The Space Complexity of Inner Product Filters

Rasmus Pagh ⓘ
IT University of Copenhagen, Denmark
BARC, Copenhagen, Denmark
pagh@itu.dk

Johan Sivertsen ⓘ
IT University of Copenhagen, Denmark
johanvts@gmail.com

—————— **Abstract** ——————

Motivated by the problem of filtering candidate pairs in inner product similarity joins we study the following *inner product estimation* problem: Given parameters $d \in \mathbb{N}$, $\alpha > \beta \geq 0$ and unit vectors $x, y \in \mathbb{R}^d$ consider the task of distinguishing between the cases $\langle x, y \rangle \leq \beta$ and $\langle x, y \rangle \geq \alpha$ where $\langle x, y \rangle = \sum_{i=1}^{d} x_i y_i$ is the inner product of vectors x and y. The goal is to distinguish these cases based on information on each vector encoded independently in a bit string of the shortest length possible. In contrast to much work on compressing vectors using randomized dimensionality reduction, we seek to solve the problem *deterministically*, with no probability of error. Inner product estimation can be solved in general via estimating $\langle x, y \rangle$ with an additive error bounded by $\varepsilon = \alpha - \beta$. We show that $d \log_2 \left(\frac{\sqrt{1-\beta}}{\varepsilon} \right) \pm \Theta(d)$ bits of information about each vector is necessary and sufficient. Our upper bound is constructive and improves a known upper bound of $d \log_2(1/\varepsilon) + O(d)$ by up to a factor of 2 when β is close to 1. The lower bound holds even in a stronger model where one of the vectors is known exactly, and an arbitrary estimation function is allowed.

1 Introduction

Modern data sets increasingly consist of noisy or incomplete information, which means that traditional ways of matching database records often fall short. One approach to dealing with this in database systems is to provide *similarity join* operators that find pairs of tuples satisfying a similarity predicate. We refer to the book of Augsten and Böhlen [5] for a survey of similarity joins in relational database systems, and to [30] for an overview of theoretical results in the area. Note that joins can be implemented using similarity search *indexes* that allow searching a relation for tuples that satisfy a similarity predicate with a given query q. Thus we include works on similarity search indexes in discussion of previous work on similarity join.

In this paper we consider *inner product similarity* predicates of the form $\langle x, y \rangle / geq\alpha$, where $x, y \in \mathbb{R}^d$ are real-valued vectors, i.e., the predicate is true for vectors whose inner product $\sum_{i=1}^{d} x_i y_i$ exceeds a user-specified threshold α. This notion of similarity is important, for example, in neural network training and inference, (see [38]). In that context, an inner product similarity join can be used for classifying a collection of vectors according to a neural network model.

Inner product similarity join is a special case of similarity join under Euclidean distance, implemented for example in the Apache Spark framework[1]. Conversely, it generalizes *cosine similarity*, which has been studied for more than a decade (see, e.g., the influential papers [8, 11] and more recent works such as [4, 7, 13, 35]). In recent years the general inner product similarity join problem has attracted increasing attention (see e.g. [2, 34, 37, 39, 43]). Recently proposed practical inner product similarity join algorithms work by reducing the general problem to a number of instances with unit length vectors, which is equivalent to join under cosine similarity [43].

Candidate generation approach. State-of-the-art algorithms computing similarity joins on high-dimensional vectors use a two-phase approach:
1. Generate a set of *candidate pairs* (x, y) that contains all pairs satisfying the predicate (keeping track of the corresponding tuples in the relations).
2. Iterate over the candidate pairs to *check* which ones satisfy the predicate.

Suppose for simplicity that both relations in the similarity join contain n tuples. A naïve candidate generation phase would output all n^2 corresponding pairs of vectors. For many data sets it is possible to reduce the number of candidate pairs significantly below n^2, but the check phase remains a bottleneck. A direct implementation of the check phase would require full information about the vectors (x, y), in practice d floating point numbers per vector. Though the inner product computation is trivial, for high-dimensional vectors the cost of transferring data from memory can be a bottleneck.

Filtering candidate pairs using approximation. An approach to reducing communication is to *approximate* inner products, which is enough to handle those candidate pairs that do not have inner product close to the threshold α. The exact inner product is computed only for the remaining pairs, often a small fraction of the set of all candidates. We stress that globally, the join computation we consider is not approximate, but approximations are used to speed up parts of the algorithm. (Note that under common assumptions in fine-grained complexity, the inner product similarity join problem is difficult in the worst case, even with approximation [1, 2].)

Such additional filtering of candidate pairs has been successfully used in "Monte Carlo" style randomized algorithms that allow the algorithm to sometimes fail to identify a pair satisfying the predicate, e.g. [35, 6]. While the error in Monte Carlo algorithms can usually be made very small at a reasonable computation cost, such algorithms are not suitable in all settings. For example:
- Firm guarantees may be needed to comply with regulation, or to ensure a clear and consistent semantics of a system (such as a DBMS) in which the similarity estimation algorithm is part.
- Guarantees on accuracy are shown under the assumption that the input data is independent of the random choices made by the algorithm. Technically this assumption may not hold

---

[1] `https://spark.apache.org/`

if output from the algorithm can affect future inputs. Maybe more seriously, if vectors can be chosen adversarially based on (partial) knowledge of the randomness of the algorithm, for example obtained by timing attacks, the guarantees cease to hold (see e.g. [14] for more discussion of adversarial settings).

In this paper we study what kind of approximation is possible without randomization, targeting settings where false negatives are not permitted, or where we cannot ensure that inputs are independent of any randomization used by the algorithm.

We seek to efficiently eliminate all candidate pairs that have inner product less than β, for some β smaller than the threshold α, so that the number of remaining candidate pairs (for which an expensive inner product computation must be done) may be significantly reduced. In order to not eliminate any candidate pair passing the threshold it is necessary and sufficient that the approximation is strong enough to distinguish the cases $\langle x, y \rangle \leq \beta$ and $\langle x, y \rangle \geq \alpha$.

