Greedy Adversarial Equilibrium: An Efficient Alternative to Nonconvex-Nonconcave Min-Max Optimization

Oren Mangoubi
Worcester Polytechnic Institute
USA

Nisheeth K. Vishnoi
Yale University
USA

ABSTRACT

Min-max optimization of an objective function $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is an important model for robustness in an adversarial setting, with applications to many areas including optimization, economics, and deep learning. In many applications f may be nonconvex-nonconcave, and finding a global min-max point may be computationally intractable. There is a long line of work that seeks computationally tractable algorithms for alternatives to the min-max optimization model. However, many of the alternative models have solution points which are only guaranteed to exist under strong assumptions on f, such as convexity, monotonicity, or special properties of the starting point. We propose an optimization model, the ε-greedy adversarial equilibrium, and show that it can serve as a computationally tractable alternative to the min-max optimization model. Roughly, we say that a point $(x^\star, y^\star)$ is an ε-greedy adversarial equilibrium if $y^\star$ is an ε-approximate local maximum for $f(x^\star, \cdot)$, and $x^\star$ is an ε-approximate local minimum for a "greedy approximation" to the function $\max_z f(x, z)$ which can be efficiently estimated using second-order optimization algorithms. We prove the existence of such a point for any smooth function which is bounded and has Lipschitz Hessian. To prove existence, we introduce an algorithm that converges from any starting point to an ε-greedy adversarial equilibrium in a number of evaluations of the function f, the max-player's gradient $\nabla_y f(x, y)$, and its Hessian $\nabla^2_y f(x, y)$, that is polynomial in the dimension d, $1/\varepsilon$, and the bounds on f and its Lipschitz constant.

CCS CONCEPTS

• **Theory of computation → Nonconvex optimization**; • **Computing methodologies → Machine learning algorithms**.

KEYWORDS

min-max optimization, nonconvex optimization

ACM Reference Format:
Oren Mangoubi and Nisheeth K. Vishnoi. 2021. Greedy Adversarial Equilibrium: An Efficient Alternative to Nonconvex-Nonconcave Min-Max Optimization. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC '21), June 21–25, 2021, Virtual, Italy*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3406325.3451097

STOC '21, June 21–25, 2021, Virtual, Italy

ACM ISBN 978-1-4503-8053-9/21/06…$15.00
https://doi.org/10.1145/3406325.3451097

1 INTRODUCTION

Min-max optimization of functions $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, where $f(x, y)$ may be nonconvex and nonconcave in both x and y, is an important model for robustness which arises in optimization and game theory [35] with recent applications in machine learning such as generative adversarial networks (GANs) [16] and robust training [25]. Specifically, in a min-max problem, one wishes to find a global min-max point $(x^\star, y^\star)$ that is a solution to the following optimization problem:

$$\min_{x\in\mathbb{R}^d} \max_{y\in\mathbb{R}^d} f(x, y).$$

In other words,

$$f(x^\star, y^\star) = \max_{y\in\mathbb{R}^d} f(x^\star, y) \text{ and}$$

$$\max_{y\in\mathbb{R}^d} f(x^\star, y) = \min_{x\in\mathbb{R}^d} \max_{y\in\mathbb{R}^d} f(x, y).$$

We consider the setting where f is a C^2-smooth nonconvex- nonconcave function that is uniformly bounded by some $b > 0$ with L-Lipschitz Hessian for some $L > 0$, and we are given access to oracles for f, its gradient, and its Hessian. The setting where f is a bounded function with unconstrained domain arises in many machine learning applications, including generative adversarial networks (GANs).[1]

In the unconstrained setting, a global min-max point may not exist, even when f is bounded above and below. However, in the setting where f is bounded, the extreme value theorem guarantees that one can find a point where each player's objective is very close to its min-max optimal value. Namely, for every $\varepsilon > 0$ one can always find a point $(x^\star, y^\star)$ such that

$$f(x^\star, y^\star) \geq \max_{y\in\mathbb{R}^d} f(x^\star, y) - \varepsilon \text{ and}$$

$$\max_{y\in\mathbb{R}^d} f(x^\star, y) \leq \min_{x\in\mathbb{R}^d} \max_{y\in\mathbb{R}^d} f(x, y) + \varepsilon$$

However, even in the special case of minimization, finding a point whose function value is within a fixed $\varepsilon > 0$ of the minimum value is hard. For instance, this problem is NP-hard in settings when the objective function is given by a depth-2 neural network with mean-squared error loss [27]; see also [6]. The problem of minimizing a function or determining if it can achieve a minimum value of 0 remains hard when f is uniformly b-bounded with L-Lipschitz Hessian when one is given access to oracles for the gradient and Hessian of f, and requires a number of oracle queries which is

[1] In GANs, the objective function value is bounded both above and below if one uses a mean-squared-error loss [28]. For the cross-entropy loss [16], the objective function is uniformly bounded above by 0 (but need not be bounded below).

exponential in d (see the arXiv version of our paper [26] for a detailed discussion). Consequently, there has been interest in finding computationally tractable alternatives to min-max optimization.

One popular alternative to min-max optimization is to consider a model where the min- and max- players are only allowed to make small "local" updates, rather than requiring each player to solve a global optimization problem [2, 9, 10, 18]. A stationary point for such a model, sometimes referred to as a local min-max point, is a point where the min-player is unable to decrease the loss, and the max-player is unable to increase the loss, if they are restricted to local updates inside a ball of some small radius. More specifically, for $\varepsilon, \delta > 0$, an (ε, δ)-local min-max point $(x^\star, y^\star)$ is a point where $\forall x, y \in \mathbb{R}^d$ such that $\|y - y^\star\| \le \delta$ and $\|x - x^\star\| \le \delta$,

$$f(x^\star, y^\star) \ge f(x^\star, y) - \varepsilon \quad \text{and} \quad f(x^\star, y^\star) \le f(x, y^\star) + \varepsilon.$$

One can also define a notion of (ε, δ)-local minimum and maximum point in a similar manner. In the "local" regime where $\delta < O(\sqrt{\varepsilon})$, if $f \in [-1, 1]$ with $O(1)$-Lipschitz gradient, any point which is an (ε, δ)-local minimum of the function f with respect to the variable (x, y), or an (ε, δ)-local maximum of f with respect to (x, y), will also be a $(\Omega(\varepsilon), O(\delta))$-local min-max point of this function; thus, in this regime the problem of finding local min-max points is equivalent to the problem of finding a local minimum or maximum point. Unfortunately, outside of the regime $\delta < O(\sqrt{\varepsilon})$, local min-max points may not exist even for functions f where the value of $f \in [-1, 1]$ and f is $O(1)$-Lipschitz with $O(1)$-Lipschitz gradient and Hessian.[2] Thus, in the regime $\delta > \Omega(\sqrt{\varepsilon})$ local min-max points are not guaranteed to exist, and, when $\delta < O(\sqrt{\varepsilon})$ finding an (ε, δ)-local min-max point is not as interesting since it reduces to finding a minimum (or maximum) point of f.

1.1 Our Contributions

We depart from prior approaches and make a novel assumption on the adversary (the max-player), namely, that the adversary is computationally bounded. This is motivated from real-world applications where the adversary itself may be an algorithm. Roughly, we show that when the adversary is restricted to computing a greedy approximation to the global maximum $\max_z f(x, z)$, a type of equilibrium –*greedy adversarial equilibrium*– always exists and can be found efficiently for general f in time polynomial in d, b, and L. This is in contrast to previous works which seek local min-max points and make strong assumptions on f, for instance assuming that $f(x, y)$ is concave in y but possibly nonconvex in x [33, 36], that f is sufficiently bilinear [1], or that the gradient of f satisfies a monotonicity property [14, 22].

Our greedy adversarial equilibrium builds on the second-order notion of approximate local minimum introduced by [31].[3] Roughly, a second-order (ε, θ)-approximate local minimum of a function $\psi : \mathbb{R}^d \to \mathbb{R}$ is a point $x^\star$ which satisfies the following second-order conditions

$$\|\nabla \psi(x^\star)\| \le \varepsilon \text{ and } \lambda_{\min}(\nabla^2 \psi(x^\star)) \ge -\theta.$$

[2] For instance, the function $f(x, y) := \sin(x + y)$ has no (ε, δ)-local min-max points when $\frac{1}{100} > \delta > \sqrt{\varepsilon}$.

[3] We often refer to this second-order approximate local minimum by approximate local minimum.

[31] and other recent works [3–5, 8, 11, 19], have shown that one can find an (ε, θ)-approximate local minimum in time roughly poly $\left(\frac{1}{\varepsilon}, \log d, \frac{1}{\theta}, b, L\right)$ gradient evaluations.

In our model, the min-player is empowered to simulate updates of the max-player by computing a tractable second-order approximation to the global max function $\max_z f(x, z)$, which we refer to as the *greedy max function* $g_\varepsilon(x, y)$. Here, the parameter $\varepsilon > 0$ is a measure of approximation. Ideally, we would like a point $(x^\star, y^\star)$ to be a greedy adversarial equilibrium if $y^\star$ is a second-order approximate local maximum for $f(x^\star, \cdot)$, and $x^\star$ is a second-order approximate local minimum for $g_\varepsilon(\cdot, y^\star)$. However, the function $g_\varepsilon(x, y)$ that arises is hard to evaluate and also discontinuous. We overcome these issues in part by "truncating" and "smoothing" g_ε by convolving it with a Gaussian $N(0, \sigma^2 I_d)$ for some $\sigma > 0$ to obtain a smooth approximation $S_{\varepsilon,\sigma}(\cdot)$ to $g_\varepsilon(\cdot, y)$. This allows us to apply the definition of approximate local minimum above to g_ε, and to obtain our definition of (ε, σ)-greedy adversarial equilibrium; see Definition 2.5.

Our main technical result is an algorithm which finds an (ε, σ)-greedy adversarial equilibrium in a number of gradient, Hessian, and function evaluations that is polynomial in $\frac{1}{\varepsilon}, \frac{1}{\sigma}, b, L, d$; see Theorem 3.1. In particular, providing such an algorithm proves the existence of an approximate greedy adversarial equilibrium. Our algorithm requires access to a zeroth-order oracle for the value of f, and to oracles for the gradient $\nabla_y f(x, y)$ and Hessian $\nabla_y^2 f(x, y)$ for the max-player variable y, but *not* to oracles $\nabla_x f(x, y)$ and $\nabla_x^2 f(x, y)$ for the min-player variable x. Note that the polynomial dependence on d in our bounds comes from the fact that we do not assume that the x-player has access to a gradient oracle $\nabla_x f$ or Hessian oracle $\nabla_x^2 f$.

1.2 Discussion of Our Contributions

Computationally Bounded Adversaries. The main conceptual insight in this paper is that we can obtain a model which is an efficient alternative to min-max optimization by placing computational restrictions on the adversary. In comparison to models where each player is restricted to local updates, this can allow the model to be robust to a greater diversity of adversaries from a much larger set of parameters y than just the current value of y namely, all the values of y reachable by the tractable approximation–while still allowing for efficient algorithms for modeling the adversary (in particular, this can allow for more stable training of machine learning algorithms). We note that analogous computationally bounded adversaries can lead to useful models in many other settings including coding theory [17, 23, 30] and cryptography [7, 15].

Results Hold for any Bounded and Lipschitz f. Aside from the bounded and Lipschitz assumptions, Theorem 3.1 does not make any additional assumptions on f. As mentioned earlier, prior results which seek solutions to other alternative models to min-max optimization (such as local min-max points), assume that either $f(x, y)$ is concave [33, 36] in y, or monotone [14, 22], or sufficiently bilinear [1]. Although there are other prior works which do not assume that f is convex-concave or monotone, many of these works instead assume that there exists a stationary point for their algorithm on the function f, and that their algorithm is initialized somewhere in the region of attraction for this stationary point [2, 18, 29, 37].

In contrast, Theorem 3.1 guarantees that our algorithm converges from *any* initial point (x, y).

Extension of Second-Order Local Minimum Definition to Discontinuous Functions. To handle minimization of the discontinuous greedy max function g_ε, when defining our greedy adversarial equilibrium we introduce a second-order notion of approximate local minimum which applies to discontinuous functions. This leads to an algorithm which, in the special case when the objective function depends only on x, reduces to a "derivative-free" minimization algorithm, that is, it does not require access to derivatives of the objective function. The novel techniques and definitions we develop here for minimization of the greedy max function may be of interest to other problems in discontinuous or derivative-free minimization of non-convex objectives (for applications of derivative-free methods to adversarial bandit convex optimization, see for instance [12]).

Greedy Adversarial Equilibria Corresponds to Global Min-Max under Strong Convexity/Concavity. If f is 1-strongly convex- strongly concave, then for $\varepsilon > 0$ and small enough σ, we show that at any (ε, σ)-greedy adversarial equilibrium $(x^\star, y^\star)$ the duality gap satisfies

$$\max_{y\in\mathbb{R}^d} f(x^\star, y) - \min_{x\in\mathbb{R}^d} f(x, y^\star) \le O(\varepsilon^2);$$

see the arXiv version of our paper [26] for a precise statement of this result and its proof.

Applications to GANs. In a subsequent paper, [21] use a related first-order version of our greedy adversarial equilibrium to obtain an algorithm and show that it can enable more stable training of generative adversarial networks (GANs). Roughly speaking, the first-order equilibrium in [21] is a point $(x^\star, y^\star)$ such that $\|\nabla_y f(x^\star, y^\star)\| \le \varepsilon$ and $\|\nabla_x \hat{g}(x^\star, y^\star)\| \le \varepsilon$, where $\hat{g}$ is a first-order approximation to the greedy-max function. This means that, unlike here, in [21] min-min points (points where both players are at a local minimum) are included in the local equilibrium proposed. Including second-order conditions for both the maximizing and minimizing players in our Definition 2.5 allows us to ensure that our definition excludes points which may be (approximate) min-min points. The second-order conditions also end up making the proofs in this paper significantly harder.

Difference between Constrained and Unconstrained Settings. Finally, we note that, in a subsequent work, [10] prove PPAD-hardness results for finding approximate local min-max points in the constrained setting. Their result does not have any implication to our framework as we consider the unconstrained setting (domain is $\mathbb{R}^d \times \mathbb{R}^d$).

