On the Complexity of Inner Product Similarity Join*

Thomas D. Ahle*, Rasmus Pagh*, Ilya Razenshteyn**, and Francesco Silvestri*

*IT University of Copenhagen, {thdy,pagh,fras}@itu.dk

**MIT CSAIL, ilyaraz@mit.edu

ABSTRACT

A number of tasks in classification, information retrieval, recommendation systems, and record linkage reduce to the core problem of *inner product similarity join* (IPS join): identifying pairs of vectors in a collection that have a sufficiently large inner product. IPS join is well understood when vectors are normalized and some *approximation* of inner products is allowed. However, the general case where vectors may have any length appears much more challenging. Recently, new upper bounds based on asymmetric locality-sensitive hashing (ALSH) and asymmetric embeddings have emerged, but little has been known on the lower bound side. In this paper we initiate a systematic study of inner product similarity join, showing new lower and upper bounds. Our main results are:

- Approximation hardness of IPS join in subquadratic time, assuming the strong exponential time hypothesis.
- New upper and lower bounds for (A)LSH-based algorithms. In particular, we show that asymmetry can be avoided by relaxing the LSH definition to only consider the collision probability of *distinct* elements.
- A new indexing method for IPS based on linear sketches, implying that our hardness results are not far from being tight.

Our technical contributions include new asymmetric embeddings that may be of independent interest. At the conceptual level we strive to provide greater clarity, for example by distinguishing among signed and unsigned variants of IPS join and shedding new light on the effect of asymmetry.

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

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) [44, 46].

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. [7, 62]). 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) [26, 25]. In the database community the similarity join problem was originally motivated by applications in data cleaning [18, 11]. 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. [45] for references and further examples). We refer to the recent book by Augsten and Böhlen [12] 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 [32, 49], but not captured by techniques for metric similarity joins such as [27, 62]. Teflioudi et al. [51] studied 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 [24] and multi-class prediction [23, 29]. IPS join also captures

[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 so-called *maximum kernel search*, a general machine learning approach with applications such as image matching and finding similar protein/DNA sequences [21].

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 that other kinds of similarity search: Williams [57, 5] 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 [52, 30], 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 [46, 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 [46, 47, 40]. 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.

1.1 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 1. *Given two point sets P, Q and a value $0 < c < 1$, 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, 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 [46].

DEFINITION 2. *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 $\rho = \log P_1 / \log P_2$ [7]. 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 $h_p(p) = h_q(q)$.

1.2 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 hypothesis 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 known [57] that OVP cannot be solved in $O\left(n^{2-\epsilon} d^{O(1)}\right)$ time, for any given constant $\epsilon > 0$, if the Strong Exponential Time Hypothesis (SETH) is true.

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 1. *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, for at least one of the following IPS join problems:*

1. *Signed (cs, s) join of $P, Q \subseteq \{-1, 1\}^d$ with $c > 0$.*
2. *Unsigned (cs, s) join of $P, Q \subseteq \{-1, 1\}^d$ with $c = e^{-o(\sqrt{\log n}/\log\log n)}$.*
3. *Unsigned (cs, s) join of $P, Q \subseteq \{0, 1\}^d$ with $c = 1 - o(1)$.*

Then the OVP conjecture is false.

Discussion. It is interesting to compare our conditional lower bound to the recent upper bounds by Karppa et al. [30], who get sub-quadratic running time for unsigned join of normalized vectors in $\{-1, 1\}^d$, when $\log(s)/\log(cs)$ is a constant smaller than 1[2]. It's not too hard to derive from OVP that we cannot expect sub-quadratic running time when $\log(s)/\log(cs) = 1 - o(1/\log n)$, however point 2. of Theorem 1 shows that even the case where $\log(s)/\log(cs) = 1 - o(1/\sqrt{\log n})$ is hard.

Theorem 1 also implies that (assuming the OVP conjecture) there does not exist a *data structure* for unsigned (cs, s) MIPS 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 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 MIPS have failed to provide sub-linear query times for small s. Under the OVP conjecture this is inevitable, at least if we support vectors with negative values. It is possible to show hardness for even super polynomial dependency on d, if we only want to show hardness for say $c = 1/\text{polylog} n$.

The $\{-1, 1\}^d$ case seems to be harder than the $\{0, 1\}^d$ case. In fact Valiant [52] reduces the case of $P, Q \subseteq \mathbb{R}^d$ to the case $P, Q \subseteq \{-1, 1\}^d$ using the LSH by Charikar [16]. Another piece of evidence is that the state-of-the-art LSH approaches give better results for $\{0, 1\}^d$ than $\{-1, 1\}^d$ as we show in section 4.2. The $\{0, 1\}^d$ case is on the other hand particularly interesting, as it is occurs often in practice, for example when the vectors represent sets. A better understanding of the upper bounds for this case, would be very interesting.

