Evolvability from Learning Algorithms

Vitaly Feldman*
IBM Almaden Research Center

April 16, 2008

Abstract

Valiant has recently introduced a framework for analyzing the capabilities and the limitations of the evolutionary process of random change guided by selection [Val06]. In his framework the process of acquiring a complex functionality is viewed as a substantially restricted form of PAC learning of an unknown function from a certain set of functions [Val84]. Valiant showed that classes of functions evolvable in his model are also learnable in the statistical query (SQ) model of Kearns [Kea98] and asked whether the converse is true.

We show that evolvability is equivalent to learnability by a restricted form of statistical queries. Based on this equivalence we prove that for any fixed distribution D over the instance space, every class of functions learnable by SQs over D is evolvable over D. Previously, only the evolvability of monotone conjunctions of Boolean variables over the uniform distribution was known [Val07]. On the other hand, we prove that the answer to Valiant's question is negative when distribution-independent evolvability is considered. To demonstrate this, we develop a technique for proving lower bounds on evolvability and use it to show that decision lists and linear threshold functions are not evolvable in a distribution-independent way. This is in contrast to distribution-independent learnability of decision lists and linear threshold functions in the statistical query model.

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. The fundamental insight of his theory is that evolution is a form of learning 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 as a mechanism of learning is that of computational learning theory [Val84]. Accordingly, in his model, evolvability of 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 certain class of functions is considered evolvable if there exists a single mechanism that guarantees convergence to the desired function for any function in this class. We refer to such mechanisms as *evolutionary algorithms*. Here the

*Part of the work was done while the author was at Harvard University supported by grants NSF-CCF-04-32037 and NSF-CCF-04-27129.

requirements closely follow those of the celebrated PAC learning model introduced by Valiant in 1984 [Val84]. In fact, every evolutionary algorithm (here and below by evolution we mean evolution in Valiant's model) can be simulated by an algorithm that is given random examples of the desired function. As in PAC learning, restriction on evolutionary algorithms make random search through all the possible functions of the given type impossible and thereby force evolutionary algorithms to exploit the underlying structure of the desired function.

As a more concrete example, one can take the desired function to be the function that recognizes predators of an organism given sensory information about the surroundings. The ability of the organism to recognize predators can be thought of as determined by the genome of the organism or by expression of a fixed set of proteins. Then for every genome, mutations would provide the candidate functions, of which the most fit will be the most likely to occur in the future generations. The reader is referred to Valiant's original publication for a detailed discussion of his model.

While the constrained way in which evolutionary algorithms have to find the desired function makes such algorithms more biologically plausible, it also makes designing such algorithms substantially more involved than designing PAC learning algorithms. In fact, the only concept class that was shown to be evolvable prior to this work is the class of monotone conjunctions (or disjunctions) of Boolean variables and only when the distribution on the domain is uniform[1] [Val07]. While the resulting algorithm is relatively simple it required significantly more delicate analysis than PAC learning algorithms for conjunctions. Another related result was recently given by Michael who proved that decision lists (see Section 2 for the definition) are evolvable when the distribution is restricted to be uniform [Mic07] and a slightly different measure of fitness is used to guide the evolutionary algorithm.

Valiant also demonstrates that even the restriction of PAC learning to *statistical queries* is sufficient to simulate any evolutionary algorithm. The statistical query (SQ) learning model was introduced by Kearns and is a natural restriction of the PAC learning that models algorithms that use statistical properties of a data set rather than individual examples [Kea98]. Kearns has shown that any SQ algorithm can be converted to a PAC algorithm that tolerates arbitrarily high levels of random noise (bounded away from the information-theoretic barrier) and since then his model became the main tool in designing noise-tolerant algorithms [Kea98, Byl94, BFKV97, JSS97, BF02, DV04, Fel07]. Surprisingly, almost all the classes of functions (referred to as *concept classes* in the context of learning) known to be PAC learnable are also learnable by statistical queries (and hence learnable in the presence of random noise in the labels). The notable exception is the concept class of parities that includes all functions equal to a XOR of some subset of n Boolean variables. Interestingly, parities are provably not learnable by SQs (unconditionally, i.e. without relying on any computational hardness assumptions) [Kea98, BFJ$^+$94]. This result is a special case of lower bounds based on the *SQ dimension* which was introduced by Blum *et al.* [BFJ$^+$94] and studied in a number of subsequent works [BKW03, BF02, Yan05, KS06, She07]. Limitations on learnability in the SQ model imply limitations on evolutionary algorithms. In particular, parities are not evolvable in Valiant's model. Indeed the class of parities appears to be biologically unnatural and hence the prediction of Valiant's model is consistent with our intuition.

Outside of the biological context, evolutionary algorithms can be seen as learning algorithms that rely on the minimal amount of feedback from the environment. Namely they allow to adapt the behavior of a system given only the information on the overall performance of the system. Furthermore the process of the adaptation is based solely on local and greedy decisions. This is in contrast to the usual learning paradigm which relies on the analysis of instances of

[1] To simplify the discussion, the domain in this problem and all the problems we discuss in this work is $\{0,1\}^n$ and Boolean functions have range $\{-1,1\}$. As in the PAC model of learning, all the models are applicable to other domains and can also be easily extended to non-Boolean functions.

the learning problem together with the correct behavior on these instances. This suggests that evolutionary algorithms might bring the advantages of learning to problems for which previous learning approaches were not applicable.

1.1 Our Contribution

The primary goal of this work is to elucidate the capabilities and the limitations of Valiant's model. To this end we prove that evolvability is equivalent to learning by a natural restriction of statistical queries, referred to as *correlational statistical queries* [BF02]. A correlational statistical query (or CSQ) is a query for the correlation of a given function g with the unknown target function f. The correlation is measured relative to the distribution D over the domain of the learning problem and equals $\mathbf{E}_{x\sim D}[f(x)g(x)]$. To such a query a CSQ oracle returns an estimate of $\mathbf{E}_{x\sim D}[f(x)g(x)]$ within certain tolerance. For comparison, the general SQ model allows queries that provide estimates of $\mathbf{E}_{x\sim D}[\psi(x, f(x))]$ for any function on labeled examples $\psi : \{0,1\}^n \times \{-1,1\} \to \{-1,1\}$. One side of the equivalence (stated in Theorem 4.1) is just a refinement of Valiant's observation that evolvability implies learnability by general statistical queries. For the other direction we prove the following result.

Theorem 1.1 *Let $\mathcal{C}$ be a concept class CSQ learnable over a class of distributions $\mathcal{D}$ by a polynomial-time algorithm $\mathcal{A}$. There exists an evolutionary algorithm $N(\mathcal{A})$ such that $\mathcal{C}$ is evolvable by $N(\mathcal{A})$ over $\mathcal{D}$.*

To prove Theorem 1.1, we first decompose CSQs into correlational queries that only allow comparisons of the correlation with a fixed value. We then show how to build a general "fitness landscape" that provides answers to all the "simple" correlational queries and generates a hypothesis approximating the target function. We are not aware of any similar techniques having been used before.

We give several applications of our equivalence. Building on an observation of Bshouty and Feldman, we show that any SQ learning algorithm over a distribution D can be converted to a CSQ learning algorithm over D. In this result if D is not an efficiently samplable distribution (and not close to one), then the algorithm we obtain is non-uniform, that is, different algorithms are used for problems of different sizes (Theorems 3.2 and 3.3 give formal statements of these results). This implies that every concept class known to be SQ learnable (and hence almost every concept class known to be PAC learnable) is evolvable when the distribution over the domain is fixed. In particular, this includes the concept class of linear threshold functions (that includes conjunctions, disjunctions, and decision lists) [BFKV97], k-term DNF/CNF for constant k [Val84, Kea98], and decision trees of constant rank [Riv87, Kea98] (*cf.* [KV94] for the definitions of the above concept classes).

We also use our equivalence to give lower bounds specific to evolvability. In particular, building on ideas of Blum *et al.* [BFJ$^+$94] and Bshouty and Feldman [BF02], and using Freund's boosting algorithm [Fre95], we show a simple structural characterization of weak distribution-independent evolvability (here *weak* and *distribution-independent* are used in the same sense as in learning; see Section 2.1 for the definitions). Namely, only concept classes in which every concept is representable as a linear threshold with "small" integer weights over a "small" set of basis functions are weakly distribution-independently evolvable. A formal statement of this equivalence is given in Theorem 5.5. Coupled with recent results of Buhrman *et al.* [BVdW07] and Sherstov [She07] our characterization implies the following theorem.

Theorem 1.2 *The concept classes of decision lists and linear threshold functions are not weakly distribution-independently evolvable.*

Decision lists are a subclass of linear threshold functions and we explicitly mention linear threshold functions since the lower-bound is somewhat stronger for them. Theorem 1.2 is contrasted

by a simple algorithm of Kearns that learns decision lists by statistical queries [Kea98] and the algorithm of Blum *et al.* [BFKV97] that shows that linear threshold functions are SQ learnable.

It is important to note that while our positive result implies polynomial-time evolvability of numerous concept classes, the evolutionary algorithms that are obtained from this general transformation are likely not to be the most efficient and/or natural possible for a specific concept class and distribution. In particular, the evolutionary algorithm for monotone conjunctions over the uniform distribution implied by our reduction requires more complex representations of intermediate hypotheses than the one given by Valiant [Val07].

1.2 Organization

Following the preliminaries, in Section 3 we define learning by correlational statistical queries and relate it to the SQ model. In Section 4 we prove the equivalence of evolvability to learning by correlational statistical queries. In Section 5 we give a characterization of weak distribution-independent learnability by correlational statistical queries and the resulting lower bounds on evolvability.

2 Models and Notation

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

The domain of the functions discussed in this work are objects described by n Boolean attributes, or $X_n = \{0,1\}^n$. An ensemble of distributions $D = \{D_n\}_{n=1}^{\infty}$ where D_n is a distribution over X_n is said to be samplable efficiently or *p-samplable* [BDCGL92] if there exists a randomized polynomial-time algorithm S that for every n and $x \in X_n$, on input 1^n outputs x with probability $D_n(x)$.

A *concept class* over X is a set of $\{-1,1\}$-valued functions (also called *concepts*) over X. 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 a polynomial-time algorithm, that given a representation of a function g and a point $x \in X_n$, computes $g(x)$ in time polynomial in x and the length of the representation of f. 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.

The concept class of conjunctions consists of functions equal to a conjunction of a subset of possibly negated variables. A conjunction is monotone if it does not include negated variables. A parity function is a function equal to $(-1)^{\sum_{i \in S} x_i}$ for some $S \subseteq [n]$. The concept class of linear threshold functions consists of functions representable as $\mathtt{sign}(\sum_{i \in [n]} w_i x_i - \theta)$ for some $w \in \mathbb{R}^n$ and $\theta \in \mathbb{R}$. We denote it by $\mathtt{TH}_n$. A *decision list* of length k is described by a list $((\ell_1, b_1), (\ell_2, b_2), \ldots, (\ell_k, b_k), b_{k+1})$, where for each i, ℓ_i is a variable or its negation and b_i is a Boolean value. The value of a decision list on a point x is b_i where i is the smallest index for which $\ell_i(x) = 1$, and b_{k+1} if no such i exists. We denote the class of functions representable by decision lists over n variables by $\mathtt{DL}_n$.

