Evolvability of Real Functions

PAUL VALIANT, Brown University

We formulate a notion of evolvability for functions with domain and range that are real-valued vectors, a compelling way of expressing many natural biological processes. We show that linear and fixed-degree polynomial functions are evolvable in the following dually-robust sense: There is a single evolution algorithm that, for all convex loss functions, converges for all distributions.

It is possible that such dually-robust results can be achieved by simpler and more-natural evolution algorithms. Towards this end, we introduce a simple and natural algorithm that we call *wide-scale random noise* and prove a corresponding result for the L_2 metric. We conjecture that the algorithm works for a more general class of metrics.

Categories and Subject Descriptors: F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*Self-modifying machines*; G.1.6 [**Numerical Analysis**]: Optimization—*Convex programming*; I.2.6 [**Artificial Intelligence**]: Learning—*Concept learning*; J.3 [**Computer Applications**]: Life and Medical Sciences—*Biology and genetics*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Evolvability, optimization

ACM Reference Format:
Valiant, P. 2014. Evolvability of real functions. *ACM Trans. Comput. Theory* 6, 3, Article 12 (July 2014), 19 pages.
DOI: http://dx.doi.org/10.1145/2633598

1. INTRODUCTION

Since Darwin first assembled the empirical facts that pointed to evolution as the source of the hugely diverse and yet finely honed mechanisms of life, there has been the mystery of how the biochemical mechanisms of evolution (indeed, how any mechanism) can apparently consistently succeed at finding innovative solutions to the unexpected problems of life. This is not a biological, chemical, or physical mystery, because even stripping all these details from the phenomenon leaves us with a computational kernel well beyond our current grasp: how can an algorithm reliably find innovative solutions to unexpected problems? Any satisfactory account of evolution must explain how such an algorithm is feasible, and furthermore, be efficient enough to work despite very limited resources: millions of species, billions of members of a species, billions of generations, DNA length in the billions.

The realization that evolution embodies such a powerful algorithmic framework has spawned significant effort over more than half a century to develop algorithms that capture some of the power of evolution. Perhaps the oldest and most widely known approach along these lines is the area of *genetic algorithms* (sometimes known as *evolutionary algorithms*), which are routinely used in practice for many different applications. The genetic algorithms perspective views the evolution of an individual

This work was partially supported while at UC Berkeley by a National Science Foundation Postdoctoral Research Fellowship, and by NSF Grant CCF-0905626.
Author's address: P. Valiant, Computer Science Department, Brown University; email: pvaliant@gmail.com.

DOI: http://dx.doi.org/10.1145/2633598

species as an optimization problem relative to a *fitness function*, and aims to optimize this function by simulating celebrated parts of the evolution story on a population of candidates: survival of the fittest, mutation, and reproduction with genetic recombination. More explicitly, to optimize a function f over domain X, a pool of, say, 10,000 inputs ("creatures") in X is maintained at any time, and a series of "generations" are simulated, where in each generation, an input $x \in X$ from the pool may "reproduce" either by adding a number of new inputs to the pool that are slight variants on x (mutation), or by picking a mate $y \in X$ and adding a number of inputs to the pool that are in some sense a combination of x and y (recombination); then "survival of the fittest" is implemented, where each input in the pool is evaluated via f, and only the top 10,000 performers survive to form the next generation. Unlike natural evolution, genetic algorithms seem to require considerable expertise to set up for each new context, with performance depending delicately on subtle details and parameters of the mutation, recombination, and survival of the fittest procedures, as well as on initial conditions. In this sense, genetic algorithms, instead of aiming at illuminating how biology works, have rather become an independently successful field of their own. This article aims not to compete with genetic algorithms as an optimization tool, but rather to shed light on the biological mystery of evolution from a computational perspective.

1.1. The Evolvability Model

This article sits in the context of a line of work started by Leslie Valiant that seeks to provide the first rigorous model for the types of adaptations that can (and cannot) be evolved. The *evolvability* framework seeks to characterize functions that are evolvable, analogously with how learning theory seeks to characterize what is learnable. One of the chief hallmarks of PAC (probably approximately correct) learning theory—in contrast to what can be described as a general drawback of previous studies of evolution—is *robustness*. One would hope that any model that claims to capture as flexible a phenomenon as learning or evolution would itself be flexible. Whether the model classifies something as learnable, or evolvable, should not depend on fragile combinations of parameters or criteria or initial conditions. Indeed, in a different direction, the fundamental robustness of the notion of "computation" that crystalized with the work of Turing could be said to be this single most significant fact that enabled the computer revolution.

We will define the evolvability model explicitly in the next section, but for the moment, we will describe it intuitively in relation to the PAC notion of learning. Consider a single species as it adapts to the challenges of nature. Its DNA determines how these creatures will respond to the environment, essentially encoding a function which the creatures will execute on "inputs" provided by nature. If the creatures survive and reproduce, then their DNA has encoded a good function; in this sense, nature implicitly judges the fitness of all creatures' DNA. The question is to what degree the species can "learn" good response functions (as encoded by its DNA). As in PAC learning, one of the crucial questions is the distribution of inputs that nature provides. Indeed, most creatures (and learners) cannot hope to learn an accurate model of the challenges that will be presented to them; however, one does not have to know what challenges will be presented, only how to respond to the ones that are.

1.2. Results

Since the introduction of the evolvability model by L. Valiant [2009], significant further work has shown both its power and robustness to modeling variations [Feldman 2008, 2009a, 2009b; Kanade 2011; Kanade et al. 2010; Michael 2012]. In this article, we present two complementary constructions which extend this body of work in a new and natural direction: while previous papers studied evolvability of Boolean functions

(from $\{0,1\}^n \to \{-1,1\}$), we here consider functions from $\mathbb{R}^n \to \mathbb{R}^m$. Many of the functions that evolve in biology, for example, "how should the concentration of protein A in this cell vary in response to the concentrations of compounds B through Z?", might be much more naturally represented as real functions as opposed to Boolean functions. Of course, real functions, when restricted to Boolean domain and range, become Boolean functions, so in this sense, our model may be considered more general than the original Boolean model of evolvability. Our generalization of the evolvability model can be seen as introducing into evolvability notions that Haussler had introduced into the original PAC model [Haussler 1992].

For the original Boolean framework of evolvability, Feldman showed that as long as the underlying distribution from which nature draws examples is known, then the class SQ (statistical queries) defined by Kearns [1998] exactly characterizes the classes of evolvable functions [Feldman 2008]. SQ is both a powerful and natural framework, and seems to capture most of the power of PAC learning. However, the assumption that the evolution algorithm must know the underlying distribution is decidedly unnatural, as one would hope for evolution to function across a broad range of potentially quite-intricate and varying distributions of conditions for its creatures. In subsequent work, Feldman showed that if one reinterprets the Boolean model by allowing hypotheses that take real values (even though the hypotheses are then compared, via a loss function, to target values that are Boolean), then if the performance metric is nonlinear—that is, essentially anything except the correlation metric (the L_1 metric restricted to a small enough region so that it is a linear function, instead of the usual piecewise linear)—one can take advantage of a "kink" in it to, in fact, evolve everything in SQ [Feldman 2009b, Theor. 4.3]. In a sense, by changing part of the Boolean model to allow for real values, Feldman circumvents an apparent limitation of the purely Boolean evolution setting.

In this current work, we consider all elements of the evolvability model to be real instead of Boolean. The results we derive are of a somewhat different form from those in the (partially) Boolean settings previously considered: instead of relating evolution to learning theory, we relate it to the area of *optimization algorithms*.

Specifically, in Section 4, we adapt Feldman's results on evolvability [Feldman 2009b] to our setting, which, because of the different setting, yields results of a different nature: evolvability can simulate arbitrary polynomial-time optimization algorithms that only require approximate access to the function being optimized. In the terminology of Lovász [1986], this is *weak optimization*. We show that his construction of polynomial-time weak convex optimization can be leveraged to yield evolution algorithms for the class of linear functions, and further, fixed-degree polynomials, over any distribution, for any convex loss function—including a fortiori the commonly considered linear and quadratic (L_1- and L_2-squared) loss functions.

While these results demonstrate the power of the real evolvability framework, they come at the expense of a certain unnaturalness of the underlying evolution algorithm (a reduction to the ellipsoid algorithm does not on the surface appear to be a convincing account of the biological evolution process). We thus begin the technical part of this article in Section 3 by considering perhaps the simplest and most natural algorithm that could hope to work, and show that it in fact can reproduce a significant portion of these results. In a generation of this algorithm, a parent produces a polynomial number of nearby children, each chosen in a uniformly and independently random direction from the parent. Survival of the fittest turns this into a kind of "steepest descent" strategy, which enables us to prove that, for quadratic loss functions, constant progress is made in each generation, which will rapidly lead to the optimum.

