Computational Questions in Evolution

HARVARD UNIVERSITY
Graduate School of Arts and Sciences

DISSERTATION ACCEPTANCE CERTIFICATE

The undersigned, appointed by the

School of Engineering and Applied Sciences

have examined a dissertation entitled

"Computational Questions in Evolution"

presented by Varun Nandkumar Kanade

candidate for the degree of Doctor of Philosophy and hereby certify that it is worthy of acceptance.

Signature [illegible]

Typed name: Prof. L. Valiant

Signature [illegible]

Typed name: Prof. M. Mitzenmacher

Signature [illegible]

Typed name: Prof. S. Vadhan

Signature [illegible]

Typed name: Prof. A. Kalai

Date May 30, 2012

Computational Questions in Evolution

A dissertation presented
by

Varun Nandkumar Kanade

to
The School of Engineering and Applied Sciences
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
in the subject of

Computer Science

Harvard University
Cambridge, Massachusetts
August 2012

Thesis advisor Author

Leslie G. Valiant **Varun Nandkumar Kanade**

Computational Questions in Evolution

Abstract

Darwin's theory (1859) proposes that evolution progresses by the survival of those individuals in the population that have greater fitness. Modern understanding of Darwinian evolution is that variation in *phenotype*, or functional behavior, is caused by variation in *genotype*, or the DNA sequence. However, a quantitative understanding of what functional behaviors may emerge through Darwinian mechanisms, within reasonable computational and information-theoretic resources, has not been established. Valiant (2006) proposed a computational model to address the question of the complexity of functions that may be *evolved* through Darwinian mechanisms. In Valiant's model, the goal is to evolve a representation that computes a function that is *close* to some *ideal function* under the target distribution. While this evolution model can be simulated in the statistical query learning framework of Kearns (1993), Feldman has shown that under some constraints the reverse also holds, in the sense that learning algorithms in this framework may be cast as evolutionary mechanisms in Valiant's model.

In this thesis, we present three results in Valiant's computational model of evolution. The first shows that evolutionary mechanisms in this model can be made robust to gradual drift in the ideal function, and that such drift resistance is universal, in the sense that, if some concept class is evolvable when the ideal function is stationary, it is also evolvable in the setting when the ideal function drifts at some low rate.

The second result shows that under certain definitions of recombination and for certain selection mechanisms, evolution with recombination may be substantially faster. We show that in many cases polylogarithmic, rather than polynomial, generations are sufficient to evolve a concept class, whenever a suitable parallel learning algorithm exists.

The third result shows that computation, and not just information, is a limiting resource for evolution. We show that when computational resources in Valiant's model are allowed to be unbounded, while requiring that the information-theoretic resources be polynomially bounded, more concept classes are evolvable. This result is based on widely believed conjectures from complexity theory.

Contents

Bibliographic Note

Chapter 1 is an introduction to the contents and structure of this thesis. Chapter 2 reviews some frameworks studied in computational learning theory – the probably approximately correct (PAC) learning framework [38], the statistical query (SQ) learning framework [22] and Feldman's more recent correlational SQ (or CSQ) framework [12]. Chapter 3 describes Valiant's model of evolvability [39], some extensions to the model and some existing results. The subject matter described in Chapter 3 appears in [39, 12, 14, 40, 21] and, with the exception of Section 3.7.2, is not based on research performed by the author. The author's contribution is an attempt to unify and simplify some of the existing results.

Most of the contents of Chapters 4, 5 and 6 have been previously published in some form. Chapter 4 is based on the paper – "*Evolution with Drifting Targets*" (COLT 2010 [21]) co-authored with Leslie G. Valiant and Jennifer Wortman Vaughan. Chapter 5 is based on the paper "*Evolution with Recombination*" (FOCS 2011 [20]). Chapter 6 is based on the paper "*Computational Bounds on Statistical Query Learning*" (COLT 2012 [16]) co-authored with Vitaly Feldman.

Acknowledgments

I am extremely fortunate to have had Leslie Valiant and Adam Kalai as advisers in two different stages of my graduate studies. Les has been a great source of inspiration and a vast resource of knowledge. Through interactions with him, I hope that I have begun to understand how to separate the fundamental from the incidental, and hope that I will follow his ideal in trying to find answers to questions that really matter. Adam introduced me to computational learning theory. I thank him for letting me move with him to Boston and for all the dinners, beers, poker games and most importantly research problems and ideas. To both Les and Adam, I am grateful for patiently listening to my often silly ideas and for having confidence in my ability to conduct research, even at times when I didn't. I hope I have learned from them, at least to some degree, how to be a good adviser and teacher.

I thank Michael Mitzenmacher and Salil Vadhan for serving on my thesis committee and all their help during my stay at Harvard. I thank Yiling Chen for serving on my qualifying examination committee. Michael Rabin has been a great source of inspiration, I learned a lot from his courses, both as a student and teaching fellow. To Margo Seltzer I owe my (very limited) knowledge of systems and thank her for making the experience at Harvard more enjoyable, inside and outside of her CS265 class. Sham Kakade and Santosh Vempala have played a very helpful role in my research career. To Dana Randall, I remain indebted; she was the first person with whom I started research and she was most supportive in my first (and at times trying) year in graduate school. At Harvard, Carol Harlow (in SEAS) and Darryl Ziegler (in HIO), made paperwork seem painless and for that I am extremely grateful to them.

My time at Georgia Tech was enjoyable mostly due to the wonderful (and too large to list here) theory group.

I have been fortunate to have had a chance to collaborate with and talk to a number of outstanding researchers – Pranjal Awasthi, David Bader, Nina Balcan, Vitaly Feldman, Daniel Hsu, Sham Kakade, Kamesh Madduri, Yishay Mansour, Brendan McMahan, Ohad Shamir, Jennifer Wortman Vaughan, and Santosh Vempala. I must also thank all my math teachers from high-school and IIT who stimulated by interest in theoretical computer science one way or another.

My move to and stay in the Boston area would not have been nearly as pleasant without the friendship of Karthik, Veena and Dipti. Purushottam has been a great friend whose support in matters academic and otherwise, I have always been able to count on. The years spent during graduate school would not have been as fun without times with friends from high-school and college – Anjor, Gayatri, Gireeja, Jay, Mandar G., Mandar H., Niyati, Onkar, Rohit, Samyogita, Shubhangi, Shweta, Sukhada and Sumedh. My housemates Harriotte, Véra, Cynthia, Tom and Laini made it enjoyable to occasionally go home early.

Maxwell Dworkin 138 has been the most wonderful office – Kai-Min, Zhenming, Anna, Colin, Jon, Justin, Thomas, Scott, James, SueYeon and Jiayang. From them I have learned a lot (theory and otherwise) and with them I have had a lot of fun (in theory and otherwise). Other friends in MD – Alice, Brendan, Elaine, Elena, Guille, Heather – have made a time spent here much more enjoyable. Anna (the only non-theorist in MD 138) gets particular credit for making MD 138 the wonderful place it is; Thomas for his love for the Queen; Justin for his jokes and 1 minute discussions that last a few hours; Elena for being the constant provider of chocolates; Elaine for being the provider of food, drink, French movies, a teacher of Californian accents and much more.

My family, in India and in the States, has been very supportive in innumerable ways during my graduate study – but in particular I would like to thank my parents for believing in me and (almost) always letting me do what I want.

This work would not have been possible without a fellowship by the Harvard School of Engineering and Applied Sciences and support from NSF grants CCF-04-27129 and CCF-09-64401.

to Jeeves and Wooster

Chapter 1

Introduction and summary

Darwin's theory of natural selection proposes that species evolved from a single or few common ancestors [8]. Darwin's crucial observation was that more creatures are born, than could possibly be supported by the environment, and so those that were more "fit" were more likely to survive. Central to the theory of evolution is the notion of inheritance, by which genetic material is passed down to progeny. The understanding from modern genetics is that the functional behavior of an organism is encoded in its genome (DNA sequence)[1]. For selection to play any role in adaptation of organisms to their environment, variation within organisms is essential; this variability is caused by mutations (changes) in the genetic code (DNA sequence).

Although, we know that the functional behavior or *phenotype* of an organism is controlled by the *genotype* (DNA sequence), the relationship between genotype and phenotype, in particular how changes in genotype affect phenotype, is not yet

[1]It is known that there are *epigenetic* (*i.e.* not directly encoded as a sequence in the DNA) factors that affect expression of inherited functional traits. However, the extent to which these factors affect the rate of evolution is not known.

well understood. The DNA of an organism encodes sequences of amino acids that form proteins, and also encodes regulatory networks that govern interactions between these proteins. A regulatory network can be thought of as a circuit computing a function, where the inputs are concentration levels of a certain set of proteins and other molecules, and the outputs expression levels of the various proteins. An important question that is largely unanswered in our current understanding of evolution is that of the nature of complexity that can emerge in these genetic circuits.

Valiant [39] proposed a computational model of evolution to study this question in the framework of computational learning theory and to understand the complexity of genetic circuits by the mathematical functions they represent. An organism in Valiant's model is defined by a representation (its genotype), r, encoded as a string that is also interpreted as a function over some domain $\mathcal{X}$. A point x in the domain can be though of as the input (e.g. concentration levels of proteins) that results from a certain environmental condition. For example, environmental conditions such as the presence of sugar molecules, oxygen levels, moisture and temperature can cause concentration levels of proteins to change. There is presumed to be an *ideal function* that indicates the optimal behavior (output of the circuit) in any possible environmental condition. The complexity of the ideal function is captured by a class of functions that contain it, e.g. the ideal function being a low-weight threshold function. The performance of an organism is defined as a measure of fitness and expresses how well the function represented by the organism compares to the ideal.

Valiant introduces a notion of mutation of the representation and models natural selection as favoring those mutations that increase performance. An important constraint on evolution is population and representation size, which limits the number of feasible mutations that can be explored. Valiant asks which classes of functions allow for evolution of a representation that is close to the ideal function (in that class) within realistic time frames and reasonable population sizes. Also of interest is to identify classes that are provably not evolvable within feasible resources.

Evolution in this model is viewed as a restricted form of learning, where exploration is limited to mutations and feedback is only obtained through (natural) selection. A key difference between this form of learning from the classical setting of learning from examples [38] is that a candidate hypothesis is judged only by its aggregate performance (how well it does on some range of environmental conditions) and not on its performance on specific examples. Indeed, this model of learning is related to the statistical query learning model [22]; Valiant observed that an evolutionary process could be simulated in the statistical query learning framework and it was shown by Feldman [12] that a restricted form of statistical query learning algorithms could be case as evolutionary mechanisms in Valiant's model. This thesis addresses three questions in Valiant's computational model of evolution.

Structure of the thesis

- Chapter 2 reviews a few models from computational learning theory. The main model of interest is Kearns' statistical query learning (SQ) framework [22] and its restriction – correlational statistical query (CSQ) learning framework [12].

- Chapter 3 describes in detail Valiant's model of evolution [39] and variants introduced by Feldman [14] and P. Valiant [40]. A few simple evolutionary mechanisms for some concept classes are described. This chapter also contains Feldman's proof showing the equivalence between CSQ learning and evolution.

- Chapter 4 is concerned with the question of understanding how drift in the ideal function affects evolvability in Valiant's model. The main result is that for any concept class that is evolvable at all, there exists an (adaptive) evolutionary mechanism that tolerates a non-negligible (though possibly small) drift rate in the ideal function. The main result exploits the connection of evolvability to statistical query learning. For some specific concept classes, evolutionary mechanisms that tolerate drift are presented. Most of Chapter 4 appeared in [21].

- Chapter 5 explores the role of recombination and sexual reproduction in evolution. The question is a difficult one on which no consensus has so far been reached. On the one hand there seems to be an obvious cost to sex – recombination may break down genetic combinations with high fitness. On the other hand, the fact that almost all higher organisms reproduce sexually suggests that genes that enable recombination are indeed selected for.

  Fisher [18] and Muller [30] proposed the idea that sexual reproduction increases the speed of evolution. Their explanation is the following:

  Suppose that $a \rightarrow A$ and $b \rightarrow B$ are two beneficial mutations that could occur in an organism and furthermore that the effect of these mutations is additive. Suppose that the population initially only had a and b alleles. Then, in the absence of recombination in order for both A and B to be fixed in the population, the second mutation say $b \to B$ must occur in a descendant of an organism which already has undergone the mutation $a \to A$. On the other hand, in a population that reproduces sexually, recombination allows the appearance of a member containing both A and B alleles in possibly far fewer generations.

Chapter 5 reviews a few other explanations from the biology literature on the possible advantages of recombination. The one most relevant to work in this thesis is that outlined above due to Fisher and Muller. It is shown that under certain selection mechanisms, it is possible that evolution of a certain functionality may be significantly accelerated if the corresponding statistical query learning algorithm is parallelizable. Most of the material in this chapter appeared in [20].

- Chapter 6 investigates the limitations on evolvability. The fact that evolvability of a concept class implies learnability in the statistical query model immediately imposes limitations on evolvability. For example it was shown by Kearns [22] that the class of parity functions is not learnable in the SQ model. Later other concept classes such as polynomial size decision-trees and polynomial size disjunction normal form (DNF) formulae were shown not to be learnable in the statistical query model [6]. Even shallow monotone circuits have been shown not to be SQ learnable [17]. In all of these results, it is shown that the impediment to learning is that polynomially many queries do not give sufficient information about the target function. Thus, these results hold even if the learning algorithm is allowed unbounded computation.

  The question of interest is whether computation is a barrier to evolution (or

SQ learning) in cases when information is not, *i.e.* does their exist a concept class that can be learned with few (polynomially many) queries if the learner is allowed infinite computational power, but not if the learner is computationally (polynomially) bounded. The question is answered in the affirmative by explicit construction of such a concept class. Most parts of this chapter appeared in [16].

Chapter 2

Learning Models

This chapter reviews some standard frameworks from computational learning theory. Computational learning theory [38] was introduced by Valiant in a seminal paper and initiated a formal study of mechanistic learning. We describe the following three frameworks of learning relevant to the work in this thesis:

1. The probably approximate correct (PAC) learning framework [38] – learning from random examples.

2. The statistical query (SQ) learning framework [22] - where the learner does not have access to random examples, but only aggregate statistics of the function being learned.

3. The correlational statistical query (CSQ) learning framework [12], which is a restriction of the SQ framework and is closely related to Valiant's notion of evolvability [39, 12].

2.1 Setting

In all these learning frameworks, the common goal is to *learn* an (unknown) target function, $f : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ is the instance space (typically $\mathcal{X} = \{-1, 1\}^n$ or $\mathcal{X} = \mathbb{R}^n$) and $\mathcal{Y}$ is the range of the function (typically $\mathcal{Y} = \{-1, 1\}$ corresponding to a boolean function or $\mathcal{Y} \subseteq \mathbb{R}$). The target function is assumed to come from a class of functions, C (e.g. linear threshold functions, polynomial-size DNF expressions, polynomials). A parameter[1], n, corresponds to the size of the examples in the instance space $\mathcal{X}$; in most cases this size parameter is natural such as the dimension of the boolean cube or real space. There is a distribution D over the examples in $\mathcal{X}$ and it is supposed that D comes from a class of distributions $\mathcal{D}$.

Let $\mathcal{Y}'$ be a set such that $\mathcal{Y} \subseteq \mathcal{Y}'$ and let $\ell : \mathcal{Y}' \times \mathcal{Y} \to \mathbb{R}$ be a loss function. We require that the loss function satisfy the properties: For every $y \in \mathcal{Y}$, $\min_{y' \in \mathcal{Y}'} \ell(y', y) = \ell(y, y) = 0$ and that for every $y' \in \mathcal{Y}', y \in \mathcal{Y}$, $\ell(y', y) \leq B$ for some B that depends only on $\mathcal{Y}'$. Thus, the loss function is consistent and bounded. The goal of the learning algorithm (learner) is to output a hypothesis $h : \mathcal{X} \to \mathcal{Y}'$, such that the expected loss of h with respect to the target function $f : \mathcal{X} \to \mathcal{Y}$ and the target distribution, D, is $\mathrm{L}_{f,D}(h) = \mathbb{E}_{x \sim D}[\ell(h(x), f(x))] \leq \epsilon$. We refer to ϵ as the *accuracy* parameter.

The frameworks of learning differ in the manner in which the learning algorithm is allowed access to the target function. In the PAC setting, the learner may request

[1]Typically, in computational learning theory, the instance space, concept class and distribution are defined as families over all possible size parameters $n \geq 0$. This allows one to meaningfully talk about asymptotics in terms of the size parameter n. To simplify presentation, we will implicitly assume that this is the case, rather than define these terms as families.

random labeled examples, $(x, f(x))$, where x is drawn from the distribution D; in the SQ setting, the learner may request the (approximate) statistic $\mathbb{E}_{x\sim D}[\psi(x, f(x))]$ where $\psi(x, y)$ is some function defined by the learner; finally the CSQ setting is similar to the SQ setting, but the learner may only request (approximate) correlational statistics $\mathbb{E}_{x\sim D}[\psi(x)f(x)]$.

Remark 2.1. *The standard definitions of these models typically assume that the target functions and the output hypotheses are boolean. The standard loss function is the classification error ℓ_c, where $\ell_c(y', y) = 1$ if $y' \neq y$ and 0 otherwise. We define a more general setting because some of the evolutionary algorithms (e.g. [29, 14, 13, 15, 40]) do consider the range of the output hypothesis to be different from $\{-1, 1\}$ and the loss to be different from classification error.*

2.2 PAC Learning

In the probably approximately correct (PAC) learning setting, the learner is allowed access to an example oracle, $\mathsf{EX}(f, D)$, where f is the target function and D the target distribution. On being queried, the example oracle, $\mathsf{EX}(f, D)$, returns a random example $(x, f(x))$ where $x \sim D$. Formally, PAC learning is defined as:

Definition 2.1 (PAC Learning [38])**.** *A concept class, C, is said to be* probably approximately correctly *(PAC) learnable with respect to a class of distributions, $\mathcal{D}$, and loss function, ℓ, if there exists a (possibly randomized) learning algorithm LA, that for every $\epsilon, \delta > 0$, every target function $f \in C$ and every distribution $D \in \mathcal{D}$ with access to oracle $\mathsf{EX}(f, D)$ outputs a hypothesis, h, that satisfies with probability* *at least* $1 - \delta$,

$$\mathrm{L}_{f,D}(h) = \mathbb{E}_{x \sim D}[\ell(h(x), f(x)] \leq \epsilon.$$

Furthermore, the running time of the algorithm LA *must be bounded by a polynomial in* n*,* $1/\epsilon$*, and* $1/\delta$ *and the output hypothesis,* h*, is required to be polynomially evaluable.*

2.3 SQ Learning

The statistical query (SQ) model of learning was introduced by Kearns [22] as a technique to design noise-tolerant learning algorithms. Let $f : \mathcal{X} \to \mathcal{Y}$ be the target function and D be the target distribution. In this model, the learning algorithm only has access to a statistics oracle $\mathsf{STAT}(f, D)$. The query to the statistics oracle, $\mathsf{STAT}(f, D)$, is of the form (ψ, τ), where $\psi : \mathcal{X} \times \mathcal{Y} \to [-1, 1]$ is the query function[2] and τ is a tolerance parameter. The oracle responds with a value v that satisfies $|v - \mathbb{E}_{x \sim D}[\psi(x, f(x))]| \leq \tau$. Thus, the learning algorithm has access to approximate statistics about the target function and distribution (e.g. what is the fraction of examples for which $x_1 = 1$ and the label is -1?), but not labeled examples themselves.

Definition 2.2 (SQ Learning [22])**.** *A concept class,* C*, is said to be* learnable with statistical queries *(or SQ learnable) with respect to a class of distributions,* $\mathcal{D}$*, and loss function,* ℓ*, if there exists a (possibly randomized) learning algorithm,* LA*, that for every* $\epsilon > 0$*, every target function* $f \in C$ *and every distribution* $D \in \mathcal{D}$ *with access to oracle,* $\mathsf{STAT}(f, D)$ *outputs a hypothesis,* h*, that satisfies with probability at least*

[2]Kearns requires this function to be boolean. However, as was observed by Aslam and Decatur [1], a function with range $[-1, 1]$ can be viewed as a randomized boolean function and the definition of SQ learning is essentially unchanged.

1/2,

$$\mathrm{L}_{f,D}(h) = \mathbb{E}_{x \sim D}[\ell(h(x), f(x))] \leq \epsilon.$$

The running time of the algorithm must be bounded by a polynomial in n and $1/\epsilon$. Furthermore each query (ψ, τ) made to the oracle $\mathsf{STAT}(f, D)$ must be such that ψ is polynomially evaluable, $1/\tau$ is bounded by a polynomial in n, $1/\epsilon$ and the final output hypothesis, h, must be polynomially evaluable.

We discuss some variants of the statistical query learning framework in Chapter 6.

2.4 Correlational Statistical Query Learning

In this section, we restrict attention to the case where the target function, f, is boolean and the loss function is the classification error, ℓ_c, i.e. $\ell_c(y', y) = 1$ if $y' \neq y$ and 0 otherwise. In this case, we refer to the expected loss as $\mathrm{err}_{f,D}$ rather than $\mathrm{L}_{f,D}$.

A statistical query, $\psi : \mathcal{X} \times \{-1, 1\} \rightarrow [-1, 1]$, is said to be *target-independent* if $\psi(x, b) \equiv \psi^{\mathrm{ti}}(x)$ for some function $\psi^{\mathrm{ti}} : \mathcal{X} \rightarrow [-1, 1]$. A statistical query is said to be *correlational*, if $\psi(x, b) = b\psi^{\mathrm{cor}}(x)$ for some function $\psi^{\mathrm{cor}} : \mathcal{X} \rightarrow [-1, 1]$. Bshouty and Feldman [7] showed that if a learning algorithm is only allowed to make *target-independent* or *correlational* queries to the statistics oracle, $\mathsf{STAT}(f, D)$, then to get a valid response for an arbitrary query, (ψ, τ), it is sufficient to make two queries, one target-independent and one correlational, each with tolerance $\tau/2$.

Lemma 2.1 ([7]). *Let $\mathsf{STAT}(f, D)$ be the statistics oracle. It is possible to simulate a valid response of $\mathsf{STAT}(f, D)$ to the query (ψ, τ) where $\psi : \mathcal{X} \times \{-1, 1\} \rightarrow [-1, 1]$ by making two queries $(\psi^{ti}, \tau/2)$ and $(\psi^{cor}, \tau/2)$, where ψ^{ti} is target-independent and*

ψ^{cor} *is correlational.*

Proof. The key observation is the following (since $y \in \{-1, 1\}$):

$$\begin{aligned}\psi(x, y) &= \psi(x, 1)\frac{1+y}{2} + \psi(x, -1)\frac{1-y}{2} \\ &= \frac{\psi(x, 1) + \psi(x, -1)}{2} + y \cdot \frac{\psi(x, 1) - \psi(x, -1)}{2}\end{aligned}$$

Define $\psi^{\mathrm{ti}} : \mathcal{X} \to [-1, 1]$ to be $\psi^{\mathrm{ti}}(x) = (\psi(x, 1) + \psi(x, -1))/2$ and $\psi^{\mathrm{cor}} : \mathcal{X} \to [-1, 1]$ to be $\psi^{\mathrm{cor}}(x) = (\psi(x, 1) - \psi(x, -1))/2$. If v_1 and v_2 satisfy $|\mathbb{E}_{x \sim D}[\psi^{\mathrm{ti}}(x)] - v_1| \leq \tau/2$ and $|\mathbb{E}_{x \sim D}[f(x)\psi^{\mathrm{cor}}(x)] - v_2| \leq \tau/2$, then:

$$|\mathbb{E}_{x \sim D}[\psi(x, f(x))] - (v_1 + v_2)| = |\mathbb{E}_{x \sim D}[\psi^{\mathrm{ti}}(x)] + \mathbb{E}_{x \sim D}[f(x)\psi^{\mathrm{cor}}(x)] - v_1 - v_2| \leq \tau$$

□

The correlational statistical query (CSQ) learning model was introduced by Feldman [12]. In this model, the learning algorithm is only allowed to make correlational queries to the statistics oracle. Denote by $\mathsf{CSQ}\text{-}\mathcal{O}(f, D)$, the oracle that on receiving a query $(\psi^{\mathrm{cor}}, \tau)$, where $\psi^{\mathrm{cor}} : \mathcal{X} \to [-1, 1]$ is the query function and τ is the tolerance, outputs a value, v, such that $|\mathbb{E}_{x \sim D}[\psi^{\mathrm{cor}}(x)f(x)] - v| \leq \tau$. Formally, CSQ learning is defined as:

Definition 2.3 (CSQ Learning [12]). *A concept class, C, is said to be learnable with* correlational statistical queries *(or CSQ learnable) with respect to a class of distributions, $\mathcal{D}$, if there exists (a possibly randomized) learning algorithm,* LA, *that for every $\epsilon > 0$, every target function $f \in C$ and every distribution $D \in \mathcal{D}$, with access to oracle* $\mathsf{CSQ}\text{-}\mathcal{O}(f, D)$ *outputs hypothesis, h, that satisfies with probability at* *least* $1/2$,

$$\mathrm{err}_{f,D}(h) = \Pr_{x \sim D}[h(x) \neq f(x)] \leq \epsilon.$$

The running time of the algorithm must be bounded by a polynomial in n *and* $1/\epsilon$*. Furthermore, each query* (ψ^{cor}, τ) *made to the oracle* $\mathsf{CSQ}\text{-}\mathcal{O}(f, D)$ *must be such that* ψ^{cor} *is polynomially evaluable,* $1/\tau$ *is bounded by a polynomial in* n*, and* $1/\epsilon$*, and the final output hypothesis,* h*, is polynomially evaluable.*

The CSQ learning framework is discussed in greater detail in Chapter 6.

Chapter 3

Computational Model

This chapter presents Valiant's computational model of evolution [39] and some extensions to the model by Feldman [14] and P. Valiant [40]. This model and its extensions form the basis of most of the work in this thesis. For further details the reader is referred to Valiant's paper [39].

In Valiant's study of evolution, the objective is to understand quantitatively which mechanisms can develop in living organisms within a reasonable time frame and with realistic population sizes and which are too complex. For simplicity, evolution of a single functionality is discussed. The functional behavior of an organism is encoded in the genome as a gene regulatory network or a genetic circuit. The output of a genetic circuit is typically a function of concentration levels of several proteins. The output could be, for example, whether or not an enzyme is produced, or the expression level of a protein. The question of interest is to understand how complex these circuits could be, given that they have *evolved* through the processes of mutation and natural selection.

Valiant proposes understanding the question of complexity from the viewpoint of computational complexity, where complexity of functions is measured by the mathematical functions they realize. Evolution is treated as a form of computational learning; however, only the "aggregate fitness" of the hypothesis may affect the course of evolution, not the outcome on specific instances. This restriction is placed to model natural selection which only acts on the cumulative fitness of an individual and is blind to the outcome of an individual action in specific circumstances.

3.1 Environment and Ideal Function

An organism is treated as an entity that computes a *many-argument* function on some domain $\mathcal{X}$; typically, the settings of interest are $\mathcal{X} = \{-1, 1\}^n$ and $\mathcal{X} \subseteq \mathbb{R}^n$. It is assumed that n is a parameter[1] that represents the size of instances in the input space and it is required that the resources used should be bounded by a polynomial in n. Each instance, $x \in \mathcal{X}$, is considered as a "setting" of an environmental condition – an environmental condition could be represented by whether or not certain proteins are expressed or the concentration levels of various proteins.

An *ideal* function defines what the optimal output should be for each possible input setting or environmental condition. We allow the output space of the ideal function to be an arbitrary set $\mathcal{Y}$. Since the goal is to study feasibility of evolution depending on the complexity of the ideal function, as in learning theory, the ideal function is assumed to come from some concept class. A concept class, C, of functions

[1]This avoids the cumbersome notational overhead where instance spaces, concept classes and distribution classes are expressed as families for all values of $n \geq 0$.

from $\mathcal{X} \to \mathcal{Y}$ is simply defined to be a subset of functions from $\mathcal{X} \to \mathcal{Y}$, e.g. linear threshold functions, polynomial functions, polynomial-size decision trees.

In Valiant's work [39] the output space, $\mathcal{Y}$, is restricted to be $\{-1, 1\}$ – modeling a genetic circuit as a boolean function. Thus, the ideal function could be in the class of conjunctions, threshold functions etc. Later, P. Valiant introduced the setting where the output itself could be real-valued and he was particularly interested in the setting where $\mathcal{Y} \subseteq \mathbb{R}^m$. In this case the classes of interest could be sets of linear functions, polynomial functions etc.

In this thesis, we retain generality whenever possible and allow mechanisms to be functions from an arbitrary input space $\mathcal{X}$ to an arbitrary output space $\mathcal{Y}$. When further constraints are necessary for our results, we will mention those explicitly.

3.2 Representation

An organism is treated as a string that *represents* a function, the requirement being that there exist a polynomial time Turing machine that given the string representation, r, of a function and a point, $x \in \mathcal{X}$, outputs the value $r(x)$. Here, by a slight abuse of notation, the letter r is used to denote both the representation of the function as a string and also the function itself. The representation, r, computes a function $r : \mathcal{X} \to \mathcal{Y}'$ where $\mathcal{Y} \subseteq \mathcal{Y}'$. The output of a representation or organism is deliberately allowed to be outside of the range of the *ideal function*, as this could potentially allow for richer feedback. A *representation class*, R, is a set of string representations, where each (string) representation corresponds to a polynomially evaluable function over the instance space $\mathcal{X}$.

3.3 Performance (or Expected Loss)

The performance of an organism (or its representation) is a measure of how "close" it is to the ideal function. The goal of evolution is to achieve high *performance* or alternatively low *expected loss*. Suppose $\ell : \mathcal{Y}' \times \mathcal{Y} \rightarrow \mathbb{R}^+$ is a *loss function* that satisfies for every $y \in \mathcal{Y}$, $\min_{y' \in \mathcal{Y}'} \ell(y', y) = \ell(y, y) = 0$, i.e. the loss $\ell(y', y)$ is minimized when $y = y'$ and also that for every $y' \in \mathcal{Y}', y \in \mathcal{Y}$, $\ell(y', y) \leq B$ for some B that depends only on $\mathcal{Y}'$. In other words, the loss function, ℓ, is consistent and bounded. Suppose D is a distribution over the instance space $\mathcal{X}$; in the case of evolution, D indicates how likely it is that an organism might face particular kinds of environmental conditions.

For any representation, $r : \mathcal{X} \rightarrow \mathcal{Y}'$, we can define the expected loss with respect to an *ideal function* $f : \mathcal{X} \rightarrow \mathcal{Y}$ and distribution D, as $\mathrm{L}_{f,D}(r) = \mathbb{E}_{x \sim D}[\ell(r(x), f(x)]$. Alternatively, we may define performance[2] as $\ell\text{-Perf}_{f,D}(r) = 1 - 2\mathrm{L}_{f,D}(r)/B$, so that $\ell\text{-Perf}_{f,D}(r) \in [-1, 1]$.

The *goal* of evolution is to reach a representation that is ϵ-close to the ideal function, i.e. $\mathrm{L}_{f,D}(r) \leq \epsilon$ and for evolution to be considered feasible, it is required that this be achieved using resources that are bounded by a polynomial in n and $1/\epsilon$.

[2]Valiant's original paper [39] and indeed most of the follow-up work uses the notion of performance rather than expected loss. However, in the setting where we allow arbitrary input and output spaces, it is more convenient to describe the model and results in terms of expected loss.

3.4 Mutation

Mutations are caused due to changes in the DNA sequence, but the exact nature of mutations is as yet inadequately understood. Several different kinds of mutations are observed in nature – base substitution, deletion of a few base pairs, insertion of a few base pairs, copying a few base pairs, inversions and possibly other more involved ones. It is not known whether these mutations occur uniformly at random across the genome (see for example [33]). A more serious difficulty in modeling mutations of representations directly as changes in the genome sequence is that the relationship between genotype and phenotype is not well understood. Predicting the functional behavior of a section of the genome, given the sequence of base pairs, seems currently out of reach. However, it may still be possible to understand the limitations and potential of evolution in a more abstract sense. The extended Church-Turing thesis asserts that the processes of interpreting the genome as a function and mutation of the genome should be computable by an *efficient* Turing machine. Valiant's model allows the mutations to be expressed as the output of a polynomial time Turing machine that takes as input the existing representation. In some sense, this can be seen as allowing mutations to be caused by the most powerful (computationally feasible) process.

