\
&\le \sqrt{p}\,\big(E\sum_i X_i^2\big)^{1/2} + \sqrt{p}\,\|\sum_i X_i^2 - E\sum_i X_i^2\|_{p/2}^{1/2} && \text{(Triangle)}\\
&\le \sqrt{p}\sigma + \sqrt{p}\,\|\sum_i X_i^2 - E\sum_i X_i^2\|_{p/2}^{1/2}\\
&\le \sqrt{p}\sigma + \sqrt{p}\,\|\sum_i g_i X_i^2\|_{p/2}^{1/2}\\
&\le \sqrt{p}\sigma + \sqrt{p}\,\|\max_i X_i \sum_i g_i X_i\|_{p/2}^{1/2}\\
&\le \sqrt{p}\sigma + \sqrt{p}\,\|\max_i X_i\|_p^{1/2}\|\sum_i g_i X_i\|_p^{1/2}. && \text{(Cauchy)}
\end{aligned}$$

Now let $Q = \|\sum_i g_i X_i\|_p^{1/2}$ and $K = \|\max_i X_i\|_p$, and we have $Q^2 \le \sqrt{p}\sigma + \sqrt{pK}Q$. Because it's a quadratic form, Q is upper bounded by the larger root of $Q^2 - \sqrt{pK}Q - \sqrt{p}\sigma$. By calculation, $Q^2 \le \sqrt{p}\sigma + pK$, which is the theorem. □

6.7.2 Proof of polynomial lemma

Proof of Lemma 6.6.1

Proof. For each monomial $\alpha x^S y^T$ of P, where S and T are multisets : $[d] \to \mathbb{N}$, define $[\chi^S y^T]_x$ and $[\chi^S y^T]_y$ be two vectors in R^ℓ for some ℓ such that

$$\langle [\chi^S y^T]_x, [\chi^S y^T]_y \rangle = \sum_\pi x^{\pi S} y^{\pi T} = \sum_\pi \prod_{i=1}^{d} x_i^{S_{\pi i}} y_i^{T_{\pi i}}. \tag{6.5}$$

Then $f(x)$ and $g(y)$ are simple the sketched concatenation of $\alpha[\chi^S y^T]_x$ and $[\chi^S y^T]_y$ vectors. (Note that since we get $\epsilon\|f(x)\|_2\|g(y)\|_2$ error, it doesn't matter where we put alpha, or if we split it between f and g.)

We can let $[\chi^\varnothing y^\varnothing]_x = [\chi^\varnothing y^\varnothing]_y = [1] \in \mathbb{R}^1$ (the single 1 vector) and then define recursively:

$$\begin{aligned}
[\chi^S y^T]_x &= x^{\circ S_i} \otimes [\chi^{S\setminus i} y^{T\setminus i}]_x \oplus - \bigoplus_{j\in(S\cup T)\setminus i} [\chi^{S\setminus i+\{j:S_i\}} y^{T\setminus i+\{j:T_i\}}]_x\\
[\chi^S y^T]_y &= y^{\circ T_i} \otimes [\chi^{S\setminus i} y^{T\setminus i}]_y \oplus \bigoplus_{j\in(S\cup T)\setminus i} [\chi^{S\setminus i+\{j:S_i\}} y^{T\setminus i+\{j:T_i\}}]_y,
\end{aligned}$$

where i is any index in $S \cup T$. Here we let $S \setminus i$ be S with i removed, and $S + \{j : S_i\}$ be S with S_i added to S_j. It is clear from Theorem 23 that this construction gives eq. (6.5).

We note that we can compute the norms by $\|[\chi^{\emptyset}y^{\emptyset}]\|_2^2 = 1$ and

$$\|[\chi^S y^T]_x\|_2^2 = \|x^{\circ S_i}\|_2^2 \cdot \|[\chi^{S\setminus i}y^{T\setminus i}]_x\|_2^2 + \sum_{j\in(S\cup T)\setminus i} \|[\chi^{S\setminus i+\{j:S_i\}}y^{T\setminus i+\{j:T_i\}}]_x\|_2^2$$

and equivalently for y. It does however not seem simple to get a closed form in the general case. In the simple case where x and y are $\{0,1\}^d$ vectors we can however show the simple formula:

$$\|[\chi^S y^T]_x\|_2^2 = \frac{(\|x\|_2^2 - 1 + |S\cup T|)!}{(\|x\|_2^2 - 1)!} \leq (\|x\|_2^2 - 1 + |S \cup T|)^{|S\cup T|}$$

and equivalently for y, (Here S and T are normal sets.)

Since there are only $4^{|}S \cup T|$ many states of $[\chi^S y^T]$ the running time is only that many sketching operations.

