2016 IEEE 57th Annual Symposium on Foundations of Computer Science

Optimizing Star-Convex Functions

Jasper C.H. Lee
Department of Computer Science
Brown University
Providence, RI
jasperchlee@brown.edu

Paul Valiant
Department of Computer Science
Brown University
Providence, RI
paul_valiant@brown.edu

***Abstract*—Star-convexity is a significant relaxation of the notion of convexity, that allows for functions that do not have (sub)gradients at most points, and may even be discontinuous everywhere except at the global optimum. We introduce a polynomial time algorithm for optimizing the class of star-convex functions, under no Lipschitz or other smoothness assumptions whatsoever, and no restrictions except exponential boundedness on a region about the origin, and Lebesgue measurability. The algorithm's performance is polynomial in the requested number of digits of accuracy and the dimension of the search domain. This contrasts with the previous best known algorithm of Nesterov and Polyak which has exponential dependence on the number of digits of accuracy, but only n^ω dependence on the dimension n (where ω is the matrix multiplication exponent), and which further requires Lipschitz second differentiability of the function [1].**

Despite a long history of successful gradient-based optimization algorithms, star-convex optimization is a uniquely challenging regime because 1) gradients and/or subgradients often do not exist; and 2) even in cases when gradients exist, there are star-convex functions for which gradients provably provide no information about the location of the global optimum. We thus bypass the usual approach of relying on gradient oracles and introduce a new randomized cutting plane algorithm that relies only on function evaluations. Our algorithm essentially looks for structure at all scales, since, unlike with convex functions, star-convex functions do not necessarily display simpler behavior on smaller length scales. Thus, while our cutting plane algorithm refines a feasible region of exponentially decreasing volume by iteratively removing "cuts", unlike for the standard convex case, the structure to efficiently discover such cuts may not be found within the feasible region: our novel star-convex cutting plane approach discovers cuts by sampling the function exponentially far outside the feasible region.

We emphasize that the class of star-convex functions we consider is as unrestricted as possible: the class of Lebesgue measurable star-convex functions has theoretical appeal, introducing to the domain of polynomial-time algorithms a huge class with many interesting pathologies. We view our results as a step forward in understanding the scope of optimization techniques beyond the garden of convex optimization and local gradient-based methods.

I. Introduction

Optimization is one of the most influential ideas in computer science, central to many rapidly developing areas within computer science, and also one of the primary exports to other fields, including operations research, economics and finance, bioinformatics, and many design problems in engineering. Convex optimization, in particular, has produced many general and robust algorithmic frameworks that have each become fundamental tools in many different areas: linear programming has become a general modeling tool, its simple structure powering many algorithms and reductions; semidefinite programming is an area whose scope is rapidly expanding, yielding many of the best known approximation algorithms for optimizing constraint satisfaction problems [2]; convex optimization generalizes both of these and has introduced powerful optimization techniques including interior point and cutting plane methods. Our developing understanding of optimization has also led to new algorithmic design principles, which in turn leads to new insights into optimization. Recent progress in algorithmic graph theory has benefited enormously from the optimization perspective, as many recent results on max flow/min cut [3], [4], [5], [6], [7], bipartite matching [3], and Laplacian solvers [8], [6], [9] have made breakthroughs that stem from developing a deeper understanding of the characteristics of convex optimization techniques in the context of graph theory. These successes motivate the quest for a deeper and broader understanding of optimization techniques: 1) to what degree can convex optimization techniques be extended to non-convex functions; 2) can we develop general new tools for tackling non-convex optimization problems; and 3) can new techniques from non-convex optimization yield new insights into convex optimization?

As a partial answer to the first question, gradient descent—perhaps the most natural optimization approach—has had enormous success recently in a variety of practically-motivated non-convex settings, sometimes with provable guarantees. The method and its variants are the de facto standard for training (deep) neural networks, a hot topic in high dimensional non-convex optimization, with many recent practical results (e.g. [10], [11]). Gradient descent can be thought of as a "greedy" algorithm, which repeatedly chooses the most attractive direction from the local landscape. The efficacy of gradient descent algorithms relies on local assumptions about the function: for example, if the first derivative is Lipschitz (slowly varying), then one can take large downhill steps in the gradient direction without worrying that the function will change to going uphill along this direction. Thus the con-

0272-5428/16 $31.00 © 2016 IEEE
DOI 10.1109/FOCS.2016.71

vergence of gradient descent algorithms typically depends on a Lipschitz parameter (or other smoothness measure), and conveniently does not depend on the dimension of the search space. Many of these algorithms converge to within ϵ of a local optimum in time $poly(1/\epsilon)$, with additional polynomial dependence on the Lipschitz constant or other appropriate smoothness guarantee [12], [13]. In cases where all local optima are global optima, then one has global convergence guarantees.

While one intuitively expects gradient descent algorithms to always converge to a local minimum, in this paper we study the optimization of a natural generalization of convex functions where, despite the global optimum being the only stationary point/local minimum for every function in this class, provably no variant of gradient descent converges in polynomial time. The class of star-convex functions, which we define below, includes many functions of both practical and theoretical interest, both generalizing common families of convex functions to wider parameter regimes, and introducing new "pathologies" not found in the convex case (for a complete discussion, see [14]). We show, essentially, how to make a gradient-based cutting plane algorithm "robust" to many new pathologies, including lack of gradients or subgradients, long narrow ridges and rapid oscillation in directions transverse to the global minimum, and, in fact, almost arbitrary discontinuities. This challenging setting gives new insights into what fundamentally enables cutting plane algorithms to work, which we view as progress towards answering questions 2 and 3 above.

A. Star-Convex Functions

This paper focuses on the optimization of *star-convex* functions, a particular class of (typically) non-convex functions that includes convex functions as a special case. We define these functions as follows, based on the definition in Nesterov and Polyak [1].

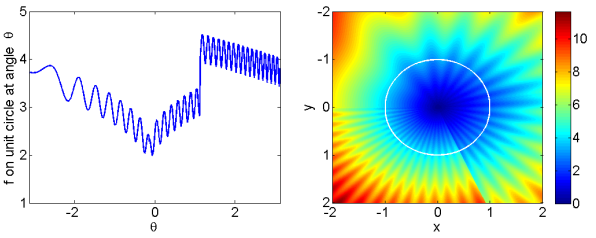
Definition 1 (Star-convex functions)**.** *A function* $f : \mathbb{R}^n \to \mathbb{R}$ *is* star-convex *if there is a global minimum* $x^* \in \mathbb{R}^n$ *such that for all* $\alpha \in [0, 1]$ *and* $x \in \mathbb{R}^n$*,*

$$f(\alpha x^* + (1-\alpha)x) \leq \alpha f(x^*) + (1-\alpha)f(x)$$

We call such an x^* *a* star center *of* f*. For convenience, we shall refer to the minimum function value as* f^**.*

The name "star-convex" comes from the fact that each sublevel set (the set of x for which $f(x) < c$, for some c) is "star-shaped".

Intuitively, if we visualize the objective function as a landscape, star-convexity means that the global optimum is "visible" from every point—there are no "ridges" on the way to the global optimum, but there could be many ridges in transverse directions. Since the global optimum is always visible in a downhill direction from every point, gradient descent methods would *seem* to be effective.



(a) g defined on the unit circle (b) Linear extension of g to f

Figure 1: An example star-convex function f defined by linearly extrapolating an *arbitrary* positive function g defined on the unit circle (in white).

Counterintuitively, these methods all fail in general (see Section I-D for details).

One broad class of star-convex functions that gives a good sense of the scope of this definition is constructed by the following process: 1) pick an *arbitrary* positive function $g(\theta)$ on the unit circle (see Figure 1a) which may be discontinuous, rapidly oscillating, or otherwise badly behaved; and 2) linearly extend this function to a function f on the entire plane, about the value $f(0) = 0$ (see Figure 1b), defining

$$f(x) = ||x||_2 \cdot g\left(\frac{x}{||x||_2}\right)$$

At any point at which a gradient or subgradient exists, the star center (global optimum) lies in the halfspace opposite the (sub)gradient. However, even for the relatively benign example of Figure 1, gradients do not exist for angles θ at which g is discontinuous, and subgradients do not exist for most angles, including those angles where g is a local maxima with respect to θ. Further, even for differentiable star-convex functions, gradients can be misleading, since a rapidly oscillating g implies that gradients typically point in the transverse direction, nearly orthogonal to the direction of the star center.

While it is often standard to design algorithms assuming one has access to an oracle that returns both function values and gradients, we instead only assume access to the function value: even in the case when gradients exist, it is unclear whether they are algorithmically helpful in our setting. (See Section I-D for details.) Indeed, we pose this as an open problem: under what assumptions (short of the Lipschitz guarantees on the second derivative of Nesterov and Polyak [1]), can one meaningfully use a gradient oracle to optimize star-convex functions? For example, is there a natural gradient-based algorithm whose performance depends polynomially or even polylogarithmically on parameter L when the function is L-Lipschitz and differentiable?