A *mutator* is defined a randomized polynomial time Turing machine that takes as input a representation, r, and a target accuracy parameter, ϵ, and outputs a new (possibly random) representation r'. Formally,

Definition 3.1 (Mutator). *A mutator for a representation class, R, is a randomized Turing machine, Mut, that takes as inputs a representation, $r \in R$, and ϵ, and outputs a representation r'. The running time of Mut is bounded by a polynomial in $|r|$ and*

$1/\epsilon$. *We think of* $\mathsf{Mut}(r, \epsilon)$ *as a random variable that takes values in* R.

3.5 Selection Rules

A key idea in the theory of evolution is natural selection, the fact that "more fit" members of a population are more likely to survive than "less fit" ones. Unfortunately, how to define fitness is often far from clear. In population genetics, fitness of a genotype is defined as the (expected) number of offspring that an individual with that genotype produces that live to maturity. However, this side-steps the question of what makes certain genotypes "more fit" than others from the point of view of functional behavior.

In Valiant's model of evolution, a natural notion of fitness is that of the performance (or expected loss) of individual genomes (representations). However, there still remains the difficulty of understanding how better performance with respect to a functionality translates to increased fitness in the sense of leaving greater numbers of progeny (which is how natural selection acts). This depends on how critical the functionality is for survival, which in turn decides the strength of natural selection. What can be said with certainty is that the fitness, or the chance of survival, of a genome should be an increasing function of its performance (decreasing function of the expected loss). A few different selection rules are discussed in this section and it has been shown by Feldman [14] and Valiant [39] that there is a certain degree of robustness in the model, in the sense that which concept classes are *evolvable* remains more or less unchanged for several choices of selection rules.

The size of the population places a constraint on the number of mutations that can be explored for natural selection to act on. Given two representations, the one with lesser expected loss (greater performance) is more likely to survive; however, it is unrealistic to expect that differences in expected loss that are negligible (superpolynomially small) will be "detected" by natural selection. Thus, selection is governed not by the exact expected loss of a particular representation, but by an empirical estimate. To simplify presentation[3], we assume that there is an oracle $\widehat{\mathrm{L}}_{f,D}(\cdot,\cdot)$ that for ideal function, f, and distribution, D, when queried with a representation, r, and some approximation parameter, τ, returns a value $v = \widehat{\mathrm{L}}_{f,D}(r,\tau)$ such that $|v - \mathrm{L}_{f,D}(r)| \leq \tau$, *i.e.* $\widehat{\mathrm{L}}_{f,D}(r,\tau)$ is a τ-(additive)-approximation to the expected loss, $\mathrm{L}_{f,D}(r)$. Furthermore, it is supposed that there is a tolerance parameter, t, that determines if the difference in (approximate) expected loss is large enough to be detected by natural selection.

The process of (natural) selection starting from a representation, r, first involves action of a mutator, Mut, which defines the neighborhood that may be explored. The number of candidate mutations explored is given by a (population) size parameter, s. The s-neighborhood, $\mathrm{Neigh}(r,\epsilon)$, of a representation, r, given accuracy parameter, ϵ, is a (random) multi-set obtained by running the mutator, $\mathsf{Mut}(r,\epsilon)$, s times. The empirical performances of each of these mutations is obtained by querying the oracle $\widehat{\mathrm{L}}_{f,D}(\cdot,\tau)$, for some approximation parameter τ. Finally, depending on the selection rule and the tolerance parameter, t, one of the mutations from the multi-set, $\mathrm{Neigh}(r,\epsilon)$, is *selected*. Each selection rule can be thought of as a random variable, that outputs a representation, r', for the next generation and we express this selection

[3]Valiant instead assumes that the expected loss (performance) is estimated by taking a sample from the distribution D.

process as:

$$r' \leftarrow \mathsf{Sel}(r, \epsilon, \mathsf{Mut}, \widehat{\mathrm{L}}_{f,D}(\cdot,\cdot), \tau, s, t)$$

Three specific selection rules are defined in this section. The first two appear in Valiant's paper [39] and the last implicitly in Feldman's work [14].

In all of the three selection rules described below, we use the following notation: Let r be the current representation, ϵ the accuracy parameter, s the size parameter (that determines the number of mutations explored), t the tolerance parameter, and τ the approximation parameter (for estimates of the expected loss). Then $\mathrm{Neigh}(r, \epsilon)$ is a multi-set obtained by running the Turing machine, $\mathsf{Mut}(r, \epsilon)$, s times. Let $\mathrm{Neigh}(r, \epsilon) = \{r_1, \ldots, r_s\}$ and denote by v_i the quantity $\widehat{\mathrm{L}}_{f,D}(r_i, \tau)$, so that v_i satisfies $|v_i - \mathrm{L}_{f,D}(r_i)| \leq \tau$. Also, let $v = \widehat{\mathrm{L}}_{f,D}(r, \tau)$, so that $|v - \mathrm{L}_{f,D}(r)| \leq \tau$.

3.5.1 Selection Based on Neutral and Beneficial Mutations

Beneficial mutations, Bene, are those whose (approximate) expected loss is noticeably lower than that of r. Neutral mutations, Neut, are those whose (approximate) expected loss is *not* noticeably different from that of r. The remaining mutations are considered deleterious. Whether or not the difference is *noticeable* is determined by a tolerance parameter, t. Formally,

$$\mathsf{Bene} = \{r_i \in \mathrm{Neigh}(r, \epsilon) \mid v_i < v - t\}$$

$$\mathsf{Neut} = \{r_i \in \mathrm{Neigh}(r, \epsilon) \mid |v_i - v| \leq t\}$$

Selection proceeds as follows: if the multi-set Bene is not empty, then a representation from Bene is chosen uniformly at random; if Bene is empty and Neut is non-empty, a representation from Neut is selected uniformly at random; if both Bene and Neut are empty the special representation, $\perp$, is selected, which indicates the end of the evolutionary process (or extinction). Although, the selection from Bene or Neut is done uniformly at random, the fact that these are multi-sets means that some beneficial (or neutral) mutations may be more likely to survive than others, depending on the probability with which the mutator generated these. We will denote this selection rule by $\mathsf{BN\text{-}Sel}$. In particular, if r' is the representation selected as an outcome of this process, we express this as:

$$r' \leftarrow \mathsf{BN\text{-}Sel}(r, \epsilon, \mathsf{Mut}, \widehat{\mathrm{L}}_{f,D}(\cdot, \cdot), \tau, s, t).$$

3.5.2 Selection based on Optimization

In this selection rule, one among the "best" (having least expected loss) mutations from the neighborhood is selected. Let $v^* = \min_{i \in [s]} v_i$. If $v^* > v + t$, *i.e.* the mutation with the least (approximate) expected loss is worse than the starting representation, r, the selection rule chooses the special representation, $\perp$, indicating an end to the evolutionary process. Otherwise, the multi-set, Opt, consists of the mutations from $\mathrm{Neigh}(r, \epsilon)$ that have least (approximate) expected loss (and are not noticeably worse than r). Formally,

$$\mathsf{Opt} = \{r_i \in \mathrm{Neigh}(r, \epsilon) \mid |v_i - v^*| \leq t \text{ and } v_i \leq v + t\}.$$

In this case, selection chooses a mutation from the multi-set Opt uniformly at random to be the representation for the next stage (generation). We denote this selection rule by $\mathsf{Opt\text{-}Sel}$. In particular, if r' is the representation selected as the outcome of this process, we express this as:

$$r' \leftarrow \mathsf{Opt\text{-}Sel}(r, \epsilon, \mathsf{Mut}, \widehat{\mathrm{L}}_{f,D}(\cdot,\cdot), \tau, s, t).$$

3.5.3 Weak Selection

In this selection rule, neutral and beneficial mutations are not distinguished explicitly. Thus, all neutral and beneficial mutations are considered feasible. Formally, denote the multi-set of feasible mutations, Feas, as:

$$\mathsf{Feas} = \{r_i \in \mathrm{Neigh}(r, \epsilon) \mid v_i \leq v + t\}.$$

Selection proceeds as follows: if Feas is not empty, a representation from Feas is selected uniformly at random; otherwise the representation for the next generation is set to $\perp$, indicating an end to the evolutionary process. We introduce this selection rule to model scenarios when selection pressures are weak, *e.g.* because the trait that is being evolved is not highly critical for the survival of an organism. We discuss this in greater detail in the context of evolution with recombination (see Chapter 5).

Feldman considers a more general (and arguably more natural) notion of weak selection [14]. However, as far as *evolvability* of concept classes is concerned, the weak selection rules seem to be identical. The interested reader is referred to Section 6 of [14].

3.6 Evolvability

In this section, we formally define the notion of *evolvability* of a concept class. Let $\mathcal{X}$ be the instance space (set of environments) and let C be a concept class of functions from $\mathcal{X} \rightarrow \mathcal{Y}$. Let $\mathcal{D}$ be a class of distributions over $\mathcal{X}$. Let n denote the parameter that characterizes the size of instances in $\mathcal{X}$ (*e.g.* if $\mathcal{X} = \{-1, 1\}^n$ or $\mathcal{X} \subseteq \mathbb{R}^n$). (Formally, we require that the length of the string representation of every $x \in \mathcal{X}$, denoted by $|x|$, is such that $|x|$ is bounded by some polynomial in n.) Let ϵ be the target *accuracy* parameter.

We consider the following components of an evolutionary mechanism:

- *Representation Class*: We say that a representation class, R, of (representations of) functions from $\mathcal{X} \rightarrow \mathcal{Y}'$, is polynomially bounded (with respect to n and ϵ), if for every $r \in R$, the length of the string encoding r, denoted by $|r|$, is bounded by a polynomial in n and $1/\epsilon$, and for every $x \in \mathcal{X}$, there is a polynomial time (in $|r|$ and $|x|$) Turing machine, that with r and x as inputs, outputs $r(x)$.

- *Mutator*: Given a polynomially bounded representation class, R, an evolutionary mechanism uses a polynomially-bounded mutator as defined in Definition 3.1.

- *Approximation Parameter*: An evolutionary mechanism uses an approximation parameter, τ. This is used to obtain (additive) approximate estimates of the expected loss of representations using the oracle $\widehat{\mathrm{L}}_{f,D}(\cdot, \tau)$.

- *Size Function*: Let R be a polynomially bounded representation class. Let $s : R \times \mathbb{R}^+ \rightarrow \mathbb{N}$ denote the *size* function, which is said to be polynomially bounded, if for every $r \in R$ and $\epsilon > 0$, $s(r, \epsilon)$ is bounded by some polynomial in n and $1/\epsilon$, and $s(r, \epsilon)$ can be computed in polynomial time. The size function determines the size of the neighborhood that is explored using the mutator.

- *Tolerance Function*: Let R be a polynomially bounded representation class. Let $t : R \times \mathbb{R}^+ \rightarrow \mathbb{R}^+$ denote the tolerance function, which is said to be polynomially sandwiched, if there exist polynomials $t\ell(\cdot, \cdot)$ and $tu(\cdot, \cdot)$ such that for every $r \in R$ and every $\epsilon > 0$

$$t\ell(n, 1/\epsilon) \leq 1/t(r, \epsilon) \leq tu(n, 1/\epsilon)$$

  and that there exists a constant c such that $tu(n, 1/\epsilon) \leq t\ell(n, 1/\epsilon)^c$. It is also required that $t(r, \epsilon)$ be computable in polynomial time.

Thus, we define an evolutionary mechanism as a 5-tuple, $\mathcal{E}M = (R, \mathsf{Mut}, \tau, s, t)$, consisting of a representation class, mutator, approximation parameter, size function and tolerance function. We say that an evolutionary mechanism, $\mathcal{E}M = (R, \mathsf{Mut}, \tau, s, t)$, is polynomially-bounded if R is a polynomially bounded representation class, Mut is a polynomially bounded mutator, $1/\tau$ is bounded by a polynomial in n and $1/\epsilon$, s is polynomially bounded and t is polynomially sandwiched.

Let $f \in C$ be the (target) ideal function and let $D \in \mathcal{D}$ be a distribution over $\mathcal{X}$. We define the notion of an evolutionary step involving an evolutionary mechanism, $\mathcal{E}M$, starting representation, $r_0 \in R$, and a selection rule, Sel. Formally,

Definition 3.2 (Evolutionary Step)**.** *For ideal function f and distribution D, an evolutionary step starting from representation r_0, using evolutionary mechanism $\mathcal{E}M$ and selection rule* Sel*, produces a representation $r' \in R$, where:*

$$r' \leftarrow \mathsf{Sel}(r, \epsilon, \mathsf{Mut}, \widehat{\mathrm{L}}_{f,D}(\cdot, \cdot), \tau, s(r, \epsilon), t(r, \epsilon)).$$

We refer to this in short as $r' \leftarrow \mathsf{ES}(r_0, \mathcal{E}M, \mathsf{Sel})$.

Thus, an evolutionary step is simply one stage of evolution involving mutation and (natural) selection steps. An evolutionary sequence, $r_0, r_1, r_2, \ldots$ is obtained by a series of evolutionary steps starting from r_0, where $r_i \leftarrow \mathsf{ES}(r_{i-1}, \mathcal{E}M, \mathsf{Sel})$ for all $i \geq 1$.

We can now define evolvability of a concept class, C, with respect to distribution class, $\mathcal{D}$, with loss function ℓ and selection rule Sel.

Definition 3.3 (Evolvability [39]). *A concept class, C, is said to be* evolvable *with respect to a class of distributions, $\mathcal{D}$, using a selection rule* Sel *and loss function ℓ, if there exists a polynomially bounded evolutionary mechanism, $\mathcal{E}M = (R, \mathsf{Mut}, \tau, s, t)$, such that for every $f \in C$ and every distribution, $D \in \mathcal{D}$, when for any $r : \mathcal{X} \rightarrow \mathcal{Y}'$, the expected loss defined as $\mathrm{L}_{f,D}(r) = \mathbb{E}_{x \sim D}[\ell(r(x), f(x))]$ and for any $r_0 \in R$, the evolutionary sequence, $r_0, r_1, \ldots, r_g$ obtained using $\mathcal{E}M$ and* Sel*, is such that $\mathrm{L}_{f,D}(r_g) \leq \epsilon$, and g is bounded by a polynomial in n and $1/\epsilon$.*

3.7 Some Evolutionary Algorithms

In order to better explain the notion of evolvability, we describe two evolutionary mechanisms. The first evolves the class of monotone conjunctions under the uniform distribution over the boolean cube. The second evolves the class of homogeneous linear separators in $\mathbb{R}^n$, with respect to radially symmetric distributions. In each case, we need to specify the representation class, mutator, τ, s and t. We also need to specify the selection rule and the loss function.

3.7.1 Monotone Conjunctions

The first evolutionary algorithm we describe is for the class of monotone conjunctions over the boolean cube, $\mathcal{X} = \{-1, 1\}^n$ (we assume that $n \geq 2$). Let $\mathcal{U}$ denote the uniform distribution over $\{-1, 1\}^n$ and in this section we assume that the loss function is the classification loss, ℓ_c (where $\ell_c(y', y) = 1$ if $y' \neq y$ and 0 otherwise)[4]. Formally, if $\mathcal{X} = \{-1, 1\}^n$, the class of monotone conjunctions is defined as

$$C = \{\bigwedge_{i \in S} x_i \mid S \subseteq [n]\}.$$

We use the representation class that is the same as the concept class, $R = C$. Thus, each representation will simply be a monotone conjunction. For a monotone conjunction $r \in R = C$, let $\mathsf{lit}(r)$ denote the set of indexes corresponding to the literals in r, so that $r \equiv \bigwedge_{i \in \mathsf{lit}(r)} x_i$. For example, if $r = x_2 \wedge x_7 \wedge x_{12}$, then $\mathsf{lit}(r) = \{2, 7, 12\}$. Let ϵ be the accuracy parameter.

Next, we define the mutator $\mathsf{Mut}(r, \epsilon)$ as follows:

- If $|\mathsf{lit}(r)| > \log_2(3/\epsilon)$, then the mutator, $\mathsf{Mut}(r, \epsilon)$, picks an $i \in \mathsf{lit}(r)$ uniformly at random and outputs a monotone conjunction, r', such that $\mathsf{lit}(r') = \mathsf{lit}(r) \backslash \{i\}$, *i.e.* r' is the monotone conjunction obtained by dropping the literal x_i from r.

- If $|\mathsf{lit}(r)| \leq \log_2(3/\epsilon)$, the mutator, $\mathsf{Mut}(r, \epsilon)$, does the following:

  1. *Deleting a literal*: With probability $1/(2n)$, it picks an $i \in \mathsf{lit}(r)$ uniformly at random and outputs r' such that $\mathsf{lit}(r') = \mathsf{lit}(r) \setminus \{i\}$.

[4]The result actually holds for every loss function, since both the ideal function and representations are boolean. In this case, as long as $\ell(1, -1) = \ell(-1, 1) \neq 0$, all loss functions are equivalent.

2. *Adding a literal*: If $|\mathsf{lit}(r)| + 1 \leq \log_2(3/\epsilon)$, with probability $1/(2|\mathsf{lit}(r)| + 2)$ (otherwise with probability 0), it picks an $i \notin \mathsf{lit}(r)$ uniformly at random and outputs, r', such that $\mathsf{lit}(r') = \mathsf{lit}(r) \cup \{i\}$ (or equivalently $r' \equiv r \wedge x_i$).

3. *Swapping literals*: With the remaining probability, it picks an $i \in \mathsf{lit}(r)$ uniformly at random and a $j \notin \mathsf{lit}(r)$ uniformly at random and outputs r', such that $\mathsf{lit}(r') = (\mathsf{lit}(r) \setminus \{i\}) \cup \{j\}$. Notice that when $n \geq 2$, the mutator outputs a mutation obtained by swapping literals with probability at least 1/4.

We use $\tau = \epsilon^2/36$ as the approximation parameter. The size function, $s : R \times \mathbb{R}^+ \to \mathbb{N}$, is defined as $s(r, \epsilon) = \lceil 4n(|\mathsf{lit}(r)| + 1) \log(36/\epsilon^3) \rceil$. Finally, the tolerance function, $t : R \times \mathbb{R}^+ \to \mathbb{R}^+$, is defined as $t(r, \epsilon) = \epsilon^2/12$, if $|\mathsf{lit}(r)| \leq \log_2(3/\epsilon)$ and $t(r, \epsilon) = \epsilon/2$, if $|\mathsf{lit}(r)| > \log_2(3/\epsilon)$. The proof of Theorem 3.1 first appeared in Valiant's work [39] and was later simplified by Diochnos and Turán [10]. The proof presented here is a minor modification of the latter.

Theorem 3.1 ([39, 10])**.** *Let C denote the class of monotone conjunctions on the instance space $\mathcal{X} = \{-1, 1\}^n$ and let $\mathcal{U}$ denote the uniform distribution over $\mathcal{X}$. Then C is evolvable to accuracy ϵ by the evolutionary mechanism, $\mathcal{E}M = (C, \mathsf{Mut}, \tau, s, t)$, described above in this section, using selection rule,* BN-Sel*, and classification error, ℓ_c, as loss function, in g generations, where $g \leq n - \lfloor \log_2(3/\epsilon) \rfloor + \lceil 36/\epsilon^2 \rceil$.*

Proof. When the loss function is the classification error, ℓ_c, for any two monotone conjunctions r_1 and r_2, $\mathrm{L}_{r_1, \mathcal{U}}(r_2) = \mathrm{L}_{r_2, \mathcal{U}}(r_1) = \Pr_{x \sim \mathcal{U}}[r_1(x) \neq r_2(x)]$. A useful observation is the following: for monotone conjunctions r_1 and r_2:

$$\begin{aligned}\Pr_{x\sim\mathcal{U}}[r_1(x) \neq r_2(x)] &= 2^{-|\mathsf{lit}(r_1)|}(1 - 2^{-|\mathsf{lit}(r_2)\setminus\mathsf{lit}(r_1)|}) + 2^{-|\mathsf{lit}(r_2)|}(1 - 2^{-|\mathsf{lit}(r_1)\setminus\mathsf{lit}(r_2)|}) \\ &= 2^{-|\mathsf{lit}(r_1)|} + 2^{-|\mathsf{lit}(r_2)|} - 2^{-|\mathsf{lit}(r_1)\cup\mathsf{lit}(r_2)|+1} \qquad (3.1)\end{aligned}$$

Let f denote the ideal function and note that $\mathsf{lit}(f)$ denotes the indexes of the literals in f. Let r_0 be the starting representation. First, suppose that $|\mathsf{lit}(r_0)| > \log_2(3/\epsilon)$. Let $r' = \mathsf{Mut}(r_0, \epsilon)$ be a random candidate mutation. Note that when $|\mathsf{lit}(r_0)| > \log_2(3/\epsilon)$, r' is simply obtained by deleting a literal from r_0. Thus, $|\mathsf{lit}(r')| = |\mathsf{lit}(r_0)| - 1$. Then using (3.1) we have,

$$\begin{aligned}\mathrm{L}_{f,\,\mathcal{U}}(r_0) - \mathrm{L}_{f,\,\mathcal{U}}(r') &= 2^{-|\mathsf{lit}(r_0)|} - 2^{-|\mathsf{lit}(r_0)\cup\mathsf{lit}(f)|+1} - 2^{-|\mathsf{lit}(r')|} + 2^{-|\mathsf{lit}(r')\cup\mathsf{lit}(f)|+1} \\ &= 2^{-|\mathsf{lit}(r_0)|} - 2^{-|\mathsf{lit}(r_0)|+1} + 2^{-|\mathsf{lit}(r')\cup\mathsf{lit}(f)|+1} - 2^{-|\mathsf{lit}(r_0)\cup\mathsf{lit}(f)|+1} \\ &\geq -2^{-|\mathsf{lit}(r_0)|} \geq -\epsilon/3\end{aligned}$$

where the last line follows from the fact that $2^{-|\mathsf{lit}(r')\cup\mathsf{lit}(f)|+1} - 2^{-|\mathsf{lit}(r_0)\cup\mathsf{lit}(f)|+1} \geq 0$ (since $\mathsf{lit}(r') \subset \mathsf{lit}(r_0)$) and the fact that $|\mathsf{lit}(r_0)| > \log_2(3/\epsilon)$. Note that when $|\mathsf{lit}(r_0)| > \log_2(3/\epsilon)$, the tolerance function is given by: $t(r_0, \epsilon) = \epsilon/2$. Thus, when $\tau = \epsilon^2/36$, let $\widehat{\mathrm{L}}_{f,\,\mathcal{U}}(r_0, \tau)$ and $\widehat{\mathrm{L}}_{f,\,\mathcal{U}}(r', \tau)$ denote the (approximate) values of the expected loss. It holds that $\widehat{\mathrm{L}}_{f,\,\mathcal{U}}(r', \tau) \leq \mathrm{L}_{f,\,\mathcal{U}}(r') + \tau$ and $\widehat{\mathrm{L}}_{f,\,\mathcal{U}}(r_0, \tau) \geq \mathrm{L}_{f,\,\mathcal{U}}(r_0) - \tau$. Then, we have the following:

$$\begin{aligned}\widehat{\mathrm{L}}_{f,\,\mathcal{U}}(r', \tau) - \widehat{\mathrm{L}}_{f,\,\mathcal{U}}(r_0, \tau) &\leq \mathrm{L}_{f,\,\mathcal{U}}(r') - \mathrm{L}_{f,\,\mathcal{U}}(r_0) + 2\tau \\ &\leq \frac{\epsilon}{3} + 2\frac{\epsilon^2}{36} \leq \frac{\epsilon}{2} = t(r_0, \epsilon)\end{aligned}$$

Thus every representation, r', that could have been output by the mutator $\mathsf{Mut}(r_0, \epsilon)$ is either neutral or beneficial. Also, since every such mutation, r', has one fewer literal compared to r_0, a representation that has at most $\log_2(3/\epsilon)$ literals will be reached in at most $n - \lfloor \log_2(3/\epsilon) \rfloor$ evolutionary steps (generations).

Thus, we may in fact assume that the starting representation, r_0, is such that $|\mathsf{lit}(r_0)| \leq \log_2(3/\epsilon)$. We show that unless a representation, r_0, for which $|\mathsf{lit}(r_0)| \leq \log_2(3/\epsilon)$, already satisfies $\mathrm{L}_{f,\,\mathcal{U}}(r_0) \leq \epsilon$, there exists a beneficial mutation, r', that would be output by the mutator, $\mathsf{Mut}(r_0, \epsilon)$ (with significant probability).

Case 1: Suppose there exists i, such that $i \in \mathsf{lit}(f)$ and $i \notin \mathsf{lit}(r_0)$ and furthermore suppose it is also the case that $|\mathsf{lit}(r_0)| + 1 \leq \log_2(3/\epsilon)$, then let r' be such that $\mathsf{lit}(r') = \mathsf{lit}(r_0) \cup \{i\}$, i.e. $r' = r_0 \wedge x_i$. Consider

$$\mathrm{L}_{f,\,\mathcal{U}}(r_0) - \mathrm{L}_{f,\,\mathcal{U}}(r') = 2^{-|\mathsf{lit}(r_0)|} - 2^{-|\mathsf{lit}(r_0) \cup \mathsf{lit}(f)|+1} - 2^{-|\mathsf{lit}(r')|} + 2^{-|\mathsf{lit}(r') \cup \mathsf{lit}(f)|+1}$$

Notice that $|\mathsf{lit}(r')| = |\mathsf{lit}(r_0)| + 1$ and $|\mathsf{lit}(r') \cup \mathsf{lit}(f)| = |\mathsf{lit}(r_0) \cup \mathsf{lit}(f)|$. Thus,

$$\mathrm{L}_{f,\,\mathcal{U}}(r_0) - \mathrm{L}_{f,\,\mathcal{U}}(r') = \frac{1}{2} \cdot 2^{-|\mathsf{lit}(r_0)|} \geq \epsilon/6$$

Note that since $t(r_0, \epsilon) = \epsilon^2/12$, when $\tau = \epsilon^2/36$ and for $\epsilon < 1$, we can easily show that $\widehat{\mathrm{L}}_{f,\,\mathcal{U}}(r', \tau) - \widehat{\mathrm{L}}_{f,\,\mathcal{U}}(r_0, \tau) \leq -\epsilon/6 + 2(\epsilon^2/36) < -\epsilon^2/12 = -t(r_0, \epsilon)$. Thus, the mutation, r', is beneficial. The last thing that remains to be checked is that the specific mutation, r', is in the neighborhood, $\mathrm{Neigh}(r_0, \epsilon)$, obtained by running the mutator, $\mathsf{Mut}(r_0, \epsilon)$, $s(r_0, \epsilon)$ times. Note that for a fixed i, the probability that mutator, $\mathsf{Mut}(r_0, \epsilon)$, outputs $r' = r_0 \wedge x_i$, is at least $1/((2|\mathsf{lit}(r_0)| + 2)n)$, thus when $s(r, \epsilon) = \lceil 4n(|\mathsf{lit}(r_0)| + 1) \ln(36/\epsilon^3) \rceil$, except with probability $\epsilon^3/36$, r' will be in $\mathrm{Neigh}(r_0, \epsilon)$. Thus, except with probability $\epsilon^3/36$, a beneficial mutation will be selected.

Case 2: Suppose that adding a literal to r_0 is not possible because $|\mathsf{lit}(r_0)| =$

$\lfloor \log_2(3/\epsilon) \rfloor$, but it is the case that there is some $i \in \mathsf{lit}(f)$ such that $i \notin \mathsf{lit}(r_0)$ and also that there is some $j \in \mathsf{lit}(r_0)$ such that $j \notin \mathsf{lit}(f)$. Let r' be the conjunction satisfying $\mathsf{lit}(r') = (\mathsf{lit}(r_0) \cup \{i\}) \setminus \{j\}$. Consider,

$$\mathrm{L}_{f,\, \mathcal{U}}(r_0) - \mathrm{L}_{f,\, \mathcal{U}}(r') = 2^{-|\mathsf{lit}(r_0)|} - 2^{-|\mathsf{lit}(r_0) \cup \mathsf{lit}(f)|+1} - 2^{-|\mathsf{lit}(r')|} + 2^{-|\mathsf{lit}(r') \cup \mathsf{lit}(f)|+1} \tag{3.2}$$

Notice that $|\mathsf{lit}(r')| = |\mathsf{lit}(r_0)|$ and $|\mathsf{lit}(r') \cup \mathsf{lit}(f)| = |\mathsf{lit}(r_0) \cup \mathsf{lit}(f)| - 1$. Also, we have assumed that $|\mathsf{lit}(r_0)| = \lfloor \log_2(3/\epsilon) \rfloor$. If $|\mathsf{lit}(f)| > \lfloor \log_2(3/\epsilon) \rfloor$, then using (3.1) it is easy to see that $\mathrm{L}_{f,\, \mathcal{U}}(r_0) \leq \epsilon$, in which case the evolutionary mechanism has succeeded. Thus, we may assume that $|\mathsf{lit}(f)| \leq \lfloor \log_2(3/\epsilon) \rfloor$, and hence $|\mathsf{lit}(r_0) \cup \mathsf{lit}(f)| \leq 2\lfloor \log_2(3/\epsilon) \rfloor$. Hence, (3.2) implies that:

$$\mathrm{L}_{f,\, \mathcal{U}}(r_0) - \mathrm{L}_{f,\, \mathcal{U}}(r') \geq 2 \cdot 2^{-|\mathsf{lit}(r_0) \cup \mathsf{lit}(f)|} \geq 2 \cdot \frac{\epsilon^2}{9}$$

Now, when $t(r_0, \epsilon) = \epsilon^2/12$ and $\tau = \epsilon^2/36$, it is easy to show that $\widehat{\mathrm{L}}_{f,\, \mathcal{U}}(r', \tau) - \widehat{\mathrm{L}}_{f,\, \mathcal{U}}(r_0, \tau) \leq \mathrm{L}_{f,\, \mathcal{U}}(r') - \mathrm{L}_{f,\, \mathcal{U}}(r_0) + 2\tau \leq -2(\epsilon^2/9) + 2(\epsilon^2/36) < -\epsilon^2/12 = -t(r_0, \epsilon)$. Thus, the mutation, r', is beneficial. Notice, that for any fixed i and j, the probability that the mutator outputs r' (as defined in this case) with probability at least $1/(4n(\mathsf{lit}(r_0) + 1))$. Thus, it is easy to see that when $s(r_0, \epsilon) = \lceil 4n(|\mathsf{lit}(r_0)| + 1) \ln(36/\epsilon^3) \rceil$, except with probability $\epsilon^3/36$, the specific (beneficial) mutation, r', will be in $\mathrm{Neigh}(r_0, \epsilon)$ (obtained by running $\mathsf{Mut}(r_0, \epsilon)$, $s(r_0, \epsilon)$ times). Thus, except with probability $\epsilon^3/36$, a beneficial mutation will be selected.

Case 3: Finally, if it is the case that no literal may be added to r_0 as in Case 1, or no literal may be swapped as in Case 2, then, $\mathsf{lit}(f) \subseteq \mathsf{lit}(r_0)$ and $|\mathsf{lit}(r_0)| \leq \log_2(3/\epsilon)$. Suppose $\mathsf{lit}(f) = \mathsf{lit}(r_0)$, then obviously evolution has succeeded. Otherwise, let $i \in \mathsf{lit}(r_0)$ such that $i \notin \mathsf{lit}(f)$. Let r' be the conjunction such that $\mathsf{lit}(r') = \mathsf{lit}(r_0) \setminus \{i\}$, i.e. $r_0 = r' \wedge x_i$. Observe that $\mathsf{lit}(r_0) \cup \mathsf{lit}(f) = \mathsf{lit}(r_0)$ and $\mathsf{lit}(r') \cup \mathsf{lit}(f) = \mathsf{lit}(r')$. Then using (3.1) we get,

$$\begin{aligned} \mathrm{L}_{f,\,\mathcal{U}}(r_0) - \mathrm{L}_{f,\,\mathcal{U}}(r') &= 2^{-|\mathsf{lit}(r_0)|} - 2 \cdot 2^{-|\mathsf{lit}(r_0)|} - 2^{-|\mathsf{lit}(r')|} + 2 \cdot 2^{-|\mathsf{lit}(r')|} \\ &= 2^{-|\mathsf{lit}(r')|} - 2^{-|\mathsf{lit}(r_0)|} \geq 2^{-|\mathsf{lit}(r_0)|} \geq \frac{\epsilon}{3} \end{aligned}$$

Thus, it is easily seen that for $t(r_0, \epsilon) = \epsilon^2/12$ and $\tau = \epsilon^2/36$, $\widehat{\mathrm{L}}_{f,\,\mathcal{U}}(r', \tau) - \widehat{\mathrm{L}}_{f,\,\mathcal{U}}(r_0, \tau) \leq -\epsilon/3 + 2\epsilon^2/36 < -\epsilon^2/12 = -t(r_0, \epsilon)$. Thus, r' is a beneficial mutation. Notice that for any $i \in \mathsf{lit}(r_0)$, the probability that r' (as defined in this case) is output by the mutator is at least $1/(2n(\mathsf{lit}(r_0) + 1))$. When $s(r_0, \epsilon) = \lceil 4n(|\mathsf{lit}(r_0)| + 1) \ln(36/\epsilon^3) \rceil$, except with probability $\epsilon^3/36$, the particular mutation, r', will be in $\mathrm{Neigh}(r_0, \epsilon)$. Thus, except with probability $\epsilon^3/36$, a beneficial mutation will be selected.