□

6.7.3 Better Approximate Matrix Multiplication

Lemma 6.7.3. *For any $x, y \in \mathbb{R}^d$, if S has the (ϵ,δ)-JL-moment-property, $(\|\|Sx\|_2 - \|x\|_2\|_p \leq \epsilon\delta^{1/p}\|x\|_2)$, then also*

$$\|(Sx)^T(Sy) - x^Ty\|_p \leq \epsilon\delta^{1/p}\|x\|_2\|y\|_2$$

Proof. We can assume by linearity of the norms that $\|x\|_2 = \|y\|_2 = 1$. We then use that $\|x-y\|_2^2 = \|x\|_2^2 + \|y\|_2^2 - 2x^Ty$ and $\|x+y\|_2^2 = \|x\|_2^2 + \|y\|_2^2 + 2x^Ty$.

$$\begin{aligned}
\|(Sx)^T(Sy) - x^Ty\|_p &= \|\|Sx + Sy\|_2^2 - \|x+y\|_2^2 - \|Sx - Sy\|_2^2 + \|x-y\|_2^2\|_p/4 \\
&\leq (\|\|S(x+y)\|_2^2 - \|x+y\|_2^2\|_p + \|\|S(x-y)\|_2^2 - \|x-y\|_2^2\|_p)/4 \\
&\leq \epsilon\delta^{1/p}(\|x+y\|_2^2 + \|x-y\|_2^2)/4 \quad \text{(JL property)} \\
&= \epsilon\delta^{1/p}(\|x\|_2^2 + \|y\|_2^2)/2 \\
&\leq \epsilon\delta^{1/p}.
\end{aligned}$$

Combined with the argument in [188] this gives that the JL-moment-property implies Approximate Matrix Multiplication without a factor 3 on ϵ. □

Bibliography

[1] Amir Abboud, Aviad Rubinstein, and Ryan Williams. Distributed PCP theorems for hardness of approximation in P. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 25–36. IEEE, 2017.

[2] Amir Abboud, Ryan Williams, and Huacheng Yu. More Applications of the Polynomial Method to Algorithm Design. In *Proc. 26th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 218–230, 2015.

[3] Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 434–443. IEEE, 2014.

[4] Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *International Colloquium on Automata, Languages, and Programming*, pages 39–51. Springer, 2014.

[5] Milton Abramowitz and Irene A Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*. Courier Corporation, 1964.

[6] Panagiotis Achlioptas, Bernhard Schölkopf, and Karsten Borgwardt. Two-locus association mapping in subquadratic time. In *Proc. 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 726–734. ACM, 2011.

[7] Pankaj K Agarwal and Jeff Erickson. Geometric range searching and its relatives. *Contemporary Mathematics 223*, pages 1–56, 1999.

[8] Parag Agrawal, Arvind Arasu, and Raghav Kaushik. On indexing error-tolerant set containment. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 927–938. ACM, 2010.

[9] Thomas Ahle. It is NP-hard to verify an LSF on the sphere. Unpublished manuscript, 2017.

[10] Thomas D Ahle, Martin Aumüller, and Rasmus Pagh. Parameter-free Locality Sensitive Hashing for Spherical Range Reporting. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 239–256. SIAM, 2017.

[11] Thomas Dybdahl Ahle. Optimal las vegas locality sensitive data structures. In *Annual Symposium on Foundations of Computer Science - Proceedings*, volume 2017-Octob, pages 938–949, 2017. `doi:10.1109/FOCS.2017.91`.

[12] Thomas Dybdahl Ahle, Rasmus Pagh, Ilya Razenshteyn, and Francesco Silvestri. On the complexity of inner product similarity join. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 151–164. ACM, 2016.

[13] Nir Ailon and Bernard Chazelle. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 557–563. ACM, 2006.

[14] Josh Alman, Timothy M Chan, and Ryan Williams. Polynomial representations of threshold functions and algorithmic applications. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 467–476. IEEE, 2016.

[15] Josh Alman and Ryan Williams. Probabilistic Polynomials and Hamming Nearest Neighbors. In *Proceedings of 56th Symposium on Foundations of Computer Science (FOCS)*, pages 136–150, 2015.

[16] Noga Alon, Dana Moshkovitz, and Shmuel Safra. Algorithmic construction of sets for k-restrictions. *ACM Transactions on Algorithms (TALG)*, 2(2):153–177, 2006.

[17] Alexandr Andoni. *Nearest Neighbor Search: the Old, the New, and the Impossible*. PhD thesis, MIT, 2009.

[18] Alexandr Andoni. HIGH FREQUENCY MOMENTS VIA MAX-STABILITY Alexandr Andoni. From Duplicate 1 (High frequency moments via max-stability - Andoni, Alexandr) Unpublished manuscript From Duplicate 2 (HIGH FREQUENCY MOMENTS VIA MAX-STABILITY Alexandr Andoni - Andoni, Alexandr) Unpublished manuscript,, 2017.

[19] Alexandr Andoni and Piotr Indyk. {E2LSH}, User Manual. *Website*, 2005.

[20] Alexandr Andoni and Piotr Indyk. Near-optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. *Commun. ACM*, 51(1):117–122, 2008.

[21] 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.

[22] Alexandr Andoni, Piotr Indyk, Huy L. Nguyen, and Ilya P. Razenshteyn. Beyond locality-sensitive hashing. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1018–1028, 2014. URL: `https://doi.org/10.1137/1.9781611973402.76`, `doi:10.1137/1.9781611973402.76`.

[23] Alexandr Andoni, Robert Krauthgamer, and Ilya P Razenshteyn. Sketching and Embedding are Equivalent for Norms. In *Proc. 47th {ACM} on Symposium on Theory of Computing, (STOC)*, pages 479–488, 2015.

