Robustness of Evolvability

Vitaly Feldman
IBM Almaden Research Center

Abstract

A framework for analyzing the computational capabilities and the limitations of the evolutionary process of random change guided by selection was recently introduced by Valiant [Val06]. In his framework the process of acquiring a complex functionality is viewed as a constrained form of PAC learning. In addition to the basic definition, a number of natural variants of the evolvability model were introduced by Valiant, and several others have been suggested since then [Val09, Mic07, Val08]. Understanding the relative power of these variants in terms of the efficiently evolvable function classes they define is one of the main open problems regarding the model [Val09, FV08].

We present several results that collectively demonstrate that the notion of evolvability is robust to a variety of reasonable modifications of the model. Our results show that the power of a model of evolvability essentially depends only on the fitness metric used in the model. In particular, we prove that the classes of functions evolvable in Valiant's original model are also evolvable with substantially weaker and simpler assumptions on the mutation algorithm and the selection rule and therefore a wide range of models of evolution are equivalent. Another consequence of our results if that evolvability with the quadratic loss fitness metric (or any other non-linear loss) is strictly stronger than evolvability with the linear loss fitness metric used in Valiant's basic definition.

1 Introduction

We study the model of evolvability recently introduced by Valiant [Val06]. Valiant's model addresses one of the most important and least understood aspects of the theory of evolution: how complex and adaptable mechanisms result from relatively short sequences of random mutations guided primarily by natural selection. His theory views the process of acquiring a complex behavior as a form of learning a circuit in which the feedback from the environment is provided solely by natural selection. Valiant therefore suggests that the appropriate framework for understanding the power of evolution to produce complex behavior is that of computational learning theory [Val84]. Accordingly, in his model, evolvability of a certain useful functionality is cast as a problem of learning the desired functionality through a process in which, at each step, the most "fit" candidate function is chosen from a small pool of candidates. Limits on the number of steps, the size of the pool of candidates, and the amount of computation performed at each step are imposed to make this process naturally plausible. A class of functions is considered evolvable if there exists a single mechanism that guarantees convergence to the desired function for every function in this class. Here the requirements closely follow those of the celebrated PAC learning model introduced by Valiant in 1984 [Val84]. In fact, every evolutionary algorithm (here and below in the sense defined in Valiant's model) can be simulated by an algorithm that is given random examples of the desired function.

1.1 Outline of the Model

We start by presenting a brief overview of the model. For detailed description and intuition behind the various choices made in model the reader is referred to [Val09]. The goal of the model is to specify how organisms can acquire complex mechanisms via a resource-efficient process based on random mutations and guided by fitness-based selection. The mechanisms are described in terms of the multi argument functions they implement. The fitness of such a mechanism is measured by evaluating the correlation of the mechanism with some "ideal" behavior function. The value of the "ideal" function on some input describes the most beneficial behavior for the condition represented by the input. The evaluation of the correlation with the "ideal" function is derived by evaluating the function on a moderate number of inputs drawn from a probability distribution over the conditions that arise. These evaluations correspond to the experiences of one or more organisms that embody the mechanism. A specific "ideal" function and a distribution over the domain of inputs effectively define a fitness landscape over all functions. As usual, bounds polynomial in the size of the problem are the ones considered moderate.

Random variation is modeled by the existence of an explicit algorithm that acts on some fixed representation of mechanisms and for each representation of a mechanism produces representations of mutated versions of the mechanism. The model essentially does not place any restrictions on the mutation algorithm other than it being efficiently implementable by a randomized Turing machine (naturally, the model can be extended to allow quantum computation but, following the original treatment, we restrict our attention to the classical case).

Natural selection is modeled by an explicit rule that determines the probabilities with which each of the mutations of a mechanism will be chosen to "survive" based on the fitness of all the mutations of the mechanism and the probabilities with which each of the mutations is produced by the mutation algorithm. The main rule used in the definition of the model distinguishes between beneficial, neutral and deleterious mutations based on the fitness of a mutation relative to the fitness of the original mechanism. A specific parameter referred to as the *tolerance* determines the difference in fitness that needs to be exhibited by a mutation to be considered beneficial or deleterious. Another type of rules considered is those that select one of the mutations with optimal performance. Here we discuss selection rules and their properties in more general terms and refer to the original selection rule as `SelNB`.

As can be seen from the above description, a fitness landscape (given by a specific "ideal" function and a distribution over the domain), a mutation algorithm, and a selection rule jointly determine how each step of an evolutionary process is performed. A class of functions C is considered evolvable with respect to a distribution D over the domain if there exist a representation of mechanisms R and a mutation algorithm M such that for every "ideal" function $f \in C$, a sequence of evolutionary steps starting from any representation in R and performed according to f, D, M and selection rule `SelNB` converges in a polynomial number of steps to f. The convergence is defined as achieving fitness (which is the correlation with f over D) of at least $1-\epsilon$ for some $\epsilon > 0$ referred to as the *accuracy* parameter. This process is essentially PAC learning of C over distribution D with the selection rule (rather than explicit examples) providing the only target-specific feedback.

An evolvable class of functions C represents the complexity of structures that can evolve in a single phase of evolution driven by a single "ideal" function. As pointed out by Valiant, if multiple phases are allowed with different ideal functions in succession then more complex structures can evolve. Therefore evolvability theory, analogously to learning theory, analyzes only the granularity of the structures that can evolve [Val09].

1.2 Background

The constrained way in which evolutionary algorithms have to converge to the desired function makes finding such algorithms a substantially more involved task than designing PAC learning algorithms. Initially, only the evolvability of monotone conjunctions of Boolean variables, and only when the distribution over the domain is uniform, was demonstrated (if not specified otherwise, the domain is $\{0,1\}^n$) [Val09]. Later Michael gave an algorithm for evolving decision lists over the uniform distribution that used a larger space of hypotheses and a different fitness metric over hypotheses (specifically, quadratic loss) [Mic07]. In our earlier work we showed that evolvability is, at least within polynomial limits, equivalent to learning by a natural restriction of well-studied statistical queries (SQ)[Kea98], referred to as *correlational statistical queries* [Fel08]. This result gives distribution-specific algorithms for any SQ learnable class of functions, showing that evolvability is a more powerful phenomenon than was previously believed. Coupled with communication-complexity-based lower bounds [GHR92, She07, BVdW07], this result also implies that evolvability by distribution-independent algorithms is strictly weaker than SQ learning [Fel08].

As in other computational models, such as Turing machines or computational learning, a number of specific aspects of the model can reasonably be defined in several ways. For example, Valiant discusses two ways to model natural selection, one which makes a distinction between neutral and beneficial mutations (namely `SelNB`) and another that makes a distinction only between optimal mutations and the rest [Val09]. Other natural models of selection that give weaker feedback have been proposed [Val08]. Another example is the fitness metric on which the selection is based. For simplicity, in Valiant's work only Boolean $\{-1,1\}$ functions were considered and the correlation used as the fitness metric. If one considers hypotheses that can take intermediate values between -1 and 1, other ways of modeling the performance evaluation of such functions might be appropriate. Some natural candidates would be the linear loss function and the quadratic loss function that penalizes large variance (used in Michael's work [Mic07]). Finally, some specific technical elements of the model were guided by the very few known examples of evolution algorithms and might need to be revisited once new algorithms are discovered and a deeper understanding of the model is achieved.

The question of how robust the model is under reasonable variations is fundamental to the understanding of the model and its applications [Val09]. Answering this question is also necessary to allow confident and accurate comparisons of results obtained in the different variants of the model. Further, understanding how a variation that models a specific feature of biological evolution affects the limits of evolvability might shed light on the role of that feature in the evolution as it occurred on Earth. Discussion and some progress along these lines are given by Valiant [Val09].

Valiant's framework models a complex phenomenon and is comprised of several components with relatively complex interactions among them. As a result, general analysis of the model has been fairly involved and a significant number of basic questions about the model and its variants have not been answered until now.

1.3 Summary of Our Contributions

In this work we demonstrate that the power of Valiant's model in terms of the efficiently evolvable classes it defines, remains unchanged under a variety of modifications to the model. In other words, we show that a number of possible alternative definitions of evolvability are equivalent to the specific model introduced by Valiant (to which we refer as the basic model) and hence are all equivalent to learning by correlational statistical queries (CSQ). For an unknown target function f and distribution D over the domain of the learning problem, a CSQ allows a learning algorithm

to obtain an estimate of the correlation of any Boolean function g with the target function or $\mathbf{E}_{x\sim D}[f(x)g(x)]$. This is exactly the performance metric used in the basic model of evolvability. In our earlier work we showed that evolvability is in fact equivalent to learnability by CSQs.

In our first result we prove that the equivalence between evolvability and learning by CSQs can be extended to performance metrics based on any other loss function if one replaces CSQs with queries that estimate the corresponding performance function (Th. 5.1). We then characterize the power of the obtained models showing that a simple property of the loss function determines whether the model is equivalent to learning by CSQs or to everything learnable by statistical queries (Th. 4.3, 4.4). As we have previously demonstrated, learnability by CSQs is not equivalent to learnability by SQs, which is widely regarded as almost as powerful as PAC learning [Fel08]. This, in particular, implies that evolvability with performance based on the quadratic loss function (or in fact any non-linear one) is strictly more powerful than evolvability with performance based on the correlation.

The second aspect of the model that we address is *fixed-tolerance* evolvability [Val09] (Sec. 5). Tolerance is the threshold value that determines whether a mutation is considered beneficial, neutral or deleterious. In the basic model this threshold can be adjusted as a function of the current hypothesis representation, in principle, giving the mutation algorithm some degree of control over the selection process performed on it. This ability is necessary in Valiant's evolutionary algorithm for monotone conjunctions and in our earlier general reduction from CSQ algorithms [Fel08]. Here we prove that this ability is not essential, in other words *fixed-tolerance* evolvability is as powerful as the the basic model.

We then examine the role of the selection rule. As a simple corollary of the properties of the reduction in Sec. 5, we obtain that evolvability with the selection rule that only distinguishes between the mutation with close-to-optimal performance and the rest is equivalent to evolvability with the original `SelNB` selection rule (Th. 5.1). The selection rules that we discussed earlier use a sharp threshold in the selection rule. That is, the selection decisions are not continuous: negligible differences in the fitness of a mutation might determine whether the probability of taking that mutation is 0 or 1. One of the most interesting questions about Valiant's model is whether and how a non-trivial evolutionary algorithm can be obtained if one uses a selection rule in which the probability of survival is a smooth function of the mutation's initial frequency and the performance of all the available mutations [Val08]. To answer this question we define a general property of selection rules, that, informally, requires that the selection rule weakly "favors" mutations with sufficiently higher performance. We then show that every CSQ learning algorithm for a concept class C can be transformed into a mutation algorithm that, when started in a specific initial representation, will evolve C with any selection rule satisfying the property we defined (Th. 6.1). This weaker form of evolvability is referred to as *evolvability with initialization* [Val09]. To establish this result we show a new general transformation from a CSQ algorithm to a CSQ algorithm that relies on a very weak form of CSQs. The resulting CSQ algorithm is then transformed into a mutation algorithm using techniques similar to those given in Section 5.

We note that there are essentially no prior results establishing equivalence of different models of evolvability. Previous work [Val09] has only considered direct reductions between algorithms in different models of evolvability, and only a single one-way implication was established.

Another significant product of this work is a simplification of the description of the model. The first simplification results from replacing the variable tolerance parameter by fixed tolerance as described above. We also show that a number of simplifications can be made to the definition of the mutation algorithm without loss of generality and provide a definition of the model that facilitates the discussion of the roles of the different components of the model. In this sense our work is analogous to simplifications made to the PAC learning model by Haussler *et al.* [HMLW91].

Finally, to complement the abstract discussion of the power of the model by a concrete example of an evolutionary algorithm, we show that the concept class of singletons is evolvable distribution independently. This is the first non-trivial class of functions for which distribution-independent evolvability is shown. This, in particular, resolves another open problem in Valiant's work [Val09].

1.4 Relation to Prior Work

With respect to other studies of evolution we note the following. Valiant's framework is the first one that directly addresses the complexity, in a precisely defined sense, of mechanisms that can provably result from computationally-feasible evolutionary processes without any need for unlikely events to occur. Therefore the results in this work and the model in general are not directly comparable to the extensive studies of evolution in biology and evolution-inspired optimization methods (such as *genetic* algorithms) [Wri78, BFM97, Weg01]. Limitations on evolutionary algorithms and the additional structure they possess also distinguish this direction from the study of the complexity of optimization problems solvable efficiently by local search (referred to as PLS) [JPY88].

2 Preliminaries

For a positive integer ℓ, let $[\ell]$ denote the set $\{1, 2, \ldots, \ell\}$.

For a *domain* X, a *concept class* over X is a set of $\{-1, 1\}$-valued functions over X referred to as *concepts*. A concept class together with a specific way to represent all the functions in the concept class is referred to as a *representation class*. We only consider *efficiently evaluatable* representation schemes, that is schemes, for which there exists an algorithm, that given a representation of a function g and a point $x \in X$, computes $g(x)$ in time polynomial in the length of x and the length of the representation of g. Whenever the meaning is clear from the context, we use one symbol to refer to both a function and its representation. Similarly, we refer to a representation class as just a concept class whenever a simple representation scheme is implicit in the definition of the concept class.

There is often a complexity parameter n associated with the domain X and the concept class C such as the number of Boolean variables describing an element in X or the number of real dimensions. In such a case it is understood that $X = \bigcup_{n \geq 1} X_n$ and $C = \bigcup_{n \geq 1} C_n$. We drop the subscript n when it is clear from the context. In some cases it useful to consider another complexity parameter associated with C: the minimum description length of f under the representation scheme of C. Here, for brevity, we assume that n (or a fixed polynomial in n) bounds the description length of all functions in C_n.

Let $\mathcal{F}_1^\infty$ denote the set of all functions from X to $[-1, 1]$ (that is all the functions with L_∞ norm bounded by 1). In addition to deterministic functions, we use hypotheses computing randomized (or probabilistic) real-valued functions (see [Hau92, KS94, Yam98] for detailed discussions of learning and use of such functions). The value of a randomized function Φ at point $x \in X$ is a real-valued random variable which we denote by $\Phi(x)$. A randomized function can be evaluated using a randomized algorithm. As in the case of deterministic functions, we will only consider representations of randomized functions that allow efficient evaluation.

2.1 PAC Learning

The models we consider are based on the well-known PAC learning model introduced by Valiant [Val84]. Let C be a representation class over X. In the basic PAC model a learning algorithm is given examples of an unknown function f from C on points randomly chosen from some unknown

distribution D over X and should produce a hypothesis h that approximates f. Formally, an *example oracle* $\mathrm{EX}(f, D)$ is an oracle that upon being invoked returns an example $\langle x, f(x)\rangle$, where x is chosen randomly with respect to D, independently of any previous examples.

An algorithm is said to PAC learn C in time t if for every $\epsilon > 0$, $\delta > 0$, $f \in C$, and distribution D over X, the algorithm given ϵ, δ, and access to $\mathrm{EX}(f, D)$ outputs, in time t and with probability at least $1 - \delta$, a hypothesis h that is evaluatable in time t and satisfies $\mathbf{Pr}_D[f(x) = h(x)] \geq 1 - \epsilon$. We say that an algorithm learns C by a representation class H if the hypotheses output by the algorithm use the representation scheme of H.