To finish the argument, consider the following: Suppose the starting representation, r'_0, was such that $|\mathsf{lit}(r'_0)| > \log_2(3/\epsilon)$, then in at most $n - \lfloor \log_2(3/\epsilon) \rfloor$ generations (evolutionary steps), a representation, r_0, such that $|\mathsf{lit}(r_0)| \leq \log_2(3/\epsilon)$ is reached. Consider the next $g' = 36/\epsilon^2$ generations (evolutionary steps) after the first such representation, r_0, satisfying $|\mathsf{lit}(r_0)| \leq \log_2(3/\epsilon)$, is reached. Call these representations $r_0, r_1, \ldots, r_{g'}$. As proved above, for each i, except with probability $\epsilon^3/36$, r_{i+1} must have been a beneficial mutation with respect to r_i. By a union bound, we can say that except with probability ϵ, all of these steps were such that a beneficial mutation was selected. But this implies that $\widehat{\mathrm{L}}_{f,\,\mathcal{U}}(r_{i+1}, \tau) \leq \widehat{\mathrm{L}}_{f,\,\mathcal{U}}(r_i, \tau) - t(r_i, \epsilon)$. Using the fact that $\mathrm{L}_{f,\,\mathcal{U}}(r) - \tau \leq \widehat{\mathrm{L}}_{f,\,\mathcal{U}}(r, \tau) \leq \mathrm{L}_{f,\,\mathcal{U}} + \tau$ for every $r \in R$, we get that $\mathrm{L}_{f,\,\mathcal{U}}(r_{i+1}) \leq \mathrm{L}_{f,\,\mathcal{U}}(r_i) - t(r_i, \epsilon) + 2\tau$. Since, for all $0 \leq i \leq g'$, $|\mathsf{lit}(r_i)| \leq \log_2(3/\epsilon)$, $t(r_i, \epsilon) = \epsilon^2/12$. Because $\tau = \epsilon^2/36$, we get that $\mathrm{L}_{f,\,\mathcal{U}}(r_{i+1}) \leq \mathrm{L}_{f,\,\mathcal{U}}(r_i) - \epsilon^2/36$. Using the fact that $\mathrm{L}_{f,\,\mathcal{U}}(r_0) \leq 1$, it must be the case that for some $i \leq g'(= 36/\epsilon^2)$, $\mathrm{L}_{f,\,\mathcal{U}}(r_i) \leq \epsilon$. $\square$

Remark 3.1. *The reader can easily check that the same evolutionary mechanism also evolves the class of monotone conjunctions under the uniform distribution, using the selection rule* $\mathsf{Opt\text{-}Sel}$*. This follows from the fact that when* r_0 *is such that* $|\mathsf{lit}(r_0)| > \log_2(3/\epsilon)$*, every mutation output by* $\mathsf{Mut}(r_0, \epsilon)$ *is either beneficial or neutral, and the number of literals in the resulting mutations is one fewer than* $|\mathsf{lit}(r_0)|$*. When* $|\mathsf{lit}(r_0)| \leq \log_2(3/\epsilon)$*, we have shown that with high probability a beneficial mutation exists in the neighborhood,* $\mathrm{Neigh}(r_0, \epsilon)$*. The selection rule,* $\mathsf{Opt\text{-}Sel}$*, chooses one among the "best" (those with least (approximate) expected loss) mutations in the neighborhood, and hence, will always choose a beneficial mutation.*

3.7.2 Homogeneous Linear Separators

In this section, we describe an evolutionary mechanism that evolves homogeneous linear separators in $\mathbb{R}^n$ under radially symmetric distributions [5]. Let $\mathbf{w} \in \mathbb{R}^n$ be a vector such that $\|\mathbf{w}\|_2 = 1$. Consider the function $h_{\mathbf{w}} : \mathbb{R}^n \to \{-1, 1\}$, such that for any point $x \in \mathbb{R}^n$, $h_{\mathbf{w}}(x) = \mathrm{sign}(\mathbf{w} \cdot x)$ (we use the convention that $\mathrm{sign}(0) = 1$). The class of homogeneous linear separators in $\mathbb{R}^n$ is defined as:

$$H_n = \{h_{\mathbf{w}} \mid \mathbf{w} \in \mathbb{R}^n, \|\mathbf{w}\|_2 = 1\}$$

Let ϵ be the target accuracy parameter. As in the case of monotone conjunctions, we will use classification error, ℓ_c, as the loss function. (Recall that when both

[5]The class of distributions where the probability density at any point in $\mathbb{R}^n$ depends only on its distance from the origin.

the representations and ideal functions are boolean, all loss functions are essentially equivalent to the classification error.)

We now describe the evolutionary mechanism. The class of representations R is the same as H_n. Each representation is a unit vector in $\mathbb{R}^n$. Next, we define the mutator, Mut, for every representation. Let $\mathbf{w}$ be the vector corresponding to representation $h_{\mathbf{w}}$. Let $\mathbf{w}, \mathbf{u}_2, \ldots, \mathbf{u}_n$ constitute an arbitrary orthonormal basis of $\mathbb{R}^n$. Then the mutator, $\mathsf{Mut}(h_{\mathbf{w}}, \epsilon)$, does the following: pick $i \in \{2, \ldots, n\}$ uniformly at random; pick $\xi \in \{-1, 1\}$ uniformly at random; set $\mathbf{w}' = \cos(\epsilon/(\pi\sqrt{n}))\mathbf{w} + \xi \sin(\epsilon/(\pi\sqrt{n}))\mathbf{u}_i$ and output the representation $h_{\mathbf{w}'}$. Note that when the orthonormal basis is fixed, there are exactly $2n-2$ possible mutations and each is output with equal probability. Define the approximation parameter, $\tau = \epsilon/(3\pi^3 n)$. Define the size function, $s : H_n \times \mathbb{R}^+ \to \mathbb{R}^+$, as $s(h_{\mathbf{w}}, \epsilon) = 2n \log(3n\pi^3/\epsilon^2)$ for every $h_{\mathbf{w}} \in H_n$. Finally, define the tolerance function, $t : H_n \times \mathbb{R}^+ \to \mathbb{R}^+$, as $t(h_{\mathbf{w}}, \epsilon) = \epsilon/(\pi^3 n)$ for every $h_{\mathbf{w}} \in H_n$. The following theorem appeared in [21].

Theorem 3.2 ([21])**.** *Let H_n be the class of homogeneous linear separators in $\mathbb{R}^n$ and let $\mathcal{D}_R$ be the class of radially symmetric distributions over $\mathbb{R}^n$. H_n is evolvable to accuracy ϵ, with respect to any distribution $D \in \mathcal{D}_R$, by the evolutionary mechanism, $\mathcal{E}M = (H_n, \mathsf{Mut}, \tau, s, t)$, described above in this section, using selection rule,* BN-Sel*, and classification error, ℓ_c, as the loss function, in g generations, where $g = 3n\pi^3/\epsilon$.*

Proof. Let $f = h_{\mathbf{w}^*}$ denote the ideal function and $D \in \mathcal{D}_R$ be the target radially symmetric distribution. Then observe that for any other representation, $h_{\mathbf{w}}$, the following holds (the observation is simple to prove, but for completeness see Dasgupta

[9]):

$$\mathrm{L}_{f,D}(h_{\mathbf{w}}) = \Pr_{x\sim D}[f(x) \neq h_{\mathbf{w}(x)}] = \Pr_{x\sim D}[\mathrm{sign}(\mathbf{w}^* \cdot x) \neq \mathrm{sign}(\mathbf{w}\cdot x)] = \frac{\arccos(\mathbf{w}^*\cdot\mathbf{w})}{\pi}$$

The proof makes frequent use of the following two trigonometric facts for any $\theta \in [0, \pi/2]$:

$$\frac{2\theta}{\pi} \leq \sin(\theta) \leq \theta \tag{3.3}$$

$$\frac{4\theta^2}{\pi^2} \leq 1 - \cos(\theta) \leq \theta^2/2 \tag{3.4}$$

Now suppose $h_{\mathbf{w}}$ is a representation such that $\mathrm{L}_{f,D}(h_{\mathbf{w}}) \geq \epsilon/2$. We consider the action of the mutator, $\mathsf{Mut}(h_{\mathbf{w}}, \epsilon)$. Let $\mathbf{w}, \mathbf{u}_2, \ldots, \mathbf{u}_n$ be the orthonormal basis used by the mutator. We show that there exists $\xi \in \{-1, 1\}$ and $i \in \{2, \ldots, n\}$ such that $\mathrm{L}_{f,D}(h_{\mathbf{w}}) - \mathrm{L}_{f,D}(h_{\mathbf{w}'}) \geq \epsilon/(\pi^3 n)$, where $\mathbf{w}' = \cos(\epsilon/(\pi\sqrt{n}))\mathbf{w} + \xi\sin(\epsilon/(\pi\sqrt{n}))\mathbf{u}_i$. Note that this particular representation, $h_{\mathbf{w}'}$ will be the output of the mutator, $\mathsf{Mut}(h_{\mathbf{w}}, \epsilon)$ with probability at least $1/(2n-2)$.

We use the fact that $\mathrm{L}_{f,D}(h_{\mathbf{w}}) = \arccos(\mathbf{w}\cdot\mathbf{w}^*)/\pi \geq \epsilon/2$. We express the vector, $\mathbf{w}^*$, in terms of the orthonormal basis, $\mathbf{w}, \mathbf{u}_2, \ldots, \mathbf{u}_n$. Suppose that,

$$\mathbf{w}^* = \lambda_1 \mathbf{w} + \sum_{j=2}^{n} \lambda_j \mathbf{u}_j$$

Let $i = \max_{j\in\{2,\ldots,n\}} |\lambda_j|$ and $\xi = \mathrm{sign}(\lambda_i)$. Then notice that $\lambda_i^2 \geq (1-\lambda_1^2)/n$ (since $\mathbf{w}^*$ has unit norm). Let $\mathbf{w}' = \cos(\epsilon/(\pi\sqrt{n}))\mathbf{w} + \xi\sin(\epsilon/(\pi\sqrt{n}))\mathbf{u}_i$. Then, consider:

$$\begin{aligned}\mathbf{w}'\cdot\mathbf{w}^* &= \lambda_1\cos\left(\frac{\epsilon}{\pi\sqrt{n}}\right) + |\lambda_i|\sin\left(\frac{\epsilon}{\pi\sqrt{n}}\right)\\ &\geq \lambda_1\cos\left(\frac{\epsilon}{\pi\sqrt{n}}\right) + \sqrt{\frac{1-\lambda_1^2}{n}}\sin\left(\frac{\epsilon}{\pi\sqrt{n}}\right) \qquad (3.5)\end{aligned}$$

We use the following trigonometric equality: Let $\alpha, \beta \in [-1, 1]$ such that $\alpha < \beta$, then

$$\arccos(\alpha) - \arccos(\beta) = \arccos\left(\alpha\beta + \sqrt{(1-\alpha^2)(1-\beta^2)}\right)$$

Then consider,

$$\begin{aligned} \arccos(\mathbf{w} \cdot \mathbf{w}^*) - \frac{\epsilon}{\pi^2 n} &= \arccos(\lambda_1) - \arccos\left(\cos\left(\frac{\epsilon}{\pi^2 n}\right)\right) \\ &= \arccos\left(\lambda_1 \cos\left(\frac{\epsilon}{\pi^2 n}\right) + \sqrt{1-\lambda_1^2}\sin\left(\frac{\epsilon}{\pi^2 n}\right)\right) \end{aligned} \tag{3.6}$$

Observe that the function $\arccos : [-1, 1] \to [0, \pi]$ is a strictly decreasing function. Thus, if we want to show that RHS in (3.6) is larger than $\arccos(\mathbf{w}' \cdot \mathbf{w}^*)$, it suffices to show that (using (3.5)):

$$\lambda_1 \cos\left(\frac{\epsilon}{\pi\sqrt{n}}\right) + \sqrt{\frac{1-\lambda_1^2}{n}}\sin\left(\frac{\epsilon}{\pi\sqrt{n}}\right) \geq \lambda_1 \cos\left(\frac{\epsilon}{\pi^2 n}\right) + \sqrt{1-\lambda_1^2}\sin\left(\frac{\epsilon}{\pi^2 n}\right)$$

After some re-arranging of terms, it suffices to show that,

$$\lambda_1\left(\cos\left(\frac{\epsilon}{\pi^2 n}\right) - \cos\left(\frac{\epsilon}{\pi\sqrt{n}}\right)\right) \leq \sqrt{1-\lambda_1^2}\left(\frac{1}{\sqrt{n}}\sin\left(\frac{\epsilon}{\pi\sqrt{n}}\right) - \sin\left(\frac{\epsilon}{\pi^2 n}\right)\right)$$

First, we make the observation that when $\lambda_1 < 0$, since $\epsilon^2/(\pi^2 n) \leq \epsilon/(\pi\sqrt{n})$, the LHS above is less than 0. The RHS on the other hand is positive using (3.3). Thus, we only concern ourselves with the case when $\lambda_1 > 0$. Notice that $\mathrm{L}_{f,D}(h_{\mathbf{w}}) = \arccos(\lambda_1)/\pi \geq \epsilon/2$, hence $\lambda_1 \leq \cos(\epsilon\pi/2)$ and $\sqrt{1-\lambda_1^2} \geq \sin(\epsilon\pi/2)$. Thus, it is sufficient to show that:

$$\cos\left(\frac{\pi\epsilon}{2}\right)\left(\cos\left(\frac{\epsilon}{\pi^2 n}\right) - \cos\left(\frac{\epsilon}{\pi\sqrt{n}}\right)\right) \leq \sin\left(\frac{\epsilon\pi}{2}\right)\left(\frac{1}{\sqrt{n}}\sin\left(\frac{\epsilon}{\pi\sqrt{n}}\right) - \sin\left(\frac{\epsilon}{\pi^2 n}\right)\right)$$

Using (3.3) and (3.4) we get:

$$\cos\left(\frac{\epsilon\pi}{2}\right)\left(\cos\left(\frac{\epsilon}{\pi^2 n}\right) - \cos\left(\frac{\epsilon}{\pi\sqrt{n}}\right)\right) \leq 1 - \cos\left(\frac{\epsilon}{\pi\sqrt{n}}\right) \leq \frac{\epsilon^2}{2\pi^2 n},$$

and

$$\sin\left(\frac{\epsilon\pi}{2}\right)\left(\frac{1}{\sqrt{n}}\sin\left(\frac{\epsilon}{\pi\sqrt{n}}\right)-\sin\left(\frac{\epsilon}{\pi^2 n}\right)\right)\geq\epsilon\left(\frac{2\epsilon}{\pi^2 n}-\frac{\epsilon}{\pi^2 n}\right)=\frac{\epsilon^2}{\pi^2 n}$$

Thus, putting everything together and using (3.6) we get:

$$\arccos(\mathbf{w}\cdot\mathbf{w}^*)-\frac{\epsilon}{\pi^2 n}\geq\arccos(\mathbf{w}'\cdot\mathbf{w}^*)$$

and hence,

$$\arccos(\mathbf{w}\cdot\mathbf{w}^*)-\arccos(\mathbf{w}'\cdot\mathbf{w}^*)\geq\frac{\epsilon}{\pi^2 n}$$

and also,

$$\mathrm{L}_{f,D}(h_{\mathbf{w}})-\mathrm{L}_{f,D}(h_{\mathbf{w}'})\geq\frac{\epsilon}{\pi^3 n}$$

To complete the proof observe the following: $\widehat{\mathrm{L}}_{f,D}(h_{\mathbf{w}'},\tau)-\widehat{\mathrm{L}}_{f,D}(h_{\mathbf{w}},\tau)\leq\mathrm{L}_{f,D}(h_{\mathbf{w}'})-\mathrm{L}_{f,D}(h_{\mathbf{w}})+2\tau\leq-\epsilon/(\pi^3 n)+2(\epsilon/3\pi^3 n)\leq-\epsilon/(3\pi^3 n)=t(h_{\mathbf{w}},\epsilon)$. Thus, the mutation, $h_{\mathbf{w}'}$, is beneficial. Since the size function, $s(h_{\mathbf{w}},\epsilon)=2n\log(3n\pi^3/\epsilon^2)$, except with probability $\epsilon^2/(3n\pi^3)$, the particular (beneficial) mutation, $h_{\mathbf{w}'}$, will be in $\mathrm{Neigh}(h_{\mathbf{w}},\epsilon)$ (obtained by running $\mathsf{Mut}(h_{\mathbf{w}},\epsilon)$ $s(h_{\mathbf{w}},\epsilon)$ times). Note that when the tolerance function is $t(h_{\mathbf{w}},\epsilon)=\epsilon/(\pi^3 n)$, for any $h_{\mathbf{w}}\in R$ and the approximation parameter, $\tau=\epsilon/(3\pi^3 n)$, any beneficial mutation must decrease the (expected) loss by $\epsilon/(3\pi^3 n)$. Thus, in at most $3\pi^3 n/\epsilon$ generations, the evolutionary mechanism will produce a representation, $h_{\bar{\mathbf{w}}}$, such that $\mathrm{L}_{f,D}(h_{\bar{\mathbf{w}}})\leq\epsilon$. Note that the probability that there is an evolutionary step (among the $3n\pi^3/\epsilon$) for which a beneficial mutation does not exist in the neighborhood is at most ϵ (by a simple union bound). $\square$

Remark 3.2. *As in the case of monotone conjunctions, the evolutionary mechanism described above also succeeds when the selection rule is* Opt-Sel *(selection based on optimization). The proof is a simple modification of the proof given above.*

3.8 Relations to Learning Theory

Valiant already observed that evolvability is more restrictive than learnability, in the sense that any concept class that is *evolvable* is also learnable in the PAC and SQ frameworks[6] [39]. Feldman observed that in fact any concept class that is *evolvable* is also learnable in the restricted CSQ (correlational statistical query) framework when the loss function is the classification error [12].

Theorem 3.3 ([39, 14])**.** *Let C be a concept class of boolean functions, i.e. $c \in C$ is such that $c : \mathcal{X} \to \mathcal{Y} = \{-1, 1\}$. Let $\mathcal{Y}' = [-1, 1]$. Suppose $\ell : \mathcal{Y}' \times \mathcal{Y} \to \mathbb{R}^+$, is a loss function that is evaluable in polynomial time and is bounded by some constant B, then if C is evolvable with respect to a class of distributions $\mathcal{D}$, any of the three selection rules* BN-Sel*,* Opt-Sel*, or* Weak-Sel*, and loss function ℓ, then C is learnable in the SQ (and hence also PAC) framework.*

Proof. The proof is the simple observation that an evolutionary mechanism can be efficiently simulated by a polynomial time algorithm with access to the statistics oracle, STAT (see Section 2.3). The only access the evolutionary process has to the target function and distribution is through obtaining (approximate) expected loss values of candidate mutations using the oracle $\widehat{\mathrm{L}}_{f,D}(\cdot, \cdot)$.

[6]Valiant's original model only used boolean functions and classification error.

For any representation, $r : \mathcal{X} \to \mathcal{Y}'$, define $\psi_r : \mathcal{X} \times \{-1, 1\} \to [-1, 1]$ as $\psi_r(x, y) = \ell(r(x), y)/B$. Note that the function, ψ_r, is efficiently computable as r and ℓ are both efficiently computable functions. Thus, the response of the statistics oracle, STAT, to the query, (ψ_r, τ), can be used as the value $\widehat{\mathrm{L}}_{f,D}(r, \tau)$. The rest of the steps required to simulate the evolutionary mechanism are straightforward, since the representations are efficiently evaluable functions, the size and tolerance functions are efficiently evaluable, and the mutator is a polynomial time Turing machine. The selection rules described in Section 3.5 can be easily implemented efficiently. $\square$

When representations are also required to be (possibly randomized) boolean functions, i.e. $\mathcal{Y}' = \mathcal{Y} = \{-1, 1\}$, and the loss function is the classification error, ℓ_c (recall that $\ell_c(y', y) = 1$ if $y' \neq y$ and 0 otherwise), any concept class, C, that is evolvable is also learnable in the more restricted CSQ framework [12].

Theorem 3.4 ([12])**.** *If C is evolvable with respect to a class of distributions $\mathcal{D}$, any of the three selection rules* $\mathsf{BN\text{-}Sel}$*,* $\mathsf{Opt\text{-}Sel}$*, or* $\mathsf{Weak\text{-}Sel}$*, and the loss function that is the classification error, ℓ_c, then C is learnable with respect to the class of distributions $\mathcal{D}$ in the CSQ framework.*

Proof. The only extra observation here is that, when the loss function is ℓ_c and the representation, r, represents a boolean function, $\mathrm{L}_{f,D}(r) = \mathrm{Pr}_{x \sim D}[r(x) \neq f(x)] = (1 - \mathbb{E}_{x \sim D}[r(x)f(x)])/2$ (in the case that r is a randomized boolean function, the probability is also taken over the randomness of r). Thus, in this case the (approximate) expected loss oracle, $\widehat{\mathrm{L}}_{f,D}(r, \tau)$, can be simulated with only access to a CSQ oracle, $\mathsf{CSQ}\text{-}\mathcal{O}$ (see Section 2.4). The rest of the proof is exactly the same as above. $\square$

3.8.1 From Learning to Evolvability

In this section, we present some results showing that in several cases, it is also possible to convert learning algorithms to evolutionary mechanisms. The first such result was shown by Feldman in the context of CSQ learning and evolvability with boolean representations and classification error as the loss function, ℓ_c [12]. Subsequently, Feldman showed that SQ learning algorithms could be simulated by evolutionary mechanisms using (a large class of) non-linear loss functions [14]. P. Valiant showed that Feldman's ideas could also be generalized to the case when the ideal functions are real-valued. He showed when the optimization problem of minimizing the expected loss over the representation class admits a (weak) optimization procedure using only (approximate) oracle access to the objective function (in this case the expected loss), such an optimization procedure can be transformed into an evolutionary mechanism [40].

For the rest of this section, we use the following notation. The ideal function, $f : \mathcal{X} \rightarrow \mathcal{Y}$, comes from some concept class, C. Let D be the target distribution over $\mathcal{X}$. We make no assumption about the sets $\mathcal{X}$ and $\mathcal{Y}$ (except that $\mathcal{Y}$ is bounded). Any representation class we use must encode functions defined from $\mathcal{X} \rightarrow \mathcal{Y}'$, where $\mathcal{Y} \subseteq \mathcal{Y}'$. $\ell : \mathcal{Y}' \times \mathcal{Y} \rightarrow \mathbb{R}^+$ is a loss function that is efficiently evaluable and bounded. We assume that for every $y \in \mathcal{Y}$, $\min_{y' \in \mathcal{Y}'} \ell(y', y) = \ell(y, y)$ and for every $y' \in \mathcal{Y}'$, $y \in \mathcal{Y}$, $\ell(y', y) \leq B$ (where B depends only on $\mathcal{Y}'$). For any candidate hypothesis, $h : \mathcal{X} \rightarrow \mathcal{Y}'$, the expected loss is defined as $\mathrm{L}_{f,D}(h) = \mathbb{E}_{x \sim D}[\ell(h(x), f(x))]$. For any h, $\mathrm{L}_{f,D}(h) \in [0, B]$.

Suppose $\mathcal{H}$ is a hypothesis space, where $C \subseteq \mathcal{H}$. We can view learning as an optimization problem, where the goal is to find some $h \in \mathcal{H}$, for which $\mathrm{L}_{f,D}(h)$ is minimized. We assume that there is a special point, $\mathbf{0} \in \mathcal{Y}'$, and that the function, $\mathbf{Z} : \mathcal{X} \to \mathcal{Y}'$, defined as $\mathbf{Z}(x) = \mathbf{0}$, for every $x \in \mathcal{X}$, is contained in $\mathcal{H}$ [7]. For reasons that will be clear in the proof of the main result in this section, it is useful to consider the objective function, $\mathsf{G}_{f,D}(\cdot)$, where $\mathsf{G}_{f,D}(h) = \mathrm{L}_{f,D}(h) - \mathrm{L}_{f,D}(\mathbf{Z})$. Minimizing $\mathsf{G}_{f,D}(\cdot)$ is equivalent to minimizing $\mathrm{L}_{f,D}(\cdot)$, since $\mathrm{L}_{f,D}(\mathbf{Z})$ is a constant. Note that for any h, $\mathsf{G}_{f,D}(h) \in [-B, B]$ (since $\mathrm{L}_{f,D}(h) \in [0, B]$).

Let $\widehat{\mathsf{G}}_{f,D}(\cdot,\cdot)$ denote an oracle that on some query (h, τ), returns a value, $\widehat{\mathsf{G}}_{f,D}(h,\tau) \in [\mathsf{G}_{f,D}(h) - \tau, \mathsf{G}_{f,D}(h) + \tau]$. We define yet another oracle, $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot,\cdot,\cdot)$, that takes a query of the form (ϕ, τ, θ), where $\phi : \mathcal{X} \to \mathcal{Y}'$, $\tau \geq 0$ and $\theta \in [-B, B]$ such that $|\theta| \geq \tau$. The output, $\widehat{\mathsf{G}}^{\leq}_{f,D}(\phi, \tau, \theta)$ is defined as follows:

$$\widehat{\mathsf{G}}^{\leq}_{f,D}(\phi,\tau,\theta) = \begin{cases} 1 & \text{if } \mathsf{G}_{f,D}(\phi) \leq \theta - \tau \\ 0 & \text{if } \mathsf{G}_{f,D}(\phi) \geq \theta + \tau \\ 1 \text{ or } 0 & \text{otherwise} \end{cases}$$

Feldman [12] proved the following simple result. The proof is a simple noisy binary search argument.

Lemma 3.1. *For any $\phi : \mathcal{X} \to \mathcal{Y}'$, $\tau \geq 0$, the oracle $\widehat{G}_{f,D}(\phi, \tau)$ can be simulated by $O(\log(B/\tau))$ queries to the oracle $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot,\cdot,\cdot)$, where each query is of the form $(\phi, \tau/2, \theta)$, where $\theta \in [-B, B]$ and $|\theta| \geq \tau/2$.*

The above lemma implies that any algorithm that uses a $\widehat{G}_{f,D}(\cdot,\cdot)$ oracle can be

[7]We may think of $\mathbf{0}$ as a randomized point in $\mathcal{Y}'$, where effectively $\mathbf{0}$ is a random variable over $\mathcal{Y}'$. For example, when $\mathcal{Y}' = \{-1, 1\}$, we think of $\mathbf{0}$ as the point that takes value -1 or $+1$ with equal probability. The function, $\mathbf{Z}$, thus defined is a randomized function.

modified to instead use a $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot,\cdot,\cdot)$ with very little extra overhead. We consider learning algorithms that use such an oracle $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot,\cdot,\cdot)$. Formally,

Definition 3.4. *Assume the notation defined in this Section. We say that some concept class, C, and be learned with respect to distribution, D, and loss function, ℓ, using a $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot,\cdot,\cdot)$ oracle, if there exists an algorithm* Alg *that for every $\epsilon > 0$, every $f \in C$, using access to a $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot,\cdot,\cdot)$ oracle, outputs a hypothesis, h, such that $\mathrm{L}_{f,D}(h) \leq \epsilon$. The running time of the algorithm is polynomial in n and $1/\epsilon$, every query (ϕ, τ, θ) made to the oracle $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot,\cdot,\cdot)$ is such that ϕ is efficiently evaluable, $1/\tau$ is bounded by a polynomial in n and $1/\epsilon$, $\theta \in [-B, B]$ and $|\theta| \geq \tau$. Also, the output hypothesis, h, is efficiently evaluable.*

The main result we present unifies the various reductions from learning algorithms to evolutionary mechanisms. The first of these was proved by Feldman [12], where he showed that CSQ algorithms could be simulated by evolutionary algorithms that use boolean representations and classification error as the loss function. Feldman generalized this to the case where the representations were real-valued and loss functions were non-linear [14]. P. Valiant showed a reduction in the case when both the ideal function and representations were real-valued [40].

Theorem 3.5 ([12, 14, 40])**.** *Any concept class, C, that is learnable with respect to distribution, D, and using loss function, ℓ, with access to a $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot,\cdot,\cdot)$ oracle, is evolvable with respect to distribution, D, using loss function, ℓ, and selection rule,* $\mathsf{BN\text{-}Sel}$.

The rest of this section is devoted to proving the above theorem. Let ϵ be the accuracy parameter. Let Alg be an algorithm that with access to oracle, $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot,\cdot,\cdot)$, learns C with respect to distribution, D. Furthermore, assume that Alg makes exactly q queries to the oracle, $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot,\cdot,\cdot)$ (if Alg makes fewer that q queries it can be forced to make frivolous queries). Let τ_{Alg} be such that every query by the Alg is of the form, $(\phi, \tau_{\mathsf{Alg}}, \theta)$, where $\phi : \mathcal{X} \rightarrow \mathcal{Y}'$, $1/\tau_{\mathsf{Alg}}$ is bounded by a polynomial in n and $1/\epsilon$, and $\tau_{\mathsf{Alg}} \leq |\theta| \leq B$. Also, let $\mathcal{H}$ be a class of hypotheses from which Alg outputs the final hypothesis, h (which satisfies $\mathrm{L}_{f,D}(h) \leq \epsilon$). We now design an evolutionary mechanism using such an algorithm Alg.

First, we construct the representation class. We will not define the entire class at once, but add representations as they are required in the proof. All representations will be randomized functions. We express a randomized function as a weighted combination of deterministic functions. Suppose $r = \sum_{i=1}^{m} w_i \psi_i$, where $\psi_i : \mathcal{X} \rightarrow \mathcal{Y}'$ is a deterministic function for every i, $w_i \geq 0$, and $\sum_{i=1}^{m} w_i \leq 1$. Notice, that we have allowed the sum of weights (probabilities) to be less than 1. The randomized function, r, is interpreted as follows. On input $x \in \mathcal{X}$, for all $1 \leq i \leq m$, with probability w_i, $r(x) = \psi_i(x)$, and with the remaining probability $1 - \sum_{i=1}^{m} w_i$, $r(x) = \mathbf{Z}(x) = \mathbf{0}$. Thus, if the weights of functions don't add up to 1, we assume that the default remaining weight is assigned to the zero function, $\mathbf{Z}$. For such a representation, r, $\mathrm{L}_{f,D}(r) = \sum_{i=1}^{m} w_i \mathrm{L}_{f,D}(\phi_i) + (1 - \sum_{i=1}^{m} w_i)\mathrm{L}_{f,D}(\mathbf{Z})$ and $\mathsf{G}_{f,D}(r) = \sum_{i=1}^{m} w_i \mathsf{G}_{f,D}(\phi_i)$ (since $\mathsf{G}_{f,D}(\mathbf{Z}) = 0$).

Let $S_q = \{z \in \{0,1\}^* \mid |z| \leq q\}$ be the set of binary strings of length at most q. For a string $z \in S_q$, we interpret the representation $r[z]$ as follows:

- If $|z| = 0$, $r[z] = \mathbf{Z}$, the zero function.

- Otherwise, let z^i denote the prefix of z of length $i - 1$. For any string z', let

$(\phi^{z'}, \tau_{\mathsf{Alg}}, \theta^{z'})$ be the $(|z'|+1)^{th}$ query that the algorithm, Alg, makes if the first $|z'|$ query responses are consistent with the string z', *i.e.* the j^{th} query response is the bit z'_j. Then, we interpret the function for representation, $r[z]$, as follows:

$$r[z] = \sum_{i: z_i = 1} \frac{\tau_{\mathsf{Alg}}}{q|\theta^{z^i}|} \phi^{z^i} \tag{3.7}$$

Note that the sum of weights of the functions in $r[z]$ adds up to at most 1. This is because $|z| \leq q$ and $|\theta^{z^i}| \geq \tau_{\mathsf{Alg}}$. (Recall that if the weight adds up to less than 1, with the remaining probability $r[z]$ is the function, $\mathbf{Z}$.)