2.1 PAC Learning

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

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

An algorithm is said to PAC learn $\mathcal{C}_n$ in time t if for every n, $\epsilon > 0$, $\delta > 0$, $f \in \mathcal{C}_n$, and distribution D_n over X_n, the algorithm given ϵ, δ, and access to $\mathrm{EX}(f, D_n)$ outputs, in time t and with probability at least $1 - \delta$, a hypothesis h that is computable in time t and satisfies[2] $\mathbf{E}_{D_n}[f(x) \cdot h(x)] \geq 1 - \epsilon$. Here t is allowed to depend on n, $1/\epsilon$, $1/\delta$ and the minimum description length of f under the representation scheme of $\mathcal{C}_n$. For simplicity we only consider representation classes for which the description length of all functions in the class is bounded by a certain polynomial in n. This allows us to drop the description length from the polynomial bounds. We also drop subscript n when it is clear from the context.

The algorithm is said to (efficiently) learn $\mathcal{C}$ if t is polynomial in n, $1/\epsilon$ and $1/\delta$. We say that an algorithm learns $\mathcal{C}$ by a representation class $\mathcal{H}$ if the hypotheses output by the algorithm use the representation scheme of $\mathcal{H}$.

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 $\mathcal{D}$.

A *weak* learning algorithm is a learning algorithm that produces a hypothesis whose disagreement with the target concept is noticeably less than $1/2$ (and not necessarily any $\epsilon > 0$). More precisely, a weak learning algorithm produces a hypothesis h such that $\mathbf{E}_D[f(x) \cdot h(x)] \geq 1/p(n)$ for some fixed polynomial p.

2.2 The Statistical Query Learning Model

In the *statistical query model* of Kearns 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 pair (ψ, τ), where $\psi : \{0,1\}^n \times \{-1,1\} \to \{-1,1\}$ is a *query function* and $\tau \in [0,1]$ is a real number called the *tolerance* of the query. The oracle may respond to the query with any value v satisfying $|\mathbf{E}_D[\psi(x, f(x))] - v| \leq \tau$. For convenience, we allow the query functions to be real valued in the range $[-1,1]$. This extension was introduced by Aslam and Decatur [AD98] and is equivalent to the original SQ model (*cf.* [BF02]).

An algorithm $\mathcal{A}$ is said to learn $\mathcal{C}$ in time t from statistical queries of tolerance τ if $\mathcal{A}$ PAC learns $\mathcal{C}$ using $\mathrm{STAT}(f, D)$ instead of $\mathrm{EX}(f, D)$ and for each query (ψ, τ') made by $\mathcal{A}$, ψ can be evaluated in time t and $\tau' \geq \tau$. Eliminating the dependence on random examples eliminates the confidence parameter δ necessary in the definition of the PAC model.

The algorithm is said to (efficiently) learn $\mathcal{C}$ if t is polynomial in n and $1/\epsilon$, and τ is lower bounded by the inverse of a polynomial in n and $1/\epsilon$. If the learning algorithm is randomized we require that it succeeds with probability at least $1/2$.

2.3 Evolvability

In this section we provide a formal definition of Valiant's evolvability model [Val07]. Our notation follows Valiant's with insignificant simplifying deviations.

Let f denote the unknown "ideal" function and D be a distribution over the domain. As in PAC learning, let ϵ denote the desired accuracy of approximation. For brevity, in Valiant's

[2]This is equivalent to $\mathbf{Pr}_{D_n}[f(x) = h(x)] \geq 1 - \epsilon/2$.

model ϵ is also used as the confidence parameter (that is $\delta = \epsilon$). The *performance* of a Boolean function r relative to f over D is defined as $\texttt{Perf}_f(r, D) = \mathbf{E}_D[r(x) \cdot f(x)]$. For an integer s the *empirical performance* $\texttt{Perf}_f(r, D, s)$ of r is a random variable that equals $\frac{1}{s}\sum_{i \le s}(r(z_i) \cdot f(z_i))$ for $z_1, z_2, \ldots, z_s \in \{0,1\}^n$ chosen randomly and independently according to D.

An *evolutionary algorithm* or "fitness landscape" N is defined by a quadruple $(R, \text{Neigh}, \mu, t)$ where:

- R is a representation class;
- $\text{Neigh}(r, \epsilon)$ is a function that for $r \in R$, equals the *neighborhood* of r, that is, the set of representations into which r randomly "mutates". For all r and ϵ, $r \in \text{Neigh}(r, \epsilon)$ and $|\text{Neigh}(r, \epsilon)| \le p_N(n, 1/\epsilon)$ for a fixed polynomial p_N.
- $\mu(r, r_1, \epsilon)$ is a function that for $r \in R$ and $r_1 \in \text{Neigh}(r, \epsilon)$, gives the probability that r "mutates" into r_1;
- $t(r, \epsilon)$ is the function that equals the *tolerance* at r. The tolerance determines the difference in performance that a "mutation" has to exhibit to be considered beneficial (or deleterious). The tolerance is bounded from above and below by two polynomially-related polynomials in $1/n$ and ϵ. That is, there exist a polynomial $\text{tu}_N(1/n, \epsilon)$ and a constant $\eta \ge 1$ such that for all $r \in R$ and $\epsilon > 0$, $\text{tu}_N(1/n, \epsilon) \ge t(r, \epsilon) \ge \text{tu}_N^{\eta}(1/n, \epsilon)$.

In addition, functions Neigh, μ, and t need to be computable by randomized Turing machines in time polynomial in n and $1/\epsilon$.

We now give the definition of the basic step of an evolutionary algorithm. For a function f, distribution D, evolutionary algorithm $N = (R, \text{Neigh}, \mu, t)$, a representation $r \in R$, accuracy ϵ, and sample size s, the *mutator* $\text{Mu}(f, D, N, r, \epsilon, s)$ is a random variable that takes a value r_1 determined as follows. For each $r' \in \text{Neigh}(r, \epsilon)$, it first computes an empirical value of $v(r') = \texttt{Perf}_f(r', D, s)$. Let $\text{Bene} = \{r' \mid v(r') \ge v(r) + t(r, \epsilon)\}$ and $\text{Neut} = \{r' \mid |v(r') - v(r)| < t(r, \epsilon)\}$. Then

(i) if $\text{Bene} \neq \emptyset$ then output $r_1 \in \text{Bene}$ with probability $\mu(r, r_1, \epsilon) / \sum_{r' \in \text{Bene}} \mu(r, r', \epsilon)$;

(ii) if $\text{Bene} = \emptyset$ then output $r_1 \in \text{Neut}$ with probability $\mu(r, r_1, \epsilon) / \sum_{r' \in \text{Neut}} \mu(r, r', \epsilon)$.

Definition 2.1 *Let $\mathcal{C}$ be a concept class, D be a distribution and $N = (R, Neigh, \mu, t)$ be an evolutionary algorithm. Concept class $\mathcal{C}$ is* evolvable *by N over D if there exist polynomials $s(n, 1/\epsilon)$ and $g(n, 1/\epsilon)$ such that for every n, every $f \in \mathcal{C}_n$, every $\epsilon > 0$, and every $r_0 \in R_n$, with probability at least $1 - \epsilon$, a sequence $r_0, r_1, r_2, \ldots$, where $r_i = Mu(f, D, N, r_{i-1}, \epsilon, s(n, 1/\epsilon))$ will have $\texttt{Perf}_f(r_{g(n,1/\epsilon)}, D_n) > 1 - \epsilon$.*

The polynomial $g(n, 1/\epsilon)$ upper bounds the number of generations needed for the evolution process. The polynomial $s(n, 1/\epsilon)$ upper bounds the size of the population required to test each mutation.

A concept class $\mathcal{C}$ is evolvable *with initialization* if convergence is required only when the evolution starts from a single fixed representation r_0. A concept class $\mathcal{C}$ is evolvable over D if there exists an evolutionary algorithm N such that $\mathcal{C}$ is evolvable by N over D. It is easy to see that evolutionary algorithms are a restricted form of PAC learning algorithms. Therefore we use the term *weak* evolvability in the same sense as for the PAC learning. A concept class $\mathcal{C}$ is evolvable if it is evolvable over all distributions by a single evolutionary algorithm (we emphasize this by saying *distribution-independently* evolvable). Similarly, we say that a concept class $\mathcal{C}$ is evolvable over a set of distributions $\mathcal{D}$ if it is evolvable over every $D \in \mathcal{D}$ (by a single evolutionary algorithm).

In this work we allow the representation class of an evolutionary algorithm to compute real-valued functions from $\{0,1\}^n$ to $[-1, 1]$ (and not only to $\{-1, 1\}$). Performance of a function $\phi : \{0,1\}^n \to [-1, 1]$ relative to the ideal function f over D_n is defined as before $\texttt{Perf}_f(\phi, D_n) =$

$\mathbf{E}_{D_n}[f(x)\phi(x)]$. Instead of thinking of such hypotheses as real-valued functions one can also think of them as randomized Boolean functions. Namely, let $\Phi(x)$ denote a random variable that equals 1 with probability $(1+\phi(x))/2$, and -1 otherwise (and hence its expectation is $\phi(x)$). Here we define the performance of $\Phi(x)$ to be $\mathtt{Perf}_f(\Phi, D_n) = \mathbf{E}_{\Phi,D_n}[f(x)\Phi(x)]$ (that is the expectation also includes the coin flips of Φ). Note that $\mathtt{Perf}_f(\Phi, D_n) = \mathtt{Perf}_f(\phi, D_n)$ and with either definition the evaluation of performance through random sampling can be done as in the original model. Removing this restriction imposed by Valiant for simplicity makes the design of evolutionary algorithms considerably more convenient. In addition, in Theorem A.3 (Appendix A) we prove that when evolving over any set of unconcentrated (that is without points with non-negligible weight) distributions, the random coin flips can be replaced by k-wise independent coin flips for polynomial k without affecting the behavior of the evolutionary algorithm. For evolutionary algorithms produced by our reduction from CSQ algorithms the above is true also when evolving over any fixed p-samplable distribution (not only unconcentrated). In particular, Corollaries 4.5 and 4.6 and similar results do not depend on this restriction.