Similarity join memory bottlenecks. The complexity of similarity joins in the I/O model was studied in [31], which assumes that a block transfer moves B vectors from or to external memory, and that internal memory can hold M vectors. Reducing the amount of data that needs to be transferred to evaluate a similarity predicate leads to a larger capacity of blocks as well as internal memory, in turn leading to a reduction in I/O complexity that is roughly proportional to the reduction in size. The exact improvement is a bit more complicated because additional I/Os are needed to evaluate the exact inner products of pairs with similarity above β. McCauley and Silvestri [29] studied the related problem of similarity joins in MapReduce where considerations similar to the I/O model can be made.

1.1 Our results

Without loss of generality we can consider *unit vectors*, since the general estimation problem can be reduced to this case by storing an (approximate) norm of each vector in space independent of the number of dimensions. Similarly, lower bounds shown for unit vectors imply lower bounds for arbitrary vector lengths by a scaling argument

We study the following version of the inner product estimation problem for unit vectors: Distinguish inner products smaller than β from inner products larger than α, for threshold parameters α and β. This problem can of course be solved by estimating the inner product with additive error less than $\alpha - \beta$. However, we will see that the number of bits needed is not a function of $\alpha - \beta$, and that guarantees can be improved when these parameters have values close to 1.

Let x and y be vectors from the d-dimensional Euclidean unit sphere $\mathbb{S}^{d-1}$. When represented in a computer with limited precision floating or fixed-point numbers, the precision we can obtain when computing the inner product $\langle x, y \rangle$ will of course depend on the precision of the representation of x and y. Suppose we round coordinates x and y to the nearest integer multiple of ε/d, for some parameter $\varepsilon > 0$, to produce "uniformly quantized" vectors x' and y'. Then it is easy to see that the difference between $\langle x, y \rangle$ and $\langle x', y' \rangle$ is at most ε. The space required to store each coordinate $x'_i, y'_i \in [-1, +1]$ is $\lceil \log_2(2d/\varepsilon) \rceil$ bits, so we get $d \log_2(d/\varepsilon) + O(d)$ bits in total using standard, uniform quantization. On the upper bound side we know that the number of bits per dimension can be made independent of d. The following lemma appears to be folklore – a proof can be found in [3, Theorem 4.1].

▶ **Lemma 1.** *For every $\varepsilon > 0$ there exists a mapping $\mathcal{E} : S^{d-1} \to \{0,1\}^\ell$, where $\ell = d \log_2(1/\varepsilon) + O(d)$ such that $\langle x, y \rangle$ can be estimated from $\mathcal{E}(x)$ and $\mathcal{E}(y)$ with additive error at most ε.*

In this paper we ask if this space usage is optimal for the problem of distinguishing between two specific inner product values. Our methods will work through a decoding function that produces a unit vector from a bit representation (i.e., the approximation is a result of quantizing the input vectors). Specifically, we consider the following problem:

▶ **Definition 2.** *For positive integers d and ℓ, and $\alpha, \beta \in [0,1]$ with $\alpha > \beta$ the (α, β, d, ℓ)-INNERPRODUCT problem is to construct mappings $\mathcal{E} : \mathbb{S}^{d-1} \to \{0,1\}^\ell$ and $\mathcal{D} : \{0,1\}^\ell \to \mathbb{S}^{d-1}$ and choose $t \in \mathbb{R}$, such that for every choice of unit vectors $x, y \in \mathbb{S}^{d-1}$ we have:*

$$\langle x, y \rangle \geq \alpha \implies \langle \mathcal{D}(\mathcal{E}(x)), \mathcal{D}(\mathcal{E}(y)) \rangle \geq t \text{ and}$$
$$\langle x, y \rangle \leq \beta \implies \langle \mathcal{D}(\mathcal{E}(x)), \mathcal{D}(\mathcal{E}(y)) \rangle < t \ .$$

We refer to the parameter ℓ as the space usage *of a construction. Whenever d and ℓ are understood from the context we omit them and talk about the (α, β)-INNERPRODUCT problem.*

On the upper bound side our main technical lemma is the following:

▶ **Theorem 3.** *(α, β)-INNERPRODUCT can be solved using space $\ell = d \log_2 \left(\frac{\sqrt{1-\beta}}{\alpha - \beta} \right) + O(d)$.*

Theorem 3 upper bounds the space needed to approximate inner products between unit vectors. For example we can distinguish pairs with inner product $\alpha = 1 - \varepsilon$ from pairs with inner product less than $\beta = 1 - 2\varepsilon$ using space $\frac{d}{2} \log_2 \frac{1}{\varepsilon} + O(d)$. The problem is closely linked to estimation, so it is unsurprising that it matches the bound in Lemma 1 for $\alpha - \beta = \varepsilon$ in the worst case of $\beta = 0$. What is interesting is that for β close to 1 we get improved constants, saving up to a factor 2 on the space when α approaches 1.

Our proof uses a variant of *pyramid vector quantization* [15] and the technique is essentially an implementation of a grid-based ε-net as described in [3], though the analysis is different. The exposition is supposed to be self-contained, and in particular we do not assume that the reader is familiar with pyramid vector quantization or ε-nets.

Finally, we show a tight lower bound. Consider a communication protocol where Alice is given $x \in \mathbb{S}^{d-1}$ and Bob is given $y \in \mathbb{S}^{d-1}$. For parameters $\alpha, \beta \in (0,1)$, with $\alpha = \beta + \varepsilon$, known to both parties, how many bits of information does Alice need to send to Bob in order for Bob to be able to distinguish the cases $\langle x, y \rangle \geq \alpha$ and $\langle x, y \rangle \leq \beta$? Specifically, how many bits must Alice send, in the worst case over all vectors x, to allow Bob to answer correctly for every vector y? We note that a solution for the (α, β)-INNERPRODUCT problem implies a communication protocol using ℓ bits, but our lower bound applies more generally to any one-way communication protocol, not necessarily based on quantization.