B. Main Result

In this paper, we show that, assuming only Lebesgue measurability and an exponential bound on the function value within a large ball, our algorithm optimizes a star-convex function in $\mathrm{polylog}(1/\epsilon)$ time where ϵ is the desired accuracy in function value.

Theorem 2. *(Informal) Given evaluation oracle access to a Lebesgue measurable star-convex function $f : \mathbb{R}^n \to \mathbb{R}$ and an error parameter ϵ, with the guarantee that x^* is within radius R of the origin, our adapted ellipsoid algorithm returns an estimate f_0 of the minimum function value such that $|f_0 - f^*| \leq \epsilon$ with high probability in time* $\mathrm{poly}(n, \log \frac{1}{\epsilon}, \log R)$.

In previous work, Nesterov and Polyak [1] introduced an adaptation of Newton's method to find local minima in functions that are twice-differentiable, with Lipschitz continuity of their second derivatives. For the class of Lipschitz-twice-differentiable star-convex functions, they show that their algorithm converges to within ϵ of the global optimum in time $O(\frac{1}{\sqrt{\epsilon}})$, using star convexity to lower bound the amount of progress in each of their optimization steps.

Our results are stronger in two significant senses: 1) as opposed to assuming Lipschitz twice-differentiability, we make *no continuity assumptions whatsoever*, optimizing over an essentially arbitrary measurable function within the class of star-convex functions; 2) while the algorithm of [1] requires exponential time to estimate the optimum to k digits of accuracy, our algorithm is *polynomial in the requested number of digits of accuracy*.

The reader may note that the complexity of our algorithm does depend polynomially on the number of dimensions n of the search space, whereas the Nesterov-Polyak algorithm uses a number of calls to a Hessian oracle that is *independent* of the dimension n. However, our dependency on the dimension n is necessary in the absence of Lipschitz guarantees, even for convex optimization [15].

We further point out an important distinction between *star-convex* optimization and *star-shaped* optimization. A star-shaped function is defined similarly to a star-convex function, except the star center is allowed to be located away from the global minimum. Certain NP-hard optimization problems, including max-clique, can be rephrased in terms of star-shaped optimization [16], and the problem is in general impossible without continuity guarantees, for the global minimum may be hidden along a single ray from the star center that is discontinuous from the rest of the function. In this sense, our star-convex optimization algorithm narrowly avoids solving an NP-hard optimization problem.

C. Applications of Star-Convex Functions

Any polynomial of $|x|, |y|, |z|, \ldots$ with positive coefficients is star-convex, and this includes sums of squares of monomials, which arise in many different contexts. More generally, finite positive linear combinations of products of positive powers of $|x|, |y|, |z|$ are star-convex when raised to a sufficiently large positive power, and thus may also be optimized with our techniques. Such functions are seen in the following more general and practical setting.

Empirical risk minimization is a central technique in machine learning [17], where, given training data x_i with labels y_i, we seek a hypothesis h so as to minimize $\frac{1}{m}\sum_{i=1}^{m} L(h(x_i), y_i)$, where L is a *loss function* that describes the penalty for misprediction, on input our prediction $h(x_i)$ and the true answer y_i. We take the hypothesis h to be a linear function (this setting includes kernel methods that preprocess x_i first and then apply a linear function). If the loss function L is convex, then finding the optimal hypothesis h is a convex optimization problem, which is widely used in practice: for exponent $p \geq 1$, we let $L(\hat{y}_i, y_i) = |\hat{y}_i - y_i|^p$ and thus aim to optimize the convex function $\arg\min_h \sum_{i=1}^{m} |h \cdot x_i - y_i|^p$.

In this paper, instead, we draw attention to the interesting regime where $p < 1$. This regime is not accessible with standard techniques. Intriguingly, when the data y_i is consistent with some linear hypothesis, then the $(1/p)$th power of the above objective function is star-convex, and thus the main result of our paper yields an efficient optimization algorithm (however, removing the guarantee that the data is consistent with a linear hypothesis, the problem becomes NP-hard). Slightly more generally:

Corollary 3. *Consider data (x_i, y_i), a loss function $L(\hat{y}_i, y_i)$, and a power $p > 0$ such that there is a true hypothesis h such that for each example i, the loss $L(\hat{h}(x_i), y_i)^{1/p}$—considered as a function of $\hat{h}$—is star-convex, with a global optimum of 0 at $\hat{h} = h$. Then the following empirical risk minimization problem can be solved to accuracy ϵ in time $poly(n, \log 1/\epsilon)$ by inputting its $(1/p)$th power to the main algorithm of this paper:*

$$\arg\min_h \frac{1}{m}\sum_{i=1}^{m} L(h(x_i), y_i),$$

The regime where $p \in (0, 1)$ has the structure that, when one is far from the right answer, small changes to the hypothesis do not significantly affect performance, but the closer one gets to the true parameters, the richer the landscape becomes. One of many interesting settings with this behavior is biological evolution, where "fit" creatures have a rich landscape to traverse, while drastically unfit creatures fail to survive. A paper by one of the authors proposed star-convex optimization as a regime where evolutionary algorithms might be unexpectedly successful [18]. This current work finds an affirmative answer to an open question raised there by showing the first

polynomial time algorithm for optimizing this general class of functions. (The previous results by Nesterov and Polyak [1] fail to apply to this setting because the objective function has regions that behave—for some parameter regimes—like $\sqrt{|x|}$, which is not Lipschitz.)

Corollary 4 (Extending Theorem 4.5 of [18])**.** *There is a single mutation algorithm under which for any $p > 0$, defining the loss function $L(\hat{y}, y) = |\hat{y} - y|^p$, the class of constant-degree polynomials from $\mathbb{R}^n \to \mathbb{R}^m$ with bounded coefficients is evolvable with respect to all distributions over the radius r ball.*

D. Need for Novel Algorithmic Techniques

Before we outline our strategy for optimizing star-convex functions, we demonstrate the need for novel algorithmic approaches by explaining how a wide variety of standard techniques fail on this challenging class of functions. We start by explaining how simple star-convex functions such as $(\sqrt{|x|} + \sqrt{|y|})^2$ confound gradient descent and its variants, and end with a pathological example with information-theoretic guarantees about its hardness to optimize, even given arbitrarily accurate first-order (gradient) oracle access.

Consider the star-convex function $(\sqrt{|x|} + \sqrt{|y|})^2$, which can be thought of essentially as $\sqrt{|x|} + \sqrt{|y|}$. This function has unique global minimum (and star center) at the origin. Gradients of this function go to infinity as either x or y goes to 0, thus the function essentially has deep canyons along both the x and y axes, with gradients near either "canyon" (axis) leading the algorithm deeper into the canyon instead of towards the origin. Many variants of gradient descent will have the following behavior on this function: rapidly jump towards one of the two axes, and then fail to make significant further progress as the search point oscillates around that axis. This is in part a reflection of the fact that the gradient of $(\sqrt{|x|} + \sqrt{|y|})^2$ is *not* Lipschitz, and in fact varies arbitrarily rapidly as (x, y) converges towards either axis; since these axes are "canyons" in the search space, typical algorithms will spend a disproportionate amount of their time stumbling around this badly-behaved region. More sophisticated and modern gradient descent techniques have been developed to handle regimes with badly scaled gradients, including the *normalized* gradient descent algorithm [13]; however the "strict local quasi-convexity" property required by their analysis fails to hold for the above function.

Complementing the above example, which illustrates how standard gradient descent variants cannot optimize star-convex functions, we now present a more sophisticated and pathological example which proves that, even with access to a first-order oracle and the assumptions of infinite differentiability and boundedness on a region around the origin, no *deterministic* algorithm can efficiently optimize star-convex functions without Lipschitz guarantees, motivating the somewhat unusual randomized flavor of the algorithm of this paper.

Proposition 5. *For any deterministic polynomial time algorithm A, there is a set $X^* \subset [0, 1]$ covering at least half of the unit interval, such that for each $x^* \in X^*$ there is an infinitely differentiable star-convex function $f : \mathbb{R}^2 \to \mathbb{R}$ with unique global minimum $f(x^*, \sqrt{1/2}) = 0$, such that, when algorithm A is run on function f, the returned values and gradients are independent of x^*:*

1) *The function value is always $f(x, y) = 1$;*
2) *The gradient is a fixed function of y only, $\nabla f(x, y) = \left(0, \frac{1}{y - \sqrt{1/2}}\right)$.*

Since the function and gradient oracles return zero information about the x-coordinate of the star center $x^* \in X^*$, algorithm A clearly has a hopeless task optimizing this class of functions.

We construct the functions of Proposition 5 via a simulation argument—given an arbitrary algorithm A, simulate it using properties 1 and 2 to determine the results of all function and gradient queries; then define the function f "after the fact" via interpolation, to be consistent with the queried values and gradients.

In summary, gradient information is very hard to use effectively in star-convex optimization, even for smooth functions. Our algorithm, outlined below, instead only queries the function value—not because we object to gradients, but rather because they do not seem to help.

E. Our Approach

Our overall approach is via the ellipsoid method, which repeatedly refines an ellipsoidal region containing the star center (global optimum) by iteratively computing "cuts" that contain the star center, while significantly reducing the overall volume of the ellipsoid.

