Abstract

We present a new efficient algorithm for the search version of the approximate Closest Vector Problem with Preprocessing (CVPP). This is the problem of finding a lattice vector whose distance from the target point is within some factor $\gamma$ of the closest lattice vector, where the algorithm is allowed to take polynomial-length advice about the lattice from an unbounded preprocessing algorithm. Our algorithm achieves an approximation factor of $O\left(\frac{n}{\sqrt{\log n}}\right)$, improving on the previous best of $O(n^{1.5})$ due to Lagarias, Lenstra, and Schnorr [LLS90]. We also show, somewhat surprisingly, that only $O(n)$ vectors of preprocessing advice are sufficient to solve the problem (with the slightly worse factor of $O(n)$). We remark that this still leaves a large gap with respect to the decisional version of CVPP, where the best known approximation factor is $O(\sqrt{n} \log n)$ due to Aharonov and Regev [AR05].

To achieve these results, we show a reduction to the same problem restricted to target points that are very close to the lattice and a more efficient reduction to a harder problem, Bounded Distance Decoding with preprocessing (BDDP). BDDP is the problem of finding the unique closest lattice point for target vectors that are very close to the lattice (with polynomial-length advice from preprocessing on the lattice). Combining either reduction with the previous best-known algorithm for BDDP by Liu, Lyubashevsky, and Micciancio [LLM06] gives our main result. We also present a substantially more efficient variant of the LLM algorithm (both in terms of run-time and amount of preprocessing advice), and via an improved analysis, show that it can decode up to a distance proportional to the reciprocal of the smoothing parameter of the dual lattice [MR07]. We show that this is never smaller than the LLM decoding radius, and that it can be up to an $\tilde{O}(\sqrt{n})$ factor larger.

*Courant Institute of Mathematical Sciences, New York University.
†Supported by the National Science Foundation (NSF) under Grant No. CCF-1320188. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.
1 Introduction

A lattice is the set of all integer combinations of \( n \) linearly independent vectors \( v_1, \ldots, v_n \) in \( \mathbb{R}^n \). These vectors are known as a basis of the lattice. In the last couple of decades, lattices became a central object of investigation in theoretical computer science due to their wide range of algorithmic and cryptographic applications.

From a computational complexity point of view, lattice problems are quite fascinating. Take for instance the Closest Vector Problem (CVP), which is one of the two main lattice problems. Here, given an \( n \)-dimensional lattice \( L \) (specified using an arbitrary basis) and a target point \( t \in \mathbb{R}^n \), the goal is to output a lattice point whose distance to \( t \) is within some approximation factor of that of the nearest (or in the decision version, approximate the distance of \( t \) to the lattice). For the nearly exponential approximation factor of \( 2^{n \log \log n / \log n} \), efficient algorithms are known \([\text{LLL}82, \text{Sch}87, \text{AKS}01]\). On the other hand, it is known that for some \( c > 0 \), approximating CVP to within \( n^{c} / \log \log n \) is hard under reasonable complexity assumptions (see \([\text{DKS}98]\) as well \([\text{Kho}10]\) and references therein). Finally, for approximation factor \( \sqrt{n} \) the problem is known to be in \( \text{NP} \cap \text{coNP} \) and hence unlikely to be \( \text{NP} \)-hard \([\text{GG}00, \text{AR}05]\). For an introduction to the area see, e.g., \([\text{MG}02, \text{Reg}10]\).

In this paper we consider a natural variant of CVP known as the Closest Vector Problem with Preprocessing (CVPP). The motivation comes from applications in coding theory and cryptography where the lattice is often fixed once and for all, and the input only consists of the target point \( t \). In CVPP, the algorithm is allowed to spend an unlimited amount of time preprocessing the given lattice, and output at the end a polynomial-size description of the lattice. Then, given that description and a target point \( t \), our goal is to efficiently solve \( \text{CVP}(L, t) \). As usual, one can consider either the search or the decision versions.

The computational hardness of CVPP was investigated in a sequence of works \([\text{Mic}01, \text{FM}02, \text{Reg}03, \text{AKKV}11]\), culminating in a hardness factor of \( 2^{\log^{1+\varepsilon} n} \) for all \( \varepsilon > 0 \) by Khot, Popat, and Vishnoi under reasonable complexity assumptions \([\text{KPV}12]\). This nearly matches the best known hardness result for the potentially much harder problem CVP. Behind the latest two hardness results is a preprocessing version of the PCP theorem.

The situation in terms of positive results, which is the focus of this work, is even more interesting. It follows from the early work of Lagarias, Lenstra, and Schnorr \([\text{LLS}90]\) on so-called Korkine-Zolotarev bases that there exists an \( n^{3/2} \) approximation algorithm for CVPP. Somewhat surprisingly, prior to this work, their algorithm was still the best known approximation algorithm for CVPP.

Improved algorithms were known only for the decision variant of CVPP in which the task is merely to approximate the distance of the target point to the lattice. An \( O(n) \) approximation algorithm was given in \([\text{Reg}03]\). It was then improved by Aharonov and Regev \([\text{AR}05]\) to an \( O(\sqrt{n} / \log n) \) approximation algorithm. We are therefore in the (somewhat absurd!) situation that we know that there is a close vector—or even that there are many—but we somehow can’t find even one! We note that an equivalence between the search and decision version of exact CVP is well known.

Since the latter algorithm is very natural and closely related to our work, we describe it here briefly. The main idea is to define for any lattice \( L \subset \mathbb{R}^n \), the periodic Gaussian function \( f : \mathbb{R}^n \to \)
Figure 1: The periodic Gaussian function

\[ f(t) = \frac{\rho(L + t)}{\rho(L)}, \]  

(1)

where \( \rho(A) = \sum_{x \in A} \exp(-\pi \|x\|^2) \). See Figure 1 for an illustration. The algorithm now follows from two observations. The first is that for points \( t \) at distance greater than \( \sqrt{n} \) from the lattice, \( f(t) \) is essentially zero, whereas for \( t \) at distance less than \( \sqrt{\log n} \), \( f(t) \) is non-negligible. The second crucial idea is that the function \( f \), despite being defined in terms of a sum over infinitely many lattice points, can be approximated to within any \( \pm 1/\text{poly}(n) \) by a function with a polynomial-size circuit. Finding that circuit is hard, but since it only depends on the lattice, we can do it in the preprocessing phase. More precisely, the [AR05] estimator is of the form

\[ f(t) \approx f_W(t) \overset{\text{def}}{=} \frac{1}{N} \sum_{i=1}^{N} \cos(2\pi \langle w_i, t \rangle) \]  

(2)

where \( W = (w_1, \ldots, w_N) \in L^* \) are dual lattice vectors sampled from a “discrete” gaussian distribution over \( L^* \). We note in passing that the \( O(\sqrt{n}/\log n) \) approximation factor is a common barrier in lattice problems, and going below it seems very difficult.

Extending these ideas to the search version of CVPP has proved to be elusive. For instance, a natural approach for solving search CVPP is try a hill climbing algorithm on the periodic Gaussian function (see again Figure 1) starting from the target point, and hope that this converges to a nearby lattice point. However, somewhat counterintuitively, there are simple examples in which the function has local maxima that are not lattice points!

While it seems unlikely that naïve hill climbing can be made to work in general, it was shown by Liu, Lyubashevsky, and Micciancio [LLM06] that such a procedure does in fact work when the target is sufficiently close to the lattice. This scenario corresponds to what is known as the Bounded Distance Decoding problem (BDD), where we assume that the target is at distance less than \( \alpha \cdot \lambda_1(L) \) (length of the shortest non-zero lattice vector). We remark that when \( \alpha \leq 1/2 \), the closest lattice vector is the unique lattice vector at distance less than \( \alpha \lambda_1(L) \) from the target. Here, Liu et al. [LLM06] give a polynomial time algorithm (subsequently dubbed LLM) for the preprocessing version of Bounded Distance Decoding (BDP), when the target is at distance \( O(\sqrt{\log n}/n) \cdot \lambda_1(L) \) (improving on earlier work of Klein [Kle00]). Their hill climbing algorithm relies on a “discrete” version of gradient ascent, where at each step they move from \( t \) to a point in \( \{t \pm \delta e_i : \delta \leq 1/2 \} \).
Our Contribution. Our first main contribution, given in Theorem 3.1, is a reduction showing that in order to solve CVPP, it suffices to answer queries that are close to the lattice. In more detail, we show that being able to approximate CVPP to within $g(n)\sqrt{n}$ for points that are within distance $\lambda_1(\mathcal{L})/(2g(n))$ of the lattice, for any non-decreasing function $g(n) \geq 1$, implies the same approximation factor for all points. We find that this statement is a bit surprising, even though in retrospect, the reduction itself is quite simple and natural. By combining this reduction with the LLM algorithm (or our improved variant described below), we immediately obtain an $O(n/\sqrt{\log n})$ approximation algorithm for search CVPP, improving on Lagarias et al.’s algorithm [LLS90].

Notice that Liu et al. [LLM06] solve BDDP, which is a harder problem than what our reduction needs; namely, for points close to the lattice it outputs the truly closest vector and not an approximation thereof. This observation is the motivation behind a modified reduction given in Theorem 3.2. Unlike Theorem 3.1, it requires an exact solution of CVPP for targets close to the lattice (i.e., a solution to BDDP), but as a result it incurs almost no blowup in the amount of needed preprocessing advice. We will return to this point below.

