Electronic Colloquium on Computational Complexity, Report No. 114 (2011)

Computational Bottlenecks for Evolvability

Varun Kanade*
SEAS, Harvard University
vkanade@fas.harvard.edu

August 7, 2011

Abstract

Valiant (2007) proposed a computational model for evolution and suggested that evolvability be studied in the framework of computational learning theory. Feldman (2008) showed that Valiant's evolution model is equivalent to the correlational statistical query (CSQ) learning model, which is a restricted setting of the statistical query (SQ) model. Evolvability in Valiant's model has been shown to be quite restricted; Valiant observed that the class of parities is not evolvable even under the uniform distribution. Subsequently, Feldman (2008, 2011) showed that in a distribution-independent sense, linear separators and decision-lists are not even weakly evolvable, and that the class of conjunctions is not evolvable. All these results are based on information-theoretic arguments.

In this paper, we show that computational bottlenecks for evolvability in Valiant's model exist, even when information-theoretic bottlenecks are absent. In particular, assuming that polynomial size circuits are not PAC learnable, we construct a concept class that is not efficiently evolvable; in contrast, when unbounded computation is allowed this concept class can be evolved in an information-theoretically efficient manner.

*This work is supported in part by grants NSF-CCF-04-27129 and NSF-CCF-09-64401

ISSN 1433-8092

1 Introduction

Darwin's theory of evolution proposed that complex life forms *evolved* from simpler ones. However, the actual process of evolution and also the nature of complexity that can arise is not well understood. The two central aspects of Darwin's theory are 1) creation of variation due to *mutations*, and 2) *natural selection* among the variants (survival of the fittest). The genome of an organism contains code for proteins and also encodes rules governing their regulation. An example of a function encoded in the genome could be a circuit that decides the level of enzyme activity based on environmental conditions, e.g. temperature, water content, availability of sugar, oxygen supply etc. The main question of interest is, how complex can such circuits be, given that they must have evolved through mutation and natural selection?

Valiant [Val09] proposed a computational model for evolution and suggested that the above question be studied in the framework of computational learning theory. In his evolution model, an organism is defined to be a representation of a boolean function. The process of evolution takes place in a number of generations, where the representation at each generation is a randomly chosen *fit* mutation of the representation from the previous generation. Valiant's model defines a *mutator* as an efficient randomized Turing machine that produces a list of mutations of a given representation, and selection is defined based on empirical performance of the mutations that occurred. Roughly speaking, the number of mutations produced in a given generation and the number of examples used to evaluate empirical performance correspond to the population size. In order for evolution to be plausible, population size and the number of generations should be polynomially bounded. However, the *computational* constraint in the model is the requirement that the mutator be (polynomial time) efficient. The focus of this paper is to show that this restriction affects the notion of what is *evolvable*.

In subsequent work, Feldman [Fel08, Fel09b] showed that evolvability in Valiant's model is equivalent to *correlational statistical query* (CSQ) learning [BF02], a restriction of the SQ learning model [Kea98]. Feldman [Fel09b, Fel09a] also showed that under a class of non-linear performance measures (such as ℓ_2 loss; Valiant's model only uses 0-1 loss), evolvability is equivalent to SQ learning.

Concept classes known to be evolvable in Valiant's model are quite limited. Valiant showed that monotone conjunctions are evolvable under the uniform distribution. Feldman [Fel09b] showed that the class of singletons can be evolved distribution-independently. Kanade, Valiant and Vaughan [KVV10] showed that the class of homogeneous hyperplanes is evolvable under radially symmetric and product normal distributions.

On the other hand, strong limitations on evolvability have been shown. Valiant [Val09] observed that since evolvability is a restricted form of statistical query learning, the class of parities is not evolvable even under the uniform distribution. Feldman [Fel08] showed that in a distribution-independent sense, linear separators and decision trees are not even *weakly* evolvable. Recently, Feldman[Fel11] showed that even the class of conjunctions is not evolvable distribution-independently. However, all of the above results are unconditional and use information-theoretic arguments. A natural question that arises is, is evolution constrained purely by information-theoretic bottlenecks, or do computational bottlenecks exist?