As mentioned in Section I-D, even for smooth functions with access to a gradient oracle, the cutting planes induced by the gradients may yield no significant progress in some directions. Finding "useful" cutting planes in the star-convex setting requires three novelties—see Section III for complete details. **First**, the star-convex function may be discontinuous, without even subgradients defined at most points, so we instead rely on a sampling process involving the "*blurred logarithm*" of the objective function. The blurred logarithm mitigates both the (potentially) exponential range of the function, and the (potentially) arbitrary discontinuities in the domain. The blurred logarithm, furthermore, is differentiable, and sampling results let us estimate and use its derivatives in our algorithm. This technique is similar to that of randomized smoothing [19]. **Second**, the negative gradient of the blurred logarithm might point away from the star center (global optimum)—despite all gradients of the unblurred function (when they exist) pointing towards

the star center—because of the way blurring interacts with sharp wedge-shaped valleys (see Figure 2 below). Addressing this requires an averaging technique that repeatedly samples Gaussians-in-Gaussians, until it detects sufficient conditions to conclude that the gradient may be used as a cut direction. **Third** and finally, the usual cutting plane criterion—that the volume of the feasible region decreases exponentially—is no longer sufficient in the star-convex setting, as an ellipsoid in two coordinates (x, y) might get repeatedly cut in the y direction without any progress restricting the range of x. This is provably not a concern in convex optimization (Theorem 5.2.1 of [20]). Our algorithm tackles this issue by "locking" axes of the ellipsoid smaller than some exponentially small threshold τ, and seeking cuts orthogonal to these axes; this orthogonal signal may be hidden by much larger derivatives in other directions, and hence requires new techniques to expose. The counterintuitive approach is that, in order to expose structure within an exponentially thin ellipsoid dimension we must search exponentially far outside the ellipsoid in this dimension.

F. Stochastic Optimization

Our approach extends easily to the stochastic optimization setting. Here, the objective function is the expectation over a distribution of star-convex functions which share the same star center and optimum value, e.g. the empirical risk minimization setting of Corollary 3.

The optimization algorithm for this new stochastic setting is actually identical to the general star-convex optimization algorithm. Since star-convex functions can take arbitrarily unrelated values on nearby rays from the star center, our algorithm is already fully equipped to deal with the situation, and adding explicitly unrelated values to the model by stochastically sampling from a family of functions L_i makes the problem no harder. The algorithm, analysis, and convergence are unchanged.

II. Problem Statement and Overview

Our aim here is to discuss algorithms for optimizing star-convex functions in the most general setting possible, and thus we must be careful about how the functions are specified to the algorithms. In particular, we do not assume continuity, so functions with one behavior on the rational points and a separate behavior on irrational points are possible. Therefore, it is essential that our algorithms have access to the function values of f at inputs beyond the usual rational points expressible via standard computer number representations. As motivation, see the following proposition.

Proposition 6. *For any integers i, j, there exists a (discontinuous) star-convex function $f_{i,j}$ with unique global minimum at $f_{i,j}(1/\sqrt{2}+i, 1/\sqrt{3}+j) = 0$, such that:*

1) *The function is essentially a cone on the irrational points—with probability 1 (over any continuous distribution on x and y),*
   $f_{i,j}(x, y) = ||(x, y) - (1/\sqrt{2}+i, 1/\sqrt{3}+j)||_2$*;*
2) *On every rational point (x, y) that is not within distance 1 of $(1/\sqrt{2}+i, 1/\sqrt{3}+j)$, the function is constant, $f_{i,j}(x, y) = 1$.*

If in Proposition 6, we choose i, j randomly from an exponentially large range, then any (deterministic or randomized) polynomial time algorithm that only queries f on rational points is exponentially unlikely to ever see a function evaluation that is not 1; and thus cannot optimize the function.

Since directly querying function values of general star-convex functions at rational points is so limiting, we instead introduce the notion of a *weak sampling evaluation oracle* for a star-convex function, adapting the definition of a *weak evaluation oracle* by Lovász [21].

Definition 7. *A* weak sampling evaluation oracle *for a function $f : \mathbb{R}^n \to \mathbb{R}$ takes as inputs a point $x \in \mathbb{Q}^n$, a positive definite covariance matrix Σ, and an error parameter γ. The oracle first chooses a random point $y \leftarrow \mathcal{N}(x, \Sigma)$, and returns a value r such that $|f(y) - r| < \gamma$.*

The Gaussian sampling in Definition 7 can be equivalently changed to choosing a random point in a ball of desired (small) radius, since any Gaussian distribution can be approximated to arbitrary precision (in the total variation sense) as the convolution of itself with a small enough ball, and vice versa.

Because inputs and outputs to the oracle must be expressible in a polynomial number of digits, we consider "well-guaranteed" star-convex functions (in analogy with Lovász [21]), where the radius bound R and function bound B below should be interpreted as huge numbers (with polynomial numbers of digits), such as 10^{100} and 10^{1000} respectively. Numerical accuracy issues in our analysis are analogous to those for the standard ellipsoid algorithm and we do not discuss them further.

Definition 8. *A weak sampling evaluation oracle for a function f is called* well-guaranteed *if the oracle comes with two bounds, R and B, such that 1) the global minimum of f is within distance R of the origin, and 2) within ℓ_2 distance $10nR$ of the origin, $|f(x)| \leq B$.*

Such sampling gets around the obstacles of the examples in Propositions 5 and 6 because even if the evaluation oracle is queried at a predictable rational point, the value it returns will represent the function evaluated at an unpredictable, (typically) irrational point nearby.

At this point, our notion of oracle access may *seem* unnatural, in part because it allows access to the function at irrational points that are not computationally expressible. However, there are two natural justifications for this approach, of different flavors. First, the oracle represents

a computational model of a mathematical abstraction (the underlying star-convex function), where even for mathematically pathological functions, we can actually in many cases implement simple code to emulate oracle access in the manner described above. For example, it is easy to implement a weak sampling evaluation oracle for the function of Proposition 6, where, with probability 1, the function equals $f_{i,j}(x,y) = ||(x,y) - (1/\sqrt{2} + i, 1/\sqrt{3}+j)||_2$: return this simple function, and ignore the complexities of $f_{i,j}$ that are infinitely unlikely to arise.

Second, setting oracle implementation issues aside, the model cleanly separates *accessing* a potentially pathological function, from *computing properties* of it, in this case, optimizing it. The more pathological a function is, the more surprising it is that any efficient automated technique can extract structure from it. Thus, in some sense, the unrealistic regime of pathological functions, for which we do not even know how to write code emulating oracle access, yields the most surprising regime for the success of the algorithmic results of this paper. This regime is the most insightful from a theory perspective almost exactly because it is the most unnatural and counterintuitive from a practical perspective.

Having expressed the input to the optimization problem in terms of function values sampled on a Gaussian, we define the output in similar terms.

Definition 9. *The* weak star-convex optimization problem *for a Lebesgue measurable star-convex function f, parameterized by δ, ϵ, F, is as follows. Given a well-guaranteed weak sampling evaluation oracle for f, return with probability at least $1 - F$ a Gaussian $\mathcal{G}$ such that $Pr[f(x) \leq f^* + \epsilon : x \leftarrow \mathcal{G}] \geq 1 - \delta$.*

Providing such a Gaussian $\mathcal{G}$ to the weak sampling evaluation oracle (Definition 7), allows one to easily estimate f^* to within ϵ by taking the minimum of a few samples. See the full version of the paper [14] for a discussion of why x^* itself cannot be estimated for general star-convex functions, and why we must instead return a region that "looks like x^* except with δ probability."

We now formally state our main result.

Theorem 10. *Algorithm 1, by providing cutting planes to the ellipsoid algorithm, optimizes (in the sense of Definition 9) any Lebesgue measurable star-convex function f in time* $\mathrm{poly}(n, 1/\delta, \log \frac{1}{\epsilon}, \log \frac{1}{F}, \log R, \log B)$.

Observe that we require only Lebesgue measurability of our objective function, and we make no further continuity or differentiability assumptions. The measurability is necessary to ensure that probabilities and expectations regarding the function are well-defined, and is essentially the weakest assumption possible for any probabilistic algorithm. The minimality in our assumptions contrasts that of the work by Nesterov and Polyak, which assumes Lipschitz continuity in the second derivative [1].

It is mathematically interesting that the *cardinality* of the set of star-convex functions which we optimize, even in the 2-dimensional case, equals the huge quantity $2^{|\mathbb{R}|}$, while the cardinality of the entire set of continuous functions—which is NP-hard to optimize, and strictly contains most standard optimization settings—is only $|\mathbb{R}|$.