3 Correlational Statistical Queries

In this section we define a restricted class of statistical query algorithms and relate it to the general SQ model. According to the definition of statistical query, the query function $\psi : \{0,1\}^n \times \{-1,1\} \to [-1,1]$ is a function of two parameters: the point and the label. We distinguish two types of statistical queries based on their query function. We say that a query is *target-independent* if $\psi(x,\ell) \equiv \phi(x)$ for a function $\phi : \{0,1\}^n \to [-1,1]$, that is, if ψ is a function of the point x alone. We say that a statistical query is *correlational*, if $\psi(x, f(x)) \equiv \phi(x)f(x)$ for a function $\phi : \{0,1\}^n \to [-1,1]$. We say that an algorithm is a correlational statistical query (CSQ) algorithm if it uses only correlational statistical queries. We use the following simple fact by Bshouty and Feldman [BF02] to relate learning by statistical queries to learning by CSQs (we include the proof for completeness).

Lemma 3.1 ([BF02]) *Any statistical query (ψ, τ) with respect to any distribution D can be answered using a statistical query that is target-independent and a correlational statistical query, each of tolerance $\tau/2$ and with respect to distribution D.*

Proof: The following equation proves the statement:

$$\begin{aligned}\mathbf{E}_D[\psi(x,f(x))] &= \mathbf{E}_D\left[\psi(x,-1)\frac{1-f(x)}{2} + \psi(x,1)\frac{1+f(x)}{2}\right] \\ &= \mathbf{E}_D\left[\frac{\psi(x,1)-\psi(x,-1)}{2}f(x)\right] + \mathbf{E}_D\left[\frac{\psi(x,1)+\psi(x,-1)}{2}\right] .\end{aligned}$$

□

A query that is target-independent can be answered by estimating the expectation of a random variable that does not depend on f. This can be done by random sampling given points randomly generated according to D. For example, if the target distribution is uniform over the domain, then a learning algorithm can estimate the desired expectations by evaluating them on randomly and uniformly generated points in $\{0,1\}^n$. More generally, for any distribution that can be sampled efficiently, we can use the sampling algorithm to create random samples from the distribution and use them to evaluate answers to queries that are independent of f. Therefore Lemma 3.1 implies that given the ability to randomly sample points according to D it is possible to simulate general statistical queries using only correlational statistical queries.

Theorem 3.2 *If $\mathcal{C}$ is a representation class SQ learnable over a p-samplable ensemble of distributions $D = \{D_n\}_{n=1}^{\infty}$ then $\mathcal{C}$ is CSQ learnable over D.*

A somewhat weaker version of this result also holds for general distributions.

Theorem 3.3 *Let $\mathcal{C}$ be a representation class SQ learnable over an arbitrary ensemble of distributions $D = \{D_n\}_{n=1}^{\infty}$. There exists a non-uniform polynomial-time algorithm that learns $\mathcal{C}$ over D from correlational statistical queries alone.*

Proof: Let $\mathcal{A}$ be an SQ algorithm for learning $\mathcal{C}$, let $t(n, 1/\epsilon)$ be a polynomial upper bounds on the running time of $\mathcal{A}$ and let $\tau(n, 1/\epsilon)$ be the lower bound on the tolerance of each SQ of $\mathcal{A}$. Hoeffding's bounds imply that estimating a target-independent statistical query to accuracy $\tau(n, 1/\epsilon)$ with confidence at least $1 - \delta$ can be done using a random independent sample of points from D of size $O(\tau^{-2}(n, 1/\epsilon) \log(1/\delta))$ [Hoe63]. Let $Q(n, \epsilon)$ be the set of all the target-independent queries that $\mathcal{A}$ might ask in any of its legal executions for the given n and ϵ (that is for all settings of random coins and all possible responses from a statistical query oracle). It is easy to see that $|Q(n, \epsilon)| \leq 2^{t(n,1/\epsilon)}$. Therefore by using a random sample of size

$$O(\tau^{-2}(n, 1/\epsilon) \ln(Q(n, \epsilon))) = O(\tau^{-2}(n, 1/\epsilon) \cdot t(n, 1/\epsilon)) \ ,$$

the estimation of each possible target-independent statistical query of $\mathcal{A}$ will be within the desired tolerance with probability at least $1/2$. This implies that there exists a sample of polynomial size that can be used to estimate all the target-independent statistical queries of $\mathcal{A}$ within the desired tolerance. Therefore given such sample as part of the description of the learning algorithm (also referred to as *advice*) we can simulate $\mathcal{A}$ with access only to CSQs. We remark that for most algorithms a significantly better estimates of $Q(n, \epsilon)$ are possible and hence significantly smaller sample will be sufficient to estimate all the target-independent queries of $\mathcal{A}$. □

We also note that certain statistical query algorithms require access to randomly unlabeled points from the target distribution in addition to statistical queries (such as the algorithm of Blum *et al.* for learning linear threshold functions [BFKV97]). Clearly, the reductions in Theorems 3.2 and 3.3 support such algorithms.

4 Equivalence of Learnability from CSQs and Evolvability

In this section we prove that a concept class is learnable via correlational statistical queries if and only if it is evolvable. We first note that one of the directions of this equivalence is immediate. As it was observed by Valiant, concept classes that are evolvable are also learnable from statistical queries [Val07]. The statistical queries are only used to replace evaluations of empirical performance of functions in R in order to find the sets Bene and Neut at every step. By the definition of $\texttt{Perf}_f$, these estimates are correlational queries.

Theorem 4.1 *If $\mathcal{C}$ is evolvable over a class of distributions $\mathcal{D}$ then $\mathcal{C}$ is learnable from correlational statistical queries over $\mathcal{D}$.*

4.1 Evolutionary Algorithms from CSQ Algorithms

To prove the other direction, of the equivalence we first prove that learning from correlational statistical queries is equivalent to learning from a further restricted form of correlational statistical queries. Namely, for a function f and a distribution D we define an oracle $\mathrm{CSQ}_{>}(f, D)$ to be the oracle that accepts queries of the form (r, θ, τ) where r is a function from $\{0, 1\}^n$ to $[-1, 1]$, $\theta > 0$ is the *threshold* and $\tau > 0$ is the tolerance. To such a query (referred to as a $\mathrm{CSQ}_{>}$ query) the oracle returns 1 when $\mathbf{E}_D[r(x)f(x)] \geq \theta + \tau$, 0 when $\mathbf{E}_D[r(x)f(x)] \leq \theta - \tau$, and either 0 or 1, otherwise. Clearly, a single CSQ (r, τ) is sufficient to simulate query (r, θ, τ) to $\mathrm{CSQ}_{>}(f, D)$. We prove that the converse is also true in the following lemma.

Lemma 4.2 *For every function f and distribution D, a correlational query (r, τ) can be replaced by $\lceil \log(1/\tau) \rceil + 1$ queries to $CSQ_>(f, D)$. Each of the produced queries is of the form (r', θ, τ'), where $r' \in \{r, -r\}$, $\tau' = \tau/4$, and $\theta \geq \tau/4$.*

Proof: Let $v = \mathbf{E}_D[r(x)f(x)]$. Queries to $\mathrm{CSQ}_>(f, D)$ allow to perform comparisons of v with any value in $(0, 1]$ up to accuracy $\tau/2$. In addition by asking queries on function $-r(x)$ it is possible to compare v with any value in $[-1, 0)$ up to accuracy $\tau/2$. Therefore we can perform a binary search on the interval $[-1, 1]$ until v is confined to an interval $[v_\ell, v_u]$ such that $v_u - v_\ell \leq \tau$. Due to possible imprecision of the answers from $\mathrm{CSQ}_>(f, D)$ we can only conclude that $v \in [v_\ell - \tau/2, v_u + \tau/2]$. Let $v' = (v_\ell + v_u)/2$. By the properties of the search, $|v' - v| \leq \tau$ and hence is a valid answer to the query (r, τ). To ensure that for every query the threshold θ is at least $\tau/4$, we replace every query $(r', \theta, \tau/2)$ with $\theta \in [0, \tau/4]$ by query $(r', \tau/4, \tau/4)$. This query satisfies the requirements of the lemma and any valid answer to this query is also a valid answer to query $(r', \theta, \tau/2)$. □

It is easy to observe that evolvability relies on comparisons of the performance of candidate hypotheses to certain threshold values. In this sense evolvability is very similar to learning from queries to a $\mathrm{CSQ}_>$ oracle. We now describe our main construction that utilizes this similarity to simulate queries to a $\mathrm{CSQ}_>$ oracle in an evolutionary algorithm. To simplify the presentation, we first give a version of the construction that handles only deterministic CSQ algorithms and requires initialization. In Section 4.2 we show a construction that does not require initialization and also allows randomization.

Theorem 4.3 *Let $\mathcal{C}$ be a concept class CSQ learnable over a class of distributions $\mathcal{D}$ by a deterministic polynomial-time algorithm $\mathcal{A}$. There exists an evolutionary algorithm $N(\mathcal{A}) = (R, Neigh, \mu, t)$ such that $\mathcal{C}$ is evolvable by $N(\mathcal{A})$ over $\mathcal{D}$ with initialization.*

Proof: Let $\mathcal{H}$ be the representation class of $\mathcal{A}$'s hypotheses. We first apply Lemma 3.1 to convert $\mathcal{A}$ to algorithm $\mathcal{A}'$ that uses only queries to $\mathrm{CSQ}_>(f, D)$. Let $q(n, 1/\epsilon)$ be a polynomial upper bound on the number of queries asked by $\mathcal{A}'$ and let $\tau(n, 1/\epsilon)$ denote the lower bound on the tolerance and thresholds of the queries asked by $\mathcal{A}'$. We can safely assume that all the queries are asked with tolerance τ. For $i \in [q]$ and $z \in \{0, 1\}^{i-1}$, let $(\phi_{\epsilon,z}(x), \theta_{\epsilon,z}, \tau)$ denote the i^{th} query that $\mathcal{A}'$ 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 $h_{\epsilon,z}$ the hypothesis produced by $\mathcal{A}'$ given that its queries were answered according to z (we can assume for simplicity that exactly q queries are asked in every possible execution of $\mathcal{A}'$).

The high-level idea of our construction is to get the answer to a $\mathrm{CSQ}_>$ (ϕ, θ, τ) of $\mathcal{A}'$ by trying to "add" $\phi(x)$ to the current hypothesis. For an appropriate choice of the threshold, this augmented hypothesis will be chosen as the next generation if and only if $\mathbf{E}_D[f(x)\phi(x)] \geq \theta$. This allows the evolutionary algorithm to "record" the answer to the $\mathrm{CSQ}_>$. All the answers that were obtained so far are "remembered" in the current representation. Finally, given the answers to all the queries of $\mathcal{A}'$, the evolutionary algorithm mutates into the representation equal to the final hypothesis of $\mathcal{A}'$ for the given answers.

Queries produced by $\mathcal{A}'$ and their number might depend on the value of ϵ. Therefore our simulation will require different hypothesis representations for different values of ϵ. To avoid creating a separate set of states for every value of ϵ we create a set of states for each $\epsilon \in \{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots, 2^{-n}\}$. To simulate the evolutionary algorithm for accuracy ϵ' we simulate the algorithm on $\epsilon = 2^{\lfloor \log \epsilon' \rfloor}$ (i.e. the largest power of 2 that does not exceed ϵ'). This will not affect any polynomial bounds.