In this paper, we formalize the above question and prove that there are indeed computational limitations to evolution. Roughly speaking, the size of the population and the number of generations correspond to information-theoretic aspects of evolution and the mutation algorithm corresponds to the computational aspect. An informal statement of our main result is:

Theorem 1. *There exists a concept class C and a distribution D over $X = \{-1,1\}^n$ such*

that C is not evolvable (in computationally bounded manner) under D, unless polynomial size circuits can be PAC-learned. On the contrary, when allowed unbounded computation C can be evolved (information-theoretically) efficiently, i.e. in polynomially many generations and with a polynomially-bounded population size. Furthermore, such a concept class exists even when D is the uniform distribution *over $X = \{-1, 1\}^n$.*

To prove this theorem, we use Feldman's result showing the equivalence between evolvability and CSQ learning. A CSQ algorithm has access to an oracle, which for any function ϕ, returns the correlation of ϕ with the target concept c, $E_{x\sim D}[\phi(x)c(x)]$, up to some inverse polynomial tolerance τ. We construct a concept class C and a distribution D, such that there exists a (computationally inefficient) CSQ algorithm that learns C by making q queries of tolerance τ, where q, τ^{-1} are polynomially bounded. We also show that a computationally efficient CSQ algorithm for C would imply an algorithm for PAC-learning polynomial size circuits.

Related Work: In the PAC learning model [Val84], it is known that the main difficulty for learning is computational, as empirical risk minimization (ERM) on a small sample suffices for the purpose of learning. However, ERM is intractable for almost all concept classes of interest. In the case of *proper* learning, i.e. when the output hypothesis is required to be in C, learning even simple concept classes such as 2-term DNF formulas is known to be hard unless NP = RP. In the case of *improper* learning, under standard cryptographic assumptions various concept class are shown to be hard to PAC-learn (see for example [GGM86, KV94, Kha93]).

In the statistical query (SQ) learning model, an efficient learning algorithm must make polynomially many queries and the tolerance parameter must be at least some inverse polynomial. In this setting, Kearns [Kea98] showed that the class of parities is not learnable. Blum et al. [BFJ$^+$94] showed that DNFs and decision trees are not efficiently learnable in the SQ model. Recently, Feldman, Lee and Servedio [FLS11] have shown that low-depth monotone formulas are also not efficiently SQ learnable. As in the case of evolvability, all these lower bounds are information-theoretic. Thus, even in the SQ learning model, one might ask the question whether computational bottlenecks exist when information theoretic ones don't? Our results also imply a separation between computationally bounded SQ learning from purely statistically efficient SQ learning.

Organization: In section 2, we briefly describe various learning models - PAC learning, SQ/CSQ learning and evolution. Section 2.3 contains formal definitions of computationally bounded and unbounded evolution. Section 3 contains the proof our main result. Section 4 concludes with a brief discussion of open problems.

2 Learning Models

In this paper, we use $\{-1, 1\}^n$ to denote the boolean cube in n dimensions, and a boolean function has range $\{-1, 1\}$. If X is an instance space, a concept class C is subset of boolean functions over X.

2.1 PAC Learning

Valiant [Val84] introduced the PAC framework to formalize the study of machine learning. Let $X = \{-1, 1\}^n$, D be a distribution over X and C be a concept class over X. For a concept $c \in C$, an example oracle $\mathsf{EX}(c, D)$ when queried, outputs an example $(x, c(x))$ where x is chosen randomly according to distribution D. Below, we give a definition of distribution-specific *weak* PAC learning.

Definition 1 (Weak-PAC Learning [KV94])**.** *A concept class C is weakly PAC-learnable with respect to distribution D, if there exists a learning algorithm $\mathcal{L}$, that for every $\delta > 0$, and for every target concept $c \in C$, with access to oracle $\mathsf{EX}(c, D)$ outputs h, such that with probability at least $1 - \delta$,*

$$\mathrm{err}_D(h, c) = \Pr_{x \sim D}[c(x) \neq h(x)] \leq 1/2 - 1/q(n)$$

for some polynomial $q(n)$. Furthermore, the running time of $\mathcal{L}$ is polynomial in $n, 1/\delta$ and h can be evaluated in polynomial time.

