Optimal Las Vegas Locality Sensitive Data Structures

IT University of Copenhagen

Thomas D. Ahle

April 6 2017

Abstract

We show that approximate similarity search (near neighbour) can be solved in high dimensions with performance matching that of state of the art Locality Sensitive Hashing, but without false negatives. In particular we give two data structures for common problems. For c-approximate near neighbour in Hamming space, for which we get query time $dn^{1/c+o(1)}$ and space $dn^{1+1/c+o(1)}$ matching that of [Indyk and Motwani, 1998] and answering a long standing open question from [Indyk, 2000a] and [Pagh, 2016]. For (s_1, s_2)-approximate Jaccard similarity we get query time $d^2n^{\rho+o(1)}$ and space $d^2n^{1+\rho+o(1)}$, $\rho = \log\frac{1+s_1}{2s_1}\Big/\log\frac{1+s_2}{2s_2}$, when sets have equal norm, matching the performance of [Pagh and Christiani, 2017] and beating the classic 'minhash' algorithm.

Alternatively the data structures can be viewed as efficient constructions of large combinatorial designs with fast decoding algorithms. In particular Turán Systems and a type of codes with the guarantee that any two close points have a common neighbour in the code. This is the first polynomial time construction of large Turán Systems with guarantees matching that of the probabilistic method.

1 Introduction

Locality Sensitive Hashing has been a leading approach to high dimensional similarity search (nearest neighbor search) data structures for the last twenty years. Intense research [Indyk and Motwani, 1998, Gionis et al., 1999, Kushilevitz et al., 2000, Indyk, 2000b, Indyk, 2001, Charikar, 2002, Datar et al., 2004, Lv et al., 2007, Panigrahy, 2006, Andoni and Indyk, 2006, Andoni et al., 2014, Andoni et al., 2017, Becker et al., 2016, Ahle et al., 2017, Aumüller et al., 2017] 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 [Williams, 2005], and give sub-linear query time on worst case instances. They achieve this by being approximate and Monte Carlo, meaning they may return points that a slightly further away than the nearest, and with a small probability they may completely fail to return any of the nearby points.

Definition 1 ((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 $x' \in X$ 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 in other words, if we could rule out

false negatives? The problem is more troublesome that it may look. First it has been observed that the correct error probability is hard to set right [Gionis et al., 1999, Arya et al., 1998] when implementing Locality Sensitive Hashing. A Las Vegas data structure entirely removes the problem. Secondly, there is no way to verify if the algorithm is correct when it says 'no near neighbours' other than iterating every point in the data base, in which case the data structure is entirely useless. Different authors have described this problem as finding algorithms that are 'Las Vegas' [Indyk, 2000a], 'Have no false negatives' [Goswami et al., 2017, Pagh, 2016], 'Have total recall' [Pham and Pagh, 2016], 'Are exact' [Arasu et al., 2006], 'Are explicit' [Karppa et al., 2016].

Recent years have shown progress in this direction. In particular [Pagh, 2016] showed that the problem in Hamming space could be solved as Las Vegas with query time $dn^{1.38/c+o(1)}$, matching the $dn^{1/c}$ data structure of [Indyk and Motwani, 1998] up to a factor 1.38 in the exponent.

In this paper we give an algorithm in the Locality Sensitive Filter framework [Becker et al., 2016, Christiani, 2017], which not only removes the 1.38, but improves to $dn^{1/(2c-1)}$ for large $cr \approx d/2$, matching the algorithms of [Andoni et al., 2015] for Hamming space.

Definition 2 (Approximate similarity search [Pagh and Christiani, 2017])**.** *Let $P \subseteq \{0,1\}^d$ be a set of $|P| = n$ data vectors; let $\mathrm{sim} : \{0,1\}^d \times \{0,1\}^d \to [0,1]$ be a similarity measure; and let $s_1, s_2 \in [0,1]$ be a similarities with $s_1 > s_2$. A solution to the (s_1, s_2)-similarity problem over sim is a data structure that supports the following query operation: on input $q \in \{0,1\}^d$, for which there exists a vector $x \in P$ with $\mathrm{sim}(x,q) \geq s_1$, return $x' \in P$ with $\mathrm{sim}(x',q) > s_2$.*

Our second algorithm solves the approximate similarity search over Jaccard similarity, which is defined for $x, y \in \{0,1\}^d$ seen as sets $x, y \subseteq [d]$ as $\mathrm{sim}(x,y) = |x \cap y|/|x \cup y|$. This has traditionally been solving using the min-hash LSH [Broder et al., 1997, Broder, 1997], which combined with the results of Indyk and Motwani [Indyk and Motwani, 1998] 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 [Pagh and Christiani, 2017] that this could be improved for vectors of equal weight to $\rho = \log \frac{2s_1}{1+s_1} \big/ \log \frac{2s_2}{1+s_2}$. We show that it is possible to achieve this recent result with a data structure that has no false negatives.

1.1 Contribution

We present the first Las Vegas algorithm for approximate near neighbour search, which gives sub-linear query time for any $c > 1$. This solves long standing open question from [Indyk, 2000a] and [Pagh, 2016]. In particular we get the following two theorems:

Theorem 1. *Let $X = \{0,1\}^d$ be the Hamming space with metric $\mathrm{dist}(x,y) = \|x \oplus y\| \in [0,d]$ where $\oplus$ is 'xor' or addition in $\mathbb{Z}_2$. For every choice of constants $0 < r$, $1 < c$ and $cr \leq d/2$, we can solve the (c,r)-approximate nearest neighbour problem in hamming space with query time $d\,n^\rho$ and space usage $d\,n^{1+\rho}$ where $\rho = \frac{1-cr/d}{c(1-r/d)} + \hat{O}((\log n)^{-1/5}) \leq \frac{1}{c} + o(1)$.*

Theorem 2. *Let sim be the Braun-Blanquet similarity $\mathrm{sim}(x,y) = |x \cap y| / \max(|x|,|y|)$. For every choice of constants $0 < b_2 < b < 1$ and $b_2 \leq 1 - \hat{O}(1/\sqrt{\log n})$, we can solve the (b, b_2)-similarity problem over sim with query time d^2, n^ρ and space usage $d^2, n^{1+\rho}$ where $\rho = (\log \frac{2s_1}{1+s_1})/(\log \frac{2s_2}{1+s_2}) + \hat{O}(1/\sqrt{\log n})$.*

The first result matches the lower bounds by [O'Donnell et al., 2014] for 'data independent' LSH data-structures for Hamming distance. The second result matches the corresponding lower bounds by [Pagh and Christiani, 2017] for Braun-Blanquet similarity and by reduction Jaccard similarity. The first result improves upon [Pagh, 2016] by a factor of $\log 4 > 1.38$ in the exponent.

See table 1.2 for more comparisons. By well known deterministic reductions [Indyk, 2007], we also give the best currently known Las Vegas data structures for l_1 and l_2 in $\mathbb{R}^d$.

Detaching the data structures from our constructions, we get the first efficient constructions of large Turán Systems. Our Systems can even be efficiently decoded, which is likely to have many other algorithmic applications.

1.2 Background and Related Work

The classic technique for similarity search is Locality-Sensitive Hashing (LSH), introduced in 1998 by [Indyk and Motwani, 1998, Har-Peled et al., 2012]. The idea is to make a random space partition in which similar points are likely to be stored in the same section. A query can then be more efficiently answered by only considering data points in the same section as the query. Even so, unless the query lands right in the middle of a section, many of the points to consider are going to not be close, and many close points are going to be in other sections. This particular happens in high dimensions, when landing close to the edges becomes very likely. To combat bad points, LSH uses quite fine space partitions. To ensure a high probability of success (good recall) one repeats the above construction, independently at random, a small polynomial number of times, n^ϵ.

In [Pagh, 2016, Arasu et al., 2006] it was shown that one could change the above algorithm to not do the repetitions independently. Doing the same thing repeatedly, independently only makes the probability of error go to zero as the repetitions go to infinity. By making correlated repetitions they showed it was possible to reach 0 much faster, after only $n^{O(\epsilon)}$ repetitions. That is they needed more repetitions than LSH did to get 99% success rate, but fewer than LSH needs for 100%.

An alternative construction to LSH was introduced by [Becker et al., 2016] called Locality Sensitive Filters, LSF. While it achieves the same bounds, it has the advantage of giving more control to the algorithm designer for balancing different values. For example it typically allows good space/time trade offs and better results for low dimensional data $d = O(\log n)$ [Andoni et al., 2017]. The idea is to sample a large number of random sections of the space. In contrast to LSH these 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 words of a random 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 points. In LSH the space partitioning provided such an algorithm. For LSF it is common to use a kind of random product code, but [Pagh and Christiani, 2017] used a more direct construction based on random branching processes. LSF is similar to LSH in that it only gets 100% success rate as the number of sections go 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 share a 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 [Andoni et al., 2014, Andoni and Razenshteyn, 2015, Andoni et al., 2017] 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, but one needs to find a way to derandomize the randomized clustering step.

There is of course also a literature of deterministic and Las Vegas data structures not connected to that of 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 $\binom{d}{r}O(r)$ point of Hamming distance most r. This of course requires $r\log(d/r) < \log n$ to be sub-linear, so for a typical example of $d = (\log n)^3$ and $r = d/4$ it won't be practical. In [Cole et al., 2004] this was somewhat approved to $n(\log n)^r$ time, but it still requires $r = o(\frac{\log n}{\log\log n})$ for queries to be sub-linear.

In l_2 the classical K-d tree algorithm [Bentley, 1975] 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 deterministically, [Arya et al., 1998] found a $(d/(c-1))^d$ algorithm for a $c > 1$ approximation. This allows us to set $d = (\frac{\log n}{\log\log n})$.

For large approximation factors [Har-Peled et al., 2012] gave a deterministic data structure with query time $O(d\log n)$, but space and preprocessing more than $nO(1/(c-1))^d$. More recently [Indyk, 2000a] gave a deterministic $(d\epsilon^{-1}\log n)^{O(1)}$ query time, fully deterministic algorithm with space usage $n^{O(1/\epsilon^6)}$.

See table 1.2 for an easier comparison of the different results and spaces.

1.3 Techniques

Our main new technique is a combination of 'splitters' as defined by [Naor et al., 1995, Alon et al., 2006], and 'tensoring' which is a common technique in the LSH literature.