The running time t is allowed to depend on n, $1/\epsilon$ and $1/\delta$. We say that an algorithm *efficiently* learns C when t is upper bounded by a polynomial in n, $1/\epsilon$ and $1/\delta$.

A number of variants of this basic framework are commonly considered. The basic PAC model is also referred to as *distribution-independent* learning to distinguish it from *distribution-specific* PAC learning in which the learning algorithm is required to learn only with respect to a single distribution D known in advance. More generally, one can restrict the target distribution to come from a class of distributions $\mathcal{D}$ known in advance (such as product distributions) to capture both scenarios. We refer to this case as learning over $\mathcal{D}$.

2.2 The Statistical Query Learning Model

In the *statistical query model* of Kearns [Kea98] the learning algorithm is given access to $\mathrm{STAT}(f, D)$ – a *statistical query oracle* for target concept f with respect to distribution D instead of $\mathrm{EX}(f, D)$. A query to this oracle is a function $\psi : X \times \{-1, 1\} \to \{-1, 1\}$. The oracle may respond to the query with any value v satisfying $|\mathbf{E}_D[\psi(x, f(x))] - v| \leq \tau$ where $\tau \in [0, 1]$ is a real number called the *tolerance* of the query. For convenience, we allow the query functions to be real-valued in the range $[-1, 1]$. As it has been observed by Aslam and Decatur [AD98], this extension is equivalent to the original SQ model.

An algorithm A is said to learn C in time t from statistical queries of tolerance τ if A PAC learns C using $\mathrm{STAT}(f, D)$ in place of the example oracle. In addition, each query ψ made by A has tolerance τ and can be evaluated in time t.

The algorithm is said to (efficiently) SQ learn C if t is polynomial in n, $1/\epsilon$ and $1/\delta$, and τ is lower bounded by the inverse of a polynomial in n and $1/\epsilon$.

3 Model of Evolvability

In this section we give the definitions of the basic evolvability model [Val09] and a number of its variants. Our goal is to both describe the original definition of the model of evolvability and to present a somewhat different way to define the model in order to facilitate a more general discussion of Valiant's framework and to provide a simpler description of the basic model. Our presentation follows the outline in Section 1.1 in dividing the definition of the model into the descriptions of fitness metric (Sec. 3.2), mutation algorithm (Sec. 3.3), selection rule (Sec. 3.4), and the definition of evolvability of a function class (Sec. 3.5). Finally, in Section 3.6 we demonstrate that a new way to define the model is equivalent to the original definition via simple and direct reductions.

3.1 Original Definition

Let f denote the unknown "ideal" function and D be a distribution over the domain X. The *performance* of a hypothesis h relative to the target f and distribution D is defined to be $\mathtt{Perf}_f(h, D) = \mathbf{E}_{x \sim D}[f(x) \cdot h(x)]$. Note that for Boolean hypotheses (both deterministic and randomized), this is

equivalent to measuring the probability of agreement of f and h (as $\mathbf{E}_D[f(x)\cdot h(x)] = 2\cdot\mathbf{Pr}_D[f(x) = h(x)]-1$). For an integer s, the *empirical performance* $\mathtt{Perf}_f(h, D, s)$ of h is a random variable that equals $\frac{1}{s}\sum_{i\in[s]} f(z_i)\cdot h(z_i)$ for $z_1, z_2, \ldots, z_s \in X$ chosen randomly and independently according to D.

Let R be a representation class of functions over X. The class R denotes the space of hypotheses available to the evolving organism. Each representation $r \in R$ defines a possibly randomized Boolean function on X.

The first part of the definition is the *neighborhood* of each representation r which specifies all the possible mutations of r and probabilities of their generation.

Definition 3.1 *For a polynomial $p(\cdot,\cdot)$ and a representation class R, a* p-neighborhood *N on R is a pair (M_1, M_2) of randomized polynomial time Turing machines such that on input the numbers n and $\lceil 1/\epsilon\rceil$ (in unary) and a representation $r \in R$ act as follows: M_1 outputs all the members of a set $\mathtt{Neigh}_N(r,\epsilon) \subseteq R$, that contains r and may depend on random coin tosses of M_1, and has size at most $p(n, 1/\epsilon)$. If M_2 is then run on this output of M_1, it in turn outputs one member of $\mathtt{Neigh}_N(r,\epsilon)$, with member r_1 being output with a probability $\mathbf{Pr}_N(r, r_1) \geq 1/p(n, 1/\epsilon)$. For a set $T \subseteq \mathtt{Neigh}_N(r,\epsilon)$, we denote $\mathbf{Pr}_N(r, T) = \sum_{r'\in T}\mathbf{Pr}_N(r, r')$.*

The conditions on M_1 and M_2 ensure that for each r the number of variants, determined by M_1, that can be searched effectively is not unlimited, because the population at any time is not unlimited, and that a significant number of experiences with each variant, generated by M, must be available so that differences in performance can be detected reliably.

The next part of the definition describes how a single step of selection is performed.

Definition 3.2 *For error parameter ϵ, positive integers n and s, an ideal function $f \in C$, a representation class R with $p(n, 1/\epsilon)$-neighborhood N on R, a distribution D, a representation $r \in R$ and a real number t, the* mutator $\mathtt{Mu}(f, p(n, 1/\epsilon), R, N, D, s, r, t)$ *outputs a random variable that takes a value r_1 determined as follows. For each $r' \in \mathtt{Neigh}_N(r,\epsilon)$, it first computes an empirical value of $v(r') = \mathtt{Perf}_f(r', D, s)$. Let $\mathtt{Bene} = \{r' \mid v(r') \geq v(r) + t\}$ and $\mathtt{Neut} = \{r' \mid |v(r') - v(r)| < t\}$. Then*

(i) if $\mathtt{Bene} \neq \emptyset$ then output $r_1 \in \mathtt{Bene}$ with probability $\mathbf{Pr}_N(r, r_1)/\mathbf{Pr}_N(r, \mathtt{Bene})$;

(ii) if $\mathtt{Bene} = \emptyset$ then output $r_1 \in \mathtt{Neut}$ with probability $\mathbf{Pr}_N(r, r_1)/\mathbf{Pr}_N(r, \mathtt{Neut})$.

In this definition a distinction is made between beneficial and neutral mutations as revealed by a set of s experiments. If some beneficial mutations are available one is chosen according to the relative probabilities of their generation by M_2. If none is available then one of the neutral mutations is taken according to the relative probabilities of their generation by M_2. Since $r \in \mathtt{Neigh}_N(r,\epsilon)$, r will always be empirically neutral, by definition, and hence $\mathtt{Neut}$ will be nonempty.

Finally, we define how representations evolve to functions in C. We say that tolerance parameter $t(r, n, 1/\epsilon)$ is *poly-bounded* if $t(r, n, 1/\epsilon)$ is computable by a randomized TM T in time polynomial in n and $1/\epsilon$, and satisfies

$$1/\mathrm{tu}^{\eta}(n, 1/\epsilon) \leq t(r, n, 1/\epsilon) \leq 1/\mathrm{tu}(n, 1/\epsilon)$$

for all n, $r \in R$ and $\epsilon > 0$, where $\mathrm{tu}(\cdot,\cdot)$ is a polynomial and η is a fixed constant.

Definition 3.3 *For a concept class C, distribution D, polynomials $p(\cdot,\cdot)$ and $s(\cdot,\cdot)$, a poly-bounded tolerance $t(r, n, 1/\epsilon)$, representation class R, $p(n, 1/\epsilon)$-neighborhood N on R, C is said to be t-evolvable by $(p(n, 1/\epsilon), R, N, s(n, 1/\epsilon))$ over D if there exists a polynomial $g(\cdot,\cdot)$ such that for every*

n, $f \in C$, $\epsilon > 0$, and every $r_0 \in R$, with probability at least $1-\epsilon$, a sequence $r_0, r_1, r_2, \ldots$, where

$$r_i \leftarrow \mathtt{Mu}(f, p(n,1/\epsilon), R, N, D, s(n,1/\epsilon), r_{i-1}, t(r_{i-1}, n, 1/\epsilon))$$

will have $\mathtt{Perf}_f(r_{g(n,1/\epsilon)}, D) \geq 1-\epsilon$.

The polynomial $g(n, 1/\epsilon)$ upper bounds the number of generations needed for the evolution process.

Definition 3.4 *A concept class C is evolvable over a class of distributions $\mathcal{D}$ if there exist polynomials $p(\cdot,\cdot)$ and $s(\cdot,\cdot)$, a poly-bounded tolerance $t(r,n,1/\epsilon)$, representation class R, $p(n,1/\epsilon)$-neighborhood N on R, such that C is t-evolvable by $(p(n,1/\epsilon), R, N, s(n,1/\epsilon))$ over every $D \in \mathcal{D}$.*

Note that the basic model allows the tolerance t to vary with the representation. In Section 5 we prove that the dependence on r is not essential. A concept class C is said to be *fixed-tolerance* evolvable if it is evolvable for t that is independent of r.

A more relaxed notion of evolvability requires convergence only when the evolution starts from a single fixed representation r_0. Such evolvability is referred to as evolvability *with initialization*. Evolvability without initialization allows for successive phases of evolution without a need to restart. This, for example, would allow an organism to adapt if the target function changes.

A concept class C is evolvable if it is evolvable over all distributions (by a single evolutionary algorithm). We emphasize this by saying *distribution-independently* evolvable.

We now provide a more general way to define the model.

3.2 Measuring Performance

In the original definition the performance of a Boolean hypothesis h relative to the target f and distribution D effectively measures the probability of agreement of f and h. Here we also consider real-valued hypotheses and hence will need to extend the notion of performance to real-valued functions. A *loss function* is usually used to formalize the notion of "closeness", in learning theory. For functions with range Y, a loss function L is a non-negative mapping $L : Y \times Y \to \mathbb{R}^+$. $L(y, y')$ measures the "distance" between the desired value y and the predicted value y' and is interpreted as the loss suffered due to predicting y' when the correct prediction is y. Commonly considered loss functions are *discrete loss* $L_=$, *linear loss* $L_1(y, y') = |y' - y|$ and the *quadratic loss* $L_Q(y, y') = (y' - y)^2$. Since inputs are coming from a distribution D we can define the *expected loss* of a hypothesis h relative to the target f and distribution D: $\mathbf{E}_{x \sim D}[L(f(x), h(x))]$. For brevity we denote it as $\mathbf{E}_D[L(f,h)]$. When considering randomized hypotheses, the expectation in the definition of the expected loss is also taken over the random values of the hypothesis. That is, for a randomized hypothesis Φ, $\mathbf{E}_D[L(f,\Phi)] = \mathbf{E}_{x \sim D, \Phi}[L(f(x), \Phi(x))]$. We will only consider loss functions $L : \{-1,1\} \times [-1,1] \to \mathbb{R}^+$ that are efficiently computable and satisfy

1. $L(1,-1) = L(-1,1)$ and $L(-1,-1) = L(1,1) = 0$,
2. monotone: for all $y, y' \in [-1,1]$, if $y \leq y'$ then $L(-1,y) \leq L(-1,y')$ and $L(1,y') \leq L(1,y)$.
3. non-degenerate: for every $y \in [-1,1]$, $L(1,y) + L(-1,y) > 0$.

We refer to such loss functions as *admissible*. Let L be an admissible loss function. For consistency with Valiant's definition of performance, we define a corresponding performance function as

$$L\mathtt{Perf}_f(\Phi, D) = 1 - 2 \cdot \mathbf{E}_D[L(f,\Phi)]/L(-1,1) \ .$$

Where Φ is a (possibly randomized) function and the factor $2/L(-1,1)$ is used to normalize $L\mathtt{Perf}_f(\Phi, D)$ to be in the same range as $\mathtt{Perf}_f(\Phi, D)$ (namely $[-1,1]$).

Remark 3.5 *For any admissible loss function L and a Boolean function Φ, $L\mathtt{Perf}_f(\Phi, D) = \mathtt{Perf}_f(\Phi, D)$.*

This remark implies that all performance functions derived from an admissible loss function are equivalent on Boolean functions. In particular, this implies that Valiant's and our earlier results [Val09, Fel08] hold for any loss function. We also note that for any function Φ taking values in $[-1, 1]$,

$$L_1\mathtt{Perf}_f(\Phi, D) = \mathtt{Perf}_f(\Phi, D) = \mathtt{Perf}_f(\Phi', D) = \mathtt{Perf}_f(\phi, D) \ ,$$

where $\Phi'(x)$ is a Boolean randomized function and $\phi(x)$ a deterministic (real-valued) function such that for every x, $\mathbf{E}_{\Phi'}[\Phi'(x)] = \phi(x) = \mathbf{E}_{\Phi}[\Phi(x)]$. This implies that evolvability with the linear loss L_1 (over real-valued hypotheses) is equivalent to evolvability with $\mathtt{Perf}$ over Boolean randomized hypotheses [Fel08]. It will be convenient to represent Boolean randomized hypotheses as deterministic real-valued ones when evolving with respect to $\mathtt{Perf}$.

For an integer s, function Φ and loss function L, the *empirical performance* $L\mathtt{Perf}_f(\Phi, D, s)$ of Φ is a random variable that equals $1 - \frac{1}{s}\frac{2}{L(-1,1)}\sum_{i\in[s]} L(f(z_i), \Phi(z_i))$ for $z_1, z_2, \ldots, z_s \in X$ chosen randomly and independently according to D.

One can also consider hypotheses with the range outside of $[-1, 1]$ (as is done in Michael's work [Mic07]). This is not required to prove our results. We also note that when evolving to a Boolean target function, outputting a value v outside of $[-1, 1]$ will always incur higher loss than outputting $\mathtt{sign}(v)$ and hence values outside of $[-1, 1]$ are always detrimental.

3.3 Mutation Algorithm

The description of a mutation algorithm A consists of the definition of the representation class R of possibly randomized hypotheses in $\mathcal{F}_1^\infty$ and the description of an algorithm that for every $r \in R$, outputs a random mutation of r. More formally,

Definition 3.6 *A* mutation algorithm *A is defined by a pair (R, M) where*

- *R is a representation class of functions over X with range in $[-1, 1]$.*
- *M is a randomized polynomial (in n and $1/\epsilon$) time Turing machine that given $r \in R$ and $1/\epsilon$ as input outputs a representation $r_1 \in R$ with probability $\mathbf{Pr}_A(r, r_1)$. The set of representations that can be output by $M(r, \epsilon)$ is referred to as the* neighborhood *of r for ϵ and denoted by $\mathtt{Neigh}_A(r, \epsilon)$. For a set $T \subseteq \mathtt{Neigh}_A(r, \epsilon)$, we denote $\mathbf{Pr}_A(r, T) = \sum_{r'\in T} \mathbf{Pr}_A(r, r')$.*

This definition is the analogue of p-neighborhood in the basic model.