This algorithm, which we call *wide-scale random noise* to emphasize its simple unstructured nature, has in fact been found in simulations to converge rapidly in many

cases beyond that of the quadratic loss function, though how it achieves this convergence seems rather different than its provable behavior in the quadratic case. In particular, while for quadratic loss functions, the algorithm consistently produces offspring which perform better than the parent, leading to a guaranteed improvement, in the case of the L_1 loss function, the ability to evolve to a descendent whose performance is worse than that of the parent seems crucial for efficient progress. It would seem counterintuitive that such backtracking would help in a convex landscape with no spurious local minima. However, this was exactly the effect found in a paper on simulated annealing that also considered a very similar L_1 optimization setting [Kalai and Vempala 2006]. It would appear that both evolvability and simulated annealing seem effective in unexpected cases, and one might hope that new analysis of one might shed light on the other.[1] We conjecture here that the results of Theorem 3.3 hold much more generally than for just the case of L_2 loss that we prove here: we conjecture that similar results hold for any loss function $L(x,y) = ||x-y||_c$ for $c > 0$, including specifically those functions for $c < 1$ which are not convex and where even the powerful tools of Section 4 fail. Specifically, the expected performance of the hypothesis h (for some fixed loss function L, distribution D, and target function x), may be a non-convex function $L\mathsf{Perf}_{f,D}(h)$ of h, when $c < 1$ (see Definition 2.3 and Theorem 4.4), yet must be effectively optimized if evolution is to be successful. Though these functions may not be convex, they are always *star-convex* functions, a generalization of the notion of a *star-convex set*, where the function will be convex on any line through the global optimum [Nesterov and Polyak 2009]. This definition forbids local optima and further implies that from any point, the global optimum lies in a downhill direction. Such functions are a tantalizing frontier for research in optimization with significant structure that could potentially be exploited, though not much is yet known.

1.3. Related Issues

Given the emphasis, both in this work and in the field of genetic algorithms, on *optimization* as the perspective from which to analyze evolution, it is worth pointing out the limitations of this view. A recent line of work started by Livnat et al. [2008, 2009] asks what role sex plays in evolution, since breaking up successful gene combinations (by randomly mixing two inputs x and y that are individually good) seems a suboptimal optimization strategy. Yet the phenomenon of creatures going to great lengths to combine their genes with those of other individuals is near-universal, so it must be vital to evolution—vital to a sense of evolution broader than just "fitness optimization". What was found in a series of papers is that genetic recombination promotes *mixability* instead of fitness, where this new notion of mixability describes genes that work well with a variety of other genes, as opposed to achieving some sort of brittle perfection only when combined in a unique way. Mixability is useful for a variety of strange and subtle biological ends: mixability is very related to modularity—if a given chunk of DNA helps the creature no matter which other chunks are present, one could call this chunk a self-contained module. Modular functionality is an investment in the future of evolution, enabling creatures to easily adapt to new conditions with small (modular) changes to their DNA, instead of a very costly global reorganization. The spontaneous transition from a global approach to a modular approach within a single

[1]Feldman has defined *monotone* evolvability to be the restriction that each generation's performance must be at least that of the previous generation, in part inspired by a desire to consider evolution algorithms that seem more "natural". Monotone evolvability has been shown in a distribution-independent setting for point functions under the L_1 metric [Feldman 2009b], conjunctions under the L_2 metric [Feldman 2009a], and recently, for linear threshold functions with nonnegligible margin, for L_2 and related metrics [Feldman 2011]. Such work raises interesting questions about the potential difference in power between monotone vs. non-monotone evolvability, and indeed, monotone vs. non-monotone optimization in general.

optimization problem is beyond what we can satisfactorily model with mathematical tools, but it provides tantalizing hints at the unusual effectiveness of evolution. More concretely, while this article (and, in general, the optimization view of evolution) considers evolution on a short enough timescale that the environment is *fixed*, the mixability papers [Livnat et al. 2008, 2009] provide evidence that the great challenges lie in providing precise models of ever-longer and ever-richer slices of evolution.

In a different direction, one feature of the evolvability model is that at every generation, there is only a single *parent* that produces all the children that constitute the next generation. However, this simplifying assumption is less of a restriction than it might appear: even in the original paper introducing evolvability, it was argued that the genome of this parent—instead of representing a single creature—could store the concatenated genomes of a whole population of parents; in each generation, the lifetime of a single member of the population would be simulated (see Valiant [2009], Sect. 6). Further, interactions among the genomes of a population of N parents—such as recombination via sexual reproduction—can be modeled in the original evolvability model via *internal* interactions within this concatenated genome from generation to generation, namely, mutation. Thus, a population of size N, whose genomes may interact in complicated ways, can be emulated in the evolvability model (via only a single parent per generation). However, a recent paper [Kanade 2011] has considered explicitly introducing a population of parents into the evolvability model, along with explicitly modeling recombination of pairs of their genomes. The paper demonstrates that a polynomially-large parent population size may dramatically reduce the number of generations required to evolve certain functions, from polynomial to (in some cases) polylogarithmic.

2. DEFINITIONS AND COMMON FRAMEWORK

The definitions are adapted from the Boolean setting of evolvability [Feldman 2009b; Valiant 2009], which are themselves inspired by the PAC-learning framework. We start with a high-level overview of PAC learning: while nothing in this article relies directly on PAC-learning results, the analogy will shed light on the definitions to follow.

2.1. Motivation: PAC Learning

In PAC learning, there is a target function $f : X \to Y$ that we wish to "learn" via labeled examples: we can request examples $x_i \leftarrow D$ from a certain distribution D over X, and from the resulting pairs $(x_i, f(x_i))$, we hope to be able to construct a *hypothesis* function h that is "probably approximately correct". That is, our goal is a learning procedure that, with high probability over the random training data, will yield a function h that well-approximates the target f, at least with respect to further inputs x drawn from the same distribution D that gave us the training data. Explicitly, we say that a class F of functions $f : X \to Y$ is PAC-learnable if there is an algorithm A such that for every $\epsilon, \delta > 0$, every target function $f \in F$, and every distribution D over X, if we run A on a sample $\{x_i\} \leftarrow D$, then with probability at least $1 - \delta$, algorithm A will return a hypothesis $h : X \to Y$ such that $\Pr_{x \leftarrow D}[f(x) = h(x)] \geq 1 - \epsilon$. The runtime of algorithm A and the size of the sample given to A must both be polynomial in the parameters of the domain, as well as $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$.

The fact that the distribution D is allowed to be arbitrary (and unknown) is crucial to the goal of modeling learning in a realistic setting. For example, if we are modeling how a certain creature learns to, say, classify other animals as predators and prey, based on certain observable features that lie in some space X, then the distribution D (over X) captures the "distribution of animals that our creature encounters, as

represented by their features in X". This distribution D is almost certainly not going to be a clean mathematical object, such as a uniform distribution. Nevertheless, effective learning algorithms exist, because as it turns out, one does not need to "understand" the distribution of examples in order to learn to classify them.

The evolvability model is inspired by the notion that Darwinian evolution can be viewed as a kind of learning. A creature's environment presents it with a series of random challenges to which the creature responds; these responses are in some sense encoded in the creature's DNA. Depending on whether the creature reacts appropriately to the challenges of its environment, it may either survive and reproduce, or die off.

We can model this interaction with notation from the PAC framework. Let $h : X \to Y$ denote the *hypothesis* function encoded by the creature's DNA, mapping environmental challenges in X to its responses in Y, and let D denote nature's distribution over X. For each situation $x \in X$, we can represent the *ideal response* of the creature via the *target function* $f(x)$. The creature's performance results from comparing $h(x)$ to $f(x)$, for x drawn from nature's distribution D. As is sometimes done with PAC learning, instead of simply checking whether $h(x) = f(x)$, we can evaluate performance via a general *loss function*, $L : Y \times Y \to [0, \infty)$, that takes $h(x)$ and $f(x)$ and returns a number evaluating, in our case, "how much the creature's choice of $h(x)$ instead of $f(x)$ hurt its reproductive chances". Thus, a "fit" creature is one for which $E_{x \leftarrow D}[L(h(x), f(x))]$ is close to 0. (As a philosophical note, the target function f, representing the ideal response of a creature to environment x so as to maximize its chances of reproducing, is a somewhat mysterious object in explicit biological settings. However, even if f is not observable, it in principle exists and is explicitly definable. Our goal is to show that no matter what f is—in a suitably generous class, which in this article is low-degree polynomials—good approximations to f will evolve efficiently and with high probability.)