III. Computing Cuts for Star-Convex Functions

The general goal of this section is to explain how to compute a single cut, in the context of the ellipsoid algorithm. Namely, given (weak sampling, in the sense of Definition 7) access to a star-convex function f, and a bounding ellipsoid in which we know the global optimum lies, we present an algorithm (Algorithm 1) that will either 1) return a cut passing close to the center of the ellipsoid and containing the global optimum, or 2) directly return an answer to the overall optimization problem. For space reasons, we do not rederive the standard ellipsoid algorithm analysis (see the full version of the paper [14]). To summarize the ellipsoid method: starting with the huge ball of radius R about the origin, it repeatedly reduces the volume (by ratio $1-\Omega(1/n)$ per iteration) of the "feasible ellipsoid", in which the global optimum is known to lie, via cuts produced by Algorithm 1; it stops within a polynomial number of iterations either when the ellipsoid has exponentially small diameter, or when Algorithm 1 declares that the optimum has already been found.

Designing Algorithm 1 to construct valid cuts requires surmounting three major obstacles, introduced in Section I-E. **First**, the star-convex function f might be discontinuous, without any gradients or even subgradients defined at most points; we instead work with a proxy for f, the "blurred logarithm", which is continuous and differentiable, and further, we may efficiently estimate its value and derivatives (see Definition 11 and Proposition 12). **Second**, this solution, however, introduces a new problem: while the original star-convex f always decreases when moving towards the star center, the blurred logarithm of f may not, and its gradients may point in misleading directions (see Figure 2 for an illustration). **Third**, the ellipsoid algorithm produces a sequence of feasible ellipsoids of exponentially decreasing volume, and for *convex* functions this additionally guarantees exponentially convergence of both the feasible region's diameter and overall optimization accuracy (see Theorem 5.2.1 of [20]); however, for *star-convex* functions, neither of these guarantees hold, and successive iterations of the ellipsoid method might repeatedly cut certain dimensions while neglecting others, and fail to converge to the optimum. In order to solve this, we introduce a new cutting plane approach that produces cuts orthogonal to any dimensions of the ellipsoid that are already too thin.

To implement the intuition of the first point above, we define $L_z(x)$, a truncated and translated logarithm of the

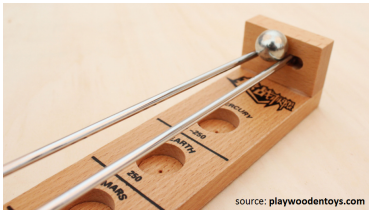


Figure 2: The classic "Shoot the Moon" game, with a steel ball resting on two diverging rails sloping upwards to the left. One wins the game by carefully adjusting the (horizontal) angle between the two rails, counterintuitively getting the ball to roll "uphill", as far left as possible. In the context of this paper, the landscape of the rails is star-convex, sloping down to the right, towards a hypothetical star center at the convergence point of the rails; however, the large steel ball effectively "blurs" the landscape f, changing it so that from the ball's local perspective, left is the downhill direction, and the ball may roll far away from the star center (winning the game, but losing in the optimization context!).

star-convex function f, which maps the potentially exponentially large range of f to the (polynomial-sized) range $[\log \epsilon', \log 2B]$, where ϵ' is defined below (Definition 22), and is slightly smaller than our function accuracy bound ϵ. In the below definition, z intuitively represents our estimate of the global optimum function value, and will record, essentially, the smallest function evaluation seen so far (see Step 2 of Algorithm 1).

Definition 11. *Given an objective function f with bound $|f(x)| \leq B$ when $||x||_2 \leq 10nR$ and an offset value $z \geq f^*$, we define the* truncated logarithmic version of f *to be*

$$L_z(x) = \begin{cases} \log \epsilon' & f(x) - z \leq \epsilon' \\ \log 2B & f(x) - z \geq 2B \\ \log(f(x) - z) & \textit{otherwise} \end{cases}$$

While mapping to a small range, $L_z(x)$ nonetheless gives us a precise view of small changes in the function as we converge to the optimum. The next result shows that, if we "blur" $L_z(x)$ by drawing x from a Gaussian distribution, then not only can we efficiently estimate the expected value of the "blurred logarithm of f", we can also estimate the derivatives of this expectation with respect to changing either the mean or the variance of the Gaussian.

For an arbitrary bounded (measurable) function h, the derivative of its expected value over a Gaussian of width σ with respect to either 1) moving the mean of the Gaussian or 2) changing its width σ, is bounded by $O(\frac{1}{\sigma})$. Thus we normalize the estimates below in terms of the product of the Gaussian width σ and the derivative, instead of estimating the derivative alone.

Proposition 12. *Let $\mathcal{N}(\mu, \Sigma)$ be a Gaussian with diagonal covariance matrix Σ consisting of elements $\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2$. For an error bound $\kappa > 0$ and a probability of error $\delta > 0$, we can estimate each of the following functions to within error κ with probability at least $1 - \delta$ using $\text{poly}(n, \frac{1}{\kappa}, \log \frac{1}{\delta}, \log 2B/\epsilon')$ samples: 1) the expectation $\mathbb{E}[L_z(x) : x \leftarrow \mathcal{N}(\mu, \Sigma)]$; 2) the (scaled) derivative $\sigma_1 \cdot \frac{\mathrm{d}}{\mathrm{d}\mu_1}\mathbb{E}[L_z(x) : x \leftarrow \mathcal{N}(\mu, \Sigma)]$; and 3) the derivative with respect to scaling $\sigma_1 \cdot \frac{\mathrm{d}}{\mathrm{d}\sigma_1}\mathbb{E}[L_z(x) : x \leftarrow \mathcal{N}(\mu, \Sigma)]$.* A fortiori, *these derivatives exist.*

This proposition allows us to efficiently estimate gradients, which we then aim to use as cut directions. The caveat, as mentioned above however, is that the gradient of the blurred logarithm of f may unfortunately point in the wrong direction, potentially inducing a cut that would catastrophically exclude the star center from the feasible region. See Figure 2 for an illustration of this phenomenon.

A. The Cutting Plane Algorithm

One of the primary concerns for our cutting plane algorithm is to prevent any axis of the feasible ellipsoid from getting too small. Therefore, we establish a small threshold τ (defined to be *exponentially* small in Definition 22 below), where we insist on finding a cut to the ellipsoid that is orthogonal to any ellipsoid axes that are already smaller than τ. The notation introduced in the below definition will let us separately analyze the "thin" and "non-thin" dimensions of the ellipsoid.

Definition 13. *Given an ellipsoid, consider an orthonormal basis parallel to its axes. Each semi-principal axis of the ellipsoid whose length is less than τ, we call a "thin dimension", and the rest are "non-thin dimensions". Given a vector μ, we decompose it into $\mu = \mu_\perp + \mu_\top$ where $\mu_\perp$ is non-zero only in the non-thin dimensions, and $\mu_\top$ is non-zero only in the thin dimensions. Similarly, given the identity matrix I, we decompose it into $I = I_\perp + I_\top$.*

We apply a scaling to the *non-thin* dimensions, scaling the non-thin semi-principal axes of the ellipsoid to unit vectors (making the ellipsoid a unit ball in the non-thin dimensions). We keep the thin dimensions as they are.

We now present the cutting plane algorithm, Algorithm 1, along with Proposition 14, our main result describing the structural properties needed for the ellipsoid algorithm, yielding Theorem 10. We make use of constants defined in Definition 22 that may be interpreted as follows: k is a polynomial number of mesh points; η is the mesh spacing; τ' is the minimum size of $\sigma_\top$, a Gaussian width in the thin dimensions that is somewhat larger than τ, the size of the ellipsoid in the thin dimensions; $\sigma'_\perp$ is a Gaussian width in the $\perp$ (non-thin) dimensions, of inverse polynomial size; $\sigma_\perp$ is polynomially smaller than $\sigma'_\perp$, and s is a polynomial quantity. S is a polynomial

Algorithm 1 (Single cut with locked dimensions)
Take an orthonormal basis for the ellipsoid, as in Definition 13. We apply an affine transformation so that 1) the ellipsoid is centered at the origin, and 2) the ellipsoid, when restricted to the $\perp$ dimensions, is the unit ball.

Input: An ellipsoid containing the star center, under an affine transformation as above.
Output: *Either* 1) A cut direction $\mathbf{d}_\perp$ *or* 2) A Gaussian $\mathcal{G}$.

1) For each $i \in [0, k]$
   1a. Evaluate f at S samples from the Gaussian $\mathcal{G}_i$ of width $\tau'\eta^i$ in the $\top$ dimensions and width $\sigma'_\perp$ in the $\perp$ dimensions (that is, $\mathcal{G}_i = \mathcal{N}(\mathbf{0}, \sigma'^2_\perp I_\perp + \tau'^2\eta^{2i} I_\top)$).
   1b. If at least $1 - \frac{31\delta}{32}$ fraction of the evaluations are within ϵ' of the minimum evaluation (at this iteration i), then **Return** $\mathcal{G}_i$ and **Halt**.
2) Otherwise, let z be the minimum of all samples in Step 1.
3) Repeatedly sample the following, estimating g_z to within $\pm\frac{\delta}{32}$ each time
   $g_z(\mu'_\perp, \sigma_\top) : \mu'_\perp \leftarrow \mathcal{N}(\mathbf{0}, (\sigma'^2_\perp - \sigma^2_\perp) I_\perp)$
   and $(\sigma_\top = e^X; X \leftarrow \mathrm{Unif}[\log \tau', \log R/s])$
   3a. Accept the first pair $(\mu'_\perp, \sigma_\top)$ such that $g_z(\mu'_\perp, \sigma_\top) > \frac{7}{32}\delta$.
4) **Return** the gradient $\mathbf{d}_\perp = \nabla_\perp[\mathbb{E}[L_z(x) : x \leftarrow \mathcal{N}(\mu'_\perp, \sigma^2_\perp I_\perp + \sigma^2_\top I_\top)]]$ (the derivative as $\mu'_\perp$ changes, computed via Proposition 12).

number of samples defined in the proof of Proposition 14 (in the full version of this paper [14]).