3.4 Selection Rules

We now define several models of how natural selection acts on a mutation algorithm evolving towards a specific target function f over a specific distribution D. A *selection rule* is essentially a transformation of the distribution over representations in R output by the mutation algorithm that favors representation with higher performance. In other words, selection rule $\mathtt{Sel}$ has access to r and M, can evaluate the empirical performance of representations and outputs the "surviving" representation. The output of a selection rule $\mathtt{Sel}$ can be randomized and we will only consider selection rules that can be implemented efficiently and output only representations in the neighborhood of r or $\perp$ that means that no mutation has survived.

We first describe a selection rule that is analogous to the mutator in the basic model but can be applied to any mutation algorithm and not only to a p-neighborhood.

Definition 3.7 *For a loss function L, tolerance t, candidate pool size p, sample size s, selection rule* $\texttt{SelNB}[L,t,p,s]$ *is an algorithm that for any function f, distribution D, mutation algorithm $A=(R,M)$, a representation $r \in R$, accuracy ϵ,* $\texttt{SelNB}[L,t,p,s](f,D,A,r)$ *outputs a random variable that takes a value r_1 determined as follows. First run $M(r,\epsilon)$ p times and let Z be the set of representations obtained. For $r' \in Z$, let $\mathbf{Pr}_Z(r')$ 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 $v(r') = L\texttt{Perf}_f(r',D,s)$. Let $\texttt{Bene}(Z) = \{r' \mid v(r') \geq v(r)+t\}$ and $\texttt{Neut}(Z) = \{r' \mid |v(r')-v(r)| < t\}$. Then*

(i) if $\texttt{Bene}(Z) \neq \emptyset$ then output $r_1 \in \texttt{Bene}$ with probability $\mathbf{Pr}_Z(r_1)/\sum_{r' \in \texttt{Bene}(Z)} \mathbf{Pr}_Z(r')$;

(ii) if $\texttt{Bene}(Z) = \emptyset$ and $\texttt{Neut}(Z) \neq \emptyset$ then output $r_1 \in \texttt{Neut}(Z)$ with probability $\mathbf{Pr}_Z(r_1)/\sum_{r' \in \texttt{Neut}(Z)} \mathbf{Pr}_Z(r')$.

(iii) If $\texttt{Neut}(Z) \cup \texttt{Bene}(Z) = \emptyset$ then output $\perp$.

The main difference between this selection rule and the mutator with the same parameters is that in the basic model the definition of p-neighborhood requires that M_1 produce a neighborhood of polynomial size whereas mutation algorithms are not explicitly restricted and might produce an exponential number of representations each with negligible probability. Hence $\texttt{SelNB}[L,t,p,s]$ first samples the mutations produced by $M(r,\epsilon)$ and then uses the empirical neighborhood instead of the actual $\texttt{Neigh}_A(r,\epsilon)$. The second difference is that in the basic model $r \in \texttt{Neigh}_N(r,\epsilon)$ ensures that the performance does not drop by more than the value of the threshold t. Definition 3.7 enforces this explicitly by outputting $\perp$ and thereby stopping process of evolution when no neutral or beneficial mutations are present.

Valiant also considers selection rules that always choose a mutation with empirical performance within tolerance t of the best one in the neighborhood $\texttt{Neigh}_N(r,\epsilon)$. He defines *evolvability with optimization* as evolvability with *every* selection rule that has this property and shows that every concept class evolvable with optimization is evolvable in the basic model. Here we define a particular optimizing analogue of SelNB that outputs all the mutations whose performance is within t of the best, each with probability proportional to their probability in the empirical neighborhood of r.

Definition 3.8 *For a loss function L, tolerance t, candidate pool size p and sample size s, the selection rule* $\texttt{SelOpt}[L,t,p,s]$ *is an algorithm that for any function f, distribution D, mutation algorithm A, a representation $r \in R$, accuracy ϵ,* $\texttt{SelOpt}[L,t,p,s](f,D,A,r)$ *outputs a random variable that takes a value r_1 determined as follows. First run $M(r,\epsilon)$ p times and let Z be the set of representations obtained. For $r' \in Z$, let $\mathbf{Pr}_Z(r')$ be the relative frequency with which r' was generated among the s observed representations. For each $r' \in Z \cup \{r\}$, compute an empirical value of performance $v(r') = L\texttt{Perf}_f(r',D,s)$. Let $p^* = \max_{r' \in Z \cup \{r\}}\{v(r')\}$ and let $\texttt{Opt} = \{r' \mid r' \in Z,\ v(r') \geq p^* - t\}$. Output $r_1 \in \texttt{Opt}$ with probability $\mathbf{Pr}_Z(r_1)/\sum_{r' \in \texttt{Opt}} \mathbf{Pr}_Z(r')$. If $\texttt{Opt}$ is empty $\texttt{SelOpt}[L,t,p,s]$ outputs $\perp$.*

Note that like $\texttt{SelNB}[L,t,p,s]$, $\texttt{SelOpt}[L,t,p,s]$ does not allow the performance to drop by more than t. As we will demonstrate in Section 5, evolvability with this simpler SelOpt is equivalent to evolvability with SelNB.

3.4.1 Smooth Selection Rules

The selection rules that we discussed earlier use a sharp threshold in the transformation of probabilities. That is, the selection decisions are not continuous: negligible differences in the empirical performance of a mutation might determine whether the probability of taking that mutation is 0

or 1. In many realistic conditions the probability that a certain mutation is adopted is a smooth function of its initial frequency and the performance of all the available mutations. We informally refer to such selection rules as *smooth*.

Instead of considering various possible selection rules that might arise we define a general property of selection rules that is satisfied by a variety of smooth selection rules and, as we prove later, does not reduce the power of evolvability. Informally, the property of a selection rule that we need is that when selecting from between two candidate hypotheses r_1 and r_2 such that performance of r_2 is "observably" higher than the performance of r_1, the output probability of r_2 is "observably" higher than the output probability of r_1. In addition, the selection rule has to be "representation-neutral"; that is if two representations compute the same function then their relative frequencies in the output should remain the same. Finally, we assume that the selection rule does not output $\perp$ if the neighborhood of r contains representations with performance higher or equal than the performance of r with significant probability (we use $1/2$ for concreteness).

Definition 3.9 *Let Π be a selection rule. For real numbers $t > 0$ and $\gamma > 0$ we say that Π is (t, γ)-*distinguishing *if for every mutation algorithm $A = (R, M)$, $r \in R$, $\epsilon > 0$, $r_1, r_2 \in \mathtt{Neigh}_A(r, \epsilon)$*

1. *$v(r_1) < v(r_2) - t$ implies that $\frac{\mathbf{Pr}_{A,\Pi}(r,r_1)}{\mathbf{Pr}_{A,\Pi}(r,r_2)} < (1-\gamma)\frac{\mathbf{Pr}_A(r,r_1)}{\mathbf{Pr}_A(r,r_2)}$, where $v(r')$ denoted the performance of r' and $\mathbf{Pr}_{A,\Pi}(r, r')$ denotes the probability that Π applied to A on r and ϵ outputs r'.*

2. *$r_1 \equiv r_2$ implies that $\frac{\mathbf{Pr}_{A,\Pi}(r,r_1)}{\mathbf{Pr}_{A,\Pi}(r,r_2)} = \frac{\mathbf{Pr}_A(r,r_1)}{\mathbf{Pr}_A(r,r_2)}$;*

3. *for $P = \{r' \in \mathtt{Neigh}_A(r, \epsilon) \mid v(r') \geq v(r)\}$, if $\mathbf{Pr}_A(r, P) \geq 1/2$ then $\mathbf{Pr}_{A,\Pi}(r, \perp) = 0$.*

(We treat $\frac{0}{0}$ as 0).

Note that the output of a (t, γ)-distinguishing selection rule is not constrained at all when $0 < |q_1 - q_2| \leq t$. For brevity we also say that a selection rule is t-distinguishing if it is $(t, 1)$-distinguishing.

3.5 Convergence

A concept class C is defined to be evolvable by a mutation algorithm A guided by a selection rule $\mathtt{Sel}$ over distribution D if for every target concept $f \in C$, mutation steps as defined by A and guided by $\mathtt{Sel}$ will converge to f.

Definition 3.10 *For concept class C over X, distribution D, mutation algorithm A, loss function L and a selection rule $\mathtt{Sel}$ based on $L\mathtt{Perf}$ we say that the class C is evolvable over D by A in $\mathtt{Sel}$ if there exists a polynomial $g(n, 1/\epsilon)$ such that for every n, $f \in C$, $\epsilon > 0$, and every $r_0 \in R$, with probability at least $1 - \epsilon$, a sequence $r_0, r_1, r_2, \ldots$, where $r_i \leftarrow \mathtt{Sel}(f, D, A, r_{i-1})$ will have $L\mathtt{Perf}_f(r_{g(n,1/\epsilon)}, D) > 1 - \epsilon$. We refer to the algorithm obtained as* evolutionary algorithm $(A, \mathtt{Sel})$.

One particular important issue addressed by the model is the ability of an organism to adjust to a change of the target function or distribution without sacrificing the performance of the current hypothesis (beyond the decrease caused by the change itself). Formally, we say that an evolutionary algorithm $(A, \mathtt{Sel})$ evolves C over D *monotonically* if with probability at least $1 - \epsilon$, for every $i \leq g(n, 1/\epsilon)$, $L\mathtt{Perf}_f(r_i, D) \geq L\mathtt{Perf}_f(r_0, D)$, where $g(n, 1/\epsilon)$ and $r_0, r_1, r_2, \ldots$ are defined as above.

3.6 Equivalence of Definitions

The main reason why we consider the new definition is that it clearly differentiates the roles of the mutation algorithm and the selection rules whereas, in the original definition, efficiency constraints on the selection process were part of the definition of a p-neighborhood and a mutator. This differentiation is useful for a more general discussion of the model and is also move convenient notationally. We now prove that the original definition is equivalent to a special case of our definition.

To achieve this we first show that under certain simple conditions $\texttt{SelNB}$ effectively outputs mutations according to their true probabilities of generation rather than the empirical ones.

Lemma 3.11 *Let $A = (R, M)$ be a mutation algorithm. For any $f, D, r \in R$ and numbers ϵ, p, s and t the probability that $\texttt{SelNB}[L_1, t, p, s](f, D, A, r)$ outputs $r_1 \in \texttt{Neigh}_A(r, \epsilon)$ conditioned on $\texttt{Bene}(Z) \neq \emptyset$ is $\mathbf{Pr}_A(r, r_1)/\mathbf{Pr}_A(r, \texttt{Bene})$ if $r_1 \in \texttt{Bene}$ and 0 otherwise. Here $\texttt{Bene}$ is itself a random variable defined in Definition 3.2, in other words, conditioned on $\texttt{Bene}(Z) \neq \emptyset$, r_1 is output with probability*

$$\sum_{r_1 \in T \subseteq \texttt{Neigh}_A(r,\epsilon)} \frac{\mathbf{Pr}_A(r, r_1)}{\mathbf{Pr}_A(r, T)} \mathbf{Pr}[\texttt{Bene} = T].$$

Similarly, the probability that $\texttt{SelNB}[L_1, t, p, s](f, D, A, r)$ outputs r_1 conditioned on $\texttt{Bene}(Z) = \emptyset$ and $\texttt{Neut}(Z) \neq \emptyset$ is $\mathbf{Pr}_A(r, r_1)/\mathbf{Pr}_A(r, \texttt{Neut})$ if $r_1 \in \texttt{Neut}$ and 0 otherwise.

Proof: We first observe that we can assume that $\texttt{SelNB}[L_1, t, p, s](f, D, A, r)$ evaluates an empirical performance $v(r')$ for every $r' \in \texttt{Neigh}_A(r, \epsilon) \cup \{r\}$ (and not just for $r' \in Z \cup \{r\}$). This does not affect the behavior of $\texttt{SelNB}$ since each $v(r')$ is an independent random variable. Now let $r'_1, r'_2, \ldots, r'_p$ denote the p representations obtained by sampling $M(r, 1/\epsilon)$. By the definition, $\texttt{Bene}(Z) = \texttt{Bene} \cap Z$ and therefore the first rule of $\texttt{SelNB}$ effectively means that if at least one representation in $\texttt{Bene}$ is present in the p samples from M then $\texttt{SelNB}$ outputs r'_i where i is chosen randomly and uniformly from $\{j \mid r'_j \in \texttt{Bene}\}$. For each i in this set, $r'_i = r_1$ with probability $\mathbf{Pr}_A[r'_i = r_1 \mid r'_i \in \texttt{Bene}]$ that equals $\mathbf{Pr}_A(r, r_1)/\mathbf{Pr}_A(r, \texttt{Bene})$ if $r_1 \in \texttt{Bene}$ and equals 0 otherwise. Therefore, r_1 is output with the same probability for a randomly chosen i. This gives us the first part of the lemma. The second part of the lemma is obtained analogously. □

We can now prove that $\texttt{SelNB}[L_1, t, p, s]$ when applied to a p'-neighborhood N (in the same fashion as it is applied to a mutation algorithm) produces almost the same distribution as the mutator $\texttt{Mu}(f, p', R, N, D, s, r, t)$ if p is a sufficiently large polynomial (for the purposes of this result we abuse our notation and write $\texttt{SelNB}[L_1, t, p, s](f, D, N, r)$ to refer to the application of $\texttt{SelNB}[L_1, t, p, s]$ to N).

Corollary 3.12 *Let N be a p'-neighborhood on R. For any $f, D, r \in R$ and numbers ϵ, p, s and t the statistical distance between the output of $\texttt{SelNB}[L_1, t, p, s](f, D, N, r)$ and the output of the mutator $\texttt{Mu}(f, p', R, N, D, s, r, t)$ is at most $(1 - 1/p')^p \leq e^{-p/p'}$.*

Proof: As in Lemma 3.11, we can assume that $\texttt{SelNB}$ evaluates empirical performance of all representations in $\texttt{Neigh}_N(r, \epsilon)$ and obtains the sets $\texttt{Bene}$ and $\texttt{Neut}$. If $\texttt{Bene} \neq \emptyset$ then, by properties of N, $\mathbf{Pr}_N(r, \texttt{Bene}) \geq 1/p'$. This implies that with probability at least $(1 - 1/p')^p$, $\texttt{Bene}(Z) \neq \emptyset$. By Lemma 3.11, when $\texttt{Bene}(Z) \neq \emptyset$ the output of $\texttt{SelNB}$ is distributed exactly in the same way as the output of $\texttt{Mu}(f, p', R, N, D, s, r, t)$ for the same set $\texttt{Bene}$. Similarly if $\texttt{Bene} = \emptyset$ and $\texttt{Neut} \neq \emptyset$ then $\texttt{Bene}(Z) = \emptyset$ and, with probability at least $(1 - 1/p')^p$, $\texttt{Neut}(Z) \neq \emptyset$. In this case the output distribution of $\texttt{SelNB}$ is the same as the output distribution of $\texttt{Mu}(f, p', R, N, D, s, r, t)$. These are

the only two possibilities and therefore we obtain that the statistical distance between the output distributions of $\texttt{SelNB}[L_1, t, p, s](f, D, N, r)$ and $\texttt{Mu}(f, p', R, N, D, s, r, t)$ is at most $(1 - 1/p')^p$. $\square$

We now prove that every evolutionary algorithm $(A = (R, M), \texttt{SelNB}[L_1, t, p, s])$ for evolving concept class C over D, can be easily converted to a p'-neighborhood on R' that will evolve C over D with tolerance t in the basic model and vice versa.