▶ **Theorem 4.** *For each choice of $\alpha, \beta \in (0,1)$ with $\alpha > \beta$, suppose that there exists a mapping $\mathcal{E} : S^{d-1} \to \{0,1\}^\ell$ such that for all $x, y \in S^{d-1}$ we can determine from $\mathcal{E}(x)$ and $\mathcal{E}(y)$ whether $\langle x, y \rangle \leq \beta$ or $\langle x, y \rangle \geq \alpha$ (or output anything if $\langle x, y \rangle \in (\beta, \alpha)$). Then $\ell \geq d \log_2 \left(\frac{\sqrt{1-\beta}}{\alpha - \beta} \right) - O(d)$.*

This matches the upper bound up to the additive term of $O(d)$ bits.

2   Further related work

Motivating applications.   Calculating the inner product of two vectors is a frequently used sub-routine in linear algebra, and many machine learning algorithms rely heavily on inner product calculation. For example, the inner loop of algorithms for training of complex neural networks uses millions and millions of inner product computations. Often what is ultimately learned is an embedding onto a high dimensional unit sphere where the inner product can be used directly as a similarity measure.

In such large scale computations the bottleneck is often the limited bandwidth of the hardware in question, and having slightly smaller vector representations can massively improve the execution time. This gives rise to the idea of computing inner products with reduced precision. Recently, several studies showed that deep neural networks can be trained using low precision arithmetic, see e.g. [12, 18, 19]. This has led to a new generation of software and reduced-precision hardware for machine learning algorithms:

- NVIDIA's TensorRT GPU framework and Google's TensorFlow and Tensor Processing Unit, that both operate with 8- or 16-bit fixed point number representations, and
- Intel's Nervana processor that uses the so-called *FlexPoint* vector representation [28], combining 16-bit uniform quantization with a shared exponent that allows representing vectors in a large dynamic range of magnitudes.

From a theoretical point of view these hardware representations use at least $\log_2 d \pm O(1)$ more bits per dimension than what is required to ensure a given additive error.

Reduced precision inner products have also been employed in knowledge discovery [9] and similarity search [17, 25].

Dimensionality reduction.   There is a large literature on the space complexity of estimating Euclidean distances, usually studied in the setting where a certain failure probability $\delta > 0$ is allowed, and with number of *dimensions* (rather than bits) as the measure of space. For certain "random projection" mappings $f : \mathbb{R}^d \to \mathbb{R}^D$ one can estimate the Euclidean distance $||x - y||_2$ from $f(x)$ and $f(y)$ up to a multiplicative error of $1 + \varepsilon$, with failure probability δ. It is known that using $D = O(\log(\delta^{-1})\varepsilon^{-2})$ dimensions is necessary [24, 27] and sufficient [26]. For unit vectors this implies an approximation of inner products with $O(\varepsilon)$ additive error through the identity

$$\langle x, y \rangle = \tfrac{1}{2}(||x||_2^2 + ||y||_2^2 - ||x - y||_2^2) \ . \tag{1}$$

Using (a specific type of) random projections to estimate inner products, with an additive error guarantee, is known as "feature hashing" [42].

Indyk et al. [20, 21] considered the bit complexity of representing all distances, up to a given relative error $1 + \varepsilon$, within a given set S of n vectors in $\mathbb{R}^d$. For this problem one can assume without loss of generality that $d = O(\varepsilon^{-2} \log n)$, using dimension reduction. Suppose that we only need to preserve distances of unit vectors up to an additive ε, which implies that inner products are preserved up to $O(\varepsilon)$. Then for $d = \varepsilon^{-2} \log n$ the space usage per point of the method described in [21] is $O(d \log(1/\varepsilon))$. This is within a constant factor of our upper bound, but not directly comparable to our result which works for all unit vectors. Recently, Indyk and Wagner [22] studied the space required to solve the d-dimensional Euclidean $(1 + \varepsilon)$-approximate nearest neighbor problem in the setting where vector coordinates are integers in a bounded range (e.g. of size $n^{O(1)}$). While this method gives guarantees for new vectors outside of S their method is randomized and can fail to correctly determine an approximate nearest neighbor, while our method is deterministic.

Vector quantization.   In a nutshell, vector quantization [16] is the process of mapping vectors in a space (usually a bounded subset of Euclidean space) to the nearest in a finite set of vectors Q. The goal is to minimize the size of Q and the distance between vectors and their quantized versions, often with respect to a certain distribution of source (or input) vectors. Fischer first described *pyramid vector quantization* [15], showing that it is near-optimal for Laplacian sources. Since high-dimensional Laplacian vectors have lengths that are tightly concentrated around the expectation, it is natural to speculate if the method is also near-optimal for fixed-length (or unit) vectors. It turns out to be easier to analyze a variant of pyramid vector quantization for which we can show that this is indeed the case. This is described in section 3.

Quantization methods have previously been used to speed up nearest neighbor search. The technique of *product quantization* [25] has been particularly successful for this application. Product quantization uses an initial random rotation of input vectors followed by application of an optimal quantization method on low-dimensional blocks. Since the size of the codebook is fixed for each block the resulting quantization error cannot be bounded with probability 1.

Quantization of the unit sphere has been studied in complexity theory as *ε-nets for spherical caps*. Rabani and Shpilka [33] give a construction in which $|Q|$ is polynomially related to the best size possible with a given quantization error. Along and Klartag [3] use such nets to achieve $|Q|$ that is within a factor $\exp(O(d))$ of optimal, improving [33] whenever the quantization error is $o(1)$, such that $|Q|$ must be superexponential in d.

In the literature on machine learning (and its application areas) there is a myriad of methods for learning a data-dependent quantization mapping that exploits structure in a data set to decrease the quantization error. We refer to the survey of Wang et al. [41] for details. In contrast to such methods, we seek guarantees for all vectors, not just vectors from a given data set or distribution.