The crucial function g_z used in the algorithm is defined as the difference $\mathbf{P} - (\mathbf{C} + \mathbf{D})$, where $\mathbf{P}$ is the expression on the right hand side of Proposition 16, and $\mathbf{C}$ and $\mathbf{D}$ are the last two terms on the right hand side of Lemma 15, motivated and discussed below.

Proposition 14 (Correctness of Algorithm 1)**.** *With negligible probability of failure, Algorithm 1 either 1) returns a Gaussian region $\mathcal{G}$ such that*

$$\mathbb{P}[f(x) \le f^* + \epsilon : x \leftarrow \mathcal{G}] \ge 1 - \delta$$

or 2) returns a direction $\mathbf{d}_\perp$, restricted to the $\perp$ dimensions, such that when normalized to a unit vector $\hat{\mathbf{d}}_\perp$, the cut $\{x : x \cdot \hat{\mathbf{d}}_\perp \le \frac{1}{3n}\}$ contains the global minimum.

We note that a special case of Algorithm 1 is when there are no $\top$ (very thin, thinner than τ) dimensions. This applies, for example, at the beginning of the optimization.

B. Proof Sketches

The rest of this section outlines the key ideas in the proof of Proposition 14.

The overall algorithm will hope to return a valid cut by returning the gradient $\nabla_\perp[\mathbb{E}[L_z(x) : x \leftarrow \mathcal{G}]]$ of (the blurred logarithm of) f with respect to translation in the $\perp$ dimensions—which is in some sense the most natural way to use the tools we have set up: blurring the logarithm of f by a Gaussian $\mathcal{G}$ makes f differentiable; and we seek a gradient that is restricted to the non-thin (i.e., $\perp$) dimensions because we seek a cut that will "make progress" in the dimensions that are not already thin. Recall that we can efficiently estimate such gradients via Proposition 12. The crucial difficulty, however, is that using the $\perp$ gradient may not yield a valid cut, pointing in a direction that may accidentally cut the star center from the feasible region. See Figure 2 for an illustration of a case where this gradient points in the opposite direction from the intended signal, and "the ball rolls uphill." Therefore, our algorithmic task is to search for a pair z and $\mathcal{G}$ such that the gradient $\nabla_\perp[\mathbb{E}[L_z(x) : x \leftarrow \mathcal{G}]]$ has a (detectably) positive component in the direction away from the star center. To analyze this component and bound it away from zero, we use the fundamental decomposition illustrated in Figure 3, where the derivative of (the logarithm of) the star-convex function f with respect to scaling each point in the Gaussian $\mathcal{G}$ towards the star center is expressed as the sum of four derivative terms, where the first term is exactly the component of the $\perp$ gradient in the direction away from the star center, and the remaining three derivatives are confounding terms that we must measure and bound.

We refer to this four-way decomposition as the equation "$\mathbf{L} = \mathbf{A} + \mathbf{B} + \mathbf{C} + \mathbf{D}$", where, the aim is to show that $\mathbf{A} > 0$; namely, $\mathbf{A} = \mathbf{L} - (\mathbf{B} + \mathbf{C} + \mathbf{D}) > 0$. Intuitively, $\mathbf{L}$ is large, as it describes the fact that, because of star convexity, "moving towards the star center strongly decreases the function value". Essentially, we show that $\mathbf{L}$ will be large in every case *except* when we have already successfully found a Gaussian $\mathcal{G}$ that optimizes the function in the desired sense (of Definition 9). In this case, Algorithm 1 halts and returns such a Gaussian $\mathcal{G}$ in Step 1b; otherwise, $\mathbf{L}$ is large—$\Omega(\delta)$ as we will explain in Section III-D—and it remains to bound the potential negative effects of $\mathbf{B} + \mathbf{C} + \mathbf{D}$.

For the second case, where $\mathbf{L}$ is large, our strategy is to sample (in Step 3 of Algorithm 1) a Gaussian $\mathcal{G}$ for which the confounding terms $\mathbf{B} + \mathbf{C} + \mathbf{D}$ are small in expectation—namely, $O(\delta)$—by leveraging the following properties of each term: 1) $\mathbf{B}$ is always small because of the thinness, τ, of the thin dimensions; 2) $\mathbf{C}$ can be made small via a "Gaussian in Gaussian subsampling" technique, where the value of $\mathbf{C}$ is small, in expectation, with respect to minor random translations in the $\perp$ dimensions; 3) $\mathbf{D}$, the derivative with respect to expanding

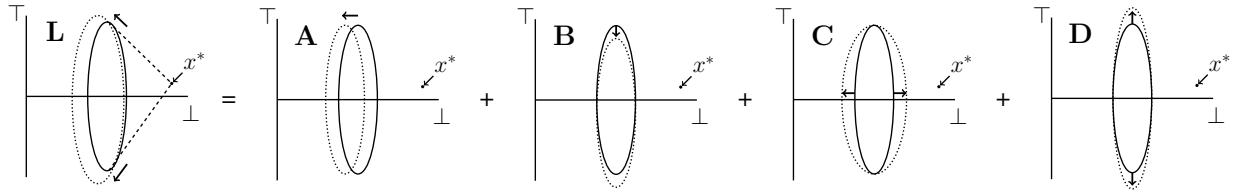


Figure 3: The derivative of (the logarithm of) the star-convex function f with respect to scaling each point in a Gaussian $\mathcal{G}$ away from the star center, pictorially expressed as the sum of four terms, the derivative of the average over $\mathcal{G}$ of f with respect to four different modifications of $\mathcal{G}$: **A**) translating $\mathcal{G}$ away from the star center in the $\perp$ directions; **B**) translating $\mathcal{G}$ away from the star center in the $\top$ directions; **C**) scaling the variance of $\mathcal{G}$ in the $\perp$ directions; and **D**) scaling the variance of $\mathcal{G}$ in the $\top$ directions.

Note that the center $(\mu_\perp, \mu_\top)$ of the Gaussian is very close to (within τ of) x^* in the $\top$ dimensions where the ellipsoid algorithm has essentially already converged, but far from x^* in the $\perp$ dimensions; conversely, the Gaussian $\mathcal{G}$ may have very large variance in the $\top$ dimensions (as depicted in the diagram), since in Step 3 of Algorithm 1, $\sigma_\top$ is drawn from a distribution that extends exponentially far outside the feasible region.

the variance of $\mathcal{G}$ in the $\top$ dimensions, cannot remain large over an exponentially large range of variances, or else the function value would blow up, and thus $\mathbf{D}$ is also small in expectation, like the previous $\mathbf{C}$ term. Hence we randomly sample a Gaussian from a product distribution that simultaneously makes each of the terms $\mathbf{C}$ and $\mathbf{D}$ small in expectation, (in Step 3 of Algorithm 1) and test the resulting Gaussian for suitability by estimating $\mathbf{A} = \mathbf{L} - (\mathbf{B} + \mathbf{C} + \mathbf{D})$, halting when we are confident that $\mathbf{A} \gg 0$.

In the following subsections, we discuss the terms in more detail. For the sake of clarity, our lemmas are stated in the coordinate system where the star center is translated to the origin. The point μ, namely the center of Gaussians we consider, will be a point that is inverse polynomially close to the center of our current ellipsoid.

C. The "$\mathbf{A}$" Term:

The formal statement of the equation $\mathbf{A} = \mathbf{L} - (\mathbf{B} + \mathbf{C} + \mathbf{D})$, when expressed as $\mathbf{L} = \mathbf{A} + \mathbf{B} + \mathbf{C} + \mathbf{D}$ is the following straightforward application of the multivariate chain rule. See Figure 3 for an illustration.