Theorem 3.13 *For a concept class C over X, a distribution D, a polynomial $s(\cdot, \cdot)$ and an inverse polynomial $t(\cdot, \cdot)$, there exist a polynomial $p(\cdot, \cdot)$ and a mutation algorithm $A = (R, M)$ such that C is evolvable over D by A in $\texttt{SelNB}[L_1, t(n, 1/\epsilon), p(n, 1/\epsilon), s(n, 1/\epsilon)]$ if and only if there exist a polynomial $p'(\cdot, \cdot)$ and a p'-neighborhood N on a representation class R' such that C is $t(n, 1/\epsilon)$-evolvable by $(p'(n, 1/\epsilon), R', N, s(n, 1/\epsilon))$ over D.*

Proof: For the first direction we need to emulate the selection based on the empirical neighborhood using a p'-neighborhood. To do this we use M_1 to create the empirical neighborhood by sampling M and then use M_2 to sample from the empirical neighborhood. To give M_2 the information on the multiplicities of each representation in the empirical neighborhood we augment R with multiplicity information. Specifically, we define $R' = (R \times [p(n, 1/\epsilon)])$, where for each $i \in [p(n, 1/\epsilon)]$, (r, i) represents the same function as r. Now for every $r' = (r, i) \in R$, the output of M_1 is a subset of R' obtained by sampling $M(r, \epsilon)$ $p(n, 1/\epsilon)$ times and outputting the set that contains the obtained representations with their multiplicities and also representation r'. (A minor complication will arise if r occurred with multiplicity i and cannot be added twice but this can be resolved by adding a special representation to the neighborhood to indicate this situation to M_2). Given such $\texttt{Neigh}(r', \epsilon)$, M_2 outputs r' with probability $\Delta = \epsilon / g(n, 1/\epsilon)$ and outputs a random representations in $\texttt{Neigh}(r', \epsilon) \setminus \{r'\}$ according to its multiplicity with probability $1 - \Delta$. Here $g(n, 1/\epsilon)$ is the number of generations required for the convergence of A. This defines a $p'(n, 1/\epsilon)$-neighborhood N on R' for $p'(n, 1/\epsilon) = \max\{2(p(n, 1/\epsilon) + 1), g(n, 1/\epsilon)/\epsilon)\}$.

It is easy to verify that for every $r' = (r, i)$ the statistical distance between the output of the mutator

$$\texttt{Mu}(f, p'(n, 1/\epsilon), R', N, D, s(n, 1/\epsilon), r', t(n, 1/\epsilon))$$

and the selection rule $\texttt{SelNB}[L_1, t(n, 1/\epsilon), p(n, 1/\epsilon), s(n, 1/\epsilon)](f, D, A, r)$ is at most Δ. Hence, after $g(n, 1/\epsilon)$ steps the statistical distance between the output of $\texttt{Mu}$ applied to N and $\texttt{SelNB}$ applied to A will be at most ϵ. In particular, with probability at least $1 - 2\epsilon$, the output at step $g(n, 1/\epsilon)$ will have performance at least $1 - \epsilon$.

For the other direction we first note that if M_1 is deterministic then it always produces the same neighborhood and hence M can simply simulate M_2 on the output of M_1 and produce exactly the same distribution over the neighborhood. Further, by Corollary 3.12, the statistical distance between the output of $\texttt{SelNB}[L_1, t(n, 1/\epsilon), p, s(n, 1/\epsilon)](f, D, (R', M), r')$ and the output of $\texttt{Mu}(f, p'(n, 1/\epsilon), R', N, D, s(n, 1/\epsilon), r', t(n, 1/\epsilon))$ is at most $e^{-p/p'}$. By taking $p(n, 1/\epsilon) = p'(n, 1/\epsilon) \cdot \ln(g'(n, 1/\epsilon)/\epsilon)$, we will obtain statistical distance of at most $\epsilon / g'(n, 1/\epsilon)$ (where $g'(n, 1/\epsilon)$ is the number of generations required for the convergence of N).

Now if M_1 is randomized we can emulate its function by adding an extra neutral step before each mutation step. The purpose of each neutral step is to choose a neighborhood of size at most $p'(n, 1/\epsilon)$ in the same way as M_1 does. Given the choice of M_1, M can produce the same output as M_2. Formally let $\ell(n, 1/\epsilon)$ be the polynomial bound on the number of coin flips used by M_1. We set $R = R' \cup (R' \times \{0, 1\}^\ell)$, where for each $v \in \{0, 1\}^\ell$, (r', v) represents the same function as r'. For $r' \in R'$, M chooses $v \in \{0, 1\}^\ell$ randomly and uniformly and outputs representation (r', v). For (r', v), M runs M_1 with its coin flips set to v, then runs M_2 on the output of M_1 and outputs the representation output by M_2. Now when $\texttt{SelNB}$ is applied to M at representation $r' \in R'$ it will output (r', v)

for a randomly and uniformly chosen v since all (r', v)'s represent the same function. When SelNB is applied to M at $(r', v) \in R$, the same analysis as in the deterministic case applies. Altogether, the outcome of two steps of $A = (R, M)$ in $\mathtt{SelNB}[L_1, t'(n, 1/\epsilon), p(n, 1/\epsilon), s'(n, 1/\epsilon)](f, D, A, R')$ is within statistical distance $\epsilon/g'(n, 1/\epsilon)$ of the outcome of one step of the mutator in the original p'-neighborhood N. □

We remark that in Theorem 3.13 the equivalence is given for fixed tolerance t. Clearly, the equivalence also holds for variable tolerance. We also note that the equivalence of our new definition to the original definition is also implied by Theorem 5.1. However Theorem 3.13 preserves (up to small factors) all the parameters of the equivalence and properties such as monotonicity and can also be viewed as supporting robustness of the model of evolvability to reasonable variations.

4 Statistical Query Learning with General Loss Functions

It was observed by Valiant that any evolutionary algorithm can, in fact, be simulated using statistical queries and hence any evolvable concept class is also SQ learnable [Val09]. Further, it was shown in our earlier work that evolvability is equivalent to learnability by a constrained form of statistical queries referred to as *correlational*. A correlational statistical query is a statistical query for a correlation of a function over X with the target [BF02]. Namely the query function $\psi(x, f(x)) \equiv \phi(x)f(x)$ for a function $\phi \in \mathcal{F}_1^\infty$. We say that an algorithm is a correlational statistical query (CSQ) algorithm if it uses only correlational statistical queries.

Therefore, in order to understand the relative power of a particular model of evolvability it is sufficient to relate it to learnability by CSQs. Note that a CSQ ϕ measures the performance $\mathtt{Perf}_f(\phi, D) = L_1\mathtt{Perf}_f(\phi, D)$. Our goal is to show that the equivalence between CSQ learning and evolvability with the linear loss can be extended to other loss functions. For this purpose we define an L-SQ to be a statistical query for which the query function $\psi(x, f(x)) \equiv L\mathtt{Perf}_f(\phi(x), D)$ for some function $\phi \in \mathcal{F}_1^\infty$. Formally, for a function f and distribution D, let L-STAT(f, D) be the oracle that given a function $\phi : X \to [-1, 1]$ as query responds with any value v satisfying $|L\mathtt{Perf}_f(\phi, D) - v| \leq \tau$, where $\tau \in [0, 1]$ is the tolerance of the query. Similarly we say that a concept class C is L-SQ learnable if it is learnable by a statistical query algorithm that uses only L-SQ queries and the output hypothesis h satisfies $L\mathtt{Perf}_f(h, D) \geq 1 - \epsilon$.

Definition 4.1 *For a concept class C, distribution class $\mathcal{D}$ over X and admissible loss function L, an algorithm A is said to L-SQ learn C from queries of tolerance τ in time t if for every $\epsilon > 0$, $\delta > 0$, $f \in C$, $D \in \mathcal{D}$, A given ϵ, δ, and access to L-STAT(f, D) outputs, in time t and with probability at least $1 - \delta$, a hypothesis h that is evaluatable in time t and satisfies $L\mathtt{Perf}_f(h, D) \geq 1 - \epsilon$. Each query ϕ made by A can be evaluated in time t and has tolerance τ. The algorithm is said to (efficiently) learn C if t is polynomial in n, $1/\epsilon$ and $1/\delta$, and τ is lower bounded by the inverse of a polynomial in n and $1/\epsilon$.*

A simple consequence of this definition is the following claim.

Theorem 4.2 *If a concept class C is evolvable over a class of distributions $\mathcal{D}$ in a selection rule* Sel *that uses loss function L then C is efficiently L-SQ learnable over $\mathcal{D}$.*

The proof follows easily from the fact that both the mutation algorithm and the selection rule can be efficiently simulated given the ability to estimate $L\mathtt{Perf}_f$. In the next section we prove the converse of this claim.

We use L-SQ learnability primarily for convenience as, for every admissible loss function L, if L is not, in a sense, similar to L_1 then L-SQ learnability is equivalent to SQ learnability. Specifically, we say that a loss function L is *quasi-linear* if for every $y \in [1, 1]$, $L(1, y) + L(-1, y) = L(-1, 1)$.

Theorem 4.3 *Let L be an admissible loss function that is not quasi-linear. A concept class C is efficiently L-SQ learnable over a class of distributions $\mathcal{D}$ if and only if it is efficiently SQ learnable over $\mathcal{D}$.*

Proof: We first prove that if C is L-SQ learnable then it is SQ learnable. Let A be an efficient L-SQ algorithm for C over $\mathcal{D}$. Clearly every L-SQ is in particular a SQ. Therefore A is a SQ algorithm. The only problem is that L-SQ learnability requires that the final hypothesis h satisfy $L\mathtt{Perf}_f(h, D) \geq 1 - \epsilon$ whereas we need a hypothesis h' that satisfies $\mathbf{Pr}_D[h'(x) = f(x)] \geq 1 - \epsilon$. To obtain such a hypothesis we define the function $\mathsf{sign}_L : [-1, 1] \to \{-1, 1\}$

$$\mathsf{sign}_L(y) = \begin{cases} 1 & L(1, y) \leq L(-1, y) \\ -1 & \text{otherwise} \end{cases}$$

We set $h'(x) \equiv \mathsf{sign}_L(h(x))$. We claim that there exists a constant c such that $\mathbf{Pr}_D[h'(x) = f(x)] \geq 1 - c \cdot \epsilon$. This implies that by running A with $\epsilon' = \epsilon/c$ and applying sign_L to the output we will obtain a hypothesis h' that satisfies $\mathbf{Pr}_D[h'(x) = f(x)] \geq 1 - \epsilon$.

By the definition, L is non-degenerate and hence for every y, $y \in [-1, 1]$, $L(1, y) + L(-1, y) > 0$. Together with monotonicity of L and compactness of $[-1, 1]$ this implies that there exists $\alpha_0 > 0$ such that for every $y \in [-1, 1]$, $L(1, y) + L(-1, y) \geq \alpha_0$. In particular, this implies that if for $b \in \{-1, 1\}$, $\mathsf{sign}_L(y) \neq b$ then $L(b, y) \geq L(-b, y)$ and thus $L(b, y) \geq \alpha_0/2$. Therefore $\mathsf{sign}_L(h(x)) \neq f(x)$ holds only if $L(f(x), h(x)) \geq \alpha_0/2$. This implies that

$$\mathbf{Pr}_D[\mathsf{sign}_L(h(x)) \neq f(x)] \leq \frac{2}{\alpha_0}\mathbf{E}_D[L(f, h)] = \frac{2}{\alpha_0}(1 - L\mathtt{Perf}_f(h, D)) \cdot L(-1, 1)/2 \leq \frac{L(-1, 1)}{\alpha_0} \cdot \epsilon \ ,$$

giving us the claim for $c = \frac{L(-1,1)}{\alpha_0}$.

To prove the other direction let A be a SQ algorithm for C over $\mathcal{D}$. Without loss of generality we can assume that all the queries of A are Boolean (*cf.* [AD98, BF02]). For a statistical query $\psi(x, b)$ we use the decomposition of ψ into two Boolean correlational statistical queries and two queries independent of the target function [BF02]. Namely:

$$\psi(x, b) = \psi(x, -1)\frac{1 - b}{2} + \psi(x, 1)\frac{1 + b}{2} = \frac{1}{2}(\psi(x, 1) - \psi(x, -1))b + \frac{1}{2}(\psi(x, 1) + \psi(x, -1)) \ .$$

Therefore in order to convert a SQ algorithm to an L-SQ algorithm it is sufficient to be able to find answers to a Boolean correlational query and a Boolean query that is independent of the target function. As we have noted in Remark 3.5, for Boolean functions $L\mathtt{Perf}$ is equivalent to $\mathtt{Perf}$ which is exactly a correlational statistical query.

Now let $g(x)$ be a Boolean function. In order to estimate $\mathbf{E}_D[g(x)]$ we use the fact that L is not quasi-linear, that is there exist $y_0 \in [-1, 1]$ such that $L(1, y_0) + L(-1, y_0) \neq L(-1, 1)$. Define $g^+(x) \equiv (1 + g(x))/2$, $g^-(x) \equiv (1 - g(x))/2$, $f^+(x) \equiv (1 + f(x))/2$ and $f^-(x) \equiv (1 - f(x))/2$. Thus,

$$\mathbf{E}_D[f^+(x)g^+(x)] - \mathbf{E}_D[f^-(x)g^+(x)] = \mathbf{E}_D[f(x)g^+(x)] = (\mathbf{E}_D[f(x)] + \mathbf{E}_D[f(x)g(x)])/2 \ .$$

We can therefore obtain the value of

$$\mathbf{E}_D[f^+(x)g^+(x)] - \mathbf{E}_D[f^-(x)g^+(x)] = \mathbf{Pr}_D[f = 1 \ \wedge \ g = 1] - \mathbf{Pr}_D[f = -1 \ \wedge \ g = 1] \qquad (1)$$

using two CSQs. Now we note that $L\texttt{Perf}_f(1, D) + L\texttt{Perf}_f(-1, D) = 0$ and therefore

$$L\texttt{Perf}_f[y_0 g^+(x) - g^-, D] + L\texttt{Perf}_f[y_0 g^+(x) + g^-, D] =$$
$$\mathbf{Pr}_D[f = 1 \ \wedge \ g = 1]\left(1 - \frac{2L(1, y_0)}{L(-1, 1)}\right) + \mathbf{Pr}_D[f = -1 \ \wedge \ g = 1]\left(1 - \frac{2L(-1, y_0)}{L(-1, 1)}\right) \qquad (2)$$

This gives another way to obtain a linear combination of $\mathbf{Pr}_D[f = 1 \ \wedge \ g = 1]$ and $\mathbf{Pr}_D[f = -1 \ \wedge \ g = 1]$. By combining equations (1) and (2), we can find $\mathbf{Pr}_D[f = 1 \ \wedge \ g = 1]$ and $\mathbf{Pr}_D[f = -1 \ \wedge \ g = 1]$ whenever the equations are linearly independent. The equations are linearly independent if

$$1 - \frac{2L(1, y_0)}{L(-1, 1)} \neq -\left(1 - \frac{2L(-1, y_0)}{L(-1, 1)}\right)$$

which is equivalent to $L(1, y_0) + L(-1, y_0) \neq L(-1, 1)$. Hence we can obtain

$$\mathbf{E}_D[g(x)] = 2 \cdot \mathbf{Pr}_D[g = 1] - 1 = 2(\mathbf{Pr}_D[f = 1 \ \wedge \ g = 1] + \mathbf{Pr}_D[f = -1 \ \wedge \ g = 1]) - 1 \ .$$

In order to complete the proof we observe that the final hypothesis h output by A satisfies

$$L\texttt{Perf}_f(h, D) \geq 1 - 2\mathbf{Pr}_D[h(x) \neq f(x)] \geq 1 - 2\epsilon \ .$$

□

On the other hand, all quasi-linear loss functions give statistical queries that are equivalent to correlational statistical queries (or L_1-SQ).