Viewed in this light, Darwinian evolution is a form of PAC learning, where DNA stores the hypothesis function, and each generation of life constitutes a cycle of an algorithm whereby the hypothesis implicit in the DNA is evaluated, and if sufficiently successful, will produce variant hypotheses populating the next generation. However, evolution is a *restricted* form of PAC learning. In the PAC model, one may examine a polynomially-large number of labeled examples and process them to determine a hypothesis function h; this would be analogous to nature presenting a creature with a history of all the situations the creature encountered, along with a labeling of what the "correct" response to each situation would have been, and then challenging the creature to optimize its offspring accordingly. In some ways, this description sounds like the discredited Lamarckian model of evolution. The Darwinian model is much cleaner, however. Your DNA gets no handbook of labeled examples: the only feedback from nature is survival or death.

We note that this may be a slightly controversial point among biologists—there is recent excitement about the notion of "epigenetics", which provides unusual and ill-understood biochemical mechanisms allowing creatures to pass down information from their lifetimes to their offspring. However, we stress that the model of evolvability does not seek to capture the totality of biological phenomena, but rather is aimed at abstracting certain key features that may shed light on why evolution seems unlike any computational model we are familiar with. When one designs an iterative learning algorithm—or when pre-Darwin thinkers imagined how successive generations of life adapt to fit their conditions—one imagines communication between successive iterations/generations to be fundamental. Lamarck envisioned that if a parent uses and develops a certain muscle over its lifetime, then it will have children born with that muscle, even larger—for how else could the species, over the course of generations,

adapt to have all its muscles appropriate to the tasks that species does? Darwin shocked the world in describing how such results could be achieved with no such communication, with only random variation and survival of the fittest connecting subsequent generations to the previous ones. The model of evolvability seeks to model this surprising phenomenon in a way that is both simple and general.

2.2. Definitions

We introduce the following definitions for evolvability in the (multidimensional) real number setting.

Definition 2.1. For positive integers n, m, the (n, m)-dimensional evolvability model considers the evolution of a hypothesis function $h : \mathbb{R}^n \to \mathbb{R}^m$ towards a target function f with the same domain and range. These functions are evaluated at points drawn from a distribution D over their domain $X = \mathbb{R}^n$, where D is from a set of distributions $\mathcal{D}$. We will consider the evolution of target functions f that are unknown members of a certain set C of functions, known as a *concept class*.

Definition 2.2. In the (n, m)-dimensional evolvability model, with hypotheses having range $\mathbb{R}^m$, a *loss function* is a nonnegative function $L : \mathbb{R}^m \times \mathbb{R}^m \to [0, \infty)$ such that for any $y \in \mathbb{R}^m$, $L(y, y) = 0$.

In this article, we will exclusively use n to denote the dimension of the domain of the hypothesis and target functions, and m to denote the dimension of their range, and thus for convenience will omit repeatedly reintroducing them.

Definition 2.3. Given a loss function L and a target function $f : \mathbb{R}^n \to \mathbb{R}^m$, the *performance* of a hypothesis function $h : \mathbb{R}^n \to \mathbb{R}^m$ relative to a distribution D (over $\mathbb{R}^n$) is defined as $L\mathsf{Perf}_{f,D}(h) = \mathbf{E}_{x \leftarrow D}[L(f(x), h(x))]$. Given a positive integer s, the s-sample *empirical performance* $L\mathsf{Perf}^s_{f,D}(h)$ is defined to be the random variable resulting from drawing a sample of size s, $x_1, \ldots, x_s \leftarrow D$ and evaluating $\frac{1}{s}\sum_{i=1}^{s} L(f(x_i), h(x_i))$.

In a manner which will be made precise shortly, evolution evaluates a variety of hypotheses h via their empirical performance $L\mathsf{Perf}^s_{f,D}(h)$, aiming at the ideal performance of the target function f, which has loss 0.

Hypotheses change with each generation via a *mutation algorithm*, which is a randomized procedure that is run on a creature's DNA to produce the DNA of an offspring. The randomness in the mutation algorithm means that running it repeatedly from the same input may yield different results. In our context, when a parent produces a pool of p children, the DNA of each child is produced by a separate call to the mutation algorithm, taking the DNA of the parent as input. The set of all possible DNA sequences that the mutation algorithm may produce from a given DNA sequence is called the *neighborhood* of the sequence, even though in any particular instance, the actual offspring may represent only a small portion of this neighborhood.

Definition 2.4 (*Feldman [2009b], Def. 3.6*). Given parameter $\epsilon > 0$, a *mutation algorithm* A is defined by a pair (R, M) where the following hold.

— R is a set of hypothesis functions $\mathbb{R}^n \to \mathbb{R}^m$, where to each function h corresponds a string of finite length r that we say *represents* the function.
— M is a randomized polynomial (in n, m, and $1/\epsilon$) time Turing machine that, given $r \in R$ and $1/\epsilon$ as inputs, outputs a representation $r_1 \in R$ with probability that we denote $\Pr_A(r, r_1)$. The set of representations that can be output by $M(r, \epsilon)$ is referred to as the *neighborhood* of r for ϵ and is denoted by $\mathsf{Neigh}_A(r, \epsilon)$.

Recall that the hypothesis functions are ultimately stored as the "DNA" of our creatures, and thus are represented as strings over a finite alphabet. For the results of Section 4, we explicitly represent functions as binary strings, though the class of functions represented by the scheme of Section 4 is somewhat artificial. In Section 3, we consider genomes that can represent the entire class of fixed-degree polynomial functions, and implicitly consider these polynomials as having each real-valued coefficient represented to fixed precision via a short string in the genome.

The mutation algorithm is the source of genomes for the next generation; which genomes survive is determined by the *selection rule*, an efficiently-implementable algorithm that we imagine being implicit in nature: in each generation, the parent produces a *candidate pool* of p creatures via running the mutation algorithm p times on its DNA; the p creatures then compete with each other for survival. Instead of selecting the best one, the evolvability model suggests something weaker, more realistic, and more robust: there is a threshold t such that if all of the offspring perform within t of their parent, then they are viewed as being essentially indistinguishable, and a random one of them survives; if some offspring perform more than t better than the parent (their loss is t *less* than the loss of the parent), then a random one of these "noticeably better offspring" is selected for the next generation; in the unusual case where all offspring are noticeably worse than the parent, then the parent survives to the next generation (or, equivalently, a clone of the parent survives). Explicitly, we define SelNB, the selection rule that distinguishes neutral and beneficial mutations (adapted from [Feldman 2009b], Definition 3.7).

Definition 2.5. For a loss function L, tolerance t, candidate pool size p, and sample size s, the *selection rule* $\mathsf{SelNB}[L,t,p,s]$ is the algorithm that for any function f, distribution D, mutation algorithm $A = (R,M)$, representation $r \in R$, and accuracy ϵ, $\mathsf{SelNB}[L,t,p,s]\,(f,D,A,r)$ outputs a random variable that takes a value r_1 determined as follows. First, run the mutator $M(r,\epsilon)$ p times and let Z be the set of representations obtained. For $r' \in Z$, let $\Pr_Z(r')$ be the relative frequency with which r' was generated among the p observed representations. For each $r' \in Z \cup \{r\}$, compute an empirical value of performance $v(r') \leftarrow L\mathsf{Perf}^s_{f,D}(r')$. Let $\mathsf{Bene}(Z)$ denote the set of empirically beneficial mutations, $\{r' \in Z : v(r') \leq v(r) - t\}$, and $\mathsf{Neut}(Z)$ denote the set of empirically neutral mutations, $\{r' \in Z : |v(r') - v(r)| < t\}$. Then, the following.

(i) If $\mathsf{Bene}(Z) \neq \emptyset$, then output a random $r_1 \in \mathsf{Bene}(Z)$ distributed with relative probabilities according to $\Pr_Z$.
(ii) If $\mathsf{Bene}(Z) = \emptyset$ and $\mathsf{Neut}(Z) \neq \emptyset$, then output a random $r_1 \in \mathsf{Neut}(Z)$ distributed with relative probabilities according to $\Pr_Z$.
(iii) If $\mathsf{Neut}(Z) \cup \mathsf{Bene}(Z) = \emptyset$, then output r, indicating that the genome of the parent survives unaltered.

If for a concept class and set of distributions, there exists a mutation algorithm that, under selection rule SelNB, efficiently converges to any target function in the class, then we say that the concept class is evolvable.