Our main result is based on the assumption that the class of polynomial-size circuits is not weakly learnable under the uniform distribution; this statement is true if the existence of one-way functions is assumed (cf. [GGM86, ABX08]). Let $\mathsf{CKT}^n_{p(n)}$ denote the class of circuits with n inputs and of size $p(n)$. We use the following assumption:

Assumption 1. *There exists a polynomial p such that $\mathsf{CKT}^n_{p(n)}$ is not weakly PAC learnable under the uniform distribution.*

2.2 SQ and CSQ Learning

The statistical query (SQ) model of learning was introduced by Kearns [Kea98] in the context of noise-tolerant learning. An SQ algorithm has access to a $\mathsf{STAT}(c, D)$ oracle, instead of $\mathsf{EX}(c, D)$. A query to the STAT oracle is of the form (ψ, τ) where $\psi : X \times \{-1, 1\} \to [-1, 1]$ is a polynomially evaluatable function and τ is a tolerance parameter. The STAT oracle returns a value v, such that $|v - \mathbb{E}_{x \sim D}[\psi(x, c(x))]| \leq \tau$. An SQ algorithm is said to be efficient if it runs in polynomial time and makes queries with tolerance parameter τ that is at least some inverse polynomial.

The correlational statistical query (CSQ) model (see [BF02, Fel08]) is a restriction of the SQ model where the algorithm is only allowed to query the correlation of any function ϕ with the target function. Formally, a $\mathsf{CSQ}(c, D)$ oracle on receiving a query (ϕ, τ), outputs a value v such that $|v - \mathbb{E}_{x \sim D}[\phi(x)c(x)]| \leq \tau$. The function $\phi : X \to [-1, 1]$ must be polynomially evaluatable and τ must be at least some inverse polynomial.

As in the case of PAC-learning, an SQ or CSQ algorithm is required to output a hypothesis h, that with high probability satisfies $\mathrm{err}_D(h, c) \leq \epsilon$.

2.3 Evolvability

In this section, we briefly describe Valiant's evolution model [Val09] and formally define the notion of computationally bounded and unbounded evolution. For further details, the reader is referred to [Val09, Fel08, Fel09b]. Let $X = \{-1, 1\}^n$, D be a distribution over X and C be a concept class over X. Let R be a representation class of boolean functions from $X \to \{-1, 1\}$. For a target function $c \in C$, we define the performance of a boolean function r with respect to D as,

$$\mathsf{Perf}_{c,D}(r) = \mathbb{E}_{x \sim D}[c(x)r(x)]$$

An evolutionary algorithm starts with some $r_0 \in R$ and at each round i, starting from r_{i-1} determines the possible mutations of r_{i-1} and selects one of the mutations as r_i, the representation for the next round. Formally, for some polynomial $p(\cdot, \cdot)$, a p-bounded evolutionary algorithm $\mathcal{E} = (R, \mathsf{Neigh}, \mu, t, s)$ is defined by the following components:

- R is the representation class. Each $r \in R$ represents a boolean function, $|r| \leq p(n, 1/\epsilon)$ and for any $x \in X$, $r(x)$ can be evaluated in polynomial time.

- $\mathsf{Neigh}(r, \epsilon) \subseteq R$ is the (possibly randomized) neighborhood function that specifies the list of possible mutations of r. It must be the case that $r \in \mathsf{Neigh}(r, \epsilon)$ and that $|\mathsf{Neigh}(r, \epsilon)| \leq p(n, 1/\epsilon)$.
- The function $\mu(r, r', \epsilon)$ specifies for each $r \in R$, $r' \in \mathsf{Neigh}(r, \epsilon)$, the probability that r mutates into r'. For each $r' \in \mathsf{Neigh}(r, \epsilon)$, $\mu(r, r', \epsilon) \geq 1/p(n, 1/\epsilon)$.
- The function $t(r, \epsilon) \geq 1/p(n, 1/\epsilon)$ is the tolerance function and is used to classify mutations as beneficial, neutral or deleterious.
- Finally, $s(r, \epsilon) \leq p(n, 1/\epsilon)$ is the sample size, the number of examples used to evaluate the empirical performance of each r' in $\mathsf{Neigh}(r, \epsilon)$.