Lemma 15.

$$\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}\alpha}\mathbb{E}\left[L_z(x) : x \leftarrow \mathcal{N}(\alpha\mu, \alpha^2\sigma_\perp^2 I_\perp + \alpha^2\sigma_\top^2 I_\top)\right]\Big|_{\alpha=1} \\
= &\frac{\mathrm{d}}{\mathrm{d}\alpha}\mathbb{E}\left[L_z(x) : x \leftarrow \mathcal{N}(\alpha\mu_\perp + \mu_\top, \sigma_\perp^2 I_\perp + \sigma_\top^2 I_\top)\right]\Big|_{\alpha=1} \\
+ &\frac{\mathrm{d}}{\mathrm{d}\alpha}\mathbb{E}\left[L_z(x) : x \leftarrow \mathcal{N}(\mu_\perp + \alpha\mu_\top, \sigma_\perp^2 I_\perp + \sigma_\top^2 I_\top)\right]\Big|_{\alpha=1} \\
+ &\frac{\mathrm{d}}{\mathrm{d}\alpha}\mathbb{E}\left[L_z(x) : x \leftarrow \mathcal{N}(\mu_\perp + \mu_\top, \alpha^2\sigma_\perp^2 I_\perp + \sigma_\top^2 I_\top)\right]\Big|_{\alpha=1} \\
+ &\frac{\mathrm{d}}{\mathrm{d}\alpha}\mathbb{E}\left[L_z(x) : x \leftarrow \mathcal{N}(\mu_\perp + \mu_\top, \sigma_\perp^2 I_\perp + \alpha^2\sigma_\top^2 I_\top)\right]\Big|_{\alpha=1}
\end{aligned}$$

D. The "$\mathbf{L}$" Term Inequality "$\mathbf{L} \geq \mathbf{P}$":

The following proposition formalizes our lower bound for the "$\mathbf{L}$" term, and is effectively the main structural proposition we leverage of star-convex functions for optimization. For compactness, we later refer to the following inequality as $\mathbf{L} \geq \mathbf{P}$.

Proposition 16. *For a star-convex function f and $z \geq f^*$:*

$$\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}\alpha}\mathbb{E}\left[L_z(x) : x \leftarrow \mathcal{N}(\alpha\mu, \alpha^2\sigma_\perp^2 I_\perp + \alpha^2\sigma_\top^2 I_\top)\right]\Big|_{\alpha=1} \\
&\quad \geq \mathbb{P}[f(x) - z \in (\epsilon', 2B) : x \leftarrow \mathcal{N}(\mu, \sigma_\perp^2 I_\perp + \sigma_\top^2 I_\top)]
\end{aligned}$$

As mentioned before, the intuition behind the lower bound is that $\mathbf{L}$ is a derivative moving every point away from the star center. For a star-convex function f, this derivative must be high unless most of the points in the Gaussian are already close to the global optimum. Recall that the function $L_z(x)$ (from Definition 11) is clamped when $f(x) - z$ is outside the interval $(\epsilon', 2B)$, and thus our bound for the derivative $\mathbf{L}$ depends on—and in fact is equal to—the probability that $f(x) - z$ is "not clamped". Thus $\mathbf{P}$ is the probability that $f(x) - z$ is in the interval $(\epsilon', 2B)$, yielding the convenient bound $\mathbf{L} \geq \mathbf{P}$.

When the probability $\mathbf{P}$ (and hence $\mathbf{L}$) is large, the bounds we outline in the following sections (showing that $\mathbf{B} + \mathbf{C} + \mathbf{D}$ is small) let us conclude $\mathbf{A} = \mathbf{L} - (\mathbf{B} + \mathbf{C} + \mathbf{D}) \gg 0$ and allow us to output a valid cut for our ellipsoid, as described above. However, when the probability $\mathbf{P}$ is small—say, $\mathbf{P} \leq \frac{31}{32}\delta$—we need to argue that in fact, we have already found a Gaussian $\mathcal{G}$ that optimizes the function f in the sense of Definition 9. Consider the distribution of $\{f(x) : x \leftarrow \mathcal{G}\}$ in the case that $\mathbf{P} \leq \frac{31}{32}\delta$, bounding the probability that $f(x) - z \in (\epsilon', 2B)$. The crucial observation is that $f(x) - z$ is extremely unlikely to be larger than $2B$, because B was our global bound for the absolute value of any function evaluation within a giant ball around the origin; and further, $f(x) - z$ is extremely unlikely to be negative, as z was computed to be the minimum of a large number of samples of $f(x)$ across a range of Gaussians that cover $\mathcal{G}$ in a δ-net (described in Step 1a of Algorithm 1), and we are unlikely

to find a better sample now. Thus $\mathbf{P} \leq \frac{31}{32}\delta$ implies that $\mathbb{P}[f(x) \in [z, z+\epsilon'] : x \leftarrow \mathcal{G}] > 1-\delta$.

Hence, when $\mathbf{P}$ is small, the function f is "flat" to within ϵ' on a $1-\delta$ fraction of the points in Gaussian $\mathcal{G}$. The following lemma says that for star-convex functions, such a large, flat Gaussian $\mathcal{G}$ must be close to the global optimum in function value, and we may in fact return $\mathcal{G}$ as the overall solution to the optimization problem.

Lemma 17. *Given a star-convex function f with global optimum at the origin, if for some location μ and number z we have $\mathbb{P}[f(x) \in [z, z+\epsilon'] : x \leftarrow \mathcal{N}(\mu, I)] > 0.95$ then the function value at the global optimum, $f^* = f(0)$, satisfies $f^* \geq z - 6\epsilon' \max\{||\mu||_2, \sqrt{n}\}$.*

The above lemma is useful when $\delta \leq 0.05$; we may easily reduce larger δ to 0.05 and only improve the optimization result. We choose ϵ' in Definition 22 polynomially smaller than ϵ so that, for the Gaussians considered in Algorithm 1, the optimization error promised by Lemma 17 is bounded as $\epsilon' + 6\epsilon' \max\{||\mu||_2, \sqrt{n}\} \leq \epsilon$.

E. The "$\mathbf{B}$" Term, $\mathbf{B} \leq \frac{1}{16}\delta$:

In general, our aim is to bound $\mathbf{B} + \mathbf{C} + \mathbf{D} \leq \frac{1}{4}\delta$ (in expectation). In this section, we discuss the term $\mathbf{B}$, which is the derivative of the blurred logarithm with respect to translation in the exponentially thin ($\top$) directions. The expression for $\mathbf{B}$, that is the left hand side of the inequality in Lemma 18 below, is defined as the α derivative (evaluated at $\alpha = 1$) of an expression where α only appears in the term $\alpha\mu_\top$, and thus this derivative will naturally be proportional to $||\mu_\top||_2$, the distance from the star center to the Gaussian mean in the $\top$ dimensions. By construction, this distance will always be less than the exponentially small quantity τ, implying that the overall derivative defining $\mathbf{B}$ will also be small, as desired. (More technically, $\mathbf{B}$ is bounded by a quantity proportional to the ratio $\frac{||\mu_\top||_2}{\sigma_\top}$, and we choose $\sigma_\top$ to be always $\Omega(\frac{1}{\delta})$ greater than τ in Step 3 of Algorithm 1, by the choice of parameters in Definition 22.) Formally, we have the following lemma.

Lemma 18. *For all $z, \mu_\perp, \sigma_\perp$ and all $||\mu_\top||_2 \leq \tau$ and $\sigma_\top \geq \tau'$ we have:*

$$\left| \frac{\mathrm{d}}{\mathrm{d}\alpha} \mathbb{E}\left[L_z(x) : x \leftarrow \mathcal{N}(\mu_\perp + \alpha\mu_\top, \sigma_\perp^2 I_\perp + \sigma_\top^2 I_\top)\right]\Big|_{\alpha=1} \right| \leq \frac{\delta}{16}$$

F. The "$\mathbf{C}$" Term, $\mathbb{E}_{\mu'_\perp}[\mathbf{C}] \leq \frac{1}{8}\delta$:

The bound for the term $\mathbf{C}$ (and later, $\mathbf{D}$) is less straightforward than that for $\mathbf{B}$, in part because it will only hold in expectation. We first show in Lemma 19 the crude bound $\mathbf{C} \leq \sqrt{2n}\log\left(\frac{2B}{\epsilon'}\right)$, whose right hand side is unfortunately somewhat larger than 1, but which holds in *all* cases, and which we leverage below to yield a tighter bound, in expectation, over a carefully chosen distribution. Lemma 19 may be thought of as a general "reverse isoperimetric inequality", which relies on no structural properties of f or $L_z(x)$ except the ambient dimension, which we denote as $\dim(\perp) \leq n$, and the fact that $L_z(x)$ maps to a bounded interval of size $\log\left(\frac{2B}{\epsilon'}\right)$.

Lemma 19. *For any $z, \mu, \sigma_\perp$ and $\sigma_\top$ we have the following inequality:*

$$\left| \frac{\mathrm{d}}{\mathrm{d}\alpha} \mathbb{E}\left[L_z(x) : x \leftarrow \mathcal{N}(\mu, \alpha^2\sigma_\perp^2 I_\perp + \sigma_\top^2 I_\top)\right]\Big|_{\alpha=1} \right| \leq \sqrt{2\dim(\perp)}\log\left(\frac{2B}{\epsilon'}\right) \leq \sqrt{2n}\log\left(\frac{2B}{\epsilon'}\right)$$

where $\dim(\perp)$ is the number of $\perp$ dimensions.

Recall our goal of showing that, in expectation over some distribution of $\mathcal{G}$, we may bound $\mathbb{E}[\mathbf{C}] \ll \delta$, which is unfortunately rather smaller than the quantity $\sqrt{2n}\log\left(\frac{2B}{\epsilon'}\right)$ shown by Lemma 19.