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" [52], 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 [40, 46], that there cannot exist an asymmetric LSH for unbounded query vectors. Specifically we get the following theorem:

Theorem 2. *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:*

1. *if $d \geq 1$ and $s \leq O(1/d)$, we obtain $P_1 - P_2 = O\left(1/\log(d \log_{1/c}(U/s))\right)$ for signed and unsigned IPS;*
2. *if $2 \leq d \leq \Theta\left(U^5/(c^2 s^5)\right)$ and $s \leq O(1/d)$, we obtain $P_1 - P_2 = O\left(1/\log(dU/(s(1-c)))\right)$ for signed IPS;*
3. *if $d > \Theta\left(U^5/(c^2 s^5)\right)$, we obtain $P_1 - P_2 = O\left(\sqrt{s/U}\right)$ 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 [38, 41] 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 [40]: Use a collision matrix given by two sequences of data and query vectors that force the gap to be small. The proof in [40] then applies an asymptotic analysis of the margin complexity of this matrix [50], 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 [40]. 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 [40, 47] by just plugging the best known data structure for Approximate Near Neighbor for ℓ_2 on a sphere [10] into the reduction in [40, 13]. Then we show how to circumvent the impossibility results in [40, 46] 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 the data and query vectors are identical. We conclude by describing a data structure based on the linear sketches for ℓ_p in [6] for unsigned (cs, s) search: for any given $0 < \kappa \leq 1/2$, the data structure yields a $c = 1/n^\kappa$ approximation with

[2]More precisely they need $\log(s)/\log(cs) < 2/\omega$, where ω is the matrix multiplication constant.

$\tilde{O}\left(dn^{2-2/\kappa}\right)$ construction time and $\tilde{O}\left(dn^{1-2/\kappa}\right)$ query time. Theorem 1 suggests that we cannot substantially improve the approximation with similar performance.

1.3 Previous work

Similarity join. Similarity join problems have been extensively studied in the database literature (e.g. [18, 19, 20, 27, 28, 35, 37, 48, 53, 54, 59]), as well as in information retrieval (e.g. [15, 22, 60]), and knowledge discovery (e.g. [4, 14, 55, 61, 63]). 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 [42].

IPS join. The inner product similarity for the case of *normalized* vectors is known as "cosine similarity" and it is well understood [17, 34, 44]. While the general case where vectors may have any length appears theoretically challenging, *practically* efficient indexes for unsigned search were proposed in [44, 31], based on tree data structures combined with a branch-and-bound space partitioning technique similar to k-d trees, and in [13] based on principal component axes trees. For document term vectors Low and Zheng [36] 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 [56].

The efficiency of approximate IPS approaches based on LSH is studied in [46, 40]. 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 [47]. 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 [52] 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. [30], 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 [5] 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.

2. HARDNESS OF IPS JOIN

We first provide an overview of OVP and of the associated conjecture in next Section 2.1. Then, in Section 2.2, we prove Theorem 1 by describing some reductions from the OVP to signed/unsigned joins.

2.1 Preliminaries

The Orthogonal Vectors Problem (OVP) is defined as follows:

DEFINITION 3 (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, and the connection was proved by Williams [57]. We remark that the conjectures are assumed to hold even against randomized algorithms [1]. We will therefore assume the following plausible conjecture:

CONJECTURE 1 ([57]). *For all $\epsilon > 0$ and (randomized) algorithms for the OVP problem, there exists $\gamma > 0$ such that the running time of the algorithm, with $|P| = |Q| = n$ and $\gamma \log n$ dimensions, is not in $O(n^{2-\epsilon})$.*

We note that the conjecture does not hold for $d = O(\log n)$: It has recently been proved in [2] that there exists an algorithm for OVP running in $O\left(n^{2-\varepsilon}\right)$, for some $\varepsilon > 0$, when $d = O(\log n)$.

Hence, in order to disprove OVP, an algorithm must be strongly subquadratic when $d = \gamma \log n$ for *all* γ. It is instructive to state the negation of the conjecture, which is: "There exists $\epsilon > 0$ and an algorithm such that for all constant γ the algorithm runs in time $O(n^{2-\epsilon})$ on instances of dimension $\gamma \log n$."

The OVP conjecture, as usually stated, concerns the case where the two sets have equal size. However in order to get hardness for the case where $|P| \neq |Q|$ we consider the following generalization of OVP which has the same hardness as the usual OVP for all parameter values.

LEMMA 1 (GENERALIZED OVP). *Suppose that there exist constants $\epsilon > 0$ and $\alpha > 0$ and a (randomized) algorithm such that for every constant γ the algorithm solves OVP for $P, Q \subseteq \{0,1\}^{\gamma \log n}$ 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 $n^{1+\alpha-\epsilon}$ 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 $n^{1-\alpha} n^{1+\alpha-\epsilon} = n^{2-\epsilon}$, contradicting OVP. $\square$

2.2 Reductions from OVP

In this section we prove Theorem 1, about hardness of approximate joins. We will do this by showing the existence of certain efficient embeddings of the OVP problem. We need the following definition:

DEFINITION 4. *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* $d_2' \le d_2$*,* $\mathcal{A} \subseteq \mathbb{R}$*, and for any* $x, y \in \{0,1\}^{d_1}$*:*

$$|f(x)^T g(y)| \le cs \quad \text{when} \quad (x, y) \ge 1$$
$$|f(x)^T g(y)| \ge s \quad \text{when} \quad (x, y) = 0$$