1.3 Organization of the Paper

In Section 2, we present the definition of greedy adversarial equilibrium and in Section 3 we state our main result. Section 4 contains a technical overview of our algorithm and proof. For additional discussions about our definition of greedy adversarial equilibrium and connections to previous notions see Section 5.

In Section 6 we give a full description of the algorithm. In Section 7 we present the proof of our main result, and the main lemmas and propositions we use to prove our main result, along with short summaries of their proofs; for the full proofs see the arXiv version of our paper [26].

2 GREEDY ADVERSARIAL EQUILIBRIUM

Preliminaries. In the following, we say that a function is C^2 smooth if its second derivatives are continuous on its domain. $\lambda_{\max}(A)$ denotes the largest eigenvalue of any square matrix A, and $\lambda_{\min}(A)$ is its smallest eigenvalue. $\|\cdot\|$ denotes the Euclidean ℓ_2 norm, and $\|A\|_{\text{op}} = \sup_{v\neq 0} \frac{v^\top A v}{\|v\|^2}$ the operator norm of any square matrix A. We assume[4] that for some $b, L > 0$, $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is b-bounded, i.e., $|f(x, y)| \le b$, and has L-Lipschitz Hessian:

$$\|\nabla^2 f(x, y) - \nabla^2 f(\tilde{x}, \tilde{y})\|_{\text{op}} \le L\sqrt{\|x - \tilde{x}\|^2 + \|y - \tilde{y}\|^2}.$$

We start by considering the special case of minimization. We say that a point $x^\star \in \mathbb{R}^d$ is an *exact* local minimum point of a function $\psi : \mathbb{R}^d \to \mathbb{R}$ if there exists $\delta > 0$ such that

$$\psi(x^\star) \le \psi(x), \quad \forall x \in \mathbb{R}^d \text{ such that } \|x - x^\star\| \le \delta. \tag{1}$$

Unfortunately, even if the objective function $\psi : \mathbb{R}^d \to \mathbb{R}$ is bounded and Lipschitz, it is not always possible to find an exact local minimum for ψ in poly(d) gradient evaluations (see the arXiv version of our paper [26] for a detailed discussion).

On the other hand, suppose we just wanted to minimize a function ψ, and we start from any point x where

$$\|\nabla\psi(x)\| > \varepsilon \quad \text{or} \quad \lambda_{\min}(\nabla^2\psi(x)) < -\theta$$

for some $\varepsilon, \theta > 0$. Then we can always find a direction to travel in along which either ψ decreases rapidly at a rate of at least ε, or the second derivative of ψ is less than $-\theta$ (see Remark 2.2). By searching in such a direction we can easily find a new point which has a smaller value of ψ using only local information about the gradient and Hessian of ψ. This means that we can keep decreasing ψ until we reach a point where $\|\nabla\psi(x)\| \le \varepsilon$ and $\lambda_{\min}(\nabla^2\psi(x)) \ge -\theta$. If ψ is Lipschitz smooth and bounded, we will reach such a point in polynomial time from any starting point [13, 31]. This fact, together with the fact that any point which satisfies these conditions for $\varepsilon = \theta = 0$ is also an exact local minimum, motivates the second-order notion of an approximate local minimum of [31]. For any $\varepsilon, \theta \ge 0$, say that a point $x^\star$ is an (ε, θ)-approximate local minimum for a C^2-smooth function $\psi : \mathbb{R}^d \to \mathbb{R}$ if

$$\|\nabla\psi(x^\star)\| \le \varepsilon \qquad \text{and} \qquad \lambda_{\min}(\nabla^2\psi(x^\star)) \ge -\theta. \tag{2}$$

We say that $x^\star$ is an (ε, θ)-approximate local *maximum* of ψ if $x^\star$ is an (ε, θ)-approximate local *minimum* of $-\psi$. We use two different values of θ: when referring to an (ε, θ)-approximate local maximum on $f(x, \cdot)$, we use $\theta = \sqrt{L\varepsilon}$ and, roughly, when defining an (ε, θ)-approximate local minimum on g_ε, we use $\theta = \sqrt{\varepsilon}$. We explain these choices of θ in Remark 2.2 .

Importantly, one can view the definition given by Inequality (2) as being motivated by a class of second-order optimization algorithms as, roughly speaking, a second-order optimization algorithm

[4] We note that a uniform bound on a function and the Lipschitz constant of its Hessian also implies a bound on the Lipschitz constants of the function and its gradient. Namely, if f is b-bounded with L-Lipschitz Hessian, it is also L_1-Lipschitz with $L_1 \le 4b^{2/3}L^{1/3}$ and has L_2-Lipschitz gradient with $L_2 \le 2b^{1/3}L^{2/3}$. We say $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is L_1-Lipschitz if $|f(x, y) - f(\tilde{x}, \tilde{y})| \le L_1\sqrt{\|x - \tilde{x}\|^2 + \|y - \tilde{y}\|^2}$, and that f has L_2-Lipschitz gradient if $\|\nabla f(x, y) - \nabla f(\tilde{x}, \tilde{y})\| \le L_2\sqrt{\|x - \tilde{x}\|^2 + \|y - \tilde{y}\|^2}$.

can rapidly decrease the value of ψ when starting from any point which is *not* an approximate local minimum.

2.1 Greedy Path and Greedy Max

When defining a greedy path, we restrict the max-player to updating y by traveling along continuous paths which start at the current value of y and along which either f is increasing or the second derivative of f is positive.

Definition 2.1 (Greedy Path). *Let $x \in \mathbb{R}^d$, and suppose a continuous path $\varphi_t : [0, \tau] \to \mathbb{R}^d$ is differentiable except at a finite number of points, and at the points where it is differentiable $\left\| \frac{\mathrm{d}}{\mathrm{d}t}\varphi_t \right\| = 1$ (i.e., the path travels at unit speed). Then for any $\varepsilon \geq 0$, we say φ is an ε-greedy path for $f(x, \cdot)$ if at all points where φ is differentiable $\frac{\mathrm{d}}{\mathrm{d}t} f(x, \varphi_t) \geq -\varepsilon$ and*

$$\frac{\mathrm{d}}{\mathrm{d}t} f(x, \varphi_t) > \varepsilon \qquad \text{or} \qquad \frac{\mathrm{d}^2}{\mathrm{d}t^2} f(x, \varphi_t) > \sqrt{L\varepsilon}. \tag{3}$$

Roughly speaking, when restricted to updates obtained from ε-greedy paths, the max-player will always be able to reach a point which is a second-order $(\varepsilon, \sqrt{L\varepsilon})$-approximate local maximum for $f(x, \cdot)$, although there may not be an ε-greedy path which leads the max-player to a global maximum.

To define an alternative to $\max_z f(\cdot, z)$, we consider the local maximum point with the largest value of $f(x, \cdot)$ attainable from a given starting point y by any ε-greedy path. Towards this end, we define the set $S_{\varepsilon,x,y} \subseteq \mathbb{R}^d$ of endpoints of ε-greedy paths, for any $x, y \in \mathbb{R}^d$ and $\varepsilon > 0$. We say that a point $z \in S_{\varepsilon,x,y}$ if there is a number $\tau \geq 0$ and a path $\varphi : [0, \tau] \to \mathbb{R}^d$ which is an ε-greedy path for $f(x, \cdot)$, with initial point $\varphi_0 = y$ and endpoint $\varphi_\tau = z$. The greedy max function $g_\varepsilon(x, y)$ is the maximum value of $f(x, \cdot)$ attainable by *any* ε-greedy path in the set $S_{\varepsilon,x,y}$:

$$g_\varepsilon(x, y) := \sup\{f(x, z) : z \in S_{\varepsilon,x,y}\}. \tag{4}$$

Remark 2.2 (Greedy Paths can Escape Saddle Points and Local Minima). *Equations (3) together ensure that for any y where either (i) the gradient $\nabla_y f(x, y)$ has magnitude greater than ε or (ii) the eigenvalues of the Hessian $\nabla_y^2 f(x, y)$ are bounded below by $-\sqrt{L\varepsilon}$, there is always a unit-speed greedy path (with parameter ε) starting at y which can increase the value of f at an average rate [5] of at least $\frac{1}{2}\varepsilon$ by traveling a distance of at most $\frac{1}{2}\frac{\sqrt{\varepsilon}}{\sqrt{L}}$. Moreover, since one such greedy path is always a straight line in the direction of either the gradient $\nabla_y f(x, y)$ or the largest eigenvector of $\nabla_y^2 f(x, y)$, all one needs to compute such a path is access to the gradient and Hessian of $f(x, \cdot)$. This fact can also be viewed as a motivation for the definition of approximate local maximum (Inequality (2)): roughly, any point which does not satisfy both conditions (i) and (ii) (up to a constant factor) is an approximate local maximum. Thus, starting from any point y which is not an approximate local maximum of $f(x, \cdot)$ (with parameters $(\varepsilon, \sqrt{L\varepsilon})$), there is always an easy-to-compute greedy path (with parameter ε) which allows one to increase the value of f.*

[5] By "average rate" of at least $\frac{1}{2}\varepsilon$ we mean that the increase in f divided by the length of the path is $\geq \frac{1}{2}\varepsilon$.

2.2 Dealing with Discontinuities and Other Difficulties of the Greedy Max Function

Unfortunately, even if f is smooth, the greedy max function may not be differentiable with respect to x and may even be discontinuous (see Example 2.4 for a simple example of a smooth function f whose greedy max function is discontinuous). This lack of smoothness creates a problem, since the current definition of approximate local minimum (Inequality (2)) only applies to C^2-smooth functions. To solve this problem we would ideally like to smooth a discontinuous function by convolution with a Gaussian (see Section 5.4 for further discussion on why we use convolution for smoothing).

Another difficulty is that the value of $g_\varepsilon(x, y)$ may be intractable to compute at some points (x, y), since one may need to compute a very large number[6] of ε-greedy paths (possibly an infinite number of paths), each with the same initial point y, before finding the ε-greedy path with the largest value of f. This is because, starting from a point near a local minimum or saddle point, there may be many directions to choose from which allow one to increase the value of f, and, depending on which direction one chooses, one may end up at a different local maximum. Realistically, this means that in general we cannot hope to give our algorithm access to the exact value of g_ε. Our algorithm overcomes this by instead computing a lower bound h_ε for g_ε, and uses only access to this lower bound to minimize g_ε (In Sections 4.2-4.4 of our technical overview we show how this can be done by using some additional properties of the greedy max function). To allow us to handle this more difficult setting, we would like our notion of approximate local minimum to satisfy the property that any point which is an *exact* local minimum, is also an *approximate* local minimum under our definition.

Unfortunately, convolution can cause the local minima of a function to "shift"– a point which is a local minimum for a function $\psi : \mathbb{R}^d \to \mathbb{R}$ may no longer be a local minimum for the convolved version of ψ (for instance, in Example 5.3, we show that this happens if we convolve the function $\psi(x) = x - 3x\mathbf{1}(\mathbf{x} \leq \mathbf{0}) + \mathbf{1}(\mathbf{x} \leq \mathbf{0})$ with a Gaussian $N(0, \sigma^2)$ for any $\sigma > 0$). To avoid this, we instead consider a "truncated" version of ψ, and convolve this function with a Gaussian to obtain our smoothed version of ψ (Definition 2.3).

Definition 2.3 (Approximate Local Minimum for Discontinuous Functions). *For any $\varepsilon, \sigma \geq 0$, we say that $x^\star$ is an (ε, σ)-approximate local minimum for a uniformly bounded function ψ if*

$$\|\nabla_x S(x^\star)\| \leq \varepsilon \qquad \text{and} \qquad \lambda_{\min}(\nabla_x^2 S(x^\star)) \geq -\sqrt{\varepsilon}, \tag{5}$$

where $S(x) := \mathbb{E}_{\zeta \sim N(0, I_d)}[\min(\psi(x + \sigma\zeta), \psi(x^\star))]$.

Example 2.4 (A Simple Example of a Discontinuous Greedy Max Function). *Consider the function*

$$f(x, y) = \cos(x + y)\sin(2x + 2y) - e^{-x^2}.$$

For any $0 < \varepsilon < 0.1$, the greedy max function $g_\varepsilon(x, y)$ is discontinuous at the (parallel) lines $x + y = -2.52$ and $x + y = -0.62$, with $g_\varepsilon(x, y) = -e^{-x^2}$ in the region enclosed between the two lines and $g_\varepsilon(x, y) = -e^{-x^2} + 0.77$ on each side of that region. Such examples are easy to come by and extend to higher dimensions.

[6] Even in the setting where f is bounded with Lipschitz Hessian, the number of ε-greedy paths that share a given starting point may be infinite.

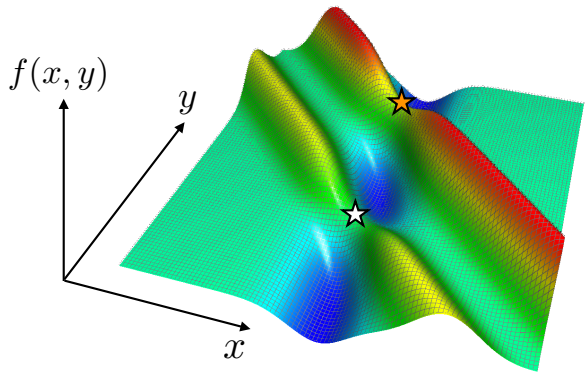

Figure 1: In this example we have $f : \mathbb{R}^1 \times \mathbb{R}^1 \to \mathbb{R}$ where $f(x,y) = 1.2e^{-(x+y+2)^2} + 2e^{-(x+y-2)^2} - e^{-x^2}$. This function has two (ε, σ)-greedy adversarial equilibria (for $\varepsilon = 0$ and any $0 < \sigma \le \frac{1}{100}$), at the points $(0,-2)$ (white star) and $(0,2)$ (orange star). The greedy adversarial equilibrium at $(0,2)$ is also the unique global min-max point of f. On the other hand, f has no local min-max points: there does not exist a point (x,y) where y is a local maximum of $f(x,\cdot)$ and x is a local minimum of $f(\cdot,y)$.