Theorem 4.4 *Let L be an admissible quasi-linear loss function. A concept class C is efficiently L-SQ learnable over a class of distributions $\mathcal{D}$ if and only if it is efficiently CSQ learnable over $\mathcal{D}$.*

Proof: We first prove that if C is L-SQ learnable then it is CSQ learnable. Given an L-SQ $\phi(x)$ we let $\phi'(x) = 1 - 2L(1, \phi(x))/L(-1, 1)$. The condition that L is quasi-linear implies that

$$1 - \frac{2L(-1, \phi(x))}{L(-1, 1)} = -\left(1 - \frac{2L(1, \phi(x))}{L(-1, 1)}\right) = -\phi'(x) \ .$$

This implies that $f(x)\phi'(x) = 1 - 2L(f(x), \phi(x))/L(-1, 1)$ and therefore $L\texttt{Perf}_f(\phi(x), D) = \mathbf{E}_D[f(x)\phi'(x)]$. Therefore we can simulate every L-SQ using a single CSQ. In order to obtain a final hypothesis h' that satisfies $\mathbf{Pr}_D[h'(x) = f(x)] \geq 1 - \epsilon$ we use the same argument as in the proof of Theorem 4.3.

For the other direction use Remark 3.5 that implies that for any Boolean function ϕ, $L\texttt{Perf}_f(\phi, D) = \mathbf{E}_D[\phi \cdot f]$. Therefore one can use L-SQs to simulate CSQs. The final hypothesis h output by the CSQ algorithm satisfies

$$L\texttt{Perf}_f(h, D) \geq 1 - \mathbf{Pr}_D[h(x) \neq f(x)] \geq 1 - 2\epsilon \ .$$

□

5 Fixed-Tolerance Evolvability for Any Loss Function

In this section we examine the roles of the tolerance of the selection process and the loss function used in it. We show that for an admissible loss function L, any L-SQ learnable concept class is evolvable in $\texttt{SelNB}[L, t, s, s]$ for some polynomial s (used for both the candidate pool size and the

sample size), and fixed t lower bounded by the inverse of a polynomial in n and $1/\epsilon$. This strengthens our earlier result [Fel08] for the linear loss and extends it to other loss functions. In addition the algorithm that we produce also evolves with the optimizing selection rule $\texttt{SelOpt}[L, t, s, s]$. This demonstrates that evolvability with a specific optimizing selection rule is equivalent to the basic model. We note that one direction of this equivalence was showed by Valiant using a more direct method [Val09].

Theorem 5.1 *Let L be an admissible loss function and C be a concept class L-SQ learnable over a class of distributions $\mathcal{D}$ by a randomized polynomial-time algorithm B. There exists a polynomial $s(\cdot,\cdot)$, inverse-polynomial $t(\cdot,\cdot)$ and an evolutionary algorithm $A = (R, M)$ such that C is evolvable by A over $\mathcal{D}$ in $\texttt{SelNB}[L, t(n, 1/\epsilon), s(n, 1/\epsilon), s(n, 1/\epsilon)]$ and in $\texttt{SelOpt}[L, t(n, 1/\epsilon), s(n, 1/\epsilon), s(n, 1/\epsilon)]$.*

As in [Fel08] the proof is based on a reduction of general L-SQs to comparisons. Namely, for a function f and a distribution D we define an oracle $L\text{-SQ}_>(f, D)$ to be the oracle that accepts queries of the form (r, θ, τ) where r is a function from X to $[-1, 1]$, θ is the *threshold* and $\tau > 0$ is the tolerance. To such a query (referred to as an $L\text{-SQ}_>$ query) the oracle returns 1 when $L\texttt{Perf}_f(r, D) \geq \theta + \tau$, 0 when $L\texttt{Perf}_f(r, D) \leq \theta - \tau$, and either 0 or 1, otherwise. The following lemma is analogous to a lemma from [Fel08] (a subtle but significant difference is that $\texttt{Perf}_f(-r, D) = -\texttt{Perf}_f(r, D)$ does not hold for general loss functions and hence some of the resulting comparisons are with a negative threshold).

Lemma 5.2 *For every function f and distribution D, an L-SQ (r, τ) can be replaced by $\lceil \log(1/\tau) \rceil + 1$ queries to L-SQ$_>(f, D)$. Each of the produced queries is of the form (r, θ, τ'), where $\tau' = \tau/2$, and $|\theta| \geq \tau/2$.*

Proof: Let $v = L\texttt{Perf}_f(r, D)$. Queries to $L\text{-SQ}_>(f, D)$ allow to perform comparisons of v with any value in $[-1, 1]$ up to accuracy $\tau/2$. Let T denote the set

$$\{\tau/2 + i \cdot \tau \mid -1/\tau - 1/2 \leq i \leq 1/\tau - 1/2\} \cup \{-1, 1\} .$$

We perform a binary search for a value $u \in T$ such that the oracle returns 1 on the comparison with u (meaning ">") and 0 on the comparison with u', where u' is the smallest value in T larger than u. Such value will always exist since we can assume that the comparison with -1 returns 1 and the comparison with 1 returns 0. The procedure returns the value $v' = (u + u')/2$.

The answers of the oracle imply that $v \in [u - \tau/2, u' + \tau/2]$. By our choice of the set T, $u' - u \leq \tau$ and therefore $|v' - v| \leq \tau$. The number of comparison queries performed is at most $\lceil \log(1/\tau) \rceil + 1$ and by the choice of T, every comparison is with a threshold θ such that $|\theta| \geq \tau/2$. □

The reduction from a $L\text{-SQ}_>$ algorithm to a mutation algorithm and its analysis are similar to those in [Fel08]. The high-level idea of this construction is to get the answer to an $L\text{-SQ}_>$ (ϕ, θ, τ) by creating a mutation that outputs $\phi(x)$ with small probability. For an appropriately chosen probability, this mutation will be chosen as the next generation if and only if $L\texttt{Perf}_f(\phi, D) \geq \theta$. This allows the evolutionary algorithm to "record" the answer to the $L\text{-SQ}_>$. All the answers that were obtained so far are "remembered" in the current representation. Finally, given the answers to all the queries of the $L\text{-SQ}_>$ algorithm, the mutation algorithm outputs the representation equal to the final hypothesis of the $L\text{-SQ}_>$ algorithm for the given answers of its queries. To achieve fixed tolerance, the probability that the mutation outputs $\phi(x)$ is scaled appropriately. Also, compared with our earlier construction, a different type of mutations needs to be used for $L\text{-SQ}_>$ queries with a negative threshold. The complete proof appears in Appendix A.

6 Evolvability with Weak Selection Rules

We now show that the analogues of the results of the previous section can also be obtained in any selection rule that weakly "favors" mutations with sufficiently higher performance. We only prove the result for performance based on linear loss. The extension to general loss functions can be obtained using similar methods.

To prove the result we design a new general transformation from a CSQ algorithm to a mutation algorithm that evolves in any (t, γ)-distinguishing selection rule for sufficiently small t and sufficiently large γ. Note that $\texttt{SelNB}[L_1, t, s, s]$ and $\texttt{SelOpt}[L_1, t, s, s]$ are both t-distinguishing selection rules (with high probability and for sufficiently large s).

Theorem 6.1 *Let C be a concept class CSQ learnable over a class of distributions $\mathcal{D}$ by a polynomial-time algorithm B. There exists an inverse polynomial $t(n, 1/\epsilon)$ such that for every inverse polynomial $\gamma(n, 1/\epsilon)$, there exists a mutation algorithm $A_\gamma = (R_\gamma, M_\gamma)$ that evolves C over $\mathcal{D}$ in Π with initialization for every (t, γ)-distinguishing selection rule Π.*

The key new step of this proof is a conversion of any CSQ algorithm to an algorithm that uses a similarly restricted version of statistical queries. For a function f and distribution D over X, let (τ, γ)-CSQ(f, D) oracle be the oracle that given a function $\phi : X \to [-1, 1]$ returns a value $b \in \{0, 1\}$ such that:

- if $\mathbf{E}_D[f \cdot \phi] \geq \tau$ then $b = 1$ with probability at least γ' where γ' satisfies $(1 - \gamma')/\gamma' \leq 1 - \gamma$ (independently of any previous queries);
- if $\mathbf{E}_D[f \cdot \phi] \leq -\tau$ then $b = 0$ with probability at least γ' where γ' satisfies $(1 - \gamma')/\gamma' \leq 1 - \gamma$ (independently of any previous queries).

Lemma 6.2 *Let C be a concept class CSQ learnable over a class of distributions $\mathcal{D}$ by a polynomial-time algorithm B. There exists an inverse polynomial $\tau'(n, 1/\epsilon)$ such that for every inverse polynomial $\gamma(n, 1/\epsilon)$, there exists an algorithm B_γ that given access to (τ', γ)-CSQ(f, D) oracle learns C over $\mathcal{D}$.*

Proof: We first note that by repeating a query ϕ to (τ', γ)-CSQ(f, D) oracle and taking the majority vote over the outcomes, we can boost the confidence in the answer to the query. That is, for every $\delta > 0$, by repeating the query $O(\gamma^{-2} \log(1/\delta))$ times we can simulate $(\tau', 1-\delta)$-CSQ(f, D) oracle. By taking small enough δ we can assume that we are given $(\tau', 1)$-CSQ(f, D) oracle, that is, the oracle that always returns 1 if (but not only if) $\mathbf{E}_D[f \cdot \phi] \geq \tau'$ and returns 0 if (but not only if) $\mathbf{E}_D[f \cdot \phi] \leq -\tau'$.

Our next step is to reduce B to an algorithm B' that uses queries to CSQ$_>(f, D)$ using Lemma 5.2. Let τ_0 denote the tolerance of B. The general idea of the rest of the transformation is as follows. The $(\tau', 1)$-CSQ(f, D) oracle can answer a query $(\phi, 0, \tau')$ to CSQ$_>(f, D)$ oracle, that is it can compare $\mathbf{E}_D[f \cdot \phi]$ to 0 with tolerance τ'. Assume, for simplicity, that we are given a function $g(x)$ such that $\mathbf{E}_D[f \cdot g] = \theta$. Then, by giving a query $\phi(x) - g(x)$ to $(\tau', 1)$-CSQ(f, D) oracle, we can find (up to tolerance τ') whether $\mathbf{E}_D[f \cdot (\phi - g)] \geq 0$ which is equivalent to $\mathbf{E}_D[f \cdot \phi] \geq \mathbf{E}_D[f \cdot g] = \theta$. Therefore, given g as above, we can directly simulate a query to B' using a query to $(\tau', 1)$-CSQ(f, D) oracle. The main problem is, therefore, to find a function g that can be used in place of the threshold.

We note that given a function g with performance α we can still obtain the result of a comparison by scaling the functions appropriately. To find an appropriate g and α we use the characterization of (weak) CSQ learning [Fel08]. Specifically, in [Fel08] it is proved that CSQ learnability of C over

$\mathcal{D}$ implies that there exist a polynomial $q(n)$ and an efficiently computable polynomial-sized set of Boolean functions S such that for every $f \in C$ and $D \in \mathcal{D}$, there exists $g' \in S$ such that $|\mathbf{E}_D[f \cdot g']| \geq 1/q(n)$. We can assume that $\mathbf{E}_D[f \cdot g'] \geq 1/q(n)$ by replacing g' with its negation if necessary. Let $g = \mathrm{argmax}_{g' \in S}\{\mathbf{E}_D[f \cdot g']\}$ and let α be a value such that $1/q(n) \leq \alpha \leq \mathbf{E}_D[f \cdot g] + \tau/(3q(n))$. For every query (ϕ, θ, τ_0) to $\mathrm{CSQ}_>(f, D)$ we ask query $\phi' = (\alpha \cdot \phi - \theta \cdot g)/2$ to $(\tau', 1)$-CSQ(f, D) for $\tau' = \tau_0/(3q(n))$. First note that the range of ϕ' is $[-1, 1]$ and hence the query is legal. If the oracle returns 1, then $\mathbf{E}_D[\phi' \cdot f] \geq -\tau'$ or

$$\mathbf{E}_D[(\alpha \cdot \phi - \theta \cdot g) \cdot f]/2 = (\alpha \cdot \mathbf{E}_D[\phi \cdot f] - \theta \cdot \mathbf{E}_D[g \cdot f])/2 \geq -\tau' = -\tau_0/(3q(n)).$$

This implies that

$$\begin{aligned}\mathbf{E}_D[\phi \cdot f] &\geq \frac{\theta \cdot \mathbf{E}_D[g \cdot f]}{\alpha} - \frac{2\tau_0}{3\alpha \cdot q(n)} \geq \frac{\theta(\alpha - \tau_0/(3q(n)))}{\alpha} - \frac{2\tau_0}{3\alpha \cdot q(n)} \\ &= \theta - \frac{\theta \cdot \tau_0 + 2\tau_0}{3\alpha \cdot q(n)} \geq \theta - \frac{\tau_0}{\alpha \cdot q(n)} \geq \theta - \tau_0\end{aligned}$$

and therefore 1 is also a valid response to query (ϕ, θ, τ_0) for $\mathrm{CSQ}_>(f, D)$ oracle. If $(\tau', 1)$-CSQ(f, D) oracle returns 0, then $\mathbf{E}_D[\phi' \cdot f] \leq \tau'$ and this, by a similar argument implies that 0 is a valid response to query (ϕ, θ, τ) for $\mathrm{CSQ}_>(f, D)$.