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 2. *Suppose we have the following:*

- *For given constants* $\alpha \ge 0$*,* $a \ge 0$ *and* $\epsilon > 0$*, there exists an algorithm for (un)signed* (cs, s) *join for* $|Q| = n$ *and* $|P| = n^\alpha$ *over* $\mathcal{A}$*, running in time* $O(d^a n^{1+\alpha-\epsilon})$*.*
- *For some constants* $\delta \ge 0$ *and* $b \ge 0$ *and for all* $\gamma > 0$*, there exists an (un)signed* $(\gamma \log n, n^\delta, cs, s)$*-gap embedding into* $\mathcal{A}$ *running in time* $n^{\delta b}$*.*

If $b\delta < \alpha$ *and* $a\delta < \epsilon$*, then the OVP conjecture is false.*

PROOF. Given an OVP instance with $|Q| = n$, $|P| = n^\alpha$ and dimension $\gamma \log n$, take a (un)signed $(\gamma \log n, n^\delta, cs, s)$-gap embedding (f, g) and apply it 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 $\le cs$ otherwise. Now run the algorithm for (un)signed (cs, s) join on $(f(P), g(Q))$, which produces the orthogonal pair, if it exists.

As $f(P)$,$g(Q) \subseteq \mathcal{A}^{n^\delta}$, the above procedure runs in time

$$O\left(n^{1+b\delta} + n^{1+\alpha-\epsilon+\delta a}\right).$$

The first term is the time to perform the embedding of all vectors in P and Q, while the second term is for computing the join on the embedded vectors. Since we assume $b\delta < \alpha$ and $a\delta < \epsilon$, we get that OVP can be solved

$$O\left(n^{1+\alpha-\epsilon_1} + n^{1+\alpha-\epsilon_2}\right) = O\left(n^{1+\alpha-\min(\epsilon_1,\epsilon_2)}\right)$$

where $\epsilon_1 = \alpha - b\delta$ and $\epsilon_2 = \epsilon - a\delta$, which contradicts the OVP conjecture. □

The last ingredient we need to show Theorem 1 is a suitable family of embeddings to use with Lemma 2:

LEMMA 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* k*.*

PROOF. We will use the following notation in our constructions: Let $x \boxplus y$ be the concatenation of vectors x and y;[3] Let x^n mean x concatenated with itself n times;[4] 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 points 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.

(Embedding 2) We recall the recursive definition of the q-th order Chebyshev polynomial of first kind (see, e.g., [3] page 782):

$$T_0(x) = 1$$
$$T_1(x) = x$$
$$T_{q+1}(x) = 2xT_q(x) - T_{q-1}(x)$$

The polynomials have the following properties [52]:

$$|T_q(x)| \le 1 \quad \text{when} \quad |x| \le 1$$
$$|T_q(1+\epsilon)| \ge e^{q/\sqrt{\epsilon}} \quad \text{when} \quad 1 > \epsilon > 0$$

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.

[3] $\boxplus$ for concatenation and $\boxtimes$ for tensoring stresses their dual relationship with $+$ and $\times$ 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.

[4] 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.

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:

$$\begin{aligned}
&f_1(x),\ g_1(y) := 1,\ 1\\
&f_2(x),\ g_2(y) := x,\ y\\
&f_{n+2}(x) := (x \boxtimes f_{n+1}(x))^2 \boxplus f_n(x)^{(2d)^2}\\
&g_{n+2}(y) := (y \boxtimes g_{n+1}(y))^2 \boxplus (-g_n(y))^{(2d)^2}
\end{aligned}$$

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:

$$\begin{aligned}
f_1(x)^T g_1(y) &= 1\\
f_2(x)^T g_2(y) &= x^T y\\
f_{q+2}(x)^T g_{q+2}(y) &= 2x^T y\ f_{q+1}(x)^T g_{q+1}(y)\\
&\quad - (2d)^2 f_q(x)^T g_q(y)
\end{aligned}$$

  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:

$$\begin{aligned}
d_1 &= 1\\
d_2 &= 4d - 4\\
d_n &= 2(4d-4)d_{n-1} + (2d)^2 d_{n-2}
\end{aligned}$$

  It can be verified that $d_q \le (9d)^q$ for any $q \ge 0$ and $d \ge 8$. 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 $q \ge 0$ 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 [52] 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:

$$(1 - x_1y_1)(1 - x_2y_2)\cdots(1 - x_dy_d)$$

since $\{0,1\}$ is closed under tensoring and

$$1 - x_iy_i = (1 - x_i, 1)^T (y_i, 1 - y_i)$$

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 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 $d_2 = k2^{d/k}$ 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 $(1 - x_iy_i)$ 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:

$$\begin{aligned}
f(x) &:= \boxplus_{i=0}^{k-1} \boxtimes_{j=1}^{d/k} (1 - x_{ik/d+j}, 1)\\
g(x) &:= \boxplus_{i=0}^{k-1} \boxtimes_{j=1}^{d/k} (y_{ik/d+j}, 1 - y_{ik/d+j})
\end{aligned}$$

And the running time is linear in the output dimension. ☐

Finally we parametize and prove Theorem 1. 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 $\delta > 0$.