Definition 2.6 (*Adapted from [Feldman 2009b], Def. 3.3*). A concept class C, set of distributions $\mathcal{D}$, and loss function L are said to be *evolvable* if there exists a mutation algorithm $A = (R,M)$, polynomials $p(n,m,\frac{1}{\epsilon})$, $s(n,m,\frac{1}{\epsilon})$, a tolerance $t(n,m,\frac{1}{\epsilon})$ whose inverse is poly-bounded, and a polynomial number of generations $g(n,m,\frac{1}{\epsilon})$ such that for all n,m, target functions $f \in C$, distributions $D \in \mathcal{D}$, $\epsilon > 0$, and any initial genome

$r_0 \in R$, and any number of generations $g' \geq g$, with probability at least $1-\epsilon$, the random sequence defined by $r_i \leftarrow \mathsf{SelNB}[L,t,p,s](f,D,A,r_{i-1})$ will have $L\mathsf{Perf}_{f,D}(r_{g'}) \leq \epsilon$, namely, the mechanism of evolution will yield an almost-optimal genome in a polynomial number of generations, g.

We note that the sign convention most natural for the real case is opposite that used in previous work for the Boolean case, and in particular, a "perfect organism" in our setting has $L\mathsf{Perf} = 0$, while in Feldman [2009b], would have $L\mathsf{Perf} = 1$.

One important distinction between the model of this article and the previous Boolean model is that working over the real numbers introduces problems of scale that are not present in the Boolean case. For example, since the "feedback" that nature gives the evolution algorithm in any generation consists of simply choosing which of a bounded (polynomial) number of potential children survives to the next generation, there is no way to evolve in bounded time a good approximation to an unbounded real number. It is thus important to work with concept classes C that are in some sense bounded. A related issue arises with the set of distributions $\mathcal{D}$. Suppose we are working with the L_1 loss function, $L(x,y) = |x-y|$, and suppose distribution D is such that, with probability $1-\tau$, D samples the point $\mathbf{0}$, and with probability τ, D samples a point more than $\frac{1}{\tau}$-far from the origin. If τ is super-polynomially small, then evolution will likely never see any points other than $\mathbf{0}$ in the sample, but meanwhile, the expected loss of hypotheses is unknown and potentially huge. Thus, D (and L) must also be reasonably bounded. Thus, while such bounds do not appear in the Boolean case, they are necessary to making sense of real evolvability. We make precise the bounds we use in the statements of the theorems. We believe that they are as mild as is reasonable.

In this article, we work with functions that range over $\mathbb{R}^m$, and one may ask about the role of $m > 1$. For example, it might seem natural and sufficient to decompose a function $p : \mathbb{R}^n \to \mathbb{R}^m$ into a vector of m separate functions $\mathbb{R}^n \to \mathbb{R}$ and optimize the performance of each separately, applying the loss function to the vector where each of the other functions is assumed to take some default value, perhaps 0. However, this approach is in some sense analogous to trying to evolve walking by optimizing each leg separately, assuming each other leg were fixed immobile. Evolution seems an inherently high-dimensional problem, in many senses, thus why we emphasize the $m > 1$ case here. It is perhaps an unexpected bonus that the $m > 1$ case in the following proofs essentially "comes for free", requiring no special analysis.

3. A DIRECT APPROACH

In this section, we construct what is perhaps the simplest conceivable random mutator that could hope to do "evolutionary hill-climbing" (technically, in our case, the less glamorous sounding "valley descent") and show that it is in fact surprisingly adept. In particular, it is capable of efficiently evolving arbitrary low-degree multivariate polynomials with bounded coefficients. We analyze the particularly well-behaved case of the quadratic loss function, that is, $L(x,y) = ||x-y||_2^2$, and show that our particularly simple mutator significantly improves fitness with each generation, for any hypothesis function that is not already very close to the target.

Definition 3.1. *Wide-scale random noise* parameterized by a lower and upper bound (ℓ, u) and a dimension k is the result of the following process: choose a uniformly random real number ρ from the interval $[\log_2 \ell, \log_2 u]$; return 2^ρ times a randomly chosen element of the k-dimensional unit ball.

Our mutation algorithm consists simply of producing several offspring, each chosen by adding to the parent an independent draw from the wide-scale random noise distribution.[2] Specifically, the following.

Definition 3.2. For degree d polynomials from $\mathbb{R}^n \to \mathbb{R}^m$, the mutation algorithm $A = (R, M)$ for wide-scale random noise, parameterized by bounds ℓ and u, is defined as follows.

—R is the representation expressing a degree d polynomial $\mathbb{R}^n \to \mathbb{R}^m$ via the vector of its $k = m \cdot \binom{n+d}{n}$ coefficients.
—M consists of generating k-dimensional, wide-scale random noise, parameterized by bounds ℓ, u, and adding it to the input vector.

THEOREM 3.3. *Given any positive integer constant d, then for any real number r, there exist bounds ℓ, u, and an integer $c = poly(n, m, r)$ such that the wide-scale random noise mutator with scale in $[\ell, u]$ and c children per generation evolves (according to Definition 2.6) the class of degree $\leq d$ polynomials from $\mathbb{R}^n \to \mathbb{R}^m$ with coefficients at most r over the set of all distributions on the n-dimensional radius r ball, with respect to the quadratic (L_2) loss function.*

We show that with very high probability, at least one of the children in the next generation will have fitness significantly better than that of the parent, where "significant"+ means greater than the tolerance threshold t of Definition 2.5. We analyze the random mutation procedure in two steps: we first show that progress can always be made if we choose the "right" radius in Definition 3.1, and then we observe that, because of the exponential way in which the radius is chosen, the wide-scale random noise mutator is very likely to choose a radius that is almost exactly "right".

We start with a basic structural result.

LEMMA 3.4. *Given an arbitrary fixed target function $f : \mathbb{R}^n \to \mathbb{R}^m$ and a fixed distribution D over the input space $\mathbb{R}^n$, then the expectation over D of the quadratic loss function between f and a degree d polynomial hypothesis function h, namely, $E_{x \leftarrow D}[\,||f(x) - h(x)||^2]$, is a convex quadratic function of the coefficients of h.*

PROOF. Consider an arbitrary $x \in \mathbb{R}^n$; the function $f(x)$ has some fixed value here, in $\mathbb{R}^m$. To analyze $||f(x) - h(x)||^2$, we must first recall how $h(x)$ depends on the coefficients of h. Recall how a degree d polynomial mapping $x = (x_1, \ldots, x_n)$ to the real numbers is defined: for each sequence $\alpha = (\alpha_1, \ldots, \alpha_n)$ of nonnegative integers with sum at most d, the polynomial has a term $c_\alpha \cdot x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdot \ldots \cdot x_n^{\alpha_n}$, where c_α is an arbitrary real number. The coefficients c_α collectively define the polynomial. In our case, h maps $\mathbb{R}^n \to \mathbb{R}^m$, thus we may regard each coefficient c_α itself as being an m-element vector. (Note that there are $\binom{n+d}{n}$ valid sequences α, and hence coefficients c_α; hence a degree d polynomial from $\mathbb{R}^n \to \mathbb{R}^m$ is specified by $m \cdot \binom{n+d}{n}$ real numbers.)

Thus $h(x)$, for fixed x, is a linear function of its coefficients c_α. Thus $f(x) - h(x)$ is also a linear function of the coefficients of h, and $||f(x) - h(x)||^2$ is hence a convex quadratic function of the coefficients of h.

Having shown that for each x, the expression $||f(x) - h(x)||^2$ is a convex quadratic function of the coefficients of h, we now note that $E_{x \leftarrow D}[\,||f(x) - h(x)||^2]$ is hence a

[2]Vitaly Feldman has pointed out [personal communication] that one can reproduce the results of this section with a mutator with much smaller neighborhood: as opposed to choosing random elements of the k-dimensional unit ball, one can instead just take one of the k standard unit basis vectors or its negation. This discrete process is arguably more like the mutations that occur in DNA in nature, though in either case, the result must still be scaled by a carefully chosen wide-scale multiplier.

weighted average of convex quadratic functions and is hence a convex quadratic function itself, as claimed. □

Before applying this lemma, we note that for any convex quadratic function on k variables, there exists a rotation and translation of its input domain that puts the function in the form $\sum_{i=1}^{k} c_i \cdot x_i^2$, for nonnegative c_i (such a transformation may be found from the eigenvalues of the positive semidefinite matrix that defines the quadratic portion of the function). Viewing the expected loss of a polynomial hypothesis evolving in the context of Theorem 3.3 in this form—as $\sum_{i=1}^{k} c_i \cdot x_i^2$, for nonnegative c_i, where $\{x_i\}$ are a rotated and translated form of the polynomial's $k = m \cdot \binom{n+d}{n}$ coefficients—we show the following lemma, implying that provided the right radius is chosen in Definition 3.1, each child has a significant chance of being significantly fitter than the parent:

LEMMA 3.5. *Given $\epsilon > 0$ and a vector of nonnegative coefficients, $(c_1, \ldots, c_k)$, with a bound $\sigma \geq \sum_{i=1}^{k} c_i$, then the quadratic function $q : \mathbb{R}^k \to \mathbb{R}$ defined as $q(x) = \sum_{i=1}^{k} c_i \cdot x_i^2$ has the property that for any vector x of length at most 1, if $q(x) > \epsilon$, then with probability at least $\frac{1}{4}$, a randomly chosen vector y in the ball of radius $\frac{\epsilon}{6\sigma\sqrt{k}}$ about x will have $q(y) < q(x) - \frac{\epsilon^2}{12\sigma k}$.*

The restriction that x has length at most 1 is for the sake of convenience of the proof; when we apply the lemma in the context of Theorem 3.3, we will scale the inputs so that the global bound (r) on the size of the inputs becomes scaled to 1.