Our second main contribution, given in Section 5, is a full analysis of the hill climbing algorithm that replaces the “discrete” gradient ascent of LLM with exact gradient ascent (i.e., we approximate the exact gradient of $f$ and move in this direction). Apart from being a simpler and more natural algorithm, it allows us to make improvements on several fronts. Firstly, our improved analysis allows us to deal with targets that are further away from the lattice. In particular, instead of distance at most $\sqrt{(\log n)/n \cdot \lambda_1(\mathcal{L})}$, we can handle distances up to $\Omega(\sqrt{\log(1/\epsilon)/\eta_1(\mathcal{L}^*)})$ for $\epsilon = 1/\poly(n)$, which is never worse and sometimes significantly better than the bound in LLM (even for $\epsilon = \Omega(1)$). Already for $\mathbb{Z}^n$, we get distance $\Omega(1)$, which is a constant factor of $\lambda_1$, compared with $\sqrt{(\log n)/n}$ in LLM. A second improvement is in the size of the advice required from preprocessing, which apart from being inherently interesting, is a good proxy for the efficiency of the algorithm. In LLM, the advice consisted of an unspecified polynomial number of dual lattice vectors. In our algorithm, we require only $O(n \log(1/\epsilon)(1/\epsilon)^{1/50})$ dual lattice vectors, which is $O(n)$ when $\epsilon$ is constant. By combining the reduction from Theorem 3.2 with our improved BDDP algorithm, we obtain an algorithm for $O(n)$-CVPP that uses only $O(n)$ vectors of advice. This is quite remarkable since $O(n)$ vectors are “not much more” than the $n$ needed to form a basis, and using even the “best” basis as advice does not seem sufficient to obtain even a polynomial approximation for CVPP. Apart from the theoretical interest in minimizing the advice, this might have applications in cryptography or coding theory. As a final improvement, we show that exact gradient ascent converges in just $O(\log n)$ steps (where the final step uses a simple rounding procedure) compared to $O(\sqrt{n} \log n)$ steps for LLM, where the time complexity of each gradient ascent step is $O(n) \times$ # preprocessing vectors (in both algorithms).

The exact steps of our gradient ascent BDDP algorithm are extremely simple: at each step, we replace the current point $t$ with an approximation of $t + \nabla f(t)/(2\pi f(t))$. We obtain the approximation of $f$ and its gradient $\nabla f$ using the [AR05] estimator (Equation (2)). The main effort is in analyzing the algorithm. This involves a new analysis of the behavior of the periodic Gaussian (see
Section 4) which may be of independent interest. In particular, we show that $t + \nabla f(t)/(2\pi f(t))$ reduces the distance to the nearest lattice point by a constant factor, assuming $t$ is sufficiently close to the lattice and that the interference between the Gaussian peaks is reasonably small (the rate of decrease for LLM is $1 - c/\sqrt{n}$). We also prove that only a small number of dual samples are sufficient to get a good approximation of the true gradient, where we use the smoothness and concentration properties of the [AR05] estimator. See Section 5 for the analysis.

From a technical point of view, the improved decoding radius of our algorithm comes from the fact that we can handle “mild interference” among the individual Gaussians in the periodic Gaussian function. In LLM, the interference between the Gaussian peaks is exponentially small, whereas in our setting we can handle an interference of constant size (here the interference is quantified by the $\rho(L \setminus \{0\})$). See Section 4 for the analysis. For the improvement in the amount of preprocessing advice, we show that the number of dual samples needed to get a good approximation of the gradient of $f$ is essentially the same as the number of samples needed to approximate the function values of $f$. In LLM, they need to explicitly compare the function values at different points, all of which are at nearly the same distance from the lattice, and as a result they require a high degree of resolution on the function values to guarantee progress.

Open questions and Discussion. To summarize, the complexity of CVPP exhibits an intriguing behavior in which the known search/decision approximation factors now differ by a factor of $\Omega(\sqrt{n})$ (previously $\Omega(n \sqrt{\log n})$). Further closing this gap, or showing that the search version is inherently harder than the decision version would both be interesting.

At a high level, our CVPP algorithm operates by combining two different kinds of lattice algorithms (in their preprocessing variants). On one side, we use a BDDP solver, where almost all implementations are “dual” algorithms (i.e. they rely on properties of the dual lattice), which guarantees an exact closest vector when the target is close to the lattice. On the other side, we use an algorithm that finds a lattice point within a fixed distance of any target point – generally quantified by a factor times the covering radius $\mu(L)$ (the largest distance of any target to the lattice) – which corresponds in the literature to the Guaranteed Distance Decoding problem with preprocessing (GDDP) [MR07]. For GDDP, all known algorithms are all essentially “primal” (i.e. they only rely on vectors from the original lattice).

Currently, our algorithms for both BDDP and GDDP are stuck at a $\tilde{O}(\sqrt{n})$ factor from optimal. In particular, for BDD, the optimal decoding radius is $\lambda_1(L)/2$ and the best BDDP algorithm can operate at $\Omega(\sqrt{\log n}/n)\lambda_1(L)$ (achieved by LLM and our algorithm). On the GDDP side, the optimal rounding error is $\mu(L)$ whereas the best GDDP algorithm can round at $\sqrt{n}\mu(L)$ (achieved by Babai rounding with respect to an HKZ basis [Bab86]).

From our reductions, we know that any improvement in the decoding radius for BDDP would lead to a corresponding improvement for CVPP. Currently, we do not have the corresponding result for relating improvements in GDDP to CVPP, though we highly suspect this should be true. We leave this as an open problem.

Another tantalizing problem is to provide a deeper understanding of the computational complexity of BDD and GDD both with and without preprocessing. For BDD, it was shown by Liu et al. [LLM06] that $\frac{1}{\sqrt{2}}$-BDD is NP-hard in the non-preprocessing version, however nothing is known for smaller approximation factors. This is in constrast to the situation for CVPP and SVP, where any constant factor approximations are NP-hard. A natural question is therefore: is $\alpha$-BDD NP-hard for any constant $\alpha$? For GDDP, the problem is intricately related to approximating the
covering radius, where no hardness of approximation results are known (except under high \( \ell_p \) norms [HR12]). A natural question is therefore is \((1 + \varepsilon)\)-GDD NP-hard for some \( \varepsilon > 0 \)?

\section{Preliminaries}

\subsection{Lattices}

A \( d \)-dimensional lattice \( L \subset \mathbb{R}^n \) is the set of all integer linear combinations of \( n \) linearly independent vectors \( B = (b_1, \ldots, b_d) \). \( B \) is called a basis of the lattice and is not unique. We sometimes write \( L(B) \) to signify the lattice generated by \( B \).

We typically consider only lattices \( L \subset \mathbb{R}^n \) whose dimension is \( n \) (lattices of full rank). We note that all of our theorems and lemmas apply to more general lattices by thinking of the lattice as embedded in \( \text{span}(L) \). We sometimes make use of this fact implicitly.

For any point \( x \in \mathbb{R}^n \), we define \( \text{dist}(x, L) \) as the minimum of \( \|x - y\| \) for all \( y \in L \). For any lattice \( L \), the dual lattice, denoted \( L^* \), is defined as the set of all points in \( \text{span}(L) \) that have integer inner products with all lattice points,

\[ L^* = \{ y \in \text{span}(L) : \forall x \in L, \langle x, y \rangle \in \mathbb{Z} \} . \]

Similarly, for a lattice basis \( B = (b_1, \ldots, b_n) \), we define the dual basis \( B^* = (b_1^*, \ldots, b_n^*) \) to be the unique set of vectors satisfying \( \langle b_i^*, b_j \rangle = \delta_{ij} \). It is easy to show that \( L^* \) is itself an \( n \)-dimensional lattice and \( B^* \) is a basis of \( L^* \).

\subsection{Gram-Schmidt Orthogonalization}

Given a basis, \( B = (b_1, \ldots, b_n) \), we define the corresponding Gram-Schmidt orthogonalized vectors \( (\tilde{b}_1, \ldots, \tilde{b}_n) \) by

\[ \tilde{b}_i = b_i - \sum_{j=1}^{i-1} \mu_{ij} \tilde{b}_j , \]

\[ \mu_{ij} = \frac{\langle b_i, \tilde{b}_j \rangle}{\|b_j\|^2} . \]

In other words, \( \tilde{b}_i \) is the component of \( b_i \) that is orthogonal to \( b_1, \ldots, b_{i-1} \).

\subsection{Successive Minima and HKZ Bases}

\textbf{Definition 2.1.} For any lattice \( L \), the \( i \)th successive minimum of \( L \) is

\[ \lambda_i(L) = \inf \{ r : \dim(\text{span}(L \cap B(0, r))) \geq i \} . \]

Intuitively, the \( i \)th successive minimum of \( L \) is the smallest value \( r \) such that there are \( i \) linearly independent vectors in \( L \) of length at most \( r \). Banaszczyk proved that the successive minima of the primal and the dual are related.

\textbf{Lemma 2.2 ([Ban93 Theorem 2.2])}. For any lattice \( L \subset \mathbb{R}^n \) of dimension \( n \), \( \lambda_i(L) \lambda_{n-i}(L^*) \leq n \) for \( i = 1, \ldots, n \).
It is natural to ask for a basis \( b_1, \ldots, b_n \) of \( L \) such that \( \|b_i\| \leq \lambda_i(L) \) for all \( i \). However, such a basis does not always exist\(^1\). Instead, we have HKZ bases (Hermite, Korkin, Zolotarev), which do always exist.

**Definition 2.3.** A basis \( B = (b_1, \ldots, b_n) \) of \( L \) is an HKZ basis if

1. \( \|b_1\| = \lambda_1(L) \);
2. for \( j < i \), the Gram-Schmidt coefficients, \( \mu_{i,j} \) of \( B \) satisfy \( |\mu_{i,j}| \leq \frac{1}{2} \); and
3. \( \pi\{b_1\}(b_2), \ldots, \pi\{b_1\}(b_n) \) is an HKZ basis of \( \pi\{b_1\}(L) \).

### 2.4 CVPP and BDD

**Definition 2.4.** For any approximation parameter \( \gamma = \gamma(n) \geq 1 \), the search problem \( \gamma\text{-CVP} \) (Closest Vector Problem) is defined as follows: The input is a basis \( B \) for a lattice \( L \subset \mathbb{R}^n \) and a vector \( t \in \mathbb{R}^n \), the target. The goal is to output a vector \( y \in L \) satisfying \( \|t - y\| \leq \gamma \cdot \text{dist}(t, L) \).

We often ignore the basis and simply refer to \( L \) and \( t \) as the input.