Proof. (Theorem 1)

1. Letting $d = \gamma \log n$ in embedding 1 gives us a signed $(\gamma \log n, n^{o(1)}, 0, 1)$-embedding into $\{-1,1\}$.
2. Letting $q = \delta \frac{\log n}{\log 9d}$ in embedding 2 gives us an unsigned $(\gamma \log n, n^\delta, cs, s)$-embedding into $\{-1,1\}$ for any $\delta, \gamma > 0$ with

$$\begin{aligned}
cs &= n^{\delta(1 - \frac{\log 9 - \log 2}{\log 9\gamma + \log\log n})} = n^{\delta - O(1/\log\log n)}\\
c &= e^{-\frac{\delta\sqrt{\log n}}{\sqrt{\gamma}\log\log n}(1 - \frac{\log 9\gamma}{\log\log n})}/2 = e^{-O(\frac{\sqrt{\log n}}{\sqrt{\gamma}\log\log n})}
\end{aligned}$$

3. Letting $k = \gamma/\delta + o(1)$ we can get an unsigned $(\gamma \log n$, n^δ, cs, $s)$-embedding into $\{0,1\}$ for any $\delta, \gamma > 0$ with

$$\begin{aligned}
cs &= \gamma/\delta - 1 + o(1)\\
c &= \frac{cs}{s} = \frac{\gamma/\delta - 1 + o(1)}{\gamma/\delta + o(1)} = 1 - O(1/\gamma)
\end{aligned}$$

For the second embedding, we also calculate the $\log(s)/\log(cs)$ constant, as described in the introduction. To do so, we must normalize the vectors onto the unit sphere, that is, we calculate:

$$\frac{\log(s/d_2)}{\log(cs/d_2)} = \frac{q\log(2/9) + q/\sqrt{d} - \log 2}{q\log(2/9)}$$
$$= 1 - \frac{1}{\log(9/2)\sqrt{d}} + \frac{\log 2}{q\log(9/2)}$$
$$= 1 - O(1/\sqrt{\gamma \log n})$$

Very crucially, since we want the results to hold for all $\gamma > 0$, we think of it as $\omega(1)$, which then translates the $O(1/\gamma)$s into $o(1)$ in the theorem. $\square$

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 points, so it applies also to data dependent LSH [10]. A consequence of our result is that there cannot exist an asymmetric LSH for any dimension $d \geq 1$ when the set of query points is unbounded, getting a result similar to that of [40], which however requires even the data space to be unbounded and $d \geq 2$.

We firsts show in Lemma 4 that the gap $P_1 - P_2$ can be expressed as a function of the length h of two sequences of query and data points with suitable collision properties. Then we provide the proof of the aforementioned Theorem 2, where we derive some of such sequences and then apply the lemma.

LEMMA 4. *Suppose that there exists a sequence of data vectors $P = \{p_0, \ldots, p_{n-1}\}$ and a sequence of query vectors $Q = \{q_0, \ldots, q_{n-1}\}$ 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 1. Specifically, we split the lower triangle into *squares* $G_{r,s}$ for every r and s with

$$0 \leq r < \log(n+1) = \ell \text{ and } 0 \leq s < (n+1)/2^{r+1} = 2^{\ell-r-1},$$

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 $i \leq j' \leq j$, or $j' = j$ and $i \leq i' \leq j$) 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 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 f: 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

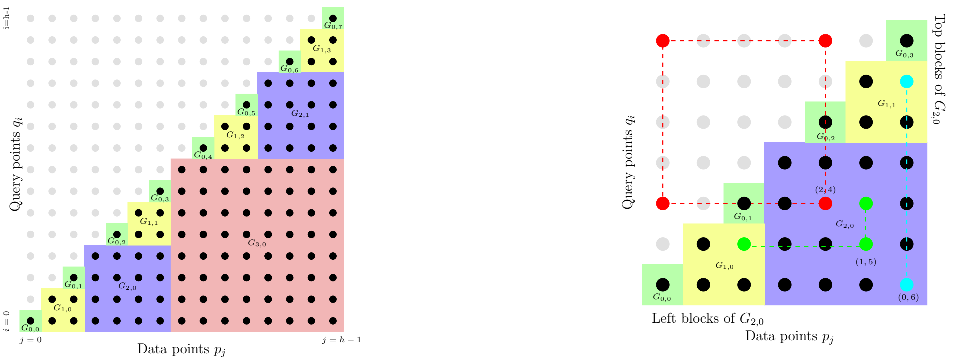

Figure 1: On the left, a 15×15 grid: black nodes are P_1-nodes, gray nodes are P_2-nodes; the colored blocks denote the partitioning of the lower triangle into squares. On the right, a zoom of the $G_{2,0}$ square and of its left and top squares: the red nodes collide under a $(2,4)$-shared function; the green nodes collide under a $(1,5)$-partially shared function; the cyan node collide under a $(0,6)$-proper function (specifically, row proper).