[24] Alexandr Andoni, Thijs Laarhoven, Ilya Razenshteyn, and Erik Waingarten. Lower Bounds on Time-Space Trade-Offs for Approximate Near Neighbors. *arXiv:1605.02701*, 2016.

[25] Alexandr Andoni, Thijs Laarhoven, Ilya Razenshteyn, and Erik Waingarten. Optimal hashing-based time-space trade-offs for approximate near neighbors. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 47–66. Society for Industrial and Applied Mathematics, 2017.

[26] Alexandr Andoni, Thijs Laarhoven, Ilya Razenshteyn, and Erik Waingarten. Optimal hashing-based time-space trade-offs for approximate near neighbors. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 47–66. SIAM, 2017.

[27] Alexandr Andoni, Assaf Naor, Aleksandar Nikolov, Ilya Razenshteyn, and Erik Waingarten. Data-dependent hashing via nonlinear spectral gaps. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 787–800. ACM, 2018.

[28] Alexandr Andoni, Assaf Naor, Aleksandar Nikolov, Ilya Razenshteyn, and Erik Waingarten. Hölder homeomorphisms and approximate nearest neighbors. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 159–169. IEEE, 2018.

[29] Alexandr Andoni, Ilya Razenshteyn, and Negev Shekel Nosatzki. Lsh forest: Practical algorithms made theoretical. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 67–78. SIAM, 2017.

[30] Alexandr Andoni, Ilya Razenshteyn, and Negev Shekel Nosatzki. {LSH} Forest: {P}ractical Algorithms Made Theoretical. In *To appear in Proceedings of 28th {ACM-SIAM} Symposium on Discrete Algorithms (SODA)*, 2017.

[31] Alexandr Andoni and Ilya P. Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 793–801, 2015. URL: `https://doi.org/10.1145/2746539.2746553`, `doi:10.1145/2746539.2746553`.

[32] Arvind Arasu, Venkatesh Ganti, and Raghav Kaushik. Efficient Exact Set-Similarity Joins. In *Proc. International Conference on Very Large Data Bases (VLDB)*, pages 918–929, 2006.

[33] Sunil Arya, Guilherme D Da Fonseca, and David M Mount. A unified approach to approximate proximity searching. In *Proceedings of 18th European Symposium on Algorithms (ESA)*, pages 374–385. Springer, 2010.

[34] Sunil Arya, David M Mount, Nathan S Netanyahu, Ruth Silverman, and Angela Y Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM)*, 45(6):891–923, 1998.

[35] Nikolaus Augsten and Michael H Böhlen. Similarity joins in relational database systems. *Synthesis Lectures on Data Management*, 5(5):1–124, 2013.

[36] Martin Aumüller, Tobias Christiani, Rasmus Pagh, and Francesco Silvestri. Distance-sensitive hashing. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10-15, 2018*, pages 89–104, 2018. URL: `https://doi.org/10.1145/3196959.3196976`, `doi:10.1145/3196959.3196976`.

[37] Haim Avron, Huy L Nguyen, and David P Woodruff. Subspace Embeddings for the Polynomial Kernel. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2258–2266, 2014. URL: `http://papers.nips.cc/paper/5240-subspace-embeddings-for-the-polynomial-kernel`.

[38] Yoram Bachrach, Yehuda Finkelstein, Ran Gilad-Bachrach, Liran Katzir, Noam Koenigstein, Nir Nice, and Ulrich Paquet. Speeding Up the Xbox Recommender System Using a Euclidean Transformation for Inner-product Spaces. In *Proc. 8th ACM Conference on Recommender Systems*, pages 257–264, 2014.

[39] Arturs Backurs and Piotr Indyk. Edit Distance Cannot Be Computed in Strongly Subquadratic Time (Unless SETH is False). In *Proc. 47th ACM on Symposium on Theory of Computing (STOC)*, pages 51–58, 2015.

[40] Bahman Bahmani, Ashish Goel, and Rajendra Shinde. Efficient distributed locality sensitive hashing. In *Proc. ACM International Conference on Information and Knowledge Management (CIKM)*, pages 2174–2178, 2012.

[41] Mayank Bawa, Tyson Condie, and Prasanna Ganesan. LSH Forest: Self-tuning Indexes for Similarity Search. In *Proceedings of the 14th International Conference on World Wide Web*, pages 651–660. ACM, 2005. URL: `http://doi.acm.org/10.1145/1060745.1060840`, `doi:10.1145/1060745.1060840`.

[42] Roberto J Bayardo, Yiming Ma, and Ramakrishnan Srikant. Scaling up all pairs similarity search. In *Proc. International Conference on World Wide Web (WWW)*, pages 131–140, 2007.

[43] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 10–24. SIAM, 2016.

[44] Iddo Ben-Ari and Keith Conrad. Maclaurin's Inequality and a Generalized Bernoulli Inequality. *Mathematics Magazine*, 87(1):14–24, 2014.

[45] Vidmantas Bentkus. A Lyapunov-type bound in Rd. *Theory of Probability & Its Applications*, 49(2):311–323, 2005.

[46] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[47] Andrei Z Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings*, pages 21–29. IEEE, 1997.

[48] Andrei Z Broder, Steven C Glassman, Mark S Manasse, and Geoffrey Zweig. Syntactic clustering of the web. *Computer Networks and ISDN Systems*, 29(8-13):1157–1166, 1997.