Communication complexity.   Consider a communication protocol in which Alice and Bob are given unit vectors $x, y \in \left\{ \pm \frac{1}{\sqrt{d}} \right\}^d$ and need to approximate $\langle x, y \rangle$. The *gap hamming* problem is the special case where the task is to distinguish between cases of weak positive and negative correlation: Is $\langle x, y \rangle > 1/\sqrt{d}$ or is $\langle x, y \rangle < -1/\sqrt{d}$? This problem is known to require $\Omega(d)$ bits of communication [10, 23, 36], even for randomized protocols with error probability, say, $1/3$. In turn, this implies a lower bound for the space complexity of estimating inner products, since a space complexity of ℓ bits implies a (one-way) communication protocol using ℓ bits of communication. The lower bound extends to arbitrary thresholds α and β with $\alpha - \beta = \Theta(1/\sqrt{d})$ by translation. In this paper we consider general unit vectors $x, y \in \mathbb{R}^d$ and are able to show a higher lower bound of $\frac{1}{2} d \log_2 d - O(d)$ bits for distinguishing inner products of distance $\varepsilon = \Theta(1/\sqrt{d})$.

3 Upper bound

We use a well-known grid-based rounding method to construct our representation, see e.g. [3, 15]. For completeness we provide a simple, self-contained description of a representation and show that it has the properties described in Theorem 3. The grid resolution is controlled by a parameter $\delta \in [0, 1]$. For every vector $x \in \mathbb{R}^d$ let $f(x) \in \mathbb{R}^d$ be defined by

$$f(x) = x'/||x'||_2, \text{ where } x_i' = \left\lfloor x_i \frac{\sqrt{d}}{\delta} + \frac{1}{2} \right\rfloor \frac{\delta}{\sqrt{d}} \text{ for } i = 1, \ldots, d \ . \tag{2}$$

It is clear that the number of bits for storing a single integer coordinate x_i' can be large in high dimensions, up to $\log_2(2\sqrt{d}/\delta)$ bits, but we can give a much better bound on the *average* number of bits per coordinate. If $\|x\| \leq 1$ we can store $f(x)$ using $\ell = d\log_2(1/\delta) + O(d)$ bits of space. To compute x' it suffices to know the integers $z_i = \lfloor x_i \frac{\sqrt{d}}{\delta} + \frac{1}{2}\rfloor$. We first allocate d bits to store the set $\{i \mid z_i < 0\}$, such that it only remains to store the sequence of absolute values $|z_1|, \ldots, |z_d|$. Next, using $\|x\|_2 \leq 1$ we observe that

$$\sum_{i=1}^{d} |z_i| \leq \|x\|_1 \frac{\sqrt{d}}{\delta} + d/2 \leq \sqrt{d}\,\|x\|_2 \frac{\sqrt{d}}{\delta} + d/2 \leq d/\delta + d/2 \ .$$

Thus if we set $s = \lfloor d/\delta + d/2 \rfloor$ we can encode $|z_1|, \ldots, |z_d|$ by specifying a partitioning of s elements into $d+1$ parts. There are $\binom{s+d}{d}$ such partitionings so we can assign each vector a unique representation of $\ell = \left\lceil \log_2 \binom{s+d}{d}\right\rceil + d = d\log_2(1/\delta) + O(d)$ bits.

Before proving Theorem 3 we give a simple space bound for distance distortion which is useful in it own right.

▶ **Lemma 5.** *For $\delta \leq 1$ and every choice of $x, y \in \mathbb{R}^d$, defining f according to (2) we have:*

$$\|x - y\|_2 - \delta \leq \|f(x) - f(y)\|_2 \leq \|x - y\|_2 + \delta \ .$$

Proof. Observe that $|x_i - x_i'| \leq \frac{\delta}{2\sqrt{d}}$. This means that:

$$\|x - x'\|_2 = \sqrt{\sum_{i=1}^{d}(x_i - x_i')^2} \leq \sqrt{\sum_{i=1}^{d}\frac{\delta^2}{4d}} = \frac{\delta}{2} \ .$$

Since $\delta \leq 1$ the angle between x and x' is bounded by $\pi/3$, and hence $\|x - f(x)\|_2 \leq 1$. This implies that $\|x - f(x)\|_2^2 = 2 - 2\langle x, x'/\|x'\|_2\rangle \leq 1 + \|x'\|_2 - 2\langle x, x'\rangle = \|x - x'\|_2^2$, and in particular we get $\|x - f(x)\|_2 \leq \|x - x'\|_2 \leq \delta/2$. Finally, using the triangle inequality:

$$\|f(x) - f(y)\|_2 \leq \|x - f(x)\|_2 + \|x - y\|_2 + \|y - f(y)\|_2 \leq \|x - y\|_2 + \delta, \text{ and}$$
$$\|f(x) - f(y)\|_2 \geq \|x - y\|_2 - \|x - f(x)\|_2 - \|y - f(y)\|_2 \geq \|x - y\|_2 - \delta \ . \qquad \blacktriangleleft$$

We are now ready to prove Theorem 3.

Proof of Theorem 3. Let the encoding function $\mathcal{E}(\cdot)$ map a vector x to the ℓ-bit representation of x' as defined in (2). The decoding function $\mathcal{D}(\cdot)$ is defined such that $\mathcal{D}(\mathcal{E}(x)) = f(x)$.

By Lemma 5 we have:

$$\max\{0, \|x - y\| - \delta\} \leq \|f(x) - f(y)\| \leq \|x - y\| + \delta \ .$$

Using the distance bounds and the identity (1) several times we get:

$$\langle f(x), f(y)\rangle = \tfrac{1}{2}(\|f(x)\|^2 + \|f(y)\|^2 - \|f(x) - f(y)\|^2)$$
$$\leq \min\{1, \tfrac{1}{2}(2 + 2\|x - y\|\delta - \|x - y\|^2 - \delta^2)\}$$
$$= \min\{1, \langle x, y\rangle + \|x - y\|\delta - \delta^2/2\}, \text{ and}$$

$$\langle f(x), f(y)\rangle \geq \tfrac{1}{2}(2 - 2\|x - y\|\delta - \|x - y\|^2 - \delta^2)$$
$$= \langle x, y\rangle - \|x - y\|\delta - \delta^2/2 \ .$$

We can then see

$$\langle x, y\rangle \geq \alpha \implies \langle \mathcal{D}(\mathcal{E}(x)), \mathcal{D}(\mathcal{E}(y))\rangle \geq \alpha - \delta\sqrt{2 - 2\alpha} - \delta^2/2 \text{ and}$$

$$\langle x, y\rangle \leq \beta \implies \langle \mathcal{D}(\mathcal{E}(x)), \mathcal{D}(\mathcal{E}(y))\rangle \leq \min\{1, \beta + \delta\sqrt{2 - 2\beta} - \delta^2/2\}$$