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. $\square$

We are now ready to prove Theorem 2.

Proof. (Theorem 2) In all cases we use Lemma 4 with different sequence length n.

First case. Let $d \geq 1$. Consider the following query and data points:

$$q_i = (Uc^i, 0, \ldots, 0) \qquad p_j = (s/(Uc^j), 0, \ldots, 0)$$

with $0 \leq i, j < n = \lfloor \log_{1/c}(U/n) \rfloor$. 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. We observe that data and query points are respectively contained in the unit ball and in the ball of radius U since $0 \leq i, j < \lfloor \log_{1/c}(U/n) \rfloor$. When $d \geq 3$ and $s \leq O(1/d)$, a longer sequence can be obtained by concatenating suitable rotations of the previous sequences. Suppose for the sake of generality that $d = 2d'$ (the general case requires some more tedious computations). Consider the points $q_{i,\ell}$ and $p_{j,\ell}$ with $0 \leq i, j < \Theta\left(\log_{1/c}(U/s)\right)$ and $0 \leq \ell < d'$. Point $q_{i,\ell}$ (resp., $p_{j,\ell}$) is defined as follows: positions $2n+1$ are set to $\sqrt{s}$ for each $\ell \leq n < d'$ (resp., $0 \leq h < \ell$); position $2\ell - 1$ is set to Uc^i (resp., $s/(Uc^j)$); the remaining positions are set to 0. We get; $p^T_{j,\ell} q_{i,\ell} = sc^{i-j}$ as in the previous case; $p^T_{j,\ell'} q_{i,\ell} = 0$ if $\ell' < \ell$; $p^T_{j,\ell'} q_{i,\ell} \geq s$ if $\ell' \geq \ell$. If $s = O(1/d)$, data and query points are contained in balls of radius 1 and U, respectively. Therefore, by suitably concatenating the above data and query points by the index ℓ we get a sequence of size $\Theta\left(d \log_{1/c}(U/s)\right)$ and the claim follows. Note that all inner products are non negative and then the sequence is valid for signed and unsigned IPS.

Second case. For $d \geq 2$, a longer sequence can constructed for signed IPS. We use the following query and data points:

$$q_i = \left(i\sqrt{\frac{s(1-c)}{U}}, \sqrt{\frac{s}{U}}, 0, \ldots, 0\right),$$

$$p_j = \left(\sqrt{sU(1-c)}, \sqrt{sU}(1-(1-c)j), 0, \ldots, 0\right),$$

with $0 \leq i, j < n = \Theta\left(\sqrt{U/(s(1-c))}\right)$. Note that these sequences are similar to the one used by [40]. We have $p_j^T q_i = s(1-c)(i-j)+s$: then, $p_j^T q_i \geq s$ if $j \geq i$ and $p_j^T q_i \leq cs$ otherwise. As soon as $0 \leq i, j < n = \Theta\left(\sqrt{U/(s(1-c))}\right)$, data and query points are guarantee to be in balls of radius respectively 1 and U. By applying the trick used in the previous case, it is possible to increase the length of the sequence by a factor $d/4$ if $s = O(1/d)$ by concatenating different sequences given by rotations of the previous one. The second 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 consider the case $d \geq \Theta\left(U^5/(c^2 s^5)\right)$. We now provide sequences of data/query points 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 $d = \Omega\left(\epsilon^{-2}\log n\right) = \Omega\left((\log^5 h)/c^2\right)$ (for an analysis see e.g. [43]). For notational convenience, we denote the vectors in $\mathcal{Z}$ as follows: $z_{b_0}, z_{b_0,b_1}, \ldots, z_{b_0,b_1,\ldots,b_{\log n-1}}$ for each possible value $b_0,\ldots,b_{\log n-1} \in \{0,1\}$. 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. We construct the following query and data sets:

$$q_i = \sqrt{2sU}\sum_{\ell=0}^{\log n-1} \bar{b}_{i,\ell} z_{b_{i,0},\ldots b_{i,\ell-1},\bar{b}_{i,\ell}}$$

$$p_j = \sqrt{2s/U}\sum_{\ell=0}^{\log n-1} b_{j,\ell} z_{b_{j,0},\ldots b_{j,\ell-1},b_{j,\ell}}$$

Since the inner product of two distinct points in $\mathcal{Z}$ is in the range $[-\epsilon,\epsilon]$, we have that $p_j^T q_i$ can be upper bounded as

$$p_j^T q_i \leq \epsilon 2s(\log^2 n - \log n) + \\ + 2s\sum_{\ell=0}^{\log n-1} b_{j,\ell}\bar{b}_{i,\ell} z_{b_{j,0},\ldots b_{j,\ell-1},b_{j,\ell}} z_{b_{i,0},\ldots b_{i,\ell-1},\bar{b}_{i,\ell}}$$