[49] Timothy M Chan. Orthogonal range searching in moderate dimensions: kd trees and range trees strike back. In *33rd International Symposium on Computational Geometry (SoCG 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[50] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*, pages 693–703. Springer, 2002.

[51] Moses Charikar, Piotr Indyk, and Rina Panigrahy. New algorithms for subset query, partial match, orthogonal range searching, and related problems. In *International Colloquium on Automata, Languages, and Programming*, pages 451–462. Springer, 2002.

[52] 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.

[53] Surajit Chaudhuri, Venkatesh Ganti, and Raghav Kaushik. A Primitive Operator for Similarity Joins in Data Cleaning. In *Proc.22nd International Conference on Data Engineering (ICDE)*, page 5, 2006.

[54] Bernard Chazelle, Ding Liu, and Avner Magen. Approximate range searching in higher dimension. *Comput. Geom.*, 39(1):24–29, 2008.

[55] Lijie Chen. On the hardness of approximate and exact (bichromatic) maximum inner product. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:26, 2018. URL: `https://eccc.weizmann.ac.il/report/2018/026`.

[56] Lijie Chen and Ryan Williams. An equivalence class for orthogonal vectors. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 21–40. SIAM, 2019.

[57] Yun Chen and Jignesh M Patel. Efficient evaluation of all-nearest-neighbor queries. In *Proc. International Conference on Data Engineering (ICDE)*, pages 1056–1065, 2007.

[58] Flavio Chierichetti and Ravi Kumar. LSH-preserving functions and their applications. *J. ACM*, 62(5):33, 2015.

[59] Flavio Chierichetti and Ravi Kumar. Lsh-preserving functions and their applications. *Journal of the ACM (JACM)*, 62(5):33, 2015.

[60] Seung-Seok Choi, Sung-Hyuk Cha, and Charles C Tappert. A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics*, 8(1):43–48, 2010.

[61] Tobias Christiani. A framework for similarity search with space-time tradeoffs using locality-sensitive filtering. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 31–46. SIAM, 2017.

[62] Tobias Christiani. Fast Locality-Sensitive Hashing Frameworks for Approximate Near Neighbor Search. *arXiv preprint arXiv:1708.07586*, 2017.

[63] Tobias Christiani and Rasmus Pagh. Set similarity search beyond minhash. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1094–1107, 2017. URL: `https://doi.org/10.1145/3055399.3055443`, `doi:10.1145/3055399.3055443`.

[64] Tobias Christiani, Rasmus Pagh, and Mikkel Thorup. Confirmation Sampling for Exact Nearest Neighbor Search. *arXiv preprint arXiv:1812.02603*, 2018.

[65] Kenneth L Clarkson. A randomized algorithm for closest-point queries. *SIAM Journal on Computing*, 17(4):830–847, 1988.

[66] Edith Cohen, Mayur Datar, Shinji Fujiwara, Aristides Gionis, Piotr Indyk, Rajeev Motwani, Jeffrey D Ullman, and Cheng Yang. Finding Interesting Associations without Support Pruning. *{IEEE} Trans. Knowl. Data Eng.*, 13(1):64–78, 2001.

[67] Michael B Cohen, T S Jayram, and Jelani Nelson. Simple analyses of the sparse Johnson-Lindenstrauss transform. In *1st Symposium on Simplicity in Algorithms (SOSA 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[68] Charles J Colbourn and Jeffrey H Dinitz. *Handbook of combinatorial designs*. CRC press, 2006.

[69] Richard Cole, Lee-Ad Gottlieb, and Moshe Lewenstein. Dictionary matching and indexing with errors and don't cares. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 91–100. ACM, 2004.

[70] Ryan R Curtin, Alexander G Gray, and Parikshit Ram. Fast Exact Max-Kernel Search. In *Proc. 13th SIAM International Conference on Data Mining (SDM)*, pages 1–9, 2013.

[71] Søren Dahlgaard, Mathias Bæk Tejs Knudsen, and Mikkel Thorup. Fast similarity sketching. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 663–671. IEEE, 2017.

[72] Abhinandan Das, Mayur Datar, Ashutosh Garg, and ShyamSundar Rajaram. Google news personalization: scalable online collaborative filtering. In *Proc. International Conference on World Wide Web (WWW)*, pages 271–280, 2007.

[73] 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.

[74] Thomas Dean, Mark Ruzon, Mark Segal, Jonathon Shlens, Sudheendra Vijayanarasimhan, and Jay Yagnik. Fast, Accurate Detection of 100,000 Object Classes on a Single Machine. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Washington, DC, USA, 2013.

[75] Ian H Dinwoodie. Large Deviations Techniques and Applications (Amir Dembo and Ofer Zeitouni). *{SIAM} Review*, 36(2):303–304, 1994. URL: `https://doi.org/10.1137/1036078`, `doi:10.1137/1036078`.

[76] Wei Dong, Zhe Wang, William Josephson, Moses Charikar, and Kai Li. Modeling {LSH} for performance tuning. In *Proceedings of 17th {ACM} Conference on Information and Knowledge Management (CIKM)*, pages 669–678. ACM, 2008.

[77] Moshe Dubiner. Bucketing coding and information theory for the statistical high-dimensional nearest-neighbor problem. *IEEE Transactions on Information Theory*, 56(8):4166–4179, 2010.