PROOF OF LEMMA 3.5. To aid with the proof, we first note the following elementary fact (see e.g., [Ball 1997, Chapter 1]).

Fact. A k-dimensional ball of unit radius centered at the origin has at least $\frac{1}{4}$ of its volume in the region where its first coordinate exceeds $\frac{1}{3\sqrt{k}}$.

Next, consider, for a vector x in the unit ball, the quadratic function q restricted to the line connecting x to the origin. Since q has value 0 and derivative 0 at the origin and is quadratic, it must have derivative (along this line) of $\frac{2q(x)}{||x||}$ at x; since by assumption, $q(x) \geq \epsilon$ and $||x|| \leq 1$, this is at least 2ϵ. Since we have bounded the derivative in just one direction, the gradient of q at x must have magnitude at least this bound of 2ϵ.

Consider the value of q in the ball of radius $r \triangleq \frac{\epsilon}{6\sigma\sqrt{k}}$ about x, and specifically, in the portion that is at least $\frac{r}{3\sqrt{k}}$ in the direction of the (downward) gradient from x. By this fact, this portion comprises at least a quarter of the ball.

For a point y in this portion, consider the second-degree Taylor expansion of q about x, which is exact, since q is quadratic. Letting g be the gradient of q at x, and b be the directional second derivative of q at x in the direction of y, we have $q(y) = q(x) + g \cdot (y - x) + \frac{1}{2} b \cdot ||y - x||^2$. By the condition that y more than $\frac{r}{3\sqrt{k}}$ in the direction of the (downward) gradient from x, and the gradient has magnitude at least 2ϵ, we bound the linear term as $g \cdot (y - x) < -2\epsilon \frac{r}{3\sqrt{k}} = -\frac{\epsilon^2}{9\sigma k}$. We bound the remaining term by noting that since σ is a bound on the sum of the coefficients of q, then 2σ bounds the second derivative in any direction, and hence b. Thus, $\frac{1}{2} b \cdot ||y - x||^2 \leq \sigma r^2 = \frac{\epsilon^2}{36\sigma k}$.

Combining these results yields the desired bound: over the entire portion of the ball that we are considering, the quadratic function q has value less than $q(x) - \frac{\epsilon^2}{9\sigma k} + \frac{\epsilon^2}{36\sigma k} = q(x) - \frac{\epsilon^2}{12\sigma k}$, namely, $\frac{\epsilon^2}{12\sigma k}$ less than its value at x; because this portion of the ball

comprises at least $\frac{1}{4}$ its volume, a randomly chosen vector in the ball will decrease the value of q by more than $\frac{\epsilon^2}{12\sigma k}$ with probability at least $\frac{1}{4}$, as claimed. □

We now assemble the pieces into a proof of Theorem 3.3.

PROOF OF THEOREM 3.3. Both the hypothesis functions and the target functions are degree d polynomial functions $f : \mathbb{R}^n \to \mathbb{R}^m$, which we represent via their $k = m \cdot \binom{n+d}{n}$ coefficients, each of which have magnitude at most r by assumption. Given any pair of these functions, h, f, and any point $x \in \mathbb{R}^n$, whose coordinates have magnitude at most r, consider the L_2 loss function $||h(x) - f(x)||^2$ as a function of the k coefficients $\{h_i\}$ of the hypothesis function h; after a suitable rotation and translation, the loss can be represented in the form $\sum_{i=1}^k c_i h_i^2$ for nonnegative coefficients c_i; let b be a bound on $\sum_{i=1}^k c_i$ over all pairs f, h and all points x. Both k and b are bounded by polynomials for constant d, and the same bound b applies for the expected loss, over any distribution D over such points $\{x \in \mathbb{R}^n : \forall i \leq k, |x_i| \leq r\}$.

With a view towards applying Lemma 3.5, we note that if we rescale the parameters $\{h_i\}$ by $\frac{1}{r\sqrt{k}}$, then the rescaled parameter vector will lie in the unit ball, and the coefficients of the quadratic loss function, in terms of these scaled parameters, will increase by a factor of $(r\sqrt{k})^2$. Thus the coefficients now have a sum that is bounded by br^2k. We thus consider the application of Lemma 3.5 to this transformed expected loss function, using the bound $\sigma = br^2k$ just computed. For any point in the k-dimensional unit ball that has expected loss greater than ϵ, Lemma 3.5 guarantees that there exists a "magic radius" $\mu = \frac{\epsilon}{6\sigma\sqrt{k}}$ such that moving the (rotated, translated, and rescaled) hypothesis by a vector randomly chosen in the k-dimensional ball of radius μ will, with probability at least $\frac{1}{4}$, improve the expected loss by more than $\frac{\epsilon^2}{12\sigma k}$. Since we have polynomial upper bounds on σ, Lemma 3.5 thus provides for inverse-polynomial progress, in exactly those cases where we are not already within ϵ of optimal (0) loss.

This analysis is predicated on choosing exactly the right "magic" radius μ, but in fact, the wide-scale random noise procedure chooses the radius at random, as we would not expect evolution to "know" all the parameters involved and exactly customize itself to them. We thus consider wide-scale random noise as choosing from a huge but polynomially-sized range $[\ell, u]$ that is guaranteed to contain the radius μ, and consider the following general fact: a pair of k-dimensional balls with the same center, whose radii r, r' have logarithms within $\frac{1}{k}$ of each other, namely, $|\log r - \log r'| \leq \frac{1}{k}$, will share at least a $\frac{1}{e}$ fraction of their volume.

Thus, with at least inverse-polynomial probability, choosing a radius that is 2 to the power of a number uniformly chosen between $\log_2 \ell$ and $\log_2 u$ will yield a radius whose logarithm is within $\frac{1}{k}$ of the logarithm of μ, and from there, Lemma 3.5 guarantees that with constant probability $\frac{1}{4e}$, the resulting mutation will improve the expected loss by at least $\frac{\epsilon^2}{12\sigma k}$, provided the expected loss is not already within ϵ of optimal. Recall that in each generation, many "candidates" are generated for the next generation; we set the number of candidates high enough so that with overwhelming probability, say, probability at least $1 - \frac{\epsilon}{2}$, such a mutation will be present in each generation.

Thus, we may choose s, that is, the size of the sample with which to evaluate the empirical performance, high enough so that via Chernoff bounds, with probability at least $1 - \frac{\epsilon}{2}$, over the entire course of the algorithm, all estimates will be accurate to within a third of the minimum improvement, $\frac{\epsilon^2}{36\sigma k}$. Further, we choose t, the threshold

for declaring a mutation beneficial, to be equal to $\frac{\epsilon^2}{18\sigma k}$, so that assuming each empirical estimate is in fact accurate to within $\frac{\epsilon^2}{36\sigma k}$, each of the beneficial mutations guaranteed by Lemma 3.5 will be recognized and declared to be beneficial. Thus, with probability at least $1-\epsilon$, the performance of every generation will be at least $\frac{\epsilon^2}{36\sigma k}$ better than that of the previous, unless we are already within $\epsilon + \frac{\epsilon^2}{36\sigma k}$ of optimum, yielding the desired result.

To give a sense of the parameters involved, given our bound b on the expected loss, as already defined, the total number of generations needed for convergence is thus at most $b \Big/ \frac{\epsilon^2}{36\sigma k} = \frac{36b\sigma k}{\epsilon^2}$. For the parameters of wide-scale random noise, we note that the magic radius of Lemma 3.5, $\mu = \frac{\epsilon}{6\sigma\sqrt{k}}$, after reversing the rescaling by $\frac{1}{r\sqrt{k}}$, equals $\frac{r\epsilon}{6\sigma}$, which by definition of $\sigma = br^2k$ equals $\frac{\epsilon}{6brk}$; hence, the interval $[\ell, u]$ should be chosen so as to contain this number. □

We note that we insist on constant r and d in Theorem 3.3 because the definition of evolvability (Definition 2.6) insists that each parameter of performance must be bounded by a polynomial function of only n, m, and $\frac{1}{\epsilon}$. Explicitly, each of the parameters of the proof of Theorem 3.3 in fact depends as mildly as might be expected on r and d, depending polynomially on the number of coefficients needed to describe the hypothesis class of degree-d polynomials, $k = m \cdot \binom{n+d}{n}$, and on r^d, which captures the growth of the output of degree-d polynomials on inputs of magnitude up to r. This same will hold true in the next section, with Theorem 4.5.