We will now define the evolutionary algorithm $N(\mathcal{A}) = (R, \mathrm{Neigh}, \mu, t)$ for $\mathcal{C}$ formally. For $i \in [q(n, 1/\epsilon)]$ and $z \in \{0, 1\}^i$, we define

$$r_{\epsilon,z}(x) = \frac{1}{q(n, 1/\epsilon)} \sum_{j \in [i],\ z_j = 1} \phi_{\epsilon, z^{j-1}}(x) \quad \text{and } R_\epsilon = \{r_{\epsilon,z}\}_{i \in [q(n,1/\epsilon)], z \in \{0,1\}^i}.$$

We remind the reader that z^{j-1} denotes the prefix of length $j-1$ of z. Let

$$R = \mathcal{H} \cup \{r_\sigma\} \cup \left(\bigcup_{k \in [n]} R_{2^{-k}} \right) ,$$

where $r_\sigma \equiv 0$ (σ denotes the empty string).

We now define functions Neigh, μ and t on R. Let $\Delta \in (0,1)$ be a real value to be defined later.

1a $r = r_\sigma$: $\mathrm{Neigh}(r,\epsilon) = \{r_\sigma, r_{\epsilon,0}, r_{\epsilon,1}\}$; $\mu(r,r,\epsilon) = \mu(r,r_{\epsilon,1},\epsilon) = \Delta$, $\mu(r,r_{\epsilon,0},\epsilon) = 1-2\Delta$; $t(r,\epsilon) = \frac{\theta_{\epsilon,\sigma}}{q(n,1/\epsilon)}$.

1b $r = r_{\epsilon,z}$ for $z \in \{0,1\}^i$ where $i \in [q(n,1/\epsilon)-1]$: $\mathrm{Neigh}(r,\epsilon) = \{r, r_{\epsilon,z0}, r_{\epsilon,z1}\}$; $\mu(r,r,\epsilon) = \mu(r,r_{\epsilon,z1},\epsilon) = \Delta$, $\mu(r,r_{\epsilon,z0},\epsilon) = 1-2\Delta$; $t(r,\epsilon) = \frac{\theta_{\epsilon,z}}{q(n,1/\epsilon)}$.

2 $r = r_{\epsilon,z}$ for $z \in \{0,1\}^{q(n,1/\epsilon)}$: $\mathrm{Neigh}(r,\epsilon) = \{r, h_{\epsilon,z}\}$; $\mu(r,r,\epsilon) = \Delta$, $\mu(r,h_{\epsilon,z},\epsilon) = 1-\Delta$; $t(r,\epsilon) = \frac{1}{q(n,1/\epsilon)}$.

3 $r = h$ for $h \in \mathcal{H}$: $\mathrm{Neigh}(r,\epsilon) = \{r\}$; $\mu(r,r,\epsilon) = 1$, $t(r,\epsilon) = \frac{1}{q(n,1/\epsilon)}$.

This evolutionary algorithm requires initialization to state r_σ. The initialization ensures that for every value of ϵ, the evolution process only goes through representations in R_ϵ before reaching a representation in $\mathcal{H}$ (in which it stays forever). Also note that the neighborhood of r_σ is defined exactly as the neighborhood of $r_{\epsilon,\sigma}$. Therefore the definition given in item 1a is essentially a partial case of the definition in 1b.

Claim 4.4 *$\mathcal{C}$ is evolvable by $N(\mathcal{A})$ over every distribution $D \in \mathcal{D}$.*

Proof: As before (and without loss of generality) we assume that ϵ is an integer power of 2. We define the bound on the number of generations $g(n,1/\epsilon) = q(n,1/\epsilon)+1$ and let $\Delta = \frac{\epsilon}{4g(n,1/\epsilon)}$ (used in the definition of $N(\mathcal{A})$). Let $s(n,1/\epsilon)$ be the size of a sample sufficient to estimate any random variable $V \in [-1,1]$ to tolerance $\tau' = \frac{\tau(n,1/\epsilon)}{2q(n,1/\epsilon)}$ with probability at least $1 - \frac{\epsilon}{6g(n,1/\epsilon)}$. Hoeffding's bound implies that $s(n,1/\epsilon) = c_0 q^2(n,1/\epsilon) \cdot \tau^{-2}(n,1/\epsilon) \cdot \log(1/\epsilon)$ for a constant c_0 will suffice [Hoe63].

Let $f \in \mathcal{C}$ be the ideal function and let $r_0 = r_\sigma, r_1, r_2, \ldots, r_g$ be a sequence of representation produced by $r_k = \mathrm{Mu}(f, D, N(\mathcal{A}), r_{k-1}, \epsilon, s(n,1/\epsilon))$. Our goal is to prove that with probability at least $1-\epsilon$, $\mathtt{Perf}_f(r_g, D) \geq 1-\epsilon$. We first note that for every representation r, the neighborhood of r contains at most three 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 $\tau' = \frac{\tau(n,1/\epsilon)}{2q(n,1/\epsilon)}$ of the true performance with probability at least $1 - \frac{\epsilon}{6p(n,1/\epsilon)}$ and therefore all of them are within τ' with probability at least $1-\epsilon/2$. For a representation r, we denote the obtained estimate by $v(r)$.

Next, assuming that all the estimates are within τ', we prove that for every z of length $i \in \{0,1,\ldots,q-1\}$, if $r_k = r_{\epsilon,z}$ then with probability at least $1 - \frac{\epsilon}{2p(n,1/\epsilon)}$, $r_{k+1} = r_{\epsilon,zb}$, where b is a valid answer to query $(\phi_{\epsilon,z}(x), \theta_{\epsilon,z}, \tau)$ from $\mathrm{CSQ}_>(f,D)$. Here $r_{\epsilon,\sigma}$ refers to r_σ.

According to the definition of $N(\mathcal{A})$, $\mathrm{Neigh}(r_{\epsilon,z},\epsilon) = \{r_{\epsilon,z}, r_{\epsilon,z0}, r_{\epsilon,z1}\}$. Representations $r_{\epsilon,z}$ and $r_{\epsilon,z0}$ compute function $\phi' = \frac{1}{q(n,1/\epsilon)} \sum_{j \leq i,\ z_j=1} \phi_{\epsilon,z^{j-1}}(x)$ and $r_{\epsilon,z1}$ computes function

$$\frac{1}{q(n,1/\epsilon)} \left(\sum_{j \leq i,\ z_j=1} \phi_{\epsilon,z^{j-1}}(x) + \phi_{\epsilon,z} \right) = \phi' + \frac{\phi_{\epsilon,z}}{q(n,1/\epsilon)}$$

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

If $r_{\epsilon,z1} \in Bene$ then $r_{k+1} = r_{\epsilon,z1}$ and $v(r_{\epsilon,z1}) - v(r_{\epsilon,z}) \geq t(r_{\epsilon,z}, \epsilon)$. But

$$v(r_{\epsilon,z1}) - v(r_{\epsilon,z}) \leq \mathtt{Perf}_f(r_{\epsilon,z1}, D) - \mathtt{Perf}_f(r_{\epsilon,z}, D) + 2\tau' = \frac{\mathtt{Perf}_f(\phi_{\epsilon,z}, D)}{q(n, 1/\epsilon)} + 2\tau' \ .$$

That is,

$$\mathbf{E}_D[f \cdot \phi_{\epsilon,z}] \geq q(n, 1/\epsilon)(t(r_{\epsilon,z}, \epsilon) - 2\tau') = \theta_{\epsilon,z} - \tau(n, 1/\epsilon) \ .$$

Therefore $b = 1$ is indeed a valid answer from $\mathrm{CSQ}_>(f, D)$ to query $(\phi_{\epsilon,z}(x), \theta_{\epsilon,z}, \tau)$.

If $r_{\epsilon,z1} \notin Bene$ then a representation from Neut will be chosen according to its relative probability. By the definition of $\mu(r_{\epsilon,z}, \epsilon)$, with probability at least $1 - 2\Delta = 1 - \frac{\epsilon}{2g(n,1/\epsilon)}$, $r_{k+1} = r_{\epsilon,z0}$. In this case $r_{\epsilon,z1} \notin Bene$ implies that $v(r_{\epsilon,z1}) - v(r_{\epsilon,z}) < t(r_{\epsilon,z}, \epsilon)$. By the same argument as in the previous case, this implies that

$$\mathbf{E}_D[f \cdot \phi_{\epsilon,z}] \leq q(n, 1/\epsilon)(t(r_{\epsilon,z}, \epsilon) + 2\tau') = \theta_{\epsilon,z} + \tau(n, 1/\epsilon) \ .$$

Therefore $b = 0$ is indeed a valid answer from $\mathrm{CSQ}_>(f, D)$ to query $(\phi_{\epsilon,z}(x), \theta_{\epsilon,z}, \tau)$.

By the properties of $\mathcal{A}'$, if z is of length $q(n, 1/\epsilon)$ and for each $i \leq q(n, 1/\epsilon)$, z_i is a valid answer to query $(\phi_{\epsilon,z}(x), \theta_{\epsilon,z}, \tau)$ then $h_{\epsilon,z}$ (the output of $\mathcal{A}'$ on responses z) satisfies $\mathtt{Perf}_f(h_{\epsilon,z}, D) = \mathbf{E}_D(f \cdot h_{\epsilon,z}) \geq 1 - \epsilon$. The neighborhood of $r_{\epsilon,z}$ is $\{r_{\epsilon,z}, h_{\epsilon,z}\}$. If $h_{\epsilon,z} \in Bene$ or $Neut = \{r_{\epsilon,z}, h_{\epsilon,z}\}$ then with probability at least $1 - \Delta = 1 - \frac{\epsilon}{4g(n,1/\epsilon)}$, $r_g = h_{\epsilon,z}$. Otherwise, $Bene = \emptyset$ and $Neut = \{r_{\epsilon,z}\}$ and therefore $r_g = r_{\epsilon,z}$. This only holds if $\mathtt{Perf}_f(r_{\epsilon,z}, D) - \mathtt{Perf}_f(h_{\epsilon,z}, D) \geq t(r_{\epsilon,z}, \epsilon) - 2\tau' > 0$. Hence $\mathtt{Perf}_f(r_{\epsilon,z}, D) > 1 - \epsilon$.