2.3 Greedy Adversarial Equilibrium

We say that $(x^\star, y^\star)$ is an (ε,σ)-greedy adversarial equilibrium of a function $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ with L-Lipschitz Hessian if $y^\star$ is an $(\varepsilon, \sqrt{L\varepsilon})$-approximate local maximum of $f(x^\star,\cdot)$ (in the sense of Inequality (2)), and if $x^\star$ is an $(\varepsilon,\sqrt{\varepsilon})$-approximate local minimum of the (possibly) discontinuous function $g_\varepsilon(\cdot, y^\star)$ (in the sense of Definition 2.3). See Figure 1 for an example of greedy adversarial equilibria.

Definition 2.5 (Greedy Adversarial Equilibrium). *For any $\varepsilon, \sigma \ge 0$, we say that $(x^\star, y^\star) \in \mathbb{R}^d \times \mathbb{R}^d$ is an (ε,σ)-greedy adversarial equilibrium of a C^2-smooth function $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ with L-Lipschitz Hessian, if we have*

$$\|\nabla_y f(x^\star, y^\star)\| \le \varepsilon \quad \text{and} \quad \lambda_{\max}(\nabla_y^2 f(x^\star, y^\star)) \le \sqrt{L\varepsilon}, \ \text{and} \tag{6}$$

$$\|\nabla_x S(x^\star)\| \le \varepsilon \quad \text{and} \quad \lambda_{\min}(\nabla_x^2 S(x^\star)) \ge -\sqrt{\varepsilon}, \tag{7}$$

where $S(x) := \mathbb{E}_{\zeta\sim N(0,I_d)}[\min(g_\varepsilon(x+\sigma\zeta, y^\star), g_\varepsilon(x^\star, y^\star))]$.

We can view the point $(x^\star, y^\star)$ in Definition 2.5 as a type of equilibrium. Namely, suppose that the max-player can only make updates in the set $S_{\varepsilon,x,y^\star}$ of points attainable by an ε-greedy path initialized at $y^\star$. Then under this constraint, the max-player cannot make any update to $y^\star$ that will increase the value of $f(x^\star,\cdot)$. Moreover, we have that $x^\star$ is an (ε,σ)-approximate local minimum (in the sense of Definition 2.3) of the function $\max_z f(x,z)$ if the maximum is taken over the set $S_{\varepsilon,x,y^\star}$ of updates available to the max-player.

A key feature of greedy adversarial equilibrium is that it empowers the min-player to simulate the updates of the max-player via a class of second-order optimization algorithms which we model using greedy paths. This is in contrast to previous models, such as the local min-max point considered in [2, 9, 18] or [20], which restrict the min-player and max-player to making updates inside a small ball.

3 MAIN RESULT

Theorem 3.1 (Main Result). *Let $\varepsilon, \sigma > 0$, with $\sigma \le \frac{1}{\sqrt{\varepsilon d}}$, and consider any C^2-smooth uniformly bounded function $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ with Lipschitz Hessian. Then there exists a point $(x^\star, y^\star) \in \mathbb{R}^d \times \mathbb{R}^d$ which is an $(\varepsilon^\star, \sigma)$-greedy adversarial equilibrium for f, for some $\varepsilon^\star \le \varepsilon$. Moreover, there exists an algorithm which, given access to oracles for the value of a C^2-smooth function $f : \mathbb{R}^d \times \mathbb{R}^d \to [-b, b]$, and to oracles for $\nabla_y f$ and $\nabla_y^2 f$, where f has L-Lipschitz Hessian for some $b, L > 0$, and numbers $\varepsilon, \sigma \ge 0$, with probability at least $\frac{9}{10}$ generates a point $(x^\star, y^\star) \in \mathbb{R}^d \times \mathbb{R}^d$ which is an $(\varepsilon^\star, \sigma)$-greedy adversarial equilibrium for f, for some $\varepsilon^\star \le \varepsilon$. Moreover, this algorithm takes a number of gradient, Hessian, and function evaluations which is polynomial in $\frac{1}{\varepsilon}, d, b, L, \frac{1}{\sigma}$.*

As noted earlier, our result does not require additional assumptions on f such as convexity or monotonicity [32, 33, 36] or sufficient bilinearity [1]. Our algorithm also converges from any initial point. This is in contrast to many previous works [2, 18, 20, 37], which assume that there exists a stationary point for their algorithm on the function f, and that their algorithm is initialized somewhere in the region of attraction for this stationary point. To the best of our knowledge, greedy adversarial equilibrium is the first model for optimization in the presence of an adversary which is both guaranteed to exist and can be found efficiently in a general setting where f is C^2-smooth, bounded, and has Lipschitz Hessian. We expect it to find further use in learning in the presence of adversarial agents.

We note that we have not tried to optimize the order of the polynomial running time bound in Theorem 3.1. Finally, the greedy adversarial equilibrium our algorithm finds depends on the initial point (x_0, y_0). To find other greedy adversarial equilibria, one can start from different initial points.

4 TECHNICAL OVERVIEW

In this section we give an overview of our algorithm and of the proof of Theorem 3.1. In Section 6 we give a full description of the algorithm. In Section 7 we present the main lemmas and propositions that we use to prove Theorem 3.1, along with short summaries of their proofs; for the full proofs see the arXiv version of our paper [26]. To simplify the exposition, we set $L = b = 1$, $0 < \varepsilon < 1$, and $\sigma = \frac{1}{\sqrt{d}}$. In particular, if f is 1-bounded with 1-Lipschitz Hessian, it is also 4-Lipschitz with 2-Lipschitz gradient.

4.1 An Efficiently Computable Second-Order Local Approximation to $\max_z f(x,z)$

Ideally, we would like our algorithm to be able to compute the global maximum $\max_z f(x,z)$ at any point x. However, since f may be nonconvex-nonconcave, finding the global maximum may be intractable in our setting.

Instead, starting from some initial point $z \leftarrow y$, we use a second-order maximization algorithm (Algorithm 1) to find an $(\varepsilon,\sqrt{\varepsilon})$-approximate local maximum of $f(x,\cdot)$. At each step, we would like our maximization algorithm to be able to rapidly decrease the value of $f(x,z)$ from any point z that is not an $(\varepsilon,\sqrt{\varepsilon})$-approximate local maximum of $f(x,\cdot)$, that is, if z is such that either $\|\nabla_y f(x,z)\| > \varepsilon$ or

$\lambda_{\max}(\nabla_y^2 f(x,z)) > \sqrt{\varepsilon}$. Towards this end, if $\|\nabla_y f(x,z)\| > \varepsilon$, we have the max-player make an update

$$z \longleftarrow z + \mu_1 \nabla_y f(x,z)$$

for some step size $\mu_1 > 0$. If we set

$$\mu_1 \le \frac{1}{4}, \tag{8}$$

we have that, since f has 2-Lipschitz gradient,

$$\nabla_y f(x,p)^\top \frac{\nabla_y f(x,z)}{\|\nabla_y f(x,z)\|} \ge \frac{1}{2}\|\nabla_y f(x,z)\| > \frac{1}{2}\varepsilon$$

at every point p on the line segment $[z, z+\mu_1 \nabla_y f(x,z)]$. This means that f increases by at least $\frac{1}{2}\varepsilon \times \mu_1 \|\nabla_y f(x,z)\| > \frac{1}{2}\mu_1 \varepsilon^2$ if the max-player makes this update. On the other hand, if $\|\nabla_y f(x,z)\| \le \varepsilon$ but $\lambda_{\max}(\nabla_y^2 f(x,z)) > \sqrt{\varepsilon}$, we have the max-player make an update $z \longleftarrow z + \mu_3 \mathsf{a} v$ in the direction of the largest eigenvector v of $\nabla_y^2 f(x,z)$ (with $\|v\| = 1$), for some step size $\mu_3 > 0$, where the sign $\mathsf{a} \in \{-1, 1, 1\}$ is chosen such that $(\mathsf{a}v)^\top \nabla_y f \ge 0$. If we set

$$\mu_3 \le \frac{1}{2}\sqrt{\varepsilon} \tag{9}$$

we have that, since f has 1-Lipschitz Hessian,

$$v^\top \nabla_y^2 f(x,p) v > \frac{1}{2}\sqrt{\varepsilon}$$

for every point p on the line segment $[z, z+\mu_3 \mathsf{a} v]$. This means that f increases by at least $\frac{1}{2}(\mu_3)^2\sqrt{\varepsilon}$ if the max-player makes this update. Finally, if $\|\nabla_y f(x,z)\| \le \varepsilon$ and $\lambda_{\max}(\nabla_y^2 f(x,z)) \le \sqrt{\varepsilon}$, our maximization algorithm (Algorithm 1) has reached an $(\varepsilon, \sqrt{\varepsilon})$-approximate local maximum y', and it returns this point y'.

The above discussion implies that the value of f increases by at least $\Delta := \min(\frac{1}{2}\mu_1\varepsilon^2, \frac{1}{2}(\mu_3)^2\sqrt{\varepsilon})$ at each iteration. Since f is also 1-bounded, and each step of our method requires $O(1)$ oracle calls to the gradient and Hessian of f, and our method only stops once it reaches an $(\varepsilon, \sqrt{\varepsilon})$-approximate local maximum, it uses at most $O(\frac{1}{\Delta})$ oracle calls to find an $(\varepsilon, \sqrt{\varepsilon})$-approximate local maximum y' (Lemmas 7.1, 7.2). Thus, the value of f at this $(\varepsilon, \sqrt{\varepsilon})$-approximate local maximum y', which we denote by the function h_ε,

$$h_\varepsilon(x,y) = f(x,y'), \tag{10}$$

gives us a local approximation for $\max_z f(x,z)$, which can be computed in $O(\frac{1}{\Delta})$ oracle calls for the gradient, Hessian, and value of f. Moreover, as we explain in Remark 4.1, the steps of our optimization method form an ε-greedy path from y to y'.

Remark 4.1 (Our Second-Order Maximization Algorithm Computes an ε-Greedy Path). *We have shown that at each point p on a line segment $[z, z']$ connecting two consecutive steps z and z' of our maximization algorithm, we have that either $\nabla_y f(x,p)^\top u > \frac{1}{2}\varepsilon$ or $u^\top \nabla_y^2 f(x,p) u > \frac{1}{2}\sqrt{\varepsilon}$, where u is the unit vector $u = \frac{z'-z}{\|z'-z\|}$. Therefore, at any point on the unit-speed path φ_t made up of the line segments connecting the consecutive steps of our algorithm, we have that either $\frac{\mathrm{d}}{\mathrm{d}t} f(x, \varphi_t) > \frac{1}{2}\varepsilon$, or $\frac{\mathrm{d}^2}{\mathrm{d}t^2} f(x, \varphi_t) > \frac{1}{2}\sqrt{\varepsilon}$. This implies that the path traced by our algorithm is a $\frac{1}{2}\varepsilon$-greedy path.*

We show that, for a smaller choice of step sizes than required by (8) and (9), namely for $\mu_1, \mu_3 = \mathrm{poly}\left(\frac{1}{d}, \varepsilon\right)$, this path is ε'-greedy where $|\varepsilon - \varepsilon'| = \mathrm{poly}\left(\frac{1}{d}, \varepsilon\right)$. The fact that $\varepsilon' \neq \varepsilon$ causes some technical issues which we deal with in the full algorithm[7] in Section 6 and in our proof (Section 7). To improve readability, we ignore these issues in this technical overview and assume that the path is ε-greedy.

4.2 Finding a Point $(x^\star, y^\star)$ Which Is a Local Min for $h_\varepsilon(\cdot, y^\star)$ and a Local Max for $f(x^\star, \cdot)$

Now that we have a subroutine for computing h_ε, our next goal is to find an (ε, σ)-approximate local minimum in the x variable for h_ε. If we were able to compute the global maximum $\max_z f(x,z)$, it would be enough for our algorithm to find a global minimizer x_{global} for $\max_z f(x,z)$, and to then find a global maximizer y_{global} for $f(x_{\text{global}}, \cdot)$. Since $\max_z f(x,z)$ is only a function of x, the point $(x_{\text{global}}, y_{\text{global}})$ would still be a global min-max point regardless of which global maximizer y_{global} we find. Here we encounter a difficulty:

Obstacle 1: Since the function $h_\varepsilon(x,y)$ is a *local* approximation for $\max_z f(x,z)$, it depends both on x and on the initial point y. If our algorithm were to first find an (ε, σ)-approximate local minimum $x^\star$ for $h_\varepsilon(\cdot, y)$, and then search for an $(\varepsilon, \sqrt{\varepsilon})$-approximate local maximum $y^\star$ for $f(x^\star, \cdot)$, the point $x^\star$ may not be an (ε, σ)-approximate local minimum for $h_\varepsilon(\cdot, y^\star)$ even though it is an (ε, σ)-approximate local minimum for $h_\varepsilon(\cdot, y)$. To get around this problem, we use a different update rule for the min-player and max-player:

Idea 1: Alternate between a step where the min-player makes an update w to x which decreases the value of $h_\varepsilon(\cdot, y)$ by some amount γ_1, and a step where the max-player uses the maximization subroutine discussed in Section 4.1 (Algorithm 1), with initial point y, to find a $(\varepsilon, \sqrt{\varepsilon})$-approximate local maximum y' for $f(x, \cdot)$.

Since h_ε is the value of f at the $(\varepsilon, \sqrt{\varepsilon})$-approximate local maximum y', we therefore have that $h_\varepsilon(w,y) = f(w,y')$. Moreover, since y' is an $(\varepsilon, \sqrt{\varepsilon})$-approximate local maximum for f, and Algorithm 1 stops whenever it reaches an $(\varepsilon, \sqrt{\varepsilon})$-approximate local maximum, y' is a stationary point for Algorithm 1, which means that $f(w,y') = h_\varepsilon(w,y')$. Thus,

$$h_\varepsilon(w,y) = f(w,y') = h_\varepsilon(w,y'). \tag{11}$$

Moreover, since f is b-bounded, and $h_\varepsilon(x,y)$ is the value of $f(x,\cdot)$ at the approximate local maximum obtained by the maximization subroutine in Section 4.1, h_ε must also be b-bounded. Thus, since the max-player's update does not change the value of h_ε (11), if we can show that whenever x is not an (ε, σ)-approximate local minimum of $f(\cdot, y)$ the min player can find an update to x which decreases the value of $h_\varepsilon(x,y)$ by at least some fixed amount $\gamma_1 > 0$, then we would have that the value of h_ε decreases monotonically at each iteration by at least γ_1. In that case, our algorithm would have to converge after $\frac{2b}{\gamma_1}$ iterations to a point (x, y') where x is an (ε, σ)-approximate local minimum for $h_\varepsilon(\cdot, y')$ and y' is an $(\varepsilon, \sqrt{\varepsilon})$-approximate local maximum for $f(x, \cdot)$.

