Hardness results for the Discrete Gauss Transform

Kyriakos Axiotis, Aleksandar Makelov, Chenyang Yuan

1 Introduction

The *Discrete Gauss Transform (DGT)* is an important computational primitive that arises in various areas of mathematics and its applications. Given a set of n "source" points $A \in \mathbb{R}^d$ that defines a Gaussian kernel and a set of n "query" points $B \in \mathbb{R}^d$, it asks to evaluate the Gaussian kernel at every query point. As an example, in the study of heat equations and fluid dynamics DGT arises as a discretization of certain families of differential equation solutions. It is thus widely used in physical simulations and computer modeling. Furthermore, evaluating and optimizing over Gaussian kernels is important for Machine Learning.

The trivial algorithm for this problem runs in time $O(n^2 d)$ and comprises of explicitly evaluating the Gaussian kernel at each query point. Since there are $\Theta(n^2)$ pairs of points in total and it takes $O(d)$ time to compute the distance between two points, the bound follows. This algorithm, however, is somewhat impractical because of the quadratic dependence on n. In practice, n will correspond to the number of particles or bodies in a physical system, and it can be expected to be large. On the other hand, the dependence on d might not be extremely crucial, at least for physics applications, in which the space can be expected to be intrinsically low-dimensional. (Note, by the way, that this is not the case in most Machine Learning applications, where one expects the dimension to be very large).

In light of the above, it is natural to ask whether there exists an algorithm with much better (ideally linear) dependence on n. This question was answered affirmatively by [GS91], who proposed the *Fast Gauss Transform*, an algorithm that computes an additive ε-approximation to DGT in time $O\left(n\left(\log\frac{1}{\varepsilon}\right)^{O(d)}\right)$. While the dependence on n is optimal, the dependence on d renders the algorithm practical only for applications with very low dimension. As such, a lot of research has been done on devising various heuristics that seem to work somewhat better in practice, but do not enjoy theoretically better guarantees (see e.g. [GW13, RYDG05, TW09]).

As we will see in this exposition, the apparent barrier in the improvement of the runtime of Gauss transform algorithms is inherent, as fast algorithms for this problem would imply faster algorithms for SAT, refuting the well-known Strong Exponential Time Hypothesis.

2 Preliminaries

We assume familiarity with the basics of computational complexity theory. In this section, we describe the computational problems relevant to our reductions. Key to our approach is the following classical problem, known as the Orthogonal Vectors Problem (or **OV** for short):

Problem 2.1 (Orthogonal Vectors problem (monochromatic version))**.** Given n vectors $v_1, \ldots, v_n \in \{0,1\}^d$, are there i, j such that $\langle v_i, v_j \rangle = 0$?

More specifically, we will use the bichromatic version of **OV**, in which we are only concerned with the inner products between vectors from two different sets:

Problem 2.2 (Orthogonal Vectors problem (bichromatic version))**.** Given two sets of n vectors $A = \{v_1, \ldots, v_n\} \in \{0,1\}^d$ and $B = \{u_1, \ldots, u_n\} \in \{0,1\}^d$, are there i, j such that $\langle v_i, u_j \rangle = 0$?

As we will show, the complexity of mono- and bi-chromatic **OV** is the same, so we will call both problems just **OV**. We will also refer to the orthogonal vectors problem for n vectors (or n vectors in each set for the bichromatic version) in dimension d as $\mathbf{OV}(n,d)$ to emphasize the dependence on parameters. Central to our method is the following standard conjecture from fine-grained complexity:

Conjecture 2.3 (Orthogonal vector conjecture)**.** For every $\varepsilon > 0$, there is a $c \geq 1$ such that **OV** cannot be solved in $n^{2-\varepsilon}$ time on instances with $d = c \log n$.

In particular, this conjecture is refuted if there exists an algorithm that solves **OV** in $c \log n$ dimensions, for large enough constant c, in $n^{2-\Omega(1)}$ time. As we will be concerned with various geometric problems where points can have arbitrary coordinates, a stepping stone in our reductions is the following problem, which is a version of **OV** over $\mathbb{Z}$ (here, we only need the bichromatic version):

Problem 2.4 (Hopcroft's problem (bichromatic version))**.** Given two sets of n vectors $A = \{v_1, \ldots, v_n\} \in \mathbb{Z}^d$ and $B = \{u_1, \ldots, u_n\} \in \mathbb{Z}^d$, are there i, j such that $\langle v_i, u_j \rangle = 0$? All input numbers consist of D bits.