We say that a p-bounded evolutionary algorithm $\mathcal{E}$ is information-theoretically efficient. In addition, if the functions Neigh, μ, t and s can be evaluated in polynomial time. we say that $\mathcal{E}$ is *computationally* efficient.

We define a single round of the evolutionary process. Starting from r_{i-1}, $r_i = M(r_{i-1}, \epsilon; c, D, \mathcal{E})$ is selected randomly from $\mathsf{Neigh}(r_{i-1}, \epsilon)$ as follows:

- Generate $\mathsf{Neigh}(r_{i-1}, \epsilon)$ and draw a sample of size $s(r_{i-1}, \epsilon)$ to compute the empirical performance for each $r \in \mathsf{Neigh}(r_{i-1}, \epsilon)$; denote this by $v(r)$.
- Define $\mathsf{Bene} = \{r \mid r \in \mathsf{Neigh}(r_{i-1}, \epsilon), v(r) \geq v(r_{i-1}) + t(r_{i-1}, \epsilon)\}$ and $\mathsf{Neut} = \{r \mid r \in \mathsf{Neigh}(r_{i-1}, \epsilon), |v(r) - v(r_{i-1})| \leq t(r_{i-1}, \epsilon)\}$. Observe that $\mathsf{Neut} \neq \emptyset$, since $r_{i-1} \in \mathsf{Neigh}(r_{i-1}, \epsilon)$.
- If $\mathsf{Bene} \neq \emptyset$, choose r randomly from Bene according to relative probability $\mu(r_{i-1}, r, \epsilon)$ as r_i. Else, choose r randomly from Neut according to relative probability $\mu(r_{i-1}, r, \epsilon)$ as r_i.

Definition 2 (Evolvability [Val09])**.** *For a concept class C, distribution D, we say that C is evolvable under D using representation class R, if for some polynomial $p(\cdot, \cdot)$, there exists a p-bounded evolutionary algorithm $\mathcal{E} = (R, \mathsf{Neigh}, \mu, t, s)$ that for every $c \in C$, $r_0 \in R$, $\epsilon > 0$, for some $g = \mathrm{poly}(n, 1/\epsilon)$, with probability at least $1 - \epsilon$ outputs a sequence $r_0, r_1, \ldots, r_g$, where $r_i = M(r_{i-1}, \epsilon; c, D, \mathcal{E})$ for all i, and $\mathsf{Perf}_{c,D}(r_g) \geq 1 - \epsilon$. Furthermore,*

- *If $\mathcal{E}$ is computationally efficient, we say that C is* computationally efficiently *evolvable under D.*
- *If $\mathcal{E}$ is not computationally efficient, but is information-theoretically efficient, we say that C is evolvable under D with unbounded computation.*

Feldman[Fel08] showed that a CSQ algorithm for learning C can be converted to an evolutionary algorithm. Theorem 2 is a restatement of the main theorem of [Fel08].

Theorem 2. *[Fel08] Suppose $\mathcal{A}$ is a CSQ algorithm for learning C under distribution D, such that $\mathcal{A}$ makes q queries each of tolerance τ. Furthermore, for each query ϕ assume that $|\phi|$ is polynomially bounded, ϕ is polynomially evaluatable and that τ^{-1} is polynomially bounded. Then,*

- *There exists an* information-theoretically *efficient evolutionary algorithm $\mathcal{E}$ that evolves C under D with* unbounded computation.
- *If in addition, $\mathcal{A}$ runs in polynomial time, then $\mathcal{E}$ is also computationally-efficient. Thus $\mathcal{E}$* computationally efficiently *evolves C under D.*

3 Computational Hardness for Evolution

In this section, we prove our main result that shows that computational bottlenecks exist for evolvability, even when information-theoretic ones don't. Let $X = \{-1,1\}^n$ and let D be a fixed distribution over X. Let C be a concept class over X such that $\log|C| = \text{poly}(n)$. In the rest of this section, we assume that ϵ is fixed, and that $1/\epsilon$ is bounded by some polynomial in n.