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 $z_{b_{j,0},\ldots b_{j,\ell-1},b_{j,\ell}} \neq z_{b_{i,0},\ldots b_{i,\ell},\bar{b}_{i,\ell}}$ for all $\ell \neq \ell'$ and $b_{j,\ell} b_{i,\ell} = 0$ for all $\ell = \ell'$. It then follows that $p_j^T q_i \leq \epsilon 2s\log^2 n$ when $i > j$. On the other hand we get that $p_j^T q_i$ can be lower bounded as

$$p_j^T q_i \geq -\epsilon 2s(\log^2 n - \log n) + \\ + 2s\sum_{\ell=0}^{\log n-1} b_{j,\ell}\bar{b}_{i,\ell} z_{b_{j,0},\ldots b_{j,\ell-1},b_{j,\ell}} z_{b_{i,0},\ldots b_{i,\ell-1},\bar{b}_{i,\ell}}$$

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 $z_{b_{j,0},\ldots b_{j,\ell'-1},b_{j,\ell'}} = z_{b_{i,0},\ldots b_{i,\ell'-1},\bar{b}_{i,\ell}}$. It then follows that $p_j^T q_i \geq -\epsilon 2s\log^2 n + 2s$. By setting $\epsilon = c/(2\log^2 n)$, we get that $p_j^T q_i \leq cs$ if $j < i$ and $p_j^T q_i \geq s$ if $j \geq i$. We now observe that each q_i (resp., p_j) is given by the sum of at most $\log n$ vectors with norm at most $\sqrt{2sU}(1+\epsilon)$ (resp., $\sqrt{2s/U}(1+\epsilon)$). Since $n = 2^{\sqrt{U/(8s)}}$, data and query points are respectively contained within balls of radius 1 and U.

Finally, we observe that in all three cases the gap becomes 0 if the query ball is unbounded. Then, there cannot exist an asymmetric LSH with $P_1 > P_2$. □

4. UPPER BOUNDS

This section contains three observations with implications for IPS join and its indexing version. We first notice in Section 4.1 that by plugging the best known LSH for ℓ_2 distance on a sphere [10] into a reduction presented in [13, 40], we get a data structure based on LSH for signed MIPS with search time exponent $\rho = (1-s)/(1+(1-2c)s)$.

Then, in Section 4.2, we show how to circumvent the results in [40, 46] 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 [39] to implement a symmetric version of the reduction in [13, 40].

Finally, in Section 4.3 we solve unsigned (cs,s) join using linear sketches for ℓ_p norms from [6]. Given $\kappa \geq 2$ we obtain a approximation factor $c \geq 1/n^{1/\kappa}$ using $\tilde{O}\left(dn^{2-2/\kappa}\right)$ time. Although this trade-off is not that strong, it is not far from the conditional lower bound in Theorem 1.

4.1 Asymmetric LSH for signed IPS

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 [40]: a data vector p is mapped to $(p, \sqrt{||p||^2}, 0)$, while a query q is mapped to $(q/U, 0, \sqrt{||q||^2/U})$. This transformation does not change inner products and then signed inner product search can be seen as an instance of ANN in ℓ_2 with distance threshold $r = \sqrt{2(1-s)}$ and approximation $c' = \sqrt{(1-cs)/(1-s)}$. The latter can be solved in space $O(n^{1+\rho} + dn)$ and query time $O(n^\rho)$ using the LSH construction of [10]. We get the following ρ value (for the LSH gap as well as for the exponent of the running time):

$$\rho = \frac{1}{2c'^2 - 1} = \frac{1-s}{1+(1-2c)s} \;. \tag{1}$$

The obtained running time is:

- stronger than the one from [40] in all regimes;
- stronger than the one in [47], tailored to binary data, in most parameter settings.

The latter conclusion is somewhat surprising, since the data structure we obtain works for non-binary vectors as well. We point out that in practice one may want to use a recent LSH family from [8] that—both in theory and in practice—is superior to the hyperplane LSH from [17] used in [40].

In Figure 2, we plot the ρ values of three LSH constructions: the one proposed here, the one from [40], and the one from [47]. The latter works only for binary vectors. We point out that our bound is always stronger than the one from [40] and sometimes stronger than the one from [47], despite that the latter is tailored for binary vectors.

4.2 Symmetric LSH for almost all vectors

Neyshabur and Srebro [40] 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 points are identical.

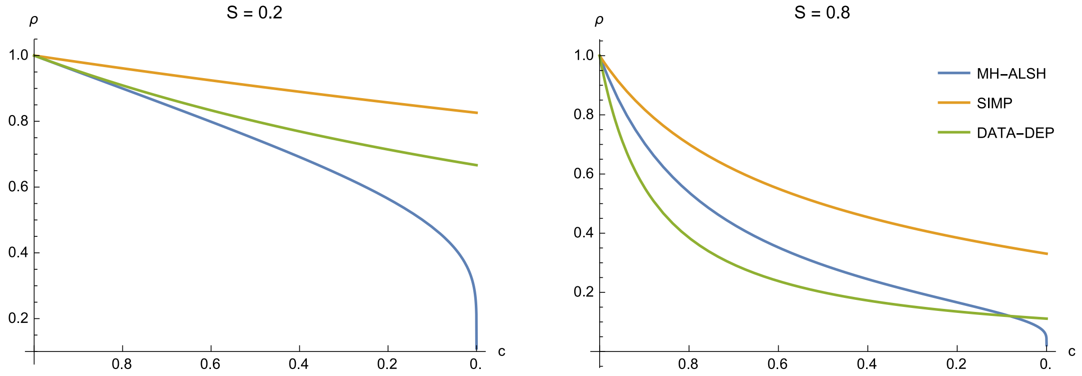

Figure 2: Our ρ value (DATA-DEP) compared to that of [40] (SIMP) and the binary data only of [47] (MH-ALSH).