In order to show that the representation class is polynomially bounded, first observe that the representation size of any function is at most q (which is polynomially bounded since Alg is efficient). We also need to show that given $r[z]$ and x as input, there exists a polynomial time Turing machine, that outputs $r[z](x)$. It is easy to see how to do this: given $r[z]$ and x, starting with $i = 0$, run the algorithm, Alg, until it makes a query $(\phi^{z^i}, \tau_{\mathsf{Alg}}, \theta^{z^i})$, with probability $\tau_{\mathsf{Alg}}/(|\theta^{z^i}|q)$ output $\phi^{z^i}(x)$. Otherwise, assume that the query response is z_{i+1}, increment i and continue simulation of Alg. If the value $i = |z| + 1$ is reached, output $\mathbf{Z}(x) = \mathbf{0}$.

Let $R_1 = \{r[z] \mid z \in S_q\}$ – this is part of the representation class. The final representation class will also contain the set of hypotheses, $\mathcal{H}$, as a subset. If z is a string of length q, let h_z be the hypothesis in $\mathcal{H}$ that algorithm, Alg, would output, if the query responses it received were as encoded in the string z, *i.e.* the i^{th} query response it received was z_i. We define the mutator, $\mathsf{Mut}(r[z], q)$ for representations in $r[z] \in R_1$. The parameter, η, will be defined later (it will be the case that $1/\eta$ is bounded by a polynomial in n and $1/\epsilon$):

- When $|z| = q$, then $\mathsf{Mut}(r[z], \epsilon) = h_z$ with probability $1 - \eta$ and $\mathsf{Mut}(r[z], \epsilon) = r[z]$ with the remaining probability η.

- When $|z| < q$, let $(\phi^z, \tau_{\mathsf{Alg}}, \theta^z)$ be the query Alg would make if the responses it received to the first $|z|$ queries are consistent with the string z. Then,

  1. If $\theta^z > 0$, $\mathsf{Mut}(r[z], \epsilon) = r[z1]$ with probability $1-\eta$ and $\mathsf{Mut}(r[z], \epsilon) = r[z0]$ with probability η.

  2. If $\theta^z < 0$, $\mathsf{Mut}(r[z], \epsilon) = r[z0]$ with probability $1-\eta$ and $\mathsf{Mut}(r[z], \epsilon) = r[z1]$ with probability η.

Let $\tau = \tau_{\mathsf{Alg}}^2/(2Bq)$ be the approximation parameter, $s(r[z], \epsilon) = (1/\eta)\log(1/\eta)$ be the size function and $t(r[z], \epsilon) = \tau_{\mathsf{Alg}}/q$ be the tolerance function. So let $\mathcal{E}M = (R, \mathsf{Mut}, \tau, s, t)$ be the evolutionary mechanism.

For now, we assume that the evolutionary process begins with the starting representation, $r[\sigma]$, where σ denotes the empty string. We consider the first q evolutionary steps. We prove the following claim by induction.

Claim 3.1. *After j evolutionary steps, except with probability $2\eta j$, the representation $r[z]$ is such that $|z| = j$, and for all $1 \leq i \leq j$, z_i is a valid answer to the query, $(\phi^{z^i}, \tau_{\mathsf{Alg}}, \theta^{z^i})$, made by algorithm* Alg *(recall that z^i is the prefix of z of length $i-1$).*

The claim is obviously true for $j = 0$. Assume that the claim holds for some value j. Let z be a string of length j, and $r[z]$ the representation after j evolutionary steps. We consider the representation at the $(j+1)^{th}$ evolutionary step: let $(\phi^z, \tau_{\mathsf{Alg}}, \theta^z)$ be the corresponding query that Alg would make (given previous responses as encoded in z). We consider two cases:

Case 1: ($\theta^z > 0$) The mutator, $\mathsf{Mut}(r[z], \epsilon)$ outputs $r[z1]$ with probability $1-\eta$ and $r[z0]$ with probability η. Using (3.7), we can see that $\mathrm{L}_{f,D}(r[z0]) = \mathrm{L}_{f,D}(r[z])$ (this is because when the extra bit is 0, the function represented is unchanged). When the approximation parameter, τ, is such that $2\tau \leq t(r[z], \epsilon)$, $r[z0]$ is always a neutral mutation, since $|\widehat{\mathrm{L}}_{f,D}(r[z0]) - \widehat{\mathrm{L}}_{f,D}(r[z])| \leq |\mathrm{L}_{f,D}(r[z0]) - \mathrm{L}_{f,D}(r[z])| + 2\tau \leq t(r[z], \epsilon)$. Now, if $r[z1]$ is not a deleterious mutation, the probability that $r[z1]$ will be chosen by selection rule $\mathsf{BN\text{-}Sel}$ is at least $1-\eta$.

To complete the induction step, we show the following: if 1 is an invalid answer to the query $(\phi^z, \tau_{\mathsf{Alg}}, \theta^z)$, then $r[z1]$ is a deleterious mutation. Note that the only case when 1 is an invalid answer to the query is when $\mathsf{G}_{f,D}(\phi^z) \geq \theta^z + \tau_{\mathsf{Alg}}$. Then, consider the following:

$$\mathrm{L}_{f,D}(r[z1]) - \mathrm{L}_{f,D}(r[z]) = \mathsf{G}_{f,D}(r[z1]) - \mathsf{G}_{f,D}(r[z])$$

Since $\mathsf{G}_{f,D}(\mathbf{Z}) = 0$, we get,

$$\begin{aligned} \mathrm{L}_{f,D}(r[z1]) - \mathrm{L}_{f,D}(r[z]) &= \frac{\tau_{\mathsf{Alg}}}{q|\theta^z|}\mathsf{G}_{f,D}(\phi^z) \\ &\geq \frac{\tau_{\mathsf{Alg}}}{q\theta^z}(\theta^z + \tau_{\mathsf{Alg}}) \geq \frac{\tau_{\mathsf{Alg}}}{q} + \frac{\tau_{\mathsf{Alg}}^2}{Bq} \end{aligned}$$

where the last step holds since $\theta_z \leq B$. But, this implies that $\mathrm{L}_{f,D}(r[z1]) - \mathrm{L}_{f,D}(r[z]) \geq t(r[z], \epsilon) + 2\tau$, *i.e.* $r[z1]$ is a deleterious mutation.

One last thing we need to show is that at least one copy of the mutation, $r[z0]$, is in the neighborhood, $\mathrm{Neigh}(r[z], \epsilon)$. This is because if $r[z1]$ is deleterious, without having at least one copy of $r[z0]$ in $\mathrm{Neigh}(r[z], \epsilon)$, the representation, $\perp$, would be selected and the evolutionary process would come to an end. The probability of there being no copy of $r[z0]$ in $\mathrm{Neigh}(r[z], \epsilon)$ is at most $(1-\eta)^{s(r[z],\epsilon)} \leq \eta$.

Let $r[zb]$ be the mutation selected at this evolutionary step. Then, if zb does not encode a valid response to the query $(\phi^z, \tau_{\mathsf{Alg}}, \theta^z)$ it must be because $r[z0]$ was chosen even though $r[z1]$ is neutral (this happens with probability at most η), or $r[z1]$ was deleterious and $r[z0]$ did not exist in $\mathrm{Neigh}(r[z], \epsilon)$ (also with probability at most η). Thus, the claim holds for $j + 1$.

Case 2: $\theta_j < 0$: The calculations in this case are very similar to the previous one. Thus, we only briefly describe the basic idea. The probability that $\mathsf{Mut}(r[z], \epsilon) = r[z1]$ is η and the probability that $\mathsf{Mut}(r[z], \epsilon) = r[z0]$ is $1 - \eta$. Again, $r[z0]$ is certainly a neutral mutation. Thus, unless $r[z1]$ is beneficial, $r[z0]$ will be selected with probability at least $1 - \eta$.

We show that if 0 is an invalid answer to the query $(\phi^z, \tau_{\mathsf{Alg}}, \theta^z)$, then $r[z1]$ must be beneficial. When 0 is an invalid answer, it must be the case that $\mathsf{G}_{f,D}(\phi^z) \leq \theta_z - \tau_{\mathsf{Alg}}$. Thus, we can show that $\mathrm{L}_{f,D}(r[z1]) - \mathrm{L}_{f,D}(r[z]) \leq -(\tau_{\mathsf{Alg}}/q) - (\tau_{\mathsf{Alg}}^2/Bq)$. Thus, $r[z1]$ must be a beneficial mutation. As in the previous case, the probability that there is no copy of the mutation, $r[z1]$, in $\mathrm{Neigh}(r[z1], \epsilon)$ is at most η. So except, with probability η, if $\mathsf{r}[z1]$ is beneficial, it is the mutation that will be selected. Thus, the claim holds for $j + 1$.

The above argument shows that after q evolutionary steps (generations), it must be the case that the string z, $|z| = q$, encodes the correct query responses corresponding to the simulation of algorithm, Alg. For such a representation, $r[z]$, $\mathsf{Mut}(r[z], \epsilon) = h_z$ with probability $1 - \eta$. Thus, unless h_z is a deleterious mutation, with probability at least $1 - \eta$, h_z will be selected. Note that $\mathrm{L}_{f,D}(h_z) \leq \epsilon$ (since Alg correctly learns C). So if h_z is deleterious, it must be the case that, $\mathrm{L}_{f,D}(r[z]) \leq \mathrm{L}_{f,D}(h_z) + 2\tau - t(r[z], \epsilon) \leq$

ϵ. Thus, in either case, a representation with expected loss at most ϵ is produced. The total probability of failure is at most $(2q+1)\eta$. If we set $\eta = \epsilon/(2q+1)$, the total failure probability is at most ϵ. This completes the proof for the case when the evolutionary process starts from the representation, $r[\sigma]$.

Remark 3.3. *In the proof above, we have assumed that the algorithm,* Alg*, is deterministic. If* Alg *is randomized, the first evolutionary step can be made to consist of a (neutral) mutation which encodes a random string of the length required by* Alg *into the representation. Subsequently, the simulation of* Alg *will use the random bits stored in the representation. For further details, see [12].*

Remark 3.4. *It is clear from the reduction that the selection rule,* Opt-Sel*, would also result in successful simulation of the algorithm,* Alg.

Removing the initialization requirement

In the proof above, we assumed that evolutionary mechanism is allowed to start from a fixed representation, $r[\sigma]$. Feldman [12] also showed that this is not necessary. However, this requires adding a few extra representations. Our presentation here differs slightly from Feldman's because we have described evolution in terms of losses rather than performances. Also, for Feldman's *trick* of re-initialization to work, it is important to know the value of $\mathrm{L}_{f,D}(\mathbf{Z})$ (see Remark 3.5), which we assume is at least 2ϵ [8]. Thus, $\mathrm{L}_{f,D}(\mathbf{Z}) - \epsilon \geq \epsilon$. Our presentation is brief and the reader interested in greater detail is referred to Feldman's paper [12].

[8]If this is not the case, we can always modify the mutator to output $\mathbf{Z}$ as a mutation with non-trivial probability. Note that (by slight re-scaling of ϵ) $\mathbf{Z}$ essentially is close to the ideal function. Thus, the evolutionary mechanism succeeds in essentially 1 generation.

Let $r \in R$ be a representation of the form $r[z]$ with $|z| = q$ or some $h \in \mathcal{H}$. Let $t(r, \epsilon)$ be the tolerance function and τ the approximation parameter. Let r' be the representation, $r' = (1 - (t(r, \epsilon) - 2\tau)/(\mathrm{L}_{f,D}(\mathbf{Z}) - \epsilon))r$ (we assume that $(t(r, \epsilon) - 2\tau)/(\mathrm{L}_{f,D}(\mathbf{Z}) - \epsilon) \leq 1$ – if necessary $t(r, \epsilon)$ and τ can be scaled down to make this happen, while keeping them still polynomially bounded). Then, we have,

$$\mathrm{L}_{f,D}(r') = \left(1 - \frac{t(r, \epsilon) - 2\tau}{\mathrm{L}_{f,D}(\mathbf{Z}) - \epsilon}\right) \mathrm{L}_{f,D}(r) + \frac{t(r, \epsilon) - 2\tau}{\mathrm{L}_{f,D}(\mathbf{Z}) - \epsilon} \mathrm{L}_{f,D}(\mathbf{Z})$$

Consider the following,

$$\mathrm{L}_{f,D}(r') - \mathrm{L}_{f,D}(r) = \frac{t(r, \epsilon) - 2\tau}{\mathrm{L}_{f,D}(\mathbf{Z}) - \epsilon}(\mathrm{L}_{f,D}(\mathbf{Z}) - \mathrm{L}_{f,D}(r))$$

Note that unless $\mathrm{L}_{f,D}(r) \leq \epsilon$, *i.e.* evolution has already succeeded, r' must be a neutral mutation. Let the mutator be such that $\mathsf{Mut}(r, \epsilon) = r'$, with probability $1 - \eta$, and $\mathsf{Mut}(r, \epsilon) = r$ with probability η, for some small enough value of η. If r' is neutral, the evolutionary mechanism will take r to r' with probability at least η.

Now, we define several more representations. Let $\alpha = t(r, \epsilon) - 2\tau$ and for notational convenience assume that $1/\alpha = K$ is an integer. Let $r^{(0)} = r'$. For $1 \leq k \leq K$, let $r^{(k)} = (1 - k\alpha)r'$. Note that $\mathrm{L}_{f,D}(r^{(k+1)}) - \mathrm{L}_{f,D}(r^{(k)}) = \alpha = t(r, \epsilon) - 2\tau$, thus $r^{(k+1)}$ is definitely a neutral mutation with respect to $r^{(k)}$. Define the mutator to act as $\mathsf{Mut}(r^{(k)}, \epsilon) = r^{(k+1)}$ with probability 1. Thus starting from $r^{(0)}$, in exactly K steps, the evolutionary sequence reaches the representation $r^{(K)} \equiv \mathbf{Z} \equiv r[\sigma]$. Recall that the proof above shows that starting from $r[\sigma]$, in an additional q steps, the evolutionary mechanism reaches a representation that is highly accurate. We add all such representations $r^{(k)}$ to the representation class, and note that from any such representation the evolutionary mechanism is guaranteed to succeed in at most $K + q$

steps.

Finally, suppose that the evolutionary mechanism is started from some arbitrary starting representation, r, of the form $r[z]$ for $z \in S_q$, or $h \in \mathcal{H}$, or if $r \equiv r[z]$ for $|z| < q$, then the evolutionary mechanism continues simulation of Alg as if z encoded the correct answers to the query responses, until a representation of the form $r[z]$ with $|z| = q$ is reached, also let h_z be the hypothesis in $\mathcal{H}$ that would be output if Alg received query responses as encoded in z. (Of course, this simulation may be wrong if the starting representation was not $r[\sigma]$.) Note that if $\mathrm{L}_{f,D}(r[z]) \leq \epsilon$ or $\mathrm{L}_{f,D}(h_z) \leq \epsilon$, the evolutionary mechanism has already succeeded. If not, it can *slide back* to the starting representation $r[\sigma]$ as described above and now the evolutionary mechanism starts from $r[\sigma]$ and is guaranteed to succeed by Theorem 3.5. Thus, no matter what the starting state, in at most $2q + K + 1$ steps, the evolutionary mechanism reaches a representation that has expected loss at most ϵ.

Remark 3.5. *Here we assumed that the value* $\mathrm{L}_{f,D}(\mathbf{Z})$ *is known – and typically this also means that it should be independent of the target function* f*. Below, we show that this is the case when the concept class* C *encodes boolean functions. However, even when the value* $\mathrm{L}_{f,D}(\mathbf{Z})$ *is not known, similar tricks can be used to allow* back-sliding *to* $r[\sigma]$ *to start the simulation. One possible approach is presented in Chapter 4 and another one appears in P. Valiant's paper [40].*

3.8.2 Making the representation class independent of ϵ

In the definition of evolvability, we have allowed the representation class to depend on the target accuracy, ϵ. This is not strictly necessary. Note that representations constructed in the simulation above depend on ϵ (because the queries made by Alg depend on ϵ). We may assume that ϵ is always a power of 2, because otherwise evolution may be run by setting $\epsilon' = 2^{\lfloor \log(\epsilon) \rfloor}$. The resulting accuracy is only better since $\epsilon' < \epsilon$, but the polynomial bounds are not compromised since $\epsilon' \geq \epsilon/2$.

Consider ϵ to be from the set $\{1/2, 1/4, \ldots, 2^{-n}\}$ and suppose that the representation class includes representations (as constructed above) corresponding to each of these values of ϵ. (Note that $\epsilon < 2^{-n}$ is not required, since at such low values of ϵ, the evolutionary algorithm is already allowed to use resources exponential in n.)

We observe the difficulty in this approach: Suppose ϵ is the true target parameter. Suppose ϵ' is such that the representation that was designed for ϵ' cannot be described by a string whose length is at most polynomial in n and $1/\epsilon$. This may be because, the number of queries made by Alg, $q(n, 1/\epsilon')$ may be asymptotically larger than $q(n, 1/\epsilon)$. Thus, we add the requirement that if the target accuracy of evolution is ϵ, then the starting representation must have size that is bounded by some polynomial in n and $1/\epsilon$. (See also [12]).

3.8.3 Reduction from Learning Frameworks to Evolvability

We discuss how the general reduction described above can be applied to specific learning frameworks. In particular, we describe three situations:

1. The ideal function is boolean and the representations are also (possibly randomized) boolean functions. In this situation, the only meaningful loss is the classification error, $\ell_c(y', y) = 1$ if $y' \neq y$ and $\ell_c(y', y) = 0$ if $y = y'$.

2. The ideal function is boolean but the representations may express real-valued

function in the range $[-1, 1]$.

3. Both the ideal function and the representation may be real-valued functions.

Boolean Ideal Function and Representation

In this case, $\mathcal{Y} = \mathcal{Y}' = \{-1, 1\}$. Let $\mathbf{Z} : \mathcal{X} \to \{-1, 1\}$ be the function that for every $x \in \mathcal{X}$, outputs $+1$ or -1 uniformly at random. Note, that in this case, $\mathrm{L}_{f,D}(\mathbf{Z}) = 1/2$ for every boolean function f, where expected loss is taken for the classification error.

Note that for any boolean function $\phi : \mathcal{X} \to \{-1, 1\}$, $\mathrm{L}_{f,D}(\phi) = \mathbb{E}_{x\sim D}[\ell_c(\phi(x), f(x))] = \mathbb{E}_{x\sim D}[(1-\phi(x)f(x))/2] = 1/2-(1/2)\mathbb{E}_{x\sim D}[\phi(x)f(x)]$. Note that, $\mathsf{G}_{f,D}(\phi) = \mathrm{L}_{f,D}(\phi) - \mathrm{L}_{f,D}(0) = -(1/2)\mathbb{E}_{x\sim D}[\phi(x)f(x)]$. Thus, the oracle $\widehat{G}_{f,D}(\cdot)$ is equivalent to the CSQ oracle. Thus, Theorem 3.5 implies the following result:

Theorem 3.6 ([12])**.** *Suppose a concept class, C, is learnable in the CSQ framework with respect to a class of distributions, $\mathcal{D}$, then C is evolvable with respect to the class of distributions, $\mathcal{D}$, using the classification error, ℓ_c, as the loss function and with selection rule,* $\mathsf{BN\text{-}Sel}$.

Boolean Ideal Function and Real-Valued Representations

Michael [29] first considered the generalization of evolvability to the setting where the representations are allowed to express real-valued functions. Feldman [14] showed that when the loss function satisfies the constraints listed below, any concept class that is learnable in the SQ framework is also evolvable.

Assume that $\mathcal{Y} = \{-1, 1\}$ and $\mathcal{Y}' = [-1, 1]$. Suppose $\ell : \mathcal{Y}' \times \mathcal{Y} \to \mathbb{R}^+$ is a loss function such that:

1. $\ell(1, -1) = \ell(-1, 1) = 2$ and $\ell(-1, -1) = \ell(1, 1) = 0$.
2. (Symmetric) $\ell(y, 1) = \ell(-y, -1)$.
3. (Monotone) The function $\ell(y, 1)$ is continuous and monotonically decreasing for $y \in [-1, 1]$ (and hence the function $\ell(y, -1)$ is continuous and monotonically increasing for $y \in [-1, 1]$).
4. (Non-degenerate) For every $y \in [-1, 1]$, $\ell(y, 1) + \ell(y, -1) > 0$.
5. (Non-Quasilinear) It is not the case that for every $y \in [-1, 1]$, $\ell(y, 1) + \ell(y, -1) = \ell(1, -1)$.

An obvious example of a loss that satisfies the above conditions is the squared loss $\ell(y', y) = (1/2)(y' - y)^2$. However, it is clear that a large class of loss functions satisfy the above constraints.

As in the previous case, let $\mathbf{Z} : \mathcal{X} \to [-1, 1]$ denote the randomized boolean function that on any input $x \in \mathcal{X}$ outputs 1 or -1 with equal probability. Then note that for any loss function ℓ that satisfies the condition above, it is the case that for the corresponding expected loss, $\mathrm{L}_{f,D}(\mathbf{Z}) = 1$. Thus, for any function $\phi : \mathcal{X} \to [-1, 1]$, $\mathsf{G}_{f,D}(\phi) = \mathrm{L}_{f,D}(\phi) - 1$. Feldman [14] showed that a concept class is learnable in the SQ framework if and only if it is learnable only with access to a $\widehat{G}_{f,D}(\cdot)$ oracle (for a loss function satisfying the conditions above). Thus, Theorem 3.5 implies the following theorem:

Theorem 3.7 ([14]). *Suppose a concept class, C, is learnable in the SQ framework with respect to a class of distributions, $\mathcal{D}$, then C is evolvable with respect to the class of distributions, $\mathcal{D}$, using the any loss function, ℓ, that satisfies the conditions listed above and with selection rule,* $\mathsf{BN\text{-}Sel}$.

Real-Valued Ideal Function and Representations

P. Valiant [40] introduced the notion of evolvability of real-valued functions, *i.e.* in the case when the ideal function is real-valued. Suppose the class of representations can be parametrized in some convex set $\mathcal{K} \subseteq \mathbb{R}^d$. (This is the case for example if each representation is a polynomial with bounded coefficients or a linear function with bounded coefficients.) Suppose $r \in \mathcal{K}$ and let ℓ be a loss function such that for any ideal function, $f \in C$, the expected loss, $\mathrm{L}_{f,D}(r)$, is a convex function (of $r \in \mathcal{K}$). Suppose that $\mathbf{Z}$ is also a function in $\mathcal{K}$. Then $\mathsf{G}_{f,D}(r) = \mathrm{L}_{f,D}(r) - \mathrm{L}_{f,D}(\mathbf{Z})$ is also a convex function of r. The goal of evolution is *weak-optimization*, i.e. to find a $r \in \mathcal{K}$ that is a near minimizer of $\mathsf{G}_{f,D}$, i.e. r such that $\mathsf{G}_{f,D}(r) \leq \min_{r' \in \mathcal{K}} \mathsf{G}_{f,D}(r') + \epsilon$ (or equivalently $\mathrm{L}_{f,D}(r) \leq \epsilon$, since by assumption $\min_{r' \in \mathcal{K}} \mathrm{L}_{f,D}(r') = 0$).

P. Valiant [40] observes that when $\mathsf{G}_{f,D}(\cdot)$ is convex and bounded, the weak optimization problem can often be solved only using access to the oracle, $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot,\cdot,\cdot)$, using the ellipsoid method (when certain boundedness conditions are met). In particular, he uses this to show that the class of constant-degree polynomials (with bounded coefficients) can be evolved with respect to any convex loss function (including linear loss).

Chapter 4

Drifting Targets

In this chapter, we consider the issue of stability of an evolutionary mechanism to gradual change, or *drift* in the ideal (or target) function. Such stability is a desirable property for evolutionary mechanisms, that is not explicitly captured in the definition of evolvability discussed in Chapter 3. Another property desirable in an evolutionary mechanism is *monotonicity*, i.e. the performance should not degrade during the process of evolution. In this chapter, two main results are presented. The first shows that all evolutionary mechanisms can be adapted so as to be robust with respect to a (gradually) drifting ideal function. The second shows that any evolutionary mechanism may be made *quasi-monotonic*, i.e. the performance does not degrade substantially during the process of evolution. For specific learning algorithms, we also provide rates of drift in the *ideal function* that may be tolerated by certain evolutionary mechanisms.

In this chapter, we describe three different notions of monotonicity: (i) quasi-monotonicity, where for any ϵ, the expected loss of any intermediate representation must not be greater than the loss of the starting representation, r_0, by more than ϵ, (ii) monotonicity, where the expected loss of each intermediate representation should be non-increasing (and in particular is always at most that of the expected loss of the starting representation, r_0), and (iii) strict monotonicity, where the expected loss of each intermediate representation should decrease noticeably (at least by an inverse polynomial amount) at each generation. The notion of monotonic evolution appears in Feldman's work [13] and the notion of strict monotonicity is implicit in the work of Michael [29].

We define the notion of an evolutionary mechanism as being stable to drift in the sense that for some inverse polynomial amount of drift in the ideal function, using only polynomial resources, the evolutionary mechanism will converge so that the representation has expected loss at most ϵ, and will stay at representations with such low rates of expected loss in perpetuity, in the sense that at every subsequent time-step, except with probability ϵ, the expected loss of the representation at that time will be at most ϵ. We show a general result that shows that any strictly monotonic evolutionary mechanism is automatically robust to drift in the ideal function. For two specific evolutionary mechanisms discussed in the previous chapter – those for monotonic conjunctions and homogeneous linear separators, we quantify explicitly the amount of drift that the evolutionary mechanisms can tolerate.

4.1 Notions of Monotonicity

In this section, we describe formally the three notions of monotonicity mentioned above. Feldman [13, 14] introduced the notion of monotonic evolution as stated in

Definition 4.1 below. This notion of monotonicity requires that with high probability the expected loss of the current representation, r_i, is not more than the expected loss of the initial representation r_0.

4.1.1 Notation

Let C be a concept class and $\mathcal{D}$ a class of distributions defined over $\mathcal{X}$, where each $c \in C$ is some function $c : \mathcal{X} \rightarrow \mathcal{Y}$. Let n be the size parameter associated with instances, $x \in \mathcal{X}$, *e.g.* $\mathcal{X} = \{-1, 1\}^n$ or $\mathcal{X} \subseteq \mathbb{R}^n$. Let $\mathcal{E}M = (R, \mathsf{Mut}, \tau, s, t)$ be an evolutionary mechanism consisting of a representation class, R, mutator, Mut, approximation parameter, τ, size function, s, and tolerance function, t. Let ℓ be some loss function that is consistent and bounded. Suppose that $\mathcal{E}M$ evolves C with respect to any distribution, $D \in \mathcal{D}$, and loss function, ℓ, using some selection rule, Sel. Let g be the maximum number of generations required by $\mathcal{E}M$ to evolve C, and let $r_0, r_1, \ldots, r_g$ denote the evolutionary sequence.

We define the following notions of monotonicity:

Definition 4.1 (Monotonic Evolution)**.** *An evolutionary mechanism, $\mathcal{E}M$, monotonically evolves a concept class, C, with respect to a class of distributions, $\mathcal{D}$, using loss function, ℓ, and selection rule,* Sel*, if $\mathcal{E}M$ evolves C over the class of distributions, $\mathcal{D}$, using loss function, ℓ, and selection rule,* Sel*, in g generations and furthermore with probability at least $1 - \epsilon$, for every $i \leq g$, $\mathrm{L}_{f,D}(r_i) \leq \mathrm{L}_{f,D}(r_0)$, where $r_0, \ldots, r_g$ is the evolutionary sequence.*

When explicit initialization of the starting representation, r_0, is not allowed, the definition above is equivalent to requiring that $\mathrm{L}_{f,D}(r_i) \leq \mathrm{L}_{f,D}(r_{i-1})$, for every $i \leq$

g, with high probability. In other words, it is equivalent to requiring that with high probability, the expected loss never increases during the evolutionary process. Feldman showed that if representations are allowed to express real-valued functions and the loss function, ℓ, is the squared loss, i.e. $\ell(y', y) = (1/2)(y' - y)^2$, then any class that is efficiently learnable in the statistical query (SQ) framework with respect to a fixed samplable distribution, D, is monotonically evolvable over D [13]. Feldman also showed that under a large class of non-linear loss functions (which includes the squared loss), the class of large margin halfspaces is monotonically evolvable distribution independently [15].

A stronger notion of monotonicity was used by Michael [29], in the context of real-valued representations and quadratic loss functions. He proposed an evolutionary mechanism for the class of 1-decision lists[1] which always allowed for strictly beneficial mutations at every time step. In this spirit, we define the notion of *strict monotonic evolution*, which requires a significant (inverse polynomial) decrease in the expected loss at each stage of the evolutionary process. The definition below also assumes notation defined at the beginning of this section.

Definition 4.2 (Strict Monotonic Evolution). *An evolutionary mechanism, $\mathcal{E}M$, strictly monotonically evolves a class, C, with respect to a class of distributions, $\mathcal{D}$, using loss function, ℓ, and selection rule,* Sel, *if $\mathcal{E}M$ evolves C with respect to the class of distributions, $\mathcal{D}$, using loss function, ℓ, and selection rule,* Sel, *and for some polynomial, $m(n, 1/\epsilon)$, with probability at least $1 - \epsilon$, for every $i \leq g$, either $\mathrm{L}_{f,D}(r_i) \leq \epsilon$ or $\mathrm{L}_{f,D}(r_i) \leq \mathrm{L}_{f,D}(r_{i-1}) - 1/m(n, 1/\epsilon)$, where g and $r_0, \ldots, r_g$ are as defined at the*

[1]Readers not familiar with the notion of 1-decision lists are referred to Kearns and Vazirani [23].

beginning of this section.

A notion related to that of strict monotonic evolution is that of a strictly beneficial neighborhood mutator. Informally, a mutator is a strictly beneficial neighborhood mutator if it outputs a mutation that is guaranteed to have expected loss noticeably lower than that of the starting representation, with significant probability. Formally, we define the notion of a (b, ρ)-strictly beneficial neighborhood mutator:

Definition 4.3 (Strictly Beneficial Neighbourhood Mutator)**.** *For a concept class, C, class of distributions, $\mathcal{D}$, loss function, ℓ, and representation class, R, we say that a mutator,* Mut*, is a (b, ρ)-*strictly beneficial neighborhood mutator *if the following is true for every $f \in C$ and every $D \in \mathcal{D}$: For any starting representation, r, let $\mathsf{Bene}(r) = \{r' \in R \mid \mathrm{L}_{f,D}(r') < \mathrm{L}_{f,D}(r) - b\}$. Then either, $\mathrm{L}_{f,D}(r) \leq b$ or $\Pr[\mathsf{Mut}(r, \epsilon) \in \mathsf{Bene}_r] \geq \rho$. We require that $1/b$ and $1/\rho$ are bounded by some polynomial in n and $1/\epsilon$.*

In Section 4.4, we show that the evolutionary mechanisms described in Chapter 3 – for evolving monotone conjunctions and homogeneous linear separators – have strictly beneficial neighborhood mutators. It easily follows that evolutionary mechanisms that have strictly beneficial neighborhood mutators are strictly monotonically evolvable. In Section 4.2, we show that such evolutionary mechanisms are also robust to drift in the ideal (target) function.

Finally, we define quasi-monotonic evolution. This is similar to monotonic evolution, except that the performance is allowed to go slightly below that of the starting representation, r_0. Section 4.3 shows that this notion is essentially universal, in the sense that every evolvable class is also evolvable quasi-monotonically.

Definition 4.4 (Quasi-Monotonic Evolution). *An evolutionary mechanism $\mathcal{E}M$ quasi-monotonically evolves C with respect to class of distributions $\mathcal{D}$ using selection rule* Sel *if $\mathcal{E}M$ evolves C with respect to class of distributions $\mathcal{D}$ using selection rule* Sel *and with probability at least $1-\epsilon$, for every $i \leq g(n, 1/\epsilon)$, $\mathrm{L}_{f,D}(r_i) \leq \mathrm{L}_{f,D}(r_0)+\epsilon$, where $g(n, 1/\epsilon)4$ and $r_0, r_1, \ldots,$ are as defined at the beginning of this section.*

4.2 Resistance to Drift

There are several ways one could choose to formalize the notion of resistance to drift. Our formalization is closely related to ideas from the work on tracking drifting concepts in the computational learning literature. The first models of concept drift were proposed around the same time by Helmbold and Long [19] and Kuh et al. [24]. In both these models, at each time, t, and input point, x_t, is drawn from a mixed but unknown distribution, D, and labeled according to some target function, $f_t \in C$. It is assumed that the error of f_t with respect to f_{t-1} on the distribution, D, is less than a fixed value Δ. Helmbold and Long [19] showed that a simple algorithm that chooses a concept to (approximately) minimize error over recent time-steps achieves an average error rate of $\tilde{O}(\sqrt{\Delta d})$ where d is the VC dimension of C [2]. More general models of drift have also been proposed [2, 4].