We define notation used in the rest of this section. Let $M = (16/\epsilon^2)\log|C|$ and let $m = (n+1)M$, thus $m = O(\text{poly}(n))$. Let $s \in \{-1,1\}^m$ be a string of length m; s can be interpreted as a sample, $S = \langle (x^i, y^i)\rangle_{i=1}^M$, where $x^i \in X$ and $y^i \in \{-1,1\}$. Let ERM be the empirical risk minimizer, a Turing machine, that when given $s \in \{-1,1\}^m$ as input, constructs sample S and outputs some $h \in C$ that is consistent with S. If no such h exists, it outputs $\perp$. The ERM Turing machine is not required to run in polynomial time; however, it always halts. Assume that one such ERM machine is fixed ahead of time.

We define the notion of a string s being *useful* for a concept $c \in C$.

Definition 3. *We say that a string $s \in \{-1,1\}^m$ is* useful *for a concept $c \in C$, if s when interpreted as a sample $S = \langle (x^i, y^i)\rangle_{i=1}^M$ satisfies $y^i = c(x^i)$ for all i, and ERM with s as input, outputs a hypothesis $h \in C$ that satisfies,*

$$\text{err}_D(h,c) = \Pr_{x\sim D}[h(x) \neq c(x)] \leq \epsilon$$

Fix a concept $c \in C$. Then it is easy to see that there exist strings $s \in \{-1,1\}^m$ that are useful for c; for example, let $S = \langle (x^i, y^i)\rangle_{i=1}^M$ be a random sample drawn from $\mathsf{EX}(c,D)$. If $M = (16/\epsilon^2)\log|C|$ by the Chernoff-Hoefdding bound, with probability at least $1/2$, for every $h \in C$,

$$|\Pr_{x\sim D}[c(x) \neq h(x)] - (1/M)\sum_{i=1}^{M} \mathbb{I}(c(x^i) \neq h(x^i))| \leq \epsilon,$$

where $\mathbb{I}$ is the indicator function. This immediately implies that if h is consistent with S, then $\text{err}_D(h,c) \leq \epsilon$. Of course, since $c \in C$ there always exists at least one $h \in C$ that is consistent with S. Let S be encoded as a string $s \in \{-1,1\}^m$, then s is useful for c.

Let m be as defined above and define $n' = m + n$. Let $X' = \{-1,1\}^{n'}$. For any k, let e_i^k denote a string of length k which is -1 everywhere except in position i where it is 1. Let $(-1)^k$ be a string of length k consisting of all -1s. For any strings $x, \tilde{x}$, $x\tilde{x}$ denotes their concatenation. Recall that D is a fixed distribution over X, we define a distribution D' over X' as follows:

1. $D'(e_i^m(-1)^n) = 1/2m$ for $i = 1, \ldots m$.
2. $D'((-1)^m x) = D(x)/2$ for all $x \in X_n$.
3. $D'(x') = 0$ elsewhere.

Next, we define a concept class C' over X' as follows.

$$C' = \{(s,c) \mid c \in C,\ s \in \{-1,1\}^m \text{ such that } s \text{ is } \textit{useful} \text{ for } c\}$$

Let $c' = (s,c)$, then $c' : X' \to \{-1,1\}$ is defined as:

1. $c'(e_i^m(-1)^n) = s_i$ for $i = 1, \ldots, m$.
2. $c'((-1)^m x) = c(x)$, where $x \in X$.

3. $c'(x') = 1$ elsewhere.

We show that there exists an algorithm that learns C' under D' in the CSQ model. The algorithm is not computationally efficient, but makes only polynomially many queries, each with inverse polynomial tolerance to the CSQ oracle.

Proposition 1. *There exists an algorithm $\mathcal{A}$ that CSQ learns C' with respect to distribution D', and furthermore,*

1. *$\mathcal{A}$ makes m queries to the CSQ oracle, each with tolerance $1/3m$.*
2. *$\mathcal{A}$ need not be computationally efficient.*

Proof. Suppose $c' = (s, c)$ is the target concept in C'.