As with **OV**, we will also call this problem $\mathbf{Hop}(n,d,D)$ to emphasize the dependence on parameters. The next problem we need is the closest pair problem; here, we again only need the bichromatic variant:

Problem 2.5 (ℓ_2 Bichromatic Closest pair)**.** Given two sets of n vectors $A = \{v_1, \ldots, v_n\} \in \mathbb{R}^d$ and $B = \{u_1, \ldots, u_n\} \in \mathbb{R}^d$, where each coordinate of each vector has $\leq D$ bits, find the pair (i,j) such that the ℓ_2 distance between v_i and u_j is the smallest.

We will call this problem $\mathbf{BCP}(n,d,D)$. Finally, we formally define the central problem that we want to prove hardness of: the approximate Gauss transform. Here, we define it over $\mathbb{Z}$, as this will be sufficient for proving hardness:

Problem 2.6 (Approximate Gauss transform)**.** Given two sets of n vectors $A = \{v_1, \ldots, v_n\} \in \mathbb{Z}^d$ and $B = \{u_1, \ldots, u_n\} \in \mathbb{Z}^d$, and $\varepsilon > 0$, define

$$F(u) = \frac{1}{n} \sum_{i=1}^{n} e^{-\|u-v_i\|_2^2}$$

The approximate Gauss transform problem is to compute $\widehat{F}(u_1), \ldots, \widehat{F}(u_n)$ such that for all $i = 1, \ldots, n$ we have

$$\left|\widehat{F}(u_i) - F(u_i)\right| \leq \varepsilon$$

We will call this problem $\mathbf{GT}(n, d, \varepsilon)$.

2.1 Reductions between monochromatic and bichromatic versions of OV

Here are the promised reductions showing that the mono- and bi-chromatic versions of **OV** are equivalent for our purposes.

Lemma 2.7. An instance of monochromatic $\mathbf{OV}(n, d)$ reduces to an instance of bichromatic $\mathbf{OV}(n, d+1)$ in $O(nd)$ (i.e. linear) time.

Proof. First, search for the zero vector among our input vectors and if it is present, reduce to a trivially true instance of bichromatic **OV**.

Otherwise, construct the vectors $A = \{(v_1, 0), \ldots, (v_n, 0)\}$ and $B = \{(v_1, 1), \ldots, (v_n, 1)\}$ in dimension $d + 1$ - this is our bichromatic **OV** instance. □

Lemma 2.8. An instance of bichromatic $\mathbf{OV}(n, d)$ reduces to an instance of monochromatic $\mathbf{OV}(2n, d+2)$ in $O(nd)$ (i.e. linear) time.

Proof. If our vectors are $A = \{u_1, \ldots, u_n\}$ and $B = \{v_1, \ldots, v_n\}$, our monochromatic **OV** instance consists of the vectors $(u_i, 1, 0)$ and $(v_i, 0, 1)$ for $1 \leq i \leq n$. Note that no two vectors coming from A and no two vectors coming from B can now have dot product 0, so we're in effect searching only over the pairs of vectors in different sets. □

3 Previous work

In this section, we summarize the prior work relevant to our main result about the hardness of the approximate Gauss transform.

3.1 Williams's reduction from OV to Hopcroft's problem

We will use the following reduction of Williams from **OV** to **Hop**:

Lemma 3.1 (Lemma 1.1 from [Wil18])**.** Let $\ell \in [1, d]$. There exists a $nd^{O(d/\ell)}$-time reduction from $\mathbf{OV}(n, d)$ to $d^{O(d/\ell)}$ instances of $\mathbf{Hop}\left(n, \ell + 1, O\left(\frac{d \log d}{\ell}\right)\right)$.

3.2 Chen's improved reduction from OV to Hopcroft's problem

Lijie Chen [Che18] proves the following reduction:

Lemma 3.2 (Lemma 1.17 from [Che18])**.** Let $\ell \in [1, d]$. There is a

$$O\left(n\ell^{O\left(6^{\log^* d} d/\ell\right)} \operatorname{poly}(d)\right)\text{-time}$$

reduction from $\mathbf{OV}(n, d)$ to $\ell^{O\left(6^{\log^* d} d/\ell\right)}$ instances of $\mathbf{Hop}\left(n, \ell+1, O\left(\frac{d}{\ell} \log \ell 6^{\log^* d}\right)\right)$.