**Definition 2.5.** Let \( \phi \) be a positive-valued function on lattices and \( \gamma(n) \geq 1 \), then \( \gamma\text{-CVP}^\phi \) is the problem of solving \( \gamma\text{-CVP} \) when the input lattice \( L \) and target point \( t \) satisfy \( \text{dist}(t, L) < \phi(L) \). If the target point is outside of this range, any output is acceptable\(^2\).

**Definition 2.6.** An algorithm with preprocessing consists of two phases. The first phase, called the preprocessing algorithm, takes input \( P \) and outputs an advice string \( A \). The second phase, called the query algorithm, takes input \( A \) and \( Q \), the query, and outputs a solution \( S \). We say that such an algorithm runs in polynomial time if the advice \( A \) is polynomial in the length of \( P \) and the query algorithm runs in time polynomial in the lengths of \( P \) and \( Q \). The preprocessing algorithm may take arbitrary time.

**Definition 2.7.** The search problems \( \gamma\text{-CVPP} \) and \( \gamma\text{-CVPP}^\phi \) (Closest Vector Problem with Preprocessing) are the preprocessing analogues of \( \gamma\text{-CVP} \) and \( \gamma\text{-CVP}^\phi \) respectively, defined as follows: The input to preprocessing is a basis \( B \) of a lattice \( L \subset \mathbb{R}^n \). The input to the query phase is a vector \( t \in \mathbb{R}^n \). The goal is to return a valid solution to \( \gamma\text{-CVP} \) or \( \gamma\text{-CVP}^\phi \) respectively.

**Definition 2.8.** The decision problem \( \gamma\text{-GapCVPP} \) is the decision analogue of \( \gamma\text{-CVP} \), defined as follows: The input to preprocessing is a basis \( B \) of a lattice \( L \subset \mathbb{R}^n \) and a distance \( d \). The input to the query phase is a vector \( t \in \mathbb{R}^n \). It is a YES instance if \( \text{dist}(t, L) \leq 1 \). It is a NO instance if \( \text{dist}(t, L) > \gamma \).

**Definition 2.9.** For any approximation parameter \( \alpha = \alpha(n) \), the search problem with preprocessing \( \alpha\text{-BDDP} \) (Bounded Distance Decoding) is simply \( 1\text{-CVPP}^\phi \) where \( \phi(L) = \alpha \cdot \lambda_1(L) \) for any lattice \( L \).

---

\(^1\)Consider, for example, the lattice \( L(e_1, e_2, \ldots, e_n, 1, \frac{1}{2} \sum_{i=1}^n e_i) \). For \( n > 4 \), the successive minima of this lattice are clearly all 1, but no set of unit vectors is a basis for this lattice.

\(^2\)Note, however, that an algorithm is only considered to run in polynomial time if it halts in polynomial time on all input, including target points outside of \( \phi(L) \).
2.5 The Discrete Gaussian and the Smoothing Parameter

For any \( s > 0 \), we define the function \( \rho_s : \mathbb{R}^n \to \mathbb{R} \) as \( ρ_s(t) = e^{-π∥t∥^2/s^2} \). For a discrete subset \( S \) of \( \mathbb{R}^n \), we define \( ρ_s(S) = \sum_{x \in S} ρ_s(x) \). When \( s = 1 \), we simply write \( ρ(t) \).

**Definition 2.10.** Let \( D_{\mathcal{L},s} \) be the probability distribution over \( \mathcal{L} \) such that the probability of drawing \( y \in \mathcal{L} \) is proportional to \( ρ_s(y) \). We call this the discrete Gaussian distribution over \( \mathcal{L} \) with parameter \( s \).

For any lattice \( \mathcal{L} \subset \mathbb{R}^n \) and \( t \in \mathbb{R}^n \), define \( f(t) = f_{\mathcal{L}}(t) = \rho(\mathcal{L} + t)/\rho(\mathcal{L}) \).

Banaszczyk proved the following in [Ban93].

**Lemma 2.11.** Let \( \mathcal{L} \subset \mathbb{R}^n \) be a lattice of dimension \( n \). Then, for all \( t \in \mathbb{R}^n \), \( f(t) \geq ρ(t) \).

**Proof.**

\[
\rho(\mathcal{L} + t) = ρ(t) \sum_{y \in \mathcal{L}} \cosh(2π⟨y, t⟩)ρ(y) ≥ ρ(t)ρ(\mathcal{L}).
\]

**Definition 2.12.** For \( ε > 0 \) and \( \mathcal{L} \subset \mathbb{R}^n \) a lattice, we define the smoothing parameter \( η_{\varepsilon}(\mathcal{L}) \) as the unique value satisfying \( ρ_{1/η_{\varepsilon}(\mathcal{L})}(\mathcal{L}^* \setminus \{0\}) = ε \).

The name smoothing parameter comes from the fact that, for \( s ≥ η_{\varepsilon}(\mathcal{L}) \), \( ρ_s(\mathcal{L} + t) \) varies by at most a multiplicative factor of \( (1 ± ε) \) [Reg09].

**Lemma 2.13.** Let \( \mathcal{L} \subset \mathbb{R}^n \) be a lattice of dimension \( n \geq 1 \) and \( 2^{-n} ≤ ε < 1 \). Then,

\[
\frac{λ_1(\mathcal{L})}{\sqrt{n}} ≤ \frac{1}{η_{\varepsilon}(\mathcal{L}^*)} < \sqrt{\frac{π}{\ln(2/ε)}} \cdot λ_1(\mathcal{L}).
\]

**Proof.** The first inequality is Lemma 3.2 of [MR07]. For the second, set

\[
s = \sqrt{\frac{\ln(2/ε)}{π}} \cdot \frac{1}{λ_1(\mathcal{L})}.
\]

Then,

\[
ρ_{1/s}(\mathcal{L} \setminus \{0\}) > 2 ρ_{1/s}(λ_1(\mathcal{L})) = 2 e^{-ln(2/ε)} = ε.
\]

2.6 Subgaussian and subexponential random variables

We next introduce subgaussian and subexponential random variables, and in particular, the subgaussianity of \( D_{\mathcal{L},s} \).

**Definition 2.14.** We say that a random variable \( X \) (or its distribution) over \( \mathbb{R}^n \) is subgaussian with parameter \( s > 0 \) if \( \mathbb{E}[X] = 0 \), and for all \( t \in \mathbb{R} \) and all unit vectors \( v \in \mathbb{R}^n \),

\[
\Pr[|⟨X, v⟩| ≥ t] ≤ 2 e^{-t^2/s^2}.
\]

**Lemma 2.15** ([MP12 Lemma 2.8]). Let \( \mathcal{L} \subset \mathbb{R}^n \) be a lattice of dimension \( n \). Then for any \( s > 0 \), \( D_{\mathcal{L},s} \) is subgaussian with parameter \( s \).
**Definition 2.16.** We say that a random variable $X$ (or its distribution) over $\mathbb{R}$ is subexponential with parameter $s$ if, for any $t > 0$

$$\Pr[|X| \geq t] \leq e^{1-t/s}.$$ 

Vershynin proved a basic relationship between subgaussian and subexponential random variables, from which we derive a simple corollary.

**Lemma 2.17** ([Ver12, Lemma 5.14]). If $X$ is a subgaussian random variable over $\mathbb{R}^n$ with parameter $s$, then for any unit vector $\mathbf{v} \in \mathbb{R}^n$, $\langle X, \mathbf{v} \rangle^2$ is subexponential with parameter $O(s)$.

**Corollary 2.18.** If $X$ and $Y$ are subgaussian random variables over $\mathbb{R}^n$ with parameter $s$, then for any two unit vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$, $\langle X, \mathbf{u} \rangle \langle Y, \mathbf{v} \rangle$ is subexponential with parameter $O(s)$.

**Proof.** It follows immediately from the definitions that subgaussian random variables with parameter $O(s)$ are closed under addition and multiplication by constants, as are subexponential random variables with parameter $O(s)$. Therefore,

$$\langle X, \mathbf{u} \rangle \langle Y, \mathbf{v} \rangle = \frac{1}{2} \left( \langle X, \mathbf{u} \rangle + \langle Y, \mathbf{v} \rangle \right)^2 - \frac{1}{2} \langle X, \mathbf{u} \rangle^2 - \frac{1}{2} \langle Y, \mathbf{v} \rangle^2.$$ 

is subexponential with parameter $O(s)$ as claimed. \hfill \Box

Vershynin showed the next useful property of subexponential random variables.

**Lemma 2.19** ([Ver12, Proposition 5.16]). Let $X_1, \ldots, X_N$ be independent subexponential random variables over $\mathbb{R}$ with parameter $s$, and suppose $\mathbb{E}[X_i] = 0$ for all $i$. Then, for any $t \geq 0$ and $\mathbf{v} \in \mathbb{R}^n$ a unit vector,

$$\Pr \left[ \frac{1}{N} \left| \sum_i X_i \right| \geq t \right] \leq 2^{-\Omega(N \min(t/s, t^2/s^2))}.$$ 

### 2.7 Babai’s nearest plane algorithm

Babai’s nearest plane algorithm (denoted BABAI) is an algorithm introduced by Babai [Bab86] for rounding a target vector to a nearby lattice point one coordinate at a time. The input is a basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ for a lattice $\mathcal{L}$ and a target $\mathbf{t} \in \mathbb{R}^n$.

We first project $\mathbf{t}$ onto $\text{span}(\mathcal{L})$. We then choose the last coordinate $c_n \in \mathbb{Z}$ of our nearby lattice point by simple rounding, setting

$$c_n = \lfloor \langle \mathbf{t}, \mathbf{b}_n^* \rangle \rfloor.$$ 

Next we call BABAI recursively on $(\mathbf{b}_1, \ldots, \mathbf{b}_{n-1})$ and $\mathbf{t} - c_n \mathbf{b}_n$ and receive the result $\mathbf{y}$. We then return $\mathbf{y} + c_n \mathbf{b}_n$.