4.3 Escaping "Saddle Points" of the Discontinuous Function h_ε

Before we can apply Idea 1, we would like to find a way for the min-player to find updates for x which decrease the value of $h_\varepsilon(x,y)$ by

[7] In the full algorithm we deal with this issue by initially computing an ε'-greedy path for $\varepsilon' = \frac{\varepsilon}{2}$ at the first iteration, and then slowly increasing ε' at each iteration.

some amount at least γ_1 whenever the current value of x is not an (ε, σ)-approximate local minimum (in the sense of Definition 2.3). However, since h_ε is discontinuous we encounter a second obstacle:

Obstacle 2: Finding an (ε, σ)-approximate local minimum of h_ε requires our algorithm to escape saddle points of the truncated and smoothed function[8] $s(w) = \mathbb{E}_{\zeta \sim N(0,I_d)}[\min(h_\varepsilon(w + \sigma\zeta, y), h_\varepsilon(x, y))]$. Ideally, we would like to run a "noisy" version of gradient descent, which can allow us to escape saddle points (see e.g., [13]), but we do not have access to the gradient of s.

To get around this problem, we compute a stochastic gradient for $s(w)$ which can be computed without access to a gradient:

Idea 2: We use a stochastic gradient $\Gamma(w)$ for $s(w)$ which can be computed with access only to the value of h_ε; roughly

$$\Gamma(w) = \frac{\zeta}{\sigma} \min\left(h_\varepsilon(w + \sigma\zeta, y), h_\varepsilon(x, y)\right) \tag{12}$$

where $\zeta \sim N(0, I_d)$, and $\mathbb{E}[\Gamma(w)] = s(w)$ (see e.g. [12]). This allows us to use a noisy version of stochastic gradient descent (SGD) to escape saddle points of $s(w)$.

More specifically, starting at the initial point $w \longleftarrow x$, each step of this "noisy" SGD is given by

$$w \longleftarrow w - \eta\Gamma(w) + \alpha\xi \tag{13}$$

where $\xi \sim N(0, I_d)$ and η, α are hyperparameters . We then apply concentration bounds for our stochastic gradient (Proposition 7.6) to results about noisy SGD [19] to show that, whenever x is not an (ε, σ)-approximate local minimum for $h_\varepsilon(\cdot, y)$, with high probability, this noisy SGD, with hyperparameters $\eta, \alpha = \text{poly}\left(\frac{1}{d}, \varepsilon\right)$, can find an update for x which decreases the value of h_ε by at least $\gamma_1 = \text{poly}\left(\frac{1}{d}, \varepsilon\right)$ after $\mathcal{I} = \text{poly}\left(d, \frac{1}{\varepsilon}\right)$ iterations of Equation (13) (Proposition 7.8).

4.4 Using h_ε to Minimize Greedy Max Function

Although we have shown how to find an (ε, σ)-approximate local minimum of h_ε (In the sense of Definition 2.3), our goal is to find an (ε, σ)-approximate local minimum of the greedy max function g_ε. For simplicity, let us start by supposing that we were able to find an *exact* local minimum for h_ε. Since $g_\varepsilon(x, y)$ is the maximum value of $f(x, z)$ that is attainable at the endpoint z of *any* ε-greedy path that starts at y, and, as we have shown in Remark 4.1, the steps of the second-order optimization algorithm we use to compute $h_\varepsilon(x, y)$ form one such greedy path whose endpoint y' determines the value of h_ε (Equation (10)), we have that (Proposition 7.4)

$$h_\varepsilon(x, y) \le g_\varepsilon(x, y) \qquad \forall (x, y) \in \mathbb{R}^d \times \mathbb{R}^d. \tag{14}$$

However, it is still not clear how h_ε can help us find a local minimizer for g_ε:

Obstacle 3: We want to minimize the greedy max function g_ε, but computing the value of g_ε may be intractable, and we only have access to a lower bound $h_\varepsilon \le g_\varepsilon$.

We would like to somehow use our ability to compute h_ε to find a local minimum of g_ε. Towards this end, we observe that if any point $y^\star$ is an $(\varepsilon, \sqrt{\varepsilon})$-approximate local maximum, then the conditions from the definition of approximate local maximum, $\|\nabla_y f(x, y^\star)\| \le \varepsilon$ and $\lambda_{\min}(\nabla^2 f(x, y^\star)) \le \sqrt{\varepsilon}$, imply that there is no unit speed path $\varphi : [0, \tau] \longrightarrow \mathbb{R}^d$ starting at the point $\varphi_0 = y^\star$ for which $\frac{\mathrm{d}}{\mathrm{d}t} f(x, \varphi_0) > \varepsilon$ or $\frac{\mathrm{d}^2}{\mathrm{d}t^2} f(x, \varphi_t) > \sqrt{\varepsilon}$ on all $t \in (0, \tau]$. This means that, if $y^\star$ is an $(\varepsilon, \sqrt{\varepsilon})$-approximate local maximum of $f(x, \cdot)$, then it is the *only* $(\varepsilon, \sqrt{\varepsilon})$-approximate local maximum reachable by any ε-greedy path starting at $y^\star$. In other words, we have that,

$$h_\varepsilon(x, y^\star) = g_\varepsilon(x, y^\star) \tag{15}$$

whenever $y^\star$ is an $(\varepsilon, \sqrt{\varepsilon})$-approximate local maximum for $f(x, \cdot)$ (Proposition 7.5). Together, (14) and (15) imply the following:

Idea 3: For any pair of points $(x^\star, y^\star)$ where $y^\star$ is an $(\varepsilon, \sqrt{\varepsilon})$-approximate local maximum for $f(x^\star, \cdot)$, we have that if $x^\star$ is an exact local minimum for $h_\varepsilon(\cdot, y^\star)$ it must also be an exact local minimum for $g_\varepsilon(\cdot, y^\star)$.

This is because, if $x^\star$ is an exact local minimum, then there is an open ball B containing $x^\star$ where $h_\varepsilon(x^\star, y^\star) = \min_{w \in B} h_\varepsilon(w, y^\star)$. This implies that

$$g_\varepsilon(x^\star, y^\star) \overset{(15)}{=} h_\varepsilon(x^\star, y^\star) = \min_{w \in B} h_\varepsilon(w, y^\star) \overset{(14)}{\le} \min_{w \in B} g_\varepsilon(w, y^\star) \tag{16}$$

and hence that $x^\star$ minimizes $g_\varepsilon(\cdot, y^\star)$ on the ball B.

Finally, we extend the result in Idea 3, which holds for exact local minima, to a similar result (Lemma 7.3) that holds for *approximate* local minima. We show that if, roughly, the variance of our stochastic gradient Γ (Equation (12)) satisfies a $\text{poly}(\frac{1}{d}, \varepsilon)$ upper bound (Proposition 7.6), then for any pair of points $(x^\star, y^\star)$ where $y^\star$ is an $(\varepsilon, \sqrt{\varepsilon})$-approximate local maximum of $f(x^\star, \cdot)$ (in the sense of Equation (2)), if $x^\star$ is an (ε, σ)-approximate local minimum of $h_\varepsilon(\cdot, y^\star)$ then it is also an (ε, σ)-approximate local minimum for $g_\varepsilon(\cdot, y^\star)$ (in the sense of Definition 2.3). The proofs of Lemma 7.3 and Proposition 7.6 are technical and summarized in Section 7.

4.5 Showing Convergence to Greedy Adversarial Equilibrium in $\text{poly}\left(d, \frac{1}{\varepsilon}\right)$ Oracle Calls

From Idea 1 we have that our algorithm terminates after $O\left(\frac{1}{\gamma_1}\right)$ iterations consisting of an update for the min-player and max-player, if one can bound by some number $\gamma_1 > 0$ the amount by which each update for the min-player decreases the value of $h_\varepsilon(x, y)$.

From Idea 2 (Section 4.3) we have that, with high probability, noisy SGD can allow the min-player to find an update which decreases the value of h_ε by an amount $\gamma_1 = \text{poly}\left(\frac{1}{d}, \varepsilon\right)$, and that this can be accomplished in $\mathcal{I} = \text{poly}\left(d, \frac{1}{\varepsilon}\right)$ computations of h_ε.

In Section 4.1, we show that our maximization subroutine can compute the value of h_ε in at most $O\left(\frac{1}{\Delta}\right)$ oracle calls, where $\Delta = \min\left\{\frac{1}{2}\mu_1\varepsilon^2, \frac{1}{2}(\mu_3)^2\sqrt{\varepsilon}\right\}$. From Equations (8) and (9), roughly speaking, we may set $\mu_1 = \frac{1}{4}$ and $\mu_3 = \frac{1}{2}\sqrt{\varepsilon}$. [9]

This means that, with high probability, the number of oracle calls until our algorithm terminates is $O\left(\frac{1}{\Delta} \times \mathcal{I} \times \frac{1}{\gamma_1}\right) = \text{poly}\left(d, \frac{1}{\varepsilon}\right)$. Finally, from Idea 1 we also have that, if our algorithm terminates, it returns a pair of points $(x^\star, y^\star)$ where $x^\star$ is an (ε, σ)-approximate local minimum for $h_\varepsilon(\cdot, y^\star)$ (in the sense of Definition 2.3) and $y^\star$

[8] We use a lowercase s for the truncated and smoothed version of h_ε to distinguish it from the truncated and smoothed version of g_ε used in Definition 2.5.

[9] As mentioned in Remark 4.1, in the full algorithm we use somewhat smaller hyperparameter values $\mu_1, \mu_3 = \text{poly}(\frac{1}{d}, \varepsilon)$ to ensure that the path computed by Algorithm 1 is ε-greedy instead of $\frac{1}{2}\varepsilon$-greedy.

is an $(\varepsilon, \sqrt{\varepsilon})$-approximate local maximum for $f(x^\star, \cdot)$ (in the sense of Equation (2)).

Applying Idea 3 (or rather its extension to *approximate* local minima), we have that $x^\star$ is an (ε, σ)-approximate local minimum for $h_\varepsilon(\cdot, y^\star)$, which implies that $(x^\star, y^\star)$ is a (ε, σ)-greedy adversarial equilibrium. In other words, our algorithm returns an (ε, σ)-greedy adversarial equilibrium after at most poly $\left(d, \frac{1}{\varepsilon}\right)$ oracle calls for the gradient, Hessian, and value of f.

4.6 Summary of Algorithm

The discussion in Sections 4.1-4.5 leads us to the following algorithm (see Algorithm 2 for the full description). In addition to oracles for $f, \nabla_y f$ and $\nabla^2_y f$, our algorithm also takes as input an initial point (x_0, y_0) in $\mathbb{R}^d \times \mathbb{R}^d$, and parameters $\varepsilon, \sigma > 0$ (recall we have set $\sigma = \frac{1}{\sqrt{d}}$ in this section). [10]

(1) Starting at the initial point (x_0, y_0), our algorithm first uses the second-order optimization method described in Section 4.1 (Algorithm 1) to find a point y_1 which is an $(\varepsilon, \sqrt{\varepsilon})$-approximate local maximum for $f(x_0, \cdot)$.
(2) Next, starting from iteration $i = 1$, and setting $x_1 \leftarrow x_0$, our algorithm uses noisy SGD (Equation (13))[11] to search for a point x_{i+1} for which,
$$h_\varepsilon(x_{i+1}, y_i) \le h_\varepsilon(x_i, y_i) - \gamma_1, \tag{17}$$
where $\gamma_1 = \text{poly}(\frac{1}{d}, \varepsilon)$ (Lines 21-37 of Algorithm 2). When running noisy SGD, roughly speaking, our algorithm uses the stochastic gradient Γ (the same stochastic gradient as in Equation (12)),
$$\Gamma(w) = \frac{\zeta}{\sigma} \min\left(h_\varepsilon(w + \sigma\zeta, y_i), h_\varepsilon(x_i, y_i)\right), \tag{18}$$
where $\zeta \sim N(0, I_d)$ and h_ε is computed using the second-order optimization method of Section 4.1 (Algorithm 1).
(3) If our algorithm is able to find an update x_{i+1} which satisfies Inequality (17), it uses Algorithm 1 to compute a point y_{i+1}, which is an $(\varepsilon, \sqrt{\varepsilon})$-approximate local maximum for $f(x_{i+1}, \cdot)$, sets $i \leftarrow i + 1$, and goes back to Step 2. Otherwise, if it cannot find such an update, it concludes that x_i is an (ε, σ)-approximate local minimum for $h_\varepsilon(\cdot, y_i)$, and, hence, that (x_i, y_i) is a (ε, σ)-greedy adversarial equilibrium.

5 DISCUSSIONS AND LIMITATIONS

5.1 How Does Our Greedy Adversarial Equilibrium Compare to Previous Models?

In previous papers different models which can be seen as alternatives to min-max optimization have been considered in the nonconvex setting. A number of papers [2, 9, 18] consider the local min-max point model (sometimes called a "local Nash" point or "local saddle" point). Any point which is a local min-max point is also a greedy adversarial equilibrium for small enough $\sigma > 0$ (Corollary 5.1; see the arXiv version of our paper a proof [26]).