We now deal with the problem of finding $g = \mathrm{argmax}_{g' \in S}\{\mathbf{E}_D[f \cdot g']\}$ and α such that $1/q(n) \leq \alpha \leq \mathbf{E}_D[f \cdot g] + \tau_0/(3q(n))$. Instead of actually finding the pair, we try all pairs (g', α') such that $g' \in S$ and $\alpha' \in \{i \cdot \tau_0/(3q(n))\}_{i \in [3q(n)/\tau_0]}$. That is for every pair we simulate B' while using g' and α' to answer queries as described above. Let $h_{g',\alpha'}$ denote the hypothesis output by B' in this simulation and let H be the set of all hypotheses we obtain. Clearly, g and a suitable α will be among the pairs we try and hence, there exists $h \in H$ such that $\mathbf{E}_D[f \cdot h] \geq 1 - \epsilon$. Further our $(\tau', 1)$-CSQ(f, D) oracle allows us to compare $\mathbf{E}_D[f \cdot h_1]$ with $\mathbf{E}_D[f \cdot h_2]$ by asking query $(h_1 - h_2)/2$. A result of the query is only guaranteed to be correct when $|\mathbf{E}_D[f \cdot h_1] - \mathbf{E}_D[f \cdot h_2]| > 2\tau'$. Our goal is, then, to use these comparisons to find a hypothesis h_{max} with the maximum value of the correlation with the target concept. Clearly h_{max} satisfies $\mathbf{E}_D[f \cdot h_{max}] \geq 1 - \epsilon$. Our comparisons are imprecise and therefore we might not be able to find h_{max}. However, we claim that one can find a hypothesis h^* such that $\mathbf{E}_D[f \cdot h^*] \geq \mathbf{E}_D[f \cdot h_{max}] - 4\tau' \geq 1 - \epsilon - 4\tau'$. To achieve this we compare each pair of hypotheses $h_1, h_2 \in H$. For a hypothesis $h \in H$ let k_h be the number of comparisons in which the correlation of h was larger. Let h^* be a hypothesis such that $k_{h^*} = \max\{k_h\}_{h \in H}$. If, according to the above comparison, the correlation of h^* is larger than the correlation of h_{max} then $\mathbf{E}_D[f \cdot h^*] \geq \mathbf{E}_D[f \cdot h_{max}] - 2\tau'$ and therefore h^* has the desired property. Otherwise (if the correlation of h^* is smaller then the correlation of h_{max}), the condition $k_{h^*} \geq k_{h_{max}}$ implies that there exist a function $h' \in H$ such that, the correlation of h^* is larger than the correlation of h' and the correlation of h' is larger than the correlation of h_{max}. In both cases the comparison is correct up to the tolerance of $2\tau'$ and therefore

$$\mathbf{E}_D[f \cdot h^*] \geq \mathbf{E}_D[f \cdot h'] - 2\tau' \geq \mathbf{E}_D[f \cdot h_{max}] - 4\tau' \ .$$

We note that this problem of finding a close-to-maximum element from imprecise comparisons was recently studied by Ajtai *et al.* who give substantially more efficient methods of finding a close-to-maximum element [AFHN09].

As usual we can use B to learn to accuracy $\epsilon/2$ and set $\tau' = \min\{\epsilon/8, \tau_0/(3q(n))\}$ to ensure that $\mathbf{E}_D[f \cdot h^*] \geq 1 - \epsilon$. Using the bounds on τ_0 and $q(n)$, it is easy to verify that the running time of the algorithm we constructed is polynomial and that τ' can be lower bounded by an inverse of a polynomial in n and $1/\epsilon$. $\square$

Now we can use a (t, γ)-distinguishing selection rule to provide responses to queries for (τ', γ)-CSQ(f, D) oracle in essentially the same way as SelNB was used to answer CSQ$_>$ queries in the proof of Theorem A.1. The details are included in Appendix A.

It would certainly be desirable to obtain the same result without initialization. We note that by using a selection rule that satisfies additional conditions our construction can be strengthened to not require initialization as in the proof of Theorem 5.1. However our current method that adds the ability to "re-initialize" to an evolutionary algorithm requires gradual reduction of performance to 0. This, while legal in the basic model, seems to contradict the main motivation behind evolvability from any state: the ability to adjust to new environmental conditions without significant decrease in performance. Therefore we regard such methods of achieving "re-initialization" as less useful for understanding the power of evolutionary algorithms and hence do not adapt them to this setting.

7 Distribution-Independent Evolvability of Singletons

In this section we show that the concept class of singletons is evolvable distribution-independently. The concept class of singletons over $\{0,1\}^n$ is defined as $C_1 = \{\mathtt{Ind}[z] \mid z \in \{0,1\}^n\}$ where $\mathtt{Ind}[z]$ is the indicator function of the set $\{z\}$.

Theorem 7.1 *C_1 is evolvable distribution-independently.*

Proof: To prove this we show that there exists a mutation algorithm M that for every current hypothesis h, can produce a random hypothesis h' such that for every distribution D and target function $\mathtt{Ind}[z]$, with probability at least $1/p(n, 1/\epsilon)$, $M(h, \epsilon)$ outputs a hypothesis h' such that

$$\mathbf{E}_D[\mathtt{Ind}[z] \cdot h'] \geq 1 - \epsilon/2$$

for some polynomial $p(n, 1/\epsilon)$. For such M, for sufficiently large polynomial $s(\cdot, \cdot)$ the empirical neighborhood of h will include a hypothesis with sufficiently high performance. This means that the desired performance will be achieved in a single step of

$$\mathtt{SelNB}[L_1, \epsilon/4, s(n, 1/\epsilon), s(n, 1/\epsilon)]$$

applied to M.

We now describe the design and analysis of M. M outputs function $\mathtt{Ind}[S]$, where $S \subseteq \{0,1\}^n$ is chosen randomly and has the following properties

- uniformity: for every $x \in \{0,1\}^n$, $\mathbf{Pr}_M[x \in S] = 2^{-k}$, for $k = \lceil \log 1/\epsilon \rceil + 3$.
- pairwise independence: for every $x, y \in \{0,1\}^n$ such that $x \neq y$, $\mathbf{Pr}_M[x \in S \mid y \in S] = 2^{-k}$.

We first note that it is easy to choose S in such a way. One possible way is as follows: choose randomly and uniformly $k+1$ vectors in $\{0,1\}^n$, and let B denote the first k vectors and v denote the last one. Then let

$$S_{B,v} = \left\{ a \in \{0,1\}^n \mid \forall b \in B, \sum_{i \in [n]} ((a_i + v_i) \cdot b_i) = 0 \pmod 2 \right\}$$

It is well-known and can be easily verified that this procedure will produce a pairwise-independent subset of $\{0,1\}^n$. We also observe that $\mathtt{Ind}[S]$ as above can be computed efficiently for every choice of random vectors.

With probability 2^{-k}, $z \in S$. In addition, pairwise independence implies that for every $y \neq z$, $\mathbf{Pr}_M[y \in S \mid z \in S] = 2^{-k}$. Therefore

$$\mathbf{E}_M[\mathbf{Pr}_D[x \in S \setminus \{z\}] \mid z \in S] = \sum_{y \neq z} (\mathbf{Pr}_M[y \in S \mid z \in S] \cdot \mathbf{Pr}_D[x = y]) = 2^{-k} \sum_{y \neq z} \mathbf{Pr}_D[x = y] \leq 2^{-k}.$$

Therefore, by Markov's inequality, with probability at least $\frac{1}{2}2^{-k}$ we have that $z \in S$ and $\mathbf{Pr}_D[x \in S \setminus \{y\}] \leq 2 \cdot 2^{-k}$. This implies that with probability at least $2^{-k-1} \geq \epsilon/32$ over the choice of S,

$$\mathbf{E}_D[\mathtt{Ind}[S] \cdot \mathtt{Ind}[z]] \geq 1 - 2 \cdot \mathbf{Pr}_D[x \in S \setminus z] \geq 1 - 2^{-k+2} \geq 1 - \epsilon/2.$$

We note that the representation class R used by this algorithm includes $\mathtt{Ind}[S]$ for each set B and vector $b \in \{0,1\}^n$. However the mutation algorithm can start from any hypothesis adding a new degree of robustness to the algorithm. □

Additional properties of this mutation algorithm can be observed. First, the algorithm can be easily extended to the concept class of indicator functions of sets of polynomial in n size. In addition, the mutation algorithm evolves C_1 in any $(\epsilon/8)$-distinguishing selection rule. This implies that C_1 is evolvable monotonically giving the first example of a monotone evolvability in the basic model. Finally, this algorithm is in fact agnostic, that is if the target function f is not $\mathtt{Ind}[z]$ for some z, this algorithm will converge to a function with performance within ϵ of the best possible by a singleton.

8 Conclusions

In this work we have addressed the roles of the main components of Valiant's model of evolvability: performance (or fitness) metric, selection rule and its tolerance, constraints on mutation algorithms and initialization. Our results clarify many aspects of the interactions among these components, thereby, we believe, greatly enhancing the understanding of Valiant's framework for modeling evolution. Most of the results demonstrate that the limits of the efficiently evolvable are not changed when a reasonable variant of the definition is considered. These limits coincide with all of the learnable using the corresponding restriction of statistical queries. Our results also demonstrate that evolvability with hypotheses that use intermediate values in $[-1, 1]$ and non-linear loss is strictly stronger than the basic model. Whether and when such feedback is biologically plausible is an interesting question.

From the algorithmic standpoint our results give new ways to automatically obtain numerous new algorithms that satisfy the onerous requirements of the evolvability framework. While the general transformations produce algorithms that are not necessarily the most natural and efficient, they can be used as a basis for finding simpler and more efficient algorithms for specific concept classes and distributions. Our singleton learning algorithm is an example of this approach.

Some important questions about the power of models of evolvability are not answered by this work. The most interesting among them is whether the fairly unnatural "reinitialization" used in our reduction when the evolutionary algorithm is not initialized is necessary. In other words, whether monotone evolvability is equivalent to the basic model. Currently, the only known examples of monotone evolvability is Michael's algorithm for evolving decision lists over the uniform distribution using quadratic loss and the algorithm for evolving singletons presented here. In a very recent work we present some progress on this question [Fel09] but the question remain unresolved even for the concept class of conjunctions.

Acknowledgements

I am grateful to Les Valiant for suggesting several of the problems addressed here and numerous helpful discussions. I also thank Ron Fagin for helpful comments on this manuscript.

References

[AD98] J. Aslam and S. Decatur. Specification and simulation of statistical query algorithms for efficiency and noise tolerance. *Journal of Computer and System Sciences*, 56:191–208, 1998.

[AFHN09] M. Ajtai, V. Feldman, A. Hassidim, and J. Nelson. Sorting and selection with imprecise comparisons. Proceedings of ICALP(A), 2009.

[BF02] N. Bshouty and V. Feldman. On using extended statistical queries to avoid membership queries. *Journal of Machine Learning Research*, 2:359–395, 2002.

[BFM97] T. Back, D. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computation*. IOP Publishing Ltd., Bristol, UK, 1997.

[BVdW07] H. Buhrman, N. Vereshchagin, and R. de Wolf. On computation and communication with small bias. In *Proceedings of IEEE Conference on Computational Complexity*, pages 24–32, 2007.

[Fel08] V. Feldman. Evolvability from learning algorithms. In *Proceedings of STOC*, pages 619–628, 2008.

[Fel09] V. Feldman. A complete characterization of statistical query learning with applications to evolvability. Manuscript. Available on the author's web page, 2009.

[FV08] V. Feldman and L. G. Valiant. The learning power of evolution. In *Proceedings of COLT*, pages 513–514, 2008.

[GHR92] M. Goldmann, J. Håstad, and A. Razborov. Majority gates vs. general weighted threshold gates. *Computational Complexity*, 2:277–300, 1992.

[Hau92] D. Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Computation*, 100(1):78–150, 1992.

[HMLW91] D. Haussler, M.Kearns, N. Littlestone, and M. Warmuth. Equivalence of models for polynomial learnability. *Information and Computation*, 95(2):129–161, 1991.

[Hoe63] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

[JPY88] D. Johnson, C. Papadimitriou, and M. Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37:79–100, 1988.

[Kea98] M. Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM*, 45(6):983–1006, 1998.

[KS94] M. Kearns and R. Schapire. Efficient distribution-free learning of probabilistic concepts. *Journal of Computer and System Sciences*, 48:464–497, 1994.

[Mic07] L. Michael. Evolving decision lists. Manuscript, 2007.

[She07] A. A. Sherstov. Halfspace matrices. In *Proceedings of IEEE Conference on Computational Complexity*, pages 83–95, 2007.

[Val84] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[Val06] L. G. Valiant. Evolvability. *Electronic Colloquium on Computational Complexity (ECCC)*, 6(120), 2006.

[Val08] L. G. Valiant, 2008. Personal communication.

[Val09] L. G. Valiant. Evolvability. *Journal of the ACM*, 56(1):3.1–3.21, 2009.

[Weg01] I. Wegener. Theoretical aspects of evolutionary algorithms. In *Proceedings of ICALP*, pages 64–78, 2001.

[Wri78] S. Wright. *Evolution and the Genetics of Populations, A Treatise*. University of Chicago Press, Chicago, 1968-78.

[Yam98] K. Yamanishi. A decision-theoretic extension of stochcastic complexity and its applications to learning. *IEEE Transactions on Information Theory*, 44(4):1424–1439, 1998.

A Proofs

A.1 Section 5

We first prove the claimed result (5.1) for evolvability with initialization. For a vector z and $j \in [|z|]$ let z_j denote the j^{th} element of z and let z^j denote the prefix of length j of z. For every z, z^0 equals the empty string λ.

Theorem A.1 *Let L be an admissible loss function and C be a concept class L-SQ learnable over a class of distributions $\mathcal{D}$ by a randomized polynomial-time algorithm B. There exists a polynomial $s(\cdot,\cdot)$, inverse-polynomial $t(\cdot,\cdot)$ and an evolutionary algorithm $A = (R, M)$ such that C is evolvable by A over $\mathcal{D}$ in $\mathtt{SelNB}[L, t(n, 1/\epsilon), s(n, 1/\epsilon), s(n, 1/\epsilon)]$ with initialization.*

Proof: Let $\mathcal{H}$ be the representation class of B's hypotheses. We first assume that B is deterministic and apply Lemma 5.2 to convert B to algorithm B' that uses only queries to L-SQ$_>(f, D)$. Let $q = q(n, 1/\epsilon)$ be a polynomial upper bound on the number of queries asked by B' and $\tau = \tau(n, 1/\epsilon)$ denote the tolerance of the queries asked by B'.

For $i \in [q]$ and $z \in \{0,1\}^{i-1}$, let $(\phi_z(x), \theta_z, \tau)$ denote the i^{th} query that B' asks given that the answers to the previous $i-1$ queries are as specified by z. Here for $j \leq i-1$, bit z_j is the answer to the j^{th} query. For $z \in \{0,1\}^q$ we denote by h_z the hypothesis produced by B' given that its queries were answered according to z (we can assume without loss of generality that exactly q queries are asked in every possible execution of B'). Note that the queries produced by B' implicitly depend on the values of ϵ and n.

The high-level idea of our construction is to get the answer to a L-SQ$_>$ (ϕ, θ, τ) of B' is to use a mutation that outputs $\phi(x)$ with small probability. For an appropriately chosen probability, this mutation will be chosen as the next generation if and only if $L\mathtt{Perf}_f(\phi, D) \geq \theta$. This allows the

evolutionary algorithm to "record" the answer to the L-SQ$_>$. All the answers that were obtained so far are "remembered" in the current representation. Finally, given the answers to all the queries of B', the mutation algorithm outputs the representation equal to the final hypothesis of B' for the given answers to B's queries.