Setting $\delta = \frac{\alpha - \beta}{2\sqrt{2 - 2\beta}}$ and $t = \alpha - \delta\sqrt{2 - 2\alpha} - \delta^2/2$ we get a valid solution to the (α, β)-INNERPRODUCT problem.   ◀

The same grid, as specified by δ, works for every (α', β')-INNERPRODUCT instance (with a suitable choice of threshold t) as long as

$$\delta < \frac{\alpha' - \beta'}{\sqrt{2 - 2\alpha'} + \sqrt{2 - 2\beta'}} \ . \tag{3}$$

Note that Lemma 1 also follows from Theorem 3: For a given inner product value p consider $\alpha' = p + \varepsilon/2$ and $\beta' = p - \varepsilon/2$. Setting $\delta = \varepsilon/4$, we satisfy (3) for all p and get that for every choice of unit vectors $x, y \in \mathbb{S}^{d-1}$:

$$\langle x, y\rangle - \varepsilon \leq \langle \mathcal{D}(\mathcal{E}(x)), \mathcal{D}(\mathcal{E}(x))\rangle \leq \langle x, y\rangle + \varepsilon \ .$$

4   Lower bound

Consider a one-way communication protocol solving the (α, β)-INNERPRODUCT problem where Alice sends a string $\mathcal{E}(x)$ to Bob, and Bob must be able to output a real number $p(\mathcal{E}(x), y)$ and a threshold $t \in \mathbb{R}$ such that

$$\langle x, y\rangle \geq \alpha \implies p(\mathcal{E}(x), y) \geq t, \text{ and}$$

$$\langle x, y\rangle \leq \beta \implies p(\mathcal{E}(x), y) < t \ .$$

Note that there is no requirement on $p(\mathcal{E}(x), y)$ whenever $\langle x, y\rangle \in (\beta, \alpha)$. We wish to answer the following question: How many bits must Alice send, in the worst case over all vectors x, to allow Bob to answer correctly for every vector y?

Let $d > 1$ be an integer. For $\varepsilon > 0$ and $x \in \mathbb{R}^d$ let $B_\varepsilon(x) = \{y \in \mathbb{R}^d \mid ||x - y||_2 \leq \varepsilon\}$ be the ball of radius ε centered at x, let $B_1 = B_1(0)$ be the unit ball centered at the origin, and denote by $\mathrm{cap}_\Theta(x) = \{y \in \mathbb{S}^{d-1} \mid \langle x, y\rangle \geq \cos\Theta\}$ the unit spherical cap around x with polar angle Θ.

4.1   Preliminaries

The gamma function is an extension of the factorial function to complex numbers. We will need the following formulas for the value of the gamma function on integers and half-integers (see e.g. [40]). For integer $n > 0$:

$$\Gamma(n + \tfrac{1}{2}) = \frac{(2n)!\sqrt{\pi}}{2^{2n}n!}$$

$$\Gamma(n + 1) = n! \tag{4}$$

▶ **Lemma 6.** *For integer $d > 1$, $\Gamma(d/2 + \tfrac{1}{2})/\Gamma(d/2 + 1) > 1/(3\sqrt{d})$.*

Proof. We use Stirling's approximation to the factorial:

$$\sqrt{2\pi}n^{n+1/2}e^{-n} \leq n! \leq e\,n^{n+1/2}e^{-n}$$

Together with (4) we get, for even d:

$$\Gamma(d/2 + \tfrac{1}{2})/\Gamma(d/2+1) = \frac{d!\sqrt{\pi}}{2^d((d/2)!)^2} \geq \frac{\sqrt{2\pi}\,d^{d+1/2}e^{-d}}{2^d e^2 (d/2)^{d+1}e^{-d}} = \frac{2\sqrt{2\pi}}{e^2\sqrt{d}} > 1/(3\sqrt{d})\ .$$

The case of odd d is similar.   ◀

▶ **Lemma 7.** *Let* $c_d = \frac{\pi^{d/2}}{\Gamma(d/2+1)}$, $r > 0$, *and* $\delta \in (0,1)$. *Then:*

1. *The volume of* B_r^d, *the d-dimensional ball of radius* r, *is* $c_d\,r^d$.
2. *The surface area of* $\mathbb{S}_r^{d-1}$, *the d-dimensional sphere of radius* r, *is* $c_d\,d\,r^{d-1}$.
3. *The surface area of* $cap_\Theta^d(x)$, *a unit spherical cap with polar angle* Θ, *is at most*

$$c_{d-1}\,d\,(2(1 - \cos\Theta))^{(d-1)/2}\ .$$

Proof. Volume bound 1. is standard, see e.g. [32, page 11]. Differentiating with respect to r gives the surface area in line 2. For the upper bound 3. we express the surface area as an integral. Let $r(h) = \sqrt{h(2 - h)} < \sqrt{2h}$ be the radius of the $d-1$-dimensional sphere at the base of the cap of height h. Note that the sphere has surface area $c_{d-1}\,d\,r(x)^{d-1}$. Integrating over cap heights from 0 to $1 - \cos\Theta$ we bound the surface area:

$$\int_0^{1-\cos\Theta} c_{d-1}\,d\,r(x)^{d-1}dx \leq (1 - \cos\Theta)c_{d-1}\,d\,(2(1 - \cos\Theta))^{(d-1)/2}\ .$$

The inequality uses that $r(x) \leq r(1 - \cos\Theta) \leq \sqrt{2(1 - \cos\Theta)}$ for $x \in [0, 1 - \cos\Theta]$.   ◀

▶ **Lemma 8.** *For every* $\Theta \in (0, \pi/2)$ *there exists a code* $\mathcal{C}_\Theta \subset S^{d-1}$ *of size*

$$|\mathcal{C}_\Theta| \geq (2(1 - \cos\Theta))^{(d-1)/2}/(3\sqrt{d})$$

such that for all $x, y \in \mathcal{C}_\Theta$ *with* $x \neq y$ *we have* $\langle x, y \rangle \geq \cos\Theta$.