Let $f_t \in C$ denote the ideal function at time step, t, of the evolutionary process. Following Helmbold and Long [19], we make the assumption that for every t, $\Pr_{x\sim D}[f_{t-1}(x) \neq f_t(x)] \leq \Delta$ for some value of Δ. We say that a sequence

[2]Here, the notation $\tilde{O}(\cdot)$ suppresses logarithmic factors.

$f_1, f_2, \ldots, f_t, \ldots$ is a Δ-drifting[3] sequence with respect to a distribution, D, if for every t, $\Pr_{x\sim D}[f_{t-1}(x) \neq f_t(x)] \leq \Delta$. We may then define the notion of evolvability with respect to a drifting target (ideal function).

Definition 4.5 (Evolvability with Drifting Targets)**.** *For a concept class, C, distribution, D, we say that C is evolvable with* drifting targets *under distribution, D, using loss function, ℓ, and selection rule,* Sel*, by an evolutionary mechanism, $\mathcal{E}M = (R, \mathsf{Mut}, \tau, s, t)$, if there exists some polynomially bounded g, and a polynomial, $d(n, 1/\epsilon)$, such that for every $r_0 \in R$, $\epsilon > 0$, for any $\Delta \leq 1/d(n, 1/\epsilon)$, and for every Δ-drifting sequence $f_1, f_2, \ldots, f_t, \ldots$, (with each $f_t \in C$), if $r_0, r_1, \ldots$ is the evolutionary sequence (resulting from $\mathcal{E}M$ and* Sel*) then for every $l \geq g$, with probability at least $1 - \epsilon$, $\mathrm{L}_{f_l,D}(r_l) \leq \epsilon$. We refer to $d(n, 1/\epsilon)$ as the* drift polynomial.

The drift polynomial, $d(n, 1/\epsilon)$, defined above characterizes the amount of drift that can be tolerated by a specific evolutionary mechanism. Theorem 4.1 shows that whenever an evolutionary mechanism has a strictly beneficial neighborhood mutator for some concept class, C, under distribution, D, such a mechanism is robust to drift in the ideal function. Roughly speaking, the amount of drift that may be tolerated is the same as the advantage (in terms of expected loss) the beneficial mutations in the neighborhood enjoy over the starting representation.

Theorem 4.1. *For a concept class, C, distribution, D, if $\mathcal{E}M = (R, \mathsf{Mut}, \tau, s, t)$ is*

[3]Note that when f_t are all boolean functions, this is a very natural notion of drifting target (ideal) functions. In the case that f are real-valued functions, it may be possible to consider a more general notion of distance between f_{t-1} and f_t. Then, with further assumptions on the loss function such as some Lipschitz constraint on $\ell(\cdot, \cdot)$, it would be possible to obtain similar results as described here. However, in this thesis we stick to the definition of drift as described here, even for real-valued ideal functions.

an evolutionary mechanism, such that Mut *is a (b, ρ)-strictly beneficial neighborhood mutator (with respect to loss function, ℓ), then if the selection rule is* BN-Sel*, $\mathcal{E}M$ evolves C with drifting targets, under the following conditions:*

1. *$b < \epsilon/2$.*
2. *The drift rate, $\Delta \leq b/(24B)$.*
3. *The approximation parameter $\tau = b/6$.*
4. *The size function, $s(r, \epsilon) = (1/\rho)\log(8B/(b\epsilon))$, for every $r \in R$.*
5. *The tolerance function, $t(r, \epsilon) = b/2$, for every $r \in R$.*

Proof. Define $g = 8B/b$, we will show that for any Δ-drifting sequence of ideal functions, $f_0, f_1, f_2, \ldots, f_g, f_{g+1}, \ldots$, if the corresponding evolutionary sequence of representations is $r_0, r_1, \ldots, r_g, r_{g+1}, \ldots$, then for any $l \geq g$, with probability at least $1 - \epsilon$, $\mathrm{L}_{f_l,D}(r_l) \leq \epsilon$.

Note that Mut is a (b, ρ)-strictly beneficial neighborhood mutator. Let $l \geq g$. We consider the evolutionary sequence, starting from r_{l-g}. When $s(r, \epsilon) = (1/\rho)\log(8B/(b\epsilon))$, for any representation, r, except with probability $b\epsilon/(8B)$, there exists $r' \in \mathrm{Neigh}(r, \epsilon)$ (obtained by running $\mathsf{Mut}(r, \epsilon)$ $s(r, \epsilon)$ times), such that $\mathrm{L}_{f,D}(r') \leq \mathrm{L}_{f,D}(r) - b$. This holds for every possible ideal function, f.

Now suppose f and f' are such that, $\Pr_{x\sim D}[f(x) \neq f'(x)] \leq \Delta$. Then for any representation, r, we have,

$$\begin{aligned} |\mathrm{L}_{f,D}(r) - \mathrm{L}_{f',D}(r)| &= |\mathbb{E}_{x\sim D}[\ell(r(x), f(x)) - \ell(r(x), f'(x))]| \\ &\leq \Pr_{x\sim D}[f(x) \neq f'(x)]B \leq \Delta B \leq b/24 \end{aligned}$$

The last assertion holds because the quantity inside the representation makes any contribution to the expectation, only when $f'(x) \neq f(x)$. Also, $|\ell(r(x), f(x)) - \ell(r(x), f'(x))| \leq B$, since $0 \leq \ell(y', y) \leq B$ for every y', y.

Observe that, except with probability ϵ, for $8B/b$ evolutionary steps starting from representation, r_{l-g}, for any $l - g \leq i < l$, there is some r'_i in $\mathrm{Neigh}(r_i, \epsilon)$, such that $\mathrm{L}_{f_i,D}(r'_i) \leq \mathrm{L}_{f_i,D}(r_i) - b$. This follows from a simple union bound over failure probabilities at each time-step. We assume that this is the case, allowing a failure probability of ϵ.

To complete the proof, we show that for any $l - g \leq i < l$, either, $\mathrm{L}_{f_i,D}(r_i) \leq b$, or $\mathrm{L}_{f_{i+1},D}(r_{i+1}) - \mathrm{L}_{f_i,D}(r_i) \leq -b/8$.

Fix some time step, $i+1$. Let $r'_i \in \mathrm{Neigh}(r_i, \epsilon)$ such that $\mathrm{L}_{f_i,D}(r'_i) \leq \mathrm{L}_{f_i,D}(r_i) - b$. Then, it must be the case that $\widehat{\mathrm{L}}_{f_i,D}(r'_i, \tau) \leq \widehat{\mathrm{L}}_{f_i,D}(r_i, \tau) - b + 2\tau \leq -2b/3 \leq -t(r_i, \epsilon)$. Thus, the selection rule, BN-Sel, certainly picks a beneficial mutation. Let r_{i+1} be the chosen beneficial mutation, then $\mathrm{L}_{f_i,D}(r_{i+1}) \leq \mathrm{L}_{f_i,D}(r_i) - t(r_i, \epsilon) + 2\tau \leq \mathrm{L}_{f_i,D}(r_i) - b/6$. Note that this implies, $\mathrm{L}_{f_{i+1},D}(r_{i+1}) \leq \mathrm{L}_{f_i,D}(r_{i+1}) + \Delta B \leq \mathrm{L}_{f_i}(r_i) - b/6 + \Delta B \leq \mathrm{L}_{f_i,D}(r_i) - b/8$. Thus, in at most $8B/b$ generations (starting) from $l - g$ (since, obviously $\mathrm{L}_{f_{l-g},D}(r_{l-g}) \leq B$), there must be a performance with expected loss at most b.

Also, note that if r_i was such that $\mathrm{L}_{f_i,D}(r_i) \leq b$, then r_{i+1} must be at least a neutral mutation. It must then be the case that, $\mathrm{L}_{f_i,D}(r_{i+1}) \leq \mathrm{L}_{f_i,D}(r_i) + t + 2\tau \leq b + 5b/6$. Thus, $\mathrm{L}_{f_{i+1},D}(r_{i+1}) \leq b + 5b/6 + \Delta B \leq 2b$. Thus, we can see that except with probability ϵ, once there is some i in the sequence, $r_{l-g}, \ldots, r_l$, such that $\mathrm{L}_{f,D}(r_i) \leq b$, then for every subsequent r_j, $j > i$, it must be the case that $\mathrm{L}_{f,D}(r_j) \leq 2b$. As long as $b < \epsilon/2$, this completes the proof. □

Section 4.4 appeals to the above theorem to show that certain specific evolutionary mechanisms for evolving monotone conjunctions and homogeneous linear separators are robust to drift in the ideal function.

4.2.1 Universality of Drift Resistance

In this section, we prove our main result that resistance to drift is in some sense universal, *i.e.* for any class, C, that is evolvable, it is also evolvable with some (inverse polynomial) drift. We show that the reduction from learning algorithms to evolutionary mechanisms described in Section 3.8.1 can be modified to make the reduction robust to drift in ideal functions.

We recall some notation from Section 3.8.1. Let $\mathcal{X}$ be the instance space and n be the representation size parameter associated with $\mathcal{X}$ (*e.g.* $\mathcal{X} = \{-1, 1\}^n$). Let C be the target concept class, where each $c \in C$ is defined as $c : \mathcal{X} \rightarrow \mathcal{Y}$. Let $f \in C$ be the ideal function (target) and let $\ell : \mathcal{Y}' \times \mathcal{Y} \rightarrow \mathbb{R}^+$ be a consistent and bounded loss function, *i.e.* for every $y \in \mathcal{Y}$, $\min_{y' \in \mathcal{Y}'} \ell(y', y) = \ell(y, y) = 0$ and for every $y' \in \mathcal{Y}', y \in \mathcal{Y}$, $\ell(y', y) \leq B$. Let a candidate hypothesis function be defined as $h : \mathcal{X} \rightarrow \mathcal{Y}'$. Recall that the expected loss of h is $\mathrm{L}_{f,D}(h) = \mathbb{E}_{x \sim D}[\ell(h(x), f(x))]$.

As in the reduction, we assume that $\mathbf{0} \in \mathcal{Y}'$ is the *zero* (possibly randomized) point in $\mathcal{Y}'$ and let $\mathbf{Z} : \mathcal{X} \rightarrow \mathcal{Y}'$ be the function, $\mathbf{Z}(x) = 0$ for every $x \in \mathcal{X}$. For any $\phi : \mathcal{X} \rightarrow \mathcal{Y}'$, recall the function $\mathsf{G}_{f,D}(\phi) = \mathrm{L}_{f,D}(\phi) - \mathrm{L}_{f,D}(\mathbf{Z})$. Recall that the oracle, $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot, \cdot, \cdot)$, on receiving query, $(\phi, \tau_{\mathsf{Alg}}, \theta)$, where $\phi : \mathcal{X} \rightarrow \mathcal{Y}'$, $1/\tau_{\mathsf{Alg}}$ is bounded by a polynomial in n and $1/\epsilon$ and $\tau_{\mathsf{Alg}} \leq |\theta| \leq B$, responds as follows:

$$\widehat{\mathsf{G}}^{\leq}_{f,D}(\phi, \tau_{\mathsf{Alg}}, \theta) = \begin{cases} 1 & \text{if } \mathsf{G}_{f,D}(\phi) \leq \theta - \tau_{\mathsf{Alg}} \\ 0 & \text{if } \mathsf{G}_{f,D}(\phi) \geq \theta + \tau_{\mathsf{Alg}} \\ 0 \text{ or } 1 & \text{otherwise} \end{cases}$$

Suppose Alg is an algorithm that learns the concept class, C, under distribution, D, with access to a $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot, \cdot, \cdot)$ oracle, in polynomial time, such that any oracle query it makes is of the form $(\phi, \tau_{\mathsf{Alg}}, \theta)$, where $\phi : \mathcal{X} \rightarrow \mathcal{Y}'$ is an efficiently evaluable function, $1/\tau_{\mathsf{Alg}}$ is polynomially bounded (in n and $1/\epsilon$) and $\tau_{\mathsf{Alg}} \leq |\theta| \leq B$. Let q be the number of queries made by Alg and if h is the hypothesis output by Alg, then $\mathrm{L}_{f,D}(h) \leq \epsilon/2$ (note the stronger requirement on accuracy).

Our goal is to show that there exists some drift rate, Δ, such that $1/\Delta$ is bounded by a polynomial in n and $1/\epsilon$, such that there exists an evolutionary mechanism (obtained using algorithm Alg) that is robust to the ideal function drifting at a rate Δ. The proof we present assumes that the selection rule is $\mathsf{BN\text{-}Sel}$. (The proof is equally applicable when the selection rule is $\mathsf{Opt\text{-}Sel}$, but not directly for $\mathsf{Weak\text{-}Sel}$.)

For the rest of the section, we assume that the target accuracy ϵ is fixed. We define a representation class R (that depends on ϵ). The construction of the evolutionary mechanism is along the same lines as the one described in Section 3.8.1.

The high-level idea of the reduction is as follows: Let Φ denote a representation of the form that we used in Section 3.8.1 to simulate the learning algorithm, Alg. Also, let $\mathcal{H}$ denote the class of hypothesis from which the algorithm Alg outputs its hypotheses. We will use representations that are a combination of these two kinds of representations. In particular, define:

$$r' = (1 - \frac{\epsilon}{2B})h + \frac{\epsilon}{2B}\Phi, \tag{4.1}$$

where $h \in \mathcal{H}$ is a possible hypothesis and Φ is a representation of the form used in Section 3.8.1. (Note that B is the bound on the loss function.) The representation, r', is interpreted as a randomized function that behaves exactly like h with probability $1 - \epsilon/(2B)$ and like Φ with probability $\epsilon/(2B)$.

We show that when Δ (drift rate) is small enough, in the short range (approximately q generations that are required to simulate algorithm Alg), the ideal function remains essentially fixed. This is because the algorithm, Alg, only uses queries that require approximate access to some property of the (target) ideal function, in our case through the oracle, $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot,\cdot,\cdot)$. This simulation produces a hypothesis, h, that is accurate for a few generations (until the ideal function drifts substantially). This hypothesis, h, is encoded in the $(1 - \epsilon/(2B))h$ part of the hypothesis. However, the $(\epsilon/(2B))\Phi$ part of the hypothesis restarts simulation with respect to the possibly drifted ideal function. Thus, the evolutionary mechanism always *catches up* with the drifting ideal function.

Construction of Evolutionary Mechanism

We now give a formal construction of the outline described above. We refer to notation as described above. Note that the algorithm, Alg, makes q queries to the $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot,\cdot,\cdot)$ oracle and furthermore each query is of the form $(\phi, \tau_{\mathsf{Alg}}, \theta)$, where $1/\tau_{\mathsf{Alg}}$ is bounded by a polynomial in n and $1/\epsilon$, and $\tau_{\mathsf{Alg}} \leq |\theta| \leq B$. Also, recall that the final output hypothesis of algorithm, Alg, comes from some class $\mathcal{H}$.

Let $S_q = \{z \in \{0,1\}^* \mid |z| \leq q\}$ be the set of binary strings of length at most q. For some $h \in \mathcal{H}$ and $z \in S_q$, we interpret the representation[4] $r[h,z] = (1 - \epsilon/(2B))h + (\epsilon/(2B))r[z]$ as follows: When $x \in \mathcal{X}$ is received as input $r[h,z](x)$ is evaluated as –

- With probability $1 - (\epsilon/(2B))$, $r[h,z](x) = h(x)$.
- With the remaining probability, i.e. probability $\epsilon/(2B)$, $r[h,z](x) = r[z](x)$, where $r[z](x)$ is as defined in Section3.8.1, *i.e.*

$$r[z] = \sum_{i:z_i=1} \frac{\tau_{\mathsf{Alg}}}{q|\theta^{z^i}|}\phi^{z^i}$$

  , where $(\phi^{z^i}, \tau_{\mathsf{Alg}}, \theta^{z^i})$ is the i^{th} query made by Alg, if the responses to the first $i-1$ queries were as encoded in the string z^i. (Recall that z^i is the prefix of z of length $i-1$.)

The complete evolutionary mechanism also needs a few more representations, which we introduce later to maintain the clarity of presentation. We define the mutator operator, Mut, on the representations defined so far. Let $r = r[h,z] \in R$ and ϵ be given, then $\mathsf{Mut}(r[h,z], \epsilon)$ behaves as follows:

- When $|z| < q$, let $(\phi^z, \tau_{\mathsf{Alg}}, \theta^z)$ be the query that Alg makes to the $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot,\cdot,\cdot)$ oracle, if the answers to the first $j-1$ queries are as encoded in z. The parameter η used below is defined later. Then,

  1. If $\theta_j > 0$, $\mathsf{Mut}(r[h,z],\epsilon) = r[h,z1]$ with probability $1-\eta$ and $\mathsf{Mut}(r[h,z],\epsilon) = r[h,z0]$ with probability η.

[4]We abuse notation slightly to simplify presentation. Here $r[z]$ indicates the representation as used in Section 3.8.1 in the reduction from learning algorithms to evolutionary mechanisms and $r[h,z]$ is the representation of the form r' defined in (4.1) that is a randomized function combining some hypothesis $h \in \mathcal{H}$ with weight $(1-(\epsilon/2))$ and a representation of the form $r[z]$ (which is used to simulate the learning algorithm) with weight $\epsilon/2$

2. If $\theta_j < 0$, $\mathsf{Mut}(r[h,z],\epsilon) = r[h,z0]$ with probability $1-\eta$ and $\mathsf{Mut}(r[h,z],\epsilon) = r[h,z1]$ with probability η.

Let $\tau = \epsilon\tau_{\mathsf{Alg}}^2/(6q)$ be the approximation parameter, $s(r[h,z],\epsilon) = (1/\eta)\log(1/\eta)$ be the size function (as stated earlier, η, will be defined later, but in any case it is bounded by a polynomial in n and $1/\epsilon$, so that the size function is polynomially bounded), and $t(r[h,z],\epsilon) = \epsilon\tau_{\mathsf{Alg}}/(2q)$ be the tolerance function. The size and tolerance function are defined for representations of the form $r[h,z]$, where $|z| < q$.

So far we have used $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot,\cdot,\cdot)$ to denote the oracle accessed by the algorithm, Alg. However, note that this depends on the target (ideal) function. Suppose, $f_0, f_1, \ldots,$ is a Δ-drifting sequence of ideal functions. We show a simple fact that depending on the drift rate, Δ, the query response, $\widehat{\mathsf{G}}^{\leq}_{f,D}(\phi,\tau_{\mathsf{Alg}},\theta)$ does not change significantly at each evolutionary step. In particular, we show that if i and j are such that $|i-j| \leq \tau_{\mathsf{Alg}}/(4B\Delta)$, then the query response, $\widehat{\mathsf{G}}^{\leq}_{f_i,D}(\phi,\tau_{\mathsf{Alg}}/2,\theta)$, is a valid response to the query, $(\phi,\tau_{\mathsf{Alg}},\theta)$, made to the $\widehat{\mathsf{G}}^{\leq}_{f_j,D}(\cdot,\cdot,\cdot)$ oracle. (Note the subscripts on the ideal functions in the two oracles).

Lemma 4.1. *Let $f_0, f_1, f_2, \ldots,$ be a Δ-drifting sequence of ideal functions with respect to distribution, D, For any indexes, i and j, such that $|i-j| \leq \tau_{\mathsf{Alg}}/(4B\Delta)$, for any function, $\phi : \mathcal{X} \to \mathcal{Y}'$ and any θ such that $\tau_{\mathsf{Alg}}/2 \leq |\theta| \leq B$, the following hold:*

1. *If $\mathsf{G}_{f_j,D}(\phi) \leq \theta - \tau_{\mathsf{Alg}}$, then $\mathsf{G}_{f_i,D}(\phi) \leq \theta - (\tau_{\mathsf{Alg}}/2)$.*

2. *If $\mathsf{G}_{f_j,D}(\phi) \geq \theta + \tau_{\mathsf{Alg}}$, then $\mathsf{G}_{f_i,D}(\phi) \leq \theta + (\tau_{\mathsf{Alg}}/2)$.*

In particular, this shows that the query response, $\widehat{\mathsf{G}}^{\leq}_{f_i,D}(\phi,\tau_{\mathsf{Alg}}/2,\theta)$ is valid as a response to the query, $(\phi,\tau_{\mathsf{Alg}},\theta)$ made to the oracle, $\widehat{\mathsf{G}}^{\leq}_{f_j,D}(\cdot,\cdot,\cdot)$.

Proof. Assume that $i < j$. The proof in the case that $i > j$ is nearly identical, and the result is trivial when $i = j$. For any τ_{Alg} and any function $\phi : \mathcal{X} \to \mathcal{Y}'$,

$$|\mathsf{G}_{f_i,D}(\phi) - \mathsf{G}_{f_j,D}(\phi)| = |\mathrm{L}_{f_i,D}(\phi) - \mathrm{L}_{f_j,D}(\phi)| + |\mathrm{L}_{f_i,D}(\mathbf{Z}) - \mathrm{L}_{f_j,D}(\mathbf{Z})|$$

We use the fact that for any function, $\phi : \mathcal{X} \to \mathcal{Y}'$, $|\mathrm{L}_{f_i,D}(\phi) - \mathrm{L}_{f_j,D}(\phi)| \leq \Pr_{x\sim D}[f_i(x) \neq f_j(x)]B$. Thus, we have:

$$|\mathsf{G}_{f_i,D}(\phi) - \mathsf{G}_{f_j,D}(\phi)| \leq 2B \Pr_{x\sim D} \Pr[f_j(x) \neq f_i(x)]$$

Since f_i and f_j come from a Δ drifting sequence, $\Pr_{x\sim D}[f_i(x) \neq f_j(x)] \leq |j - i|\Delta$. Thus, we have:

$$|\mathsf{G}_{f_i,D}(\phi) - \mathsf{G}_{f_j,D}(\phi)| \leq 2B\Delta|j - i| \leq \tau_{\mathsf{Alg}}/2$$

Thus, if $\mathsf{G}_{f_j,D}(\phi) \geq \theta + \tau_{\mathsf{Alg}}$, $\mathsf{G}_{f_i,D}(\phi) \geq \theta + (\tau_{\mathsf{Alg}}/2)$ and if $\mathsf{G}_{f_j,D}(\phi) \leq \theta - \tau_{\mathsf{Alg}}$, then $\mathsf{G}_{f_i,D}(\phi) \leq \theta - (\tau_{\mathsf{Alg}}/2)$. □

For a string z of length q, we say that z is consistent with target function f, if for all $1 \leq i \leq |z|$, z_i is a valid query response to the query $\widehat{\mathsf{G}}^{\leq}_{f,D}(\phi^{z^{i-1}}, \tau_{\mathsf{Alg}}, \theta^{z^{i-1}})$. Consider the evolutionary mechanism, $\mathcal{E}M = (R, \mathsf{Mut}, \tau, s, t)$, and consider the evolutionary sequence of representations, $r_0, r_1, \ldots, r_q$, for the first q time steps. If $r_0 = r[h, \sigma]$, then using the proof of Theorem 3.5, we can claim that except with probability $2\eta q$, $r_q = r[h, z]$, where z_i is a valid response to the query, $(\theta^{z^i}, \tau_{\mathsf{Alg}}/2, \phi^{z^i})$, made to the oracle, $\widehat{\mathsf{G}}^{\leq}_{f_i,D}(\cdot,\cdot,\cdot)$. Lemma 4.1 implies that z_i is also a valid response to the query, $(\phi^{z^i}, \tau_{\mathsf{Alg}}, \theta^{z^i})$, made to the oracle, $\widehat{\mathsf{G}}^{\leq}_{f_q,D}(\cdot,\cdot,\cdot)$, as long as $\Delta \leq \tau_{\mathsf{Alg}}/(4Bq)$. We state this as Lemma 4.2.

Lemma 4.2. *Let $\Delta \leq \tau_{\mathsf{Alg}}/(4Bq)$, then for any Δ drifting sequence $f_0, f_1, \ldots, f_q$, if $r_0 = r[h, \sigma]$, is the starting representation, then the evolutionary mechanism, $\mathcal{E}M$, with selection rule,* BN-Sel, *with probability at least $1 - 2\eta q$, after q evolutionary steps, reaches a representation, $r_q = r[h, z]$, where z is a string of length q and z is consistent with f_q.*

Suppose that z is a string that is consistent with respect to a function f_q. Then, let h_z denote the hypothesis that Alg would output if it had received query responses as encoded in z, and furthermore it is the case that $\mathrm{L}_{f_q,D}(h_z) \leq \epsilon/2$ (by assumption). At this point, we want the evolutionary sequence to mutate from the representation, $r[h, z]$ to $r[h_z, \sigma]$, where h_z is the hypothesis output by Alg and σ is the empty string. This will allow the evolutionary mechanism to essentially restart the simulation of Alg (using the $r[\sigma]$ part of the representation). Note that $\mathrm{L}_{f_q,D}(r[h_z, \sigma]) = (1 - \epsilon/(2B))\mathrm{L}_{f_q,D}(h_z) + (\epsilon/(2B))\mathrm{L}_{f_q,D}(r[z]) \leq \epsilon$. However, it may be the case that $\mathrm{L}_{f_q,D}(r[h, z]) \leq \mathrm{L}_{f_q,D}(r[h_z, \sigma])$. Thus, we cannot simply define the mutator, $\mathsf{Mut}(r[h, z], \epsilon)$, to output $r[h_z, \sigma]$, as it may not be a beneficial mutation (or for that matter even a neutral one). We discuss below how this transition can be achieved in a small number of steps using Feldman's *backsliding* trick [12].

Restarting the Simulation

Suppose the evolutionary sequence, at time-step q, has some representation, $r[h, z]$, where z is *consistent* with respect to f_q. Ideally, we would simply define the mutator, $\mathsf{Mut}(r[h, z], \epsilon)$, to output $r[h_z, \sigma]$. But discussed above, this may be a deleterious mutation. This transition can be achieved by introducing a series of intermediate representations. These representations are along the lines of those defined in Section 3.8.1 (see section heading *Removing the initialization requirement*).

Define $r^{(0)} = r[h, z]$, where $h \in \mathcal{H}$ and $|z| = q$. Let $t = \epsilon\tau_{\mathsf{Alg}}/(2q)$ be the tolerance function, and let $K = 2/t$. For simplicity of notation, we assume that K is an integer. For $k = 1, \ldots, K$, define $r^{(k)} = (1 - kt/(2B))r^{(0)}$, *i.e.* for any $x \in \mathcal{X}$ with probability $(1 - kt/(2B))$, $r^{(k)}(x) = r^{(0)}(x)$, and with the remaining probability $w_k(x) = \mathbf{Z}(x)$. Let $R^{(*)} = \{r^{(k)} \mid r^{(0)} = r[h, z], h \in \mathcal{H}, |z| = q, k \in \{0, \ldots, K\}\}$. Note that $R^{(*)}$ implicitly depends on the representation $r[h, z]$; thus, for every such representations we add additional representations as defined in $R^{(*)}$ to the total set of representations.

We define the mutator operator, Mut, on representations in $R^{(*)}$ as follows:

- $\mathsf{Mut}(r^{(K)}, \epsilon)$ outputs $r[\mathbf{Z}, \sigma]$ with probability 1. (We assume without loss of generality that $Z \in \mathcal{H}$.)
- For $0 \leq k < K$, $\mathsf{Mut}(r^{(k)}, \epsilon)$ outputs $r^{(k+1)}$ with probability η and $r[h_z, \sigma]$ with probability $1 - \eta$.

We now show that within at most $(1/K) + 1$ generations, starting from representation $r_q \equiv r[h, z]$, we reach a representation that is either $r[\mathbf{Z}, \sigma]$ or $r[h_z, \sigma]$. Now, note that the following hold for any ideal function f:

For any function h,

$$\mathrm{L}_{f,D}(r[h, \sigma]) = \left(1 - \frac{\epsilon}{2B}\right)\mathrm{L}_{f,D}(h_z) + \frac{\epsilon}{2B}\mathrm{L}_{f,D}(\mathbf{Z})$$

For any $1 \leq k \leq K$,

$$\mathrm{L}_{f,D}(r^{(k)}) = \left(1 - \frac{kt}{2B}\right)\mathrm{L}_{f,D}(r^{(0)}) + \frac{kt}{2B}\mathrm{L}_{f,D}(\mathbf{Z})$$

In particular, $\mathrm{L}_{f,D}(r^{(K)}) = \mathrm{L}_{f,D}(\mathbf{Z})$ and $\mathrm{L}_{f,D}(r[\mathbf{Z}, \sigma]) = \mathrm{L}_{f,D}(\mathbf{Z})$. Also, observe that,

$$\mathrm{L}_{f,D}(r^{(k)}) - \mathrm{L}_{f,D}(r^{(k-1)}) = \frac{t}{2B}\mathrm{L}_{f,D}(r^{(0)}) \leq \frac{t}{2}$$

Since, $2\tau \leq t$, $r^{(k)}$ is a neutral mutation with respect to $r^{(k-1)}$, for every ideal function, f. Now, starting at $r^{(0)}$, at any subsequent time-step, $k < K$, either the representation is of the form $r[h_z, \sigma]$, in which case our claim is true. Otherwise, the representation after K steps will be $r^{(K)}$ and at the $K + 1^{th}$ step the representation will be $r[\mathbf{Z}, \sigma]$ (which is always a neutral mutation with respect to $r^{(K)}$).

Note that except with probability η, the representation $r^{(k+1)}$ will be in $\mathrm{Neigh}(r^{(k)}, \epsilon)$, when $s \geq (1/\eta)\log(1/\eta)$. Note that $r^{(k+1)}$ is always neutral (with respect to $r^{(k)}$). Suppose $r^{(k+1)}$ is indeed in $\mathrm{Neigh}(r^{(k)}, \epsilon)$ (allowing failure probability of η), then if $r[h_z, \sigma]$ is deleterious, it is chosen with probability 0, if it is neutral, with probability at least $1 - \eta$, and if it is beneficial, then with probability 1. Thus, if at some time step when the current representation is $r^{(k)}$, if at some step $r[h_z, \sigma]$ is not chosen to be the representation at the next generation, then it must be the case that $\mathrm{L}_{f',D}(r^{(k+1)}) \leq \mathrm{L}_{f',D}(r[h_z, \sigma])$ ($r[h_z, \sigma]$ must have been deleterious), where f' is the ideal function at that time-step. Thus, we can claim the following Lemma:

Lemma 4.3. *For any Δ-drifting sequence $f_0, f_1, \ldots,$ if $r_0, r_1, \ldots,$ is the sequence of representations resulting from the evolutionary mechanism, $\mathcal{E}M$, described in above, starting at $r_0 = r^{(k)}$, then except with probability $2(K - k + 1)\eta$, there exists a $j \leq K - k + 1$, such that $r_j = r[h_z, \sigma]$ or $r_j = r[\mathbf{Z}, \sigma]$. Furthermore, for all $1 \leq i < j$, $\mathrm{L}_{f_i,D}(r_i) \leq \mathrm{L}_{f_i,D}(r[h_z, \sigma])$.*

Equivalence to Evolvability with Drifting Targets

Combining these results, we prove the equivalence between evolvability and evolvability with drifting target starting from any representation in $\mathcal{R} = R \cup R^{(*)}$, where $R^{(*)}$ includes representations defined in the above section. In the proof below we assume that the value of ϵ is known. For generalization to the case when ϵ is unknown, ideas described in Section 3.8.2 can be used (also see Section 4.3.1). Theorem 4.2 shows that every concept class that is evolvable (as defined in Chapter 3) is also evolvable with drifting targets. We do this by showing that any concept class that can be learned using access to a $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot,\cdot,\cdot)$ oracle, can also be evolved with drifting targets.

Theorem 4.2. *If C is evolvable with respect to a distribution D, then C is evolvable with respect to drifting targets over distribution D.*

Proof. We will use the constructions described in this Section to prove this result. Note that Alg is an algorithm that only makes queries to the $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot,\cdot,\cdot)$ oracle, of the form $(\phi, \tau_{\mathsf{Alg}}, \theta)$, and makes exactly q queries. Also, we assume that the algorithm outputs a hypothesis, h, that has $\mathrm{L}_{f,D}(h) \leq \epsilon/2$.

Let $\mathcal{E}M$ be the evolutionary mechanism as described above. Note that the set of representations $\mathcal{R} = R \cup R^{(*)}$. Each representation $r \in R$ is of the form $r[h, z]$, where $h \in \mathcal{H}$ and $z \in S_q$. The representations in $R^{(*)}$, are the intermediate representations defined above to transition from $r[h, z]$ to $r[h_z, \sigma]$.