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 $P = \{p_1 \cap p_2, \dots \mid p_1 \in P_1, p_2 \in P_2, \dots\}$. Usually Tensoring is used to speed up locating the section of a partition that covers a point, however it can also be used to make a low dimensional space partition partition a higher dimensional space.

Unfortunately tensoring doesn't seem to be directly useful for deterministic constructions, since deterministic space partitions have some overhead that gets amplified by the product construction. This is the reason why [Pagh, 2016] construct their 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 the exist, but they tend to be rare, and it would be a tough order to find them for every similarity measure we would like to make a data structure for.

Using our combination of tensoring and splitters, we can amplify small space partitions found using the probabilistic method and verified, or using the greedy set cover algorithm as in [Alon et al., 2006]. We don't quite get a general reduction from Monte Carlo to Las Vegas LSH data structures, but we show how to state of the art algorithms may be converted at a negligible overhead.

As a lemma, we show, for every $\epsilon, r > 0$, the existence of a family of most d functions $f_i : \{0,1\}^d \to \{0,1\}^{O((\log n)/\epsilon^3)}$, with the property that for every $x, y \in \{0,1\}^d$, f doesn't increase distances $\operatorname{dist}(f(x), f(y)) \leq S\operatorname{dist}(x,y)$ and whp. when $\operatorname{dist}(x,y) \geq r$ it doesn't decrease them much: $\operatorname{dist}(f(x), f(y)) \geq (1 - O(\epsilon))S\operatorname{dist}(x,y)$. We believe this improves a factor ϵ in the dimension over previous mappings based on codes [Indyk, 2000a] by allowing randomization in the lower bound.

In the process of showing our results, we prove asymptotic bounds on the intersection ham hamming balls, and the ratio between two binomial coefficients, which we believe may be of separate interest.

Reference	Space	Exponent, search time	Comments
[Bentley, 1975]	l_2	$1 - 1/d$	Exact algorithm, Fully deterministic.
[Cole et al., 2004]	Hamming	$r\frac{\log\log n}{\log n}$	Sub-linear for $r = o(\frac{\log n}{\log\log n})$. Exact.
[Arya et al., 1998]	l_2	$d\frac{\log(d/(c-1))}{\log n}$	Sub-linear for $d = o(\frac{\log n}{\log\log n})$.
[Har-Peled et al., 2012]	Hamming	$o(1)$	c-approximation, Fully deterministic, $(1/(c-1))^d$ space.
[Indyk, 2000a]	Hamming	$o(1)$	$(3+\epsilon)$-approximation, Fully deterministic, $n^{\Omega(1/\epsilon^6)}$ space.
[Arasu et al., 2006]	Hamming	$\approx 3/c$	The paper makes no theoretical claims.
[Pagh, 2016]	Hamming	$1.38/c$	Exponent $1/c$ when $r = o(\log n)$ or $(\log n)/(cr) \in \mathbb{N}$.
This paper	Hamming	$1/c$	Actual exponent is $\frac{1-cr/d}{c(1-r/d)}$ which improves to $1/(2c-1)$ for $cr \approx d/2$.
[Pagh, 2016]	Braun-Blanquet	$1.38\,\frac{1-b}{1-b_2}$	Via reduction to Hamming. Requires sets of equal weight.
This paper	Braun-Blanquet	$\frac{\log 1/b}{\log 1/b_2}$	See [Pagh and Christiani, 2017] figure 2 for a comparison with [Pagh, 2016].

Table 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 [Indyk, 2000a] use less than n^2 space. All algorithms give c-approximations, except for the first two, and for [Indyk, 2000a], which is a $(3+\epsilon)$-approximation.

1.4 Notation

We use $[d] = \{1, \dots, d\}$ and $[a, b] = \{a, \dots, b\}$ as convenient notation for ranges. However sometimes we also use the Iversonian notation $[P]$ for a value that is 1 when P is true and 0 when P is false.

For a set $x \subseteq [d]$, we will sometimes think of it as a subset of universe $[d]$, and at other times as a vector $x \in \{0,1\}^d$, where $x_i = 1$ indicates that element i of $[d]$ is 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 $|x \cap y|$.

For at set $S \subseteq [d]$ and a vector $x \in \{0,1\}^d$, we let x_S be the projection of x unto S. This is an $|S|$ dimensional vector, consisting of the coordinates $\{x_i : i \in S\}$ in the natural order.

We use $S \times T = \{(s,t) : s \in S, t \in T\}$ for the cross product; $S \setminus T$ for set difference, and $x \oplus y$ or 'xor' or 'symmetric difference'. $\mathcal{P}(X)$ is the power set of X.

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.

2 Hamming Space Data Structure

The structure of the algorithm is that of the Locality Sensitive Filter system: We are given a set of points $P \subseteq \{0,1\}^d$. For some set $\mathcal{K}$ of 'keys', we keep a hash table $T : \mathcal{K} \to \mathcal{P}(P)$. For every point $x \in P$ we compute a number of keys $\text{keys}(x) \subseteq \mathcal{K}$ and store is x in each bucket $T[k_1], \dots$ for $k_1 \cdots \in \text{keys}(x)$.

Queries q are answered by iterating all points $x \in \bigcup\{T[\kappa] : \kappa \in \text{keys}(q) \subseteq \mathcal{K}\}$ and calculating $\text{dist}(q, x)$. As soon as we find a point x with $\text{dist}(q, r) \le r$ we return it. In expectation we want that there are only few 'false positives', with $\text{dist}(x, q) > cr$, to consider while every close point should be present.

The performance of the algorithm will be a function of three parts: The complexity of computing keys; the number of keys computed; and the number of false positives to compute the distance to. We will show how to compute $\text{keys}(x)$ with good performance on all three accounts.

Ideally we want to make $\mathcal{K}$ a large code in $\{0,1\}^d$ and assign $\text{keys}(x)$ to the code words with short distance to x. This is a kind of 'spherical' space partition, going to back to [Andoni and Indyk, 2006]. The algorithm can be seen as a way to carefully combining very small, 'brute force constructed' codes into gradually larger ones. We however go the opposite direction, gradually decomposing x into small pieces that can be decoded in the small codes and then reassembled. This provides the decoding algorithm we need.

The method is very distantly related to dimensionality reduction. However techniques such as bit sampling and the Johnson Lindenstrauss lemma are probabilistic and can only take the dimension to $\approx O(\log n)^2$ if we don't want too much noise. In our case we need deterministic guarantees and we need the dimension reduced to $\approx O(\log n)^{3/4}$.

For this we do the following steps

1. Use a non-expanding dimensionality reduction, f to reduce the dimension to $B = (\log n)^{8/5}$, such that $(1-\epsilon) S \,\text{dist}(x, y) \le \text{dist}(f(x), f(y)) \le S \,\text{dist}(x, y)$ for some $S > 0$ and $\epsilon = O((\log n)^{-1/5})$.

2. Use a splitter to reduce the dimension further to $b = \sqrt{B} = (\log n)^{4/5}$. Such a splitter has size $\le B^b = n^{\hat{O}((\log n)^{-1/5})}$.

3. Build a filter matrix $A \in \{0,1\}^{m \times B}$, such that for any pair $x, y \in \{0,1\}^b$ of close points, there is a row $j \in [m]$ $\text{dist}(A_j, x) \le t$ and $\text{dist}(A_j, y) \le t$. Here $m \approx n^{(\log n)^{-1/5}}$, $t = b/2 - s\sqrt{b/4}$ and

$s \approx b^{1/4}$.

4. For each partition $\{\pi_1, \ldots, \pi_{B/b} \subseteq [B]\}$ in the splitter, return every value of the product

$$\{j \in [m] : \mathrm{dist}(A_j, x_{\pi_1}) \leq t\} \times \cdots \times \{j \in [m] : \mathrm{dist}(A_j, x_{\pi_{B/b}}) \leq t\}. \tag{1}$$

The algorithm solves the approximate near neighbour problem, because for any pair $x, y \in \{0,1\}^d$ with $\mathrm{dist}(x,y) \leq r$, there is a shared key $\kappa \in \mathrm{keys}(x) \cap \mathrm{keys}(y)$. We will show this, together with bounds on $E|\mathrm{keys}(x)|$ and $E\sum_{y \in P} |\mathrm{keys}(x) \cap \mathrm{keys}(y)|$, by walking through the steps in more details.

Non-expansive dimensionality reduction. If $d < B = (\log n)^{8/5}$ we can just use d as it is.

If $d/(cr)$ is bounded above by a constant, we can partition in blocks of size $B = \frac{2d \log n}{cr\epsilon^2}$. This guarantees with high probability that a pair of points is not contracted more than a factor $(1-\epsilon)B/d$, while one block must always exists in which the distance is at least B/d times the original value.

However $d/(cr)$ could be as large as $n^{o(1)}$. Hence we use lemma 1 with $\epsilon = (\log n)^{-1/5}$ which gives us $B = O((\log n)^{8/5})$ with the same guarantees. As with simple partitioning, the non-expansion may only hold for one of a number of blocks, but the low contraction always hold, and there are at most d blocks each on which we perform the remaining algorithm.

In any case, in the remaining we may consider instead solving the $((1-\epsilon)c, Sr)$-approximate near neighbour problem in hamming space with dimension B, where $S \geq 0$.

Second dimensionality reduction with splitter. We use lemma 2 to make a (B, b) splitter. This is a set of partitions

$$\{\{\pi_{1,1}, \ldots, \pi_{1,B/b}\}, \ldots, \{\pi_{l,1}, \ldots, \pi_{l,B/b}\}\}$$

of $[B]$ into size b subsets. (We pad B by at most $O(B)$ 0-coordinates to make it a square.)

For any pair $x, y \in \{0,1\}^B$ there is a partition $\{\pi_1, \ldots, \pi_{B/b}\}$ in the splitter such that $\mathrm{dist}(x_{\pi_1}, y_{\pi_1})$, $\ldots$, $\mathrm{dist}(x_{\pi_{B/b}}, y_{\pi_{B/b}}) \leq \lceil \mathrm{dist}(x,y) b/B \rceil$.

This is similar to the first dimensionality reduction, in that we repeat the remaining algorithm for each choice of partition. This time however we don't get a lower bound on contraction. Instead we get the bound $\mathrm{dist}(x_{\pi_1}, y_{\pi_1}) + \cdots + \mathrm{dist}(x_{\pi_{B/b}}, y_{\pi_{B/b}}) = \mathrm{dist}(x,y)$ which turns out to be enough.