Proof. We follow the outline of the standard non-constructive proof of the Gilbert-Varshamov bound. That is, we argue that $\mathcal{C}_\Theta$ can be constructed in a greedy manner by adding an additional point from

$$\mathbb{S}^{d-1} \backslash \bigcup_{x \in \mathcal{C}_\Theta} cap_\Theta^d(x)$$

until this set is empty. Clearly this construction produces a set $\mathcal{C}_\Theta$ with the property that every pair of points have angle greater than Θ. We observe that the procedure can stop only when the area of $\bigcup_{x \in \mathcal{C}_\Theta} cap_\Theta(x)$ exceeds that of $\mathbb{S}^{d-1}$. The number of iterations, and thus the size of $\mathcal{C}_\Theta$ is at least the ratio between the surface area of the unit sphere and a spherical cap, $cap_\Theta(\cdot)$. In turn, this is lower bounded by the ratio of bound number 2 (with $r = 1$) and 3 from Lemma 7. Using Lemma 6 we get:

$$\frac{c_d\,d}{c_{d-1}\,d\,(2(1 - \cos\Theta))^{(d-1)/2}} \geq (2(1 - \cos\Theta))^{-(d-1)/2}/(3\sqrt{d})\ .$$ ◀

4.2   Space complexity

Define $\Theta = \arccos \beta - \arccos \alpha$. We claim that for every pair of vectors $x_1, x_2 \in \mathbb{S}^{d-1}$ with angle $\theta = \arccos \langle x_1, x_2 \rangle \geq \Theta$ there exists a vector $y \in \mathbb{S}^{d-1}$ such that $\langle x_1, y \rangle = \beta$ and $\langle x_2, y \rangle \geq \alpha$. Specifically, let

$$y = y(x_1, x_2) = \left(\beta - \sqrt{1 - \beta^2} \, \tfrac{\cos \theta}{\sin \theta} \right) x_1 + \tfrac{\sqrt{1 - \beta^2}}{\sin \theta} x_2 \ . \tag{5}$$

To see that y is indeed a unit vector we compute $\|y\|_2^2 = \langle y, y \rangle$. Since $\langle x_1, x_2 \rangle = \cos \theta$ (by definition of θ) and $\langle x_1, x_1 \rangle = \langle x_2, x_2 \rangle = 1$ (since x_1 and x_2 are unit vectors) we get:

$$\langle y, y \rangle = \left(\beta - \sqrt{1 - \beta^2} \, \tfrac{\cos \theta}{\sin \theta} \right)^2 + \left(\tfrac{\sqrt{1 - \beta^2}}{\sin \theta} \right)^2 + 2 \left(\beta - \sqrt{1 - \beta^2} \, \tfrac{\cos \theta}{\sin \theta} \right) \left(\tfrac{\sqrt{1 - \beta^2}}{\sin \theta} \right) \cos \theta$$

$$= \beta^2 + \tfrac{1 - \beta^2}{\sin^2 \theta} (1 - \cos^2 \theta)$$

$$= \beta^2 + (1 - \beta^2) = 1 \ .$$

The third equality uses the Pythagorean identity $\cos^2 \theta + \sin^2 \theta = 1$. Next, we check that y has the claimed inner products with x_1 and x_2:

$$\langle x_1, y \rangle = \left(\beta - \sqrt{1 - \beta^2} \, \tfrac{\cos \theta}{\sin \theta} \right) + \sqrt{1 - \beta^2} \, \tfrac{\cos \theta}{\sin \theta} = \beta,$$

$$\langle x_2, y \rangle = \left(\beta - \sqrt{1 - \beta^2} \, \tfrac{\cos \theta}{\sin \theta} \right) \cos \theta + \tfrac{\sqrt{1 - \beta^2}}{\sin \theta}$$

$$= \langle (\beta, \sqrt{1 - \beta^2}), (\cos \theta, \tfrac{1 - \cos^2 \theta}{\sin \theta}) \rangle$$

$$= \langle (\beta, \sqrt{1 - \beta^2}), (\cos \theta, \sin \theta) \rangle \geq \alpha \ .$$

The final inequality follows since the angle between the vectors $(\beta, \sqrt{1 - \beta^2})$ and $(\cos \theta, \sin \theta)$ is at most $\arccos(\beta) - \theta \leq \arccos(\beta) - \Theta = \arccos \alpha$.

Now consider the code $\mathcal{C}_\Theta$. For distinct $x_1, x_2 \in \mathcal{C}_\Theta$ we must have, for $y = y(x_1, x_2)$ as defined in (5), $p(e(x_1), y) < t \leq p(e(x_2), y)$, which means that $e(x_1) \neq e(x_2)$. Hence $R = \{\mathcal{E}(x) \mid x \in \mathbb{S}^{d-1}\}$ must contain at least $|\mathcal{C}_\Theta|$ binary strings, and in particular

$$\ell \geq \log_2 |\mathcal{C}_\Theta| \geq \log_2 \left((2(1 - \cos \Theta))^{-(d-1)/2} / (3\sqrt{d}) \right)$$

$$\geq \tfrac{d}{2} \log_2 \left(\tfrac{1}{1 - \cos \Theta} \right) - O(d)$$

$$\geq d \log_2 (1/\Theta) - O(d) \ .$$

For the final inequality we use that $1 - \cos \Theta \leq \Theta^2$, which holds when $\Theta \in (0, \pi/2)$. To finish the proof we will show that whenever $0 \leq \beta \leq \alpha \leq 1$:

$$\Theta = \arccos \beta - \arccos \alpha \leq \tfrac{\pi}{2} \tfrac{\alpha - \beta}{\sqrt{1 - \beta}} \ . \tag{6}$$

For each fixed $\beta \in [0, 1]$ we must show that $\arccos \alpha \geq \arccos \beta - \tfrac{\pi}{2} \tfrac{\alpha - \beta}{\sqrt{1 - \beta}}$ for all $\alpha \in [\beta, 1]$. Since $\alpha \mapsto \arccos \alpha$ is concave for $\alpha \in [0, 1]$, and the function $\alpha \mapsto \arccos \beta - \tfrac{\pi}{2} \tfrac{\alpha - \beta}{\sqrt{1 - \beta}}$ is linear, it suffices to check the inequality at the endpoints where $\alpha = \beta$ and $\alpha = 1$. In the former case we clearly get equality. In the latter we use the fact $\arccos \beta - \tfrac{\pi}{2} \sqrt{1 - \beta} \leq 0$ for $\beta \in [0, 1]$ to see that the inequality (6) holds.