We also point out that the radius of wide-scale random noise needed by our proof of Theorem 3.3 is a single constant, and thus one could argue that a random radius (the "scale" of wide-scale random noise) is not needed for evolution. However, the appropriate radius, $\frac{\epsilon}{6brk}$ (as derived at the end of the proof of Theorem 3.3), depends on many of the parameters of evolution, and we consider it an appealing feature of these results that a single wide-scale mutator is effective over a wide range of parameters of evolution, without any need to be customized for new parameter settings.

4. EVOLVABILITY AS "WEAK" OPTIMIZATION

Having shown that a very simple and robust approach can yield evolvability of fixed-degree polynomials under the L_2 loss function, we now turn to showing a much stronger result, though one that relies on decidedly "unnatural" algorithms.

The idea at the center of this section is that evolvability can reproduce any result efficiently obtainable from *approximate oracle access to* LPerf. In this section, we demonstrate this connection, which lets us then leverage the entire field of optimization algorithms towards our goal of evolvability, yielding immediate fruits at the end of this section.

As noted in the introduction, we prove this connection via an adaptation of the analysis of the analogous result from the Boolean case—which appears as Theorem 5.1 in Feldman [2009b] (see, specifically, the proof of Theorem A.3 in the appendix of that paper). The main hurdle in both cases is showing that the selection rule SelNB can efficiently simulate approximate responses to questions of the form "is $L\mathsf{Perf}_{f,D}(h)$ greater than a threshold θ?" In particular, this will be achieved in a single generation of evolution.

One difference between the real case and the Boolean case—or, more specifically, between how LPerf is defined here versus in Feldman [2009b]—is that in our case, we have no functions whose performance we know a priori, while in the Boolean case,

the function that returns an independent unbiased coin flip is guaranteed to have performance 0. Without such a reference point, evolvability has no hope of addressing such threshold queries. In lieu of an absolute benchmark like that, we instead adopt a relative benchmark, comparing performance always against $L\mathsf{Perf}(\mathbf{0})$. Namely, our evolution algorithm will function as though it had approximate oracle access to $L\mathsf{Perf}(\cdot) - L\mathsf{Perf}(\mathbf{0})$.

We note that here and for the rest of the article, we use no special properties of the $\mathbf{0}$ function, and indeed, any arbitrary function from the hypothesis class could be substituted here and throughout. We use $\mathbf{0}$ simply to avoid introducing further notation. A more meticulous reader might mentally substitute an arbitrarily-chosen element of the hypothesis class instead of $\mathbf{0}$, in each of the following results, to handle the odd but perfectly legitimate case that $\mathbf{0}$ is not in the hypothesis class C of Theorem 4.4.

We start with an overview of the intuitive idea for the construction to approximately answer, in a single generation, *threshold queries* of the form "is $L\mathsf{Perf}(h) - L\mathsf{Perf}(\mathbf{0}) > \theta$?". We assume genomes may represent probabilistic functions and, moreover, assume as a sort of induction hypothesis, that the parent's genome defines a function that is the $\mathbf{0}$ function a "large" fraction of the time.

Denoting the parent's genome by r, its performance is $L\mathsf{Perf}_{f,D}(r)$, and for a given tolerance t, the selection rule SelNB treats children very differently according to whether their observed loss is within t of $L\mathsf{Perf}_{f,D}(r)$ (neutral mutations), more than t lower than this (beneficial mutations), or more than t higher than this and doomed to be culled. Since our goal is to make the selection rule have a sharp threshold near where $L\mathsf{Perf}(h) - L\mathsf{Perf}(\mathbf{0}) \approx \theta$, and the selection rule already has sharp thresholds built-in at $L\mathsf{Perf}_{f,D}(r) \pm t$, the natural approach, as in Feldman [2009b], is to make use of these thresholds for our purposes, having r produce two types of children: the first type has (probabilistic) hypothesis function identical to the parent; the second type outputs the function $\mathbf{0}$ with probability $\frac{t}{|\theta|}$ less than its parent and h with probability $\frac{t}{|\theta|}$ more than its parent. Thus, given some bound $q = q(n, m, \frac{1}{\epsilon})$ on the total number of threshold queries we would ever need to resolve, and a lower bound on $|\theta_i|$, we can set t sufficiently small so that the expressions $\frac{t}{|\theta_i|}$ of probability mass for a sequence of q queries $\theta_1, \ldots, \theta_q$ can be readily made to sum up to a probability less than 1.

To state the result more cleanly, we introduce "weak" optimization terminology adapted from Lovász [1986].

Definition 4.1. A *μ–weak evaluation oracle* for a function $p : \mathbb{R}^k \to \mathbb{R}$ is an oracle that on input x returns a number a such that $|p(x) - a| < \mu$.

Definition 4.2. The *ν–weak function minimization problem* for a function $p : \mathbb{R}^k \to \mathbb{R}$ is that of finding an x such that $\forall y \in \mathbb{R}^k, p(y) > p(x) - \nu$. Namely, x has function value within ν of the global optimum.

Definition 4.3. A class of functions is *weakly optimizable* if there exists a randomized polynomial-time oracle algorithm A and a polynomial $\mu = \mu(\nu, \frac{1}{k})$ such that for every $\nu > 0$ and any function $p : \mathbb{R}^k \to \mathbb{R}$ in the class, A solves the ν-weak function minimization problem when given access to a $\mu(\nu, \frac{1}{k})$-weak evaluation oracle for p.

THEOREM 4.4. *If L is a loss function, C is a concept class, and $\mathcal{D}$ is a set of distributions such that there is a polynomial $b(n, m)$ that bounds $L(f_1(x), f_2(x))$ for any $f_1, f_2 \in C$ and any x in the support of a distribution in $\mathcal{D}$, and such that the class of functions $L\mathsf{Perf}_{f,D}(h) - L\mathsf{Perf}_{f,D}(\mathbf{0})$ indexed by $f \in C, D \in \mathcal{D}$ and evaluated on $h \in C$ is weakly optimizable, then $(C, \mathcal{D}, L)$ is evolvable.*

We will find it convenient to first prove this result in a restricted model referred to as "evolvability with initialization", where Definition 2.6 is modified so that instead of analyzing evolution starting with an arbitrary genome $r_0 \in R$, we instead assume a fixed starting configuration. (This is analogous to the approach of Theorem A.1 of Feldman [2009b].) Namely, we show a mutation algorithm that, starting from a fixed initial genome "$\star$", will emulate the performance of any desired optimization algorithm and yield a very good hypothesis after a reasonable number of generations; this proposition is the first step to proving this result in the standard model, where the initial genome is arbitrary.

PROPOSITION 1. *Theorem 4.4 holds under the restricted* evolvability with initialization *model, where Definition* 2.6 *is changed by replacing the phrase "any initial genome $r_0 \in R$" by "initial genome $r_0 = \star$".*

PROOF. By the assumption that performance is weakly optimizable as a function of the hypothesis, there is a randomized polynomial-time algorithm A and a polynomial $\mu = \mu(\nu, k)$ such that for every $\nu > 0$ and any $f \in C$ and $D \in \mathcal{D}$, algorithm A—when given μ-weak oracle access to $L\mathsf{Perf}_{f,D}(\cdot) - L\mathsf{Perf}_{f,D}(\mathbf{0})$—will return a hypothesis $h \in C$ that is within ν of optimal. Denoting by T a (polynomial) bound on the runtime of algorithm A, we note that we may equivalently reexpress A as a deterministic algorithm that is given as auxiliary input a T-bit uniformly-random string. Given such a deterministic algorithm, our task is to show how to simulate its operation in the evolvability framework.

As a first step, we will replace the weak evaluation oracle of Definition 4.1 with a simpler oracle, the *weak comparison oracle*, at a cost of a factor of 2 in accuracy and a logarithmic factor in the number of oracle calls. Thus, reparameterize $\frac{\mu}{2} \to \mu$ before doing the following analysis.