We will now define the mutation algorithm $A = (R, M)$ for C formally. For $i \in [q]$ and $z \in \{0,1\}^i$, we define Φ_0 to be the random coin flip hypothesis, that is for every $x \in X$, Φ_0 equals 1 with probability 1/2 and -1 with probability 1/2. Let $r_z(x)$ be the hypothesis that is computed as follows. For each $j \in [i]$ such that $z_j = 1$, $\phi_{z^{j-1}}(x)$ is output with probability $\frac{\tau}{|\theta_{z^{j-1}}| \cdot q}$. Note that according to Lemma 5.2, for every query ϕ_z, $|\theta_z| \geq \tau$ and hence the total probability that either of $\phi_{z^{j-1}}(x)$ is output is less than 1. With the remaining probability $r_z(x)$ equals $\Phi_0(x)$. Let

$$R = \mathcal{H} \cup \{r_\lambda\} \cup \{r_z\}_{i \in [q], z \in \{0,1\}^i} ,$$

where $r_\lambda \equiv \Phi_0$ (λ denotes the empty string).

We now define M by specifying for each r, $\mathtt{Neigh}_A(r, \epsilon)$ and probabilities of mutations in $\mathtt{Neigh}_A(r, \epsilon)$. Let $\Delta \in (0, 1)$ be a real value to be defined later.

1. $r = r_z$ for $z \in \{0,1\}^i$ where $0 \leq i \leq q-1$: $\mathtt{Neigh}_A(r, \epsilon) = \{r_{z0}, r_{z1}\}$;
   if $\theta_z > 0$ then $\mathbf{Pr}_A(r, r_{z1}) = \Delta$ and $\mathbf{Pr}_A(r, r_{z0}) = 1 - \Delta$, otherwise $\mathbf{Pr}_A(r, r_{z1}) = 1 - \Delta$ and $\mathbf{Pr}_A(r, r_{z0}) = \Delta$.

2. $r = r_z$ for $z \in \{0,1\}^q$: $\mathtt{Neigh}_A(r, \epsilon) = \{r, h_z\}$; $\mathbf{Pr}_A(r, h_z) = 1 - \Delta$ and $\mathbf{Pr}_A(r, r) = \Delta$.

3. $r = h$ for $h \in \mathcal{H}$: $\mathtt{Neigh}_A(r, \epsilon) = \{r\}$; $\mathbf{Pr}_A(r, r) = 1$.

Claim A.2 *There exists a polynomial $s(\cdot, \cdot)$ and inverse-polynomial $t(\cdot, \cdot)$ such that C is evolvable by A over $\mathcal{D}$ in $\mathtt{SelNB}[L, t(n, 1/\epsilon), s(n, 1/\epsilon), s(n, 1/\epsilon)]$ with initialization.*

Proof: We define the bound on the number of generations $g(n, 1/\epsilon) = q(n, 1/\epsilon) + 1$, let $t(n, 1/\epsilon) = \tau(n, 1/\epsilon)/q(n, 1/\epsilon)$ and let $\Delta = \frac{\epsilon}{4g(n,1/\epsilon)}$. Let $s'(n, 1/\epsilon)$ be the size of a sample sufficient to produce an estimate of a random variable $V \in [-1, 1]$ within $\tau' = \frac{\tau^2}{2q}$ with probability at least $1 - \frac{\epsilon}{12 \cdot g(n,1/\epsilon)}$. Hoeffding's bound implies that $s'(n, 1/\epsilon) = c_0 q^2 \cdot \tau^{-4} \cdot \log(1/\epsilon)$ samples for a constant c_0 will suffice for this [Hoe63]. We also choose the candidate pool size to be large enough to ensure that with high probability, every possible mutation of the current representation occurs in the empirical neighborhood of $\mathtt{SelNB}$. As follows from Lemma 3.11, in this condition one can replace the empirical probabilities of mutations (or $\mathbf{Pr}_Z(r, r_1)$) with their actual probabilities (or $\mathbf{Pr}_A(r, r_1)$). Neighborhood of every representation has at most two representation and each occurs with probability at least Δ. Therefore by choosing candidate pool size to be at least $p'(n, 1/\epsilon) = \ln(8 \cdot g(n, 1/\epsilon)/\epsilon)/\Delta$, we will guarantee that for every representation, its empirical neighborhood will equal the actual neighborhood with probability at least $1 - \frac{\epsilon}{4g(n,1/\epsilon)}$. We set $s(n, 1/\epsilon) = \max\{s'(n, 1/\epsilon), p'(n, 1/\epsilon)\}$.

Let $f \in C$ be the ideal function and let $r_0 = r_\lambda, r_1, r_2, \ldots, r_g$ be a sequence of representations produced by

$$r_k \leftarrow \mathtt{SelNB}[L, t, s(n, 1/\epsilon), s(n, 1/\epsilon)](f, D, A, r_{k-1}) .$$

Our goal is to prove that with probability at least $1 - 3\epsilon/4$, $L\mathtt{Perf}_f(r_g, D) \geq 1 - \epsilon$. We first note that for every representation r, the neighborhood of r contains at most two representations. Therefore at most $3 \cdot g(n, 1/\epsilon)$ estimations of performance on a sample of size s will be required. By the choice of s, each of these estimates is within $\tau' = \frac{\tau^2}{2q}$ of the true performance with probability at least $1 - \frac{\epsilon}{12 \cdot g(n,1/\epsilon)}$ and therefore all of them are within τ' with probability at least $1 - \epsilon/4$. For

a representation r, we denote the obtained estimate by $v(r)$. For the given choice of $s(n, 1/\epsilon)$, in $g(n, 1/\epsilon)$ steps the empirical neighborhoods are the same as actual neighborhoods with probability at least $1 - \epsilon/4$.

Next, assuming that all the estimates are within τ' and the empirical neighborhoods are the same as actual neighborhoods, we prove that for every z of length $k \in \{0, 1, \ldots, q-1\}$, if $r_k = r_z$ then with probability at least $1 - \Delta$, $r_{k+1} = r_{zb}$, where b is a valid answer to query $(\phi_z(x), \theta_z, \tau)$ from $L\text{-SQ}_>(f, D)$.

By the definition of A, $\texttt{Neigh}_A(r_z, \epsilon) = \{r_{z0}, r_{z1}\}$. According to the definition of the function computed by r_z and using the linearity of expectation, we obtain

$$L\texttt{Perf}_f(r_z, D) = \frac{1}{q} \sum_{j \in [i],\ z_j = 1} \frac{\tau}{|\theta_{z^{j-1}}|} L\texttt{Perf}_f(\phi_{z^{j-1}}, D) \ .$$

Representations r_z and r_{z0} compute the same function and therefore have the same performance which we denote by ρ. Using the definition of the function computed by r_{z1}, we get that

$$\begin{aligned} L\texttt{Perf}_f(r_{z1}, D) &= \frac{1}{q} \left(\sum_{j \in [i],\ z_j = 1} \frac{\tau}{|\theta_{z^{j-1}}|} L\texttt{Perf}_f(\phi_{z^{j-1}}, D) \right) + \frac{\tau}{q \cdot |\theta_z|} L\texttt{Perf}_f(\phi_z, D) \\ &= \rho + \frac{\tau}{q \cdot |\theta_z|} L\texttt{Perf}_f(\phi_z, D) \ . \end{aligned}$$

since $(z1)_{i+1} = 1$ and $(z1)^i = z$. By the definition, $t = \tau/q \geq 2\tau'$ and therefore $|v(r_{z0}) - v(r_z)| \leq 2\tau' \leq t$ meaning that $r_{z0} \in \texttt{Neut}$.

We now consider two cases

- $\theta_z > 0$: If $r_{z1} \in \texttt{Bene}$ then $r_{k+1} = r_{z1}$ and $v(r_{z1}) - v(r_z) \geq t$. But

  $$v(r_{z1}) - v(r_z) \leq L\texttt{Perf}_f(r_{z1}, D) - L\texttt{Perf}_f(r_z, D) + 2\tau' = \frac{\tau}{q \cdot \theta_z} L\texttt{Perf}_f(\phi_z, D) + 2\tau' \ .$$

  That is,

  $$L\texttt{Perf}_f(\phi_z, D) \geq \frac{\theta_z \cdot q}{\tau}(t - 2\tau') = \theta_z - \theta_z \cdot \tau \geq \theta_z - \tau.$$

  Therefore $b = 1$ is indeed a valid answer from $L\text{-SQ}_>(f, D)$ to query $(\phi_z(x), \theta_z, \tau)$.

  If $r_{z1} \notin \texttt{Bene}$ then a representation from $\texttt{Neut}$ will be chosen according to its relative probability. This means that with probability at least $1 - \Delta$, $r_{k+1} = r_{z0}$. In this case $r_{z1} \notin \texttt{Bene}$ implies that $v(r_{z1}) - v(r_z) < t$. By the same argument as in the previous case, this implies that

  $$L\texttt{Perf}_f(\phi_z, D) \leq \frac{\theta_z \cdot q}{\tau}(t + 2\tau') \leq \theta_z + \tau \ .$$

  Therefore $b = 0$ is indeed a valid answer from $L\text{-SQ}_>(f, D)$ to query $(\phi_z(x), \theta_z, \tau)$.

- $\theta_z < 0$: If $r_{z1} \notin \texttt{Bene} \cup \texttt{Neut}$ then $r_{k+1} = r_{z0}$ and $v(r_z) - v(r_{z1}) \geq t$. By the same argument as in the previous cases we have that

  $$v(r_z) - v(r_{z1}) \leq \frac{\tau}{\theta_z \cdot q} L\texttt{Perf}_f(\phi_z, D) + 2\tau' \ .$$

  That is,

  $$L\texttt{Perf}_f(\phi_z, D) \leq \frac{\theta_z \cdot q}{\tau}(t - 2\tau') = \theta_z - \theta_z \cdot \tau \leq \theta_z + \tau.$$

Therefore $b = 0$ is indeed a valid answer from L-SQ$_>(f, D)$ to query $(\phi_z(x), \theta_z, \tau)$.

If $r_{z1} \in \texttt{Bene} \cup \texttt{Neut}$ then either a representation from $\texttt{Neut}$ will be chosen according to its relative probability or r_{z1} will be output. By the definition, $\mathbf{Pr}_A(r_z, r_{z1}) = 1-\Delta$, and therefore in either case with probability at least $1 - \Delta$, $r_{k+1} = r_{z1}$. The condition $r_{z1} \in \texttt{Bene} \cup \texttt{Neut}$ implies that $v(r_z) - v(r_{z1}) \leq t$. By the same argument as in the previous case, we obtain that

$$v(r_z) - v(r_{z1}) \geq \frac{\tau}{\theta_z \cdot q} L\texttt{Perf}_f(\phi_z, D) - 2\tau' \ .$$

Therefore

$$L\texttt{Perf}_f(\phi_z, D) \geq \theta_z - \tau$$

and $b = 1$ is indeed a valid answer from L-SQ$_>(f, D)$ to query $(\phi_z(x), \theta_z, \tau)$.

The property of each step that we proved implies that with probability at least $1 - \epsilon/4$, $r_{q(n,1/\epsilon)} = r_z$, where z is of length $q(n, 1/\epsilon)$ and for each $i \in [q(n, 1/\epsilon)]$, z_i is a valid answer to query $(\phi_{z^{i-1}}(x), \theta_{z^{i-1}}, \tau)$. By the properties of B', this in turn implies that h_z (the output of B' on responses z) satisfies $L\texttt{Perf}_f(h_z, D) \geq 1-\epsilon$. The neighborhood of r_z is $\{r_z, h_z\}$. If $h_z \in \texttt{Bene} \cup \texttt{Neut}$ then $\texttt{SelNB}$ will output h_z with probability at least $1 - \Delta$. Otherwise, $\texttt{SelNB}$ will output r_z. This only holds if

$$L\texttt{Perf}_f(r_z, D) - L\texttt{Perf}_f(h_z, D) \geq t - 2\tau' > 0 \ .$$

Hence $L\texttt{Perf}_f(r_z, D) > 1 - \epsilon$.

Therefore, under our assumptions on empirical performance estimates and empirical neighborhoods, with probability at least $1-\epsilon/4$, $L\texttt{Perf}_f(r_g, D) > 1-\epsilon$. This implied that $L\texttt{Perf}_f(r_g, D) > 1 - \epsilon$ with probability at least $1 - 3\epsilon/4$. □(Cl. A.2)

So far, in the definition of R we have assumed that B' is deterministic. A randomized algorithm might produce different queries for different settings of its random bits and might have some additional probability of failure. Therefore to handle randomized algorithms the representation class R needs to be defined to include representations that will be computed for every possible setting of B's random bits. Let ℓ be the number of random bits required to achieve failure probability of at most $\epsilon/4$ and let $R = \{r_\lambda\} \cup \{R_v \mid v \in \{0,1\}^\ell\}$ where R_v is the representation class for B' with its random bits set to v (as described above). On input r_λ M outputs $r_{v,\lambda}$ for a randomly and uniformly chosen $v \in \{0,1\}^\ell$. By, symmetry and the semantics of $\texttt{SelNB}[L, t, s, s]$, after the first step the evolutionary algorithm will be at representation $r_{v,\lambda}$ for a uniformly chosen $v \in \{0,1\}^\ell$. From there the analysis of the deterministic case applies. The total probability of failure of the resulting evolutionary algorithm is at most $3\epsilon/4 + \epsilon/4 = \epsilon$.

Finally, we need to establish that A is an efficient evolutionary algorithm. The efficiency of B implies that the representation class R is polynomially evaluatable and M is efficiently computable. The bounds on $g(n, 1/\epsilon)$, $s(n, 1/\epsilon)$ and $1/t(n, 1/\epsilon)$ are polynomial in n and $1/\epsilon$. □(Th. A.1)

We now describe how to create evolutionary algorithms that do not require initialization, that is converge when started in any state. This construction is a simple modification of the construction in [Fel08] that uses variable tolerance. The idea of the modification is to use the fact that evolutionary mechanisms that we create inexorably reach some representation with a final hypothesis (that is they do not get "stuck"). Once the algorithm is in one of such representations it is possible to test whether the representation computes a hypothesis with appropriately high performance. If this does not hold the algorithm if forced to "re-initialize" and to start from the initial representation r_λ.

Theorem A.3 (Restated 5.1) *Let L be an admissible loss function and C be a concept class L-SQ learnable over a class of distributions $\mathcal{D}$ by a randomized polynomial-time algorithm B. There*

exists a polynomial $s(\cdot,\cdot)$, inverse-polynomial $t(\cdot,\cdot)$ and an evolutionary algorithm $A=(R,M)$ such that for every n and ϵ, C is evolvable by A over $\mathcal{D}$ in $\texttt{SelNB}[L, t(n,1/\epsilon), s(n,1/\epsilon), s(n,1/\epsilon)]$.