Let $g = 2q + K$, and let $\Delta = \min\{\epsilon/(2(q + K)), \tau_{\mathsf{Alg}}/(4Bq)\}$, where $K = 1/t = 2q/(\epsilon\tau_{\mathsf{Alg}})$. Note that the assertion of Lemma 4.2 holds for this value of Δ. We show that for any Δ-drifting sequence, $f_0, f_1, \ldots, f_l, \ldots$, if $r_0, r_1, \ldots, r_l, \ldots$ is the evolutionary sequence obtained using evolutionary mechanism, $\mathcal{E}M$, and selection rule, BN-Sel (with respect to loss function, ℓ), then for any $l \geq g$, with probability at least $1-\epsilon$, $\mathrm{L}_{f_l,D}(r_l) \leq \epsilon$.

Lemmas 4.2 and 4.3 imply that there must be some $j \leq 2q+K$, such that, r_{l-j} is of the form $r[h,\sigma]$, and consider the smallest such j as long as it is at least q. Consider the j steps after that and assume that the low probability events in the statements of Lemmas 4.2 and 4.3 do not occur (this happens with probability at least $1-2j\eta$). Then, it must be the case that either (i) $r_l = r^{(k)}$ for some $0 \leq k \leq K$ (and recall that $r^{(0)} = r[h,z]$ where z is consistent with f_{l-j+q}), or (ii) $r_l = r[h_z, z_1]$, where z_1 is some string such that $|z_1| < q$ (also, note that $\mathrm{L}_{f_{l-j+q},D}(h_z) \leq \epsilon/2$). (Note that $r_l = r[\mathbf{Z}, z_1]$ is also possible, but this only happens if $\mathrm{L}_{f_i,D}(\mathbf{Z}) \approx \mathrm{L}_{f_i,D}(h_z)$ for some $l-j+q \leq i \leq l$. We assume that this does not happen, since $\mathrm{L}_{f_{l-j+q},D}(h_z) \leq \epsilon/2$ and $\Delta \leq \epsilon/(2(q+K))$, since that would mean that $\mathbf{Z}$ is a representation with low expected loss.)

If r_l is as in (i) described above, then using Lemma 4.3, it must be the case that $\mathrm{L}_{f_l,D}(r_l) \leq \mathrm{L}_{f_l,D}(r[h_z,\epsilon])$. On the other hand, if r_l is as in (ii) above, then $\mathrm{L}_{f_l,D}(r_l) \leq (1-(\epsilon/(2B)))\mathrm{L}_{f_l,D}(h_z) + \epsilon/(2B)\mathrm{L}_{f_l,D}(r[z_1])$. Note that, $\mathrm{L}_{f_l,D}(h_z) \leq \mathrm{L}_{f_{l-j+q},D}(h_z) + \Delta|j-q| \leq \Delta(q+K)$. Thus, as long as $\Delta \leq \epsilon/(2(q+K))$, we get $\mathrm{L}_{f_l,D}(r_l) \leq \epsilon + \epsilon/(2B)$. (The required result can be obtained by re-scaling ϵ). Note that setting $\eta = 1/(4(q+K))$ ensures that the failure probability is at most ϵ. $\square$

4.3 Quasi-Monotonic Evolution

In this section, we show that any concept class that is evolvable, is also evolvable quasi-monotonically. In Theorem 4.2, we showed that after $g = 2q + K$ time-steps, for any time-step $l \geq g$, with high probability $\mathrm{L}_{f_l,D}(r_l) \leq \epsilon$. Thus, it is only necessary to show quasi-monotonicity in the initial time-horizon. For the low drift rates that we considered in Theorem 4.2, we can essentially assume that the ideal (target) function is fixed for this time horizon. Thus, we discuss the notion of quasi-monotonic evolution, only with respect to a fixed ideal function. We use the same construction as used in Section 4.2.1 with minor modifications. We still assume that the representation class depends on the (fixed) accuracy parameter ϵ. We discuss in Section 4.3.1, how this assumption may be removed using a more involved construction.

Theorem 4.3. *If C is evolvable over distribution D, then C is quasi-monotonically evolvable over D.*

Proof. We use the set-up from the proof of Theorem 4.2. We omit the parts of the argument that are identical to the one in Theorem 4.2.

Let r_0 be the starting representation. There are two possible cases:

1. $r_0 = r[\tilde{h}, \tilde{z}]$ for some $\tilde{h} \in \mathcal{H}$ and some string $\tilde{z}$, such that $0 \leq |\tilde{z}| < q$.

2. $r_0 = r^{(k)}$, for some $0 \leq k < K$, where $r^{(0)} = r[\bar{h}, \bar{z}]$, where $\bar{h} \in \mathcal{H}$ and $\bar{z}$ is a string of length q. $r^{(k)} = (1 - kt/(2B))r^{(0)}$.

First, we show that starting from r_0 as in Case 1, we can move to an r_0 as in Case 2 quasi-monotonically. This follows easily (along the lines of Theorems 3.5 and 4.2), since for any two representations, $r[h, z']$, $r[h, z'']$, satisfy:

$$|\mathrm{L}_{f,D}(r[h, z_1]) - \mathrm{L}_{f,D}(r[h, z_2])| = \frac{\epsilon}{2B}|\mathrm{L}_{f,D}(r[z]) - \mathrm{L}_{f,D}(r[z_1])| \leq \epsilon$$

Thus, starting from representation, $r_0 = r[\tilde{h}, \tilde{z}]$, in exactly $q - |\tilde{z}|$ steps the representation will be of the form $r^{(0)} = r[\tilde{h}, \tilde{z}']$, where $\tilde{z}$ is a prefix of the string $\tilde{z}'$ (which has length q). Also, each intermediate representation, r, satisfies, $\mathrm{L}_{f,D}(r) \leq \mathrm{L}_{f,D}(r_0) + \epsilon$.

Thus, we may assume instead that the starting representation is of the form $r^{(k)}$, for $0 \leq k < K$. We show that in at most $K - k + 1$ steps, a representation of the form $r[h', \sigma]$ is reached, where $h' \in \mathcal{H}$ and σ is the empty string. This is the same as Lemma 4.3. But, we show that in fact this transition is quasi-monotonic.

Let $r^{(0)} = r[\bar{h}, \bar{z}]$, $|\bar{z}| = q$. Let $r_0 = r^{(k)}$, for some $0 \leq k < K$, be the starting representation. We change the behavior of the mutator on these representations as follows:

- For $0 \leq k \leq K-1$, $\mathsf{Mut}(r^{(k)}, \epsilon)$ outputs with probability $1-\eta$, the representation $r[\bar{h}, \sigma]$. With $\eta/2$ probability it outputs $r^{(k+1)}$ and with the remaining $\eta/2$ probability $r[h_z, \sigma]$ (where h_z is the hypothesis that Alg would output when the query responses it received were consistent with those encoded in z).

- For $r^{(K)}$, $\mathsf{Mut}(r^{(K)}, \epsilon)$ outputs with probability $1 - \eta$, the representation $r[\bar{h}, \sigma]$ and with remaining probability $r[\mathbf{Z}, \sigma]$

Note that the mutator is essentially the same as used in Lemma 4.3, except that now with probability $1 - \eta$, the representation $r[\bar{h}, \sigma]$ is output. This ensures quasi-monotonicity.

The first thing we note is that except with probability $2\sqrt{\eta}$, if $\text{Neigh}(r^{(k)}, \epsilon)$ is the multi-set obtained by running the mutator, $\mathsf{Mut}(r^{(k)}, \epsilon)$ s times, then $\text{Neigh}(r^{(k)})$ will contain at least one copy of $r[h_z, \sigma]$ and at least one copy of $r^{(k+1)}$ (Note that $r^{(k+1)}$ is always neutral with respect to $r^{(k)}$.) We consider two cases:

Case 1: *$r[h_z, \sigma]$ is beneficial with respect to $r^{(k)}$ and $r[\bar{h}, \sigma]$ is not beneficial.* In this case, the selection rule, $\mathsf{BN\text{-}Sel}$, will pick $r[h_z, \sigma]$. But note that in this case, $\mathrm{L}_{f,D}(r[h_z, \sigma]) \leq \mathrm{L}_{f,D}(r[\bar{h}, \sigma])$.

Case 2: *$r[h_z, \sigma]$ and $r[\bar{h}, \sigma]$ are either both beneficial or both neutral.* In this case, except with probability, η, $r[\bar{h}, \sigma]$ will be selected.

What we can say is the following (except with probability $2\sqrt{\eta} + \eta$), if

$$r \leftarrow \mathsf{BN\text{-}Sel}(r^{(k)}, \epsilon, \widehat{\mathrm{L}}_{f,D}, \tau, s, t)$$

then, $\mathrm{L}_{f,D}(r) \leq lerr_{f,D}(r[\bar{h}, \sigma])$.

Thus, evolution reaches a representation of the form $r[h', \sigma]$. Now, consider the next $q + K + 1$ steps after that. Note that evolution is quasi-monotonic for these $q+K+1$ steps, in the sense that the expected loss of any intermediate representation does not exceed that of $r[h', \sigma]$ by more than ϵ. (Note that this means that it does not exceed by more than 2ϵ from the actual starting representation, r_0.)

But then using the analysis of Theorem 3.5 and Theorem 4.2, the representation after at most $q + K + 1$ steps (starting from $r[h', \sigma]$) reaches some r^* such that $\mathrm{L}_{f,D}(r^*) \leq \epsilon$.

Thus, by rescaling the accuracy requirements of the algorithm, Alg, and by setting $\eta = \epsilon^2/(64(q+K+1))$ (this ensures that the total failure probability is at most ϵ), we can claim that, $\mathcal{E}M$, evolves C quasi-monotonically in at most $2q+2K+1$ generations, with respect to selection rule, BN-Sel. □

4.3.1 Removing the Need to Know ϵ

We described above how, if some concept class C is evolvable with respect to a distribution D, then it is also evolvable quasi-monotonically, provided ϵ is fixed and the representation class $\mathcal{R}$ is allowed to depend on ϵ. Here, we describe how a representation class that simultaneously encodes all values of ϵ can be constructed. Note that the definition of *evolvability* allows the mutator and other operations to depend on the accuracy parameter ϵ, but the starting representation may be arbitrarily selected from the set of representations.

We assume that the accuracy parameter ϵ is always a power of 2. Were this not the case, we could simply assume that the evolutionary mechanism uses $\epsilon' = 2^{\lceil \log \epsilon \rceil}$ instead. The performance guarantees of such a mechanism would only be better since $\epsilon' \leq \epsilon$, but the polynomial bounds would not be (significantly) affected since $\epsilon' \geq \epsilon/2$. We consider the values of ϵ in the range $S_\epsilon = \{1/2, 1/4, 1/8, \ldots, 2^{-n}\}$, where n is the size parameter associated with the instance space. Note that it is unnecessary to consider values of ϵ lower than 2^{-n}, since this effectively allows the evolutionary process resources that are exponential in n. Thus, evolvability of (almost) any concept class is trivial.

We use notation defined earlier in this chapter. In particular, Alg is a learning algorithm that makes queries to a $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot,\cdot,\cdot)$ oracle – Alg makes exactly q such queries and outputs a hypothesis h, such that $\mathrm{L}_{f,D}(h) \leq \epsilon$. Let the representation $r[z]$ be as defined in Section 3.8.1. Recall that $\mathcal{H}$ is the class from which Alg outputs its final hypothesis.

Define a **term** as follows:

- Every $h \in \mathcal{H}$ is a term, and h is said to encode no value of ϵ.

- For any $\epsilon_1 \in S_\epsilon$, let T_1 be a **term** that either encodes no ϵ, or encodes only values $\epsilon' > \epsilon_1$. Then $T = (1-\epsilon_1/(2B))T_1 + (\epsilon_1/(2B))r_{\epsilon_1}[z]$ is a **term** if $|z| \leq q(n, 1/\epsilon_1)$. Furthermore, T is said to encode all values of ϵ that T_1 encodes and also the value ϵ_1.

Thus any **term** T may encode up to n values of ϵ, and the values of ϵ will increase as we get *deeper* in the **term** T. This ensures that all terms have representation size that is polynomial in n and $q(n, 1/\epsilon^*)$, where ϵ^* is the smallest value of ϵ that T encodes. Note, also that the total number of **term**s is finite.

Let $\texttt{list}(T)$ denote the list of all $\epsilon \in S_\epsilon$ that are encoded in T. Observe that the definition of term implies that the smallest $\epsilon \in \texttt{list}(T)$ is encoded at the outermost level, and the values increase as we move to the interior. In particular if ϵ_1 is the smallest value in $\texttt{list}(T)$, then $T = (1-\epsilon_1/(2B))T_1 + (\epsilon_1/(2B))r_{\epsilon_1}[z]$ for some **term** T_1 and string z, and it is the case that $\texttt{list}(T_1) = \texttt{list}(T) \setminus \{\epsilon_1\}$. Denote by $\texttt{out}(T)$ the smallest value of ϵ in $\texttt{list}(T)$ and let $\texttt{next}(T)$ be T_1 such that $T = (1 - \texttt{out}(T)/(2B))T_1 + (\texttt{out}(T)/(2B))r_{\texttt{out}(T)}[z]$ for some z.

We consider all terms except those of the form $h \in \mathcal{H}$ to be valid representations. The representation class also contains additional representations that are defined shortly. Consider the following three cases:

(a) The evolutionary process is in some representation, T, such that $\texttt{out}(T) = \epsilon$, where ϵ is the true accuracy parameter required. Then (pretending as if T

is in $\mathcal{H}$) the proof of Theorem 4.3 applies directly. In particular, let $T = r_\epsilon[T_1, z] = (1 - \epsilon/(2B))T_1 + (\epsilon/(2B))r_\epsilon[z]$. Then $\mathsf{Mut}(T, \epsilon)$ outputs either $r_\epsilon[T_1, z0]$ or $r_\epsilon[T_1, z1]$ with probabilities as defined in Section 4.2.1 if $|z| \leq q(n, 1/\epsilon)$. When $|z| = q(n, 1/\epsilon)$, additional representations of the form $R^{(*)}$ (see heading **Restarting the Simulation** from Section 4.2.1) may be defined that allow the evolutionary process in at most K steps to transition to a representation of the form $r_\epsilon[h', \sigma]$, where $h' \in \mathcal{H} \cup \{\mathbf{Z}, T_1\}$. Notice, that in the event that $h' \neq T_1$, $r_\epsilon[h', \sigma]$ is a term that only encodes one value of ϵ. Also, notice that unless $\mathrm{L}_{f,D}(T_1) \leq \epsilon$, the evolutionary process will only allow $h' = T_1$ once (as a result of partially incorrect simulation), except with a very small probability.

(b) The case when $\texttt{out}(T) < \epsilon$: Let $T_0 = T$, and define $T_i = \texttt{next}(T_{i-1})$ for all i. Let k be the smallest such that $\texttt{out}(T_k) \geq \epsilon$ (it may be the case that $T_k = h$ for some $h \in \mathcal{H}$). Note that,

$$\begin{aligned} T_0 = (1 - \epsilon_1/(2B))\,((1 - \epsilon_2/(2B))\,(\cdots \\ (1 - \epsilon_{k-1}/(2B))T_k + (\epsilon_{k-1}/(2B))r_{\epsilon_{k-1}}[z_{k-1}] \\ \cdots) + (\epsilon_2/(2B))r_{\epsilon_2}[z_2]) + (\epsilon_1/(2B))r_{\epsilon_1}[z_1] \end{aligned}$$

Then since $\epsilon_1 < \epsilon_2 < \cdots < \epsilon_{k-1} \leq \epsilon/2$, and since every ϵ_i is a power of 2,

$$\begin{aligned}
\mathrm{L}_{f,D}(T_k) &\leq \mathrm{L}_{f,D}(T_0) + \sum_{i=1}^{k-1} (\epsilon_1/(2B))\mathrm{L}_{f,D}(r_{\epsilon_i}[z_i]) \\
&\leq \mathrm{L}_{f,D}(T_0) + \frac{1}{2}\sum_{i=1}^{k-1} \epsilon_i && \text{Since } \mathrm{L}_{f,D}(r) \leq B \text{ for all } r \\
&\leq \mathrm{L}_{f,D}(T_0) + \epsilon_2
\end{aligned}$$

If $\texttt{out}(T_k) = \epsilon$, let $r_b = T_k$. Otherwise, let $r_b = (1 - (\epsilon/(2B))T_k + (\epsilon/(2B))r_\epsilon[\sigma]$. Note that r_b is always a **term** and that $\mathrm{L}_{f,D}(r_b) \leq \mathrm{L}_{f,D}(T) + \epsilon$.

(c) When $\texttt{out}(T) > \epsilon$. Let $r_c = (1 - \epsilon/(2B))T + (\epsilon/(2B))r_\epsilon[\sigma]$. Again, r_c is a valid **term** and it is easy to see that $\mathrm{L}_{f,D}(r_c) \leq \mathrm{L}_{f,D}(T) + \epsilon/2$.

In cases (b) and (c) above, if a transition to r_b or r_c respectively can be made, the evolutionary process can then proceed as in case (a). Also note that transitioning to r_b (respectively r_c) does not compromise on the quasi-monotonicity. None the less, r_b (respectively r_c) may be deleterious with respect to the starting representation T. However, we can use the by now common trick of adding new representations of the form $R^{(*)}$, that allow the evolutionary mechanism to slide from T to r_b (or r_c). Note that as discussed in Section 3.8.2, it is required to assume that the starting representation is such that its size is bounded by some polynomial in n and $1/\epsilon$.

4.4 Some Specific Evolutionary Algorithms

Finally, in this section we show that the two evolutionary mechanisms, for monotone conjunctions and homogeneous linear separators, described in Section 3.7 are robust to drift. We do this by showing that the mutators used in the evolutionary mechanisms are indeed strictly beneficial neighborhood mutators, and appeal directly to Theorem 4.1.

4.4.1 Monotone Conjunctions

Consider the evolutionary mechanism, described in Section 3.7.1, for evolving monotone conjunctions under the uniform distribution over the boolean cube, $\mathcal{X} = \{-1, 1\}^n$. Theorem 3.1 shows that for any representation, r, such that $|\mathsf{lit}(r)| \leq \log_2(3/\epsilon)$ the mutator, Mut, is an $(\epsilon^2/36, 1/(4n^2))$ strictly beneficial neighborhood mutator. Notice that if the starting representation, r_0, does not satisfy $|\mathsf{lit}(r_0)| \leq \log_2(3/\epsilon)$ after at most $n - \log_2(3/\epsilon)$ such a representation is reached. Also, after that stage, the evolutionary mechanism never produces a representation, r', for which $|\mathsf{lit}(r'| > \log_2(3/\epsilon)$. Thus, appealing to Theorem 4.1, we may prove the following result:

Theorem 4.4. *Let C denote the class of monotone conjunctions on the boolean cube, $\mathcal{X} = \{-1, 1\}^n$ and let $\mathcal{U}$ denote the uniform distribution over $\mathcal{X}$. Then C is evolvable using evolutionary mechanism, $\mathcal{E}M = (R, \mathsf{Mut}, \tau, s, t)$ (as described in Section 3.7.1), using selection rule* $\mathsf{BN\text{-}Sel}$*, with respect to distribution $\mathcal{U}$, with drifting targets for any drift rate, $\Delta \leq \epsilon^2/864$.*

4.4.2 Homogeneous Linear Separators

Consider the evolutionary mechanism described in Section 3.7.2 for evolving homogeneous linear separators in $\mathbb{R}^n$ under radially symmetric distributions. The proof of Theorem 3.2 essentially proves that the mutator operator defined there is an $(\epsilon/(\pi^3 n), 1/(2n-2))$ strictly beneficial neighborhood mutator. Thus, appealing to Theorem 4.1, we can prove the following result:

Theorem 4.5. *Let H_n be the class of homogeneous linear separators in $\mathbb{R}^n$ and D be any radially symmetric distribution over $\mathbb{R}^n$. Then H_n is evolvable under drifting target (ideal) functions with respect to distribution D, using the evolutionary mechanism, $\mathcal{E}M = (R, \mathsf{Mut}, \tau, s, t)$ (as described in Section 3.7.2), for drift rate, $\Delta \leq \epsilon/(24\pi^3 n)$.*

Chapter 5

The Role of Recombination

One of the most important aspects of the biological world not modeled explicitly by Valiant is the existence of two sexes and the process of recombination. Sexual reproduction is nearly universal in higher organisms and thus is thought to be an important factor in evolution. There are several proposed explanations for the role of sex and recombination in evolution. Some of these are discussed in Section 5.1, but the most relevant argument for our work is the one that sexual reproduction can accelerate evolution through parallelism. Fisher [18] first proposed that sexual reproduction can speed up evolution (see also [30, 34]):

> A consequence of sexual reproduction which seems to be of fundamental importance to evolutionary theory is that advantageous changes in different structural elements of the germ plasm can be taken advantage of independently; whereas with asexual organisms either genetic uniformity of the whole group must be such that evolutionary progress is greatly retarded, or if there is considerable genetic diversity, many beneficial changes will be lost through occurring in individuals destined to leave no ultimate descendants in the species.
>
> (from The Genetical Theory of Natural Selection - R. A. Fisher 1930)

A simple explanation for this is the following: Suppose that there are two allelic mutation $a \rightarrow A$ and $b \rightarrow B$ that are both favorable and also additive in their effect. Thus, an individual having both A and B alleles will be the fittest. However, beneficial mutations are extremely rare in nature. Under asexual reproduction an individual would possess both alleles A and B, only if a mutation, say $b \rightarrow B$, occurs in an individual already possessing allele A. For this to occur with high probability, the A allele must have already spread in the population. For a large fraction of the population to acquire both A and B, several generations may be required. Under sexual reproduction, even if the two mutations $a \rightarrow A$ and $b \rightarrow B$ occur in different members of the population, they are able to spread quickly in the population and via recombination there will be a member with both mutations in much fewer generations. Roughly speaking, if there are n loci on which selection acts additively, asexual reproduction may require $O(n)$ generations to produce the fittest variant, while sexual reproduction is able to achieve the same in only $O(\log(n))$. Our main result shows that recombination allows for *parallelism*, this result is discussed in Section 5.4. We show that the reduction from learning algorithms to evolutionary mechanisms described in Chapter 3 can be modified to be such that if there exists an efficient parallel algorithm for learning a concept class, C, then there is an evolutionary mechanism that exploits the parallelism to achieve faster evolution in a model of evolution with recombination. It can be shown that these evolutionary mechanisms are provably faster than those possible without recombination in Valiant's original model. This is shown in Section 5.6.

5.1 Related Work

Several explanations have been proposed to understand the factors responsible for maintaining sex and recombination. However, there seems to be no single answer. We have discussed above Fisher's argument that sexual reproduction may accelerate evolution. Another advantage, proposed by Muller [30], is that recombination is useful as a means to weed out deleterious mutations. In a finite population, mildly deleterious mutations are likely to be fixed in the population one at a time merely by chance and eventually a larger number of these will be accumulated. In the absence of recombination, a deleterious mutation cannot be removed except by a back-mutation which is extremely rare. This effect is known as Muller's ratchet.

Epistasis refers to the non-independence between loci with respect to their effect on fitness. Hitchhiking is the phenomenon where certain (possibly deleterious) alleles are maintained because they are coupled to other beneficial mutations. Epistasis, hitchhiking and other factors are thought to play a role in the maintenance of recombination. Livnat et al. [26, 27] have also suggested that sexual reproduction gives rise to mixability or modularity. Maynard Smith [35, 36] and Barton and Charlesworth [3] survey several proposed explanations of sex and recombination.

5.2 Evolution Model with Recombination

In this work, the question of interest is whether evolution can be sped up in Valiant's computational model. Thus, Fisher's ideas are most relevant to this work, and indeed this work is inspired by his. In order to model the process of recombination, it is necessary to consider a finite population with variation, whereas in Valiant's model all members of the population were considered identical. The model considered here is inspired by those in population genetics, particularly the Wright-Fisher model [18, 41]. As in the Wright-Fisher model, we have discrete generations and a fixed population in each generation. The members of each generation choose their parents randomly from the previous generation[1]. However, unlike the models in population genetics, the individuals in our population are representations of functions (rather than abstract entities). The goal as in Valiant's model is to evolve a representation that predicts an *unknown* ideal function (almost) accurately using feedback obtained through (natural) selection.

We briefly recall some notions from Chapter 3. The input space, $\mathcal{X}$, is thought to model all possible environments that may be faced by an organism; typical settings of interest are $\mathcal{X} = \{-1, 1\}^n$ and $\mathcal{X} \subseteq \mathbb{R}^n$. The parameter, n, represents the size of instances $x \in \mathcal{X}$. The *ideal function* is defined as a function, $f : \mathcal{X} \rightarrow \mathcal{Y}$, and indicates the best behavior in every possible environmental condition. The ideal function comes from some concept class, C, of functions defined from $\mathcal{X} \rightarrow \mathcal{Y}$.

Recall that an organism is treated as a string that *represents* a function from $\mathcal{X} \rightarrow \mathcal{Y}'$, such that $\mathcal{Y} \subseteq \mathcal{Y}'$. The requirement is that given the string representation, r, and some input, $x \in \mathcal{X}$, there is a polynomial time Turing machine that outputs the value, $r(x) \in \mathcal{Y}'$. Thus, by slight abuse of notation, the representation, r, is treated as a string and also a function from $\mathcal{X} \rightarrow \mathcal{Y}'$. Let R be a class of representations of functions $\mathcal{X} \rightarrow \mathcal{Y}'$.

[1]However, the term generation in our model does not necessarily refer to one biological generation. This is discussed further in Section 5.7.

The performance of an organism is measured with respect to a loss function $\ell : \mathcal{Y}' \times \mathcal{Y} \rightarrow \mathbb{R}^+$, such that for every $y \in \mathcal{Y}$, $\min_{y' \in \mathcal{Y}'} \ell(y', y) = \ell(y, y) = 0$, and for every $y' \in \mathcal{Y}', y \in \mathcal{Y}$, $\ell(y', y) \leq B$, *i.e.* ℓ is consistent and bounded. Suppose that D is target distribution over the instance space $\mathcal{X}$; then for any representation $r : \mathcal{X} \rightarrow \mathcal{Y}'$, the *expected loss* is defined as $\mathrm{L}_{f,D}(r) = \mathbb{E}_{x \sim D}[\ell(r(x), f(x))]$. Alternatively, one may view the performance of r as $\ell\text{-Perf}_{f,D}(r) = 1 - 2\mathrm{L}_{f,D}(r)/B$.

5.2.1 Recombination

Recombination is a process that takes two genotypes and produces a new one combining the two. The exact process of recombination is not completely understood, and modeling such a process seems inherently problematic. Following Valiant's idea of defining mutations as output of a randomized polynomial time Turing machine, we can define recombination in a similar fashion.

A *recombinator* is defined as a randomized polynomial time Turing machine that takes as input two representations r', r'' and the target loss parameter ϵ and outputs a representation r. Formally,

Definition 5.1 (Recombinator). . *A recombinator for a representation class, R, is a randomized Turing machine,* Rec*, that takes as inputs, $r', r'' \in R$ and ϵ, and outputs a representation $r \in R \cup \{\perp\}$. Th running time of* Rec *is bounded by a polynomial in $|r'| + |r''|$ and $1/\epsilon$. We think of* $\mathsf{Rec}(r', r'', \epsilon)$ *as a random variable (random descendant or recombinant of r' and r'') that takes values in R. If the recombinator outputs $\perp$ that is considered as r' and r'' being unable to leave a descendant in the next generation.*

5.2.2 Selection Rules

We define selection rules similarly to those defined in Section 3.5. Let $r', r'' \in R$ be two representations and let ϵ be the accuracy parameter. Recall from Section 3.5, that $\widehat{\mathrm{L}}_{f,D}(\cdot,\cdot)$ is an oracle that when queried (r, τ), returns an additive approximate value of the expected loss of representation, r, *i.e.* $|\widehat{\mathrm{L}}_{f,D}(r,\tau) - \mathrm{L}_{f,D}(r)| \leq \tau$. Let t be the tolerance parameter and $s \in \mathbb{N}$ the size parameter, then we can define the selection of one *descendant* of r' and r'' as follows:

- Let $\mathrm{Desc}(r', r'', \epsilon) = \{r_1, \ldots, r_s\}$ be a multi-set obtained by running $\mathsf{Rec}(r', r'', \epsilon)$ s times. This is the candidate pool of possible descendants (recombinants) of r' and r''. Let $v_i = \widehat{\mathrm{L}}_{f,D}(r_i, \tau)$. Let $v' = \widehat{\mathrm{L}}_{f,D}(r', \tau)$ and $v'' = \widehat{\mathrm{L}}_{f,D}(r'', \tau)$.

- We consider the three possible selection rules – selection based on beneficial and neutral recombinants (BN-Sel), selection based on optimization (Opt-Sel) and weak selection (Weak-Sel), to select *one* descendant that will survive to live in the next generation.

  1. *Selection based on beneficial and neutral recombinants*: Beneficial recombinants (descendants) are defined as those whose expected loss is noticeably lower than *at least one* of its two parents – r' and r''. Neutral recombinants (descendants) are those whose expected loss is close to that of the (worse) of its two parents. Thus,

$$\mathsf{Bene} = \{r_i \in \mathrm{Desc}(r', r'', \epsilon) \mid v_i \leq \max(v', v'') - t\}$$

$$\mathsf{Neut} = \{r_i \in \mathrm{Desc}(r_1, r_2, \epsilon) \mid |v_i - \max(v', v'')| \leq t\}$$

Selection proceeds as follows: if the multi-set Bene is not empty, then a representation from Bene is chosen to be a *descendant* uniformly at random; if Bene is empty and Neut is non-empty, a representation from Neut is selected uniformly at random; if both Bene and Neut are empty, then $\perp$ is selected, and it is understood as r', r'' leaving no descendant in the next generation. Let r denote the representation *selected* by such a process, we express this as:

$$r \leftarrow \mathsf{BN\text{-}Sel}(r', r'', \epsilon, \mathsf{Rec}, \widehat{\mathrm{L}}_{f,D}, \tau, s, t)$$

2. *Selection based on optimization*: This selection rule chooses one among the *best* possible candidate recombinants. Let $v^* = \min_{i=1}^{s} v_i$. Then, if $v^* \geq \max(v', v'') + t$, *i.e.* the best descendant is noticeably worse than the worse of the two parents, $\perp$, is selected and it is understood as r' and r'' leaving no descendants in the next generation. Otherwise, define:

$$\mathsf{Opt} = \{r_i \in \mathrm{Desc}(r', r'', \epsilon) \mid |v_i - v^*| \leq t \text{ and } v_i \leq \max(v', v'') + t\}$$

Note that when $v^* \leq \max(v', v'') + t$, Opt is guaranteed to be non-empty. The selection rule chooses a random descendant, r, from Opt. We express this selection rule as:

$$r \leftarrow \mathsf{Opt\text{-}Sel}(r', r'', \epsilon, \mathsf{Rec}, \widehat{\mathrm{L}}_{f,D}, \tau, s, t).$$

3. *Weak Selection*: In this selection rule, a distinction is not made between beneficial and neutral recombinants (descendants). Any recombinant that is not noticeably worse than the worse among its two parents is considered

feasible. Define the multi-set, Feas, as:

$$\mathsf{Feas} = \{r_i \in \mathrm{Desc}(r', r'', \epsilon) \mid v_i \leq \max(v', v'') + t\}$$

Selection proceeds as follows: if Feas is not empty, a representation, r, from multi-set Feas is selected uniformly at random; otherwise it is assumed that r' and r'' are unable to leave a descendant to the next generation. We express this selection process as:

$$r \leftarrow \mathsf{Weak\text{-}Sel}(r', r'', \epsilon, \mathsf{Rec}, \widehat{\mathrm{L}}_{f,D}, \tau, s, t).$$

In this chapter, we will consider all three selection rules described above. The case for weak selection is particularly compelling, since as we discuss in Section 5.7, recombination is most likely to play a role in accelerating evolution when selection pressures are weak.

5.2.3 Evolution with Recombination

An evolutionary mechanism (with recombination) may be defined as the 5-tuple, $\mathcal{E}M = (R, \mathsf{Rec}, \tau, s, t)$. Here, R is the representation class, Rec the recombinator operator, τ the approximation parameter (that is used to make queries to get estimates of expected loss from oracle $\widehat{\mathrm{L}}_{f,D}(\cdot,\cdot)$), $s : R \times R \times \mathbb{R}^+ \to \mathbb{N}$ is the size function (here the size function depends on two representations, r', r'', to define the size of the multi-set $\mathrm{Desc}(r', r'', \epsilon)$), and similarly the tolerance function, $t : R \times R \times \mathbb{R}^+ \to \mathbb{R}^+$. As defined in Section 3.6, we say that the evolutionary mechanism, $\mathcal{E}M$, is polynomially bounded, if the representation class, R, is polynomially bounded, the recombinator, Rec, is polynomially bounded, the approximation parameter, τ, is such that $1/\tau$ is bounded by a polynomial in n and $1/\epsilon$, the size function, s, is polynomially bounded and the tolerance function, t, is polynomially sandwiched. It is also required that the size function, s, and tolerance function, t, are polynomially evaluable.