3.3 Williams' reduction from Hopcroft's problem to bichromatic closest pair

Williams [Wil18] also gives the following reduction from **Hop** to **BCP**:

Lemma 3.3. There is a

$$O\left(nd^2(D + O(\log d))^2\right)\text{-time}$$

reduction from $\mathbf{Hop}(n, d, D)$ to $\mathbf{BCP}\left(n, d^2, O(D + O(\log d))\right)$.

Proof. Let $A = \{u_1, \ldots, u_n\}$ and $B = \{v_1, \ldots, v_n\}$ be the two sets of vectors. We construct new sets

$$A' = \left\{u_i' = \operatorname{vec}(u_i u_i^T)/\left\|u_i\right\|_2^2 \;\middle|\; 1 \leq i \leq n\right\}$$

and

$$B' = \left\{v_i' = -\operatorname{vec}(v_i v_i^T)/\left\|v_i\right\|_2^2 \;\middle|\; 1 \leq i \leq n\right\}$$

where we regard the matrices as unfolded vectors. Now note that

$$\left\|\operatorname{vec}(u_i u_i^T)\right\|_2^2 = \operatorname{tr}(u_i u_i^T u_i u_i^T) = \operatorname{tr}\left(u_i^T u_i u_i^T u_i\right) = \left\|u_i\right\|_2^4$$

and most importantly

$$\left\langle \operatorname{vec}(u_i u_i^T), -\operatorname{vec}(v_i v_i^T)\right\rangle = -\operatorname{tr}\left(u_i u_i^T v_i v_i^T\right) = -\left\langle u_i, v_i\right\rangle^2$$

which means that

$$\left\|u_i' - v_i'\right\|_2^2 = 2 + \frac{\left\langle u_i, v_i\right\rangle^2}{\left\|u_i\right\|_2^2 \left\|v_i\right\|_2^2}.$$

Thus, finding the closest pair in ℓ_2 between A' and B' and checking if the distance is 2 or more tells us the solution to Hopcroft's problem. Note that the resulting dimension is d^2 and we have n vectors in each set.

Now it remains to deal with the bit complexity of the vectors we produce. Note that if $\left\langle u_i, v_i\right\rangle^2 \neq 0$, then $\left\langle u_i, v_i\right\rangle^2 \geq 1$, which means that for such pairs we have

$$\left\|u_i' - v_i'\right\|_2^2 \geq 2 + \frac{1}{d^2 2^{2D+1}}$$

So now suppose we truncate u'_i and v'_i to D' bits after the binary point; then $\|u'_i - v'_i\|_2^2$ changes by at most about $\frac{d^2}{2^{2D'}}$, so we want pretty much

$$2^{2D'} > d^4 2^{2D}$$

or $D' \geq D + 4\log d$ guarantees that the closest pair instance will be able to detect a dot product of 0.

Finally, we scale up all our points so that they have integer coordinates.

Running time: Observe that to multiply/divide two $O(B)$-bit numbers naively it takes $O(B^2)$ time; therefore,

- computing all the $u_i u_i^T$ and $v_i v_i^T$ takes $O(nd^2D^2)$;
- computing all the $\|u_i\|_2^2, \|v_i\|_2^2$ takes $O(ndD^2)$;
- performing the normalization takes $O(nd^2(D + O(\log d))^2)$;
- scaling up the vectors to $\mathbb{Z}$ takes $O(nd^2D)$

So overall the running time is $O(nd^2(D + O(\log d))^2)$. □

4 The hardness of computing the Discrete Gauss Transform

In this section we give our main results for the inapproximability of the Gauss transform assuming the orthogonal vectors conjecture.

4.1 From bichromatic closest pair to approximate Gauss transform

We start with a reduction from the bichromatic closest pair problem to the approximate Gauss transform that will be used in all our results:

Lemma 4.1. There is a $O\left(ndD\log\log n + d2^D \log n\right)$-time reduction from an instance of $\mathbf{BCP}(n, d, D)$ to an instance of $\mathbf{GT}\left(n, d, \frac{1}{O(n^{O(d2^D)})}\right)$.

Proof. If A and B are the two sets of vectors for the closest pair instance, we run our approximate Gauss transform algorithm on λ-scaled versions A', B' of these sets with approximation ε for λ, ε to be chosen later.