Corollary 5.1. *Suppose that for any $\delta > 0$, $(x^\star, y^\star)$ is a $(0, \delta)$-local min-max point of a C^2-smooth function $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, and that there is a number $b > 0$ such that $|f(x, y)| \le b$ for all $x, y \in \mathbb{R}^d$. Then for any $\varepsilon > 0$ there exists $\sigma^\star > 0$ such that for every $0 < \sigma \le \sigma^\star$ we have that $(x^\star, y^\star)$ is a (ε, σ)-greedy adversarial equilibrium of f.*

Since local min-max points are not guaranteed to exist in general, previous algorithms which seek local min-max points oftentimes make strong assumptions on the function f. For instance, [1] show that if f has 1-Lipschitz gradient and satisfies a "sufficient bilinearity" condition– that is, roughly speaking, if the cross derivative $\nabla^2_{xy} f(x, y)$ has all its singular values greater than some $\gamma > 2$ at every $x, y \in \mathbb{R}^d$–then their algorithm reaches a point $(x^\star, y^\star)$ where $\|\nabla_x f(x^\star, y^\star)\| \le \varepsilon$ and $\|\nabla_y f(x^\star, y^\star)\| \le \varepsilon$ in $O\left(\frac{1}{\gamma^2} \log \frac{M}{\varepsilon}\right)$ evaluations of a Hessian-vector product of f, where M is the magnitude of ∇f at the point where their algorithm is initialized.[12]

In [20] the authors consider an alternative model to min-max optimization which incorporates the fact that in min-max optimization the min-player reveals her strategy before the max-player. In their notion, both players are restricted to making updates in vanishingly small neighborhoods of the optimum point (although the size of the neighborhood for the min-player is allowed to vanish at a much faster rate than the neighborhood for the max-player). One difference between our greedy adversarial equilibrium model and the model considered in [20] is that in [20] the max-player is able to compute a global maximum (albeit when restricted to a ball of vanishingly small radius), while in our greedy adversarial equilibrium model the max-player is constrained to points reachable by a greedy path of any length. That being said, our main result still holds if we restrict the greedy path of the max-player to be proportional to the updates made by the minimizing player (see Remark 5.2). Another difference is that, while the solution point for the model in [20] is not guaranteed to exist in a general nonconvex-nonconcave setting, our main result (Theorem 3.1) guarantees that any uniformly bounded function with Lipschitz Hessian has a greedy adversarial equilibrium.

5.2 Applicability and Limitations of Our Definition

The class of algorithms that our definition allows the players to use includes a range of algorithms, e.g., gradient descent and negative curvature descent [24, 34], which only take steps in directions where the gradient or second derivative is above some threshold value.

Moreover, one can expand our definition to allow the max-player to also use randomized algorithms such as noisy gradient descent [19], as long as the algorithm stops once an approximate local maximum is reached. For this class of algorithms, any point $(x^\star, y^\star)$ which satisfies our original Definition 2.5 also is a greedy adversarial equilibrium under this expanded definition. Roughly speaking, this is because as long as the max-player is at a local maximum for the function $f(x^\star, \cdot)$, expanding the choice of algorithms available to the max-player may increase the value of the greedy max function at points other than $x^\star$ but will not increase the value of the

[10] In the full description of the algorithm we set $(x_0, y_0) = (0, 0)$ for simplicity.
[11] In the full algorithm we combine noisy SGD with a a random hill-climbing method (Lines 11-20 of Algorithm 2).
[12] Note that, since such algorithms' dynamics should ideally not be attracted to minima or maxima of f, additional assumptions on f are typically needed even to show convergence to a first-order stationary point–that is, a point $(x^\star, y^\star)$ where $\|\nabla_x f(x^\star, y^\star)\| \le \varepsilon$ and $\|\nabla_y f(x^\star, y^\star)\| \le \varepsilon$.

greedy max function at the current point $x^\star$. In other words, the minimizing player will not have an incentive to deviate from $x^\star$ if more algorithms are made available to the max-player.

On the other hand, if we allow the max-player to use algorithms which do not stop at local maxima, for instance algorithms such as simulated annealing, a solution $(x^\star, y^\star)$ which satisfies our current definition may no longer be a solution in this expanded sense. This is because, giving the max-player the option to use algorithms which do not stop once a local maximum is reached may cause the greedy max function to increase at $x^\star$ more than at neighboring points, incentivizing the minimizing player to deviate from $x^\star$.

Remark 5.2. *Theorem 3.1 still holds if we restrict the greedy path to a ball whose radius is proportional to ε, L_1 and the distance $\|x - x^\star\|$ between $x^\star$ and the minimizing player's update x. This is because, roughly speaking, any greedy path that leaves this ball would reach a point y for which the value of f at (x, y) is greater than the value of f at $(x^\star, y^\star)$. This implies that the truncated greedy max function $\min(g_\varepsilon(x, y), g_\varepsilon(x^\star, y^\star))$ would have the exact same value regardless of whether we restrict the max-player to such a ball, and the point $(x^\star, y^\star)$ guaranteed by Theorem 3.1 would therefore still satisfy Definition 2.5.*

5.3 The Necessity of Dealing with Discontinuities in the Greedy Max Function

At first glance, it may seem that we can simply restrict ourselves to considering functions $f(x, y)$ for which the greedy max function $g_\varepsilon(x, y)$ is continuous. This would greatly simplify our proof, since we could exclude "unstable" situations where the min-player proposes a small change in x which would then cause the max-player to respond by making a large change in her strategy. A second difficulty involving discontinuous greedy max functions is that, since we allow our algorithm to start at any point, even greedy max functions with discontinuities far from the greedy adversarial equilibrium point(s) are challenging to analyze. Unfortunately, even very simple functions $f(x, y)$ oftentimes have discontinuous greedy max functions $g_\varepsilon(x, y)$ (see Example 2.4). Excluding functions where such discontinuities arise would greatly restrict the applicability of our results, and a large part of our proof is devoted to dealing with the possibility of discontinuities in the greedy max function.

5.4 Additional Discussion of Approximate Local Minimum for Discontinuous Functions

When choosing our definition for approximate local minimum of discontinuous functions, we would like this definition to be as close as possible to the notion of approximate local minimum for C^2-smooth functions (Inequality (2)). This allows us to more easily relate our results to past work in the optimization literature. For instance, in our proof, we would like to adapt results from [19] about escaping saddle points in polynomial time to the setting of discontinuous functions. However, we cannot expect our algorithm to have direct access to the discontinuous function $g_\varepsilon(\cdot, y)$ we wish to minimize. To allow us to handle this more difficult setting, we would like our notion of approximate local minimum to satisfy the property that any point which is an exact local minimum is also an approximate local minimum under our definition.

To obtain a definition which applies to discontinuous functions yet is as close as possible to the definition in (2), we would like to approximate any discontinuous function ψ with a C^2-smooth function.When choosing which C^2-smooth approximation to use, we would like it to satisfy the following three properties.

(1) **C^2-Smooth with Lipschitz Hessian.** We would like each function in our family of approximation functions to be C^2-smooth with Lipschitz Hessian. This would allow us to apply the definition of approximate local minimum for C^2-smooth functions (Inequality (2)) to any function in this family.
(2) **Shared Local Minima.**We want our family of C^2-smooth approximation functions to have the property that for any $x^\star$ which is an (exact) local minimum of the objective function ψ, and any $\varepsilon > 0$, there is a function in this family such that $x^\star$ is also an $(\varepsilon, \sqrt{\varepsilon})$-approximate local minimum of this C^2-smooth function (in the sense of Inequality (2)).
(3) **Easy to Compute.** We want each function in our family of approximation functions to be easily computed within some error ε at any point x in $\text{poly}(d, 1/\varepsilon, b)$ evaluations of ψ.

Towards this end, we consider the family of functions $\mathcal{F}$ where we convolve ψ with a Gaussian density $N(0, \sigma^2 I_d)$ of some variance σ^2 and zero mean. That is, we consider functions of the form

$$\psi_\sigma(x) := \mathbb{E}_{\zeta \sim N(0, I_d)}[\psi(x + \sigma\zeta)]$$

for some $\sigma > 0$. This family of functions is C^2-smooth and has Lipschitz Hessian, which satisfies our first property (1). This is because a Gaussian density is C^2-smooth, and any function convolved with a C^2-smooth function is also C^2-smooth. Moreover, if ψ is b-bounded, then convolving ψ with a Gaussian gives a b-bounded function with the magnitude of its k'th-derivatives bounded by $2b$ times an upper bound on the k'th derivative of the standard Gaussian density, that is, $2b \times \frac{1}{\sigma^{2k+1}\sqrt{2\pi}}$ for every $k > 0$. In particular, this means our smoothed function $\psi_\sigma(x)$ is also b-bounded, with $b \times \frac{1}{\sigma^7\sqrt{2\pi}}$-Lipschitz Hessian.

The family of functions $\mathcal{F}$ also has the advantage that, if ψ is b-bounded, it can be computed within error ε in $\text{poly}(d, 1/\varepsilon, b)$ evaluations of ψ with high probability if one uses a Monte-Carlo computation of the expectation $\mathbb{E}_{\zeta \sim N(0, I_d)}[\psi(x + \sigma\zeta)]$, which satisfies our third property (3).

To satisfy our second property (2), we would ideally like to ensure that, for every exact local minimum $x^\star$ of ψ, and every $\varepsilon > 0$, there is a small enough $\sigma > 0$ such that $x^\star$ is an $(\varepsilon, \sqrt{\varepsilon})$-approximate local minimum (in the sense of Inequality (2)) of the smoothed function $\psi_\sigma = \mathbb{E}_{\zeta \sim N(0, I_d)}[\psi(x + \sigma\zeta)]$. Unfortunately, smoothing ψ by convolution alone does not directly allow us to satisfy property (2). The following example illustrates this problem.

Example 5.3 (Convolution Can Shift Local Minima). *Consider the function $\psi : \mathbb{R} \to \mathbb{R}$, where*

$$\psi(x) = x - 3x\mathbf{1}(\mathbf{x} \le \mathbf{0}) + \mathbf{1}(\mathbf{x} \le \mathbf{0}).$$

This function is discontinuous at $x = 0$, and has an exact local minimum at the point $x = 0$ (which also happens to be its global minimum point). If we smooth ψ by convolving it with a Gaussian distribution

$N(0, \sigma^2)$ for any $\sigma > 0$, we get the smooth function

$$\psi_\sigma(x) = 3\sigma\frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2\sigma^2}} - x + x\Phi\left(\frac{1}{\sigma}x\right) + \Phi\left(-\frac{1}{\sigma}x\right),$$

where $\Phi(\cdot)$ is the standard Gaussian cumulative distribution function. This function is C^2-smooth since $\Phi(\cdot)$ is C^2-smooth. However, for any $\sigma > 0$, the gradient at $x = 0$ of the smoothed function is $-1.5 - \frac{1}{\sigma\sqrt{2\pi}}$. Thus, for any $\sigma > 0$, $x = 0$ is not an approximate local minimum of the smoothed function for any parameter $\varepsilon \le 1.5$.

In Example 5.3 the gradient of the smoothed function ψ_σ at $x^\star = 0$ has magnitude at least 1.5 for any $\sigma > 0$ even though $x^\star = 0$ is a local minimum of ψ. To understand how this is possible, we consider the following stochastic gradient (see e.g. [12]) for the smoothed function ψ_σ:

$$\nabla\psi_\sigma(x) = \sigma^{-1}\mathbb{E}_{\zeta\sim N(0,I_d)}[(\psi(x+\sigma\zeta) - \psi(x))\zeta]. \tag{19}$$

One can obtain a non-zero gradient $\nabla\psi_\sigma(x)$ even if all of the sampled points $x + \sigma\zeta$ in (19) give values $\psi(x + \sigma\zeta)$ greater than $\psi(x)$.

If ψ were smooth, finding a small step $\sigma\zeta$ which increases the value of ψ (by at least some amount proportional to the step size) would imply that ψ decreases in the direction $-\sigma\zeta$. For smooth objective functions one can therefore find a descent direction (a direction in which ψ decreases) simply by first finding an ascent direction $\sigma\zeta$ and then moving in the opposite direction $-\sigma\zeta$. Unfortunately, this is not true for discontinuous functions, since if ψ is discontinuous, it may be that $\psi(x^\star + \sigma\zeta) > \psi(x^\star)$ does not imply that $\psi(x^\star - \sigma\zeta) < \psi(x^\star)$ no matter how small a step $\sigma\zeta$ we take. In other words, for discontinuous objective functions the presence of an "ascent direction" along which the objective function increases does not imply the existence of a "descent direction" along which the objective function decreases. The only thing that matters when determining whether a discontinuous function has a local minimum at some point $x^\star$ is whether, in every ball containing $x^\star$, there are points $x^\star + \sigma\zeta$ for which $\psi(x^\star + \sigma\zeta) < \psi(x^\star)$.

To enable our definition of approximate local minimum to only consider those directions which decrease the value of ψ, when determining whether a point $x^\star$ is an (approximate) local minimum we instead consider the truncated function $\min(\psi(x), \psi(x^\star))$. We then smooth this truncated function by convolving it with a Gaussian, to obtain the following smoothed function of x:

$$\mathbb{E}_{\zeta\sim N(0,I_d)}\left[\min(\psi(x+\sigma\zeta), \psi(x^\star))\right].$$

This function has the property that it is both C^2-smooth and has $\frac{b}{\sigma^7}$-Lipschitz Hessian, since it is the convolution of a b-bounded function ψ with a Gaussian of variance σ^2.

This leads us to Definition 2.3, which says that $x^\star$ is an approximate local minimum "with smoothing σ" for a discontinuous function ψ, if $x^\star$ is an approximate local minimum of the smooth function $\mathbb{E}_{\zeta\sim N(0,I_d)}[\min(\psi(x+\sigma\zeta), \psi(x^\star))]$.

6 THE FULL ALGORITHM

In this section we present the full algorithm for computing a greedy adversarial equilibrium (Algorithm 2), as well as an algorithm for computing a greedy path (Algorithm 1) which Algorithm 2 uses as a subroutine.