Valiant's original model did not require a *diverse* population. Indeed in an evolutionary process in the sense of Valiant [39], at each generation only a single genotype was preserved and changes across generations were caused solely due to mutation. For recombination to influence evolution in any way, variation in population at each generation is a must. In this chapter, we assume the existence of a finite population and if its size is bounded by a polynomial (in n and $1/\epsilon$), we consider the population size to be *reasonable*.

Define a *population* to be a subset of the set of representations, R. We define an *evolutionary step* as taking a population, $P_0 \subseteq R$, to population, $P_1 \subseteq R$, at the next generation, using some evolutionary mechanism, $\mathcal{E}M$, and selection rule, Sel. This transition evolves the action of the recombinator operator on existing representations (genotypes) in P_0, and then selection using rule, Sel. The population P_1 is produced as follows: Each member of P_1 is picked by picking parents r' and r'' uniformly at random from P_0 (with replacement). This is followed by selection according selection rule, Sel, as described above to possibly produce a *descendant*, r. Note that it is possible for r' and r'' to produce no viable descendant [2]. This process is repeated until $|P_1| = |P_0|$. Formally,

Definition 5.2 (Evolutionary Step)**.** *An* evolutionary step *using evolutionary mech-*

[2]One can imagine imposing several kinds of restriction on mating – being of the same sex, being geographically distant, or other ones. We believe that using string representations is general enough, so any such information can be effectively encoded in the recombinator Turing machine, forcing it to output $\perp$ when the mating is infeasible.

anism, $\mathcal{E}M = (R, \mathsf{Rec}, \tau, s, t)$, and selection rule, Sel, (with respect to loss function, ℓ, ideal function, f, and target distribution, D,) beginning with a starting population, $P_0 \subseteq R$, produces a population, $P_1 \subseteq R$, for the next generation as follows:

Let $P_1 = \emptyset$. While $|P_1| < |P_0|$:

1. *Select uniformly at random (with replacement) parents $r', r'' \in P_0$*
2. *Let $r \leftarrow \mathsf{Sel}(r', r'', \epsilon, \mathsf{Rec}, \widehat{\mathrm{L}}_{f,D}, \tau, s, t)$*
3. *If $r \neq \perp$, then $P_1 = P_1 \cup \{r\}$*

Observe that as long as the population size $|P_0|$ is polynomially bounded, an evolutionary step can be simulated in polynomial time (with high probability). Starting from some starting population, $P_0 \subseteq R$, we can consider a sequence of evolutionary steps that result from evolutionary mechanism, $\mathcal{E}M$, and selection rule, Sel. Denote the resulting sequence of populations as $P_0, P_1, P_2, \ldots$, where P_{i+1} results from P_i by an evolutionary step. We define the notion of evolvability as requiring that for some (polynomially bounded) time step, g, there exist a representation, $r \in P_g$, such that $\mathrm{L}_{f,D}(r) \leq \epsilon$. Formally,

Definition 5.3 (Evolvability with Recombination)**.** *We say that a concept class, C, is evolvable with respect to distribution, D over $\mathcal{X}$, using loss function, ℓ, and selection rule, Sel, in g generations, if there exists a polynomially bounded evolutionary mechanism, $\mathcal{E}M = (R, \mathsf{Rec}, \tau, s, t)$, that for every $\epsilon > 0$, starting from any (polynomially bounded) population, $P_0 \subseteq R$, and for every ideal function, $f \in C$, with selection rule, Sel, and oracle, $\widehat{\mathrm{L}}_{f,D}(\cdot,\cdot)$, produces a sequence of populations, $P_0, P_1, \ldots, P_g$,* *such that with probability at least $1 - \epsilon$ for some $g' \leq g$, there exists a representation $r^* \in P_{g'}$ such that $\mathrm{L}_{f,D}(r^*) \leq \epsilon$.*

We also consider the case where the evolutionary process succeeds only if it starts from a specific initial population, P_0. We refer to this as *evolution with recombination with initialization.*

Readers familiar with the Wright-Fisher model from the population genetics will notice the similarity between their model and the one presented here. However, while population genetics is concerned mainly with allele frequency distributions, the focus of our work is on understanding the computational limits of evolution. A natural question that arises is, does augmenting Valiant's model of evolution with recombination increase the power of evolvability? In the sense of polynomially bounded evolution the answer is negative. This follows from the reduction from learning algorithms to evolvability as described in Section 3.8.1 (also see the original result due to Feldman [12]). This is because a polynomially bounded evolutionary mechanism (with recombination) can be simulated by a polynomial time algorithm that only makes queries to the oracle, $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot,\cdot,\cdot)$. Theorem 3.5 shows that every such algorithm can be simulated by a polynomially bounded evolutionary mechanism in Valiant's model (without recombination).

However, we show that evolvability with recombination may be exponentially faster for some concept classes under certain distributions; this follows via a reduction from parallel learning algorithms to evolution with recombination. The reduction described here requires *initialization*. In Section 5.3, we first formally define a model of parallel learning that is convenient to prove the main result. In Section 5.4, we present the reduction from parallel learning algorithms to evolution with recombination. In Section 5.5, we consider specific concept classes that can be evolved substantially faster with recombination. Finally, in Section 5.6 we show a lower bound that evolution in Valiant's model (without recombination) may indeed be exponentially slower in certain cases.

The algorithms presented here may appear somewhat unnatural; however, the goal is not to model exact physical processes, but to understand their computational limitations. This model can be understood as specifying the outer limits of the evolutionary process, as we do not expect nature to perform computations not captured by efficient Turing machines.

5.3 Model of Parallel Learning

Recall some notation defined in Section 3.8.1. Let C be the concept class, $f \in C$ the ideal function, $f : \mathcal{X} \to \mathcal{Y}$ and let D be a distribution over $\mathcal{X}$. Let $h : \mathcal{X} \to \mathcal{Y}'$ be a hypothesis such that $\mathcal{Y} \subseteq \mathcal{Y}'$. We consider a loss function, $\ell : \mathcal{Y}' \times \mathcal{Y} \to \mathbb{R}^+$, that is consistent and bounded, *i.e.* for every $y \in \mathcal{Y}$, $\min_{y' \in \mathcal{Y}'} = \ell(y, y)$ and for every $y' \in \mathcal{Y}', y \in \mathcal{Y}$, $\ell(y', y) \leq B$. The expected loss of hypothesis, h, with respect to ideal function, f, and distribution, D, is $\mathrm{L}_{f,D} = \mathbb{E}_{x \sim D}[\ell(h(x), f(x))]$.

We have supposed that $\mathbf{0} \in \mathcal{Y}'$ and the function, $\mathbf{Z} : \mathcal{X} \to \mathcal{Y}'$, defined as $\mathbf{Z}(x) = \mathbf{0}$, for every $x \in \mathcal{X}$ is the *zero* function[3]. Recall the (translated loss) function, $\mathsf{G}_{f,D}(h) = \mathrm{L}_{f,D}(h) - \mathrm{L}_{f,D}(\mathbf{Z})$, and the oracle, $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot, \cdot, \cdot)$, that on query (ϕ, τ, θ),

[3]We may think of $\mathbf{0}$ as a randomized point in $\mathcal{Y}'$, where effectively $\mathbf{0}$ is a random variable over $\mathcal{Y}'$. For example, when $\mathcal{Y}' = \{-1, 1\}$, we think of $\mathbf{0}$ as the point that takes value -1 or $+1$ with equal probability. The function, $\mathbf{Z}$, thus defined becomes a randomized function.

where $\phi : \mathcal{X} \to \mathcal{Y}'$, $\tau \in \mathbb{R}^+$, $\theta \in \mathbb{R}$ such that $\tau \leq |\theta| \leq B$, outputs a value as defined below:

$$\widehat{\mathsf{G}}^{\leq}_{f,D}(\phi, \tau, \theta) = \begin{cases} 1 & \text{if } \mathsf{G}_{f,D}(\phi) \leq \theta - \tau \\ 0 & \text{if } \mathsf{G}_{f,D}(\phi) \geq \theta + \tau \\ 0 \text{ or } 1 & \text{otherwise} \end{cases}$$

We considered algorithms that learn concept class, C, with respect to distribution, D, using only access to the $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot, \cdot, \cdot)$ oracle. We now extend this to describe *parallel* learning algorithms that only access the $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot, \cdot, \cdot)$ oracle.

We consider a parallel algorithm that uses p (polynomially bounded) processors and we assume that there is a common clock that defines parallel time steps. During each parallel time step a processor can do the following:

1. Make a query to the oracle $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot, \cdot, \cdot)$.

2. Perform some polynomially-bounded computation.

3. Write a message that can be read by *every* other processor at the end of the current time step. Effectively, we view this as a message *broadcast*.

The oracle, $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot, \cdot, \cdot)$ responds to the queries made by all the p processors in parallel. We are not concerned with the exact mechanism for message passing, for example it could be performed using shared memory. Also, the main resource that we want to exploit by using *parallelism* is oracle queries. Thus, we allow the processors to perform arbitrary (polynomially-bounded) computation and write arbitrarily long (polynomially-bounded) messages during each parallel time-step.

Definition 5.4 (Parallel Learning with $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot,\cdot,\cdot)$ oracle)**.** *We say that a concept class, C, over instance space, $\mathcal{X}$, is (τ_{Alg}, T)-parallel learnable using algorithm,* Alg*, using the $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot,\cdot,\cdot)$ oracle and p processors, under distribution, D, and with respect to loss function, ℓ, if for every $\epsilon > 0$ and target (ideal) function, $f \in C$, in at most T parallel time-steps,* Alg*, outputs a hypothesis, h, that satisfies, $\mathrm{L}_{f,D}(h) \leq \epsilon$. Every query made by* Alg *is of the form $(\phi, \tau_{\mathsf{Alg}}, \theta)$, where $\phi : \mathcal{X} \to \mathcal{Y}'$ is efficiently evaluable, $1/\tau_{\mathsf{Alg}}$ is bounded by a polynomial in n and $1/\epsilon$, and $\tau_{\mathsf{Alg}} \leq |\theta| \leq B$. The final hypothesis, h, must be polynomially evaluable and during each parallel time-step, each of the p processors must complete their computation and message writing in polynomial time.*

5.4 Reduction to Evolution with Recombination

In this section, we prove the result that fast parallel learning algorithms lead to faster (in terms of number of generations) evolution when recombination is allowed. The ideas used in the reduction are similar as those in Section 3.8.1. We first show the result using selection rule, $\mathsf{BN\text{-}Sel}$ (and also $\mathsf{Opt\text{-}Sel}$). The *weak selection* rule is considered separately in Section 5.4.1. We show the following result:

Theorem 5.1. *Suppose concept class, C, is (τ_{Alg}, T)-parallel learnable by algorithm,* Alg*, using $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot,\cdot,\cdot)$ oracle and p processors with respect to distribution, D, and loss function, ℓ, then C is evolvable with recombination by a polynomially bounded evolutionary mechanism, using selection rule,* $\mathsf{BN\text{-}Sel}$*, under distribution, D, and with respect to loss function, ℓ, if evolution is started with an* initialized population, *P_0, in $T(\log(p) + 2) + 1$ generations.*

Remark 5.1. *Although we prove this theorem for selection rule,* $\mathsf{BN\text{-}Sel}$*, it will be clear from the proof that the reduction is also valid for selection rule,* $\mathsf{Opt\text{-}Sel}$*. The situation for* $\mathsf{Weak\text{-}Sel}$ *is more involved and is discussed in greater detail in Section 5.4.1.*

We first describe a high-level outline of the proof strategy. Let Alg be a (τ_{Alg}, T)-parallel $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot,\cdot,\cdot)$ algorithm for learning concept class, C. We define (as in Section 3.8.1) a representation class, R, that contains representations encoding the state of a simulation of Alg. We assume that at the end of each parallel step, all processors used by algorithm, Alg, have the same *state information* denoted by some string, z. This is without loss of generality, since processors are allowed any polynomially bounded communication. However, each processor has a unique identity, i, that differentiates its actions from those of other processors. We start with a population, P_0, in which every member is identical. We assume that at the beginning, each member of the population is $\mathsf{r}^0 \equiv \mathbf{Z}$. We show that the behavior of the algorithm, Alg, during each parallel-step can be simulated using at exactly $\log(p) + 2$ evolutionary steps. We refer to the simulation of the k^{th} parallel step as phase k. The outline of the simulation is as follows:

1. At the start of phase k, the population is identical, denoted by some representation, r^{k-1}. Each phase begins with a differentiation step, at the end of which each member of the population has an identity of the processor that it will simulate. Furthermore, there are roughly equal number of members simulating each processor. This is achieved by defining $\mathsf{Rec}(\mathsf{r}^{k-1}, \mathsf{r}^{k-1}, \epsilon)$, that outputs $\mathsf{r}^{i,k-1}$ with probability $1/p$ for $1 \leq i \leq p$. The representation, $\mathsf{r}^{i,k-1}$, encodes exactly

the same function as r^{k-1} but also encodes the identity, i, of the processor it will simulate.

2. After differentiation, each individual simulates the processor whose identity is encoded in the representation of that individual. This step is similar to the reduction in Section 3.8.1, but is now carried out using a recombinator, Rec, rather than a mutator. The descendants that survive encode a valid query response to the query that the processor makes during the k^{th} parallel time-step (given the state), and also encode the message that the processor broadcasts at the end of the k^{th} parallel time-step.

3. The communication (message passing) in each parallel step is simulated using $\log(p)$ evolutionary steps. At the end of these, the population once again becomes identical, denoted by representation, r^k, and phase k is over.

At the end of phase T, each representation, r^T, encodes a state of a valid simulation of algorithm, Alg, and hence can produce a hypothesis, h, that has low (at most ϵ) expected loss. Thus, in one additional evolutionary step, it is possible to have a representation in the population that has expected loss at most ϵ.

We now describe the construction of the evolutionary mechanism in detail. In order to facilitate and easier reading of the proof, we describe representations that are contained in the representation class, R, as they are required in the proof. We define the recombination operator, Rec, on any pair of representations as we need them. If for any pair of representations, the recombinator is left undefined, we assume that it always outputs, $\perp$, *i.e.* mating between the pair is considered impossible.

Recall that each individual (member) of the population has a (string) representation in R. Each representation is thought of as a randomized function (see Section 3.8.1 for more details) – we express $r \in R$ as $r = \sum_{i=1}^{m} w_i \psi_i$, where each $\psi_i : \mathcal{X} \to \mathcal{Y}'$ is a deterministic function, $w_i \geq 0$ and $\sum_{i=1}^{m} w_i \leq 1$. For any $x \in \mathcal{X}$, $r(x) = \psi_i(x)$ with probability w_i, for $1 \leq i \leq m$, and $r(x) = \mathbf{Z}(x) = \mathbf{0}$ with probability $1 - \sum_{i=1}^{m} w_i$. In addition to encoding the randomized function, the representation, r, may also encode state information.

We describe the simulation of the k^{th} parallel step of the algorithm, Alg, for some k. We postpone the exact description of representation, r^{k-1}, but for now assume it encodes the state of a valid simulation of Alg up to the end of the first $k-1$ parallel steps. At the end of any parallel step, we may assume that the state of all processors used by the algorithm is identical because of the messages broadcast.

The recombinator, $\mathsf{Rec}(\mathsf{r}^{k-1}, \mathsf{r}^{k-1}, \epsilon)$, outputs $\mathsf{r}^{i,k-1}$ with probability $1/p$ for $1 \leq i \leq p$. The representation, $\mathsf{r}^{i,k-1}$, encodes exactly the same (randomized) function (from $\mathcal{X} \to \mathcal{Y}'$) as r^{k-1}, but the *string representation* of $\mathsf{r}^{i,k-1}$ additionally encodes i, the identity of the i^{th} processor.

Thus, suppose that P^{k-1} is a population that consists only of representations, r^{k-1}, then after one evolutionary step we get a population P_1^{k-1} such that roughly $1/p$ fraction of P_1^{k-1} is $\mathsf{r}^{i,k-1}$ for every $1 \leq i \leq p$.

In the next evolutionary step, the individuals simulate the $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot,\cdot,\cdot)$ query made by the processor whose identity is encoded in their representation. Let $(\phi^{i,k-1}_{z[k-1]}, \tau_{\mathsf{Alg}}, \theta^{i,k-1}_{z[k-1]})$ be the query made by the i^{th} processor during parallel time-step k, where $z[k-1]$ is the string that captures the state information of Alg, up to the end of $k-1$ parallel time-steps. Note that the query response will be either 0 or 1, and then depending on the query response the processor may perform some computation and broadcast a message. Let $\sigma_b^{i,k-1}$ be the message that the i^{th} processor broadcasts when the query response is b, for $b \in \{0,1\}$. Define the representation:

$$\mathsf{r}^{i,k-1}_{b,\sigma_b^{i,k-1}} = \mathsf{r}^{i,k-1} + \frac{\mathbb{I}(b=1)}{|\theta^{i,k-1}_{z[k-1]}|N}\phi^{i,k-1}_{z[k-1]},$$

where N is a number that will make all the representations valid (*i.e.* make the weights sum up to at most 1) and will be defined later. $\mathbb{I}()$ is the identity function which is 1 if its argument is true and 0 otherwise. Thus, the representation, $\mathsf{r}^{i,k-1}_{0,\sigma_0^{i,k-1}}$ is functionally identical to $\mathsf{r}^{i,k-1}$ and r^{k-1}, but encodes additional information such as the candidate query response, 0, and the message, $\sigma_0^{i,k-1}$. The representation, $\mathsf{r}^{i,k-1}_{1,\sigma_1^{i,k-1}}$ in addition to encoding the candidate query response, 1, and the message, $\sigma_1^{i,k-1}$, also encodes an additional function, $\phi^{i,k-1}_{z[k-1]}$ with weight $1/(|\theta^{i,k-1}_{z[k-1]}|N)$.

We now define the recombination operator, Rec, for the representations, $\mathsf{r}^{i,k-1}$ and $\mathsf{r}^{i,k-1}$, as follows: Let $(\phi^{i,k-1}_{z[k-1]}, \tau_{\mathsf{Alg}}, \theta^{i,k-1}_{z[k-1]})$ be the query made by processor i during the k^{th} parallel step. Then we have the following (the parameter η is to bound probabilities appropriately and will be determined later):

- If $\theta^{i,k-1}_{z[k-1]} > 0$, then with probability $1-\eta$, $\mathsf{Rec}(\mathsf{r}^{i,k-1}, \mathsf{r}^{i,k-1}, \epsilon) = \mathsf{r}^{i,k-1}_{1,\sigma_1^{i,k-1}}$ and with probability η, $\mathsf{Rec}(\mathsf{r}^{i,k-1}, \mathsf{r}^{i,k-1}, \epsilon) = \mathsf{r}^{i,k-1}_{0,\sigma_0^{i,k-1}}$.
- If $\theta^{i,k-1}_{z[k-1]} < 0$, then with probability $1-\eta$, $\mathsf{Rec}(\mathsf{r}^{i,k-1}, \mathsf{r}^{i,k-1}, \epsilon) = \mathsf{r}^{i,k-1}_{0,\sigma_0^{i,k-1}}$ and with probability η, $\mathsf{Rec}(\mathsf{r}^{i,k-1}, \mathsf{r}^{i,k-1}, \epsilon) = \mathsf{r}^{i,k-1}_{1,\sigma_1^{i,k-1}}$.

We show in Claim 5.1 that with high probability, the descendant chosen will be will be such that it encodes a valid answer (and hence also the corresponding broadcast message) to the query $(\phi^{i,k-1}_{z[k-1]}, \tau_{\mathsf{Alg}}, \theta^{i,k-1}_{z[k-1]})$. Also, in the case that $i \neq i'$, we define the recombinator so that, $\mathsf{Rec}(\mathsf{r}^{i,k-1}, \mathsf{r}^{i',k-1}, \epsilon) = \perp$, with probability 1. We will make sure that N and $1/\eta$ are polynomially bounded (these are defined in the proof of Theorem 5.1). Then we define the approximation parameter, $\tau = \tau_{\mathsf{Alg}}/(2BN)$, the size function, $s(r', r'', \epsilon) = (1/\eta)\log(1/\eta)$ and the tolerance function, $t(r', r'', \epsilon) = 1/N$ (where the size function and tolerance function are defined for any pair of representations of the form r^{k-1} or $\mathsf{r}^{i,k-1}$). The evolutionary mechanism, $\mathcal{E}M(R, \mathsf{Rec}, \tau, s, t)$, evolves C with recombination, using selection rule, $\mathsf{BN\text{-}Sel}$.

In the proofs in this section, we use the following version of the Chernoff-Hoeffding bound: If $\langle X_i \rangle_{i=1}^m$ are independent and identically distributed random variables, with mean μ, such that $X_i \in [0,1]$, then,

$$\Pr\left[\neg\left(\frac{\mu}{1+\delta} \leq \frac{1}{m}\sum_{i=1}^{m} X_i \leq \mu(1+\delta)\right)\right] \leq 2e^{-\delta^2 \mu m/12}$$

We now prove the following claim:

Claim 5.1. *Suppose that we begin with a population P^{k-1} in which each member is identical, r^{k-1}. Suppose $\alpha = 1/p$, then for every $0 \leq \delta \leq 2^{1/4} - 1$, after 2 evolutionary steps we get a population in which the following hold with probability at least $1 - 4pe^{-\delta^2 \alpha |P|/24} - 2\eta |P^{k-1}|$:*

1. *Each member of the population is of the form $\mathsf{r}^{i,k-1}_{b,\sigma^{i,k-1}_b}$ where b is a valid query response to the query $(\phi^{i,k-1}_{z[k-1]}, \tau_{\mathsf{Alg}}, \theta^{i,k-1}_{z[k-1]})$ made by processor i during the kth parallel step and $\sigma^{i,k-1}_b$ is the message the processor i would broadcast.*

2. *If f_i is the fraction of the population of type $\mathsf{r}^{i,k-1}_{b,\sigma^{i,k-1}_b}$, then*

$$\frac{\alpha}{(1+\delta)^5} \leq f_i \leq (1+\delta)^5 \alpha$$

Proof. Let P^{k-1} be the starting population; each member of P^{k-1} is r^{k-1}. Note that the recombinator $\mathsf{Rec}(\mathsf{r}^{k-1}, \mathsf{r}^{k-1}, \epsilon)$ outputs a representation $\mathsf{r}^{i,k-1}$ with probability $1/p$ for every $1 \leq i \leq p$. Thus, when the population for the next generation (after one evolutionary step) is picked, the probability that any fixed individual has representation, $\mathsf{r}^{i,k-1}$, is exactly $1/p$, for any i. Thus, after one generation, the population will be differentiated and the expected number of individuals $\mathsf{r}^{i,k-1}$ is $\alpha|P| = |P|/p$. Thus, except with probability $2e^{-\delta^2\alpha|P|/12}$, the fraction of $\mathsf{r}^{i,k-1}$ in the population, say $\tilde{f}_i$, satisfies $\alpha/(1+\delta) \leq \tilde{f}_i \leq (1+\delta)\alpha$. Taking union bound, this holds for all i with probability at least $1 - 2pe^{-\delta^2\alpha|P|/12}$.

Thus, after one evolutionary step, each member of the population is of the form $\mathsf{r}^{i,k-1}$, and assume that $\tilde{f}_i$ satisfies $\alpha/(1+\delta) \leq \tilde{f}_i \leq \alpha(1+\delta)$ (allowing the evolutionary process to fail with probability $2pe^{-\delta^2\alpha|P|/12}$). Next, we show the following: Suppose, $\theta^{i,k-1}_{z[k-1]} > 0$, then for tolerance function, $t = 1/N$, and approximation parameter, $\tau = \tau_{\mathsf{Alg}}/(2N)$, if 0 is the only valid answer to the query, $(\phi^{i,k-1}_{z[k-1]}, \tau_{\mathsf{Alg}}, \theta^{i,k-1}_{z[k-1]})$, then $\mathsf{r}^{i,k-1}_{1,\sigma_1^{i,k-1}}$ is a deleterious recombinant. This follows using exactly the same argument as used in the proof of Theorem 3.5. If 0 is the only valid answer to the query, it must be the case that $\mathsf{G}_{f,D}(\phi^{i,k-1}_{z[k-1]}) \geq \theta^{i,k-1}_{z[k-1]} + \tau_{\mathsf{Alg}}$. Then, $\mathrm{L}_{f,D}(\mathsf{r}^{i,k-1}_{1,\sigma_1^{i,k-1}}) - \mathrm{L}_{f,D}(\mathsf{r}^{i,k-1}) = \mathsf{G}_{f,D}(\mathsf{r}^{i,k-1}_{1,\sigma_1^{i,k-1}}) - \mathsf{G}_{f,D}(\mathsf{r}^{i,k-1}) = (1/(|\theta^{i,k-1}_{z[k-1]}|N))\mathsf{G}_{f,D}(\phi^{i,k-1}_{z[k-1]}) \geq 1/N + \tau_{\mathsf{Alg}}/N \geq t + 2\tau$. Thus, $\mathsf{r}^{i,k-1}_{1,\sigma_1^{i,k-1}}$ is a deleterious recombinant.

The following two statements are true:

(i) If 0 is not the only valid answer, *i.e.* 1 is also a valid answer, to the query $(\phi^{i,k-1}_{z[k-1]}, \tau_{\mathsf{Alg}}, \theta^{i,k-1}_{z[k-1]})$, then a descendant chosen as

$$r \leftarrow \mathsf{BN\text{-}Sel}(\mathsf{r}^{i,k-1}, \mathsf{r}^{i,k-1}, \mathsf{Rec}, \widehat{\mathrm{L}}_{f,D}, \tau, s, t)$$

will satisfy $r = \mathsf{r}^{i,k-1}_{1,\sigma_1^{i,k-1}}$ with probability at least $1 - \eta$.

(ii) Except with probability η, when the size function $s \geq (1/\eta)\log(1/\eta)$, the candidate pool, $\mathrm{Desc}(\mathsf{r}^{i,k-1}, \mathsf{r}^{i,k-1}, \epsilon)$, will contain at least one copy of the representation, $\mathsf{r}^{i,k-1}_{0,\sigma_0^{i,k-1}}$. Thus, when 0 is the only valid answer to the query, $(\phi^{i,k-1}_{z[k-1]}, \tau_{\mathsf{Alg}}, \theta^{i,k-1}_{z[k-1]})$, $\mathsf{r}^{i,k-1}_{0,\sigma_0^{i,k-1}}$ will be the chosen descendant and not $\perp$.

The case when $\theta^{i,k-1}_{z[k-1]} < 0$ is (almost exactly) symmetric – in that case we show that when 1 is only valid answer to the query, $(\phi^{i,k-1}_{z[k-1]}, \tau_{\mathsf{Alg}}, \theta^{i,k-1}_{z[k-1]})$, then the representation, $\mathsf{r}^{i,k-1}_{1,\sigma_1^{i,k-1}}$, is *beneficial.*

Thus, the following is true: Except with probability $2|P^{k-1}|\eta$, after two generations each member of the population is of the form $r^{i,k-1}_{b,\sigma_b^{i,k-1}}$, where b is a valid answer to the query, $(\phi^{i,k-1}_{z[k-1]}, \tau_{\mathsf{Alg}}, \theta^{i,k-1}_{z[k-1]})$.

The statement about the fraction of representations of each type now follows immediately using the Chernoff-Hoeffding bound, since the expected number of representations of the form $\mathsf{r}^{i,k-1}_{b,\sigma_b^{i,k-1}}$ (given $\tilde{f}_{i'}$ for all i') is $(\tilde{f}_i^2/\sum_{i'}\tilde{f}_{i'}^2)|P|$. Note that $\alpha/(1+\delta)^4 \leq \tilde{f}_i^2/\sum_{i'}\tilde{f}_{i'}^2 \leq \alpha(1+\delta)^4$. Since $\tilde{f}_i^2/\sum_{i'}\tilde{f}_{i'}^2 \geq \alpha/2$ for the value of δ in the statement of the Claim, except with probability $2pe^{-\delta^2\alpha|P|/24}$, for all i, it holds that $\alpha/(1+\delta)^5 \leq f_i \leq \alpha(1+\delta)^5$. The assertion of the claim follows by taking union bound over failure events in the two evolutionary steps. $\square$

Thus, at the end of 2 evolutionary steps in phase k, we reach a population, P_1^{k-1}, such that every member of the population is of the form $r^{i,k-1}_{b,\sigma_b^{i,k-1}}$, where b represents a valid response to the query, $(\phi^{i,k-1}_{z[k-1]}, \tau_{\mathsf{Alg}}, \theta^{i,k-1}_{z[k-1]})$, made by the i^{th}processor during the k^{th} parallel time-step. We now define the action of the recombinator on these new representations. We define a few more intermediate representations:

- Let $\rho_0^{i,k-1} = \{\mathsf{r}^{i,k-1}_{0,\sigma_0^{i,k-1}}, \mathsf{r}^{i,k-1}_{1,\sigma_1^{i,k-1}}\}$ be a set of representations of the type $\mathsf{r}^{i,k-1}_{b,\sigma_b^{i,k-1}}$. For notational convenience, we will just say representation, $\rho_0^{i,k-1}$, when, in fact, we mean some representation from the set $\rho_0^{i,k-1}$. This is because the action of the recombinator on any representation from this set is identical.)

- Assume for simplicity that p is a power of 2. For $j = 1, \ldots, \log(p)$ and for $1 \leq i \leq p/2^j$ define the set $\rho_j^{i,k-1} = \rho_{j-1}^{2i-1,k-1} + \rho_{j-1}^{2i,k-1} - \mathsf{r}^{k-1}$.

  Note: We have abused notation considerably in the above expression to keep the number of symbols manageable. It is possible to show (by induction), that the set $\rho_j^{i,k-1}$ has 2^{j+1} elements. This is immediately true for the case when $j = 0$, which has two elements, $\mathsf{r}^{i,k-1}_{0,\sigma_0^{i,k-1}}$ and $\mathsf{r}^{i,k-1}_{1,\sigma_1^{i,k-1}}$.

  To see this when $j > 0$, observe the following: Each representation from the set $\rho_{j-1}^{2i-1,k-1}$ is of the form $\mathsf{r}^{k-1} + \psi_{j-1}^{2i-1,k-1}$, where r^{k-1} is the representation that encodes the state of simulation of the algorithm, Alg, for the first $k-1$ parallel time-steps. The part, $\psi_{j-1}^{2i-1,k-1}$, encodes query responses obtained by the processors, $((2i-1)-1)2^{j-1}+1, ((2i-1)-1)2^{j-1}+2, \ldots, ((2i-1)-1)2^{j-1}+2^{j-1}$, during the k^{th} parallel time-step. Similarly, a representation in the set, $\rho_{j-1}^{2i,k-1}$, is of the form, $\mathsf{r}^{k-1} + \psi_{j-1}^{2i,k-1}$, where $\psi_{j-1}^{2i,k-1}$ encodes query responses obtained by the processors, $(2i-1)2^{j-1}+1, \ldots, (2i-1)2^{j-1}+2^{j-1}$, during the k^{th} parallel time-step. Thus, the resulting representation, $\mathsf{r}^{k-1} + \psi_{j-1}^{2i-1,k-1} + \psi_{j-1}^{2i,k-1}$, is in the set, $\rho_j^{i,k-1}$, and encodes query responses obtained by the processors, $(i-1)2^j+1, (i-1)2^j+2, \ldots, (i-1)2^j+2^j$, during the k^{th} parallel time-step.

We now define the action of the recombinator on these representations. We overload notation and use representation, $\rho_j^{i,k-1}$, to mean some representation from the set, $\rho_j^{i,k-1}$.

1. For representations, $\rho_{j-1}^{2i-1,k-1}$ and $\rho_{j-1}^{2i,k-1}$ for $j = 1, \dots \log(p)$, $\mathsf{Rec}(\rho_{j-1}^{2i-1,k-1}, \rho_{j-1}^{2i,k-1}, \epsilon)$ outputs $\rho_j^{i,k-1}$ with probability 1. We assume that the representation, $\rho_j^{i,k-1}$, also encodes all messages (broadcast by various processors) that were contained in $\rho_{j-1}^{2i-1,k-1}$ and $\rho_{j-1}^{2i,k-1}$. (Thus, $\rho_j^{i,k-1}$ encodes complete state information (query responses and message broadcasts) of processors $(i-1)2^j + 1, \dots, (i-1)2^j + 2^j$, in the k^{th} parallel time-step. Note that in particular, $\rho_{\log(p)}^{1,k-1}$ includes state information of *all* p processors and we denote this representation by r^k, the representation at the end of k parallel time-steps.)