Stated more intuitively, BABAI chooses the hyperplane

$$c_n \tilde{\mathbf{b}}_n + \text{span}(\mathbf{b}_1, \ldots, \mathbf{b}_{n-1}) = \{ \mathbf{x} \in \text{span}(\mathcal{L}) : \langle \mathbf{x}, \mathbf{b}_n^* \rangle = c_n \}$$

with $c_n \in \mathbb{Z}$ – which ensures that is a lattice hyperplane – that is nearest to the target and recurses on this hyperplane. When this hyperplane contains a nearest point in $\mathcal{L}$ to the target, we say that the algorithm chooses a correct hyperplane.

Babai proved the following standard fact about his algorithm.
Lemma 2.20 ([Bab86]). Let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice of dimension $n$. For any basis, $B = (b_1, \ldots, b_n)$ of $\mathcal{L}$ with Gram-Schmidt orthogonalization $(\tilde{b}_1, \ldots, \tilde{b}_n)$ and any target vector $t \in \mathbb{R}^n$, BABAI($t, B$) outputs $y \in \mathcal{L}$ such that
\[
\|y - t\|^2 \leq \frac{1}{4} \sum_{i=1}^{n} \|\tilde{b}_i\|^2 \leq \frac{n}{4} \cdot \max_i \|\tilde{b}_i\|^2 .
\]

The following was shown by Lagarias, Lenstra, and Schnorr [LLS90]. We include a proof for completeness.

Lemma 2.21. For any lattice $\mathcal{L} \subset \mathbb{R}^n$ of dimension $n$, there exists a basis $B$ such that BABAI($\cdot, B$) solves $\frac{1}{2n}$-BDDP.

Proof. We use a dual HKZ basis $B = (b_1, \ldots, b_n)$ of $\mathcal{L}$. Here the dual basis $B^* = (b^*_1, \ldots, b^*_n)$ satisfies that $(b^*_1, \ldots, b^*_n)$ is an HKZ basis of $\mathcal{L}^*$ (note the reversed ordering).

Let $t \in \mathbb{R}^n$ with $\text{dist}(t, \mathcal{L}) \leq \lambda_1(\mathcal{L})/(2n)$. Let $y \in \mathcal{L}$ denote the closest lattice vector to $t$. We shall argue by induction that BABAI chooses the hyperplane containing $y$ at each iteration.

In the first iteration, BABAI chooses the hyperplane $\langle \langle t, b^*_n \rangle \rangle = \langle y, b^*_n \rangle$. Using Cauchy-Schwarz followed by Minkowski’s first theorem, we get that
\[
\langle t, y \rangle \leq \|t - y\| \|b^*_n\| < \frac{1}{2n} \lambda_1(\mathcal{L}) \lambda_1(\mathcal{L}^*) 
\leq \frac{1}{2n} (\sqrt{n} \det(\mathcal{L}))(\sqrt{n} \det(\mathcal{L}^*)) = \frac{n}{2n} = 1/2 .
\]
Hence we must have $\langle \langle t, b^*_n \rangle \rangle = \langle y, b^*_n \rangle$, as needed.

Let $W = \text{span}(b_1, \ldots, b_{n-1})$. We remember that the recursive subproblem corresponds to the lattice $\mathcal{L}' = \mathcal{L}(b_1, \ldots, b_{n-1})$ with the target $t' = \pi_W(t - c_n b_n)$, where $c_n = \lceil \langle b^*_n, t \rangle \rceil$. Clearly, the closest lattice vector to $t'$ in $\mathcal{L}'$ is $y' = y - c_n b_n$, where the fact that $y' \in \mathcal{L}'$ follows from the above argument. Note that
\[
\frac{1}{2n} \lambda_1(\mathcal{L}') \geq \frac{1}{2n} \lambda_1(\mathcal{L}) > \|y - t\| \geq \|\pi_W(y - t)\| = \|y' - t'\| = \text{dist}(t', \mathcal{L}').
\]
Lastly, it is easy to check that $(b_1, \ldots, b_{n-1})$ is again a dual HKZ basis for $\mathcal{L}'$. Since the induction hypothesis holds, we get that BABAI chooses the correct hyperplane at each iteration as needed. 

\[\square\]

2.8 \(\delta\)-nets and the spectral norm

Definition 2.22. For any $\delta > 0$, $A \subset \mathbb{R}^n$ is a $\delta$-net of $S$ if $A \subseteq S$, and for each $v \in S$, there is some $u \in A$ such that $\|u - v\| \leq \delta$.

We’ll be interested in the case when $S$ is a ball, a sphere, or a shell. The next lemma shows that we can do this without many points. The proof is by a standard packing argument. (See Lemma 5.2 of [Ver12], for example.)

Lemma 2.23. For any $\delta > 0$, there exists a $\delta$-net of the unit ball in $\mathbb{R}^n$ with $(1 + 2/\delta)^n$ points. Nets of the same cardinality exist for spherical shells of outer radius one, and for the unit sphere.
A \( \delta \)-net of the unit sphere can be used to accurately approximate the length of any vector.

**Lemma 2.24.** Let \( \delta \in (0, 1) \), and let \( A \) be a \( \delta \)-net of the unit sphere in \( \mathbb{R}^n \). Then, for any \( x \in \mathbb{R}^n \),

\[
\max_{v \in A}|\langle v, x \rangle| \leq \|x\| \leq \frac{1}{1 - \delta} \cdot \max_{v \in A}|\langle v, x \rangle|
\]

**Proof.** Without loss of generality, let \( \|x\| = 1 \). The first inequality is trivial. By hypothesis, there is some \( v \in A \) such that \( \|v - x\| \leq \delta \). Then,

\[
\langle v, x \rangle = \langle x, x \rangle - \langle v - x, x \rangle \geq 1 - \delta .
\]

The result follows.

Similarly, a \( \delta \)-net can be used to approximate the spectral norm of a matrix, as defined below.

**Definition 2.25.** For any matrix \( M \in \mathbb{R}^{n \times n} \), the spectral norm of \( M \), \( \|M\| \) is defined as

\[
\|M\| = \sup_{\|x\| = 1} \|Mx\|.
\]

Equivalently, for a symmetric matrix \( M \), \( \|M\| \) is the absolute value of the largest eigenvalue of \( M \).

**Lemma 2.26 ([Ver12] Lemma 5.4).** For any symmetric matrix \( M \in \mathbb{R}^{n \times n} \) and any \( \delta \)-net of the unit sphere, \( A \), with \( 0 < \delta < 1/2 \),

\[
\|M\| \leq \frac{1}{1 - 2\delta} \cdot \max_{x \in A}|\langle Mx, x \rangle|.
\]

### 3 Reduction from CVPP to CVPP with a promise

Our first reduction shows that it suffices to solve CVPP for targets close to the lattice.

**Theorem 3.1.** Let \( \gamma(n) = g(n) \sqrt{n} \) where \( g(n) \geq 1 \) is a non-decreasing function. Let \( \phi(L) = \lambda_1(L)/(2g(n)) \) for any lattice of dimension \( n \). Then, a polynomial-time algorithm that solves \( \gamma \)-CVPP\( ^\phi \) implies a polynomial-time algorithm that solves \( \gamma \)-CVPP.

A slight modification of our reduction leads to a more efficient reduction from \( \gamma \)-CVPP\( ^\phi \) to a harder problem, \( \frac{1}{2g} \)-BDDP. In particular, this reduction needs only \( O(n) \) vectors as advice from preprocessing when the initial \( \frac{1}{2g} \)-BDDP algorithm needs only \( O(n^c) \) vectors as advice.

**Theorem 3.2.** Let \( \gamma(n) = g(n) \sqrt{n} \) where \( g(n) \geq 1 \) is a non-decreasing function. Let \( \phi(L) = \lambda_1(L)/(2g(n)) \) for any lattice \( L \) of dimension \( n \). Suppose that there exists a polynomial-time algorithm that solves \( 1 \)-CVPP\( ^\phi = \frac{1}{2g} \)-BDDP that uses at most \( O(n^c) \) vectors as advice from preprocessing for \( c \geq 1 \). Then, there exists an algorithm that solves \( \gamma \)-CVPP in polynomial-time that uses at most \( O(n^c) \) vectors as advice from preprocessing.
3.1 Proof of Theorem 3.1

Proof of Theorem 3.1 Suppose that we have an algorithm that solves $\gamma$-CVPP$^\phi$ in polynomial time, with preprocessing algorithm $P$ and query algorithm $Q$. We construct an algorithm that solves $\gamma$-CVPP as follows.

On input $L \subseteq \mathbb{R}^n$, the preprocessing algorithm first computes an HKZ basis $B = (b_1, \ldots, b_n)$ of $L$. For $i = 0, \ldots, n$, let $\pi_i = \pi_{\{b_1, \ldots, b_i\}^+}$ and $N_i = \pi_i(L)$. Then, the preprocessing algorithm returns as its advice $B$ and the advice strings $A_i = P(N_i)$ for all $i$.

On input $t \in \mathbb{R}^n$, the query algorithm does the following for each $i = 0, \ldots, n$. It computes $w_i = Q(A_i, \pi_i(t)) \in N_i$. Write $w_i = \sum_{j=i+1}^n a_{ij} \pi_i(b_j)$ for some coefficients $a_{ij} \in \mathbb{Z}$ and let $y_i = \sum_{j=i+1}^n a_{ij} b_j \in L$ be a “lift” of $w_i$. Similarly, let $M_i = L(b_1, \ldots, b_i) \subseteq L$ and

$$z_i = \text{BABAI}(\pi_{\text{span}(M_i)}(t - y_i), (b_1, \ldots, b_i)) \in L.$$ 

The query algorithm then returns the vector nearest to the target $t$ among the vectors $y_i + z_i \in L$, $i \in \{0, \ldots, n\}$.