[78] Peter Elias. List decoding for noisy channels. In *1957-IRE WESCON Convention Record, Pt.* Citeseer, 1957.

[79] Leonhard Euler. De progressionibus harmonicis observationes. *Commentarii academiae scientiarum Petropolitanae*, 7(1734-35):150–156, 1740.

[80] P F Felzenszwalb, R B Girshick, D McAllester, and D Ramanan. Object Detection with Discriminatively Trained Part-Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence,*, 32(9):1627–1645, 2010.

[81] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. In *Discrete Mathematics and Theoretical Computer Science*, pages 137–156. Discrete Mathematics and Theoretical Computer Science, 2007.

[82] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. *EMNLP*, pages 457–468, 2016. URL: `http://aclweb.org/anthology/D/D16/D16-1044.pdf`.

[83] Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. Compact bilinear pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 317–326, 2016.

[84] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity Search in High Dimensions via Hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, pages 518–529. Morgan Kaufmann Publishers Inc., 1999.

[85] Ashish Goel and Pankaj Gupta. Small subset queries and bloom filters using ternary associative memories, with applications. *ACM SIGMETRICS Performance Evaluation Review*, 38(1):143–154, 2010.

[86] Mayank Goswami, Rasmus Pagh, Francesco Silvestri, and Johan Sivertsen. Distance Sensitive Bloom filters without false negatives. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 257–269. SIAM, 2017.

[87] Uffe Haagerup and Magdalena Musat. On the best constants in noncommutative Khintchine-type inequalities. *Journal of Functional Analysis*, 250(2):588–624, 2007.

[88] Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate Nearest Neighbor: Towards Removing the Curse of Dimensionality. *Theory of Computing*, 8(1):321–350, 2012.

[89] Sariel Har-Peled and Sepideh Mahabadi. Proximity in the Age of Distraction: Robust Approximate Nearest Neighbor Search. *CoRR*, abs/1511.0, 2015.

[90] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

[91] Piotr Indyk. Dimensionality reduction techniques for proximity problems. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 371–378. Society for Industrial and Applied Mathematics, 2000.

[92] Piotr Indyk. *High-dimensional computational geometry*. PhD thesis, Stanford University, 2000.

[93] Piotr Indyk. On approximate nearest neighbors under ell-infty norm. *Journal of Computer and System Sciences*, 63(4):627–638, 2001.

[94] Piotr Indyk. Uncertainty principles, extractors, and explicit embeddings of l2 into l1. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 615–620. ACM, 2007.

[95] 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.

[96] Piotr Indyk and Rajeev Motwani. Approximate Nearest Neighbors: {T}owards Removing the Curse of Dimensionality. In *Proceedings of 30th Annual {ACM} Symposium on the Theory of Computing (STOC)*, pages 604–613, 1998.

[97] Piotr Indyk and Assaf Naor. Nearest-neighbor-preserving embeddings. *ACM Transactions on Algorithms (TALG)*, 3(3):31, 2007.

[98] Edwin H Jacox and Hanan Samet. Metric space similarity joins. *ACM Transactions on Database Systems (TODS)*, 33(2):7, 2008.

[99] Yu Jiang, Dong Deng, Jiannan Wang, Guoliang Li, and Jianhua Feng. Efficient parallel partition-based algorithms for similarity search and join with edit distance constraints. In *Proc. Joint EDBT/ICDT Workshops*, pages 341–348. ACM, 2013.

[100] Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane Training of Structural SVMs. *Mach. Learn.*, 77(1):27–59, 2009.

[101] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910, 2017.

[102] William B Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

[103] Michael Kapralov. Smooth Tradeoffs between Insert and Query Complexity in Nearest Neighbor Search. In *Proceedings of 34th {ACM} Symposium on Principles of Database Systems (PODS)*, pages 329–342, 2015.

[104] Michael Kapralov. Smooth tradeoffs between insert and query complexity in nearest neighbor search. In *Proceedings of the 34th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 329–342. ACM, 2015.

[105] Michael Kapralov, Rasmus Pagh, Ameya Velingker, David Woodruff, and Amir Zandieh. Sketching High-Degree Polynomial Kernels. 2019.

[106] Matta Karppa, Petteri Kaski, and Jukka Kohonen. A faster subquadratic algorithm for finding outlier correlations. In *Proc. 27th ACM-SIAM Symposium on Discrete Algorithms (SODA16)*, 2016.

[107] Matti Karppa, Petteri Kaski, and Jukka Kohonen. A faster subquadratic algorithm for finding outlier correlations. *ACM Transactions on Algorithms (TALG)*, 14(3):31, 2018.

[108] Matti Karppa, Petteri Kaski, Jukka Kohonen, and Padraig Ó Catháin. Explicit correlation amplifiers for finding outlier correlations in deterministic subquadratic time. *Proceedings of the 24th European Symposium Of Algorithms*, 2016.

[109] Donald E Knuth. Combinatorial Algorithms: Part 2, {The Art of Computer Programming}, vol. 4A, 2011.

[110] Noam Koenigstein, Parikshit Ram, and Yuval Shavitt. Efficient Retrieval of Recommendations in a Matrix Factorization Framework. In *Proc. 21st ACM International Conference on Information and Knowledge Management (CIKM)*, pages 535–544, 2012.