Therefore, under the assumption that all the estimates of performance are within τ', with probability at least $1 - \epsilon/2$, $\mathtt{Perf}_f(r_g, D) > 1 - \epsilon$ and hence $\mathtt{Perf}_f(r_g, D) > 1 - \epsilon$ holds with probability at least $1 - \epsilon$. □(Cl. 4.1)

To finish the proof we also need to establish that $N(\mathcal{A})$ is a valid evolutionary algorithm. The representation class R is polynomially evaluatable; the size of the neighborhood of each state is bounded by a polynomial $p_{N(\mathcal{A})} = 3$; and Neigh, μ, and t are clearly computable in polynomial (in n and $1/\epsilon$) time. We also need to bound t from above and below by two polynomially related polynomials in $1/n, \epsilon$. By the definition of t and properties of $\mathcal{A}'$, for all $r \in R$ and $\epsilon > 0$, $1/q(n, 1/\epsilon) \geq t(r, \epsilon) \geq \tau(n, 1/\epsilon)/q(n, 1/\epsilon)$. We can assume that $q(n, 1/\epsilon) \geq \max\{1/\epsilon, n\}$ ("dummy" queries can always be added to $\mathcal{A}'$ if needed) and set $\mathrm{tu}_{N(\mathcal{A})}(1/n, \epsilon) = 1/n + \epsilon$. Then since $\tau(n, 1/\epsilon)$ is lower bounded by an inverse of a polynomial in n and $1/\epsilon$ we obtain that there exists a constant $\eta \geq 1$ such that $1/n + \epsilon \geq 1/q(n, 1/\epsilon) \geq t(r, \epsilon) \geq \tau(n, 1/\epsilon)/q(n, 1/\epsilon) \geq (1/n + \epsilon)^\eta$. □(Th. 4.3)

Theorem 4.3 with the results in Section 4.2 gives Theorem 1.1. Using Theorems 3.2 and 1.1 we can obtain evolutionary algorithms from any SQ algorithm. For example,

Corollary 4.5 *(from [BFKV97]) For every p-samplable distribution D,* TH *is evolvable over D.*

Corollary 4.6 *(from [Val84, Kea98]) For every p-samplable distribution D, and constant k, k-CNF/DNF (that includes k-term DNF/CNF) is evolvable over D.*

4.2 Randomization and Initialization

In this section we augment the construction given in Theorem 1.1 to allow evolution from any state (that is without initialization) and to allow randomization in the source CSQ algorithm.

To prove that randomization can be handled we first note that random choices are inherent in the model of evolvability and therefore all that needs to be done is to translate random choices made by an evolutionary algorithm into random bits used in the computation of a CSQ algorithm. Our proof below formalizes this intuition.

Lemma 4.7 *Let $\mathcal{C}$ be a concept class CSQ learnable over a class of distributions $\mathcal{D}$ by a probabilistic polynomial-time algorithm $\mathcal{A}$. There exists an evolutionary algorithm $N(\mathcal{A}) = (R, Neigh, \mu, t)$ such that $\mathcal{C}$ is evolvable by $N(\mathcal{A})$ over $\mathcal{D}$ with initialization.*

Proof: We first need to increase the confidence of $\mathcal{A}$ from $1/2$ to $1 - \epsilon/2$. This can be done in a standard way: by running $\mathcal{A}$ $\log(2/\epsilon)$ times on independent random coin flips and testing the obtained hypotheses. Note that testing of a hypothesis can be done by a single CSQ.

Assume for simplicity that $\mathcal{A}$ flips a single random unbiased coin. For $b \in \{0,1\}$, let $\mathcal{A}^b$ denote $\mathcal{A}$ executed with the outcome of the coin flip equal to b. Let $N(\mathcal{A}^b) = (R^b, \mathrm{Neigh}^b, \mu^b, t^b)$ be the evolutionary algorithm for $\mathcal{A}^b$ constructed as in the proof of Theorem 4.3 (we refer to all the representations in R^b by using superscript b).

We now define $N(\mathcal{A}) = (R, \mathrm{Neigh}, \mu, t)$ as follows.

- $R = R^0 \cup R^1 \cup \{r_\sigma\}$, where $r_\sigma \equiv 0$.
- For $b \in \{0,1\}$ and $r \in R^b$, Neigh, μ, and t are equal to Neigh^b, μ^b and t^b.
- $\mathrm{Neigh}_N(r_\sigma, \epsilon) = \{r_\sigma, r_\sigma^0, r_\sigma^1\}$.
- $\mu(r_\sigma, r_\sigma^0, \epsilon) = \mu(r_\sigma, r_\sigma^1, \epsilon) = (1 - \Delta)/2,\ \mu(r_\sigma, r_\sigma, \epsilon) = \Delta$; $t(r_\sigma, \epsilon) = \mathrm{tu}_N(1/n, \epsilon)$.

As in the proof of Theorem 4.3, we can assume that the performance of all the states in $\mathrm{Neigh}_N(r_\sigma, \epsilon)$ is estimated with accuracy higher than $\mathrm{tu}_N(1/n, \epsilon)/2$. This implies that when r_σ is reached, $Neut = \{r_\sigma, r_\sigma^0, r_\sigma^1\}$. We choose Δ to be small enough to be absorbed by the bound on the total probability of failure $\epsilon/2$. We can therefore assume that the state r_σ is not chosen. Under this assumption, r_σ^0 and r_σ^1 will be each chosen with probability $1/2$ producing the outcome a coin flip. Starting from representation r_σ^b, $N(\mathcal{A})$ is equivalent to $N(\mathcal{A}^b)$ and therefore the result of this transformation is equivalent to running $\mathcal{A}^b$ with probability $1/2$ for each $b \in \{0,1\}$. This, by our definition, is exactly $\mathcal{A}$. Note that the number of generations required for convergence grows only by 1.

To produce any polynomial $\rho(n, 1/\epsilon)$ number of random bits the same splitting is repeated $\rho(n, 1/\epsilon)$ times. For each outcome of the coin flips $v \in \{0,1\}^\rho$ from representation r_σ^v the evolutionary algorithm executes $N(\mathcal{A}^v)$ where $\mathcal{A}^v$ is $\mathcal{A}$ with its random coin flips set to v. The number of generations required for convergence grows by $\rho(n, 1/\epsilon)$. □

We now describe how to create evolutionary algorithms that converge when started in any state. The idea of this 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_σ.

Theorem 4.8 (≡1.1) *Let $\mathcal{C}$ be a concept class CSQ learnable over a class of distributions $\mathcal{D}$ by a probabilistic polynomial-time algorithm $\mathcal{A}$. There exists an evolutionary algorithm $N(\mathcal{A}) = (R, Neigh, \mu, t)$ such that $\mathcal{C}$ is evolvable by $N(\mathcal{A})$ over $\mathcal{D}$ with initialization.*

Proof: Let $N'(\mathcal{A})$ be the evolutionary algorithm obtained in the proof of Theorem 4.7. We will modify $N(\mathcal{A}')$ to produce $N(\mathcal{A})$ that does not require initialization. First note that, if $N'(\mathcal{A})$ is started in any coin-flipping representation r_σ^w for a vector of partial results of coin flipping w (of length at most $\rho(n, 1/\epsilon)$) or representation $r_{\epsilon,z}^v \in N(\mathcal{A}^v)$ some $v \in \{0,1\}^\rho$ and a vector z of partial replies to $\mathrm{CSQ}_>$ queries, then with probability at least $1 - \epsilon$, after at most $\rho(n, 1/\epsilon) + q(n, 1/\epsilon)$ steps, the algorithm will reach either state $r_{\epsilon,z'}^v$ for some $v \in \{0,1\}^\rho$ and $z' \in \{0,1\}^q$ or $h^v \in \mathcal{H}^v$ for some $v \in \{0,1\}^\rho$. This is true since $N'(\mathcal{A})$ stays in the same representation only with small probability (that we denote by Δ). Namely, as follows easily from our prior analysis, with probability at least $1 - \Delta$, in every step of $N'(\mathcal{A})$, either the result

of a coin flip or an answer to a $\mathrm{CSQ}_>$ query is recorded in the current representation. Therefore it is sufficient to prove convergence from representations in the set

$$\left\{r^v_{\epsilon,z'} \mid v \in \{0,1\}^\rho,\ z' \in \{0,1\}^q\right\} \bigcup \left\{h^v \in \mathcal{H}^v \mid v \in \{0,1\}^\rho\right\} \ .$$

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

To check this condition we add a representation computing function $(1 - \frac{\mathrm{tu}_N}{1-\epsilon})h^v$ to the neighborhood of h^v with threshold $t = \mathrm{tu}_N$. If

$$\mathtt{Perf}_f(h^v, D) - \mathtt{Perf}_f\left(\left(1 - \frac{\mathrm{tu}_N}{1-\epsilon}\right)h^v, D\right) = \frac{\mathtt{Perf}_f(h^v, D)}{1-\epsilon}\mathrm{tu}_N \geq \mathrm{tu}_N$$

then $\mathtt{Perf}_f(h^v, D) \geq 1-\epsilon$. Therefore, if this representation is not deleterious then $\mathtt{Perf}_f(h^v, D) < 1-\epsilon$ and the algorithm will evolve into this representation for an appropriate choice of μ. Otherwise (if $\mathtt{Perf}_f(h^v, D) \geq 1-\epsilon$), the algorithm will remain in representation h^v. 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 $\mathcal{A}$ 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 $\epsilon \cdot \mathrm{tu}_N/8$ will be sufficient. We omit the straightforward details of this and other analogous modifications to simplify the presentation.

Formally, for $h^v \in \mathcal{H}^v$, we denote the representation computing $(1 - \frac{\mathrm{tu}_N}{1-\epsilon})h^v$ by $h^v_{\epsilon,0}$ and define

- $\mathrm{Neigh}(h^v, \epsilon) = \{h^v, h^v_{\epsilon,0}\}$;
- $\mu(h^v, h^v, \epsilon) = \Delta$, $\mu(h^v, h^v_{\epsilon,0}, \epsilon) = 1-\Delta$;
- $t(h^v, \epsilon) = \mathrm{tu}_N(1/n, \epsilon)$.

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 at most tu_N and therefore each of the mutations in the sequence will be always neutral. For every integer $i \leq 1/\mathrm{tu}_N - 1$, we add a representation $h^v_{\epsilon,i}$. The function computed by $h^v_{\epsilon,i}$ is $(1 - i \cdot \mathrm{tu}_N)h^v_{\epsilon,0}$. For $i \in \{0, 1, \ldots, \lceil 1/\mathrm{tu}_N \rceil - 1\}$ we define:

- $\mathrm{Neigh}(h^v_{\epsilon,i}) = \{h^v_{\epsilon,i}, h^v_{\epsilon,i+1}\}$; when $i = \lceil 1/\mathrm{tu}_N \rceil - 1$, $h^v_{\epsilon,\lceil 1/\mathrm{tu}_N \rceil}$ refers to r_σ.
- $\mu(h^v_{\epsilon,i}, h^v_{\epsilon,i}, \epsilon) = \Delta$, $\mu(h^v_{\epsilon,i}, h^v_{\epsilon,i+1}, \epsilon) = 1-\Delta$;
- $t(h^v_{\epsilon,i}, \epsilon) = \mathrm{tu}_N(1/n, \epsilon)$.