Clearly, the advice from preprocessing has polynomial length and the query algorithm runs in polynomial time. Let $i \in \{0, \ldots, n-1\}$ be minimal such that $\text{dist}(\pi_i(t), N_i) < \phi(N_i) = \|\tilde{b}_{i+1}\|/(2g(n-i))$, where $(\tilde{b}_1, \ldots, \tilde{b}_n)$ is the Gram-Schmidt orthogonalization of $B$. If no such $i$ exists, we take $i = n$. We will complete the proof by showing that $y_i + z_i$ is close to $t$. By separating the norm into its projection on the two orthogonal subspaces,

$$\|y_i + z_i - t\|^2 = \|\pi_i(y_i - t)\|^2 + \|\pi_{\text{span}(M_i)}(y_i + z_i - t)\|^2$$

$$= \|w_i - \pi_i(t)\|^2 + \|z_i - \pi_{\text{span}(M_i)}(t - y_i)\|^2.$$ 

For the first term, using the definition of $\gamma$-CVPP$^\phi$ and our choice of $i$, we have that

$$\|w_i - \pi_i(t)\| \leq \gamma(n-i) \text{ dist}(\pi_i(t), N_i) \leq g(n)\sqrt{n-\ell} \text{ dist}(t, L) .$$

For the second term, by Lemma 2.20 and again by our choice of $i$,

$$\|z_i - \pi_{\text{span}(M_i)}(t - y_i)\|^2 \leq \frac{i}{4} \max_{j \in \{0, \ldots, i-1\}} \|\tilde{b}_{j+1}\|^2$$

$$\leq i \max_{j \in \{0, \ldots, i-1\}} g(n-j)^2 \text{ dist}(\pi_j(t), N_j)^2$$

$$\leq i \cdot g(n)^2 \cdot \text{ dist}(t, L)^2 .$$

Combining the two inequalities, we get

$$\|y_i + z_i - t\|^2 \leq (n-i)g(n)^2 \cdot \text{ dist}(t, L)^2 + ig(n)^2 \cdot \text{ dist}(t, L)^2 = \gamma(n)^2 \text{ dist}(t, L)^2 .$$ 

\[ \square \]

3.2 Proof of Theorem 3.2

In order to prove Theorem 3.2, we modify the above reduction to pre-compute a sequence of lattices $N_j$ whose dimensions sum to $n$. 

12
Proof of Theorem 3.2. We start by presenting an inefficient reduction from γ-CVP to 1-CVPφ. We will later observe that with proper preprocessing, it becomes an efficient reduction from γ-CVPP to 1-CVPPφ = \( \frac{1}{2\gamma} \)-BDDP with the desired properties.

Suppose that \( A \) is an algorithm that solves 1-CVPφ, and consider the following recursive algorithm that solves γ-CVP. On input \( t \in \mathbb{R}^n \) and \( L \subset \mathbb{R}^n \) a lattice of dimension \( d \), the algorithm first computes an HKZ basis of \( L \), \( B = (b_1, \ldots, b_d) \), as well as \( v = \text{BABAI}(t, B) \). If \( d \leq 1 \), it simply returns \( v \). Otherwise, let \( i = \arg \max_i \|b_{i+1}\| / g(d - j) \), \( \pi = \pi_{(b_1, \ldots, b_i)} \), and \( N' = \pi(L) \). The algorithm then computes \( A(\pi(t), N') \) and lifts it to \( y \in L \), i.e., a vector satisfying \( \pi(y) = A(\pi(t), N') \) (as in the previous reduction). It then recursively calls itself with the lattice \( M = L(b_1, \ldots, b_i) \) and the target \( t - y \), receiving output \( z \). Finally, it returns either \( v \) or \( y + z \), whichever is closer to \( t \).

We now prove by induction on \( d \) that this algorithm correctly solves γ-CVP. If \( d \leq 1 \), we clearly output the closest point, so assume \( d \geq 2 \). Assume first that dist\((t, L) \geq \|b_{i+1}\| / (2g(d-i)) \). Then by Lemma 2.20
\[
\|v - t\| \leq g(d-i)\sqrt{d} \text{dist}(t, L) \leq \gamma(n) \text{dist}(t, L)
\]
and our output is correct. Otherwise, we have that
\[
\text{dist}(\pi(t), N') \leq \text{dist}(t, L) < \frac{\|b_{i+1}\|}{2g(d-i)} = \frac{\lambda_1(N')}{2g(d-i)} = \phi(N)
\]
and so by the definition of 1-CVPφ, \( \pi(y) \) is the closest vector to \( \pi(t) \) in \( N' \). We now claim that \( M + y \) must contain one of the closest vectors to \( t \) in \( L \) (in fact all of them). This implies by our induction hypothesis that the recursive call returns a valid solution to γ-CVP, completing the proof. To prove the claim, just notice that dist\((t, L) < \lambda_1(N') / 2 \), yet all the points \( y' \in L \setminus (M + y) \) must be at distance at least
\[
\text{dist}(\pi(y'), \pi(t)) \geq \lambda_1(N') - \text{dist}(\pi(t), N') \geq \frac{\lambda_1(N')}{2}
\]
from \( t \) and hence cannot be closest to \( t \) in \( L \).

It is easy to make this reduction efficient using preprocessing. We first equip the algorithm with an HKZ basis \( B \) for the original lattice. Since the lattices \( M_j \) appearing in the recursion are all generated by prefixes of \( B \), that same prefix gives an HKZ basis for them, and so this one HKZ basis suffices. Second, we should provide the indices \( i \) appearing in the recursion. Finally, we need to provide the advice required by the 1-CVPPφ algorithm for each \( N_j \).

Note that \( \sum \dim(N_j) = n \), so if the advice for each \( N_j \) is at most \( c_0 \dim(N_j)^c \) vectors for some \( c_0, c \geq 1 \), then the total preprocessing consists of \( n \) vectors for the HKZ basis, at most \( n \) indices, and at most \( c_0 \sum \dim(N_j)^c \leq c_0 n^c \) for the additional advice. So the total advice is \( O(n^c) \) vectors as claimed.

4 Two bounds on the discrete Gaussian

We present two bounds on the discrete Gaussian distribution. The first, illustrated in Figure 2, bounds from above the periodic Gaussian function and is a complement to the lower bound in Lemma 2.11. It will be not be used in the sequel and is included here as a warmup and for future
reference. The second shows that $\nabla f(t)/(2\pi f(t))$ is close to the difference between $t$ and the nearest lattice point when $t$ is sufficiently short (and, by periodicity, for all $t$ that are sufficient close to the lattice).

**Lemma 4.1.** Let $\epsilon > 0$ and $\mathcal{L} \subset \mathbb{R}^n$ a lattice with $\rho(\mathcal{L}) = 1 + \epsilon$. Let $s_0^2 = \frac{1}{\pi} \log \frac{2(1+\epsilon)}{\epsilon}$. Then, for any $t \in \mathbb{R}^n$,

$$f(t) \leq \rho(t) \left( \frac{1}{1+\epsilon} + \frac{\epsilon}{1+\epsilon} \cdot \cosh(2\pi s_0 \|t\|) \right) + 2\pi \|t\| \int_{s_0 - \|t\|}^{s_0 + \|t\|} e^{-\pi s^2} ds.$$

**Proof.** We can write

$$\frac{\rho(\mathcal{L} + t)}{\rho(\mathcal{L})} = \frac{\rho(t)}{\rho(\mathcal{L})} \cdot \sum_{y \in \mathcal{L}} e^{-2\pi \langle y, t \rangle} \rho(y) = \rho(t) \mathbb{E}_{y \sim D_\epsilon} [\cosh(2\pi \langle y, t \rangle)].$$

We now use the fact that for any real-valued random variable $X$ and even function $g : \mathbb{R} \rightarrow \mathbb{R}$,

$$\mathbb{E}_X[g(X)] = \mathbb{E}_X[g(|X|)] = g(0) + \int_0^\infty g'(s) \Pr_X[|X| > s] ds.$$

Therefore, the expectation in Eq. (4) is given by

$$1 + 2\pi \|t\| \int_{s_0}^{\infty} \Pr[\langle y, t \rangle > s \|t\|] \sinh(2\pi s \|t\|) ds.$$  

We can upper bound the probability using Lemma 2.15 (and noticing that $y$ is nonzero with probability $\epsilon/(1+\epsilon)$) by

$$\Pr[\langle y, t \rangle > s \|t\|] \leq \min \left( \frac{\epsilon}{1+\epsilon}, 2e^{-\pi s^2} \right).$$

The minimum is determined by the second term for $s > s_0$. We can therefore bound the integral in Eq. (5) from above by the sum of two integrals, the first being

$$\frac{\epsilon}{1+\epsilon} \int_{s_0 = 0}^{s_0} \sinh(2\pi s \|t\|) ds = \frac{\epsilon}{1+\epsilon} \cdot \frac{\cosh(2\pi s_0 \|t\|)}{2\pi \|t\|} - 1.$$
and the second being
\[
2 \int_{s=s_0}^{\infty} e^{-\pi s^2} \sinh(2\pi s \|\mathbf{t}\|) \, ds = \frac{1}{\rho(t)} \int_{s_0-\|\mathbf{t}\|}^{s_0+\|\mathbf{t}\|} e^{-\pi z^2} \, dz.
\]
Putting it all together, we obtain the desired bound
\[
\frac{\rho(L + t)}{\rho(L)} \leq \rho(t) \left( \frac{1}{1 + \epsilon} + \frac{\epsilon}{1 + \epsilon} \cdot \cosh(2\pi s_0 \|\mathbf{t}\|) \right) + 2\pi |\mathbf{t}| \frac{1}{\rho(t)} \int_{s_0-\|\mathbf{t}\|}^{s_0+\|\mathbf{t}\|} e^{-\pi z^2} \, dz .
\]

Lemma 4.2. Let \( \epsilon > 0 \) and \( L \subset \mathbb{R}^n \) a lattice with \( \rho(L) = 1 + \epsilon \). Let \( s_0^2 = \frac{1}{\pi} \log \frac{2(1+\epsilon)}{\epsilon} \). Then, for any \( t \in \mathbb{R}^n \),
\[
\left\| \frac{\nabla f(t)}{2\pi f(t)} + t \right\| \leq \frac{\epsilon}{1 + \epsilon} \cdot (s_0 \sinh(2\pi s_0 \|\mathbf{t}\|) + \|\mathbf{t}\| \cosh(2\pi s_0 \|\mathbf{t}\|)) + \frac{1}{\rho(t)} \cdot (1 + 2\pi \|\mathbf{t}\|^2) \int_{s_0-\|\mathbf{t}\|}^{s_0+\|\mathbf{t}\|} e^{-\pi z^2} \, dz .
\]