Making a filter system. Consider any two points $x, y \in \{0,1\}^b$ with distance $\leq \delta b$. If we choose $a \in \{0,1\}^b$ uniformly at random, by lemma 10 there is probability at least $s^{-2}\exp(\frac{-s^2}{2(1-\delta)})$ that both x and y have distance at most $t = b/2 - s\sqrt{b/4}$ with a. (Note that this bound sharp up to polynomial factors in s when $s = O(b^{1/4})$.) By the union bound over pairs in $\{0,1\}^b$, we can take A to be $m = (b \log 4)s^2 \exp(\frac{s^2}{2(1-\delta)})$ to get constant probability that A works for any pair.

By repeatedly sampling random A and verifying them in time $e^{O(b)}$, or by formulating a set cover instance and solving it with the greedy algorithm, we get a working A. The value of s will be determined in the next step.

Tensoring. For any pair of points $x, y \in \{0,1\}^B$ with distance most Sr, we know that there is a splitter $\{\pi_1, \ldots, \pi_{B/b}\}$ such that $\mathrm{dist}(x_{\pi_1}, y_{\pi_1}), \ldots, \mathrm{dist}(x_{\pi_{B/b}}, y_{\pi_{B/b}})$ are all bounded by $\lceil Srb/B \rceil$. Setting $\delta = \lceil Srb/B \rceil / b \geq Sr/B$ in the construction of the filters, we get that for all $1 \leq i \leq B/b$, the sets

$$\{j \in [m] : \mathrm{dist}(A_j, x_{\pi_i}) \leq t\} \cap \{j \in [m] : \mathrm{dist}(A_j, y_{\pi_i}) \leq t\},$$

are non-empty. Hence the Cartesian product, (1), likewise contains at least one value. This shows that the algorithm is correct: For two close points, it always finds a shared key.

Write $K(x)$ for the set of keys generated at (1) for some dimensionality reduction and splitter. (Note this is slightly different from keys(x), since $K(x)$ only includes points from this particular reduction and splitter.) First evaluating A B times, takes time Bmb. If we shortcut evaluation if a subset is empty, the product can be evaluated in time proportional to its size, which we write in expectation as $E|K(x)|$.

The space usage and construction time of the algorithm can then be written as $n \cdot d \cdot B^b \cdot (Bmb + E|K(x)|)$ except the construction time has an additional $e^{O(b)}$ for creating A. Similarly the expected query time is

$$d\,B^b \left(mB + E|K(x)| + \sum_{y,\,\mathrm{dist}(x,y)\geq(1-\epsilon)cr} E|K(x) \cap K(y)| \right). \tag{2}$$

To ease the analysis, we say that before calculating each set of (1), the coordinates of each x_{π_i} are xor-ed with a random string $\in \{0,1\}^b$. Then $|K(x)|$ is a random variable independent of the choice of A, and the events $[\mathrm{dist}(A_j, x_{\pi_i}) \leq t]$ are independent since x_{π_i} is uniformly random in $\{0,1\}^b$. We can then use the upper bound on the size of a hamming ball from lemma 9 to get

$$\begin{aligned} E|K(x)| &= E \prod_{i=1}^{B/b} \sum_{j=1}^{m} [\mathrm{dist}(A_j, x_{\pi_i}) \leq t] \\ &= (m \Pr[\mathrm{dist}(A_1, x_{\pi_1}) \leq t])^{B/b} \\ &\leq (O(me^{-s^2/2}))^{B/b}. \end{aligned}$$

Similarly for two points with relative distance at least $(1-\epsilon)c\delta$, let r_i be $\mathrm{dist}(x_{\pi_i}, y_{\pi_i})$. Then with the help of lemma 10 we get

$$\begin{aligned} E|K(x) \cap K(y)| &= E \prod_{i=1}^{B/b} \sum_{j=1}^{m} [\mathrm{dist}(A_j, x_{\pi_i}) \leq t \wedge \mathrm{dist}(A_j, y_{\pi_i}) \leq t] \\ &= \prod_{i=1}^{B/b} m \Pr[\mathrm{dist}(A_j, x_{\pi_i}) \leq t \wedge \mathrm{dist}(A_j, y_{\pi_i}) \leq t] \\ &\leq \prod_{i=1}^{B/b} m \exp\left(\frac{-s^2}{2(1-r_i/b)} \right) \\ &= m^{B/b} \exp\left(\frac{-s^2}{2} \sum_{i=1}^{B/b} \frac{1}{1-r_i/b} \right) \\ &\leq m^{B/b} \exp\left(\frac{-s^2}{2} \frac{B/b}{1-(1-\epsilon)c\delta} \right). \end{aligned}$$

The last inequality follows from Jensen's inequality since $x \mapsto 1/(1-x)$ is convex and $\sum_i r_i \geq (1-\epsilon)c\delta$ whp. The far points that are now closer than $(1-\epsilon)c\delta)$ only contribute a constant factor, so we haven't included them in the calculation.

What remains is choosing the right value for s. Setting $\frac{s^2}{2}\frac{B}{b} = \log n(\frac{1}{(1-\epsilon)c\delta} - 1)$ balances the

terms of (2). (So $s = (\log n)^{1/10}\frac{1-(1-\epsilon)c\delta}{2(1-\epsilon)c\delta} = O((\log n)^{1/8})$ as needed.) We then have

$$\begin{aligned}(2) &\le d\,B^b\,(Bbs)^{O(B/b)}\left(n^{\left(\frac{1}{(1-\epsilon)c\delta}-1\right)\frac{1}{1-\delta}\frac{b}{B}} + n^{\left(\frac{1}{(1-\epsilon)c\delta}-1\right)\left(\frac{1}{1-\delta}-1\right)} + n^{\left(\frac{1}{(1-\epsilon)c\delta}-1\right)\left(\frac{1}{1-\delta}-\frac{1}{(1-\epsilon)c\delta}\right)+1}\right)\\ &= d\,e^{O((\log n)^{4/5}\log\log n)}\left(n^{\frac{1-(1-\epsilon)c\delta}{(1-\epsilon)c(1-\delta)}\frac{b}{\delta B}} + n^{\frac{1-(1-\epsilon)c\delta}{(1-\epsilon)c(1-\delta)}}\right).\\ &= d\,e^{O((\log n)^{4/5}\log\log n)}\left(n^{O((\log n)^{-4/5}} + n^{\frac{1-c\delta}{c(1-\delta)}+O((\log n)^{-1/5})}\right).\\ &= d\,n^{\frac{1-c\delta}{c(1-\delta)}+O((\log n)^{-1/5}\log\log n)}.\end{aligned}$$

Here we used the expansion $\frac{1-(1-\epsilon)c\delta}{(1-\epsilon)c(1-\delta)} = \left(\frac{1-c\delta}{c(1-\delta)} + \frac{\epsilon\delta}{1-\delta}\right)\frac{1}{1-\epsilon} = \frac{1-c\delta}{c(1-\delta)} + O(\epsilon)$ and that from lemma 1, we know $\delta = \Theta(\epsilon/c)$, so the $\frac{b}{\delta B}$ term doesn't dominate.

Since we have successfully guaranteed the algorithm never produces false negatives, and our query time can be written compactly as $dn^{1/c+o(1)}$, we have shown that the O'Donnell et al. lower bound [O'Donnell et al., 2014] is achievable with a Las Vegas data structure.

3 Set Similarity Data Structure

The structure of the algorithm is the same LSF construction as the previous section. The difference is in the keys(x) function, which in the previous algorithm was the close points of a certain code. For set similarity, we instead have keys(x) be the r-element blocks of a Turán system contained in x:

Definition 3 (Turán system [Turán, 1961, Colbourn and Dinitz, 2006])**.** *A Turán (n,k,r)-system is a collection of r-element blocks of an n element set X such that every k element subset of X contains at least one of the blocks.*

This choice is inspired by [Pagh and Christiani, 2017] who use an interesting probabilistic system. However there are no previously known constructions of deterministic systems, which we need to avoid false negatives. Much less constructions that are efficiently decodable. The algorithm below provides one such construction.

One technicality is that we will assume the maximum weight of the query point $\|q\|$ and any data point $\|x\|$, $t = \max(\|q\|, \|x\|)$ is known in advance. This can be accomplished by making $(d+1)^2$ data structures, saving each point in $d+1$ of them and querying $d+1$ of them for each query.

If we are trying to solve the (b, b_2)-approximate similarity search problem with $\ge b$ the Braun-Blanquet similarity of 'close points' and $\le b_2$ the similarity of far points, we have for a particular t that the inner product between the query and the data points we are interested in is at least bt. The inner product between the query and the points we are not interested in is at most $b_2 t$.

Some of the techniques used in this section are similar to those from the previous section. However this time we have to worry about reducing three values d, t and bt, rather than just d and r. Also we can no longer use the non-expanding dimensionality reduction, since it might decrease the size of bt. Instead we will use a combination of partitioning and a type of balanced perfect hashing functions. We also use the splitter construction with quite different parameters than in the previous section, showing its generality.

Again we first describe the steps in a light way, and then go into more detail.

1. Shuffle the coordinates of x, and unless d is already small, use partitioning to split x in blocks of size $C = \frac{d\log n}{bt\epsilon^2}$. Here ϵ will be approximately $(\log n)^{-1/4}$. By averaging, any subset of bt will have at least $bt/(d/C) = \epsilon^{-2}\log n$ elements in one of the blocks.

2. On each block, use a perfect hash family, $\mathcal{H}$, to reduce the dimension further to $B = (\frac{\log n}{\epsilon^2})^2$. By the guarantees of such a family, any set S of size at most $\epsilon^{-2}\log n$, there is a function $h \in \mathcal{H}$ such that $|h(S)| = |S|$.

3. On each length B vector from the previous step, use a splitter to partition the coordinates into k sub blocks of size $D = B/k$. Here k is approximately $\sqrt{\log n}$, and the size of the splitter is at most B^k. The splitter guarantees that for any set of m elements, there is a partition in which each part gets $\lfloor m/k \rfloor$.