Our technique is to consider a certain relationship between independent Gaussian random variables: adding samples from two independent Gaussians yields a sample from a new Gaussian whose mean is the total mean of the two Gaussians and whose variance is the total variance of the two Gaussians. Phrased differently, if we draw a Gaussian sample $\mu'_\perp \leftarrow \mathcal{N}(\mu_\perp, (\sigma'^2_\perp - \sigma^2_\perp)I_\perp)$, and then use this random variable $\mu'_\perp$ as the mean of a second Gaussian $x \leftarrow \mathcal{G}_{\mu'_\perp} \triangleq \mathcal{N}(\mu'_\perp + \mu_\top, \sigma^2_\perp I_\perp + \sigma^2_\top I_\top)$, then a sample x drawn from this second Gaussian will itself have a Gaussian distribution $x \sim \mathcal{G}' \triangleq \mathcal{N}(\mu_\perp + \mu_\top, \sigma'^2_\perp I_\perp + \sigma^2_\top I_\top)$ whose variance is the total of the previous two variances.

Crucially, we ask, for the above distribution of $\mu'_\perp$, how does the value of $\mathbf{C}$ evaluated on Gaussian $\mathcal{G}'$ compare to the expected value of $\mathbf{C}$ on Gaussian $\mathcal{G}_{\mu'_\perp}$? By the above argument, the distributions of points x on which the derivative $\frac{\mathrm{d}}{\mathrm{d}\alpha}\mathbb{E}[L_z(x)]$ is computed are *identical* in the two cases. The below lemma states that, in fact, the value of $\mathbf{C}$ on Gaussian $\mathcal{G}'$ is exactly equal to the expectation of $\mathbf{C}$ on Gaussian $\mathcal{G}_{\mu'_\perp}$, times a multiplicative factor of $\left(\frac{\sigma_\perp}{\sigma'_\perp}\right)^2$; we conveniently choose these parameters in Definition 22 to make the resulting bound small enough when multiplied by the bound $\mathbf{C}(\mathcal{G}') \leq \sqrt{2n}\log\left(\frac{2B}{\epsilon'}\right)$ from Lemma 19.[1]

Explicitly, defining for the moment the notation $\mathbf{C}(\mu_\perp, \sigma_\perp)$ to denote the $\mathbf{C}$ term

$$\frac{\mathrm{d}}{\mathrm{d}\alpha} \mathbb{E}\left[L_z(x) : x \leftarrow \mathcal{N}(\mu_\perp + \mu_\top, \alpha^2\sigma_\perp^2 I_\perp + \sigma_\top^2 I_\top)\right]\Big|_{\alpha=1}$$

with the remaining parameters $\mu_\top$, $\sigma_\top$, and z fixed, we have the following lemma.

[1] It is important to be able to choose parameters so that the ratio $\frac{\sigma_\perp}{\sigma'_\perp}$ is inverse polynomial instead of inverse exponential, as the number of samples required by Proposition 12 to accurately estimate gradients in the $\perp$ dimensions is inversely proportional to the Gaussian variance $\sigma_\perp$.

Lemma 20. *For all $\mu_\perp$, and $\sigma_\perp < \sigma'_\perp$, we have:*

$$\mathbb{E}\left[\mathbf{C}(\mu'_\perp, \sigma_\perp) : \mu'_\perp \leftarrow \mathcal{N}(\mu_\perp, (\sigma'^2_\perp - \sigma^2_\perp) I_\perp)\right] = \left(\frac{\sigma_\perp}{\sigma'_\perp}\right)^2 \mathbf{C}(\mu_\perp, \sigma'_\perp) \leq \frac{\delta}{8}$$

This sampling, $\mu'_\perp \leftarrow \mathcal{N}(\mu_\perp, (\sigma'^2_\perp - \sigma^2_\perp) I_\perp)$, is exactly what occurs in Step 3 of Algorithm 1.

The explanation for the $\left(\frac{\sigma_\perp}{\sigma'_\perp}\right)^2$ term is straightforward: $\mathbf{C}(\mu_\perp, \sigma_\perp)$ is defined as the α derivative (evaluated at $\alpha = 1$) of an expression where α only appears in the term $\alpha^2 \sigma^2_\perp$, and thus this derivative will naturally be proportional to $\sigma^2_\perp$, and changing from $\sigma'_\perp$ on the right hand side to $\sigma_\perp$ on the left hand side induces an overall scaling of $\left(\frac{\sigma_\perp}{\sigma'_\perp}\right)^2$, as claimed.

G. The "$\mathbf{D}$" Term, $\mathbb{E}_{\sigma_\top}[\mathbf{D}] \leq \frac{1}{16}\delta$:

In the last section, we gave a distribution over Gaussians which, on average, takes a small value for the term $\mathbf{C}$. Analogously, we show in the following lemma that the term $\mathbf{D}$ on average takes a small value when $\sigma_\top$ is sampled from an appropriate loguniform distribution. It is important to note that the two distributions are on different parameters ($\mu'_\perp$ for the term $\mathbf{C}$ and $\sigma_\top$ for the term $\mathbf{D}$), and the bounds on the expectations individually hold for any value taken by the other parameter. Therefore, we are able to take the product distribution on the two parameters and claim that the expectation of $\mathbf{C} + \mathbf{D}$ is small.

As above, we temporarily define a convenient notation for the $\mathbf{D}$ term, letting $\mathbf{D}(\sigma_\top)$ equal

$$\frac{\mathrm{d}}{\mathrm{d}\alpha}\mathbb{E}\left[L_z(x) : x \leftarrow \mathcal{N}(\mu_\perp + \mu_\top, \sigma^2_\perp I_\perp + \alpha^2 \sigma^2_\top I_\top)\right]\Big|_{\alpha=1}$$

Lemma 21. *For all $z, \mu, \sigma_\perp$,*

$$\mathbb{E}\left[\mathbf{D}(\sigma_\top) : \left(\sigma_\top = e^X; X \leftarrow \mathrm{Unif}[\log \tau', \log R/s]\right)\right] \leq \frac{\log 2B - \log \epsilon'}{\log(R/s) - \log \tau'} = \frac{\delta}{16}$$

The equality is by the choice of τ' in Definition 22.

The proof of this lemma is surprisingly straightforward, following directly from the fundamental theorem of calculus and the boundedness of $L_z(x)$. Explicitly, the α derivative in the definition of $\mathbf{D}(\sigma_\top)$, when reweighted by the pdf of the distribution of $\sigma_\top$, becomes a derivative with respect to $\sigma_\top$; by the fundamental theorem of calculus, the integral (expectation) over $\sigma_\top$ and the derivative with respect to $\sigma_\top$ "cancel". What remains is proportional to the change in value of the blurred logarithm L_z between $\sigma_\top = \tau'$ and $\sigma_\top = R/s$; since L_z maps to the interval $(\log \epsilon', \log 2B)$, the bound of the lemma is thus proportional to $\log 2B - \log \epsilon'$, as claimed.

The denominator in the lemma, $\log(R/s) - \log \tau'$ is a normalizing constant from the probability distribution of $\sigma_\top$, and the usefulness of the lemma results from our ability to choose τ' exponentially small—in Definition 22—so that the overall ratio reaches our target of $\frac{\delta}{16}$.

H. Parameters

The parameters we have been using above are:

Definition 22. *Let*

$$s = \sqrt{n}\left(1 + \sqrt{\tfrac{4}{3}}\sqrt{n + \tfrac{1}{\delta} + \log \tfrac{BR}{\epsilon F}}\right);\ \ \eta = e^{\frac{\delta^2}{8n}}$$

$$\sigma'_\perp = \tfrac{1}{3ns};\ \sigma_\perp = \sigma'_\perp \sqrt{\tfrac{\delta/8}{\log 2B - \log \epsilon'}\sqrt{\tfrac{1}{2n}}};\ \epsilon' = \epsilon\left(1 + \tfrac{12}{\sigma'_\perp}\right)^{-1}$$

$$\tau' = \tfrac{R}{s}\left(\tfrac{2B}{\epsilon'}\right)^{-\frac{16}{\delta}};\ \tau = \tau' \tfrac{\delta}{16} \cdot \tfrac{1}{\log\left(\frac{2B}{\epsilon'}\right)} \tfrac{\sqrt{\pi}}{2\sqrt{2}};\ k = \tfrac{\log\left(\frac{2B}{\epsilon'}\right)^{\frac{16}{\delta}}}{\log \eta}$$

I. The Algorithm, in Whole

With the tools from the previous subsections in hand, we now walk through Algorithm 1, outlining its proof of correctness.

Proposition 14 (Correctness of Algorithm 1)**.** *With negligible probability of failure, Algorithm 1 either 1) returns a Gaussian region $\mathcal{G}$ such that*

$$\mathbb{P}[f(x) \leq f^* + \epsilon : x \leftarrow \mathcal{G}] \geq 1 - \delta$$

or 2) returns a direction $\mathbf{d}_\perp$, restricted to the $\perp$ dimensions, such that when normalized to a unit vector $\hat{\mathbf{d}}_\perp$, the cut $\{x : x \cdot \hat{\mathbf{d}}_\perp \leq \frac{1}{3n}\}$ contains the global minimum.