Proof. Note that
\[
\left\| \frac{\nabla f(t)}{2\pi f(t)} + t \right\| = \max_{\|\mathbf{v}\|=1} \mathbb{E} \left[ (x - t, \mathbf{v}) \right].
\]
So fix a unit vector \( \mathbf{v} \in \mathbb{R}^n \). We can write
\[
\rho(L + t) \cdot \mathbb{E}_{x \sim D_{L+t}} [(x - t, \mathbf{v})] = \rho(L) \rho(t) \mathbb{E}_{y \sim D_{L}} [e^{-2\pi \langle y, t \rangle} \langle y, \mathbf{v} \rangle]
\]
\[
= (1 + \epsilon) \rho(t) \mathbb{E}_{y \sim D_{L}} [\sinh(-2\pi \langle y, t \rangle) \langle y, \mathbf{v} \rangle] .
\]
For any \( y \), let \( P_{r,s}(y) \) be the event that \( \|\mathbf{v}\| > s, \|\mathbf{t}\| > r \|\mathbf{t}\| \), and \( \langle y, \mathbf{t} \rangle \langle y, \mathbf{v} \rangle < 0 \). Similarly, let \( N_{r,s}(y) \) be the event that \( \|\mathbf{v}\| > s, \|\mathbf{t}\| > r \|\mathbf{t}\| \), and \( \langle y, \mathbf{t} \rangle \langle y, \mathbf{v} \rangle > 0 \). Then,
\[
\sinh(-2\pi \langle y, t \rangle) \langle y, \mathbf{v} \rangle = 2\pi \|\mathbf{t}\| \int_{0}^{\infty} \int_{0}^{\infty} \cosh(2\pi \|\mathbf{t}\| r) (P_{r,s}(y) - N_{r,s}(y)) \, dsdr
\]
\[
\leq 2\pi \|\mathbf{t}\| \int_{0}^{\infty} \int_{0}^{\infty} \cosh(2\pi \|\mathbf{t}\| r) P_{r,s}(y) \, dsdr .
\]
Taking expectations on both sides, we get
\[
\mathbb{E}[\sinh(-2\pi \langle y, t \rangle) \langle y, \mathbf{v} \rangle] \leq 2\pi \|\mathbf{t}\| \int_{0}^{\infty} \int_{0}^{\infty} \cosh(2\pi \|\mathbf{t}\| r) \Pr[P_{r,s}(y)] \, dsdr .
\]
As in the previous proof, note that
\[
\Pr[P_{r,s}(y)] \leq \min \left( \frac{\epsilon}{1 + \epsilon}, 2e^{-\pi s^2}, 2e^{-\pi r^2} \right)
\]
by Lemma 2.15. So, we partition the positive quadrant of the \( (r, s) \)-plane into three regions and bound the integral separately in each section.

1. When \( s \leq s_0 \) and \( r \leq s_0 \), \( \Pr[P_{r,s}(y)] \) is at most \( \epsilon/(1 + \epsilon) \), and the integral in this region is bounded by
\[
\frac{\epsilon}{1 + \epsilon} \cdot \int_{0}^{s_0} \int_{0}^{s_0} \cosh(2\pi r \|\mathbf{t}\|) \, dsdr = \frac{\epsilon}{1 + \epsilon} \cdot \frac{s_0}{2\pi \|\mathbf{t}\|} \sinh(2\pi s_0 \|\mathbf{t}\|) .
\]
2. When \( s \leq r \) and \( r > s_0 \), \( \Pr[P_{r,s}(y)] \) is at most \( 2e^{-\pi r^2} \), and the integral in this region is bounded by

\[
2 \int_{s_0}^{\infty} \int_{0}^{r} \cosh(2\pi \|t\|r)e^{-\pi r^2} \, dsdr = \frac{1}{\rho(t)} \int_{s_0}^{\infty} \left( r e^{-\pi(r-\|t\|)^2} + r e^{-\pi(r+\|t\|)^2} \right) \, dr
\]

\[
= \frac{1}{2\pi \rho(t)} \left( e^{-\pi(s_0-\|t\|)^2} + e^{-\pi(s_0+\|t\|)^2} \right) + \frac{\|t\|}{\rho(t)} \int_{s_0-\|t\|}^{s_0+\|t\|} e^{-\pi z^2} \, dz
\]

\[
= \frac{1}{2\pi} \frac{\varepsilon}{1+\varepsilon} \cdot \cosh(2\pi s_0 \|t\|) + \frac{\|t\|}{\rho(t)} \int_{s_0-\|t\|}^{s_0+\|t\|} e^{-\pi z^2} \, dz.
\]

3. When \( s > r \) and \( s > s_0 \), \( \Pr[D_L^C | P_{r,s}(y)] \) is at most \( 2e^{-\pi s^2} \). So, the integral in this region is bounded by

\[
2 \int_{s_0}^{\infty} \int_{0}^{s} \cosh(2\pi \|t\|r)e^{-\pi s^2} \, drds = \frac{1}{\pi \|t\|} \int_{s_0}^{\infty} \sinh(2\pi \|t\|s)e^{-\pi s^2} \, drds
\]

\[
= \frac{1}{2\pi \|t\| \rho(t)} \int_{s_0-\|t\|}^{s_0+\|t\|} e^{-\pi z^2} \, dz.
\]

Combining everything together, and applying Lemma 2.11

\[
\left\| \mathbb{E}_{x \sim D_{C+1}} [x - t] \right\|
\]

\[
\leq \frac{\rho(t)}{\rho(C + t)} \left( \varepsilon \cdot (s_0 \sinh(2\pi s_0 \|t\|) + \|t\| \cosh(2\pi s_0 \|t\|)) + \frac{1+\varepsilon}{\rho(t)} (1 + 2\pi \|t\|^2) \int_{s_0-\|t\|}^{s_0+\|t\|} e^{-\pi z^2} \, dz \right)
\]

\[
\leq \frac{\varepsilon}{1+\varepsilon} \cdot (s_0 \sinh(2\pi s_0 \|t\|) + \|t\| \cosh(2\pi s_0 \|t\|)) + \frac{1}{\rho(t)} (1 + 2\pi \|t\|^2) \int_{s_0-\|t\|}^{s_0+\|t\|} e^{-\pi z^2} \, dz.
\]

\[
\square
\]

5 Exact CVPP with a promise

Recall the periodic Gaussian function \( f(t) \), defined in Eq. (3). As described in the introduction, this function is at the heart of the GapCVPP algorithm of [AR05] and the BDDP algorithm of Liu et al. [LLM06]. The reason it is so useful is that it can be well approximated by a function that can be computed by a small circuit. To see this, we first observe, following [AR05], that the Poisson summation formula implies the identity

\[
f(t) = \mathbb{E}_{w \sim D_{C}} [\cos(2\pi (w, t))], \tag{6}
\]

and then by replacing the expectation with an average over a set of samples, we obtain the approximator \( f_W \) defined as

\[
f_W(t) = \frac{1}{N} \sum_{i} \cos(2\pi (w_i, t)),
\]

where the \( W = (w_1, \ldots, w_N) \) are sampled from \( D_C \). We will use \( f_W \) to effectively perform gradient ascent on \( f \), and we will show that this finds the nearest lattice vector when \( \text{dist}(t, C) = O(\sqrt{\log(1/\varepsilon)/\eta_c(C^*)}) \) for \( \varepsilon = 1/poly(n) \). By Lemma 2.13, this implies the result of [LLM06].
Recall that Lemma 4.2 shows that when \( t \) is sufficiently close to the lattice, \( \nabla f(t)/(2\pi f(t)) \) is close to the difference between \( t \) and the nearest lattice point. Our result will follow easily from a similar proposition about \( f_W \).

**Proposition 5.1.** Let \( \mathcal{L} \subset \mathbb{R}^n \) be a lattice with \( \rho(\mathcal{L}) = 1 + \epsilon \) for \( 1/poly(n) < \epsilon = \epsilon(n) < 1/100 \) with \( 1/\epsilon \in poly(n) \). Let \( s_0^2 = \frac{1}{\pi} \log \frac{2(1+\epsilon)}{\epsilon} \) and \( W = (w_1, \ldots, w_N) \) be sampled independently from \( D_{\mathcal{L}^*} \). If \( N = \Omega(n \log(1/\epsilon)/\epsilon^{1/50}) \), then, with probability at least \( 1 - 2^{-\Omega(n)} \)

\[
\left\| \frac{\nabla f_W(t)}{2\pi f_W(t)} + t \right\| \leq \frac{3}{4} \| t \|
\]

for all \( t \in \mathbb{R}^n \) that satisfy \( \lambda_1(\mathcal{L})/(4n) \leq \| t \| \leq s_0/10 \).

We first assume the proposition and prove the main result of this section.

**Theorem 5.2.** Let \( 1/poly(n) < \epsilon = \epsilon(n) < 1/100 \). Let \( s_0^2 = \frac{1}{\pi} \log \frac{2(1+\epsilon)}{\epsilon} \) and \( \phi(\mathcal{L}) = s_0/(10\eta_1(\mathcal{L}^*)) \). Then, there exists an efficient algorithm that solves 1-CVPP\# with only \( N = O(n \log(1/\epsilon)/\epsilon^{1/50}) \) vectors from preprocessing. The run-time is \( O(Nn \log n) \) arithmetic operations.

**Proof.** We present an algorithm with probabilistic preprocessing and argue that with positive probability preprocessing will output advice that results in a successful query algorithm on all relevant inputs. Clearly this implies a deterministic algorithm.