4. Build a $(D, \frac{\log n}{k\epsilon^2}, r)$-Turán design, $\mathcal{T}$, such that for any subset $S \subseteq [D]$ of size at most $|S| \leq \frac{\log n}{k\epsilon^2}$, there is a set $R \in \mathcal{T}$ such that $R \subseteq S$. Such a design can be made of size approximately $(\frac{\log n}{\epsilon^2})^r$ and time $D^{\frac{\log n}{k\epsilon^2}}$, where we will choose $r \approx \sqrt{\log n}$.

5. For each partition $\{\pi_1, \ldots, \pi_k \subseteq [B]\}$ in the splitter, we return every value of the product

$$\{R \in \mathcal{T} : R \subseteq x_{\pi_1}\} \times \cdots \times \{R \in \mathcal{T} : R \subseteq x_{\pi_k}\} \tag{3}$$

Partitioning. When constructing the data structure we pick a random permutation, which we apply to each vector x as the first step of evaluating keys(x). This of course doesn't change the problem much, but it makes it impossible for an adversary to make a bad set of points which share keys more often than expected given their similarity, as we will see.

We then partition in the d coordinates arbitrarily in one or more blocks of size $C = \frac{d\log n}{bt\epsilon^2}$. We can think of ϵ as $O((\log n)^{-1/4})$, so we may adjust it by a constant factor, if necessary, to make d/C an integer. Intuitively the partitioning has the effect of reducing the value of t. In average we expect each part of x to have weight concentrated around $\frac{\|x\|\log n}{bt\epsilon^2}$.

By averaging note, that for any subset of x of size bt, there must be a block with $\epsilon^{-2}\log n$ of the elements. Hence if two points x, y have inner product at least bt, then their projections in one of the blocks must have inner product $\epsilon^{-2}\log n$.

Perfect hashing. The previous step allowed us to assume all points have reasonably low weight, however the dimension or 'universe', C, may still be very large (e.g. $n^{(\log n)^{-1/100}}$). To solve this problem, we use a $(C, B, \epsilon^{-2}\log n)$-perfect hash family, $\mathcal{H}$, where $B = (\epsilon^{-2}\log n)^2$. This is a set of functions $h : [C] \to [B]$, such that for any set $S \subseteq [C]$ of size at most $\epsilon^{-2}\log n$, there is a function $h \in \mathcal{H}$ with $|h(S)| = |S|$. We use the construction from lemma 3 of size $O(B\log C)^3$, which further guarantees that $|h^{-1}(i)| \leq C/B + 2$.

We use this to map x to $O(B\log C)^3$ dimension B sub-blocks, $x_{h_1}, x_{h_2}, \cdots \in \{0,1\}^B$ by setting $x_{h_i,j} = [\{l \in [C] : x_l = 1 \text{ and } h_i(l) = j\}]$. In other words, $x_{h_i,j}$ represents whether any 1-entry of x gets mapped by h_i to j.

It is useful to consider the probability that the jth entry of a sub-block x_{h_i} is 1. This happens exactly when the shuffling from the previous step places one or more 1s in $h_i^{-1}(j) \subseteq [C]$. This is hyper geometrically distributed where we pick $|h_i^{-1}(j)|$ values from $[d]$ without replacement, of which t are successes and the rest are failures. By Markov's inequality the probability that we get one or more successes is bounded by the expectation, which is less than $\frac{|h_i^{-1}(j)|\|x\|}{d} \leq \frac{C\|x\|}{Bd} + \frac{2\|x\|}{d}$.

Splitting. Even after partitioning and perfect hashing, the universes and vectors weights are still too large. By lemma 2 we can construct a set of B^k partitions, $\{\pi_1, \ldots, \pi_k \subseteq [B]\}, \ldots$ of $[B]$ into size $D = B/k$ subsets, such that for any set of m elements, there is a partition in which each part gets $\lfloor m/k \rfloor$.

In particular, for any pair $x, y \in \{0,1\}^B$ with inner product at least $\epsilon^{-2}\log n$, there is a partition $\{\pi_1, \ldots, \pi_k\}$, such that $|x_{\pi_1} \cap y_{\pi_1}|, \cdots \geq \lfloor |x \cap y|/k \rfloor$.

Intuitively, when the original partitioning split our vectors in blocks, and the hashing created sub-blocks, the guarantee was always that 'in one of the blocks' we get lucky. This time the guarantee is a bit different, since our sub-sub-blocks come in groups of size k, and we guarantee that in one of the groups *all* the sub-sub-blocks are lucky. Perhaps this also explains why the splitter needs to be so large, compared to the number of partitions and the size of the perfect hash family.

We will choose $k = \sqrt{\log n}\frac{(\log\log n)^2}{\log 1/b_2}$, which is just about $B^{1/6}$.

Building a Turán system. We want to construct a Turán $(D, \frac{\log n}{\epsilon^2 k}, r)$-system, where we set $r = \frac{\sqrt{\log n}}{(\log\log n)^2}$. For our choice of parameters, this can be done efficiently using the probabilistic method: Let $\mathcal{T}$ be the result of m times, independently choosing a random size r set $R \subseteq [D]$. Now for any set $S \subseteq [D]$ of size $|S| = \frac{\log n}{\epsilon^2 k}$, the probability that there is no $R \in \mathcal{T}$ with $R \subseteq S$ is exactly

$$\left(1 - \binom{|S|}{r}\Big/\binom{D}{r}\right)^m \leq \exp\left(-m\binom{|S|}{r}\Big/\binom{D}{r}\right).$$

If we take $m = (1 + |S|\log D)\binom{D}{r}/\binom{|S|}{r}$ we get by the union bound over all $\binom{D}{|S|}$ size $|S|$ subsets, that our construction succeeds with probability at least $1 - e^{-1} \geq 1/2$. Note that by lemma 8, we can bound m by

$$m \leq (D/|S|)^r e^{r^2/|S|}(1 + |S|\log D) = (\epsilon^{-2}\log n)^r e^{\frac{r^2\epsilon^2 k}{\log n} + O(\log\log n)}$$

After sampling $\mathcal{T}$, we can verify that it is indeed a Turán system by enumerating all possible size $|S|$ sets, and looking for a subset R in $\mathcal{T}$. This takes time $\binom{D}{|S|}m$, and since we succeed with probability 1/2, doing this twice will suffice in expectation. Alternatively we can make $\mathcal{T}$ without using any randomness, by formulating a large set-cover instance and solving it using the greedy algorithm. This creates an instance of similar size with similar time usage.

For a given set $S \subseteq [D]$, we can find all sets $R \in \mathcal{T}$ such that $R \subseteq S$ in time $|\mathcal{T}|D$ by simple enumeration. In some cases it may be more efficient to enumerate all size r subsets of S and look them up in a hash-table representing $\mathcal{T}$, however this is slightly harder to analyze, and the decoding time will be dominated by other terms in the analysis anyway.

Tensoring. By the properties of the previous steps 1-2, we are guaranteed that $x, y \in \{0,1\}^d$ have inner product at least bt, there is a sub-block into which the projections of x and y, $x', y' \in \{0,1\}^B$ have inner product $\epsilon^{-2}\log n$, and so by step 3 there is a partition, $\{\pi_1, \ldots, \pi_k\}$ in which every projection $|x'_{\pi_1} \cap y'_{\pi_1}| \ldots |x'_{\pi_k} \cap y'_{\pi_k}| \geq \epsilon^{-2}k^{-1}\log n$. The Turán system then guarantees that each set of the product

$$\{R \in \mathcal{T} : R \subseteq x_{\pi_1}\} \times \cdots \times \{R \in \mathcal{T} : R \subseteq x_{\pi_k}\} \tag{4}$$

is non-empty, which shows our algorithm will always find a shared key for x and y.

Just as for the hamming space data structure, we need to analyze the expected number of keys for an arbitrary x, and the number of shared keys with points that have inner product less than $b_2 t$ with x. For this we need to know the probability and dependencies of events $[R_1 \subseteq x'_{\pi_1} \wedge \ldots, R_k \subseteq x'_{\pi_k}]$, for arbitrary size r sets $R_1, \ldots, R_k$, arbitrary x and arbitrary π.

We observe that the event is equal to requiring rk particular bits of x' to be 1. By the analysis of step 2, the probability that a particular bit is 1 is bounded by $\frac{C\|x\|}{Bd}$. The individual coordinates are dependent, but with negative correlation, since if one coordinate is 1, that means one or more 1 of x were hashed to this coordinate, and because of sampling without repetition, there are fewer 1s to be hashed to the remaining coordinates.

With these two observations, we can bound the size of (4):

$$\begin{aligned}
E\prod_{i=1}^{k}\sum_{R\in\mathcal{T}}[R\subseteq x_{\pi_i}] &= \sum_{R_1,\ldots,R_k\in\mathcal{T}}\prod_{i=1}^{k}\Pr[R_1\subseteq x_{\pi_1}\wedge\cdots\wedge R_k\subseteq x_{\pi_k}]\\
&\le |\mathcal{T}|^k\left(\frac{C\|x\|}{Bd}\right)^{rk}\\
&\le |\mathcal{T}|^k\left(\frac{Ct}{Bd}\right)^{rk}\\
&\le \left(\frac{\log n}{\epsilon^2}\right)^{rk}e^{\frac{(rk\epsilon)^2}{\log n}}\left(\frac{\epsilon^2}{b\log n}\right)^{rk}\\
&\le (\tfrac{1}{b})^{rk}e^{\frac{(rk\epsilon)^2}{\log n}}.
\end{aligned}$$

Note that we bounded $\|x\|$ by t, so if the query is smaller than the data point, or vice versa, we actually get somewhat better performance than we guarantee. However in general $\|x\|$ can be as large as t. The analysis of shared keys is similar:

$$\begin{aligned}
E\prod_{i=1}^{k}\sum_{R\in\mathcal{T}}[R\subseteq x_{\pi_i}\wedge R\subseteq y_{\pi_i}] &= \prod_{i=1}^{k}|T|\Pr[R\subseteq(x\cap y)_{\pi_i}]\\
&\le |T|^k\left(\frac{Cb_2t}{Bd}\right)^{rk}\\
&= (\tfrac{b_2}{b})^{rk}e^{\frac{(rk\epsilon)^2}{\log n}}.
\end{aligned}$$

We can finally write the entire expected running time of the algorithm. For each of $(d/C)(B\log C)^3B^k$ projections and partitions, we spend $kD|\mathcal{T}|$ time decoding the Turán system k times; we build the product (4) in expected time proportional to its size $(\frac{1}{b})^{rk}e^{\frac{(rk\epsilon)^2}{\log n}}$; and we look in a hash table bucket for each key, calculating the distance to $n(\frac{b_2}{b})^{rk}e^{\frac{(rk\epsilon)^2}{\log n}}$ points with low similarity in expectation. Note that if we encounter a single high similarity point, we can immediately return it, so these points don't cost us anything.