Define $\phi_i : X' \to [-1, 1]$ such that $\phi_i(e_i^m(-1)^n) = 1$ and $\phi_i(x') = 0$ otherwise. Then, $\mathbb{E}_{x\sim D'}[\phi_i(x)c'(x)] = s_i/2m$. When $\tau = 1/3m$, a CSQ (ϕ_i, τ) reveals whether $s_i = 1$ or $s_i = -1$. Thus, using m CSQs $\mathcal{A}$ recovers the string s. Now $\mathcal{A}$ simply runs ERM on s to get $h : X \to \{-1, 1\}$ such that $\mathrm{err}_D(h, c) \leq \epsilon$.

$\mathcal{A}$ defines $h' : X' \to \{-1, 1\}$ as follows: $h'(e_i^m(-1)^n) = s_i$, $h'((-1)^m x) = h(x)$ for $x \in X$. Clearly, $\mathrm{err}_{D'}(h', c') \leq \epsilon$. □

Next, we show that the existence of a polynomial time algorithm for learning C' under D' in the CSQ model, implies the existence of an efficient PAC-learning algorithm for C under D.

Proposition 2. *If there exists an efficient (polytime) algorithm $\mathcal{B}$ that CSQ learns concept class C' with respect to D', then C is PAC learnable under D.*

Proof. Suppose that $\mathcal{B}$ is an efficient algorithm for learning C' under D' in the CSQ model. We construct an algorithm that PAC-learns C under distribution D.

First, draw M examples from $\mathsf{EX}(c, D)$ and encode them as a string $s \in \{-1, 1\}^m$. By Chernoff-Hoeffding bounds, with probability at least $1/2$, s is useful for c. Assume that this is the case, allowing the algorithm to fail with probability $1/2$ so far. We know that $(s, c) \in C'$ since s is useful for c. We show that we can simulate a CSQ oracle for (s, c) under distribution D'.

Let $\psi : X' \to [-1, 1]$ be a polynomially evaluatable function. Then,

$$\mathbb{E}_{x'\sim D'}[\psi(x')c'(x)] = \frac{1}{2m}\sum_{i=1}^{m} \psi(e_i^m(-1)^n)s_i + \frac{1}{2}\mathbb{E}_{x\sim D}[\psi((-1)^m x)c(x)].$$

Since we have access to ψ and s, the first term in the above expression can be computed exactly. With probability at least $1 - \delta$, the second term can be estimated to accuracy τ, by drawing a sample of size $O(\frac{1}{\tau^2}\log(1/\delta))$ and computing the empirical estimate. If $\mathcal{B}$ makes q queries, by choosing $\delta = 1/4q$, we can ensure that with probability at least $1/4$ all the queries are answered correctly.

Suppose $\mathcal{B}$ outputs a hypothesis $h' : \{-1, 1\}^{n'} \to \{-1, 1\}$ such that $\mathrm{err}_{D'}(h, c') \leq \epsilon$. Then, define $h : X \to \{-1, 1\}$, where $h(x) = h'((-1)^m x)$; it is easy to see that $\mathrm{err}_D(h, c) \leq 2\epsilon$. Also, the algorithm succeeds with probability at least $1/4$. To get the algorithm to succeed with probability $1 - \delta$, this can be repeated $O(\log(1/\delta))$ times. □

If C is the class $\mathsf{CKT}^n_{p(n)}$, $\log|C|$ is bounded by some polynomial in n. Proposition 1 and 2 together with Theorem 2 immediately imply our main result.

Theorem 3. *If Assumption 1 is true, then there exists a distribution family $\mathcal{D} = \langle D_n \rangle_{n \geq 0}$ over $\langle X_n = \{-1,1\}^n \rangle_{n \geq 0}$ and a concept class family $\mathcal{C} = \langle C_n \rangle_{n \geq 0}$ such that*

1. *$\mathcal{C}$ is evolvable under $\mathcal{D}$ by an information-theoretically efficient algorithm with unbounded computation.*
2. *$\mathcal{C}$ is not computationally efficiently evolvable under $\mathcal{D}$.*

3.1 Uniform Distribution

In this section, we show that Theorem 3 can be extended to the case where the distribution D_n is the uniform distribution over $X_n = \{-1,1\}^n$. To avoid excessive notation, we will drop the subscript n in the rest of this section. Let U denote the uniform distribution over X and let C be a concept class over X.