> *The μ–weak comparison oracle for a function p will, on given an input x and a threshold θ, return 1 if $p(x) \geq \theta + \mu$, 0 if $p(x) \leq \theta - \mu$, and either 1 or 0 otherwise.*

By assumption, b bounds the value of the function in question, that is, $L\mathsf{Perf}_{f,D}(\cdot) - L\mathsf{Perf}_{f,D}(\mathbf{0})$, and thus we have that $\log \frac{b}{\mu}$ rounds of μ-weak comparison queries will let us run a binary search to 2μ-weakly approximate the value of the function. Denote this bound by $\beta = \log \frac{b}{\mu}$, which since b and μ are polynomial, is hence polynomially bounded itself. We note, as will be important later, that such a binary search can be designed so that none of the thresholds θ_i queried ever have magnitude less than μ.

As a preliminary step, we have thus trivially shown that there is a deterministic algorithm that, when given as an auxiliary input a T-bit uniformly-random string, and given weak comparison oracle access to $L\mathsf{Perf}_{f,D}(\cdot) - L\mathsf{Perf}_{f,D}(\mathbf{0})$, will return a ν-weak minimum within $T\beta$ steps. We denote this algorithm A', and for the sake of concreteness, assume that after $T\beta$ steps have passed, it halts and outputs a hypothesis no matter what.

We now turn to the task of expressing algorithm A' in the evolvability framework. Recall that by assumption, the initial genome is uniquely fixed as "$\star$". We thus ask the mutation algorithm, when it sees the initial genome "$\star$", to produce children whose genomes each encode T bits uniformly generated at random. In each subsequent stage of mutation, these bits will be preserved in the genome; in this manner, future generations will have access to this randomly-generated T-bit string, as desired.

What remains is to describe how to simulate weak comparison queries. We will simulate one query per generation, with the result of the query being stored in the genome for the duration. At each generation, the next query to ask is computed adaptively in

terms of the results of all the previous queries, which are stored in the DNA. At time 0, the genome will consist of "$\star$", at time 1, of a T-bit random string, and at time $1+j$, we aim for the genome to consist of the concatenation of this string with a j-bit string that stores the results of the first j weak comparison queries as specified by the algorithm A' under simulation. For each such genome, we must specify how the corresponding creature responds to inputs. For the genome "$\star$" and any genome consisting solely of a T-bit string, we have the creature return the $\mathbf{0}$ vector. Otherwise, let R be the initial T bits of the genome, which is a T-bit random string by construction, and let z be the remainder of the genome, whose length we denote by j, and whose ith bit we denote z_i. Recall algorithm A' whose results we are trying to reproduce. Iteratively simulate A' starting with string R, and let (h_1, θ_1) be the first query sent to the weak comparison oracle; interpreting z_1 as the result of this query, let (h_2, θ_2) be the next query asked by A' given z_1, and so on. We thus derive (h_i, θ_i) adaptively for each $i \in \{1, \ldots, j+1\}$, all computed in polynomial time in terms of the bits $z_1, \ldots, z_j$ stored in the genome that represent responses to the first j queries.

Since each genome encodes a function $h : \mathbb{R}^n \to \mathbb{R}^m$, we define this relation now. Given a genome of length $T+j$ and an input $x \in \mathbb{R}^n$, for each $i \in \{1, \ldots, j\}$ such that $z_i = 0$, output $h_i(x)$ with probability $\frac{\mu}{|\theta_i| T\beta}$ and otherwise output the vector 0. Since $|\theta_i|$ is guaranteed to be at least μ by construction, the sum of the probabilities over the (up to) $T\beta$ generations involved will never exceed 1. Note that when $z_j = 1$, the function represented equals that of the parent. The difference in the expected performance when $z_j = 0$ versus the parent (or, equivalently, versus $z_j = 1$) equals the probability $\frac{\mu}{|\theta_j| T\beta}$ times the difference in performance between h_j and the zero function, namely, $\frac{\mu}{|\theta_j| T\beta} \left[L\mathsf{Perf}_{f,D}(\cdot) - L\mathsf{Perf}_{f,D}(\mathbf{0}) \right]$. Setting $t = \frac{\mu}{T\beta}$ means that this last expression crosses $\pm t$ (namely, $z_j = 0$ changes from being a neutral mutation to being either beneficial or negative) precisely when $L\mathsf{Perf}_{f,D}(\cdot) - L\mathsf{Perf}_{f,D}(\mathbf{0})$ crosses θ, which matches the intuition given before Definition 4.1.

A complete specification of the scheme requires only that we now specify the mutation probabilities. Namely, given a current genome Rz consisting of the (random) string R of length T and a string z of length j, where we may determine that (h_{j+1}, θ_{j+1}) is the next query to be simulated, we must choose with what probability the mutation algorithm M should output $Rz0$ as opposed to $Rz1$. Very simply, if $\theta_{j+1} < 0$, then output $Rz0$ with probability Δ and $Rz1$ with probability $1 - \Delta$, otherwise output $Rz0$ with probability $1 - \Delta$ and $Rz1$ with probability Δ, where $\Delta = \frac{\epsilon}{3\gamma}$, for $\gamma = \frac{2b}{t} + 1 + T\beta$, is chosen so that in γ rounds of coin flips, a Δ-biased coin will never land heads, except with probability less than $\epsilon/3$. (Note that in this proposition, we only use $1 + T\beta$ generations; the extra $\frac{2b}{t}$ will be used in the proof of Theorem 4.4.)

We choose the tolerance parameter t, which specifies the width of the "neutral" zone of performance, to be $t = \frac{\mu}{T\beta}$. We choose s, the size of the sample used to evaluate the empirical performance, to be large enough so that with probability $> 1 - \frac{\epsilon}{3}$, the empirical estimates are never off by more than $t\frac{\mu}{2b}$ over the entire course of $\gamma = \frac{2b}{t} + 1 + T\beta$ generations. We analyze the scheme in two cases, making use of the preceding observation that if we denote the expected performance of genome Rz by ρ, then the expected performance of $Rz1$ equals ρ, while the expected performance of $Rz0$ equals $\rho + \frac{t}{|\theta_{j+1}|} \left[L\mathsf{Perf}_{f,D}(h_{j+1}) - L\mathsf{Perf}_{f,D}(\mathbf{0}) \right]$, where as just defined, $t = \frac{\mu}{T\beta}$.

Case 1. $\theta_{j+1} < 0$. In the subcase where the weak comparison query should return "0", that is, if the expected value of $L\mathsf{Perf}_{f,D}(h_{j+1}) - L\mathsf{Perf}_{f,D}(\mathbf{0})$ is at most $\theta_{j+1} - \mu$, then the expected performance of $Rz0$ is at most $\rho + \frac{t}{|\theta_{j+1}|}(\theta_{j+1} - \mu) \leq \rho - t - \frac{t\mu}{b}$, where

b is the bound on the loss function in the theorem statement. Since by assumption, except with probability $< \frac{\epsilon}{3}$, the empirical performance will always approximate the expected performance to within $\frac{t\mu}{2b}$, we have that $Rz0$ will be found to be beneficial as compared to the parent's performance of ρ; further, $Rz1$ represents the same function as the parent and will be found to be neutral. Thus, the next genome will be $Rz0$, correctly encoding the answer to the weak comparison query. Conversely, if the weak comparison query should return "1", then by an analogous argument, the expected performance of $Rz0$ is at least $\rho - t + \frac{t\mu}{b}$, and $Rz0$ is thus either a neutral or negative mutation. Recall that by construction, in this subcase, an overwhelming majority (a $1 - \Delta$ fraction) of the mutations in this generation were constructed to be $Rz1$ instead of $Rz0$, and thus with very high probability (specifically, at least $\Delta = \frac{\epsilon}{3\gamma}$), $Rz1$ will thus be correctly chosen for the next generation.

Case 2. $\theta_{j+1} > 0$. In the subcase where the weak comparison query should return "1" then, from the preceding argument, the expected loss of $Rz0$ is at least $\rho + t + \frac{t\mu}{b}$, in which case $Rz0$ is a negative mutation, and the neutral mutation $Rz1$ will be chosen for the next generation, as desired. Otherwise, the expected loss of $Rz0$ is at most $\rho + t - \frac{t\mu}{b}$, which will be either neutral or beneficial; since the mutation algorithm will construct $Rz0$ instead of $Rz1$ an overwhelming fraction of the time $(1 - \Delta)$, with overwhelming probability, $Rz0$ will thus be correctly chosen for the next generation.