[111] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8):30–37, 2009.

[112] Felix Krahmer and Rachel Ward. New and improved Johnson–Lindenstrauss embeddings via the restricted isometry property. *SIAM Journal on Mathematical Analysis*, 43(3):1269–1281, 2011.

[113] Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM Journal on Computing*, 30(2):457–474, 2000.

[114] Thijs Laarhoven. *Search problems in cryptography*. PhD thesis, PhD thesis, Eindhoven University of Technology, 2015. http://www. thijs. com˜. . . , 2015.

[115] Thijs Laarhoven. Tradeoffs for nearest neighbors on the sphere. *arXiv preprint arXiv:1511.07527*, 2015.

[116] Thijs Laarhoven. Hypercube {LSH} for Approximate near Neighbors. In *42nd International Symposium on Mathematical Foundations of Computer Science, {MFCS} 2017, August 21-25, 2017 - Aalborg, Denmark*, pages 7:1—-7:20, 2017. URL: `https://doi.org/10.4230/LIPIcs.MFCS.2017.7`, `doi:10.4230/LIPIcs.MFCS.2017.7`.

[117] Kasper Green Larsen and Jelani Nelson. The Johnson-Lindenstrauss lemma is optimal for linear dimensionality reduction. *CoRR*, abs/1411.2, 2014.

[118] Kasper Green Larsen and Jelani Nelson. Optimality of the Johnson-Lindenstrauss lemma. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 633–638. IEEE, 2017.

[119] Quoc Viet Le, Tamás Sarlós, and Alexander Johannes Smola. Fastfood: Approximate Kernel Expansions in Loglinear Time. *CoRR*, abs/1408.3, 2014. URL: `http://arxiv.org/abs/1408.3060`, `arXiv:1408.3060`.

[120] Dongjoo Lee, Jaehui Park, Junho Shim, and Sang-goo Lee. An efficient similarity join algorithm with cosine similarity predicate. In *Database and Expert Systems Applications*, pages 422–436. Springer, 2010.

[121] Guoliang Li, Dong Deng, Jiannan Wang, and Jianhua Feng. Pass-join: A partition-based method for similarity joins. *Proc. VLDB Endowment*, 5(3):253–264, 2011.

[122] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 1449–1457, 2015.

[123] Yucheng Low and Alice X Zheng. Fast top-k similarity queries via matrix compression. In *Proc. ACM International Conference on Information and Knowledge Management (CIKM)KM*, pages 2070–2074. ACM, 2012.

[124] Jiaheng Lu, Chunbin Lin, Wei Wang, Chen Li, and Haiyong Wang. String similarity measures and joins with synonyms. In *Proc. 2013 ACM SIGMOD International Conference on Management of Data*, pages 373–384, 2013.

[125] Qin Lv, William Josephson, Zhe Wang, Moses Charikar, and Kai Li. Multi-probe {LSH}: Efficient Indexing for High-dimensional Similarity Search. In *Proceedings of 33rd International Conference on Very Large Data Bases (VLDB)*, pages 950–961. VLDB Endowment, 2007.

[126] Harry G Mairson. The program complexity of searching a table. In *Foundations of Computer Science, 1983., 24th Annual Symposium on*, pages 40–47. IEEE, 1983.

[127] Jir\'\i Matousek. Geometric Range Searching. *{ACM} Comput. Surv.*, 26(4):421–461, 1994.

[128] Stefan Meiser. Point location in arrangements of hyperplanes. *Information and Computation*, 106(2):286–303, 1993.

[129] Sergey Melnik and Hector Garcia-Molina. Adaptive algorithms for set containment joins. *ACM Transactions on Database Systems (TODS)*, 28(1):56–99, 2003.

[130] Antoine Miech, Ivan Laptev, and Josef Sivic. Learnable pooling with context gating for video classification. *arXiv preprint arXiv:1706.06905*, 2017.

[131] Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge university press, 2005.

[132] Rajeev Motwani, Assaf Naor, and Rina Panigrahi. Lower Bounds on Locality Sensitive Hashing. In *Proc. 22nd Symposium on Computational Geometry (SoCS)*, pages 154–157, 2006.

[133] Rajeev Motwani, Assaf Naor, and Rina Panigrahi. Lower bounds on locality sensitive hashing. In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 154–157. ACM, 2006.

[134] Cameron Musco and Christopher Musco. Recursive sampling for the nystrom method. In *Advances in Neural Information Processing Systems*, pages 3833–3845, 2017.

[135] Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM journal on computing*, 22(4):838–856, 1993.

[136] Moni Naor, Leonard J Schulman, and Aravind Srinivasan. Splitters and near-optimal derandomization. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 182–191. IEEE, 1995.

[137] Jelani Nelson, Huy L Nguyen, and David P Woodruff. On deterministic sketching and streaming for sparse recovery and norm estimation. *Linear Algebra and its Applications*, 441(0):152–167, 2014.

[138] Behnam Neyshabur and Nathan Srebro. On Symmetric and Asymmetric LSHs for Inner Product Search. In *International Conference on Machine Learning*, pages 1926–1934, 2015.