This proves that $\ell \geq d \log_2 \left(\tfrac{\sqrt{1 - \beta}}{\alpha - \beta} \right) - O(d)$ bits are needed, establishing Theorem 4.

Table 1 Space usage and absolute inner product error using the method of Section 3 on pairs of vectors from various real-life data sets. The space usage is measured against a baseline of using a 32-bit floating point numbers to represent each of the d dimensions. Observed errors are smaller than our worst-case bound of 4δ, probably due to cancellation effects.

Dataset	d	δ	space	median	90th pct.	max
MNIST	784	0.1	16%	0.0019	0.0052	0.0165
SIFT	128	0.1	16%	0.0034	0.0084	0.0181
DLIB	128	0.1	16%	0.0028	0.0071	0.0135
MNIST	784	0.05	19%	0.0008	0.0027	0.0103
SIFT	128	0.05	19%	0.0019	0.0044	0.0091
DLIB	128	0.05	19%	0.0012	0.0033	0.0075
MNIST	784	0.01	26%	0.0002	0.0007	0.0027
SIFT	128	0.01	26%	0.0004	0.0008	0.0100
DLIB	128	0.01	26%	0.0002	0.0005	0.0011

5 Experiments

Since the encoding in our upper bound is potentially practical, we evaluated the accuracy of the method experimentally on several data sets. We also computed the space usage of each set of vectors, based on an optimal encoding of the method of Section 3 with various values of parameter δ, and compared it to a baseline of using 32-bit floating point numbers.

We considered three data sets, MNIST (handwritten digits), SIFT (image features), and DLIB-FACES (neural net embeddings of faces on a 128-dimensional unit sphere). The data sets are chosen to span different distributions of entry magnitude. Whereas MNIST vectors are approximately sparse, DLIB vectors have smoothly distributed magnitudes. MNIST and SIFT are not natively unit vectors, so we normalized the vectors prior to encoding.

We computed the inner product error for 2000 vector pairs in each data set. Table 1 shows the maximum absolute error observed when calculating inner products using the decoded vectors compared to using the original vectors. It also shows the median absolute error and the error at the 90th percentile.

In all cases the observed errors are well below the worst case bound of $\varepsilon = 4\delta$. This can partly be explained by sparsity of vectors, since our method represents zero entries in vectors with no error. Also, while the effect of d rounding errors is d times the individual error in the worst case (which is what our theoretical results bound), in the typical case errors will tend to cancel since not all errors go in the same direction. Heuristically we could imagine that errors are independent and random, in which case we would expect the sum of all errors to be proportional to $\sqrt{d}$ rather than d.

6 Conclusion

We have established tight upper and lower bounds for the problem of representing unit vectors such that inner products can be estimated within a given additive error (with probability 1). An interesting possibility would be to consider relative error estimates of Euclidean distances (as in the recent work [22]) while not allowing any failure probability. Another potential direction would be to achieve provably smaller *expected* error, while preserving the worst-case guarantees, by applying an initial random rotation to all data vectors.

References

1 Amir Abboud, Aviad Rubinstein, and R. Ryan Williams. Distributed PCP Theorems for Hardness of Approximation in P. In *58th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 25–36, 2017. `doi:10.1109/FOCS.2017.12`.

2 Thomas Dybdahl Ahle, Rasmus Pagh, Ilya Razenshteyn, and Francesco Silvestri. On the complexity of inner product similarity join. In *Proceedings of the 35th ACM Symposium on Principles of Database Systems (PODS)*, pages 151–164. ACM, 2016.

3 Noga Alon and Bo'az Klartag. Optimal compression of approximate inner products and dimension reduction. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 639–650. IEEE, 2017.

4 David C. Anastasiu and George Karypis. L2AP: Fast cosine similarity search with prefix l-2 norm bounds. In *30th IEEE International Conference on Data Engineering (ICDE)*, pages 784–795. IEEE, 2014.

5 Nikolaus Augsten and Michael H. Böhlen. *Similarity Joins in Relational Database Systems*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2013. `doi:10.2200/S00544ED1V01Y201310DTM038`.

6 Martin Aumüller, Tobias Christiani, Rasmus Pagh, and Michael Vesterli. PUFFINN: parameterless and universally fast finding of nearest neighbors. In *27th Annual European Symposium on Algorithms (ESA)*, pages 10:1–10:16, 2019. `doi:10.4230/LIPIcs.ESA.2019.10`.

7 Raghavendran Balu, Teddy Furon, and Hervé Jégou. Beyond "project and sign" for cosine estimation with binary codes. In *IEEE International Conference on Acoustics, Speech and Signal Processing, (ICASSP)*, pages 6884–6888, 2014. `doi:10.1109/ICASSP.2014.6854934`.

8 Roberto J. Bayardo, Yiming Ma, and Ramakrishnan Srikant. Scaling up all pairs similarity search. In *Proceedings of 16th international conference on World Wide Web (WWW)*, pages 131–140. ACM, 2007.

9 Davis W. Blalock and John V. Guttag. Bolt: Accelerated data mining with fast vector compression. In *Proceedings of 23rd ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 727–735. ACM, 2017.

10 Amit Chakrabarti and Oded Regev. An optimal lower bound on the communication complexity of gap-hamming-distance. *SIAM Journal on Computing*, 41(5):1299–1317, 2012.

11 Moses Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 380–388, 2002. `doi:10.1145/509907.509965`.

12 Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *International Conference on Machine Learning (ICML)*, pages 2285–2294, 2015.