Algorithm 1: Computing a Greedy Path

Input: Oracles for the value of a function $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, the gradient $\nabla_y f$ and the Hessian $\nabla_y^2 f$
Input: $\mathsf{x}, \mathsf{y}^0, \varepsilon'$
Hyperparameters: $\delta, \mu_1, \mu_3, \mu_4$

```
Set ℓ ← 0, Stopy ← False
while Stopy = False do
    if ‖∇_y f(x, y^ℓ)‖ > ε' then
        Set y^{ℓ+1} ← y^ℓ + μ_1 ∇_y f(x, y^ℓ)
        Set ℓ ← ℓ + 1
    else
        Compute an eigenvalue-eigenvector pair (λ, v) of
          ∇²_y f(x, y^ℓ), s.t. λ ≥ λ_max(∇²_y f(x, y^ℓ)) − μ_4
        if λ > sqrt(L ε') then
            Set a = sign(∇_y f(x, y^ℓ)^⊤ v)
            Set y^{ℓ+1} ← y^ℓ + μ_3 a v
            Set ℓ ← ℓ + 1
        else
            Set Stopy = True
return y_LocalMax ← y^ℓ
```

7 PROOF OF THEOREM 3.1

7.1 Setting Constants and Notation

Since f is a b-bounded C^2-smooth function with L-Lipschitz Hessian, it is also L_1-Lipschitz with $L_1 \le 4b^{2/3}L^{1/3}$ and has L_2-Lipschitz gradient with $L_2 \le 2b^{1/3}L^{2/3}$. From now on, we set $L_1 = 4b^{2/3}L^{1/3}$ and $L_2 = 2b^{1/3}L^{2/3}$. Without loss of generality, we may assume that $b \ge 1$ (since our goal is to prove that the number of gradient evaluations is polynomial in $\frac{1}{\varepsilon}, d, b, L, \frac{1}{\sigma}$). We set the following hyperparameters and constants. While many of these parameters do not appear in the proof summaries presented here, they do appear in the full proofs which are presented in the arXiv version of our paper [26]. We include them here for completeness.

(1) $\omega := 10^{-3}$,
(2) $\gamma_1 := \frac{\epsilon^{2.1}\sigma^{16.6}}{10^4(1+b^{3.1})d^{0.6}\log(bd\sigma\epsilon)}$,
(3) $\delta := \frac{\gamma_1^2}{8b^2}$,
(4) $\mu_1 := \delta\frac{1}{L_2(L_1+1)}$,
(5) $\mu_3 := \frac{1}{7}\min\left(\frac{\delta\sqrt{\epsilon}}{\sqrt{L}}, \frac{\epsilon}{\sqrt{L}}\right)$,
(6) $\mu_4 := \frac{1}{7}\sqrt{\delta L\epsilon}$,
(7) $\eta := \frac{\sigma^9}{b^6d^2\left(1+10\frac{bd}{\sigma^{12}\epsilon^2}\right)\mathsf{c}\log^9(bd\sqrt{\sigma\epsilon})}$,
(8) $\mathcal{I}_2 := \frac{\mathsf{c}\log(bd\sqrt{\sigma\epsilon})}{\eta\sqrt{\epsilon}}$,
(9) $\mathcal{I}_3 := \frac{30b}{\gamma_1}$,
(10) $\mathcal{I}_4 := 6\log\left(\frac{2b}{\gamma_1\omega}\right)$,
(11) $\alpha := \eta\mathsf{c}\log(bd\sqrt{\sigma\epsilon})\sqrt{1+b^2d^2\sigma^{-2}}$,

where c is a large enough universal constant.

Algorithm 2: Computing a Greedy Adversarial Equilibrium

Input: Oracle for a function $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, and oracles for the gradient $\nabla_y f$ and Hessian $\nabla^2_y f$. $\sigma, \varepsilon > 0$

Hyperparamters: $\eta, \gamma_1, \mathcal{I}_2, \delta, \mathcal{I}_3, \mathcal{I}_4, \alpha, \varepsilon_0$

Initialize $(x_0, y_0) \leftarrow (0, 0)$
Set $x_1 \leftarrow x_0$.
Run Algorithm 1 with inputs $\mathsf{x} \leftarrow x_1$, $\mathsf{y}^0 \leftarrow y_0$, $\varepsilon \leftarrow \varepsilon$, $\varepsilon' \leftarrow \varepsilon_0(1+\delta)$.
Set $y_1 \leftarrow \mathsf{y}_{\text{LocalMax}}$ to be the output $\mathsf{y}_{\text{LocalMax}}$ of Alg. 1.
Set $h^0 \leftarrow f(x_1, y_1)$
Set $\texttt{Stop} \leftarrow \texttt{False}$, $i \leftarrow 0$
while $\texttt{Stop} = \texttt{False}$ **do**
  Set $i \leftarrow i+1$, $\texttt{NoProgress} \leftarrow \texttt{True}$,
  Set $\varepsilon_i \leftarrow \varepsilon_{i-1}(1+\delta)^2$
  Set $X_0 \leftarrow x_i$
  for $j = 1$ *to* $\mathcal{I}_3$ **do**
    if $\texttt{NoProgress} = \texttt{True}$ **then**
      Set $\zeta_{ij} \sim N(0, I_d)$
      Run Algorithm 1 with inputs $\mathsf{x} \leftarrow x_i + \sigma\zeta_{ij}$, $\mathsf{y}^0 \leftarrow y_i$, $\varepsilon \leftarrow \varepsilon$, and $\varepsilon' \leftarrow \varepsilon_i(1+\delta)$.
      Set $\mathcal{Y} \leftarrow \mathsf{y}_{\text{LocalMax}}$ to be the output $\mathsf{y}_{\text{LocalMax}}$ of Algorithm 1.
      if $f(x_i + \sigma\zeta_{ij}, \mathcal{Y}) \le f(x_i, y_i) - \gamma_1$ **then**
        Set $x_{i+1} \leftarrow x_i + \sigma\zeta_{ij}$
        Set $y_{i+1} \leftarrow \mathcal{Y}$,
        Set $h^0 \leftarrow f(x_i, y_i)$
        Set $i \leftarrow i+1$, $\texttt{NoProgress} \leftarrow \texttt{False}$,
  for $j = 1$ *to* $\mathcal{I}_4$ **do**
    if $\texttt{NoProgress} = \texttt{True}$ **then**
      for $k = 1$ *to* $\mathcal{I}_2$ **do**
        Set $u \sim N(0, I_d)$
        Run Algorithm 1 with inputs $\mathsf{x} \leftarrow X_{k-1} + \sigma u$, $\mathsf{y}^0 \leftarrow y_i$, $\varepsilon \leftarrow \varepsilon$, $\varepsilon' \leftarrow \varepsilon_i(1+\delta)$.
        Set $\mathcal{Y} \leftarrow \mathsf{y}_{\text{LocalMax}}$ to be the output $\mathsf{y}_{\text{LocalMax}}$ of Algorithm 1.
        Set $h^k = \min(f(X_{k-1} + \sigma u, \mathcal{Y}), f(x_i, y_i))$
        Set $\Gamma_k = (h^k - h^{k-1})\frac{1}{\sigma} u$
        Set $\xi \sim N(0, I_d)$,
        Set $X_k \leftarrow X_{k-1} - \eta\Gamma_k + \alpha\xi$
      Run Algorithm 1 with inputs $\mathsf{x} \leftarrow X_k$, $\mathsf{y}^0 \leftarrow y_i$, $\varepsilon \leftarrow \varepsilon$, and $\varepsilon' \leftarrow \varepsilon_i(1+\delta)$.
      Set $\mathcal{Y} \leftarrow \mathsf{y}_{\text{LocalMax}}$ to be the output $\mathsf{y}_{\text{LocalMax}}$ of Algorithm 1.
      if $f(X_k, \mathcal{Y}) \le f(X_0, y_i) - \gamma_1$ **then**
        Set $x_{i+1} \leftarrow X_k$
        Set $y_{i+1} \leftarrow \mathcal{Y}$,
        Set $h^0 \leftarrow f(x_i, y_i)$,
        Set $i \leftarrow i+1$, $\texttt{NoProgress} \leftarrow \texttt{False}$
  if $\texttt{NoProgress} = \texttt{True}$ **then**
    Set $\texttt{Stop} = \texttt{True}$
return $i^\star \leftarrow i$, $\varepsilon^\star \leftarrow \varepsilon_{i^\star}$, *and* $(x^\star, y^\star) \leftarrow (x_{i^\star}, y_{i^\star})$

In particular, we have set $\delta = \frac{1}{4i_{\max}^2}$, where $i_{\max} := \frac{2b}{\gamma_1}$ is an upper bound on the number of iterations of the While loop in Algorithm 2. This ensures that $(1+\delta)^i \le 2$ for all $i \in [i_{\max}]$.

In the following sections we let (x_i, y_i) denote the points (x_i, y_i) at each iteration i of the While loop in Algorithm 2, and we set $\varepsilon_i := \varepsilon_0(1+\delta)^{2i}$ for all $i \in \mathbb{N}$.

For any $\varepsilon^\circ > 0$, and any $(x, y) \in \mathbb{R}^d \times \mathbb{R}^d$, we define

$$h_{\varepsilon^\circ}(x, y) := f(x, \mathcal{Y}), \tag{20}$$

where $\mathcal{Y} \leftarrow \mathsf{y}_{\text{LocalMax}}$ is the output of Algorithm 1 with inputs $\mathsf{x} \leftarrow x$, $\mathsf{y}^0 \leftarrow y$, and $\varepsilon' \leftarrow (1+\delta)\varepsilon^\circ$.

7.2 Proof Outline

The proof of Theorem 3.1 has three main components:

(1) We start by showing that Algorithm 2 halts after $i^\star = O\left(\frac{b}{\gamma_1}\right)$ iterations, which allows us to bound the number of oracle calls until Algorithm 2 halts (Lemma 7.1).
(2) We then show that the point $(x^\star, y^\star)$ returned by Algorithm 2 is an $(\varepsilon^\star, \sigma)$-greedy adversarial equilibrium for f, for some
$$\varepsilon^\star = \varepsilon_{i^\star} = \varepsilon_0 \le (1+\delta)^{O(i^\star)} \le \varepsilon.$$
Towards this end we first show that $y^\star$ is an $(\varepsilon^\star, \sqrt{L\varepsilon^\star})$-approximate local maximum of $f(x^\star, \cdot)$ (in the sense of the definition in (2)) and therefore satisfies Inequalities (6) of Definition 2.5.
(3) Next, we show that $x^\star$ is an $(\varepsilon^\star, \sigma)$-approximate local minimum of $g_\varepsilon(\cdot, y^\star)$) (in the sense of Definition 2.3), and therefore satisfies Inequalities (7) of Definition 2.5 (see the proof in Section 7.6). Towards this end, we prove (in Lemma 7.3) that to guarantee $x^\star$ is an $(\varepsilon^\star, \sigma)$-approximate local minimum of $g_\varepsilon(\cdot, y^\star)$, it is sufficient to show that $x^\star$ is an $(\varepsilon^\star, \sigma)$-approximate local minimum of $h_\varepsilon(\cdot, y^\star)$ (Propositions 7.7 and 7.8), provided that we can also show that h_ε and g_ε satisfy a number of conditions. Namely, these conditions are:
  (a) $h_\varepsilon(x, y) \le g_\varepsilon(x, y)$ for all $(x, y) \in \mathbb{R}^d \times \mathbb{R}^d$ (Proposition 7.4).
  (b) $h_\varepsilon(x^\star, y^\star) = g_\varepsilon(x^\star, y^\star)$ (Proposition 7.5).
  (c) A stochastic gradient for a smoothed version of h_ε (defined in Equation (23)) has a very small expected magnitude (Proposition 7.6).

In the remainder of this section, we present the main lemmas and propositions we use to prove Theorem 3.1, along with short summaries of their proofs; for the full proofs see the arXiv version of our paper [26]. We conclude the proof of Theorem 3.1 in Section 7.6.

7.3 Gradient, Function, and Hessian Evaluations

Lemma 7.1 (Bounding the Number of Gradient, Hessian, and Function Evaluations). *Algorithm 2 terminates after at most $i^\star = O(\frac{b}{\gamma_1})$ iterations, and thus takes at most $O\left(\frac{b}{\gamma_1} \times (\mathcal{I}_2\mathcal{I}_4 + \mathcal{I}_3) \times \frac{b}{\mu_1\mu_3^2 L}\right)$ gradient, Hessian, and function evaluations.*

proof (summary). First, we show that Algorithm 1 terminates after $O\left(\frac{b}{\mu_1\mu_3^2 L}\right)$ oracle evaluations. Next we show the Algorithm 2

terminates after at most $O\left(\frac{b}{\gamma_1}\right)$ iterations of its "While" loop. Since Algorithm 1 is called $\mathcal{I}_3 + \mathcal{I}_2\mathcal{I}_4$ times at each iteration of Algorithm 2, running Algorithm 1 contributes $O\left(\frac{b}{\gamma_1} \times (\mathcal{I}_3 + \mathcal{I}_2\mathcal{I}_4) \times \frac{b}{\mu_1\mu_3^2 L}\right)$ oracle calls to the cost of Algorithm 2. Since the other parts of the While loop make at most $O(\mathcal{I}_3 + \mathcal{I}_2\mathcal{I}_4)$ function evaluations, they contribute no more that $O(\frac{b}{\gamma_1} \times (\mathcal{I}_3 + \mathcal{I}_2\mathcal{I}_4))$ function evaluations to the cost of Algorithm 2. Therefore, Algorithm 2 terminates after at most $O\left(\frac{b}{\gamma_1} \times (\mathcal{I}_3 + \mathcal{I}_2\mathcal{I}_4) \times \frac{b}{\mu_1\mu_3^2 L}\right)$ oracle calls. □

7.4 $y^\star$ an Approximate Local Max of $f(x^\star, \cdot)$

Lemma 7.2 (Approximate Local Maximum in y). *The output* $\mathsf{y}_{\text{LocalMax}}$ *of Algorithm 1 with inputs* $\mathsf{x}, \mathsf{y}^0, \varepsilon'$ *satisfies*

$$\|\nabla_y f(\mathsf{x}, \mathsf{y}_{\text{LocalMax}})\| \le \varepsilon'$$

and

$$\lambda_{\max}(\nabla_y^2 f(\mathsf{x}, \mathsf{y}_{\text{LocalMax}})) \le \sqrt{L\varepsilon'}.$$

In particular, this implies that the output $(x^\star, y^\star)$ *of Algorithm 2 satisfies*

$$\|\nabla_y f(x^\star, y^\star)\| \le \varepsilon_{i^\star}, \tag{21}$$

and

$$\lambda_{\max}(\nabla_y^2 f(x^\star, y^\star)) \le \sqrt{L\varepsilon_{i^\star}}. \tag{22}$$

proof (summary). The proof uses the first-order stopping condition (Line 3) and second-order stopping condition (Line 8) for Algorithm 1 to show that the point reached by that Algorithm once it stops is an $(\varepsilon', \sqrt{L\varepsilon'})$-approximate local maximum of $f(\mathsf{x}, \cdot)$ (in the sense of Definition 2.3). □