Let d_1 be the closest distance between A and B, and d_2 the second-closest. Note that $d_2^2 \geq d_1^2 + 1$ as we are working over $\mathbb{Z}$. Now, for any $u^* \in A$ such that u^* realizes the shortest distance with some vector in B, we have

$$F(u^*) \geq \frac{1}{n} e^{-\lambda^2 d_1^2}$$

and for any u that doesn't, we have

$$F(u) \leq e^{-\lambda^2 d_2^2}.$$

Our strategy now is to choose λ so that $F(u^*) - \varepsilon > F(u) + \varepsilon$ for any $u \in A$ which doesn't realize the shortest distance to B. Note that this will ensure that when we run the approximate Gauss transform and find the smallest $\widehat{F}(u^*)$ among all $u^* \in A$, it will be for a u^* which realizes the shortest distance to B. Then, for such a u^*, we just enumerate over $v \in B$ and find the shortest distance; this will be the solution to the bichromatic closest pair instance.

So, we want to have

$$e^{-\lambda^2 d_1^2} - ne^{-\lambda^2 d_2^2} \geq 2n\varepsilon$$

Choosing $\lambda = \lceil c\sqrt{\log n}\rceil$ for $c \geq 2$, we have for $n \geq 2$ that

$$\begin{aligned} e^{-\lambda^2 d_1^2} - ne^{-\lambda^2 d_2^2} &\geq e^{-\lambda^2 d_1^2}\left(1 - \frac{1}{n^{c-1}}\right) \\ &= e^{-c^2 d_1^2 \log n}\left(1 - \frac{1}{n^{c-1}}\right) \\ &\geq \frac{1}{2}\frac{1}{n^{c^2 d_1^2}}. \end{aligned}$$

Since $d_1^2 = O(d2^D)$ and $d2^D \geq 1$, it suffices to choose $\varepsilon = O(\frac{1}{n^{O(d2^D)}})$. This finishes the proof.

Running time: We just need to scale the sets A and B by the integer $\lceil c\sqrt{\log n}\rceil$ which has $O(\log\log n)$ bits, so the scaling takes $O(ndD\log\log n)$ time, and we need to write down ε, which takes $O(d2^D \log n)$ time.

□

4.2 Reduction from OV to Gauss Transform in $\log n$ Dimensions

For instances of the Gauss transform where $d = c\log n$, we can give a simple reduction from the bichromatic version of the orthogonal vector conjecture to the approximate Gauss transform problem. Note that in this regime the simple exact algorithm that checks all $\Theta(n^2)$ pairs beats the FGT algorithm.

The idea is to reduce from **OV** to **BCP** directly, without passing through **Hop**, which allows us to avoid the more complex reductions by Williams, and also results in a better dependence on the dimension. Namely, we can avoid the squaring of the dimension from Williams' reduction from **Hop** to **BCP**.

Lemma 4.2. There is an $\tilde{O}(n)$-time reduction from an instance of $\mathbf{OV}(n, c\log n)$ to an instance of $\mathbf{GT}(n, 3d, \varepsilon)$ where

$$\log(1/\varepsilon) = O(\log^2 n).$$

Proof. Given the two sets of n vectors $A = \{u_1, \ldots, u_n\} \subseteq \{0,1\}^d$ and $B = \{v_1, \ldots, v_n\} \subseteq \{0,1\}^d$ in the **OV** instance, we construct new sets $A', B' \subseteq \{0,1\}^{3d}$ such that for each $u \in A$ and $v \in B$ we include u' in A' and v' in B', where

$$\begin{array}{rccccccccccl} u' = [& u_1 & \cdots & u_d & 1-u_1 & \cdots & 1-u_d & 0 & \cdots & 0 &] \\ v' = [& -v_1 & \cdots & -v_d & 0 & \cdots & 0 & 1-v_1 & \cdots & 1-v_d &] \end{array}$$

Note that $\langle u', u' \rangle = \langle v', v' \rangle = d$ and $\langle u', v' \rangle = -\langle u, v \rangle$ by construction. Therefore $\|u' - v'\|^2 = 2d + \langle u, v \rangle$. Moreover, $\langle u, v \rangle \geq 0$ for all u, v since they are in $\{0,1\}^d$. This gives us a reduction from $\mathbf{OV}(n, d)$ to $\mathbf{BCP}(n, 3d, O(1))$, since a dot product of zero between some $u \in A$ and $v \in B$ is equivalent to to the shortest distance between $u' \in A'$ and $v' \in B'$ being $2d$. This reduction runs in $O(nd) = \tilde{O}(n)$-time.

Now we can apply this reduction with $d = c \log n$, and compose it with our reduction from $\mathbf{BCP}(n, 3d, O(1))$ to $\mathbf{GT}(n, 3d, \frac{1}{n^{O(\log n)}})$, which will work in time

$$O(nd \log\log n + d \log n) = O(n \log n \log\log n + \log^2 n) = \widetilde{O}(n)$$

as we wanted to show, and moreover we have $\log(1/\varepsilon) = O(d \log n) = O(\log^2 n)$.