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 [10]. This reduction treats data and query vectors identically, unlike the one from [40], and thus we are able to obtain a symmetric LSH.

Assume that all the coordinates of all the data and queries are encoded as k-bit numbers and that the 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 [39] it is shown how to build such vectors using Reed-Solomon codes. The resulting dimension is $O\left(\varepsilon^{-2} \log N\right) = O\left(kd/\varepsilon^2\right)$ [33, 39].

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 [10, 8], 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 (1). 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 [7], 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 $q^T q \geq s$.

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 point q, a point $p \in P$ where $|p^T q| \geq c(p'^T q)$, where p' is the point 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 points in P and then performs a query for each point 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 for unsigned c-MIPS 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 $c \geq 1/n^{1/\kappa}$ 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 1, 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 [58, 9]). More specifically, we use the following result from [6]: 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 the aforementioned result in [6] 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. REFERENCES

[1] A. Abboud, V. Vassilevska Williams, and H. Yu. Matching triangles and basing hardness on an extremely popular conjecture. In *Proc. 47th ACM on Symposium on Theory of Computing (STOC)*, pages 41–50, 2015.

[2] A. Abboud, R. Williams, and H. 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] M. Abramowitz and I. A. Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables.* Courier Corporation, 1964.

[4] P. Achlioptas, B. Schölkopf, and K. 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.

[5] J. Alman and R. Williams. Probabilistic polynomials and hamming nearest neighbors. In *Proc. 56th IEEE Symposium on Foundations of Computer Science (FOCS)*, 2015.

[6] A. Andoni. High frequency moments via max-stability. Unpublished manuscript, 2012.

[7] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008.

[8] A. Andoni, P. Indyk, M. Kapralov, T. Laarhoven, I. Razenshteyn, and L. Schmidt. Practical and optimal LSH for angular distance. In *Proc. 28th Conference on Neural Information Processing Systems (NIPS)*, 2015.

[9] A. Andoni, R. Krauthgamer, and I. P. Razenshteyn. Sketching and embedding are equivalent for norms. In *Proc. 47th ACM on Symposium on Theory of Computing, (STOC)*, pages 479–488, 2015.

[10] A. Andoni and I. Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proc. 47th Symposium on Theory of Computing (STOC)*, pages 793–801, 2015.

[11] A. Arasu, V. Ganti, and R. Kaushik. Efficient exact set-similarity joins. In *Proc. International Conference on Very Large Data Bases (VLDB)*, pages 918–929, 2006.

[12] N. Augsten and M. H. Böhlen. Similarity joins in relational database systems. *Synthesis Lectures on Data Management*, 5(5):1–124, 2013.

[13] Y. Bachrach, Y. Finkelstein, R. Gilad-Bachrach, L. Katzir, N. Koenigstein, N. Nice, and U. 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.

[14] B. Bahmani, A. Goel, and R. Shinde. Efficient distributed locality sensitive hashing. In *Proc. ACM International Conference on Information and Knowledge Management (CIKM)*, pages 2174–2178, 2012.

[15] R. J. Bayardo, Y. Ma, and R. Srikant. Scaling up all pairs similarity search. In *Proc. International Conference on World Wide Web (WWW)*, pages 131–140, 2007.

[16] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proc. 34 ACM Symposium on Theory of computing (STOC)*, pages 380–388. ACM, 2002.

[17] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proc. 34th ACM Symposium on Theory of Computing (STOC)*, pages 380–388, 2002.

[18] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In *Proc.22nd International Conference on Data Engineering (ICDE)*, page 5, 2006.

[19] Y. Chen and J. M. Patel. Efficient evaluation of all-nearest-neighbor queries. In *Proc. International Conference on Data Engineering (ICDE)*, pages 1056–1065, 2007.

[20] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. D. Ullman, and C. Yang. Finding interesting associations without support pruning. *IEEE Trans. Knowl. Data Eng.*, 13(1):64–78, 2001.

[21] R. R. Curtin, A. G. Gray, and P. Ram. Fast exact max-kernel search. In *Proc. 13th SIAM International Conference on Data Mining (SDM)*, pages 1–9, 2013.

[22] A. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *Proc. International Conference on World Wide Web (WWW)*, pages 271–280, 2007.

[23] T. Dean, M. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. 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.

[24] P. Felzenszwalb, R. 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.

[25] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proc. 25th International Conference on Very Large Data Bases (VLDB)*, pages 518–529, 1999.