[139] Behnam Neyshabur and Nathan Srebro. On symmetric and asymmetric lshs for inner product search. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1926–1934, 2015. URL: `http://jmlr.org/proceedings/papers/v37/neyshabur15.html`.

[140] Nam H Nguyen, Petros Drineas, and Trac D Tran. Tensor sparsification via a bound on the spectral norm of random tensors. *Information and Inference: A Journal of the IMA*, 4(3):195–229, 2015.

[141] Ryan O'Donnell. *Analysis of boolean functions*. Cambridge University Press, 2014.

[142] Ryan O'Donnell, Yi Wu, and Yuan Zhou. Optimal lower bounds for locality-sensitive hashing (except when q is tiny). *TOCT*, 6(1):5:1–5:13, 2014. URL: `https://doi.org/10.1145/2578221`, `doi:10.1145/2578221`.

[143] Krzysztof Oleszkiewicz. On a nonsymmetric version of the Khinchine-Kahane inequality. In *Stochastic inequalities and applications*, pages 157–168. Springer, 2003.

[144] Mark H Overmars and Jan van Leeuwen. Worst-case optimal insertion and deletion methods for decomposable searching problems. *Information Processing Letters*, 12(4):168–173, 1981.

[145] Ryan O'Donnell, Yi Wu, and Yuan Zhou. Optimal lower bounds for locality-sensitive hashing (except when q is tiny). *ACM Transactions on Computation Theory (TOCT)*, 6(1):5, 2014.

[146] Andrzej Pacuk, Piotr Sankowski, Karol Wegrzycki, and Piotr Wygocki. Locality-sensitive hashing without false negatives for lp. In *International Computing and Combinatorics Conference*, pages 105–118. Springer, 2016.

[147] Rasmus Pagh. Compressed matrix multiplication. *ACM Transactions on Computation Theory (TOCT)*, 5(3):9, 2013.

[148] Rasmus Pagh. Locality-sensitive hashing without false negatives. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1–9. SIAM, 2016.

[149] Rasmus Pagh, Ninh Pham, Francesco Silvestri, and Morten Stöckel. {I/O}-Efficient Similarity Join. In *Proc. 23rd European Symposium on Algorithms (ESA)*, pages 941–952, 2015.

[150] Rasmus Pagh, Francesco Silvestri, Johan Sivertsen, and Matthew Skala. Approximate Furthest Neighbor in High Dimensions. In *Proc. 8th International Conference on Similarity Search and Applications (SISAP)*, volume 9371 of *LNCS*, pages 3–14, 2015.

[151] Rina Panigrahy. Entropy based nearest neighbor search in high dimensions. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1186–1195. Society for Industrial and Applied Mathematics, 2006.

[152] Rina Panigrahy, Kunal Talwar, and Udi Wieder. A geometric approach to lower bounds for approximate near-neighbor search and partial match. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 414–423. IEEE, 2008.

[153] Valentin Petrov. *Sums of independent random variables*, volume 82. Springer Science & Business Media, 2012.

[154] Ninh Pham. Hybrid {LSH:} Faster Near Neighbors Reporting in High-dimensional Space. In *Proceedings of the 20th International Conference on Extending Database Technology, {EDBT} 2017, Venice, Italy, March 21-24, 2017.*, pages 454–457, 2017. URL: `https://doi.org/10.5441/002/edbt.2017.43`, `doi:10.5441/002/edbt.2017.43`.

[155] Ninh Pham and Rasmus Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 239–247. ACM, 2013.

[156] Ninh Pham and Rasmus Pagh. Scalability and total recall with fast CoveringLSH. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1109–1118. ACM, 2016.

[157] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

[158] Parikshit Ram and Alexander G Gray. Maximum Inner-product Search Using Cone Trees. In *Proc. 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 931–939, 2012.

[159] Karthikeyan Ramasamy, Jignesh M. Patel, Jeffrey F. Naughton, and Raghav Kaushik. Set containment joins: The good, the bad and the ugly. In *VLDB*, 2000.

[160] Demitri Nava Raul Castro Fernandez, Jisoo Min and Samuel Madden. Lazo: A cardinality-based method for coupled estimation of jaccard similarity and containment. In *ICDE*, 2019.

[161] Ilya P. Razenshteyn. *High-dimensional similarity search and sketching: algorithms and hardness*. PhD thesis, Massachusetts Institute of Technology, Cambridge, USA, 2017. URL: `http://hdl.handle.net/1721.1/113934`.

[162] Ronald L Rivest. Partial-match retrieval algorithms. *SIAM Journal on Computing*, 5(1):19–50, 1976.

[163] Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 515–524. ACM, 2013.

[164] Aviad Rubinstein. Hardness of approximate nearest neighbor search. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1260–1268. ACM, 2018.

[165] Venu Satuluri and Srinivasan Parthasarathy. Bayesian locality sensitive hashing for fast similarity search. *Proc. VLDB Endowment*, 5(5):430–441, 2012.

[166] Anshumali Shrivastava and Ping Li. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). In *Advances in Neural Information Processing Systems*, pages 2321–2329, 2014.

[167] Anshumali Shrivastava and Ping Li. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2321–2329, 2014. URL: `http://papers.nips.cc/paper/5329-asymmetric-lsh-alsh-for-sublinear-time-maximum-inner-product-search-mips`.