Proof: Let $A'=(R',M')$ be the evolutionary algorithm obtained in the proof of Theorem A.1. We will modify A' to produce A that does not require initialization. First note that, if A' is started in representation $r_z \in R'$ for some vector z of partial replies to L-SQ$_>$ queries, then with probability at least $1-\epsilon/2$, after at most $q(n,1/\epsilon)$ steps, the algorithm will reach either representation $r_{z'}$ for some $z' \in \{0,1\}^q$ or representation $h \in \mathcal{H}$. This is true since in every step an answer to a L-SQ$_>$ query is recorded in the current representation as before. Therefore it is sufficient to prove convergence from representations in the set

$$\left\{r_{z'} \mid z' \in \{0,1\}^q\right\} \bigcup \mathcal{H}.$$

We first handle the convergence from a representation h that has performance lower than $1-\epsilon$ (this can only happen if the algorithm was not started from r_λ). Our fix consists of checking whether $L\texttt{Perf}_f(h,D) \geq 1-\epsilon$ and if not, "re-initializing" the evolutionary algorithm.

To check this condition we add a representation h_0^- that computes $h(x)$ with probability $1-\frac{t}{1-\epsilon}$ and $\Phi_0(x)$ with probability $\frac{t}{1-\epsilon}$. Note that

$$L\texttt{Perf}_f(h_0^-,D) = (1-\frac{t}{1-\epsilon})L\texttt{Perf}_f(h,D)\ .$$

Therefore

$$L\texttt{Perf}_f(h,D) - L\texttt{Perf}_f(h_0^-,D) = \frac{L\texttt{Perf}_f(h,D)}{1-\epsilon}t \geq t$$

if and only if $L\texttt{Perf}_f(h,D) \geq 1-\epsilon$. Therefore, if this representation is not deleterious then $L\texttt{Perf}_f(h,D) < 1-\epsilon$ and the algorithm will evolve into h_0^- for an appropriate choice of $\mathbf{Pr}_A(h,h_0^-)$. Otherwise (if $L\texttt{Perf}_f(h,D) \geq 1-\epsilon$), the algorithm will remain in representation h. Note however that the transitions are based on imprecise empirical estimates of performance and not on the actual performance. To handle this imprecision we can assume that B produces a hypothesis with performance at least $1-\epsilon/2$ whereas we have to "re-initialize" only if the performance is less than $1-\epsilon$. It is possible to distinguish between these situations even when estimates of performance are imprecise. In particular, precision of $\epsilon \cdot t/4$ will be sufficient. We omit the straightforward details of this and other analogous modifications to simplify the presentation.

Formally, for $h \in \mathcal{H}$, we define

- $\texttt{Neigh}_A(h,\epsilon) = \{h, h_0^-\}$;
- $\mathbf{Pr}_A(h,h) = \Delta$, $\mathbf{Pr}_A(h,h_0^-) = 1-\Delta$;

To "re-initialize" the algorithm we add a sequence of representations with performance gradually approaching 0. The difference in the performance of any two adjacent representations in the sequence is less than t and therefore each of the mutations in the sequence will always be neutral. For every integer $i \leq 1/t-1$, we define a representation $h_i^-(x)$ that computes the function $h_0^-(x)$ with probability $(1-i\cdot t)$ and $\Phi_0(x)$ otherwise. Further for $i \in \{0,1,\ldots,\lceil 1/t\rceil - 1\}$, we define:

- $\texttt{Neigh}_A(h_i^-) = \{h_{i+1}^-\}$; when $i = \lceil 1/t\rceil - 1$, $h_{\lceil 1/t\rceil}^-$ refers to r_λ.
- $\mathbf{Pr}_A(h_i^-, h_{i+1}^-) = 1$;

With this definition, for every $i \leq \lceil 1/t \rceil - 2$,

$$|L\texttt{Perf}_f(h_i^-, D) - L\texttt{Perf}_f(h_{i+1}^-, D)| = |L\texttt{Perf}_f(h_0^-, D) \cdot t| < t \ ,$$

and

$$|L\texttt{Perf}_f(h_{\lceil 1/t \rceil - 1}^-, D) - L\texttt{Perf}_f(r_\lambda, D)| \leq |L\texttt{Perf}_f(h_0^-, D) \cdot t| \leq t \ .$$

Therefore $h_{i+1}^- \in \texttt{Neut}$ whenever the algorithm is at representation h_i^-. This implies that, with high probability, after at most $1/t$ steps the algorithm will evolve to r_λ. When the algorithm reaches r_λ the usual analysis applies. Also note that if the evolutionary algorithm starts in any of the new representations h_i^-, it will evolve to r_λ after at most $1/t - 1$ steps.

Testing and "re-initialization" at a representation $r_{z'}$ for $z' \in \{0,1\}^q$ can be done in exactly the same way. We can always assume that $L\texttt{Perf}_f(r_{z'}, D) \leq 1/2$ (as this can always be achieved by using $q(n, 1/\epsilon)$ which is twice the actual bound on the number of queries). Then the "re-initialization" sequence will need to be taken only if $h_{z'}$ is not a beneficial mutation from $r_{z'}$. Formally we define the following neighborhood of $r_{z'}$

- $\texttt{Neigh}_A(r_{z'}, \epsilon) = \{r_{z',0}^-, h_{z'}\}$, where $r_{z',0}^-(x) \equiv r_{z'}(x)$.
- $\mathbf{Pr}_A(r_{z'}, h_{z'}) = \Delta,\ \mathbf{Pr}_A(r_{z'}, r_{z',0}^-) = 1 - \Delta$.

The "reinitialization" path from $r_{z',0}^-$ to r_λ is defined exactly as for h_0^-.

Finally, note that the upper bound on the number of generations required for convergence in this construction is at most $2q(n, 1/\epsilon) + 1/t + 1$ and, in particular, remains polynomial. □

Our proof of Theorem 5.1 can be easily used to show that evolvability with the optimizing selection rule $\texttt{SelOpt}$ is equivalent to learnability by L-SQ.

Theorem A.4 (Restated 5.1) *Let L be an admissible loss function and C be a concept class L-SQ learnable over a class of distributions $\mathcal{D}$ by a randomized polynomial-time algorithm B. There exists a polynomial $s(\cdot,\cdot)$, inverse-polynomial $t(\cdot,\cdot)$ and an evolutionary algorithm $A = (R, M)$ such that C is evolvable by A over $\mathcal{D}$ in $\texttt{SelOpt}[L, t(n, 1/\epsilon), s(n, 1/\epsilon), s(n, 1/\epsilon)]$.*

Proof: We use the same algorithm A as in the proof of Theorem 5.1. We first observe that for every $r \in R$ and every $\epsilon > 0$, $\texttt{Neigh}_A(r, \epsilon)$ contains at most two representations. In this situation the semantics of $\texttt{SelOpt}[L, t(n, 1/\epsilon), s(n, 1/\epsilon), s(n, 1/\epsilon)]$ is the same as the semantics of $\texttt{SelNB}[L, t(n, 1/\epsilon), s(n, 1/\epsilon), s(n, 1/\epsilon)]$. The only exception is the representation r_λ that contains $r_{\lambda,v}$ for every $v \in zo^\ell$ in its neighborhood. In this case, considerations of symmetry still apply and $r_{\lambda,v}$ for a random and uniformly chosen v will be output by $\texttt{SelOpt}[L, t(n, 1/\epsilon), s(n, 1/\epsilon), s(n, 1/\epsilon)]$. □

A.2 Section 6

Theorem A.5 (Restated 6.1) *Let C be a concept class CSQ learnable over a class of distributions $\mathcal{D}$ by a polynomial-time algorithm B. There exists an inverse polynomial $t(n, 1/\epsilon)$ such that for every inverse polynomial $\gamma(n, 1/\epsilon)$ there exists a mutation algorithm $A_\gamma = (R_\gamma, M_\gamma)$ that evolves C over $\mathcal{D}$ in Π with initialization for every (t, γ)-distinguishing selection rule Π.*

Proof: We first use Lemma 6.2 to covert B to the algorithm B_γ that learns C over $\mathcal{D}$ using queries to (τ', γ)-CSQ oracle and assume that B_γ is deterministic. We then use a (t, γ)-distinguishing selection rule for an appropriate t to provide responses to queries for (τ', γ)-CSQ(f, D) oracle in essentially the same way as $\texttt{SelNB}$ was used to answer $\text{CSQ}_>$ queries in the proof of Theorem A.1.

Formally, for $z \in \{0,1\}^i$ and $i \in [q(n,1/\epsilon)]$ we define

$$r_z(x) = \frac{1}{q(n,1/\epsilon)} \sum_{j \le i,\ z_j = 1} \phi_{z^{j-1}}(x) \ ,$$

where $\phi_{z'}(x)$ for any $z' \in \cup_{i \le q-1}\{0,1\}^i$ is defined as in the proof of Theorem A.1 with B_γ being used in place of B'. Note that in this case (when L_1 loss function is used) it suffices to use real-valued deterministic functions in $\mathcal{F}_1^\infty$ (and the randomization is not needed). Equivalently, one can also see $r_z(x)$ as a randomized Boolean hypothesis as described in Section 3.2. Now let

$$R_\gamma = \mathcal{H}_\gamma \cup \{r_\lambda\} \cup \{r_z\}_{i \in [q], z \in \{0,1\}^i} \ ,$$

where $r_\lambda \equiv 0$ and $\mathcal{H}_\gamma$ is the representation class of B_γs hypotheses.

The mutation algorithm $A_\gamma = (R_\gamma, M_\gamma)$ is defined as follows.

1. $r = r_z$ for $z \in \{0,1\}^i$ where $0 \le i \le q(n,1/\epsilon) - 1$: $\mathtt{Neigh}_{A_\gamma}(r,\epsilon) = \{r_{z0}, r_{z1}\}$; $\mathbf{Pr}_{A_\gamma}(r, r_{z0}) = 1/2$,
   $\mathbf{Pr}_{A_\gamma}(r, r_{z1}) = 1/2$.
2. $r = r_z$ for $z \in \{0,1\}^{q(n,1/\epsilon)}$: $\mathtt{Neigh}_{A_\gamma}(r,\epsilon) = \{h_z\}$;
3. $r = h$ for $h \in \mathcal{H}_\gamma$: $\mathtt{Neigh}_{A_\gamma}(r,\epsilon) = \{r\}$.

The proof that A_γ evolves C in Π over $\mathcal{D}$ is similar to the proof of Claim A.2 since a (t,γ)-distinguishing selection rule gives answers to (τ',γ)-CSQs in almost the same way as `SelNB` gives answers to $\mathrm{CSQ}_>$s. Below we provide the details of the analysis.

Let $t = \tau'/q(n,1/\epsilon)$, $g(n,1/\epsilon) = q(n,1/\epsilon) + 1)$, let Π be a (t,γ)-distinguishing selection rule and let $r_0 = r_\lambda, r_1, r_2, \ldots, r_g$ be a sequence of representations produced by $r_k \leftarrow \Pi(f, D, A_\gamma, r_{k-1})$. We claim that for every z of length $k \in \{0,1,\ldots,q-1\}$, if $r_k = r_z$ then $r_{k+1} = r_{zb}$, where b is a random variable distributed according to a valid distribution on answers to query $\phi_z(x)$ from (τ',γ)-$\mathrm{CSQ}(f,D)$ oracle.

According to the definition of A_γ, $\mathtt{Neigh}_{A_\gamma}(r_z,\epsilon) = \{r_{z0}, r_{z1}\}$. Representations r_z and r_{z0} compute function $\phi'(x) = \frac{1}{q(n,1/\epsilon)} \sum_{j \le i,\ z_j = 1} \phi_{z^{j-1}}(x)$ and r_{z1} computes function $\phi'(x) + \phi_z(x)/q(n,1/\epsilon)$. Note that by property (3) of Π, $\mathbf{Pr}_{A_\gamma}(r, r_{z0}) = 1/2$ implies that $\mathbf{Pr}_{A_\gamma,\Pi}(r, \perp) = 0$ and hence Π outputs either r_{z0} or r_{z1}. We can now consider two cases:

1. If $\mathbf{E}_D[\phi_z(x) \cdot f] \ge \tau'$ then $\mathtt{Perf}_f(r_{z1}, D) \ge \mathtt{Perf}_f(\phi', D) + t$. This, by property (1) of Π, implies that
   $$\frac{\mathbf{Pr}_{A_\gamma,\Pi}(r, r_{z0})}{\mathbf{Pr}_{A_\gamma,\Pi}(r, r_{z1})} \le (1-\gamma)\frac{\mathbf{Pr}_{A_\gamma}(r, r_{z0})}{\mathbf{Pr}_{A_\gamma}(r, r_{z1})} = (1-\gamma) \ .$$
   By property (3) of Π, $\mathbf{Pr}_{A_\gamma,\Pi}(r, r_{z1}) = 1 - \mathbf{Pr}_{A_\gamma,\Pi}(r, r_{z0})$ and hence $(1 - \mathbf{Pr}_{A_\gamma,\Pi}(r, r_{z1}))/\mathbf{Pr}_{A_\gamma,\Pi}(r, r_{z1}) \le 1-\gamma$. This is exactly the definition of a valid distribution over answers from (τ',γ)-$\mathrm{CSQ}(f,D)$ oracle.
2. If $\mathbf{E}_D[\phi_{\epsilon,z}(x) \cdot f] \le -\tau'$ then $\mathtt{Perf}_f(\phi', D) \ge \mathtt{Perf}_f(r_{z1}, D) + t$. This, by property (1) of Π, implies that
   $$\frac{\mathbf{Pr}_{A_\gamma,\Pi}(r, r_{z1})}{\mathbf{Pr}_{A_\gamma,\Pi}(r, r_{z0})} \le (1-\gamma)\frac{\mathbf{Pr}_{A_\gamma}(r, r_{z1})}{\mathbf{Pr}_{A_\gamma}(r, r_{z0})} = (1-\gamma) \ .$$
   By the same argument as in the previous case, this implies that Π outputs r_{z0} with probability γ' that satisfies $(1-\gamma')/\gamma' \le 1-\gamma$ and r_{z0} with probability $1-\gamma'$.

Therefore when $r_q = r_z$, z is distributed according to a valid distribution over the answers to all the (τ', γ)-CSQs queries of B_γ. Hence, $r_q = r_z$ such that $z \in \{0,1\}^q$ and $\texttt{Perf}_f(h_z, D) \geq 1 - \epsilon$. We can assume that $\texttt{Perf}_f(r_z, D) \leq 1/2 < 1 - \epsilon$. This, by property (3) of Π implies that $\mathbf{Pr}_{A_\gamma,\Pi}(h_z) = 1$, that is $r_g = h_z$. We handle the case of randomized B_γ in the same way as in the proof of Theorem A.1, that is by adding a special randomizing representation. Here property (2) of Π ensures that for all $u, v \in \{0,1\}^\ell$, $\mathbf{Pr}_{A_\gamma,\Pi}(r_\lambda, r_{v,\lambda}) = \mathbf{Pr}_{A_\gamma,\Pi}(r_\lambda, r_{u,\lambda})$ and property (3) implies that $\mathbf{Pr}_{A_\gamma,\Pi}(r_\lambda, \perp) = 0$. Therefore Π outputs $r_{v,\lambda}$ for a randomly and uniformly chosen v. $\square$