With this definition, for every $i \leq \lceil 1/\mathrm{tu}_N \rceil - 2$,

$$|\mathtt{Perf}_f(h^v_{\epsilon,i}, D) - \mathtt{Perf}_f(h^v_{\epsilon,i+1}, D)| = |\mathtt{Perf}_f(h^v_{\epsilon,0}, D) \cdot \mathrm{tu}_N| \leq \mathrm{tu}_N \ ,$$

and

$$|\mathtt{Perf}_f(h^v_{\epsilon,\lceil 1/\mathrm{tu}_N \rceil - 1}, D) - \mathtt{Perf}_f(r_\sigma, D)| \leq |\mathtt{Perf}_f(h^v_{\epsilon,0}, D) \cdot \mathrm{tu}_N| \leq \mathrm{tu}_N \ .$$

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

Testing and "re-initialization" at a representation $r^v_{\epsilon,z'}$ for $v \in \{0,1\}^\rho$ and $z' \in \{0,1\}^q$ can be done in exactly the same way. There the "re-initialization" sequence will be taken only if both

the performance of $r^v_{\epsilon,z'}$ is lower than $1-\epsilon$ and $h^v_{\epsilon,z'}$ is not a beneficial mutation. This can only happen when the performance of $h^v_{\epsilon,z'}$ is less than $1-\epsilon$, and in particular, the "re-initialization" is required.

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

An immediate corollary of Theorems 4.1 and 4.8 is that evolvability with initialization is equivalent to evolvability without initialization. More generally, we can conclude that if a certain property is possessed by our construction then evolvability with this property is equivalent to general evolvability. For example we can conclude that evolvability with neighborhood sizes bounded by 3 is equivalent to evolvability with neighborhood sizes bounded by a polynomial.

5 Limitations on Evolvability

In this section we will show an information-theoretic lower bound on learning by correlational statistical queries and use it to prove that decision lists and linear thresholds are not evolvable. Previous lower bounds given by Valiant [Val07] were "inherited" from learning in the more powerful SQ model in which both decision lists and linear thresholds are learnable [BFKV97, Kea98]. Our technique is based on the ideas behind the statistical query dimension of Blum *et al.* [BFJ$^+$94] and uses Freund's boost-by-majority algorithm [Fre95] to produce a simple structural characterization of weak learnability by CSQs.

Statistical query dimension (SQ-dim) characterization of Blum *et al.* [BFJ$^+$94] states that if a concept class $\mathcal{C}$ can be weakly SQ learned with respect to D then the number of mutually "almost" uncorrelated functions in $\mathcal{C}$ is at most polynomial (with correlations measured over D). It is well-known that this also implies that there exist a polynomial-size set of functions W_D, such that every function in $\mathcal{C}$ is correlated with some $h \in W$ (such W_D can be obtained by taking a maximum set of "almost" uncorrelated functions in $\mathcal{C}$). Here we prove that if, $\mathcal{C}$ is learnable by CSQs alone then there exists a single polynomial-size set W such that for every distribution D and every $f \in \mathcal{C}$ there exists $h \in W$ that is correlated with f. By well-known results of Freund [Fre95] and Goldmann *et al.* [GHR92], this property of $\mathcal{C}$ implies that every function in $\mathcal{C}$ can be represented as a low-weight threshold of functions in W. However, according to a recent observation of Sherstov [She07], communication complexity lower bounds of Goldmann *et al.* [GHR92] and Buhrman *et al.* [BVdW07] imply that such representations do not exist for decision lists and linear threshold functions. Therefore together with our characterization these lower bounds imply lower bounds for evolvability.

5.1 Characterization

We say that a set of functions W γ-correlates with a concept class $\mathcal{C}$ over a class of distributions $\mathcal{D}$ if for every $f \in \mathcal{C}$ and $D \in \mathcal{D}$, there exists $\phi \in W$ such that $|\mathbf{E}_D[f(x)\phi(x)]| \geq \gamma$.

Definition 5.1 *We say that a concept class $\mathcal{C}$ has* correlational statistical query dimension *bounded by $d(n)$ over a class of distributions $\mathcal{D}$ if for every n, there exists a set W_n of size at most $d(n)$ that $\frac{1}{d(n)}$-correlates with $\mathcal{C}$ over $\mathcal{D}$. We denote this by CSQ-dim$(\mathcal{C},\mathcal{D}) \leq d(n)$. We denote by CSQ-dim$(\mathcal{C})$ the correlational statistical query dimension of $\mathcal{C}$ over the set of all distributions.*

We will now prove that weak learning by correlational statistical queries is information-theoretically characterized by CSQ-dim. We start with the easier direction.

Theorem 5.2 *Let $\mathcal{C}$ be a representation class and $\mathcal{D}$ be a class of distributions. If CSQ-dim$(\mathcal{C},\mathcal{D}) = d(n)$ then $\mathcal{C}$ is weakly learnable over $\mathcal{D}$ by a CSQ algorithm (not necessarily*

efficient) that uses $d(n)$ correlational statistical queries of tolerance $1/(3d(n))$ and produces $1/(3d(n))$-correlated hypotheses.

Proof: Let W_n be a set of at most $d(n)$ functions that $\frac{1}{d(n)}$-correlates with $\mathcal{C}$ over $\mathcal{D}$. For every n, the algorithm has W_n "hard-wired". It asks a query $(\phi, \frac{1}{3d(n)})$ for every $\phi \in W_n$. Denote by $v(\phi)$ the answer from the oracle to query $(\phi, \frac{1}{3d(n)})$. If for a certain $\phi' \in W_n$, $|v(\phi')| \geq \frac{2}{3d(n)}$ then the algorithm returns the hypothesis $h(x) \equiv \mathtt{sign}(v(\phi'))\phi'(x)$.

By the definition of W_n, for every $f \in \mathcal{C}$ and $D \in \mathcal{D}$, there exists $\phi_{f,D} \in W_n$ such that $|\mathbf{E}_D[f \cdot \phi_{f,D}]| \geq \frac{1}{d(n)}$. The queries have accuracy $\frac{1}{3d(n)}$ and therefore $|v(\phi_{f,D})| \geq \frac{2}{3d(n)}$. On the other hand, for every ϕ', if $|v(\phi')| \geq \frac{2}{3d(n)}$ then $\mathbf{E}_D[f \cdot \mathtt{sign}(v(\phi'))\phi'] \geq \frac{1}{3d(n)}$. Hence the algorithm will always output a hypothesis that is $\frac{1}{3d(n)}$-correlated with f over D. □

We will now prove the other direction of the characterization for deterministic CSQ algorithms.

Theorem 5.3 *Let $\mathcal{C}$ be a representation class and $\mathcal{D}$ be a class of distributions. If $\mathcal{C}$ is weakly learnable over $\mathcal{D}$ by a (deterministic) CSQ algorithm that uses $p(n)$ correlational statistical queries of tolerance $\tau(n)$ and produces $\gamma(n)$-correlated hypotheses then CSQ-dim$(\mathcal{C}, \mathcal{D}) \leq \max\{p(n) + 1, 1/\tau(n), 1/\gamma(n)\}$.*

Proof: Let $\mathcal{A}$ be the CSQ algorithm satisfying the conditions of the theorem. The set W_n is constructed as follows. Simulate algorithm $\mathcal{A}$ and for every query (ϕ_i, τ) add ϕ_i to W_n and respond with 0 to the query. Continue the simulation until it violates any of the complexity bounds of $\mathcal{A}$ or $\mathcal{A}$ stops and returns hypothesis h. In the latter case, add h to W_n.

First, by the definition of W_n, $|W_n| \leq p(n) + 1 \leq d(n)$. Now, assume that W_n does not $\frac{1}{d(n)}$-correlate with $\mathcal{C}$ over $\mathcal{D}$, that is there exists $f \in \mathcal{C}$ and $D \in \mathcal{D}$ such that for every $\phi \in W_n$, $|\mathbf{E}_D[f \cdot \phi]| < \frac{1}{d(n)} \leq \tau(n)$. This means that in our simulation, zeros are valid answers to the queries. Therefore, by the definition of $\mathcal{A}$, it returns hypothesis h such that $\mathbf{E}_D[fh] \geq \gamma(n) \geq \frac{1}{d(n)}$. But $h \in W_n$ and therefore this contradicts the assumption. □

Note that Theorem 3.3 implies that CSQ-dim$(\mathcal{C}, \{D\})$ is polynomially related to SQ-dim$(\mathcal{C}, D)$.

Our characterization produces a surprisingly simple way to describe all concept classes weakly learnable by CSQs over all distributions. Let S be a set of Boolean functions. Denote by $\mathtt{TH}(k, S)$ the set of all functions representable as $\mathtt{sign}(\sum_{i \leq k} \phi_i(x))$ where for all i, $\phi_i \in S$.

Theorem 5.4 *A representation class $\mathcal{C}$ is weakly CSQ learnable over all distributions if and only if there exist polynomials $p(n)$ and $q(n)$ such that $\mathcal{C}_n \subseteq \mathtt{TH}(q(n), S_n)$, where $|S_n| \leq p(n)$ and S_n can be generated by a polynomial-time algorithm G. Here G is randomized if and only if the learning algorithm is randomized.*

Proof: For the simple direction we use the well-known result of Hajnal *et al.* [HMP$^+$93] that states that if a function $f \in \mathtt{TH}(q(n), S_n)$ then for every distribution D, there exists a function $\phi \in S_n$ such that $|\mathbf{E}_D[f \cdot \phi]| \geq 1/q(n)$. Therefore, as in the proof of Theorem 5.2, S_n gives a weak CSQ learning algorithm for $\mathcal{C}$. The additional condition that S_n can be generated efficiently implies that the weak learning CSQ algorithm is efficient.

For the other direction, let $\mathcal{A}$ be a weak CSQ learning algorithm for $\mathcal{C}$. We first deal with the deterministic case. Theorem 5.3 implies that there exist a polynomial $d(n)$ and a set W_n of size at most $d(n)$ that $\frac{1}{d(n)}$-correlates with $\mathcal{C}$ for every distribution D. Given such set of universal weak hypotheses one can use Freund's boost-by-majority algorithm [Fre95] with functions from W_n used as weak hypotheses. When applied on uniform distribution over $\{0,1\}^n$ and $\epsilon = 2^{-n}$ the boosting algorithm will generate a hypothesis equal to f and represented as a majority (for $\{0,1\}$ valued functions) of $\ell(n) = c_0 \cdot n \cdot d^2(n)$ functions from W_n (for a constant c_0). In other words, the boosting algorithm implies that $f \in \mathtt{TH}(\ell(n), W_n)$. A simpler proof of this result was

also given by Goldmann *et al.* [GHR92]. It is easy to see from the proof of Theorem 5.3 that W_n is indeed efficiently constructible.