Putting it all together we have

$$\text{work} = \lceil d/C\rceil(B\log C)^3B^k\left(kB|\mathcal{T}| + (1/b)^{rk} + n(b_2/b)^{rk}\right)e^{\frac{(rk\epsilon)^2}{\log n}} \tag{5}$$

With our choices $k=\sqrt{\log n}(\log\log n)^2/(\log 1/b_2)$, $r=\sqrt{\log n}/(\log\log n)^2$ and $\epsilon^2=(\log\log n)^3(\log 1/b_2)/\sqrt{\log n}$, we get $rk=\log n/(\log 1/b_2)$, which balances the terms $(1/b)^{rk}$ and $n(b_2/b)^{rk}$, and so (5) is bounded by

$$\begin{aligned}
\text{work} &\le bt\left(kB^{r/2+1}+2n^{\rho}\right)e^{\frac{\epsilon^2\log n}{(\log 1/b_2)^2}+O(k(2\log 1/\epsilon+\log\log n))}\\
&= bt\,n^{\rho+O(\frac{(\log\log n)^3}{\sqrt{\log n}\log 1/b_2})}
\end{aligned}$$

where for $\rho = \frac{\log 1/b}{\log 1/b_2}$. During construction of the data structure, we also have to build the Turán system, which by the discussion above can be done in time bounded by

$$\begin{aligned} & D^{\frac{\log n}{\epsilon^2 k}} (\epsilon^{-2} \log n)^r e^{\frac{r^2\epsilon^2 k}{\log n} + O(\log\log n)} \\ &= e^{(\log\log n)^{-5} \log D + O(r \log\log n) + O(\log\log n)} \\ &= e^{O(\sqrt{\log n})}, \end{aligned}$$

and so it is negligible compared to computing the keys for n points, which takes time at least $n^{1+\rho}$.

By the reduction mentioned in the beginning of the section, we make a data structure like the above for each of $(d+1)^2$ values for $\|x\|$ and $\|q\|$. Hence the final construction takes space $O(d^2\, n^{\rho+o(1)})$ and has query time $O(d^2\, n^{\rho+o(1)})$, which is the theorem.

We note that the theorem doesn't hold if b_2 is allowed to be very close to 1. In particular we need $b_2 = 1 - \hat{o}(1/\sqrt{\log n})$ for it not to make the error term $\hat{O}(1/(\sqrt{\log n} \log 1/b_2))$ problematic.

4 Important constructions

Our first construction is of a dimensionality reduction that doesn't increase distances.

Lemma 1. *For $\epsilon = \epsilon(n) > 0$ and $r \geq 1$, there is a set of at most d functions $f_i : \{0,1\}^d \to \{0,1\}^{O((\log n)/\epsilon^3)}$ with the two properties for every $x, y \in \{0,1\}^d$:*

1. *If $\operatorname{dist}(x,y) \geq r$, for every i and with probability $O(\sqrt{m}/n)$, the mapping is ϵ-non-contractive:*

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

2. *For at least one i, with 100% probability, the mapping is strictly non-expansive:*

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

The values of S and m are $S = \Theta(\frac{\log n}{\epsilon^2 r})$ and $m = O(\frac{r}{\epsilon})$, such that $S \operatorname{dist}(x,y)$ divided by the output dimension of f is $\Theta(\epsilon \operatorname{dist}(x,y)/r)$.

Proof. Pick a random function $h : [d] \to [m]$ and let $f'(x)_i = \oplus\{x_j \mid h(j) = i\}$. That is, we throw each of the $[d]$ coordinates in one of m buckets, uniformly and independently. The distance $\operatorname{dist}(f'(x), f'(y))$ then equals $\|f(x \oplus y)\|$ which is the number of buckets that contain an odd number of coordinates on which x and y differ.

Hence, we will assume we are throwing $r = \operatorname{dist}(x,y)$ balls in m buckets and analyze the concentration around r/m. We can analyze this using a folklore transformation between the multinomial distribution and the Poisson distribution (see e.g. [Mitzenmacher and Upfal, 2005]), which says that we can assume each bucket is Poisson distributed with $\lambda = r/m$, for the small cost of a factor $e\sqrt{m}$ in the probability bounds.

The probability a Poisson distributed value is odd is $p = (1 - \exp(-2\lambda))/2 \geq \lambda - \lambda^2$. We can see this from its probability generating function $G(z) = e^{\lambda(z-1)} = \sum_k \Pr[X = k] z^k$. With that we can write $\sum_{k \text{ odd}} \Pr[X = k] = \sum_k \Pr[X = k](1 - (-1)^k)/2 = (G(1) - G(-1))/2$, which is p. The $\lambda - \lambda^2$ lower bound follows from the bound $e^{-x} \leq 1 - x + x^2/2$ when $x \geq 0$.

In the construction, we will have $m = r/\epsilon$. If $r < 2(\log n)/\epsilon^2$ we will make $S = 2(\log n)/(r\epsilon^2)$ copies of f and concatenate the result. If $r > 2(\log n)/\epsilon^2$ we only make one f, but partition in $1/S$ blocks, each of which gets Sm coordinates. (Note that we can adjust ϵ by a constant factor to make

m integer and S on the form k or $1/k$ for some integer k.) In either case we get the guarantee that in one of the blocks $X = \text{dist}(f(x), f(y)) \leq Sr$, while X is either binomially or hyper-geometrically distributed with mean Sr.

We can use a Hoeffding bound (which also holds for hyper-geometric random variables):

$$\Pr[X \leq (p-\varepsilon)Sm] \leq \exp(-D(p+\varepsilon \mid p)Sm),$$

where $D(\alpha \mid \beta) = \alpha \log \frac{\alpha}{\beta} + (1-\alpha)\log \frac{1-\alpha}{1-\beta}$. We let $\varepsilon = \lambda^2$ and note $D(\lambda \mid \lambda - \lambda^2) \geq \lambda^3/2$ to get $\Pr[X \leq (1-2\lambda)Sr] \leq \exp(-\lambda^3 Sm/2)$, which for our choice of S and m is equivalent to

$$\Pr[X \leq (1-2\epsilon)Sr] \leq \exp(-\epsilon^3 Sm/2) = 1/n,$$

which is the lemma. □

Lemma 2 (Balanced (n,k)-Splitter)**.** *For any integers n and k, there exists a set of most n^k partitions $\{s_1, \ldots, s_k\}$ of $[n]$, such that for any $x \in \{0,1\}^n$, the following properties hold:*

$$\lfloor n/k \rfloor \leq |s_1|, \ldots, |s_k| \leq \lceil n/k \rceil \tag{6}$$

$$\lfloor \|x\|/k \rfloor \leq \|x_{s_1}\|, \ldots, \|x_{s_k}\| \leq \lceil \|x\|/k \rceil \tag{7}$$

When $n = mk$ for some integer m, we can show that the exact size of the splitter family (after removing symmetries) is $\frac{1}{k}\binom{mk}{k-1}$ and can be constructed in time proportional to its size. For $m = 2$ these are the Catalan numbers, and in general the formula follows from a simple bijection with m-ary trees with k nodes: For each interval as created in the proof, starting with the last one inserted, we can make that interval the parent node of the intervals it touches. This also immediately gives an efficient way to generate the partition: Starting again backwards, insert an interval with up to $k-1$ holes of a size dividable by m. Recurse in each hole.

It is interesting to compare the construction with that of [Naor et al., 1995] which considers all ways to split mk in k consecutive intervals, not necessarily of equal length. This construction has size $\binom{mk}{k-1}$. The construction of [Alon et al., 2006] can give segments of nearly equal size, if we reduce the problem to the more sophisticated 2-way splitter problem. This construction has size $k^{2(k-1)}\binom{mk}{2(k-1)}$, though the analysis may not be optimal for our use case.

Proof. First notice the following continuity property: For any $x \in \{0,1\}^n$, consider the norm of x restricted to the cyclic intervals of length l: $\|x_{[1,l]}\|, \|x_{[2,l+2]}\|, \ldots, \|x_{[n,l-1]}\|$. The norm of these sum to $l\|x\|$, hence there must be integers i and j, such that $\|x_{[i,i+l]}\| \leq \|x\|l/n \leq \|x_{[j,j+l]}\|$. Now consider moving $[i, i+l]$ step by step until it meets $[j, j+l]$. Each move can only change the norm by 0, 1 or -1, hence there is some position i' at which $\|x_{[i',i'+l]}\| \leq \|x\|l/n \leq \|x_{[i'+1,i'+1+l]}\|$ or equivalently $\|x_{[i',i'+l]}\| = \lfloor \|x\|l/n \rfloor$ and $\|x_{[i'+1,i'+1+l]}\| = \lceil \|x\|l/n \rceil$.

We prove the lemma by induction in k on an index set $S \subseteq [n]$. For $k = 1$ we simply take all of S. For $k > 1$ take $l = \lceil |S|/k \rceil$ and find a segment s with $\|x_s\| = \lceil \|x_S\|/k \rceil$. We can do this with the continuity property because we can always find a segment with norm $\lfloor \|x_S\|l/|S| \rfloor$ and one with $\lceil \|x_S\|l/|S| \rceil$. By lemma 5, (where $|S| \geq \|x_S\|$) one of these have to equal $\lceil \|x_S\|/k \rceil$.