Recall that we have been working since Section III-B with the decomposition $\mathbf{A} = \mathbf{L} - (\mathbf{B} + \mathbf{C} + \mathbf{D})$, an identity expressing $\mathbf{L}$ as the sum of four derivatives $\mathbf{A} + \mathbf{B} + \mathbf{C} + \mathbf{D}$ via the multivariate chain rule (see Lemma 15), as illustrated in Figure 3; we add to this the inequality $\mathbf{L} \geq \mathbf{P}$ of Proposition 16, relating the derivative $\mathbf{L}$ to a probability $\mathbf{P}$. Our goal is to bound $\mathbf{A}$ positively away from 0, since the sign of the term $\mathbf{A}$ records whether the cut we make includes the star center. In the preceding subsections we have assembled algorithmic and analytical tools to help us bound each term of this decomposition.

Algorithm 1 starts in Step 1 by sampling within a fine mesh of different Gaussians, and checking if the function values within any of them are "flat" enough that Lemma 17 lets us conclude that we have already found an optimum region of the function. Comparing the probabilities computed in Lemma 17 and Proposition 16 (under an affine transformation so the Gaussian regions being sampled correspond), we see that Lemma 17 measures the probability that $f(x) \in [z, z + \epsilon']$, while Proposition 16 measures the probability that $f(x) \in (z + \epsilon', z + 2B)$. Collectively, these two intervals disjointly cover the entire interval $[z, z + 2B)$, which, as explained in Section III-D, essentially contains the entire range of function evaluations (with high probability). Thus the probabilities computed by Lemma 17 and Proposition 16 are essentially complementary, yielding complementary paths to victory: either the probability computed by Lemma 17 is small enough for *some* Gaussian examined in Step 1 that we can terminate the optimization by outputting that Gaussian, or the complementary probability $\mathbf{P}$ is large for *all* Gaussians, yielding, by the bound $\mathbf{L} \geq \mathbf{P}$ of

Proposition 16, that we always have a satisfactorily large bound for the elusive expression $\mathbf{L}$.

In the case that Algorithm 1 does not halt in Step 1, we record in Step 2 as z the minimum function value observed so far; the fine mesh of Gaussians in Step 1a is set up so that it is a δ-net, essentially "covering" all distributions sampled in Step 3, so that z can be thought of as a good approximate lower bound for any function evaluation seen in Step 3.

Next, in Step 3, we repeatedly sample new Gaussians on which we evaluate g_z, defined as $\mathbf{P} - (\mathbf{C} + \mathbf{D})$, and halt when the evaluation is sufficiently positive. This expression $\mathbf{P}-(\mathbf{C}+\mathbf{D})$ is a proxy for $\mathbf{P}-(\mathbf{B}+\mathbf{C}+\mathbf{D})$, where we omit $\mathbf{B}$ because of the convenient bound of Lemma 18, that $\mathbf{B} \leq \frac{\delta}{16}$, where δ is our desired overall error bound.

The crucial technique in Step 3 is sampling $\mu'_\perp$ and $\sigma_\top$ from carefully chosen distributions so that the expected value of each of $\mathbf{C}$ and $\mathbf{D}$ is a small multiple of δ, by Lemmas 20 and 21. Explicitly, Lemma 20 says that the expected value of the $\mathbf{C}(\mu'_\perp, \sigma_\perp)$ term is bounded by $\frac{\delta}{8}$, when $\mu'_\perp$ is drawn from a Gaussian of variance $\sigma'^2_\perp - \sigma^2_\perp$. Lemma 21 says that the expected value of the $\mathbf{D}(\sigma_\top)$ term is bounded by $\frac{\delta}{16}$ when $\sigma_\top$ is drawn from the loguniform distribution of the algorithm $(\sigma_\top = e^X; X \leftarrow \mathrm{Unif}[\log \tau', \log R/s])$.

As for the $\mathbf{P}$ term, it is large on each of the Gaussians examined in Step 1a, of variance $\sigma'^2_\perp$ in the $\perp$ dimensions; however, in Step 3 we instead subsample smaller Gaussians of variance $\sigma^2_\perp$, with centers chosen from a Gaussian of variance $\sigma'^2_\perp - \sigma^2_\perp$. Thus the bounds on $\mathbf{P}$ from Step 1a do not hold for individual Gaussians in Step 3, but instead hold on average for each *sample* in Step 3. Namely, over the choice of Gaussians considered in Step 3, the expected value of $\mathbf{P}$ is $\Omega(\delta)$.

Adding up all three bounds yields that, over the random Gaussian chosen in Step 3, the expected value of $\mathbf{P}-(\mathbf{C}+\mathbf{D})$ will be $\geq \frac{1}{2}\delta$. Because of reasonable bounds on the range of each of $\mathbf{P}$, $\mathbf{C}$, and $\mathbf{D}$, Markov's inequality implies that, with at least inverse polynomial probability, a given random Gaussian will have $\mathbf{P} - (\mathbf{C} + \mathbf{D}) \geq \frac{1}{4}\delta$. Subtracting the bound $\mathbf{B} \leq \frac{1}{16}\delta$, and accounting for inverse polynomial sampling errors (via Proposition 12) still leaves $\mathbf{P}-(\mathbf{B}+\mathbf{C}+\mathbf{D}) = \Omega(\delta)$, implying, since $\mathbf{L} \geq \mathbf{P}$ (Proposition 16), that $\mathbf{A} = \mathbf{L}-(\mathbf{B}+\mathbf{C}+\mathbf{D}) = \Omega(\delta) \gg 0$, and thus the cut returned in Step 4 of Algorithm 1 will contain the star center, as desired.

ACKNOWLEDGMENTS

This work was supported in part by a Sloan Research Fellowship, and by NSF Grant IIS-1562657. P.V. is grateful to Aaron Sidford and Santosh Vempala for enlightening discussions about cutting plane techniques while at the 2015 UW/MSR Summer Institute at Alderbrook.

REFERENCES

[1] Y. Nesterov and B. Polyak, "Cubic regularization of Newton method and its global performance," *Math. Program.*, vol. 108, no. 1, pp. 177–205, 2006.

[2] E. Chlamtac and M. Tulsiani, *Handbook on Semidefinite, Conic and Polynomial Optimization*. Boston, MA: Springer US, 2012, ch. 6. Convex Relaxations and Integrality Gaps, pp. 139–169.

[3] A. Madry, "Navigating central path with electrical flows: From flows to matchings, and back," in *Proc. FOCS'13*, pp. 253–262.

[4] J. A. Kelner, Y. T. Lee, L. Orecchia, and A. Sidford, "An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations," in *Proc. SODA'14*, pp. 217–226.

[5] Y. T. Lee and A. Sidford, "Path finding methods for linear programming: Solving linear programs in $\tilde{O}$(vrank) iterations and faster algorithms for maximum flow," in *Proc. FOCS'14*, pp. 424–433.

[6] P. Christiano, J. A. Kelner, A. Madry, D. A. Spielman, and S.-H. Teng, "Electrical flows, Laplacian systems, and faster approximation of maximum flow in undirected graphs," in *Proc. STOC'11*, pp. 273–282.

[7] J. Sherman, "Nearly maximum flows in nearly linear time," in *Proc. FOCS'13*, pp. 263–269.

[8] N. K. Vishnoi, "Lx = b," *Found. Trends Theoretical Computer Science*, vol. 8, no. 1-2, pp. 1–141, 2012.

[9] D. A. Spielman and S.-H. Teng, "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems," in *Proc. STOC'04*, pp. 81–90.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. NIPS'12*, pp. 1097–1105.

[11] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[12] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $o(1/k^2)$," *Soviet Mathematics Doklady*, vol. 27, no. 2, pp. 372–376, 1983.

[13] E. Hazan, K. Levy, and S. Shalev-Shwartz, "Beyond convexity: Stochastic quasi-convex optimization," in *Proc. NIPS'15*, pp. 1594–1602.

[14] J. C. H. Lee and P. Valiant, "Optimizing star-convex functions," *CoRR*, vol. abs/1511.04466v2, 2016. [Online]. Available: http://arxiv.org/abs/1511.04466

[15] A. Nemirovskii and D. Yudin, *Problem Complexity and Method Efficiency in Optimization*. Wiley, 1983.

[16] K. Chandrasekaran, D. Dadush, and S. Vempala, "Thin partitions: Isoperimetric inequalities and a sampling algorithm for star shaped bodies," in *Proc. SODA'10*, pp. 1630–1645.

[17] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer New York, 2013.

[18] P. Valiant, "Evolvability of real functions," *ACM Trans. Comput. Theory*, vol. 6, no. 3, pp. 12:1–12:19, 2014.

[19] J. C. Duchi, P. L. Bartlett, and M. J. Wainwright, "Randomized smoothing for stochastic optimization," *SIAM J. Optimiz.*, vol. 22, no. 2, pp. 674–701, 2012.

[20] A. N. Aharon Ben-Tal, *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*, ser. MPS-SIAM Series on Optimization. Society for Industrial Mathematics, 2001.

[21] L. Lovász, *An algorithmic theory of numbers, graphs and convexity*. SIAM, 1987.