If $\mathcal{A}$ is a randomized algorithm we denote by $W_{n,z}$ the set of queries and a hypothesis obtained by running $\mathcal{A}$ with its random coins set to $z \in \{0,1\}^{r(n)}$ as described in the proof of Theorem 5.3. Here $r(n)$ is a polynomial upper bound on the number of random bits used by $\mathcal{A}$. Let $Q = \bigcup_{z \in \{0,1\}^{r(n)}} W_{n,z}$. The algorithms $\mathcal{A}$ succeeds with probability at least $1/2$ over the choice of z and therefore, for fixed $f \in \mathcal{C}$ and distribution D, with probability at least $1/2$, there exists $\phi \in W_{n,z}$ such that $|\mathbf{E}_D[f \cdot \phi]| \geq 1/d(n)$. For each function f, each distribution generated by Freund's boosting algorithm is uniquely described by f and the weak hypotheses obtained by the boosting algorithm so far. This implies that there are at most $|\mathcal{C}||Q|^{\ell(n)} \leq |\mathcal{C}|(d(n) \cdot 2^{r(n)})^{\ell(n)}$ different distributions that might be generated by Freund's boosting algorithm when it uses $\mathcal{A}$ as a weak learner. Now let Z be a set of $t(n) = \log\left(2|\mathcal{C}|^2|Q|^{\ell(n)}\right) \leq \ell(n)(r(n)+\log d(n))+2\log(|\mathcal{C}|)+1$ randomly and independently chosen settings of the coin flips and let $W_Z = \bigcup_{z \in Z} W_{n,z}$. With probability at least $1/2$, W_Z will contain a weak hypothesis for every $f \in \mathcal{C}$ and every distribution generated by the boosting algorithm for f. Therefore with probability at least $1/2$, $\mathcal{C}_n \subseteq \mathtt{TH}(\ell(n), W_Z)$. To finish the proof we need to note that $t(n)$ is polynomial in n and hence W_Z can be generated efficiently. □

Theorems 4.3 and 4.1 imply the following characterization of weak distribution-independent evolvability.

Corollary 5.5 *A representation class $\mathcal{C}_n$ is weakly evolvable if and only if there exist polynomials $p(n)$ and $q(n)$ such that $\mathcal{C}_n \subseteq \mathtt{TH}(q(n), S_n)$, where $|S_n| \leq p(n)$ and S_n can be generated by a polynomial-time randomized algorithm.*

5.2 Lower Bounds

Our goal now is to discover which concept classes allow representation as low-weight threshold functions over a polynomial-size basis. Sherstov has recently showed that communication complexity lower bounds of Goldmann *et al.* imply that linear thresholds cannot be represented as signs of low-weight thresholds of a small number of functions [GHR92, She07]. Sherstov's argument is based on an analysis of margin complexity of $\mathtt{GHR}(x, y)$ a specific function given by Goldmann *et al.* [GHR92]. Here we give a simple and direct argument that relates the size and weights of thresholds that allow representation of $\mathcal{C}$ to the relevant communication complexity measure. We assume familiarity of the reader with the basic probabilistic two-party communication protocols (*cf.* [KN97]) and start with several related definitions.

For a function $h : \{0,1\}^m \times \{0,1\}^n \to \{-1,1\}$ we say that a randomized protocol P computes h with advantage δ if for every $(x, y) \in \{0,1\}^m \times \{0,1\}^n$, P outputs $h(x, y)$ with probability at least $1/2+\delta$ (over the coin flips of P). Let $c(P)$ denote the worst case communication complexity of protocol P (that is the maximum number of bits exchanged by the two parties). We refer to the value $c(P) + \log 1/\delta$ as the *cost* of P.

Definition 5.6 *For a function $h : \{0,1\}^n \times \{0,1\}^m \to \{-1,1\}$ let $PP(h)$ denote the cost $c(P) + \log 1/\delta$ of the minimum-cost protocol P that computes h with positive advantage δ.*

We say that a Boolean function h over $\{0,1\}^m \times \{0,1\}^n$ is *realized* by a set of functions F if for every specific setting of variables in $\{0,1\}^m$ the function obtained over $\{0,1\}^n$ is in F. That is, for $z \in \{0,1\}^m$ let $h_z(y) \equiv h(z, y)$ then h is realized by F if

$$\{h_z\}_{z \in \{0,1\}^m} \subseteq F.$$

Our main lemma gives a protocol for functions that can be realized by thresholds over some basis functions. This protocol is implicit in the work of Sherstov [She07] and is a variant of the standard protocol for computing low-weight thresholds (*cf.* [GHR92]).

Lemma 5.7 *If $h : \{0,1\}^m \times \{0,1\}^n \to \{-1,1\}$ is realized by $\mathtt{TH}(d,S)$ for some set of functions S, then $PP(h) \leq \log d + \lceil \log |S| \rceil + 2$.*

Proof: Let $s = |S|$ and $f_1, f_2, \ldots, f_s$ be the elements of S. Let P be the following protocol for h. Given x, Alice knows $h_x : \{0,1\}^m \to \{-1,1\}$ the restriction of h to specific values of her input bits. By the conditions of the lemma, there exist integers $a_1, a_2, \ldots, a_s$ such that, $w = \sum_{i \in [s]} |a_i| \leq d$ and for every y, $h_x(y) = \mathtt{sign}(\sum_{i \in [s]} a_i f_i(y))$. Therefore for each i, Alice chooses, i with probability $|a_i|/w$. She then sends the outcome j to Bob along with $b = \mathtt{sign}(a_j)$. The value computed by Bob (and the protocol) is $b \cdot f_j(y)$. For every x and y, $h_x(y) = \mathtt{sign}(\sum_{i \in [s]} a_i f_i(y))$ and therefore with probability at least $1/2 + 1/(2w) \geq 1/2 + 1/(2d)$ over the choice of j, $h_x(y) = \mathtt{sign}(a_j) \cdot f_j(y)$. This implies that the cost of P is $\lceil \log s \rceil + 1 + \log 2d$, and in particular, $\mathrm{PP}(h) \leq \log d + \lceil \log |S| \rceil + 2$. □

Goldmann *et al.* give a function $\mathtt{GHR}(x,y) : \{0,1\}^{4m^2} \times \{0,1\}^{2m} \to \{-1,1\}$ that is realized by halfspaces over $\{0,1\}^{2m}$ and satisfies $\mathrm{PP}(\mathtt{GHR}) \geq (m - \log m)/2 - O(1)$ [GHR92] (they only give this lower bound for one-way protocols but this is sufficient for our purposes and was later extended to two-way protocols). Lemma 5.7 implies that halfspaces cannot be represented as low-weight thresholds over a small basis since that would contradict the lower bound of Goldmann *et al.* [GHR92]. Specifically,

Lemma 5.8 ([She07]) *If $\mathtt{TH}_n \subseteq \mathtt{TH}(d,S)$ then $d|S| = \Omega(2^{n/4}/\sqrt{n})$.*

To obtain a similar result for decision lists we use the lower bound of Buhrman *et al.* [BVdW07]. They show that $\mathrm{PP}(\mathtt{ODDMAXBIT}) = \Omega(n^{1/3})$, where $\mathtt{ODDMAXBIT}(x,y)$ is the function that equals 1 if and only if the smallest index i such that $x_i = y_i = 1$ is odd. It is not hard to verify that $\mathtt{ODDMAXBIT}$ is realized by decision lists and therefore we obtain the following lemma.

Lemma 5.9 *If $\mathtt{DL}_n \subseteq \mathtt{TH}(d,S)$ then $d|S| = 2^{\Omega(n^{1/3})}$.*

Proof: We first verify that for every x, $\mathtt{ODDMAXBIT}_x(y)$ can be computed by a decision list. Let $i_1 < i_2 < \cdots < i_k$ be the indices of all the positions where x equals 1 and let $b_j = 1$ if i_j is odd and -1 otherwise. Then by the definition, $\mathtt{ODDMAXBIT}_x(y)$ is computed by the decision list $(y_{i_1}, b_1), (y_{i_2}, b_2), \ldots, (y_{i_k}, b_k), -1)$. That is $\mathtt{ODDMAXBIT}$ is realized by $\mathtt{DL}_n$. Therefore if for some set of functions S and integer d, $\mathtt{DL}_n \subseteq \mathtt{TH}(d,S)$ then $\mathtt{ODDMAXBIT}$ is realized by $\mathtt{TH}(d,S)$. This, by Lemma 5.7, implies that $\log d + \lceil \log |S| \rceil + 2 \geq \mathrm{PP}(\mathtt{ODDMAXBIT}) = \Omega(n^{1/3})$, proving our claim. □

Combined with Corollary 5.5, Lemmas 5.8 and 5.9 imply Theorem 1.2. More precisely, Lemmas 5.8 and 5.9 imply a tradeoff between the number of queries used by a CSQ algorithm for decision lists or linear threshold functions and the advantage it achieves over the random guessing.

6 Conclusions

Our results show that Valiant's evolvability model admits another natural theoretical characterization. Our characterization gives new insights into the model and simplifies the analysis of its power. In particular, it allows us to automatically obtain numerous new algorithms that satisfy the onerous requirements of the evolvability framework. While the generated algorithms are not necessarily the most natural and efficient, the essential parts of the transformation can be used to obtain simpler algorithms for specific concept classes and distributions. We will elaborate on this elsewhere. Using our equivalence we also obtained a characterization of weak distribution-independent evolvability and applied it to concept classes of decision lists and halfspaces.

Our positive results only give distribution-specific evolutionary algorithms and the negative results characterize weak distribution-independent evolvability. This leaves out perhaps the most interesting case of *strong* distribution-independent evolvability. For example, we do not know whether disjunctions/conjunctions are evolvable distribution-independently.

More generally, we believe that modeling of evolution as learning is a promising direction of research and hope that our findings will encourage further work.

Acknowledgments

We thank Leslie Valiant and David Woodruff for valuable discussions and comments on this research. We are grateful to Sasha Sherstov for pointing out the results of Buhrman *et al.* to us.

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.

[BDCGL92] S. Ben-David, B. Chor, O. Goldreich, and M. Luby. On the theory of average case complexity. *Journal of Computer and System Sciences*, 44(2):193–219, 1992.

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

[BFJ+94] A. Blum, M. Furst, J. Jackson, M. Kearns, Y. Mansour, and S. Rudich. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *Proceedings of STOC*, pages 253–262, 1994.

[BFKV97] A. Blum, A. Frieze, R. Kannan, and S. Vempala. A polynomial time algorithm for learning noisy linear threshold functions. *Algorithmica*, 22(1/2):35–52, 1997.

[BKW03] A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM*, 50(4):506–519, 2003.

[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.

[Byl94] T. Bylander. Learning linear threshold functions in the presence of classification noise. In *Proceedings of COLT*, pages 340–347, 1994.

[Che52] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Ann. Math. Statist.*, 23:493–507, 1952.

[CW77] J. L. Carter and M. N. Wegman. Universal classes of hash functions (extended abstract). In *Proceedings of STOC*, pages 106–112, 1977.

[DV04] J. Dunagan and S. Vempala. A simple polynomial-time rescaling algorithm for solving linear programs. In *Proceedings of STOC*, pages 315–320, 2004.

[Fel07] V. Feldman. Attribute Efficient and Non-adaptive Learning of Parities and DNF Expressions. *Journal of Machine Learning Research*, (8):1431–1460, 2007.