Let $N : X \to \{0, \ldots, 2^n - 1\}$ be a bijective map, where $N(x) = i$, if x is the binary representation of i with each 0 replaced by a -1. Note that given x, $N(x)$ can be evaluated easily in polynomial time. We define a concept class C' over $X = \{-1,1\}^n$ related to concept class C. As defined earlier $M = 16 \log |C|/\epsilon^2$, $m = (n+1)M$. Recall that for c in C, we say that $s \in \{-1,1\}^m$ is useful for c, if ERM with input s outputs an *accurate* hypothesis. (Here, ERM is successful with respect to the uniform distribution U.) Thus, define C',

$$C' = \{(s,c) \mid c \in C, s \in \{-1,1\}^m \text{ is useful for } c\}$$

A concept $c' = (s,c) \in C'$ is defined as:

1. $c'(x) = s_i$ if $2^{n-2}(i-1)/m < N(x) \leq 2^{n-2}i/m$ for $i = 1, \ldots, m$.
2. $c'(x) = c(x)$ otherwise.

Thus, for each bit s_i of the string s, $1/(4m)$ mass of the uniform distribution U, is used to store the bit s_i. Thus, a CSQ algorithm is able to recover bit s_i easily.

Proposition 3. *There exists a CSQ algorithm $\mathcal{A}$ which learns C' under U, and furthermore,*

1. *$\mathcal{A}$ makes m queries, each of tolerance $1/5m$.*
2. *$\mathcal{A}$ need not be computationally efficient.*

Proof. Let $c' = (s,c)$ be the target concept. Define $\phi_i : X \to [-1,1]$ as: $\phi_i(x) = 1$ for $2^{n-2}(i-1)/m < N(x) \leq 2^{n-2}i/m$ and $\phi_i(x) = 0$ otherwise. Note that since N is polynomially evaluatable, so is ϕ_i. Then,

$$\mathbb{E}_{x \sim U}[\phi_i(x) c'(x)] = \frac{s_i}{4m}$$

Thus, the string s can be recovered by making m queries, each of tolerance $1/5m$. Let $h : X \to \{-1,1\}$ be the hypothesis returned by ERM on s. Then define $h' : X \to \{-1,1\}$, where $h'(x) = s_i$ if $2^{n-2}(i-1)/m < N(x) \leq 2^{n-2}i/m$ and $h'(x) = h(x)$ otherwise. Clearly, $\mathrm{err}_U(h', c') \leq \epsilon$. □

It is also easy to see that a polynomial time algorithm for learning C' under U would imply a weak-PAC learning algorithm for C under U.

Proposition 4. *If there exists a polynomial-time algorithm $\mathcal{B}$ that CSQ-learns C' with respect to U, then C can be weakly PAC-learned under U*

Proof. We construct an algorithm that learns weak PAC-learns C using $\mathcal{B}$. Let $c \in C$ be the target concept. Draw a sample $S = \langle (x^i, y^i) \rangle_{i=1}^{M}$ of size M using $\mathsf{EX}(c, U)$ and let $s \in \{-1, 1\}^m$ be the encoding corresponding to S. By the Chernoff-Hoeffding bound, with probability at least $1/2$, s is useful for c. Suppose that this is the case allowing the algorithm to fail with probability $1/2$, so that $c' = (s, c) \in C'$.

Let $\psi : X \to [-1, 1]$ be a polynomially evaluatable function. Note that an example oracle $\mathsf{EX}(c', U)$ can be easily simulated using $\mathsf{EX}(c, U)$ as follows: Draw $(x, y) \sim \mathsf{EX}(c, U)$, if $2^{n-2}(i-1)/m < N(x) \leq 2^{n-2}i/m$ for some i, replace (x, y) by (x, s_i). A $\mathsf{CSQ}(c', U)$ oracle is easily simulated using $\mathsf{EX}(c', U)$ by drawing a sample and returning the empirical estimate. Let $h' : X \to \{-1, 1\}$ be the hypothesis output by $\mathcal{B}$. Note that $\mathrm{err}_U(h, c') \leq \epsilon$ implies that $\mathrm{err}_U(h, c) \leq 1/4 + \epsilon$, since except on 1/4th fraction of the instance space, $c = c'$. □

Thus, Theorem 3 holds even when D_n is the uniform distribution over $X = \{-1, 1\}^n$.