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

[27] E. H. Jacox and H. Samet. Metric space similarity joins. *ACM Transactions on Database Systems (TODS)*, 33(2):7, 2008.

[28] Y. Jiang, D. Deng, J. Wang, G. Li, and J. 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.

[29] T. Joachims, T. Finley, and C.-N. J. Yu. Cutting-plane training of structural svms. *Mach. Learn.*, 77(1):27–59, 2009.

[30] M. Karppa, P. Kaski, and J. Kohonen. A faster subquadratic algorithm for finding outlier correlations. In *Proc. 27th ACM-SIAM Symposium on Discrete Algorithms (SODA16)*, 2016.

[31] N. Koenigstein, P. Ram, and Y. 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.

[32] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009.

[33] K. G. Larsen and J. Nelson. The johnson-lindenstrauss lemma is optimal for linear dimensionality reduction. *CoRR*, abs/1411.2404, 2014.

[34] D. Lee, J. Park, J. Shim, and S.-g. Lee. An efficient similarity join algorithm with cosine similarity predicate. In *Database and Expert Systems Applications*, pages 422–436. Springer, 2010.

[35] G. Li, D. Deng, J. Wang, and J. Feng. Pass-join: A partition-based method for similarity joins. *Proc. VLDB Endowment*, 5(3):253–264, 2011.

[36] Y. Low and A. 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.

[37] J. Lu, C. Lin, W. Wang, C. Li, and H. Wang. String similarity measures and joins with synonyms. In *Proc. 2013 ACM SIGMOD International Conference on Management of Data*, pages 373–384, 2013.

[38] R. Motwani, A. Naor, and R. Panigrahi. Lower bounds on locality sensitive hashing. In *Proc. 22nd Symposium on Computational Geometry (SoCS)*, pages 154–157, 2006.

[39] J. Nelson, H. L. Nguyen, and D. P. Woodruff. On deterministic sketching and streaming for sparse recovery and norm estimation. *Linear Algebra and its Applications*, 441(0):152 – 167, 2014.

[40] B. Neyshabur and N. Srebro. On symmetric and asymmetric lshs for inner product search. In *Proc. 32nd International Conference on Machine Learning (ICML)*, 2015.

[41] R. O'Donnell, Y. Wu, and Y. Zhou. Optimal lower bounds for locality-sensitive hashing (except when q is tiny). *ACM Trans. Comput. Theory*, 6(1):5:1–5:13, 2014.

[42] R. Pagh, N. Pham, F. Silvestri, and M. Stöckel. I/O-efficient similarity join. In *Proc. 23rd European Symposium on Algorithms (ESA)*, pages 941–952, 2015.

[43] R. Pagh, F. Silvestri, J. Sivertsen, and M. 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.

[44] P. Ram and A. 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.

[45] V. Satuluri and S. Parthasarathy. Bayesian locality sensitive hashing for fast similarity search. *Proc. VLDB Endowment*, 5(5):430–441, 2012.

[46] A. Shrivastava and P. Li. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). In *Proc. 27th Conference on Neural Information Processing Systems (NIPS)*, pages 2321–2329, 2014.

[47] A. Shrivastava and P. 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.

[48] Y. N. Silva, W. G. Aref, and M. H. Ali. The similarity join database operator. In *Proc. International Conference on Data Engineering (ICDE)*, pages 892–903. IEEE, 2010.

[49] N. Srebro, J. D. M. Rennie, and T. S. Jaakola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems 17*, pages 1329–1336. MIT Press, 2005.

[50] N. Srebro and A. Shraibman. Rank, trace-norm and max-norm. In *Proc. 18th Conference on Learning Theory COLT*, volume 3559 of *LNCS*, pages 545–560, 2005.

[51] C. Teflioudi, R. Gemulla, and O. 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.

[52] G. Valiant. Finding correlations in subquadratic time, with applications to learning parities and the closest pair problem. *J. ACM*, 62(2):13:1–13:45, 2015.

[53] J. Wang, G. Li, and J. 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.

[54] J. Wang, G. Li, and J. 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.

[55] Y. Wang, A. Metwally, and S. 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.

[56] R. Weber, H.-J. Schek, and S. 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.

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

[58] D. P. Woodruff. Sketching as a tool for numerical linear

algebra. *Foundations and Trends in Theoretical Computer Science*, 10:1–157, 2014.

[59] C. Xia, H. Lu, B. C. Ooi, and J. Hu. Gorder: an efficient method for knn join processing. In *Proc. VLDB*, pages 756–767, 2004.

[60] C. Xiao, W. Wang, X. Lin, and J. X. Yu. Efficient similarity joins for near duplicate detection. In *Proc. International Conference on World Wide Web (WWW)*, pages 131–140, 2008.

[61] R. B. Zadeh and A. Goel. Dimension independent similarity computation. *The Journal of Machine Learning Research*, 14(1):1605–1626, 2013.

[62] P. Zezula, G. Amato, V. Dohnal, and M. Batko. *Similarity search: the metric space approach*, volume 32. Springer Science & Business Media, 2006.

[63] X. Zhang, F. Zou, and W. 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.