13 Tobias Christiani, Rasmus Pagh, and Johan Sivertsen. Scalable and Robust Set Similarity Join. In *34th IEEE International Conference on Data Engineering (ICDE)*, pages 1240–1243, 2018. `doi:10.1109/ICDE.2018.00120`.

14 David Clayton, Christopher Patton, and Thomas Shrimpton. Probabilistic Data Structures in Adversarial Environments. In *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, pages 1317–1334, 2019. `doi:10.1145/3319535.3354235`.

15 Thomas Fischer. A pyramid vector quantizer. *IEEE transactions on information theory*, 32(4):568–583, 1986.

16 Allen Gersho and Robert M. Gray. *Vector quantization and signal compression*, volume 159. Springer Science & Business Media, 2012.

17 Ruiqi Guo, Sanjiv Kumar, Krzysztof Choromanski, and David Simcha. Quantization based fast inner product search. In *Artificial Intelligence and Statistics*, pages 482–490, 2016.

18 Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International Conference on Machine Learning (ICML)*, pages 1737–1746, 2015.

19 Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

20 Piotr Indyk, Ilya Razenshteyn, and Tal Wagner. Practical Data-Dependent Metric Compression with Provable Guarantees. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2614–2623, 2017.

21 Piotr Indyk and Tal Wagner. Near-optimal (euclidean) metric compression. In *Proceedings of 28th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 710–723. SIAM, 2017.

22 Piotr Indyk and Tal Wagner. Approximate Nearest Neighbors in Limited Space. In *Proceedings of 31st Conference On Learning Theory (COLT)*, volume 75 of *Proceedings of Machine Learning Research*, pages 2012–2036. PMLR, 2018.

23 Piotr Indyk and David P. Woodruff. Tight lower bounds for the distinct elements problem. In *44th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 283–288. IEEE, 2003.

24 Thathachar S. Jayram and David P. Woodruff. Optimal bounds for Johnson-Lindenstrauss transforms and streaming problems with subconstant error. *ACM Transactions on Algorithms (TALG)*, 9(3):26, 2013.

25 Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2011.

26 William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

27 Daniel Kane, Raghu Meka, and Jelani Nelson. Almost optimal explicit Johnson-Lindenstrauss families. In *International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pages 628–639. Springer, 2011.

28 Urs Köster, Tristan Webb, Xin Wang, Marcel Nassar, Arjun K. Bansal, William Constable, Oguz Elibol, Stewart Hall, Luke Hornof, Amir Khosrowshahi, Carey Kloss, Ruby J. Pai, and Naveen Rao. Flexpoint: An Adaptive Numerical Format for Efficient Training of Deep Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1742–1752, 2017. URL: `http://papers.nips.cc/paper/6771-flexpoint-an-adaptive-numerical-format-for-efficient-training-of-deep-neural-networks`.

29 Samuel McCauley and Francesco Silvestri. Adaptive MapReduce Similarity Joins. In *Proceedings of the 5th ACM SIGMOD Workshop on Algorithms and Systems for MapReduce and Beyond*, page 4. ACM, 2018.

30 Rasmus Pagh. Large-Scale Similarity Joins With Guarantees (Invited Talk). In *18th International Conference on Database Theory (ICDT)*, pages 15–24, 2015. `doi:10.4230/LIPIcs.ICDT.2015.15`.

31 Rasmus Pagh, Ninh Pham, Francesco Silvestri, and Morten Stöckel. I/O-efficient similarity join. In *23rd Annual European Symposium on Algorithms (ESA)*, pages 941–952, 2015. `doi:10.1007/978-3-662-48350-3_78`.

32 Gilles Pisier. *The volume of convex bodies and Banach space geometry*, volume 94 of *Cambridge Tracts in Mathematics*. Cambridge University Press, 1999.

33 Yuval Rabani and Amir Shpilka. Explicit construction of a small ε-net for linear threshold functions. *SIAM Journal on Computing*, 39(8):3501–3520, 2010.

34 Parikshit Ram and Alexander G. Gray. Maximum inner-product search using cone trees. In *18th ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 931–939, 2012. `doi:10.1145/2339530.2339677`.

35 Venu Satuluri and Srinivasan Parthasarathy. Bayesian locality sensitive hashing for fast similarity search. *Proceedings of the VLDB Endowment*, 5(5):430–441, 2012.

36 Alexander A. Sherstov. The Communication Complexity of Gap Hamming Distance. *Theory of Computing*, 8(1):197–208, 2012.

37 Anshumali Shrivastava and Ping Li. Asymmetric LSH (ALSH) for Sublinear Time Maximum Inner Product Search (MIPS). In *Advances in Neural Information Processing Systems (NIPS)*,

pages 2321–2329, 2014. URL: `http://papers.nips.cc/paper/5329-asymmetric-lsh-alsh-for-sublinear-time-maximum-inner-product-search-mips`.

38 Ryan Spring and Anshumali Shrivastava. Scalable and Sustainable Deep Learning via Randomized Hashing. In *Proceedings of 23rd ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 445–454, 2017. `doi:10.1145/3097983.3098035`.

39 Christina Teflioudi and Rainer Gemulla. Exact and approximate maximum inner product search with LEMP. *ACM Transactions on Database Systems (TODS)*, 42(1):5, 2017.

40 Raimundas Vidunas. EXPRESSIONS FOR VALUES OF THE GAMMA FUNCTION. *Kyushu Journal of Mathematics*, 59(2):267–283, 2005.

41 Jun Wang, Wei Liu, Sanjiv Kumar, and Shih-Fu Chang. Learning to hash for indexing big data—a survey. *Proceedings of the IEEE*, 104(1):34–57, 2016.

42 Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of 26th International Conference on Machine Learning (ICML)*, pages 1113–1120. ACM, 2009.

43 Xiao Yan, Jinfeng Li, Xinyan Dai, Hongzhi Chen, and James Cheng. Norm-Ranging LSH for Maximum Inner Product Search. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2956–2965, 2018. URL: `http://papers.nips.cc/paper/7559-norm-ranging-lsh-for-maximum-inner-product-search`.