$\square$

In particular, for this lemma we get our first hardness result for the Gauss transform as a direct corollary:

Corollary 4.3. Assuming the **OV** conjecture, for any $\eta, \delta > 0$ and $d \geq c \log n$, there is no algorithm for $\mathbf{GT}(n, d, \varepsilon)$ that runs in time

$$O\left(n^{2-\delta} \log(1/\varepsilon)^{d^{1-\eta}}\right)$$

Proof. Assuming such an algorithm existed, applying the above reduction we would get an $O(n^{2-\delta'})$-time algorithm for $\mathbf{OV}(n, d')$ for some $0 < \delta' < \delta$ with $d' \geq c \log n / 3$, refuting the **OV** conjecture. $\square$

4.3 Using Williams' reduction to get a better lower bound

In this subsection, we show how we can use Williams' reduction to show a hardness result for **GT** which has worse dependence on the degree of $\log(1/\varepsilon)$, but applies to the more interesting range of dimensions $(\omega((\log\log n)^2), o(\log n))$

Lemma 4.4. Let $\ell \in [1, d]$. Then there is a

$$nd^{O(d/\ell)} + n(\ell+1)^2 O\left(\frac{d^2 \log^2 d}{\ell^2}\right) d^{O(d/\ell)} + d^{O(d/\ell)} \cdot O\left(n(\ell+1)^2 \log\log n + (\ell+1)^2 2^{O(d \log d/\ell)} \log n\right)$$

time reduction from $\mathbf{OV}(n, d)$ to $d^{O(d/\ell)}$ instances of $\mathbf{GT}(n, (\ell+1)^2, \varepsilon)$ where

$$\varepsilon = O\left(n^{-(\ell+1)^2 2^{O(d \log d/\ell)}}\right)$$

Proof. This follows by combining the reduction from **OV** to **Hop** in lemma 3.1 with the reduction from **Hop** to **BCP** in lemma 3.3 and the reduction from **BCP** to **GT** in lemma 4.1. □

In particular, we have the following corollary:

Corollary 4.5. Assuming the **OV** conjecture, there is no algorithm for $\mathbf{GT}(n, d', \varepsilon)$ in $O(n^{2-\delta}\left(\log \frac{1}{\varepsilon}\right)^{d'^{1/2-\eta}})$ time for any $\delta, \eta > 0$ and $d' = (\log\log n)^{\omega(1)} \cap O(\log n)$.

Proof. Assume the contrary. In the reduction from **OV** to **GT**, choose $d = c\log n$ and $(\ell+1)^2 = d'$ so that $\ell = \Omega((\log\log n)^{\omega(1)}) \cap O(\sqrt{\log n})$; then note that

$$\left(\log\frac{1}{\varepsilon}\right)^{\ell^{1-\eta}} = (\ell+1)^{O(\ell^{1-\eta})} 2^{O(d\ell^{-\eta}\log d)} = n^{o(1)} n^{O(\ell^{-\eta}\log\log n)} = n^{o(1)}$$

by the choice of the lower bound of ℓ.

Running time: For this choice of parameters, the total running time of our reduction is $n^{1+o(1)}$. Indeed, one sees that $d^{O(d/\ell)} = (c\log n)^{c\log n/\ell} = (c\log n)^{o(c\log n/\log\log n)}$ which is $n^{o(1)}$, and it's easy to see that all the other terms are $n^{o(1)}$ too.

Applying our supposed $O(n^{2-\delta}\left(\log \frac{1}{\varepsilon}\right)^{d'^{1/2-\eta}})$ approximate Gauss transform algorithm to this gives us a $n^{2-\delta} n^{o(1)}$ algorithm for the **OV** instance, which contradicts the **OV** conjecture. □

4.4 Improved bichromatic closest pair hardness and Gauss transform implications

In this section, we show that combining the improved reduction by Chen [Che18] from **OV** to **Hop** with our reductions gives a lower bound for **GT** for an even better range of dimensions. Firstly, by combining Chen's reduction with our above reductions we obtain the following:

Lemma 4.6. There is a

$$\begin{aligned} &O\left(n\ell^{O\left(6^{\log^* d} d/\ell\right)} \operatorname{poly}(d)\right) \\ &+ \ell^{O\left(6^{\log^* d} d/\ell\right)} O\left(n(\ell+1)^2\left(\frac{d}{\ell}\log\ell\cdot 6^{\log * d}\right)^2\right) \\ &+ \ell^{O\left(6^{\log^* d} d/\ell\right)} O\left(n(\ell+1)^2\log\log n + \ell^{O\left(6^{\log^* d} d/\ell\right)}\right) \end{aligned}$$

time reduction from $\mathbf{OV}(n, d)$ to $\ell^{O\left(6^{\log^* d} d/\ell\right)}$ instances of $\mathbf{GT}(n, (\ell+1)^2, \varepsilon)$ where

$$\varepsilon = n^{-(\ell+1)^2 2^{O\left(\frac{d}{\ell}\log\ell\cdot 6^{\log^* d} + \log\ell\right)}} = n^{-(\ell+1)^2 2^{O\left(\frac{d}{\ell}\log\ell\cdot 6^{\log^* d}\right)}}.$$

Proof. Follows by composing Chen's reduction from **OV** to **Hop** in lemma 3.2 with the reduction from **Hop** to **BCP** in lemma 3.3 and the reduction from **BCP** to **GT** in lemma 4.1. □

Using this reduction, we get a lower bound for **GT** in an improved range of dimensions:

Corollary 4.7. Assuming the **OV** conjecture, there is no algorithm for $\mathbf{GT}(n, d', \varepsilon)$ in $O(n^{2-\delta}\left(\log \frac{1}{\varepsilon}\right)^{d'^{1/2-\eta}})$ time for any $\delta, \eta > 0$ and $d' = 6^{\omega(\log^* n)} \cap o(\log n)$.

Proof. We apply Lemma 4.6 with $d' = (\ell+1)^2$. We want to get a lower bound for the running time of an algorithm for $\mathbf{GT}(n, d', \varepsilon)$ of the form $n^{2-\delta} \log\left(\frac{1}{\varepsilon}\right)^m$. The idea is to apply our reduction with $d = c \log n$ and an $m = \ell^{1-\delta}$ (where $\delta > 0$) chosen so that

$$\log \left(\frac{1}{\varepsilon}\right)^m = n^{o(1)}$$

For our specified range of d' and therefore ℓ. From our reduction with $d = c\log n$, we have

$$\log \frac{1}{\varepsilon} = (\ell+1)^2\, 2^{O\left(\frac{\log n}{\ell} \log \ell \cdot 6^{\log^* \log n}\right)} \log n$$

and we want this to the m-th power to be $n^{o(1)}$, for which it suffices that each term in the product raised to m be $n^{o(1)}$. We see that to have

$$(\log n)^m = n^{o(1)}$$

we need $m = o\left(\frac{\log n}{\log \log n}\right)$, which is true if $d' = (\ell+1)^2 = o(\log n)$. For the $(\ell+1)^2$ term it is easy to see that for our allowed values of ℓ:

$$\frac{m \log(\ell+1)}{\log n} = \frac{\ell^{1-\delta} \log(\ell+1)}{\log n} = o(1)$$

But the middle term is most troublesome. Since we choose $m = \ell^{1-\delta}$,

$$2^{\left(\frac{\log n}{\ell} \log \ell \cdot 6^{\log^* \log n}\right)^m} = n^{\ell^{-\delta} \log \ell \cdot 6^{\log^* \log n}}$$

Which is $n^{o(1)}$ if $\ell \geq 6^{\omega(\log^* n)}$ which also implies $d' \geq 6^{\omega(\log^* n)}$. This means that, for example, the lower bound works for dimensions $d' \geq \log \cdots \log n$, for any finite number of logs.

Running time: For our choices of $\ell \geq 6^{\omega(\log^* n)}$, $\ell^{O\left(6^{\log^* d} d/\ell\right)} = n^{6^{\log^* n} \log \ell/\ell} = n^{o(1)}$, and it is easy to see that the other terms are $n^{o(1)}$ too, showing that the reduction takes $O(n^{2-\xi})$-time in total for some $\xi > 0$ □

4.5 Barriers to Improving the Lower Bound

As we can see from the previous sections, the quality of dimension reduction from **OV** to **Hop** determine the lower range of allowed d' in our lower bounds for the Gauss transform. Although $6^{\omega(\log^* n)}$ is already very small, it is conjectured in [Che18] that this can be improved to $\omega(1)$, which would imply that our reduction will work for $d \geq \omega(1)$ dimensions.