[168] Anshumali Shrivastava and Ping Li. Asymmetric Minwise Hashing for Indexing Binary Inner Products and Set Containment. In *Proc. 24th International Conference on World Wide Web (WWW)*, pages 981–991, 2015.

[169] Alexander Sidorenko. What we know and what we do not know about Turán numbers. *Graphs and Combinatorics*, 11(2):179–199, 1995.

[170] Yasin N Silva, Walid G Aref, and Mohamed H Ali. The similarity join database operator. In *Proc. International Conference on Data Engineering (ICDE)*, pages 892–903. IEEE, 2010.

[171] Malcolm Slaney, Yury Lifshits, and Junfeng He. Optimal Parameters for Locality-Sensitive Hashing. *Proceedings of the {IEEE}*, 100(9):2604–2623, 2012.

[172] Ryan Spring and Anshumali Shrivastava. Scalable and sustainable deep learning via randomized hashing. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 445–454. ACM, 2017.

[173] Nathan Srebro, Jason D M Rennie, and Tommi S Jaakola. Maximum-Margin Matrix Factorization. In *Advances in Neural Information Processing Systems 17*, pages 1329–1336. MIT Press, 2005.

[174] Nathan Srebro and Adi Shraibman. Rank, Trace-Norm and Max-Norm. In *Proc. 18th Conference on Learning Theory {COLT}*, volume 3559 of *LNCS*, pages 545–560, 2005.

[175] Christina Teflioudi, Rainer Gemulla, and Olga Mykytiuk. LEMP: Fast Retrieval of Large Entries in a Matrix Product. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 107–122. ACM, 2015.

[176] Flemming Topsøe. Some bounds for the logarithmic function. *Inequality theory and applications*, 4:137, 2006.

[177] Csaba D Toth, Joseph O'Rourke, and Jacob E Goodman. *Handbook of discrete and computational geometry*. Chapman and Hall/CRC, 2017.

[178] Paul Turán. Research problems. *Közl MTA Mat. Kutató Int*, 6:417–423, 1961.

[179] Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and the closest pair problem. *Journal of the ACM (JACM)*, 62(2):13, 2015.

[180] Hongya Wang, Jiao Cao, LihChyun Shu, and Davood Rafiei. Locality sensitive hashing revisited. In *CIKM*, pages 1969–1978, 2013. doi:10.1145/2505515.2505765.

[181] Jiannan Wang, Guoliang Li, and Jianhua Fe. Fast-join: An efficient method for fuzzy token matching based string similarity join. In *Proc. International Conference on Data Engineering (ICDE)*, pages 458–469. IEEE, 2011.

[182] Jiannan Wang, Guoliang Li, and Jianhua Feng. Can we beat the prefix filtering?: an adaptive framework for similarity join and search. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 85–96. ACM, 2012.

[183] Ye Wang, Ahmed Metwally, and Srinivasan Parthasarathy. Scalable all-pairs similarity search in metric spaces. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 829–837, 2013.

[184] Roger Weber, Hans-Jörg Schek, and Stephen Blott. A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. In *Proc. 24rd International Conference on Very Large Data Bases (VLDB)*, pages 194–205, 1998.

[185] Alexander Wei. Optimal Las Vegas Approximate Near Neighbors in {\(\mathscr{l}\)}p. In *Proceedings of the Thirtieth Annual {ACM-SIAM} Symposium on Discrete Algorithms, {SODA} 2019, San Diego, California, USA, January 6-9, 2019*, pages 1794–1813, 2019. URL: https://doi.org/10.1137/1.9781611975482.108, doi:10.1137/1.9781611975482.108.

[186] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2):357–365, 2005.

[187] Paweł Wolff. Hypercontractivity of simple random variables. *Studia Mathematica*, 3(180):219–236, 2007.

[188] David P Woodruff. Sketching as a Tool for Numerical Linear Algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1–2):1–157, 2014.

[189] Xian Wu, Moses Charikar, and Vishnu Natchu. Local Density Estimation in High Dimensions. In *Proceedings of the 35th International Conference on Machine Learning, {ICML} 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 5293–5301, 2018. URL: http://proceedings.mlr.press/v80/wu18a.html.

[190] Chenyi Xia, Hongjun Lu, Beng Chin Ooi, and Jing Hu. Gorder: an efficient method for KNN join processing. In *Proc. VLDB*, pages 756–767, 2004.

[191] Chuan Xiao, Wei Wang, Xuemin Lin, and Jeffrey Xu Yu. Efficient similarity joins for near duplicate detection. In *Proc. International Conference on World Wide Web (WWW)*, pages 131–140, 2008.

[192] Xiao Yan, Jinfeng Li, Xinyan Dai, Hongzhi Chen, and James Cheng. Norm-Ranging LSH for Maximum Inner Product Search. In *Advances in Neural Information Processing Systems*, pages 2956–2965, 2018.

[193] Reza Bosagh Zadeh and Ashish Goel. Dimension independent similarity computation. *The Journal of Machine Learning Research*, 14(1):1605–1626, 2013.

[194] Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. *Similarity search: the metric space approach*, volume 32. Springer Science & Business Media, 2006.

[195] Xiang Zhang, Fei Zou, and Wei Wang. Fastanova: an efficient algorithm for genome-wide association study. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 821–829. ACM, 2008.