[Fre95] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

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

[HMP+93] A. Hajnal, W. Maass, P. Pudlak, M. Szegedy, and G. Turan. Threshold circuits of bounded depth. *Journal of Computer and System Sciences*, 46:129–154, 1993.

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

[JSS97] J. Jackson, E. Shamir, and C. Shwartzman. Learning with queries corrupted by classification noise. In *Proceedings of the Fifth Israel Symposium on the Theory of Computing Systems*, pages 45–53, 1997.

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

[KN97] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[KS06] A. Klivans and A. Sherstov. Improved lower bounds for learning intersections of halfspaces. In *Proc. 19^{th} Conference on Learning Theory (COLT)*, 2006.

[KV94] M. Kearns and U. Vazirani. *An introduction to computational learning theory*. MIT Press, Cambridge, MA, 1994.

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

[Riv87] R. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.

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

[Vad04] S. Vadhan. Lecture notes on pseudorandomness. Available at `http://www.courses.fas.harvard.edu/~cs225/`, 2004.

[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.

[Val07] L. G. Valiant. Evolvability. In *Proceedings of 32nd International Symposium on Mathematical Foundations of Computer Science*, pages 22–43, 2007.

[Yan05] Ke Yang. New lower bounds for statistical query learning. *Journal of Computer and System Sciences*, 70(4):485–509, 2005.

A Equivalence of Boolean and Real-Valued Hypotheses for Evolvability

In this section we argue that evolvability restricted to representations computing Boolean functions is not significantly different from evolvability by representations computing real-valued functions in the range $[-1, 1]$ (or equivalently, randomized Boolean functions). We first show that this is true when evolving with respect to unconcentrated distributions. Our proof is based on the use of k-wise independent families of hash functions [CW77] that we now define.

Definition A.1 *For integer k, ℓ, and n, a family of functions $\mathcal{G} : \{\pi : \{0,1\}^n \to \{0,1\}^\ell\}$ is k-wise independent if for all distinct $z_1, z_2, \ldots, z_k$, the random variables $\pi(z_1), \pi(z_2), \ldots, \pi(z_k)$ are independent and uniformly distributed in $\{0,1\}^\ell$ when π is chosen randomly from $\mathcal{G}$.*

It is well-known that there exist small families of k-wise independent hash functions (*cf.* [Vad04]).

Fact A.2 *For all integer k, ℓ, and n, there exists a family of k-wise independent hash functions $\mathcal{G} : \{\pi : \{0,1\}^n \to \{0,1\}^\ell\}$ such that choosing a random function from $\mathcal{G}$ requires $k \cdot \max\{n, \ell\}$ random bits. Furthermore, choosing and evaluating each $\pi \in \mathcal{G}$ can be done in time polynomial in k,ℓ, and n.*

Theorem A.3 *Let $\mathcal{C}$ be a concept class evolvable by $N = (R, Neigh, \mu, t)$ over a set of distributions $\mathcal{D}$. There exists an evolutionary algorithm $N' = (R', Neigh', \mu', t')$ and a polynomial $w(n, 1/\epsilon)$ such that: (i) R' contains only Boolean functions, and (ii) if for every point x and distribution $D \in \mathcal{D}$, $D(x) \leq 1/w(n, 1/\epsilon)$ then $\mathcal{C}$ is evolvable by N' over $\mathcal{D}$ to accuracy ϵ.*

Proof: Without loss of generality we can assume that N is an evolutionary algorithm produced by the construction presented in Section 4. Let $g(n, 2/\epsilon)$ be the number of generations sufficient to evolve $\mathcal{C}$ over $\mathcal{D}$ by N to accuracy and tolerance $\epsilon/2$ and let $\tau' > 0$ be the tolerance in estimations of performance sufficient for the correct execution of N (see proof of Cl. 4.1 for more details). For every representation $r \in R$ computing a function $\phi(x)$, we create a Boolean function $\psi(x)$ that behaves like $\Phi(x)$, the random Boolean variable with expectation $\phi(x)$ for every x. That is, $\psi(x)$ allows to estimate the performance of $\Phi(x)$ which, by definition, equals the performance of $\phi(x)$.

Let $s(n, 1/\epsilon)$ be the sample of size sufficient to estimate a random $\{-1, 1\}$-valued variable within τ' with probability $1 - \delta$. Chernoff's and Hoeffding's bounds imply that $s(n, 1/\epsilon) = c_0 \tau'^{-2} \cdot \log(1/\delta)$ for a constant c_0 will suffice [Che52, Hoe63]. We set δ to be small enough to ensure that all $p_N(n, 2/\epsilon) \cdot g(n, 2/\epsilon) = 3 \cdot g(n, 2/\epsilon)$ evaluations of performance are successful with probability at least $1 - \epsilon/4$.

The probability that a point x appears twice in a sample of size s is at most $\frac{s(s-1)}{2} D^2(x)$. Let $w(n, 1/\epsilon) = 6 \cdot s^2(n, 1/\epsilon) \cdot g(n, 2/\epsilon)/\epsilon$. If for all x, $D(x) \leq 1/w(n, 1/\epsilon)$ then the probability that there exists a point that appears twice in a sample of size $s(n, 1/\epsilon)$ is at most

$$\sum_{x \in \{0,1\}^n} \frac{s(s-1)}{2} D^2(x) \leq \sum_{x \in \{0,1\}^n} \frac{s(s-1)}{2w} D(x) \leq \frac{\epsilon}{12 \cdot g} \sum_{x \in \{0,1\}^n} D(x) = \frac{\epsilon}{12 \cdot g} .$$

This condition implies that, with probability at least $1 - \epsilon/4$, no point will occur twice in any of at most $3 \cdot g(n, 2/\epsilon)$ samples of size s generated during all empirical evaluations of performance.

For $r \in R$, let $\phi_r(x)$ denote the (possibly real-valued) function computed by r and let $\Phi_r(x)$ be the equivalent randomized Boolean function. To simulate $\Phi_r(x)$ one needs to produce 1 with probability $(1 + \phi_r(x))/2$, and -1 otherwise. Given ℓ random coin flips this can be done approximately by comparing the number whose binary representation is a random $z \in \{0, 1\}^\ell$ to $2^\ell(1 + \phi_r(x))/2$ and outputting 1 if the number is smaller or equal to $2^\ell(1 + \phi_r(x))/2$. We denote the function thus defined by $\texttt{Compare}(\phi_r(x), z)$. This method is not precise but the statistical difference between $\Phi_r(x)$ and $\texttt{Compare}(\phi_r(x), z)$ for a randomly chosen $z \in \{0, 1\}^\ell$ is at most $2^{-\ell}$. For large enough ℓ (e.g. $\omega(\log \frac{n}{\epsilon})$) this difference will not be detectable by samples of polynomial size.

Now let $\mathcal{G}$ be an s-wise independent family of hash functions from $\{0, 1\}^n$ to $\{0, 1\}^\ell$. Let π be a randomly chosen function from $\mathcal{G}$ and let $\Psi_r(x) = \texttt{Compare}(\phi_r(x), \pi(x))$. The properties of $\mathcal{G}$ imply that evaluation of $\texttt{Perf}_f(\Psi_r(x), D, s)$ on s distinct points is equivalent to evaluation of the performance of $\texttt{Compare}(\phi_r(x), z)$ with randomly and uniformly chosen z's and hence is within $2^{-\ell}$ of the evaluation of $\Phi_r(x)$ on the same points. By the choice of $s(n, 1/\epsilon)$ and properties of D, with probability at least $1 - \epsilon/2$, $|\texttt{Perf}_f(\Psi_r(x), D, s) - \texttt{Perf}_f(\phi_r(x), D)| \leq \tau'$ as required for the analysis of N.

Therefore N' can obtained by replacing $\phi_r(x)$ with $\Psi_r(x)$ for every $r \in R$. To obtain the $s \cdot \max\{n, \ell\}$ random bits required to choose a random and uniform π we use the same coin-flipping stage as in the proof of Theorem 1.1. A minor obstacle to this approach arises from the use of functions that compute 0 in the representations that generate the random coins. This function is itself not Boolean. It is easy to see that any fixed Boolean function with performance equal to zero (or negligible) for every $f \in \mathcal{C}$ and $D \in \mathcal{D}$ would work. Such functions are usually easy to find for a specific $\mathcal{C}$ evolvable over a specific $\mathcal{D}$. Alternatively, one can use representations computing identity $\mathbf{1}$ (i.e. $\equiv 1$) to generate the random coins and then, given π, evolve to r_σ

using a sequence of gradually changing representations as in re-initialization (Th. 4.8). In either case the re-initialization sequences would also need to be gradually evolving into the new initial representation instead of r_σ. □

Our reduction to Boolean functions relies on the assumption that the target distribution does not have "heavy" points. However, learning the values of the unknown function on "heavy" points is easy. One only needs to compare the correlation of the unknown function with two functions that differ on a "heavy" point. More formally, for a fixed distribution D, let W_D denote the set of points such that $D(x) \geq 1/w(n, 1/\epsilon)$ where w is as defined in the proof of Theorem A.3. This set has size at most $w(n, 1/\epsilon)$. For a p-samplable distribution D, it can be computed by a randomized algorithm in time polynomial in w (and the inverse of the desired accuracy). For a general distribution D, we can assume that such W_D is given as advice. Let $y^1, y^2, \ldots, y^w$ be some fixed ordering of the points in W_D. The goal of the evolutionary algorithm is to discover the values of the target on each of the points in W_D. For $z \in \{0,1\}^*$ of length $k \leq w$, let $r_z^{\mathbf{1}}$ denote the representation that computes the function equal 1 on all points but the set $\{y^i | i \in [k],\ z_i = 0\}$. For every representation $r_z^{\mathbf{1}}$, our algorithm has representations $r_{z1}^{\mathbf{1}}$ and $r_{z0}^{\mathbf{1}}$ in the neighborhood of $r_z^{\mathbf{1}}$ and tolerance equals $1/w$. This ensures that when the algorithm reaches $r_z^{\mathbf{1}}$ for z of length w from $r_\sigma^{\mathbf{1}}$, z contains the values of the target on all the point in W_D. From that point we can evolve $\mathcal{C}$ on the points outside of W_D using our transformation for unconcentrated distributions. The functions computed by all representation following $r_z^{\mathbf{1}}$ are fixed to be the same as $r_z^{\mathbf{1}}$ on points in W_D. The re-initialization sequences of representations also need to evolve to $r_\sigma^{\mathbf{1}}$. This argument implies the following lemma.

Lemma A.4 *Let $\mathcal{C}$ be a concept class evolvable over an ensemble of distributions $D = \{D_n\}_{n=1}^{\infty}$. There exist an evolutionary algorithm N such that $\mathcal{C}$ is evolvable by N over D and N uses only representations computing Boolean functions. Here if D is not p-samplable then N is non-uniform.*