The preprocessing takes as input a lattice \( \mathcal{L} \subset \mathbb{R}^n \). It computes a basis \( B \) as in Lemma 2.21 and \( \lambda_1(\mathcal{L}) \). Let \( N = O(n \log(1/\epsilon)/\epsilon^{1/50}) \), be sufficiently large to satisfy Proposition 5.1. Preprocessing then samples \( W = (w_1, \ldots, w_N) \) from \( D_{\mathcal{L}, \eta_1(\mathcal{L}^*)} \). Finally, it simply returns \( B, W, \lambda_1(\mathcal{L}) \) as advice.

The query algorithm takes a target point \( t \in \mathbb{R}^n \) as input and advice \( B, W, \lambda_1(\mathcal{L}) \). It then repeatedly updates \( t \leftarrow t + \nabla f_W(t)/(2\pi f_W(t)) \) and computes \( y = B\text{BA}B\text{A}(t, B) \) until \( \| y - t \| \leq \lambda_1(\mathcal{L})/2 \). It then returns \( y \).

We assume without loss of generality that \( \rho(\mathcal{L}) = 1 + \epsilon \) so \( \eta_1(\mathcal{L}^*) = 1 \). And, because the algorithm is periodic over the lattice, we can also assume that \( 0 \) is the closest lattice vector and therefore \( \| t \| \leq s_0/10 \).

Then, by Proposition 5.1 with positive probability taken over the preprocessing, we get closer to the nearest lattice point by a factor of at least \( 3/4 \) in each step until we are close enough that \( B\text{BA}B\text{A} \) will return the nearest point by Lemma 2.21. This takes at most \( O(\log n) \) steps. The run-time of each step is clearly dominated by the time that it takes to take \( N \) dot products. So, the total run-time is \( O(Nn \log n) \) arithmetic operations.

### 5.1 Proof of Proposition 5.1

We first show that \( f \) has the property that we would like to show that \( f_W \) has. The proof is simply a technical calculation bounding the right hand side of the inequality in Lemma 4.2.

**Lemma 5.3.** Let \( \epsilon \in (0, 1/100) \) and \( \mathcal{L} \subset \mathbb{R}^n \) a lattice with \( \rho(\mathcal{L}) = 1 + \epsilon \). Let \( s_0^2 = \frac{1}{\pi} \log \frac{2(1+\epsilon)}{\epsilon} \). Then, for any \( t \in \mathbb{R}^n \) with \( \| t \|/s_0 = \alpha \leq 1/100 \),

\[
\left\| \frac{\nabla f(t)}{2\pi f(t)} + t \right\| \leq \frac{\| t \|}{2}.
\]
Proof. Recall from by Lemma 4.2 that
\[
\|\nabla f(t) - f(t)\|_{2\pi} \leq \epsilon s_0(\sinh(2\pi s_0^2 a) + \alpha \cosh(2\pi s_0^2 a)) + (1 + \epsilon)(1 + 2\pi s_0^2 a^2) \cdot e^{\pi s_0^2 a^2} \int_{\pi a (1-\alpha)}^{\pi (1+a)} e^{-\pi z^2} dz.
\]
Because \(\sinh\) is convex on \([0, \infty)\), \(\sinh(0) = 0\) and \(a \leq 1/10\),
\[
\sinh(2\pi a s_0^2) \leq (1 - 10a) \sinh(0) + 10a \sinh(\pi s_0^2/5) \leq 5ae^{\pi s_0^2/5}.
\]
Using the above and \(2a \leq 1/5\), we have that
\[
\epsilon s_0(\sinh(2\pi s_0^2 a) + \alpha \cosh(2\pi s_0^2 a)) \leq \epsilon s_0(5ae^{\pi s_0^2/5} + \alpha \cosh(\pi s_0^2/5))
\leq 6\epsilon s_0 a e^{\pi s_0^2/5} = 6\|t\|(1 + \epsilon)^{1/5}(2/\epsilon)^{1/5}\epsilon
= 12\|t\|(1 + \epsilon)^{1/5}(\epsilon/2)^{4/5}.
\]
Turning to the integral,
\[
(1 + \epsilon)e^{\pi s_0^2 a^2} \int_{\pi a (1-\alpha)}^{\pi (1+a)} e^{-\pi z^2} dz \leq (1 + \epsilon)e^{\pi s_0^2 a^2} \cdot 2s_0 a e^{-\pi (1-a)^2 s_0^2}
\leq 2\|t\|(1 + \epsilon)\left(\frac{\epsilon}{2}\right)^{4/5}.
\]
Combining everything together,
\[
\|\nabla f(t) - f(t)\|_{2\pi} \leq \|t\|(1 + \epsilon)^{1/5}(\epsilon/2)^{4/5}(14 + 4\pi s_0^2 a^2)
\leq \|t\|(1 + \epsilon)^{1/5}(\epsilon/2)^{4/5}(14 + \pi s_0^2/25).
\]
The result follows from noting that \((\epsilon/2)^{4/5}(1 + \epsilon)^{1/5}(14 + \pi s_0^2/25) < 1/2\) for \(\epsilon \leq 1/100\). \(\square\)

So, it suffices to show that \(\nabla f_W(t) / f_W(t)\) is a good estimator of \(\nabla f(t) / f(t)\). [AR05] proved the following lemma, which shows that the denominator, \(f_W(t)\), is a good estimator of \(f(t)\), and we will show something similar for the numerator.

Lemma 5.4. For \(\mathcal{L} \subset \mathbb{R}^n\) a lattice, \(W = (w_1, \ldots, w_N)\) sampled independently from \(D_{\mathcal{L}}\), \(t \in \mathbb{R}^n\), and \(s \geq 0\),
\[
\Pr[|f_W(t) - f(t)| \geq s] \leq 2^{-\Omega(N s^2)}.
\]

Lemma 5.5. Let \(\mathcal{L} \subset \mathbb{R}^n\) be a lattice with \(\rho(\mathcal{L}) = 1 + \epsilon\) for some \(\epsilon \in (0,1)\). Let \(W = (w_1, \ldots, w_N)\) be sampled independently from \(D_{\mathcal{L}}\). Then, for any \(t \in \mathbb{R}^n\), \(s \geq 0\),
\[
\Pr[\|\nabla f_W(t) - \nabla f(t)\| > s\|t\|] \leq 2^{-\Omega(N \min(s, s^2)) + O(n)}.
\]
Proof. For any \( i \) and any unit vector \( \mathbf{v} \),
\[
|\langle \nabla f_{\{w_i\}}(t), \mathbf{v} \rangle| = 2\pi |\mathbf{w}_i, \mathbf{v}| \sin(2\pi (\mathbf{w}_i, t))| \leq 4\pi^2 |\mathbf{w}_i, \mathbf{v}| |\mathbf{w}_i, t||.
\]
It follows from the subgaussianity of the discrete Gaussian and Corollary 2.18 that \( \langle \nabla f_{\{w_i\}}(t), \mathbf{v} \rangle / \|t\| \) is subexponential with parameter \( O(1) \), and therefore by Lemma 2.19 for any \( r \geq 0 \),
\[
\Pr[|\langle \nabla f_W(t) - \nabla f(t), \mathbf{v} \rangle| \geq r \|t\|] \leq 2^{-O(n \min(r, r^2))}.
\]
By Lemma 2.23 there is a \( \frac{1}{2} \)-net of the sphere, \( A \) with \( |A| = 2^{O(n)} \). Taking a union bound over \( A \),
\[
\Pr[\exists \mathbf{v} \in A : |\langle \nabla f_W(t) - \nabla f(t), \mathbf{v} \rangle| \geq r \|t\|] \leq 2^{-O(n \min(r, r^2))} + O(n).
\]
Applying Lemma 2.24 and setting \( r = s/2 \) gives the result. \( \square \)

With this, we can show that the combined estimator \( \nabla f_W(t) / f_W(t) \) is a very good estimator of \( \nabla f(t) / f(t) \) when \( w_i \) are sampled from \( D_L^* \). Specifically,

Lemma 5.6. Let \( L \subset \mathbb{R}^n \) be a lattice with \( \rho(L) = 1 + \varepsilon \) for some \( \varepsilon \in (0, 1/100) \), and let \( s_0 \) be defined by \( s_0^2 = \frac{1}{n} \log \frac{2(1+\varepsilon)}{\varepsilon} \). Let \( W = (w_1, \ldots, w_N) \subset L^* \) be sampled independently from \( D_L^* \). Then, for any \( t \in \mathbb{R}^n \) with \( \|t\| \leq s_0/10 \) and \( 0 < s \leq 1 \), if \( N \geq \Omega(n/(\varepsilon^{1/50}s^2)) \),
\[
\Pr \left[ \| \frac{\nabla f_W(t)}{f_W(t)} - \frac{\nabla f(t)}{f(t)} \| > s \|t\| \right] \leq 2^{-O(\varepsilon^{1/50}s^2)}.
\]

Proof. Note that
\[
\frac{\rho(t)}{1 + \varepsilon} \geq e^{-\pi s_0^2/100} \geq \frac{1}{3} \cdot \varepsilon^{1/100}.
\]
Let \( r_f = |f_W(t) / f(t) - 1| \). We bound the probability of two ”bad events”. First,
\[
\Pr \left[ r_f \geq \frac{1}{2} \right] \leq 2^{-\Omega(n f(t)^2)} \quad \text{(Lemma 5.4)}
\]
\[
\leq 2^{-\Omega(\varepsilon^{1/50}N)} \quad \text{(Eq. 7)}.
\]
And, by Lemma 5.5
\[
\Pr \left[ \| \nabla f_W(t) - \nabla f(t) \| > \frac{\varepsilon^{1/100}}{12} s \|t\| \right] \leq 2^{-\Omega(\varepsilon^{1/50}N \min(s, s^2)) + O(n)}.
\]
Assuming neither of the ”bad events” happens, we have
\[
\left\| \frac{\nabla f_W(t)}{f_W(t)} - \frac{\nabla f(t)}{f(t)} \right\| \leq \left\| \frac{\nabla f_W(t) - \nabla f(t)}{(1 - r_f) f(t)} \right\| + r_f \left\| \frac{\nabla f(t)}{(1 - r_f) f(t)} \right\|
\leq \left\| \frac{\nabla f_W(t) - \nabla f(t)}{(1 - r_f) f(t)} \right\| + 3\pi \|t\| \frac{r_f}{1 - r_f}
\leq 3(1/\varepsilon)^{1/100} \cdot \| \nabla f_W(t) - \nabla f(t) \| + 3\pi \|t\| \frac{r_f}{1 - r_f}
\leq s \|t\|.
\]
The result follows from applying union bound. \( \square \)
Next, we will show that the Hessian $Hf(t)$ has small spectral norm, a result which also appeared in [AR05]. We will use this to show that $Hf_w(t)$ also has small spectral norm with high probability. (The bound that is implicit in [AR05] requires $N$ to be larger than we want.) It follows that the gradients of both functions don’t vary much over small distances.