In our lower bounds for small dimensions, the exponent on $\log(1/\epsilon)$ is $d'^{1/2-\eta}$, and this comes from the reduction from $\mathbf{Hop}(n, d, D)$ to $\mathbf{BCP}\,(n, d^2, O(D))$ in lemma 3.3. If we have a reduction to $\mathbf{BCP}\,(n, O(d), O(D))$ instead, then we can get a tight exponent of $d'^{1-\eta}$. One way to get such a reduction is to solve the following problem:

Problem 4.8. Given two sets of n vectors each, $A, B \subset \mathbb{Z}^d$, can we find a pair of mappings $f, g : \mathbb{Z}^d \to \mathbb{Z}^{O(d)}$, so that the norms of $f(u)$ are the same for all $u \in A$, the norms of all $g(v)$ are the same for all $v \in B$, and moreover $\max_{u \in A, v \in B} \langle f(u), g(v) \rangle$ is achieved iff $\langle u, v \rangle = 0$? The mappings f, g should also not increase the bit-complexity of u, v by more than a polynomial factor.

The reduction in lemma 3.3 can be seen as using the maps $f, g : \mathbb{Z}^d \to \mathbb{Z}^{d^2}$, where $f(x) = xx^T/\|x\|^2$ and $g(x) = -xx^T/\|x\|^2$. While we have spent a long time unsuccessfully trying to improve this reduction, it is unclear to us whether this is possible, or if there is a fundamental barrier preventing us from improving it.

In particular, if we look for a reduction with the property that $\langle f(u), g(v) \rangle \leq 0$ with equality iff $\langle u, v \rangle = 0$ (which is the case for the reduction in lemma 3.3), one can show that in such a reduction the functions f and g cannot depend on their inputs coordinate-wise, i.e. they cannot simply map each coordinate of u to a number of coordinates of $f(u)$ without looking at the other coordinates of u. This gives some evidence that we may need to look at all $\binom{d}{2}$ combinations of coordinates (and in fact, $f(u)$ and $g(v)$ from lemma 3.3 do have a coordinate for each pair of coordinates of u and v). However, it does not rule out the existence of a smarter reduction.

5 Algorithms for the Gauss Transform

In this section we discuss different algorithms for computing the Gauss Transform. We are presenting three algorithms, each one corresponding to one parameter regime, depending on the desired runtime dependence on the approximation parameter ε and the dimension d. In Section 5.1, we describe the naive, exact algorithm, that however is quadratic in the number of points. In Section 5.2, we present the Fast Gauss Transform, which has a linear dependence on n, and works well for very low dimensions and good approximation guarantees. Finally, by using Random Fourier Features, we show a simple way to get an algorithm that is linear in the number of points and the dimension, albeit with a polynomial dependence on the error.

5.1 Naive algorithm

Naively, one can compute the Gauss Transform by iterating over all query points, and evaluating the Gaussian kernel in linear time. This can be done by computing the distance between any two points. The total runtime of this algorithm is $O(n^2 d)$, where d comes from the time needed to compute the distance between two points.

Although trivial, any exact algorithm that has subquadratic dependence on n and polynomial dependence in d would refute SETH, as we have seen in the previous hardness results.

5.2 Fast Gauss Transform

In this section we will outline the Fast Gauss Transform [GS91]. The intuition behind the algorithm is that the Gaussian kernel decays exponentially in the distance squared, and so if one discretizes the space by subdividing into boxes, the value of a query point will only depend on points that are in boxes close to the query point. Furthermore, one does not need to evaluate the kernel between the query point and *every* point in a box: by computing a Taylor approximation around the center of the box, one only needs to evaluate just one polynomial for each box B.

More specifically, for any $t, s, s_B \in \mathbb{R}^d$ where s_B is the center of box B, the Taylor expansion of $e^{-||t-s||_2^2}$ for some $s \in B$ around the center s_B is

$$e^{-||t-s||_2^2} = \sum_{\alpha_1,\dots,\alpha_d \geq 0} \frac{(t - s_B)^{\alpha_1+\dots+\alpha_d}}{\alpha_1!\dots\alpha_d!} h_\alpha(s - s_B)$$

where

$$h_\alpha(t) = (-1)^{\alpha_1+\dots+\alpha_d} \frac{\partial^{\alpha_1}}{\partial t_1^{\alpha_1}} \dots \frac{\partial^{\alpha_d}}{\partial t_d^{\alpha_d}} e^{-t_1^2-\dots-t_d^2}$$

We now sum these Taylor series over all $s \in B$ and switch the order of summation to obtain a single Hermite expansion

$$G(t) = \sum_{\alpha_1,\dots,\alpha_d \geq 0} A_\alpha h_\alpha(t - s_B)$$

where

$$A_\alpha = \frac{1}{\alpha_1!\dots\alpha_n!} \sum_{s\in B} \frac{1}{n} (s_1 - s_{B,1})^{\alpha_1} \dots (s_d - s_{B,d})^{\alpha_d}$$