For the remaining $S \setminus s$ coordinates we invoke the induction hypothesis on $x_{S \setminus s}$ and $k-1$. In

order to use the combine the results we need

$$\begin{aligned}
&\{\lceil (n-l)/(k-1)\rceil, \lfloor (n-l)/(k-1)\rfloor\} \\
&\qquad = \{\lceil (n-\lceil n/k\rceil)/(k-1)\rceil, \lfloor (n-\lceil n/k\rceil)/(k-1)\rfloor\} \\
&\qquad \subseteq \{\lceil n/k\rceil, \lfloor n/k\rfloor\} \quad \text{and} \\
&\{\lceil \|x_{S\setminus s}\|/(k-1)\rceil, \lfloor \|x_{S\setminus s}\|/(k-1)\rfloor\} \\
&\qquad = \{\lceil (\|x_S\| - \lceil \|x_S\|/k\rceil)/(k-1)\rceil, \rfloor (\|x_S\| - \lceil \|x_S\|/k\rceil)/(k-1)\rfloor\} \\
&\qquad \subseteq \{\lceil \|x_S\|/k\rceil, \lfloor \|x_S\|/k\rfloor\}.
\end{aligned}$$

These both follow from lemma 4.

Since each of the k intervals can start at one of n positions, we get the bound on the number of different partitions. □

The following lemma is basically [Hagerup and Tholey, 2001] lemma 2 or [Fredman et al., 1984] lemma 2.

Lemma 3. *Let n and k be integers with $0 < k < n$. The family of functions*

$$\mathcal{H} = \{x \mapsto ((ax+b) \bmod p) \bmod k^2 \mid k^2 \log n \le p \le 2k^2 \log n,\ 0 < a < p,\ 0 \le b < p\},$$

where p ranges over primes, is a perfect (n, k^2, k)-hash family. The family is 'balanced' in the sense that for every $h \in \mathcal{H}$ and $i \in [k^2]$ we have $|h^{-1}(i)| \le n/k^2 + 2$. The size of the family is $O(k^6(\log n)^3)$.

The author is thankful to Rasmus Pagh for noting that the following family has a crucial balanced property. (Note that the term 'balanced' is sometimes used in the different meaning, that elements get mapped to each bucket approximately equally often.) An alternative construction is shown in [Alon et al., 2006], which gives slightly smaller families, $O(k^4 \log n)$, however it is not clear that these have the balancing properties we need.

We give a short proof the perfectness as well as the balancing.

Proof. We first show, that for any subset $S \subseteq [n]$ of size $|S| = k$ there is some p in the mentioned range, such that $f_p(x) = x \bmod p$ is 1-1 on S. We have $f_p(x) = f_p(y)$ for $x \neq y$ when p divides $|x-y|$. Now consider the product $P = \prod_{i,j \in S} |i-j|$ of all differences between values in S. If p does not divide P then $f_p(x)$ is 1-1 on S as wanted. The maximum number of prime factors is $\log P = \sum_{i,j \in S} \log |i-j| \le k^2 \log n$. By the prime number theorem, the product of the primes below x is $e^{x+o(x)}$, hence there is a prime in the range $[k^2 \log n, 2k^2 \log n]$ which doesn't divide P. By universality of $g(y) = (ay+b) \bmod p \bmod k^2$ for $y < p$ we get that $\mathcal{H}$ is perfect.

Balancing follows because each mod disturbs it by at most 1. □

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 data structures.

We also show that efficient constructions of large Turán Systems exists, even though that implicitly already follows from [Alon et al., 2006].

Open questions

1. Can we make a completely deterministic high dimensional data structure? Or even just one using only $O(\log d)$ random bits, rather than $O(d)$ as in this paper?

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.

3. What other similarity spaces do the techniques from this paper apply to? Can we get space time trade-offs and an algorithm for l_2 with optimal exponent n^{1/c^2}?

4. Can we improve on the error term $(\log n)^{-1/5}$ in the Hamming space data structure? We didn't put a lot of effort into minimizing it, but focuses on achieving $o(1)$. Randomized LSH data structures, such as [Andoni and Indyk, 2006] had a $(\log n)^{-1/3}$ term, but most algorithms have less than $(\log n)^{-1/2}$.

5.1 Acknowledgements

The author would like to thank Rasmus Pagh, Tobias Christiani and the members of the Scalable Similarity Search for many rewarding discussions on derandomization and set similarity data structures. Further thanks to Rasmus Pagh and Johan Sivertsen for useful comments on the manuscript, and to the people at University of Texas, Austin, for hosting me while doing this work.

5.2 Discussion

6 Appendix

The following are a few useful lemmas for working with floors and ceils.

Lemma 4. *For any integers $n \geq 0, k \geq 2$ we have*

$$\{\lceil (n - \lceil n/k \rceil)/(k-1) \rceil, \lfloor (n - \lceil n/k \rceil)/(k-1) \rfloor\} \subseteq \{\lceil n/k \rceil, \lfloor n/k \rfloor\}$$

Proof. Write $n = mk + r$ where $0 \leq r < k$. Then $(n - \lceil n/k \rceil)/(k-1) = m + (r - [r > 0])/(k-1)$ and $n/k = m + r/k$. Thus it suffices to show

$$\{\lceil (r - [r > 0])/(k-1) \rceil, \lfloor (r - [r > 0])/(k-1) \rfloor\} \subseteq \{[r > 0], 0\}.$$

If $r = 0$, then both sides are $\{0\}$. Otherwise we get $\{\lceil (r-1)/(k-1) \rceil, \lfloor (r-1)/(k-1) \rfloor\} \subseteq \{1, 0\}$, which is true since $0 \leq r - 1 \leq k - 1$. □

Lemma 5. *For any three positive integers c, a, b with $c \geq a$ we have $\lceil a/b \rceil \in \{\lfloor a\lceil c/b \rceil / c \rfloor, \lceil a\lceil c/b \rceil / c \rceil\}$.*

Proof. It suffices to show $\lceil a\lceil c/b \rceil / c \rceil \geq \lceil a/b \rceil \geq \lceil a\lceil c/b \rceil / c \rceil - 1$ The first inequality follows from $b\lceil s/b \rceil \geq s$ which implies $c\lceil a/b \rceil / a \geq c/b$. For the second inequality write $c = mb + r$ for $0 \leq r \leq b$, then we want to show $a/b \geq a\lceil (mb+r)/b \rceil / c - 1 = a(m + \lceil r/b \rceil)/c - 1$. It suffices to show $mb + r + cb/a = b(m + \lceil r/b \rceil)$ which follows from $cb/a \geq b \geq b\lceil r/b \rceil$. □

A Embeddings for Multi-sets

We can use embeddings as a way to generalize our results to different similarity measures. Since the host space is discrete, we have to round values to integers. E.g. if we want to work over vectors from $[0, 1]^d$, we might scale by a factor 100 and map from $\{0, \ldots, 100\}^d$ instead, incurring errors of order $1/100$. This way, we can loosely see the maximum value, k, in the following embeddings, as ϵ^{-1} where ϵ is the error we want.

We present two simple embeddings of multi-sets into simple sets, which preserve different generalizations of Jaccard similarity.

Lemma 6. *There is a mapping $f : \{0, \dots, k\}^d \to \{0,1\}^{kd}$, such that $sim(f(x), f(y)) = \frac{\sum_i \min(x_i, y_i)}{\sum_i \max(x_i, y_i)}$.*

Proof. Define $\hat{f}(n) = (\underbrace{1, \dots, 1}_{n \text{ times}}, \underbrace{0, \dots, 0}_{k-n \text{ times}})$ and let f be the concatenation $f(x) = \hat{f}(x_1) \dots \hat{f}(x_d)$.

It's easy to check that we get

$$\begin{aligned} \langle f(x), f(y) \rangle &= \sum_i \langle \hat{f}(x_i), \hat{f}(y_i) \rangle = \sum_i \min(x_i, y_i), \\ |f(x)| &= \sum_i x_i, \\ \operatorname{sim}(f(x), f(y)) &= \frac{\sum_i \min(x_i, y_i)}{\sum_i x_i + \sum_i y_i - \sum_i \min(x_i, y_i)} = \frac{\sum_i \min(x_i, y_i)}{\sum_i \max(x_i, y_i)}. \end{aligned}$$

□

Another generalization of Jaccard similarity is sometimes called Tanimoto distance. This similarity between vectors x and y is $\frac{\langle x,y \rangle}{\|x\|_2^2 + \|y\|_2^2 - \langle x,y \rangle}$. We don't quite have an exact way to embed this, but we can get the following:

Lemma 7. *There is an asymmetric mapping $f, g : \{0, \dots, k\}^d \to \{0,1\}^{k^2 d}$, such that $sim(f(x), g(y)) = \frac{\langle x,y \rangle}{k\|x\|_1 + k\|y\|_1 - \langle x,y \rangle}$.*

Proof. Define

$$\hat{f}(n) = \begin{pmatrix} \overbrace{1 \ \dots \ 1}^{n} & \overbrace{0 \ \dots \ 0}^{k-n} \\ \vdots \quad\quad \vdots & \vdots \quad\quad \vdots \\ 1 \ \dots \ 1 & 0 \ \dots \ 0 \end{pmatrix} \Big\} k, \quad \hat{g}(n) = \begin{pmatrix} \overbrace{1 \ \dots \ 1}^{k} \\ \vdots \quad\quad \vdots \\ 1 \ \dots \ 1 \\ 0 \ \dots \ 0 \\ \vdots \quad\quad \vdots \\ 0 \ \dots \ 0 \end{pmatrix} \begin{matrix} \Big\} n \\ \\ \\ \Big\} k-n \end{matrix},$$

where the matrices are read as vectors in some order.

Here $|\hat{f}(n)| = |\hat{g}(n)| = kn$ and $\langle \hat{f}(n), \hat{g}(m) \rangle = nm$. Thus we can define f and g by concatenation, $f(x) = \hat{f}(x_1) \cdots \hat{f}(x_d)$, $g(x) = \hat{g}(x_1) \cdots \hat{g}(x_d)$.

This gives us

$$\begin{aligned} \langle f(x), g(y) \rangle &= \langle x, y \rangle, \\ |f(x)| &= |g(x)| = k \sum x_i, \\ \operatorname{sim}(f(x), g(y)) &= \frac{\langle x, y \rangle}{k\|x\|_1 + k\|y\|_1 - \langle x, y \rangle}. \end{aligned}$$

□

B The Ratio of Two Binomial Coefficients

Classical bounds for the binomial coefficient: $(n/k)^k \le \binom{n}{k} \le (en/k)^k$ give us simple bounds for binomial ratios, when $n \ge m$: $(n/em)^k \le \binom{n}{k}\Big/\binom{m}{k} \le (en/m)^k$. 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 $n \ge m$, we can show $\binom{n}{k}\Big/\binom{m}{k} = \prod_{i=0}^{k-1} \frac{n-i}{m-i} \ge \prod_{i=0}^{k-1} \frac{n}{m} = \left(\frac{n}{m}\right)^k$. 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 8. *For $n \ge m \ge k \ge 0$ we have the following bounds:*

$$\left(\frac{n}{m}\right)^k \le \binom{n}{k}\Big/\binom{m}{k} \le \frac{\exp\left(n\,\mathrm{H}(k/n)\right)}{\exp\left(m\,\mathrm{H}(k/m)\right)} \le \left(\frac{n}{m}\right)^k e^{k^2/m}$$

If $m \ge n$ we can simply flip the inequalities and swap n for m. Note that $(n/em)^k \le (n/m)^k$ and $e^{k^2/m} \le e^k$, so the bounds are strictly sharper than the simple bounds stated above.