**Lemma 5.7.** Let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice. For any $t \in \mathbb{R}^n$,

$$\|Hf(t)\| \leq 2\pi.$$  

Moreover, let $W = (w_1, \ldots, w_N)$ be sampled independently from $D_{\mathcal{L}^*}$. Then, for any $s \geq 10$, if $N = \Omega(n/s)$, then

$$\Pr[\exists t \in \mathbb{R}^n : \|Hf_W(t)\| \geq s] \leq 2^{-\Omega(sN)}.$$  

**Proof.** From Eq. (6), we have that for any $r$

$$\|Hf(t)\| \leq 2\pi.$$  

By essentially the same calculation, $\|Hf_W(t)\| \leq \|Hf_W(0)\|$. So, it suffices to simply consider the two Hessians at 0.

From Eq. (6), we have a representation of $Hf(0)$ in both the primal and the dual,

$$-\frac{1}{2\pi}Hf(0) = I_n - 2\pi \mathbb{E}_{y \sim D_{\mathcal{L}^*}}[yy^T] = 2\pi \mathbb{E}_{w \sim D_{\mathcal{L}^*}}[ww^T]$$

where $I_n$ is the $n \times n$ identity matrix. Noting that all terms are positive semidefinite, it follows that the norm of $\mathbb{E}_{w \sim D_{\mathcal{L}^*}}[ww^T]$ is bounded by $1/(2\pi)$. Therefore, $\|Hf(0)\| \leq 2\pi$.

Similarly, note that

$$\|Hf_W(0)\| = \sup_{\|v\| = 1} |\langle Hf_W(0)v, v \rangle| = \frac{4\pi^2}{N} \sup_{\|v\| = 1} \sum_i \langle v, w_i \rangle^2.$$  

By Lemma 2.15, $w_i$ are subgaussian random variables with parameter 1. It follows from Lemma 2.17 that $\langle w_i, v \rangle$ is subexponential with parameter $O(1)$. Then, applying Lemma 2.19,

$$\Pr[|\langle Hf_W(0)v, v \rangle| \geq r - 2\pi] \leq 2^{-\Omega(N \min(r, r^2))}$$

for any $r \geq 0$. Then, by Lemma 2.23, there is a $\frac{1}{4}$-net of the unit sphere, $A$ with $|A| = 2^{O(n)}$. Union bounding over $A$,

$$\Pr[\exists v \in A : |\langle Hf_W(0)v, v \rangle| \geq r - 2\pi] \leq 2^{-\Omega(N \min(r, r^2))} + O(n).$$

Applying Lemma 2.26 and setting $r = s/2 - \pi$ shows that

$$\Pr[\|Hf_W(0)\| \geq s] \leq 2^{-(s-2\pi) \cdot \Omega(N) + O(n)} + 2^{-(s-2\pi)^2 \cdot \Omega(N) + O(n)} \leq 2^{-\Omega(sN) + O(n)}.$$  

The result follows. \qed
From this, we can show that $\nabla f_W(t) / f_W(t)$ can’t vary too much over small intervals.

**Lemma 5.8.** Let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice with $\rho(\mathcal{L}) = 1 + \varepsilon$ for $\varepsilon \in (0, 1/100)$. Let $s_0^2 = \frac{1}{2} \log \frac{2(1+\varepsilon)}{\varepsilon}$ and $W = (w_1, \ldots, w_N)$ be sampled independently from $D_{\mathcal{L}}$ with $N \geq \Omega(n)$. For any $t \in \mathbb{R}^n$, let $\alpha = \|t\| / s_0$ and $\delta = \varepsilon^4 \alpha$. If $0 < \alpha \leq 1/10$, then

$$\Pr \left[ \exists t' \in \mathbb{R}^n, \|t - t'\| \leq \delta : \left\| \frac{\nabla f_W(t)}{f_W(t)} - \frac{\nabla f_W(t')}{f_W(t')} \right\| \geq \frac{\pi}{8} \|t\| \right] \leq 2^{-\Omega(Ns_0^2)}.$$

**Proof.** We show that three bounds hold with high probability.

1. Note that $f(t) \geq \rho(t) / \rho(\mathcal{L}) > (1/3) \cdot \varepsilon^2$. Then, from Lemma 5.4 we have that

$$f_W(t) > \frac{1}{4} \cdot \varepsilon^2$$

with probability at least $1 - 2^{-\Omega(Ns_0^2)}$.

2. Plugging $s = \varepsilon^2 / (100 \delta)$ into Lemma 5.7 we have that with probability at least $1 - 2^{-\Omega(N/\delta)}$,

$$\|\nabla f_W(t) - \nabla f_W(x)\| < \frac{1}{100} \cdot \varepsilon^2 \|t\|$$

for all $x \in \mathbb{R}^n$ with $\|x - t\| \leq \delta$.

3. By Lemma 5.3 we have that $\nabla f(t) \leq 3\pi \|t\|$. Then, by Lemma 5.5

$$\|\nabla f_W(t)\| < 4\pi \|t\|$$

with probability at least $1 - 2^{-\Omega(N) + O(n)}$.

Now, suppose all of the above bounds hold. Then, it follows from our bound on $\|\nabla f_W(t)\|$ and $\|\nabla f_W(t) - \nabla f_W(x)\|$ that $|f_W(t) - f_W(t')| < 2\varepsilon^3$. Putting everything together,

$$\left\| \frac{\nabla f_W(t)}{f_W(t)} - \frac{\nabla f_W(t')}{f_W(t')} \right\| < \left\| \frac{\nabla f_W(t)}{f_W(t)} - \frac{\nabla f_W(t')}{f_W(t')} \right\| + 2 \frac{\varepsilon^3}{f_W(t)} \cdot \left\| \frac{\nabla f_W(t)}{f_W(t)} - 2\varepsilon^3 \right\| \leq \frac{\pi}{8} \|t\|.$$

The result follows from union bound. ∎

**Proof of Proposition 5.2** Recall that we are only concerned with $t$ such that $\lambda_1(\mathcal{L}) / (4n) \leq \|t\| \leq s_0/10$. We wish to find a set a vectors $A = \{t_j\}$ such that for any relevant target $t$, there is a $t_j$ with $\|t - t_j\| \leq \varepsilon^4 \|t_j\| / s_0$, allowing us to apply Lemma 5.8. For $i = -10 \lfloor \log n \rfloor$ to $\lfloor \log s_0 \rfloor$, let $A_i$ be a $2i \varepsilon^4 / s_0$-net of the shell of inner radius radius $2^i$ and outer radius $2^{i+1}$. By Lemma 2.23 we can take $|A_i| = 2^{O(n \log(1/\varepsilon))}$. Let $A = \cup A_i$. There are $O(\log n)$ such nets, so $|A| = 2^{O(n \log(1/\varepsilon))}$.

For any target $t$ in the relevant range, let $t_j$ be the nearest point in $A$ to $t$. We show that two bounds hold for all pairs $t, t_j$ simultaneously with high probability.

1. By Lemma 5.8 and union bound over $A$, we have that

$$\left\| \frac{\nabla f_W(t)}{f_W(t)} - \frac{\nabla f_W(t_j)}{f_W(t_j)} \right\| \leq \frac{\pi}{10} \|t\|$$

holds for all relevant pairs $t, t_j$ simultaneously with probability at least $1 - 2^{-\Omega(\varepsilon^4 s_0/100) + O(n \log(1/\varepsilon))}$. 21
2. Similarly, plugging $s = \pi/10$ into Lemma 5.6 and applying union bound over the $A$ gives that

$$\left\| \frac{\nabla f_W(t_j)}{f_W(t_j)} - \frac{\nabla f(t_j)}{f(t_j)} \right\| \leq \frac{\pi}{10} \|t\|$$

holds for all relevant pairs $t, t_j$ with probability at least $1 - 2^{-\Omega(1/50N) + O(n \log(1/\epsilon))}$.

So, assume both bounds hold. Then, applying triangle inequality,

$$\left\| \frac{\nabla f_W(t)}{f_W(t)} - \frac{\nabla f(t)}{f(t)} \right\| \leq \frac{\pi}{5} \|t\| + \left\| \frac{\nabla f(t_j) - \nabla f(t)}{f(t_j)} \right\| + |f(t) - f(t_j)| \cdot \left\| \frac{\nabla f(t)}{f(t)f(t_j)} \right\|$$

$$\leq \frac{\pi}{5} \|t\| + \frac{\epsilon^4 \|t\|}{s_0 f(t_j)} + |f(t) - f(t_j)| \cdot \left\| \frac{\nabla f(t)}{f(t)f(t_j)} \right\| \quad \text{(Lemma 5.7)}$$

$$< \frac{\pi}{4} \|t\| \quad \text{(Lemma 5.3)}.$$

So,

$$\left\| \frac{\nabla f_W(t)}{2\pi f_W(t)} + t \right\| < \left\| \frac{\nabla f(t)}{2\pi f(t)} + t \right\| + \frac{1}{8} \|t\|$$

$$< \frac{3}{4} \|t\| \quad \text{(Lemma 5.3)}.$$

The result follows from union bound.

\[\square\]

References