With probability at least $1 - \epsilon$, the simulation of algorithm A' will faithfully execute for $1 + T\beta$ generations, where the resulting genome Rz is the concatenation of the random string of A', and the results to all the (possibly adaptive) queries A' asked of the oracle. We conclude by stipulating that once the simulation of A' has completed, the mutation algorithm will compute the hypothesis function h that A' would output, and then with probability $1 - \Delta$ mutate the genome to a special representation, ans_h encoding h, and otherwise leave the genome unchanged. The mutation algorithm leaves genomes of the form ans_h unaltered, and when given an input $x \in \mathbb{R}^n$, a creature with genome ans_h will simply output $h(x)$. By definition, the output h of algorithm A' will have performance better than ν. Thus, either the mutation from Rz to ans_h is recognized as a beneficial mutation, having performance better than ν, or the genome already has performance within t of this already. Setting $\nu + t < \epsilon$ guarantees that we will end up with a genome that performs at least as well as ϵ, with probability $1 - \epsilon$, as desired. □

We now prove Theorem 4.4, resulting from the preceding proposition and a short argument that initialization is not necessary for the successful evolution of our algorithm. We take a simpler approach than Feldman [2009b] though at the expense of looser bounds: Feldman's construction relies on explicitly testing whether evolution has reached a satisfactory end state and otherwise reinitializing and running through evolution a second time; here, instead, we sidestep the need to ever evaluate absolute performance, and instead of reinitializing based on a clever test, simply continually reinitialize with suitably small probability at any point in the process.

PROOF OF THEOREM 4.4. Intuitively, evolution will follow the procedure set up in the proof of the proposition, which takes $1 + T\beta$ generations, except that at every generation, there is probability ρ to be defined shortly of *reinitializing*, that is, attempting to start evolution from scratch again. We will exhibit a reinitialization procedure that takes $\frac{2b}{t}$ generations, where b is the bound on the loss function from the theorem statement, and t is the tolerance threshold for the selection rule of Definition 2.5.

Thus one round of complete reinitialization and evolution will take $\frac{2b}{t} + 1 + T\beta$ generations, while in expectation, this will happen only once every $\frac{1}{\rho}$ generations. Let $\rho = \frac{1}{2}\epsilon \Big/ (\frac{2b}{t} + 1 + T\beta)$. Letting $g = \frac{1}{\rho}|1 + \log \epsilon|$, we have that after g generations, reinitialization will have occurred at least once with probability at least $1 - \frac{\epsilon}{2}$. At any moment in time after this, with probability at most $1 - \rho(\frac{2b}{t} + 1 + T\beta) = 1 - \frac{1}{2}\epsilon$, the most recent reinitialization happened at least $\frac{2b}{t} + 1 + T\beta$ generations in the past and has thus finished. Reinitialization succeeds, by the proposition, with probability at least $1 - \epsilon$. Combining these three expressions via the union bound yields that for any moment in time after g generations, the probability that evolution is at a weak optimum is at least $1 - 2\epsilon$. We thus reparameterize $2\epsilon \to \epsilon$.

To summarize, evolution could have started in any configuration, including configurations that mimic partial wrong simulations of the weak optimization algorithm A' of the preceding proposition; thus even when evolution has reached a state that, locally, looks like an "answer", it might be bogus. Thus, at every generation, with a certain small probability ρ, we give up on evolution and restart from scratch, from the desired initial condition "$\star$". This means that, over a long enough timescale, evolution will repeatedly find an optimum, settle there for a while, but then eventually abandon it and recalculate from scratch. Setting the parameters appropriately so that the probability of reinitialization is rather less than the inverse of the time it takes to properly reach an optimum, will guarantee that after an appropriate time, the probability of being at a weak optimum is high, regardless of how evolution was started.

We now present the very simple reinitialization procedure which will take $\frac{2b}{t}$ generations, a number we denote here as c. For each genome representation G in the scheme of the proposition, with the exception of "$\star$", we add copies labeled by integers $i \in \{0, \ldots c-1\}$, which we denote as G^i with the interpretation that G^i is "G after i out of c steps towards reinitialization." We modify the mutator described in the proposition so that every time it might output a certain representation G, now with probability ρ, it will instead output G^0. The mutation rule for G^i is even simpler: if $i \neq c-1$, then output G^{i+1}, and if $i = c - 1$, then output "$\star$", that is, reinitialize evolution to the starting point of the preceding proposition.

We now define how creatures with genome G^i evaluate an input x: with probability $\frac{c-i}{c}$, output whatever G would output; with probability $\frac{i}{c}$, output the $\mathbf{0}$ vector. We note that since the performance difference between $\mathbf{0}$ and any other hypothesis is at most b, the expected change in performance over any generation of reinitialization is thus at most $\frac{b}{c} = \frac{t}{2}$, namely, these are all "neutral" mutations and, by the parameter choice of the proposition, will be recognized as such, which guarantees that this procedure will operate as claimed. □

While it is fairly immediate that our notion of evolvability itself is indeed a weak optimization procedure, the surprising consequence of this theorem is the converse: that any optimization technique that is "noise-tolerant"—or in the terminology we use from Lovász [1986], "weak"—may be leveraged by evolution.

We may thus immediately leap to what is perhaps the most powerful and robust framework for optimization: the ellipsoid method. The ellipsoid method is famously known to solve any (reasonably bounded) convex optimization problem, and in particular, its weak formulations [Lovász 1986]. (Specifically, both the domain and range of the functions should be bounded.) We thus have that as long as we can arrange for $L\mathsf{Perf}$ to be convex and bounded, the associated triple $(C, \mathcal{D}, L)$ is evolvable.

As an immediate and important consequence, consider a degree-d polynomial $p\colon \mathbb{R}^n \to \mathbb{R}^m$, with D a distribution of bounded support. Then for a hypothesis h, performance is evaluated by taking a sample $x \leftarrow D$ and evaluating $L(p(x), h(x))$. As noted in the proof of Lemma 3.4 of the previous section, if h is a degree-d polynomial, considered as a vector of its $m \cdot \binom{n+d}{n}$ coefficients, then $h(x)$ is a linear function of this coefficient vector (though not linear in x!). Thus, if the loss function L is a convex function of its arguments, L will be a convex function of the coefficients of h. In short, finding the coefficients of h is a convex optimization problem when L is convex.

THEOREM 4.5. *There is a single mutation algorithm under which for any constant positive integer d and positive number r, and an arbitrary convex loss function L bounded on the radius r ball, the class of degree $\leq d$ polynomials from $\mathbb{R}^n \to \mathbb{R}^m$ with coefficients bounded by r is evolvable with respect to all distributions over the radius r ball.*

Implementing the ellipsoid algorithm of Lovász [1986] via Theorem 4.4 proves this theorem.

REFERENCES

K. Ball. 1997. An elementary introduction to modern convex geometry. *MSRI Pub. 31*.

V. Feldman. 2008. Evolvability from learning algorithms. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC'08)*. 619–628.

V. Feldman. 2009a. A complete characterization of statistical query learning with applications to evolvability. In *Proceedings of the 50th Annual Symposium on Foundations of Computer Science (FOCS'09)*.

V. Feldman. 2009b. Robustness of evolvability. In *Proceedings of the 22nd Conference on Learning Theory (COLT'09)*.

V. Feldman. 2011. Distribution-independent evolvability of linear threshold functions. In *Proceedings of the 24th Annual Conference on Learning Theory (COLT'11)*.

D. Haussler. 1992. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Inform. Computat. 100*, 1, 78–150.

A. Kalai and S. Vempala. 2006. Simulated annealing for convex optimization. *Math. Oper. Res. 31*, 2, 253–266.

V. Kanade. 2011. Evolution with recombination. In *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science (FOCS'11)*.

V. Kanade, L. Valiant, and J. Vaughan. 2010. Evolution with drifting targets. In *Proceedings of the 23rd Conference on Learning Theory (COLT'10)*.

M. Kearns. 1998. Efficient noise-tolerant learning from statistical queries. *J. ACM 25*, 6, 983–1006.

A. Livnat, C. Papadimitriou, J. Dushoff, and M. Feldman. 2008. A mixability theory for the role of sex in evolution. *Proc. Nat. Acad. Sci. 105*, 50.

A. Livnat, C. Papadimitriou, N. Pippenger, and M. Feldman. 2009. Sex, mixability, and modularity. *Proc. Nat. Acad. Sci. 107*, 4.

L. Lovász. 1986. *An Algorithmic Theory of Numbers, Graphs, and Convexity*. SIAM, Chapter 2.

L. Michael. 2012. Evolvability via the Fourier tranform. *Theor. Comput. Sci. 462*, 88–98.

Y. Nesterov and B. Polyak. 2009. Cubic regularization of Newton method and its global performance. *Math. Program. 108*, 1, 177–205.

L. Valiant. 2009. Evolvability. *J. ACM 56*, 1.

Received January 2013; revised January 2014; accepted May 2014