Especially the entropy bound is quite sharp, since we can also show: $\frac{\exp((n+1)\,\mathrm{H}(k/(n+1)))}{\exp((m+1)\,\mathrm{H}(k/(m+1)))} \le \binom{n}{k}\Big/\binom{m}{k}$. However it is unwieldy, and we mostly show it as a way to show the $e^{k^2/m}$ upper bound.

For the proofs, we'll use some inequalities on the logarithmic function from [Topsok, 2006], which are valid for $x \ge 0$:

$$2x/(2+x) \le \log(1+x) \le x(2+x)/(2+2x). \tag{8}$$

In particular we apply (8) to bound the the entropy function:

$$\mathrm{H}(x) \le x \log 1/x + x(2-x)/2 \tag{9}$$

$$\mathrm{H}(x) \ge x \log 1/x + 2x(1-x)/(2-x), \tag{10}$$

which is quite tight for small x.

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

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

The integral bound holds because $\log \frac{n-i}{m-i}$ is increasing in i, and so $\log \frac{n-i}{m-i} \le \int_i^{i+1} \log \frac{n-x}{m-x} dx$. 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 lemma, follows similarly from integration, using $\log \frac{n-i}{m-i} \ge \int_{i-1}^{i} \log \frac{n-x}{m-x} dx$.

For the final upper bound, we use the bounds (9) and (10) on $\mathrm{H}(k/n)$ and $\mathrm{H}(k/m)$ respectively:

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

□

C The Volume of the Intersection of two Hamming Balls

A d-dimensional "Hamming ball" is a set of points in $\{0,1\}^d$, which contains all points within some radius r from some center point x under the hamming metric. Recall that the hamming distance, $\mathrm{dist}(x,y)$, between $x, y \in \{0,1\}^d$ is the number of coordinates on which the two points differ. Specifically we define the d-dimensional t-ball centered around x:

$$\mathrm{B}_d(x,t) = \{p \in \{0,1\}^d : \mathrm{dist}(x,p) \le t\}. \tag{11}$$

The volume of such balls is easily described as a sum of binomial coefficients, but we get get the following more useful bound due to [Cramér, 1938]:

Lemma 9 (Cramér bound on Hamming balls)**.** *Let $t = d/2 - s\sqrt{d/4}$ where $s = O(\sqrt{d}) \cap \omega(1)$. Then we get the following asymptotic expansion*

$$B_d(0,t) = \sum_{s=0}^{t} \binom{d}{r} = \frac{1}{\sqrt{2\pi s}} \exp\left(d\,\mathrm{H}(t/d)\right)\left(1 + O\left(\tfrac{1}{s^2} + \tfrac{s}{\sqrt{d}}\right)\right),$$

where $\mathrm{H}(x) = x\log 1/x + (1-x)\log 1/(1-x)$ is the binary entropy function as in the previous section. Further, if $s = O(d^{1/4})$ we get the Gaussian looking expansion:

$$B_d(0,t) = \frac{1}{\sqrt{2\pi s}} 2^d \exp(-s^2/2)\left(1 + O\left(\tfrac{1}{s^2} + \tfrac{s^4}{d}\right)\right).$$

In this section we will prove an asymptotic bound for the size of the intersection between two Hamming balls, which is in the spirit of the Cramér bound.

Lemma 10. *For $t = \frac{d}{2} - \frac{s\sqrt{d}}{2}$, $1 \le s \le \sqrt{d}/2$ and $r < d/2$, let $I = |B_d(x,t) \cap B_d(y,t)|$ be the volume of the intersection between two t-balls at distance r. Then*

$$I \ge \Omega\left(\frac{1}{s^2} \exp\left[(d-r)\,\mathrm{H}\left(\frac{1}{2} - \frac{s/\sqrt{d}}{2(1-r/d)}\right) + r\log 2\right]\right) \tag{12}$$

$$I \le O\left(\exp\left[(d-r)\,\mathrm{H}\left(\frac{1}{2} - \frac{s/\sqrt{d}}{2(1-r/d)}\right) + r\log 2\right]\right) \tag{13}$$

and for $s = O(d^{1/4})$ we may write simply

$$\Omega\left(\frac{1}{s^2}\exp\left[-\frac{s^2}{2(1-r/d)}\right]\right) \le I \cdot 2^{-d} \le O\left(\exp\left[-\frac{s^2}{2(1-r/d)}\right]\right). \tag{14}$$

	$d-r$	r
x	0	0
y	0	1
z	j	i

Figure 1: To calculate how many points are within distance t from two points x and y, we consider without loss of generality $x = 0 \ldots 0$. 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 $d(y, z) = j + r - i$.

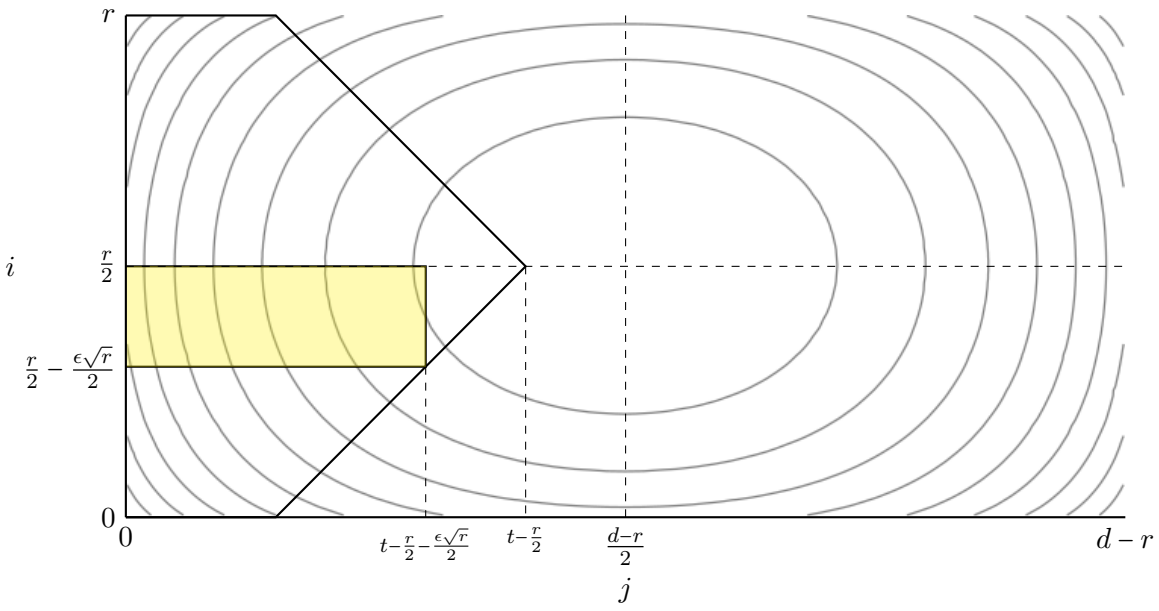

Figure 2: A contour plot over the two dimensional binomial. The pentagon on the left marks the region over which we want to sum. For the upper bound we sum i from 0 to r and j from 0 to $t - r/2$. The small yellow rectangle marks the region used for the lower bound.

For intersections we won't be able to directly apply any well known theorems like Cramér. The quantities $|\mathrm{B}(x,t) \cap \mathrm{B}(y,t)|$ are studied in the theory of error correcting codes, but unfortunately with multiplicative $d^{O(1)}$ errors, which drown the signal when the volume is about $2^d d^{-\Omega(1)}$, as we'll often want it to be.

Proof. From figure 1 we observe that the quantity I from the lemma may be computed as a two dimensional sum:

$$|B(x,t) \cap B(y,t)| = \sum_{\substack{i+j \le t \\ j-i \le t-r}} \binom{r}{i}\binom{d-r}{j}. \tag{15}$$

Here we are counting over groups of points with the same distance to x and y, namely $i+j$ and $j+r-i$. We can get a quick estimate of the sum, by considering the largest element $\binom{r}{r/2}\binom{d-r}{t-r/2}$ and using $\binom{r}{r/2} \approx 2^r$ and the normal approximation to $\binom{d-r}{t-r/2} = \binom{d-r}{\frac{d-r}{2} - \frac{s\sqrt{d-r}}{2\sqrt{1-\delta}}} = 2^{d-r}\exp(\frac{-s^2}{2(1-\delta)})$, giving combined $I \approx \exp(\frac{-s^2}{2(1-\delta)})$, which promisingly looks a lot like the lemma.

More rigorously, observe the following two inequalities:

$$\left(B_r\left(\frac{r}{2}\right) - B_r\left(\frac{r}{2} - \frac{\epsilon\sqrt{r}}{2}\right)\right) B_{d-r}\left(t - \frac{r}{2} - \frac{\epsilon\sqrt{r}}{2}\right) \le I \le B_r(r) B_{d-r}\left(t - \frac{r}{2}\right). \tag{16}$$

These are illustrated in figure 2 and correspond, respectively, to a particular rectangle inside the region specified by (15), and a rectangle bounding the region.