7.5 $x^\star$ an Approximate Local Min of $g_\varepsilon(\cdot, y^\star)$

We start by defining some functions which will be useful in showing that $x^\star$ is an approximate local minimum of $g_\varepsilon(\cdot, y^\star)$.
For any $\hat{x} \in \mathbb{R}^d$, $\varepsilon^\circ > 0$, let

$$\mathsf{g}_{\varepsilon^\circ}^{\hat{x}}(x, y) := \min(g_{\varepsilon^\circ}(x, y), g_{\varepsilon^\circ}(\hat{x}, y)),$$
$$\mathsf{h}_{\varepsilon^\circ}^{\hat{x}}(x, y) := \min(h_{\varepsilon^\circ}(x, y), h_{\varepsilon^\circ}(\hat{x}, y)),$$

and

$$\mathfrak{g}_{\varepsilon^\circ,\sigma}^{\hat{x}}(x, y) := \mathbb{E}_{\zeta\sim N(0,I_d)}\left[\mathsf{g}_{\varepsilon^\circ}^{\hat{x}}(x + \sigma\zeta, y)\right],$$
$$\mathfrak{h}_{\varepsilon^\circ,\sigma}^{\hat{x}}(x, y) := \mathbb{E}_{\zeta\sim N(0,I_d)}\left[\mathsf{h}_{\varepsilon^\circ}^{\hat{x}}(x + \sigma\zeta, y)\right].$$

Finally, for any $\hat{x} \in \mathbb{R}^d$, $\varepsilon^\circ > 0$, define the stochastic gradient

$$\mathcal{H}_{\varepsilon^\circ}^{\hat{x}}(x, y) := \frac{\zeta}{\sigma}(\mathsf{h}_{\varepsilon^\circ}^{\hat{x}}(x + \sigma\zeta, y) - \mathsf{h}_{\varepsilon^\circ}^{\hat{x}}(x, y)), \tag{23}$$

where $\zeta \sim N(0, I_d)$.

7.5.1 Showing That g_ε and h_ε Have Shared Approximate Local Minima. The following lemma allows us to guarantee that if we can show that our algorithm returns a point $(x^\star, y^\star)$ where $x^\star$ is an approximate local minimum for $h_\varepsilon(\cdot, y^\star)$ for some $y^\star \in \mathbb{R}^d$, then $x^\star$ is also an approximate local minimum for the greedy max function $g_\varepsilon(\cdot, y^\star)$, provided that we can also show that h_ε, g_ε, $x^\star$ and $y^\star$ satisfy certain conditions.

Lemma 7.3 (Shared Local Minima of g_ε and h_ε). *Consider any* $\varepsilon > 0$. *Suppose that* $\sigma \le \frac{1}{\sqrt{\varepsilon d}}$ *and that for some point* $(x^\star, y^\star) \in \mathbb{R}^d \times \mathbb{R}^d$ *we have*

$$h_\varepsilon(x, y) \le g_\varepsilon(x, y) \qquad \forall x, y \in \mathbb{R}^d \qquad \textit{(lower bound)}, \tag{24}$$
$$h_\varepsilon(x^\star, y^\star) = g_\varepsilon(x^\star, y^\star) \qquad \textit{(fixed-point property)}, \tag{25}$$
$$\mathbb{E}[\|\mathcal{H}_\varepsilon^{x^\star}(x^\star, y^\star)\|] \le \frac{1}{8000}\frac{\sigma^{14}\varepsilon^{1.5}}{b^2} \quad \textit{(low SG)}, \tag{26}$$
$$\|\nabla_x \mathfrak{h}_{\varepsilon,\sigma}^{x^\star}(x^\star, y^\star)\| \le \frac{\varepsilon^2\sigma^7}{8000b} \quad \textit{(1st-order stationarity for } \mathfrak{h}), \tag{27}$$
$$\lambda_{\min}(\nabla_x^2 \mathfrak{h}_{\varepsilon,\sigma}^{x^\star}(x^\star, y^\star)) \ge -\frac{\sqrt{\varepsilon}}{5} \quad \textit{(2nd-order stationarity)}. \tag{28}$$

Then

$$\|\nabla_x \mathfrak{g}_{\varepsilon,\sigma}^{x^\star}(x^\star, y^\star)\| \le \varepsilon \quad \textit{and} \tag{29}$$
$$\lambda_{\min}(\nabla_x^2 \mathfrak{g}_{\varepsilon,\sigma}^{x^\star}(x^\star, y^\star)) \ge -\sqrt{\varepsilon}. \tag{30}$$

proof (summary). *Proving Inequality* (29)*:* First, we use conditions (24), (25), and (26) to show that

$$\begin{aligned}
&\mathbb{E}[|g_\varepsilon(x^\star, y^\star) - (\min(g_\varepsilon(x^\star + \sigma\zeta, y^\star), g_\varepsilon(x^\star, y^\star))|] \qquad (31)\\
&\le \mathbb{E}[|h_\varepsilon(x^\star, y^\star) - \min(h_\varepsilon(x^\star + \sigma\zeta, y^\star), h_\varepsilon(x^\star, y^\star)|]\\
&\le \frac{\sigma}{\sqrt{d}}\mathbb{E}\left[\left\|\frac{\zeta}{\sigma}(\min(h_\varepsilon(x^\star + \sigma\zeta, y^\star), h_\varepsilon(x^\star, y^\star)) - h_\varepsilon(x^\star, y^\star)\right\|\right]\\
&\le O\left(\frac{\sigma}{\sqrt{d}} \times \frac{\sigma^{14}\varepsilon^{1.5}}{b^2}\right).
\end{aligned}$$

The first inequality holds by conditions (24) and (25), since the fact that h_ε is a lower bound for g_ε allows us to replace $g_\varepsilon(x^\star + \sigma\zeta, y^\star)$ on the LHS with $h_\varepsilon(x^\star + \sigma\zeta, y^\star)$ on the RHS. And (25) allows us to replace $g_\varepsilon(x^\star, y^\star)$ with $h_\varepsilon(x^\star, y^\star)$. The second inequality holds by a concentration bound for Gaussians. The last inequality follows directly from the condition (26), since the quantity inside the expectation is just the magnitude of the stochastic gradient $\mathcal{H}_\varepsilon^{x^\star}$ at the point $(x^\star, y^\star)$. We then use (31) to show that $\|\nabla_x \mathfrak{g}_{\varepsilon,\sigma}^{x^\star}(x^\star, y^\star)\| < \varepsilon$, which proves Inequality (29).

Proving Inequality (30) *:* We show (30) via a contradiction argument. Towards this end, we use Equation (29) to show that, if it were true that (30) did not hold, that is, if

$$\lambda_{\min}(\nabla_x^2 \mathfrak{g}_{\varepsilon,\sigma}^{x^\star}(x^\star, y^\star)) < -\sqrt{\varepsilon},$$

then there would be some vector v and a number $t > 0$ such that

$$\mathfrak{g}_{\varepsilon,\sigma}^{x^\star}(x^\star + tv, y^\star) - \mathfrak{g}_{\varepsilon,\sigma}^{x^\star}(x^\star, y^\star) \le -\frac{9/2}{4000}\frac{\sigma^{14}\varepsilon^{1.5}}{b^2}. \tag{32}$$

Moreover, we also show that (28) implies that

$$\mathfrak{h}_{\varepsilon,\sigma}^{x^\star}(x^\star + tv, y^\star) - \mathfrak{h}_{\varepsilon,\sigma}^{x^\star}(x^\star, y^\star) \ge -\frac{1}{4000}\frac{\sigma^{14}\varepsilon^{1.5}}{b^2}. \tag{33}$$

Next, we show that, since $g_\varepsilon(x^\star, y^\star) = h_\varepsilon(x^\star, y^\star)$, we have roughly speaking that $\mathfrak{g}_{\varepsilon,\sigma}^{x^\star}(x^\star, y^\star) \approx \mathfrak{h}_{\varepsilon,\sigma}^{x^\star}(x^\star, y^\star)$. This fact, together with (32) and (33) implies that

$$\mathfrak{g}_{\varepsilon,\sigma}^{x^\star}(x^\star + tv, y^\star) < \mathfrak{h}_{\varepsilon,\sigma}^{x^\star}(x^\star + tv, y^\star). \tag{34}$$

On the other hand, we use the fact that $h_\varepsilon(x, y) < g_\varepsilon(x, y)$ for all $x, y \in \mathbb{R}^d \times \mathbb{R}^d$ to show that

$$\mathfrak{h}_\varepsilon^{x^\star}(x, y^\star) \le \mathfrak{g}_\varepsilon^{x^\star}(x, y^\star) \qquad \forall x \in \mathbb{R}^d. \tag{35}$$

Since (35) contradicts (34), we have that (30) holds by contradiction. □

7.5.2 *Properties of g_ε and h_ε.*

Proposition 7.4 (Greedy Max Lower Bound).

$$h_{\varepsilon^\circ}(x,y) \le g_{\varepsilon^\circ}(x,y), \qquad \forall x,y \in \mathbb{R}^d, \forall \varepsilon^\circ > 0. \tag{36}$$

proof (summary). First we show that the path traced by Algorithm 1 with inputs $\mathsf{x} \leftarrow x$, $\mathsf{y}^0 \leftarrow y$, and $\varepsilon' \leftarrow (1+\delta)\varepsilon^\circ$ is an ε°-greedy path with initial point y. Since $h_{\varepsilon^\circ}(x,y) = f(x,y')$, where y' is the output of Algorithm 1 with the above-mentioned inputs, and $g_{\varepsilon^\circ}(x,y)$ is the supremum of the value of f at the endpoints of all ε°-greedy paths which seek to maximize $f(x,\cdot)$ from the starting point y, we must have that

$$h_{\varepsilon^\circ}(x,y) \le g_{\varepsilon^\circ}(x,y), \qquad \forall x,y \in \mathbb{R}^d, \ \ \forall \varepsilon^\circ > 0. \qquad \square$$

Proposition 7.5 (Fixed Point Property). *Recall that $\varepsilon_i = \varepsilon_0(1+\delta)^{2i}$, and consider the points (x_i, y_i) generated at each iteration i of the While loop in Algorithm 2. Then*

$$h_{\varepsilon_i}(x_i,y_i) = g_{\varepsilon_i}(x_i,y_i) = f(x_i,y_i), \qquad \forall i \in \mathbb{N}. \tag{37}$$

Proof. By Lemma 7.2 we have that

$$\|\nabla_y f(x_i,y_i)\| \le \varepsilon_i \quad \text{and} \quad \lambda_{\max}(\nabla_y^2 f(x_i,y_i)) \le \sqrt{L\varepsilon_i}, \tag{38}$$

since y_i is generated by Algorithm 1 with inputs $\mathsf{x} \leftarrow x_i, \mathsf{y}^0 \leftarrow y_{i-1}$, and $\varepsilon' \le \varepsilon_{i-1}(1+\delta)$.

Inequality (38) implies that there is only one greedy path (with parameter ε_i) which seeks to maximize $f(x_i,\cdot)$ with starting point y_i, namely, the path consisting of the single point y_i. Therefore,

$$h_{\varepsilon_i}(x_i,y_i) = g_{\varepsilon_i}(x_i,y_i) = f(x_i,y_i), \qquad \forall i \in \mathbb{N}. \qquad \square$$

Proposition 7.6 (Low-Magnitude Stochastic Gradient). *With probability at least $1-\omega$ the stochastic gradient $\mathcal{H}_{\varepsilon_{i^\star}}$ at the outputs $(x^\star, y^\star)$ of Algorithm 2 satisfies*

$$\mathbb{E}\left[\left\|\mathcal{H}^{x^\star}_{\varepsilon_{i^\star}}(x^\star,y^\star)\right\| \,\middle|\, (x^\star,y^\star)\right] \le 10 b\gamma_1 \sqrt{d}\sigma^{-1}\log\frac{2}{\gamma_1}.$$

proof (summary). First we note that whenever Algorithm 2 outputs a point $(x_{i^\star}, y_{i^\star}) = (x^\star, y^\star)$, the stopping condition in Line 16 implies that we have that

$$f(x^\star + \sigma\zeta, y') > f(x^\star, y^\star) - \gamma_1, \tag{39}$$

with high probability, where $\zeta \sim N(0, I_d)$ and y' is the output of Algorithm 1 with inputs $x^\star, y^\star, \varepsilon_{i^\star}(1+\delta)$.