Now we truncate this series expansion to include only those monomials where all α_i are $\leq p$. By using Cramer's inequality, i.e. the fact that

$$\frac{1}{\alpha!}|h_\alpha(t)| \leq K e^{-(t_1^2+\dots+t_d^2)/2} 2^{(a_1+\dots+a_d)/2} \frac{1}{\sqrt{a_1!\dots a_d!}}$$

where $K < (1.09)^d$, the error due to this truncation is at most

$$p^{-\Omega(pd)} r^{O(pd)}$$

where r is the side length of the box B. Setting $r = \frac{1}{2}$ and $p = O(\log \frac{1}{\varepsilon})$ we get that the error is at most ε.

Given any two points p, q with $||p - q||_\infty \geq \Omega(\log \frac{1}{\varepsilon})$, note that

$$e^{-||p-q||_2^2} \leq e^{-||p-q||_\infty^2} \leq \varepsilon^{\Omega(1)}$$

and so we only need to care about boxes within an ℓ_∞ distance of $O\left(\log \frac{1}{\varepsilon}\right)$. The number of such boxes is $\left(\log \frac{1}{\varepsilon}\right)^{O(d)}$.

From the above, we deduce that both the time needed to compute the polynomials for all boxes and the time to evaluate the Gaussian kernel at all query points is $n\left(\log \frac{1}{\varepsilon}\right)^{O(d)}$. This gives the total runtime of the algorithm.

5.3 Random Fourier Features

Another approach for approximately computing the Fast Gauss Transform is by using *Random Fourier Features*, as introduced by [RR08]. Given a set $S \subseteq [-M, M]^d$ and an error parameter $\varepsilon > 0$, there exists a mapping $f : \mathbb{R}^d \to \mathbb{R}^D$ with $D = O(\frac{d}{\varepsilon^2} \log \frac{M}{\varepsilon})$, such that

$$\forall x, y \in S \;\; e^{-||x-y||_2^2} - \varepsilon \leq \langle f(x), f(y) \rangle \leq e^{-||x-y||_2^2} + \varepsilon$$

Furthermore, such a mapping can be computed with high probability and f can be applied in $O(D)$ time. Now this implies that also for all x

$$\frac{1}{n} \sum_{i \in [n]} e^{-||x-y_i||_2^2} - \varepsilon \leq \frac{1}{n} \sum_{i \in [n]} \langle x, y_i \rangle \leq \frac{1}{n} \sum_{i \in [n]} e^{-||x-y_i||_2^2} + \varepsilon$$

however, by linearity, the middle quantity is also equal to

$$\langle x, \frac{1}{n} \sum_{i \in [n]} y_i \rangle$$

so if we just precompute $Y = \frac{1}{n} \sum_{i \in [n]} y_i$, then $\frac{1}{n} \sum_{i \in [n]} e^{-||x-y_i||_2^2}$ can be ε-approximated in $O(D)$ time.

The time to compute the mappings of all our points is $O(nD)$, and then the time to compute the inner products is also $O(nD)$, so the final runtime is

$$O\left(\frac{nd}{\varepsilon^2} \log \frac{M}{\varepsilon}\right)$$

References

[Che18] Lijie Chen. On the hardness of approximate and exact (bichromatic) maximum inner product. *arXiv preprint arXiv:1802.02325*, 2018.

[GS91] Leslie Greengard and John Strain. The fast gauss transform. *SIAM Journal on Scientific and Statistical Computing*, 12(1):79–94, 1991.

[GW13] Michael Griebel and Daniel Wissel. Fast approximation of the discrete gauss transform in higher dimensions. *Journal of Scientific Computing*, 55(1):149–172, 2013.

[RR08] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

[RYDG05] Vikas Chandrakant Raykar, Changjaing Yang, Ramani Duraiswami, and Nail Gumerov. Fast computation of sums of gaussians in high dimensions. Technical report, 2005.

[TW09] Johannes Tausch and Alexander Weckiewicz. Multidimensional fast gauss transforms by chebyshev expansions. *SIAM Journal on Scientific Computing*, 31(5):3547–3565, 2009.

[Wil18] Ryan Williams. On the difference between closest, furthest, and orthogonal pairs: Nearly-linear vs barely-subquadratic complexity. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1207–1215. SIAM, 2018.