We proceed to bound the different balls. On the upper bound side we get

$$\begin{aligned} B_r(r) &= 2^r, \text{ and} \\ B_{d-r}\left(t - \frac{r}{2}\right) &= B_{d-r}\left(\frac{d-r}{2} - \frac{s\sqrt{d-r}}{2\sqrt{1-\delta}}\right) \\ &= \frac{1}{\sqrt{2\pi x}} \exp\left[(d-r)\,\mathrm{H}\left(\frac{1}{2} - \frac{s}{2\sqrt{1-\delta}\sqrt{d-r}}\right)\right], \end{aligned}$$

which easily combines to prove (13). On the lower bound side we note by Uhlmanns theorem and Berry Esseen:

$$\begin{aligned} B_r(r/2) &\ge 2^{r-1}, \text{ and} \\ B_r\left(\frac{r}{2} - \frac{\epsilon\sqrt{r}}{2}\right) &= (\Phi(-\epsilon) + O(1/\sqrt{d})) \cdot 2^r \\ &= \left(\frac{1}{2} - \frac{\epsilon}{\sqrt{2\pi}} + O(\epsilon^2) + O(1/\sqrt{d})\right) \cdot 2^r. \end{aligned}$$

We will eventually take $\epsilon = 1/x = O(1)$, so this far everything is bounded quite well. The most difficult part is bounded the last term on the lower bound side. We expand as follows, using the

rules from the table in the preliminaries:

$$B_{d-r}\left(t-\frac{r}{2}-\frac{\epsilon\sqrt{r}}{2}\right) = \frac{1}{\sqrt{2\pi}s}\exp\left[(d-r)\,\mathrm{H}\left(\frac{1}{2}-\frac{s\sqrt{d-r}+\epsilon\sqrt{r}}{2\sqrt{1-\delta}(d-r)}\right)\right]$$
$$= \frac{1}{\sqrt{2\pi}s}\exp\left[(d-r)\,\mathrm{H}\left(\frac{1}{2}-\frac{s}{2\sqrt{1-\delta}\sqrt{d-r}}\right)-\tau\right]$$
$$\text{where } \tau = \frac{\epsilon\sqrt{r}}{2\sqrt{1-\delta}}\log\frac{1-\xi}{\xi}+O\left(\frac{\epsilon^2 r}{(d-r)\xi(1-\xi)}\right)$$
$$\text{and } \xi = \frac{1}{2}-\frac{s}{2\sqrt{1-\delta}\sqrt{d-r}}.$$

Here ξ is bounded away from 0, since $s \le \sqrt{d}/2 \le (1-\delta)\sqrt{d}$ by the assumption in the lemma. Since ξ is also bounded away from 1, and δ is bounded away from 0 and 1, the error term on τ reduces to $O(\epsilon)$. Meanwhile $\log\frac{1-\xi}{\xi} = \frac{4s}{2\sqrt{1-\delta}\sqrt{d-r}} + O(\frac{s^2}{d})$, so $\tau = O(\epsilon\sqrt{d})O(s/\sqrt{d}) = O(\epsilon\cdot s)$. Now taking $\epsilon = 1/s$, gives us our intended bound:

$$\left(B_r\left(\frac{r}{2}\right)-B_r\left(\frac{r}{2}-\frac{\epsilon\sqrt{r}}{2}\right)\right)B_{d-r}\left(t-\frac{r}{2}-\frac{\epsilon\sqrt{r}}{2}\right)$$
$$\ge 2^r\left(\frac{1}{2}-\left(\frac{1}{2}-O(\epsilon)\right)\right)\frac{1}{\sqrt{2\pi}s}\exp\left[(d-r)\,\mathrm{H}\left(\frac{1}{2}-\frac{s}{2\sqrt{1-\delta}\sqrt{d-r}}\right)-O(\epsilon s)\right]$$
$$\ge \Omega\left(\frac{1}{s^2}\exp\left[(d-r)\,\mathrm{H}\left(\frac{1}{2}-\frac{s}{2\sqrt{1-\delta}\sqrt{d-r}}\right)\right]2^r\right)$$

which is what we wanted for (12).

The simple expression for $s = O(d^{1/4})$ now follows from the expansion of H around 1/2:

$$(d-r)\,\mathrm{H}\left(\frac{1}{2}-\frac{s}{2\sqrt{1-\delta}\sqrt{d-r}}\right) = (d-r)\left(\mathrm{H}\left(\frac{1}{2}\right)-\left(\frac{s}{2\sqrt{1-\delta}\sqrt{d-r}}\right)^2\Big/\left(2(\tfrac{1}{2})^2\right)\right)$$
$$= (d-r)\log 2-\frac{s^2}{2(1-\delta)}.$$

□

References

[Ahle et al., 2017] Ahle, T. D., Aumüller, M., and Pagh, R. (2017). 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.

[Alon et al., 2006] Alon, N., Moshkovitz, D., and Safra, S. (2006). Algorithmic construction of sets for k-restrictions. *ACM Transactions on Algorithms (TALG)*, 2(2):153–177.

[Andoni and Indyk, 2006] Andoni, A. and Indyk, P. (2006). Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 459–468. IEEE.

[Andoni et al., 2015] Andoni, A., Indyk, P., Laarhoven, T., Razenshteyn, I., and Schmidt, L. (2015). Practical and optimal lsh for angular distance. In *Advances in Neural Information Processing Systems*, pages 1225–1233.

[Andoni et al., 2014] Andoni, A., Indyk, P., Nguyen, H. L., and Razenshteyn, I. (2014). Beyond locality-sensitive hashing. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1018–1028. Society for Industrial and Applied Mathematics.

[Andoni et al., 2017] Andoni, A., Laarhoven, T., Razenshteyn, I., and Waingarten, E. (2017). 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.

[Andoni and Razenshteyn, 2015] Andoni, A. and Razenshteyn, I. (2015). Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 793–801. ACM.

[Arasu et al., 2006] Arasu, A., Ganti, V., and Kaushik, R. (2006). Efficient exact set-similarity joins. In *Proceedings of the 32nd international conference on Very large data bases*, pages 918–929. VLDB Endowment.

[Arya et al., 1998] Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., and Wu, A. Y. (1998). An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM)*, 45(6):891–923.

[Aumüller et al., 2017] Aumüller, M., Christiani, T., Pagh, R., and Silvestri, F. (2017). Distance-sensitive hashing. *arXiv preprint arXiv:1703.07867*.

[Becker et al., 2016] Becker, A., Ducas, L., Gama, N., and Laarhoven, T. (2016). 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.

[Bentley, 1975] Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517.

[Broder, 1997] Broder, A. Z. (1997). On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings*, pages 21–29. IEEE.

[Broder et al., 1997] Broder, A. Z., Glassman, S. C., Manasse, M. S., and Zweig, G. (1997). Syntactic clustering of the web. *Computer Networks and ISDN Systems*, 29(8-13):1157–1166.

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

[Christiani, 2017] Christiani, T. (2017). 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.

[Colbourn and Dinitz, 2006] Colbourn, C. J. and Dinitz, J. H. (2006). *Handbook of combinatorial designs*. CRC press.

[Cole et al., 2004] Cole, R., Gottlieb, L.-A., and Lewenstein, M. (2004). 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.

[Cramér, 1938] Cramér, H. (1938). On a new théoreme-limit of théory of probabilityés. *Actualités scientific and industrial*, 736(5-23):115.

[Datar et al., 2004] Datar, M., Immorlica, N., Indyk, P., and Mirrokni, V. S. (2004). Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM.

[Fredman et al., 1984] Fredman, M. L., Komlós, J., and Szemerédi, E. (1984). Storing a sparse table with 0 (1) worst case access time. *Journal of the ACM (JACM)*, 31(3):538–544.

[Gionis et al., 1999] Gionis, A., Indyk, P., and Motwani, R. (1999). 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.

[Goswami et al., 2017] Goswami, M., Pagh, R., Silvestri, F., and Sivertsen, J. (2017). Distance sensitive bloom filters without false negatives. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 257–269. SIAM.

[Hagerup and Tholey, 2001] Hagerup, T. and Tholey, T. (2001). Efficient minimal perfect hashing in nearly minimal space. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 317–326. Springer.

[Har-Peled et al., 2012] Har-Peled, S., Indyk, P., and Motwani, R. (2012). Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of computing*, 8(1):321–350.

[Indyk, 2000a] Indyk, P. (2000a). 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.

[Indyk, 2000b] Indyk, P. (2000b). *High-dimensional computational geometry*. PhD thesis, Stanford University.

[Indyk, 2001] Indyk, P. (2001). On approximate nearest neighbors under l_∞ norm. *Journal of Computer and System Sciences*, 63(4):627–638.

[Indyk, 2007] Indyk, P. (2007). 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.

[Indyk and Motwani, 1998] Indyk, P. and Motwani, R. (1998). 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.

[Karppa et al., 2016] Karppa, M., Kaski, P., Kohonen, J., and Catháin, P. Ó. (2016). Explicit correlation amplifiers for finding outlier correlations in deterministic subquadratic time. *Proceedings of the 24th European Symposium Of Algorithms (ESA '2016)*.

[Kushilevitz et al., 2000] Kushilevitz, E., Ostrovsky, R., and Rabani, Y. (2000). Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM Journal on Computing*, 30(2):457–474.

[Lv et al., 2007] Lv, Q., Josephson, W., Wang, Z., Charikar, M., and Li, K. (2007). Multi-probe lsh: efficient indexing for high-dimensional similarity search. In *Proceedings of the 33rd international conference on Very large data bases*, pages 950–961. VLDB Endowment.

[Mitzenmacher and Upfal, 2005] Mitzenmacher, M. and Upfal, E. (2005). *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge university press.

[Naor et al., 1995] Naor, M., Schulman, L. J., and Srinivasan, A. (1995). Splitters and near-optimal derandomization. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 182–191. IEEE.

[O'Donnell et al., 2014] O'Donnell, R., Wu, Y., and Zhou, Y. (2014). Optimal lower bounds for locality-sensitive hashing (except when q is tiny). *ACM Transactions on Computation Theory (TOCT)*, 6(1):5.

[Pagh, 2016] Pagh, R. (2016). Locality-sensitive hashing without false negatives. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1–9. SIAM.

[Pagh and Christiani, 2017] Pagh, R. and Christiani, T. (2017). Beyond minhash for similarity search. Proceedings of the forty-ninth annual ACM symposium on Theory of computing.

[Panigrahy, 2006] Panigrahy, R. (2006). 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.

[Pham and Pagh, 2016] Pham, N. and Pagh, R. (2016). 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.

[Topsok, 2006] Topsok, F. (2006). Some bounds for the logarithmic function. *Inequality theory and applications*, 4:137.

[Turán, 1961] Turán, P. (1961). Research problems. *Közl MTA Mat. Kutató Int*, 6:417–423.

[Williams, 2005] Williams, R. (2005). A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2):357–365.