4 Conclusion and Future Work

Prior to our work, the known bottlenecks for evolution in Valiant's model were information-theoretic. The question whether evolution is constrained due to computational bottlenecks or merely because of information-theoretic ones is an interesting one. We have shown that computational bottlenecks exist for evolution, even when information-theoretic ones don't.

The concept class that we construct to prove our result is not very natural. Thus, the main open question is to find a more natural class with the same property. We observe that the search space for such a class is quite limited, since even conjunctions are not evolvable due to information-theoretic bottlenecks in a distribution-independent sense.

With respect to fixed distributions, Feldman [Fel08] showed that evolvability is equivalent to SQ learning. It is an interesting question to find a *natural* concept class that is *information-theoretically* efficiently learnable in the SQ model with unbounded computation, but is not computationally efficiently SQ learnable (under plausible hardness assumptions). Even in the SQ model, the search space for such a class is limited; decision-lists can be efficiently learned, while decision-trees and DNFs are not learnable for information-theoretic reasons. In the case of monotone concept classes, O'Donnell and Servedio [OS07] have shown that monotone decision trees can be learned for any constant ϵ in the SQ model[1]. On the other hand Feldman, Lee and Servedio [FLS11] have shown that depth-3 monotone formulas are not strongly learnable and depth-4 monotone formulate are not even weakly learnable[2], both based on information-theoretic arguments.

Acknowledgments

The question considered in this paper was suggested by Leslie Valiant. I would like to thank Salil Vadhan and Leslie Valiant for helpful discussions.

References

[ABX08] B. Applebaum, B. Barak, and D. Xiao. On basing lower-bounds for learning on worst-case assumptions. In *Proceedings of the 49th Annual IEEE symposium on Foundations of computer science*, 2008.

[1] Although not explicitly stated in their paper, their algorithm is an SQ algorithm.

[2] The precise statements of their result is that depth-4 monotone circuits cannot be SQ-learned to a certain $1/2 + o(1)$ accuracy. All monotone concept classes can be learned to accuracy $1/2 + \Theta(\log n)/\sqrt{n}$ [OW09].

[BF02] N. H. 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 the twenty-sixth annual ACM symposium on Theory of computing*, 1994.

[Fel08] V. Feldman. Evolvability from learning algorithms. In *Proceedings of ACM STOC 40*, 2008.

[Fel09a] V. Feldman. A complete characterization of statistical query learning with applications to evolvability. In *Proceedings of the IEEE Symposium on Foundation of Computer Science*, 2009.

[Fel09b] V. Feldman. Robustness of evolvability. In *COLT 22*, 2009.

[Fel11] V. Feldman. Distribution-independent evolvability of linear threshold functions. In *COLT*, 2011.

[FLS11] V. Feldman, H. Lee, and R. A. Servedio. Lower bounds and hardness amplification for learning shallow monotone formulas. In *COLT*, 2011.

[GGM86] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

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

[Kha93] M. Kharitonov. Cryptographic hardness of distribution-specific learning. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, 1993.

[KV94] M. J. Kearns and L. G. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *J. ACM*, 41(1):67–95, 1994.

[KVV10] V. Kanade, L. G. Valiant, and J. W. Vaughan. Evolution with drifting targets. In *COLT*, 2010.

[OS07] R. O'Donnell and R. A. Servedio. Learning monotone decision trees in polynomial time. *SIAM J. Computing*, 37(3):827–844, 2007.

[OW09] R. O'Donnell and K. Wimmer. Kkl, kruskal-katona, and monotone nets. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, 2009.

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

[Val09] L. G. Valiant. Evolvability. *Journal of the ACM*, 56(1):1–21, 2009.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de