We then use (39) together with the fact that $f(x^\star+\sigma\zeta, y') = h_{\varepsilon_{i^\star}}(x^\star+\sigma\zeta, y^\star)$ and that $f(x^\star,y^\star) = h_{\varepsilon_{i^\star}}(x^\star,y^\star)$ to show that

$$\begin{aligned}
&\mathbb{E}\left[\left\|\mathcal{H}^{x^\star}_{\varepsilon_{i^\star}}(x^\star,y^\star)\right\| \,|(x^\star,y^\star)\right] \\
&= \mathbb{E}\left[\left\|\frac{\zeta}{\sigma}(\min(h_{\varepsilon_{i^\star}}(x^\star+\sigma\zeta,y^\star), h_{\varepsilon_{i^\star}}(x^\star,y^\star)) \right.\right. \\
&\qquad\qquad \left.\left. - h_{\varepsilon_{i^\star}}(x^\star,y^\star))\right\| \,\middle|\, (x^\star,y^\star)\right] \\
&\le 10 b\gamma_1\sqrt{d}\sigma^{-1}\log(2\gamma_1^{-1}). \qquad \square
\end{aligned}$$

7.5.3 *Showing $x^\star$ is an Approximate Local Min of $h_\varepsilon(\cdot, y^\star)$.*

Proposition 7.7 (First-Order Stationary Condition).

$$\mathbb{P}\left(\left\|\nabla_x \mathfrak{h}^{x^\star}_{\varepsilon_{i^\star},\sigma}(x^\star,y^\star)\right\| > 10 b\gamma_1 \frac{\sqrt{d}}{\sigma}\log\frac{2b}{\gamma_1}\right) \le \omega. \tag{40}$$

proof (summary). We use standard concentration bounds for Gaussian random vectors, to show that the stochastic gradient $\mathcal{H}^{x^\star}_{\varepsilon_{i^\star}}(x^\star,y^\star)$ is very close to the gradient $\nabla_x \mathfrak{h}^{x^\star}_{\varepsilon_{i^\star},\sigma}(x^\star,y^\star)$ of $\mathfrak{h}$ with high probability. This fact, together with Proposition 7.6, implies Inequality (40). □

Proposition 7.8 (Second-Order Stationary Condition). *With probability at least $1-2\omega$, we have that*

$$\lambda_{\min}\left(\nabla_x^2 \mathfrak{h}^{x^\star}_{\varepsilon_{i^\star},\sigma}(x^\star,y^\star)\right) \ge -\frac{1}{5}\sqrt{\varepsilon_{i^\star}}. \tag{41}$$

proof (summary). First we show that Γ_k in Algorithm 2 is a stochastic gradient for $\mathfrak{h}^{x_i}_{\varepsilon_i,\sigma}(X^j_{k-1}, y_i)$, that is,

$$\mathbb{E}[\Gamma_k] = \nabla_x \mathfrak{h}^{x_i}_{\varepsilon_i,\sigma}\left(X^j_{k-1}, y_i\right), \tag{42}$$

and also show a concentration property for this stochastic gradient,

$$\mathbb{P}\left(\left\|\Gamma_k - \nabla_x \mathfrak{h}^{x_i}_{\varepsilon_i,\sigma}\left(X^j_{k-1},y_i\right)\right\| \ge t\right) \le 2\exp\left(-\frac{t^2}{8\left(\frac{2b\sqrt{d}}{\sigma}\right)^2}\right) \quad \forall t \ge 0. \tag{43}$$

Since Γ_k is a stochastic gradient with concentration properties for a smooth function, we can apply results from [19] which, roughly speaking, say that stochastic gradient descent with added Gaussian noise can escape saddle points in polynomial time.

More specifically, Lemma 25 of [19], together with Equations (42) and (43), imply that, if Algorithm 2 reaches a point $x^\star$ where it can no longer decrease $\mathfrak{h}^{x^\star}_{\varepsilon_{i^\star},\sigma}(\cdot,y^\star)$, then w.h.p. $x^\star$ is an $(\varepsilon_{i^\star}, \sqrt{\varepsilon_{i^\star}})$-approximate local minimum for $\mathfrak{h}^{x^\star}_{\varepsilon_{i^\star},\sigma}(\cdot,y^\star)$ (in the sense of the definition in (2)). This fact, together with Proposition 7.7, implies Inequality (41). □

7.6 Concluding the Proof of Main Theorem

Proof of Theorem 3.1. **Showing Convergence, and Bounding the Number of Oracle Calls.**
By Lemma 7.1, we have that Algorithm 2 terminates and outputs a point $(x^\star, y^\star) \in \mathbb{R}^d$ after

$$O\left(\frac{b}{\gamma_1} \times (\mathcal{I}_2\mathcal{I}_4 + \mathcal{I}_3) \times \frac{b}{\mu_1\mu_3^2 L}\right) = \text{poly}\left(\frac{1}{\varepsilon}, d, b, L, \frac{1}{\sigma}\right)$$

gradient, function, and Hessian evaluations. In particular, if $b, L \ge 1$ and if $\sigma, \varepsilon \le 1$, the number of gradient, function, and Hessian evaluations can be simplified to $O\left(\frac{d^8 L^2 b^{37}}{\epsilon^{19}\sigma^{132}}\right)$.

Showing $x^\star$ Is an Approximate Local Minimum for Greedy Max Function.
By Proposition 7.4, we have that

$$h_{\varepsilon_{i^\star}}(x,y) \le g_{\varepsilon_{i^\star}}(x,y) \qquad \forall x,y \in \mathbb{R}^d. \tag{44}$$

By Proposition 7.5, we have that

$$h_{\varepsilon_{i^\star}}(x^\star,y^\star) = g_{\varepsilon_i}(x^\star,y^\star) = f(x^\star,y^\star). \tag{45}$$

By Proposition 7.6 we have, with probability at least $1-\omega$, that

$$\mathrm{E}\left[\left\|\mathcal{H}^{x^\star}_{\varepsilon_{i^\star}}(x^\star,y^\star)\right\| \,|(x^\star,y^\star)\right] \le \frac{1}{8000}\frac{\sigma^{14}\varepsilon_{i^\star}^{1.5}}{b^2}, \tag{46}$$

since $\gamma_1 = \frac{\varepsilon^{2.1}\sigma^{16.6}}{10^4(1+b^{3.1})d^{0.6}\log(bd\sigma\varepsilon)}$.
By Proposition 7.7, with probability at least $1-\omega$, we have that

$$\left\|\nabla_x \mathfrak{h}^{x^\star}_{\varepsilon_{i^\star},\sigma}(x^\star,y^\star)\right\| \le \frac{\varepsilon_{i^\star}^2}{8000}\frac{\sigma^7}{b}, \tag{47}$$

since $\gamma_1 = \frac{\varepsilon^{2.1}\sigma^{16.6}}{10^4(1+b^{3.1})d^{0.6}\log(bd\sigma\varepsilon)}$.
By Proposition 7.8, with probability at least $1-2\omega$ we have

$$\lambda_{\min}(\nabla_x^2 \mathfrak{h}^{x^\star}_{\varepsilon_{i^\star},\sigma}(x^\star,y^\star)) \ge -\frac{1}{5}\sqrt{\varepsilon_{i^\star}}. \tag{48}$$

Thus, by Lemma 7.3, (44)-(48) imply that, w.p. at least $1-4\omega$,

$$\left\|\nabla_x \mathfrak{g}^{x^\star}_{\varepsilon_{i^\star},\sigma}(x^\star,y^\star)\right\| \le \varepsilon_{i^\star} \quad \text{and} \quad \lambda_{\min}(\nabla_x^2 \mathfrak{g}^{x^\star}_{\varepsilon_{i^\star},\sigma}(x^\star,y^\star)) \ge -\sqrt{\varepsilon_{i^\star}}. \tag{49}$$

Showing That $y^\star$ Is an Approximate Local Maximum for $f(x^\star,\cdot)$.
We also have, by Lemma 7.2 that

$$\|\nabla_y f(x^\star,y^\star)\| \le \varepsilon_{i^\star} \quad \text{and} \quad \lambda_{\max}(\nabla_y^2 f(x^\star,y^\star)) \le \sqrt{L\varepsilon_{i^\star}}. \tag{50}$$

Showing That $(x^\star,y^\star)$ Is a Greedy Adversarial Equilibrium for f.
Inequalities (49) and (50) together imply that, with probability at least $1-4\omega$, the point $(x^\star,y^\star)$ is an $(\varepsilon^\star,\sigma)$-greedy adversarial equilibrium, where $\varepsilon^\star = \varepsilon_{i^\star} \le \varepsilon$. □

ACKNOWLEDGMENTS

This research was supported in part by NSF CCF-1908347.

REFERENCES

[1] Jacob Abernethy, Kevin A Lai, and Andre Wibisono. 2019. Last-iterate convergence rates for min-max optimization. *arXiv preprint arXiv:1906.02027* (2019).
[2] Leonard Adolphs, Hadi Daneshmand, Aurelien Lucchi, and Thomas Hofmann. 2019. Local Saddle Point Optimization: A Curvature Exploitation Approach. In *The 22nd International Conference on Artificial Intelligence and Statistics*. 486–495.
[3] Naman Agarwal, Zeyuan Allen-Zhu, Brian Bullins, Elad Hazan, and Tengyu Ma. 2017. Finding approximate local minima faster than gradient descent. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. 1195–1199.
[4] Zeyuan Allen-Zhu. 2018. Natasha 2: Faster non-convex optimization than SGD. In *Advances in Neural Information Processing Systems*. 2675–2686.
[5] Yossi Arjevani, Yair Carmon, John C Duchi, Dylan J Foster, Ayush Sekhari, and Karthik Sridharan. 2020. Second-order information in non-convex stochastic optimization: Power and limitations. In *Conference on Learning Theory*. PMLR, 242–299.
[6] Avrim Blum and Ronald L Rivest. 1989. Training a 3-node neural network is NP-complete. In *Advances in Neural Information Processing Systems*. 494–501.
[7] Ran Canetti. 2000. Security and composition of multiparty cryptographic protocols. *Journal of CRYPTOLOGY* 13, 1 (2000), 143–202.
[8] Yair Carmon, John C Duchi, Oliver Hinder, and Aaron Sidford. 2018. Accelerated methods for nonconvex optimization. *SIAM Journal on Optimization* 28, 2 (2018), 1751–1772.
[9] Constantinos Daskalakis and Ioannis Panageas. 2018. The limit points of (optimistic) gradient descent in min-max optimization. In *Advances in Neural Information Processing Systems*. 9236–9246.
[10] Constantinos Daskalakis, Stratis Skoulakis, and Manolis Zampetakis. 2021. The Complexity of Constrained Min-Max Optimization. In *ACM SIGACT Symposium on Theory of Computing (STOC 2021)*.
[11] C Fang, CJ Li, Z Lin, and T Zhang. 2018. Near-optimal non-convex optimization via stochastic path integrated differential estimator. *Advances in Neural Information Processing Systems* 31 (2018), 689.
[12] Abraham D Flaxman, Adam Tauman Kalai, and H Brendan McMahan. 2005. Online convex optimization in the bandit setting: gradient descent without a gradient. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 385–394.
[13] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. 2015. Escaping from saddle points—online stochastic gradient for tensor decomposition. In *Conference on Learning Theory*. 797–842.
[14] Gauthier Gidel, Hugo Berard, Gaëtan Vignoud, Pascal Vincent, and Simon Lacoste-Julien. 2019. A variational inequality perspective on generative adversarial networks. In *International Conference on Learning Representations (ICLR 2019)*.
[15] Shafi Goldwasser and Silvio Micali. 1984. Probabilistic encryption. *Journal of computer and system sciences* 28, 2 (1984), 270–299.
[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*. 2672–2680.
[17] V. Guruswami and A. Smith. 2010. Codes for Computationally Simple Channels: Explicit Constructions with Optimal Rate. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. 723–732.
[18] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems*. 6626–6637.
[19] Chi Jin, Praneeth Netrapalli, Rong Ge, Sham M Kakade, and Michael I Jordan. 2019. On Nonconvex Optimization for Machine Learning: Gradients, Stochasticity, and Saddle Points. *arXiv preprint arXiv:1902.04811* (2019).
[20] Chi Jin, Praneeth Netrapalli, and Michael I. Jordan. 2020. What is Local Optimality in Nonconvex-Nonconcave Minimax Optimization?. In *ICML 2020*. International Machine Learning Society (IMLS).
[21] Vijay Keswani, Oren Mangoubi, Sushant Sachdeva, and Nisheeth K. Vishnoi. 2020. GANs with First-Order Greedy Discriminators. *arXiv preprint arXiv:2006.12376* (2020).
[22] Qihang Lin, Mingrui Liu, Hassan Rafique, and Tianbao Yang. 2018. Solving weakly-convex-weakly-concave saddle-point problems as weakly-monotone variational inequality. *arXiv preprint arXiv:1810.10207* (2018).
[23] Richard J Lipton. 1994. A new approach to information theory. In *Annual Symposium on Theoretical Aspects of Computer Science*. Springer, 699–708.
[24] Mingrui Liu, Zhe Li, Xiaoyu Wang, Jinfeng Yi, and Tianbao Yang. 2018. Adaptive negative curvature descent with applications in non-convex optimization. In *Advances in Neural Information Processing Systems*. 4853–4862.
[25] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. *ICLR* (2018).
[26] Oren Mangoubi and Nisheeth K Vishnoi. 2020. Greedy Adversarial Equilibrium: An Efficient Alternative to Nonconvex-Nonconcave Min-Max Optimization. *arXiv preprint arXiv:2006.12363* (2020).
[27] Pasin Manurangsi and Daniel Reichman. 2018. The computational complexity of training ReLU(s). *arXiv preprint arXiv:1810.04207* (2018).
[28] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. 2017. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*. 2794–2802.
[29] Eric V Mazumdar, Michael I Jordan, and S Shankar Sastry. 2019. On finding local nash equilibria (and only local nash equilibria) in zero-sum games. *arXiv preprint arXiv:1901.00838* (2019).
[30] Silvio Micali, Chris Peikert, Madhu Sudan, and David A Wilson. 2005. Optimal error correction against computationally bounded noise. In *Theory of Cryptography Conference*. Springer, 1–16.
[31] Yurii Nesterov and Boris T Polyak. 2006. Cubic regularization of Newton method and its global performance. *Mathematical Programming* 108, 1 (2006), 177–205.
[32] Maher Nouiehed, Maziar Sanjabi, Tianjian Huang, Jason D Lee, and Meisam Razaviyayn. 2019. Solving a class of non-convex min-max games using iterative first order methods. In *Advances in Neural Information Processing Systems*. 14934–14942.
[33] Hassan Rafique, Mingrui Liu, Qihang Lin, and Tianbao Yang. 2018. Non-convex min-max optimization: Provable algorithms and applications in machine learning. *arXiv preprint arXiv:1810.02060* (2018).
[34] Sashank Reddi, Manzil Zaheer, Suvrit Sra, Barnabas Poczos, Francis Bach, Ruslan Salakhutdinov, and Alex Smola. 2018. A Generic Approach for Escaping Saddle points. In *International Conference on Artificial Intelligence and Statistics*. 1233–1242.
[35] Maurice Sion and Philip Wolfe. 1957. On a game without a value. *Contributions to the theory of games* 3 (1957), 299–306.
[36] Kiran Koshy Thekumparampil, Prateek Jain, Praneeth Netrapalli, and Sewoong Oh. 2019. Efficient Algorithms for Smooth Minimax Optimization. *NeurIPS* (2019).
[37] Yuanhao Wang, Guodong Zhang, and Jimmy Ba. 2020. On Solving Minimax Optimization Locally: A Follow-the-Ridge Approach. In *International Conference on Learning Representations*.