2. For any other pair of representations r', r'', $\mathsf{Rec}(r', r'', \epsilon) = \perp$ with probability 1.

Define the tolerance function, $t(r', r'', \epsilon) = (p + 2)/N + \tau_{\mathsf{Alg}}/(BN)$, the approximation parameter, $\tau = \tau_{\mathsf{Alg}}/(2N)$, and the size function $s(r', r'', \epsilon) = (1/\eta) \log(1/\eta)$, where r' and r'' are representations of the form $\rho_j^{i,k-1}$ for some i, j, k.

We now prove the following claim:

Claim 5.2. *Suppose we start with a population that satisfies the assertion of Claim 5.1 and let* $\alpha = 1/p$. *Then for every* $0 \leq \delta \leq \ln(2)/(5p^3)$, *after* $\log(p)$ *evolutionary steps, with probability at least* $1 - (2p(\log(p) + 2)e^{-\delta^2 \alpha |P|/24} + 2|P|\eta)$ *the population becomes identical, represented by* r^k.

Proof. The main idea is that we combine state information of processors, two at a time in a tree-like manner and thus after $\log(p)$ steps, there are individuals that encode the state of simulation of all p processors (and hence of the parallel algorithm, Alg, itself). For simplicity, we assume that p is a power of 2, so that $\log(p)$ is an integer.

We claim that after j evolutionary steps, with probability at least $1 - (2(j+2)pe^{-\delta^2\alpha|P|/12} - 2|P|\eta)$ the following hold:

1. Each member of the population is of the form $\rho_j^{i,k-1}$, $i = 1, \ldots, 2^{\log(p)-j}$.

2. If f_i^j is the fraction of the population of the form $\rho_j^{i,k-1}$, then

$$2^j\alpha/(1+\alpha)^{a_j} \leq f_i^j \leq 2^j\alpha(1+\delta)^{a_j},$$

where a_j is the sequence defined by the recurrence, $a_0 = 5$, $a_t = 4a_{t-1} + 1$. We show this assertion by induction. It clearly holds when $j = 0$ (because the statement is then just the assertion of Claim 5.1). Suppose the statement holds for j, the consider the corresponding statement for $j+1$. Consider an individual $\rho_{j+1}^{i,k-1}$ that is a possible descendant of $\rho_j^{2i-1,k-1}$ and $\rho_j^{2i,k-1}$. The number of $(\rho_j^{2i-1,k-1}, \rho_j^{2i,k-1})$ pairs is $f_{2i-1}^j f_{2i}^j |P|^2$ and we know that,

$$\frac{(2^j\alpha|P|)^2}{(1+\delta)^{2aj}} \leq f_{2i-1}^j f_{2i}^j |P|^2 \leq (2^j\alpha|P|)^2(1+\delta)^{2a_j}$$

On the other hand the total number of possible feasible pairs that may produce any descendants is $\sum_{l=1}^{2^{\log(p)-(j+1)}} f_{2l-1}^j f_{2l}^j |P|^2$, which satisfies,

$$\frac{p}{2^{j+1}}\frac{(2^j\alpha|P|)^2}{(1+\delta)^{2a_j}} \leq \sum_{l=1}^{2^{\log(p)-(j+1)}} f_{2l-1}^j f_{2l}^j |P|^2 \leq \frac{p}{2^{j+1}}(2^j\alpha|P|)^2(1+\delta)^{2a_j}$$

Thus the probability for an individual in the $(j+1)^{th}$ generation, that the parents from the j^{th} generation are $\rho_j^{2i-1,k-1}$ and $\rho_j^{2i,k-1}$ is at least $2^{j+1}\alpha/(1+\delta)^{4a_j}$ and at most $2^{j+1}\alpha(1+\delta)^{4a_j}$. Thus, by the Chernoff-Hoeffding bound, except with a further probability loss of $2pe^{-\delta^2 f_i^{j+1}|P|/12}$ we have that $2^{j+1}\alpha/(1+\delta)^{4a_j+1} \leq f_i^{j+1} \leq 2^{j+1}\alpha(1+\delta)^{4a_j+1}$.

Notice that $a_t \leq 5^{t+1}$ for all t, and hence for the values of δ in the statement of the claim, the values f_j^i for all i, j are at least $\alpha/2$. Thus the statement holds by induction. However, after $\log(p)$ steps the population is homogeneous and every individual is the representation, $\rho_{\log(p)}^{1,k-1} \equiv \mathsf{r}^k$.

The only part remaining to be proved is that $\rho_{j+1}^{i,k-1}$ is a feasible descendant of $\rho_j^{2i-1,k-1}$ and $\rho_j^{2i-1,k-1}$. From the proof of Claim 5.1, it is clear that $\mathrm{L}_{f,D}(\mathsf{r}^{i,k-1}_{b,\sigma_b^{i,k-1}}) - \mathrm{L}_{f,D}(\mathsf{r}^{k-1}) \leq 1/N + 2\tau$ (if $\mathsf{r}^{i,k-1}_{b,\sigma_b^{i,k-1}}$ is selected to the next generation it must be beneficial or neutral and $t(\mathsf{r}^{i,k-1}, \mathsf{r}^{i,k-1}, \epsilon) = 1/N$). Thus, for any representation of the form, $\rho_j^{i,k-1}$, at most 2^j functions corresponding to some query may be included, each with weight $1/N$. Thus, when the tolerance function $t = ((p+2)/N) + \tau_{\mathsf{Alg}}/(BN)$, it must be the case that $\rho_{j+1}^{i,k-1}$ is neutral (with respect to $\rho_j^{2i-1,k-1}$ and $\rho_j^{2i,k-1}$), *i.e.* it is a feasible descendant, for any value of $j = 1, \dots, \log(p)$. $\square$

We can now prove Theorem 5.1.

of Theorem 5.1. We start with a population where each individual has the representation, $\mathsf{r}^0 \equiv \mathbf{Z}$. Each parallel step of the algorithm, Alg, is simulated by $\log(p) + 2$ evolutionary steps. Thus, a (τ_{Alg}, T)-parallel $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot,\cdot,\cdot)$ algorithm can be simulated using an evolutionary mechanism in $T(\log(p)+2)$ evolutionary steps (generations). At the end of this simulation, the population is also identical and encodes the representation r^T which contains the final state of the algorithm, Alg. Let h_T be the hypothesis that algorithm, Alg, would output, then we define the recombinator operator on r^T and r^T as follows: With probability η, $\mathsf{Rec}(\mathsf{r}^T, \mathsf{r}^T, \epsilon)$ outputs r^T and with probability

$1 - \eta$, $\mathsf{Rec}(\mathsf{r}^T, \mathsf{r}^T, \epsilon)$ outputs h. Since, the correctness of the simulation guarantees that $\mathrm{L}_{f,D}(h) \leq \epsilon$, the only reason why r^T may be selected is that it actually has lower expected loss than h_T (except with a very low probability).

Finally, define $N = pT/\tau_{\mathsf{Alg}}$, this makes sure that no (randomized) representation has weights of its constituent deterministic functions adding up to greater than 1. (This is because the total number of queries is at most pT for p processors and T time steps, and the value θ for any of the query must satisfy, $|\theta| \geq \tau_{\mathsf{Alg}}$.) The total failure probability of the algorithm is $2(p(\log(p) + 1))e^{-\delta^2\alpha|P|/24} + (2|P| + 1)\eta$. Thus, when $\delta = 1/(10p^3)$ (the one used in Claims 5.1 and 5.2), $|P| = 9600p^7\log(p)/\epsilon$ and $\eta = \epsilon/(4|P| + 2)$, except with probability, $1 - \epsilon$, evolution (with recombination) succeeds in producing a representation that has expected loss at most ϵ. $\square$

Remark 5.2. *The reader can verify that the above proof is equally valid when the selection rule used is* $\mathsf{Opt\text{-}Sel}$.

We state a simple corollary of Theorem 5.1 that shows that in fact a parallel algorithm that only uses, $\widehat{\mathsf{G}}_{f,D}(\cdot,\cdot)$, oracle can be simulated by evolution with recombination. This follows direction from Lemma 3.1, which shows that any algorithm that takes T' parallel time-steps with access to a $\widehat{\mathsf{G}}_{f,D}(\cdot,\cdot)$ oracle, can be modified to take $T'O(\log(B/\tau_{\mathrm{pa}}))$ parallel time-steps and make queries to the $\widehat{\mathsf{G}}^{\leq}_{f,D}(\cdot,\cdot,\cdot)$ oracle (where τ_{pa} is the smallest value of τ used by the parallel algorithm in any query to the $\widehat{G}_{f,D}(\cdot,\cdot)$ oracle). We also indicate the consequences for special cases of CSQ and SQ learning. Note that an *efficient* parallel algorithm in the CSQ or SQ framework uses polynomially many (in n and $1/\epsilon$) processors and only uses values of τ in queries such that $1/\tau$ is polynomially (in n and $1/\epsilon$ bounded).

Corollary 5.1. *If concept class, C, is learnable by an* efficient *parallel algorithm that only makes queries to a $\widehat{G}_{f,D}(\cdot,\cdot)$ oracle (and for each query $\tau \geq \tau_{pa}$), under distribution, D, and with respect to loss function, ℓ, in T' parallel-time steps and using p processors, then C is evolvable with recombination, with respect to loss function, ℓ, selection rule,* $\mathsf{BN\text{-}Sel}$ *(or* $\mathsf{Opt\text{-}Sel}$*), under distribution, D, in $T' \log(B/\tau_{pa})(\log(p) + 2) + 1$ generations (evolutionary time-steps). In particular,*

1. *When C is learnable by an efficient parallel CSQ algorithm (with access to a* $\mathsf{CSQ}\text{-}\mathcal{O}$ *oracle) in T' time steps, then C is evolvable with recombination, when the loss function is the classification error, ℓ_c, in $O(T'(\log(n/\epsilon))^2)$ generations.*

2. *When C is learnable by an efficient parallel SQ algorithm (with access to the* STAT *oracle) in T' time-steps, then C is evolvable with recombination, with respect to the class of loss functions defined in Section 3.8.1 (under the title* Boolean Ideal Function and Real-Valued Representations*) in $O(T'(\log(n/\epsilon))^2)$.*

5.4.1 Recombination and Weak Selection

In this section, we show how the above reduction may be modified when considering evolution with *weak selection*, *i.e.* using the selection rule, $\mathsf{Weak\text{-}Sel}$. The reduction does not directly go through, because we can no longer rely on selection distinguishing between beneficial and neutral descendants in the proof of Claim 5.1. This was required when $\theta^{i,k-1}_{z[k-1]} < 0$, to ensure that when 1 was the only valid answer to the query, $(\phi^{i,k-1}_{z[k-1]}, \tau_{\mathsf{Alg}}, \theta^{i,k-1}_{z[k-1]})$, $\mathsf{r}^{i,k-1}_{1,\sigma_1^{i,k-1}}$ would be selected, even though the (neutral) descendant, $\mathsf{r}^{i,k-1}_{0,\sigma_0^{i,k-1}}$ was produced with overwhelmingly higher probability by the recombinator, Rec.

We can only prove an equivalent version of Theorem 5.1, when considering ideal functions that are boolean, representations that are (randomized) boolean functions, and the loss function is the classification error, ℓ_c. Thus, effectively we can show that parallel CSQ algorithms can be simulated by evolution with recombination with respect to classification error as the loss function. Note that in this case, $\mathcal{Y} = \mathcal{Y}' = \{-1, 1\}$. It is conceptually easier to allow, $\mathcal{Y}' = [-1, 1]$ and use the ℓ_1 loss, i.e. $\ell_1(y', y) = (1/2)|y' - y|$. Suppose $\phi : \mathcal{X} \to \{-1, 1\}$ is a randomized boolean function, let $\phi_1 : \mathcal{X} \to [-1, 1]$ be the function where $\phi_1(x) = \mathbb{E}[\phi(x) \mid x]$, the expectation is taken only over the internal randomness of ϕ. Note that for any ideal function, f, and distribution, D,

$$\begin{aligned}\mathbb{E}_{x\sim D}[\ell_c(\phi(x), f(x))] &= \mathbb{E}_{x\sim D}[\ell_1(\phi_1(x), f(x))] \\ &= (1 - \mathbb{E}_{x\sim D}[\phi(x)f(x)])/2 = (1 - \mathbb{E}_{x\sim D}[\phi_1(x)f(x)])/2\end{aligned}$$

Thus, we abuse notation and denote both the randomized function, ϕ, and the corresponding, ϕ_1 (with range $[-1, 1]$), just by the letter, ϕ. Also, the above argument shows that access to a loss oracle with respect to the ℓ_1 loss is equivalent to the CSQ oracle, $\mathsf{CSQ}\text{-}\mathcal{O}$.

The reduction requires a new kind of oracle, $\mathsf{ZCSQ}(\cdot, \cdot)$ (this is essentially an oracle that gives a (noisy) response to the question whether $\mathbb{E}_{x\sim D}[\phi(x)f(x)] \le 0$?). On the query, (ϕ, τ), where $\phi : \mathcal{X} \to [-1, 1]$, the $\mathsf{ZCSQ}(\cdot, \cdot)$ oracle responds as follows:

$$\mathsf{ZCSQ}(\phi, \tau) = \begin{cases} 1 & \text{if } \mathbb{E}_{x\sim D}[\phi(x)f(x)] \le -\tau \\ 0 & \text{if } \mathbb{E}_{x\sim D}[\phi(x)f(x)] \ge \tau \\ 1 \text{ or } 0 & \text{otherwise} \end{cases}$$

Feldman implicitly showed that learning with ZCSQ queries is equivalent to CSQ learning [14]. It follows from his proof, that a parallel CSQ algorithm that uses T parallel-time steps, can be modified to be a parallel ZCSQ algorithm that takes $T\log(1/\tau)$ parallel time-steps, where τ is the smallest value of the approximation parameter used in query to the oracle, $\mathsf{CSQ}\text{-}\mathcal{O}$. A proof is provided in Appendix A.

In this section, we assume that $\phi : \mathcal{X} \to [-1,1]$, the loss function is the ℓ_1 loss, and that $\mathrm{L}_{f,D}(\phi) = \mathbb{E}_{x\sim D}[\ell_1(\phi(x), f(x))]$ denotes the expected loss with respect to the ℓ_1 loss function. Define the zero function, $\mathbf{Z} : \mathcal{X} \to [-1,1]$, as $\mathbf{Z}(x) = 0$ for all x. Then, $\mathrm{L}_{f,D}(\mathbf{Z}) = 1/2$ for every ideal function, f, and target distribution, D. Recall that $\mathsf{G}_{f,D}(\phi) = \mathrm{L}_{f,D}(\phi) - \mathrm{L}_{f,D}(\mathbf{Z}) = -(1/2)\mathbb{E}_{f,D}[\phi(x)f(x)]$.

We now suppose that Alg is a parallel algorithm that uses p processors and makes only ZCSQ queries. Assume that the queries always use the approximation parameter, τ_{Alg}, where $1/\tau_{\mathsf{Alg}}$ is bounded by a polynomial in n and $1/\epsilon$. Suppose that Alg successfully learns concept class, C, in T parallel time-steps. We borrow notation from the proof of Theorem 5.1 above: the representation, $\mathsf{r}^{i,k-1}$, encodes the state of simulation up to the end of $k-1$ parallel time-steps and has the identity of the i^{th} processor, and let $(\phi^{i,k-1}_{z[k-1]}, \tau_{\mathsf{Alg}})$ be the query made by the i^{th} processor in k^{th} parallel time-step. We modify the representations, $\mathsf{r}^{i,k-1}_{b,\sigma^{i,k-1}_b}$ as follows:

$$\mathsf{r}^{i,k-1}_{b,\sigma^{i,k-1}_b} = \mathsf{r}^{i,k-1} + \frac{(-1)^b}{N}\phi^{i,k-1}_{z[k-1]}$$

Then observe that $\mathrm{L}_{f,D}(\mathsf{r}^{i,k-1}_{b,\sigma^{i,k-1}_b}) - \mathrm{L}_{f,D}(\mathsf{r}^{i,k-1}) = \mathsf{G}_{f,D}(\mathsf{r}^{i,k-1}_{b,\sigma^{i,k-1}_b}) - \mathsf{G}_{f,D}(\mathsf{r}^{i,k-1}) = ((-1)^b/N)\mathsf{G}_{f,D}(\phi^{i,k-1}_{z[k-1]}) = ((-1)^{b+1}/(2N))\mathbb{E}_{x\sim D}[\phi^{i,k-1}_{z[k-1]}(x)f(x)]$.

Suppose that $\mathsf{Rec}(\mathsf{r}^{i,k-1}, \mathsf{r}^{i,k-1}, \epsilon)$ outputs $\mathsf{r}^{i,k-1}_{b,\sigma^{i,k-1}_b}$ with probability $1/3$ for each of $b = 0, 1$, and $r^{i,k-1}_{*,\sigma^{i,k-1}_*}$ with probability $1/3$. Here, $r^{i,k-1}_{*,\sigma^{i,k-1}_*}$ is a representation that is functionally identical to $\mathsf{r}^{i,k-1}$, but the $*$ encodes the fact that both 0 or 1 may be valid query responses and $\sigma_*^{i,k-1}$ is the broadcast message in that case.

Suppose that the query, $(\phi_{z[k-1]}^{i,k-1}, \tau_{\mathsf{Alg}})$, has a unique correct answer, *i.e.* either $\mathbb{E}_{x\sim D}[\phi_{z[k-1]}^{i,k-1}(x)f(x)] \geq \tau_{\mathsf{Alg}}$ or $\mathbb{E}_{x\sim D}[\phi_{z[k-1]}^{i,k-1}(x)f(x)] \leq -\tau_{\mathsf{Alg}}$. Let b_R be the right answer and $b_W = 1-b_R$ be the wrong answer. Then, we observe that $\mathrm{L}_{f,D}(\mathsf{r}^{i,k-1}_{b_R,\sigma^{i,k-1}_{b_R}}) - \mathrm{L}_{f,D}(\mathsf{r}^{i,k-1}) \leq -\tau_{\mathsf{Alg}}/(2N)$ and $\mathrm{L}_{f,D}(\mathsf{r}^{i,k-1}_{b_W,\sigma^{i,k-1}_{b_W}}) - \mathrm{L}_{f,D}(\mathsf{r}^{i,k-1}) \geq \tau_{\mathsf{Alg}}/(2N)$. Thus, if the tolerance function, $t(r', r'', \epsilon) = \tau_{\mathsf{Alg}}/(2N)$ and the approximation parameter, $\tau = \tau_{\mathsf{Alg}}/(6N)$, it will be the case that $\mathsf{r}^{i,k-1}_{b_R,\sigma^{i,k-1}_{b_R}}$ is in the multi-set Feas and $\mathsf{r}^{i,k-1}_{b_W,\sigma^{i,k-1}_{b_W}}$ is not in the multi-set Feas, when the weak selection rule, $\mathsf{Weak\text{-}Sel}$, is used. When the query, $(\phi_{z[k-1]}^{i,k-1}, \tau_{\mathsf{Alg}})$, does not have a unique correct answer, all the three representations – $\mathsf{r}^{i,k_1}_{0,\sigma^{i,k-1}_0}$, $\mathsf{r}^{i,k_1}_{1,\sigma^{i,k-1}_1}$ and $\mathsf{r}^{i,k-1}_{*,\sigma^{i,k-1}_*}$ may be in the multi-set Feas. This is not a problem, since the query response encoded in the representation is still a valid one.

The rest of the proof (the part in Claim 5.2 and also the probability calculations) is essentially identical to the proof of Theorem 5.1. We note that one minor difference arises due to the fact that when the query response is not unique, the representations, r^k, at the end of phase k may not be exactly identical in the function they represent (because they may contain parts from $\mathsf{r}^{i,k_1}_{0,\sigma^{i,k-1}_0}$, $\mathsf{r}^{i,k_1}_{1,\sigma^{i,k-1}_1}$, or $\mathsf{r}^{i,k-1}_{*,\sigma^{i,k-1}_*}$), but they *are* identical in the sense that each representation encodes a *valid* simulation of the parallel algorithm, Alg, for k time-steps (which is sufficient for our purposes).

Thus, we get the following theorem.

Theorem 5.2. *Suppose C can be learned by an efficient parallel algorithm with access to a CSQ oracle, with respect to distribution, D, in T' parallel time-steps, then C is evolvable with recombination, with selection rule,* $\mathsf{Weak\text{-}Sel}$*, using either ℓ_c (classifica-*

tion error) or ℓ_1 loss function, under distribution, D, in $O(T(\log(n/\epsilon))^2)$ generations (evolutionary steps).

Remark 5.3. *Note that when the representations are randomized boolean functions, all loss functions, ℓ, such that $\ell(1,-1) = \ell(1,-1) \neq 0$ and $\ell(1,1) = \ell(-1,-1) = 0$ are equivalent to the classification error. Thus, when the representations are randomized boolean functions, the above theorem holds essentially for every loss function.*

5.5 Some Evolutionary Algorithms

In this section, we apply the reductions described in the previous section to obtain faster evolutionary mechanisms with recombination for some concept classes (under certain distributions). All these mechanisms rely on being initialized with a particular starting population. The results in this section are reductions from parallel CSQ learning algorithm, thus the evolutionary mechanisms work for any loss function.

5.5.1 Evolving Conjunctions

Valiant showed that the class of monotone conjunctions is evolvable in $O(n \log(n/\epsilon))$ generations under the uniform distribution [39]. Diochnos and Turán showed that Valiant's algorithm can be improved and in fact showed that monotone conjunctions can be evolved in $O(\log(1/\epsilon))$ generations under the uniform distribution (with initialization) [10]. However, the logarithmic (in $1/\epsilon$) number of generations depends heavily on the assumption of uniform distribution, and it is unclear if it can be generalized easily to other distributions. In fact, our results in Section 5.6 show that there are distributions, under which monotone conjunctions may not be evolved in $o(n/\log(n))$ generations, in Valiant's model (without recombination).

The reduction from learning algorithms to evolvability shows that conjunctions may be evolved in Valiant's model in $\tilde{O}(n)$ generations, with respect to any fixed distribution. Our reduction from parallel learning algorithms shows that in a model of evolution with recombination, evolution of conjunctions is possible in $O((\log(n)/\epsilon)^2)$ generations. This follows from a simple learning constant time parallel learning algorithm in the CSQ model.

Proposition 5.1. *There is a (non-uniform) parallel CSQ algorithm that learns the class of conjunctions in constant time with respect to any fixed distribution.*

Proof. Let f be a conjunction of literals – we use the convention that f evaluates to -1 if every literal of f is -1. We show that for any $i \in [n]$, we can determine whether, x_i, $\bar{x}_i$ or neither appears as a literal in f with high probability.

Note that for any i, we can estimate $\Pr[f = b \wedge x_i = b']$ very accurately (up to additive accuracy $\epsilon/(2n)$ for example) for all $b, b' \in \{-1, 1\}$ using the queries, $\mathbb{E}[f(x)x_i]$, $\mathbb{E}[f(x)]$, $\mathbb{E}[x_i]$. The last query is not *correlational*, but for any fixed distribution this may be provided as advice.

Start with an empty conjunction, g. Perform the following for each i: If $\Pr[f = -1 \wedge x_i = 1] \leq \epsilon/(2n)$, then put x_i in g; Else, if $\Pr[f = -1 \wedge x_i = -1] \leq \epsilon/(2n)$, then put $\bar{x}_i$ in g; Else, put neither x_i nor $\bar{x}_i$ in g.

Note that if x_i (respectively $\bar{x}_i$) is in the true conjunction, f. Then, $\Pr[f = -1 \wedge x_i = 1] = 0$ (respectively $\Pr[f = -1 \wedge x_i = 1] = 0$). Therefore, all true literals of f will certainly be inserted in g (because of the accuracy $\epsilon/(2n)$). Any extra literal we may have inserted in g, cannot cause error more than $\epsilon/(2n)$. Thus the total error of the conjunction g with respect to f cannot be more than ϵ. $\square$

Thus, using Theorems 5.1 and 5.2 we get the following corollary:

Corollary 5.2. *The class of conjunctions is evolvable with recombination under any fixed distribution in at most $O((\log(n/\epsilon))^2)$ generations with respect to any of the selection rules,* Weak-Sel*,* BN-Sel *or* Opt-Sel*, with respect to any consistent and bounded loss function, ℓ, for which $\ell(1,-1) = \ell(-1,1) \neq 0$ and $\ell(1,1) = \ell(-1,-1) = 0$.*

5.5.2 Evolving Homogeneous Linear Separators

In Section 3.7.2, we presented an evolutionary mechanism for evolving homogeneous linear separators with respect to radially symmetric distributions in $O(n/\epsilon)$ generations. Here, we give a simple parallel algorithm for learning homogeneous linear separators under radially symmetric distributions in the CSQ model in constant time. This implies an evolutionary mechanism with recombination for evolving homogeneous linear separators in $O((\log(n/\epsilon))^2)$ generations under radially symmetric distributions.

Proposition 5.2. *There is a parallel CSQ algorithm for learning the class of homogeneous linear separators in $\mathbb{R}^n$ in constant time with respect to radially symmetric distributions.*

Proof. We use notation from Section 3.7.2. Note that a homogeneous linear separator is represented as $h_{\mathbf{w}}$, where $\mathbf{w} \in \mathbb{R}^n$ such that $\|\mathbf{w}\|_2 = 1$. Let $\langle \mathbf{e}_i \rangle_{i=1}^n$ be any orthonormal basis for $\mathbb{R}^n$.

Let D be a radially symmetric distribution. Then, we saw in the proof of Theorem 3.2 that $\Pr_{x\sim D}[h_{\mathbf{w}}(x) \neq h_{\mathbf{e}_i}(x)] = \arccos(\mathbf{w}\cdot\mathbf{e}_i)/\pi$. Let $p_i = \Pr_{x\sim D}[h_{\mathbf{w}}(x) \neq h_{\mathbf{e}_i}(x)]$ and observe that $p_i = (1/2) - (1/2)\mathbb{E}_{x\sim D}[h_{\mathbf{w}}(x)h_{\mathbf{e}_i}(x)]$. Let $\hat{p}_i$ be an estimate of p_i, such that $|\hat{p}_i - p_i| \leq \eta$ (This can be obtained by making a query to the CSQ-$\mathcal{O}$ oracle with 2η as the approximation parameter).

Let $w_i = \mathbf{w}\cdot\mathbf{e}_i = \cos(\pi p_i)$, let $\hat{w}_i = \cos(\pi\hat{p}_i)$. Then $|w_i - \hat{w}_i| = |\cos(\pi p_i) - \cos(\pi\hat{p}_i)| = 2|\sin(\pi(p_i+\hat{p}_i)/2)\sin(\pi(p_i-\hat{p}_i))/2)| \leq \pi|p_i-\hat{p}_i| \leq \pi\eta$. Let $\tilde{\mathbf{w}} = \sum_{i=1}^n \hat{w}_i\mathbf{e}_i$ and $\hat{\mathbf{w}} = \tilde{\mathbf{w}}/\|\tilde{\mathbf{w}}\|_2$. For unit vector, $\mathbf{w}\in\mathbb{R}^n$, note that $\|\mathbf{w}\|_1 \leq \sqrt{n}\|\mathbf{w}\|_2$; in particular, when $\|\mathbf{w}\|_2 = 1$, $\|\mathbf{w}\|_1 \leq \sqrt{n}$.

Then we have,

$$\begin{aligned}\mathbf{w}\cdot\hat{\mathbf{w}} = \frac{\sum_{i=1}^n w_i\hat{w}_i}{\sqrt{\sum_{i=1}^n \hat{w}_i^2}} &\geq \frac{\sum_{i=1}^n w_i^2 - \sqrt{n}\eta\pi}{\sqrt{(\sum_{i=1}^n w_i^2) + n(\eta\pi)^2 + 2\sqrt{n}\eta\pi}} \\ &\geq \frac{1-\sqrt{n}\eta\pi}{1+\sqrt{n}\eta\pi} \geq 1 - 2\sqrt{n}\eta\pi\end{aligned}$$

Finally, using the fact that $1-\cos(\theta) \geq 4\theta^2/\pi^2$, we get that $\mathrm{L}_{h_{\mathbf{w}},D}(h_{\hat{\mathbf{w}}}) = \Pr_{x\sim D}[h_{\mathbf{w}}(x) \neq h_{\hat{\mathbf{w}}}(x)] \leq n^{1/4}\sqrt{\pi\eta/2}$. Choosing η appropriately completes the proof. □

Corollary 5.3. *The class of homogeneous linear separators is evolvable with recombination under the class of radially symmetric distributions in $O((\log(n/\epsilon))^2)$ generations with respect to any of the selection rules,* Weak-Sel, BN-Sel, *or* Opt-Sel, *with respect to any consistent and bounded loss function, ℓ, for which $\ell(-1,1) = \ell(1,-1) \neq 0$ and $\ell(1,1) = \ell(-1,-1) = 0$.*

5.6 Lower Bounds

In this section, we present lower bounds on the number of generations required for evolution in Valiant's original model, *i.e.* without recombination. The details of the model are in Chapter 3. We consider evolutionary mechanisms for some concept class, C, under distribution, D, with respect to loss function, ℓ. We need to use the notion of an ϵ-*net*.

Definition 5.5. *We say that a set of functions, $\mathcal{N}$ defined from $\mathcal{X} \rightarrow \mathcal{Y}'$ form an ϵ-net for concept class, C, with respect to distribution, D, and loss function, ℓ, if for every $f \in C$, there exists some $\nu \in \mathcal{N}$, $\nu : \mathcal{X} \rightarrow \mathcal{Y}'$, such that $\mathrm{L}_{f,D}(\nu) \leq \epsilon$.*

We view the evolutionary process as a communication protocol between two parties, which we call A and B for lack of better terminology. For simplicity, we will assume that the evolutionary mechanism is deterministic and that it always succeeds, *i.e.* not just with probability $1 - \epsilon$. See remark 5.4 to see how these assumptions may be removed. Let R be the representation class, Mut the mutator, τ the approximation parameter, $s : R \times \mathbb{R}^+ \rightarrow \mathbb{N}$ the size function and $t : R \times \mathbb{R}^+ \rightarrow \mathbb{R}^+$ the tolerance function.

The communication protocol may take several rounds, and in fact each communication round corresponds to one generation (or evolutionary step). A and B begin with the same starting representation, $r_0 \in R$. We assume that the mutator, and the Turing machines that compute the size and tolerance functions are known two both A and B (note that the description length of these Turing machines is a constant independent of n or ϵ). We assume that B has access to the oracle, $\widehat{\mathrm{L}}_{f,D}(\cdot, \cdot)$, and hence can simulate the entire evolutionary process. We are interested in counting the minimum number of bits, B must send to A, so that A can simulate an *identical* evolutionary process.

Suppose at the beginning of round i, both A and B have some representation, $r_{i-1} \in R$. (This is true when $i = 1$, and it will remain true for $i > 0$ as a result of the communication protocol.) Then, the protocol proceeds as follows:

1. A runs the mutator, $\mathsf{Mut}(r_{i-1}, \epsilon)$, $s(r_{i-1}, \epsilon)$ times. These mutations are in an ordered list of size $s(r_{i-1}, \epsilon)$.

2. B does the exact same thing as A – in particular the order of the mutations in B's list is exactly the same as A's. B uses the oracle, $\widehat{\mathrm{L}}_{f,D}(\cdot, \cdot)$, and some selection rule, Sel, to determine the mutation that will be the new representation, r_i, at the next generation. If $r_i = \perp$, B sends the number 0 to A; otherwise B sends the index j to the first mutation in the ordered list that has representation, r_i.

3. Thus, B sends A at most $\lceil \log(s(r_{i-1}, \epsilon)) + 1 \rceil$, bits and then A simulates an identical evolutionary process.

Note that for a polynomially bounded evolutionary process, $\lceil \log(s(r_{i-1}, \epsilon)) + 1 \rceil = O(\log(n/\epsilon))$ for any $r_{i-1} \in R$. Suppose the evolutionary process is guaranteed to succeed in the g generations, then the total number of bits received by A is $O(g \log(n/\epsilon))$. If f is the ideal function, let r^f be the representation produced by B (and hence also A) at the end of the evolutionary process. Note that, $\mathrm{L}_{f,D}(r^f) \leq \epsilon$, and hence the set $\mathcal{N} = \{r^f \mid f \in C\}$ forms an ϵ-net of C, with respect to D and ℓ. But, this implies that $|\mathcal{N}| = 2^{O(g \log(n/\epsilon))}$, since A required only $O(g \log(n/\epsilon))$ bits to produce the correct representation in the ϵ-net. In fact, this argument gives an *information theo-*

retic lower bound on the number of generations, g, required for evolution in Valiant's model (without recombination). We state this formally as:

Theorem 5.3. *For some concept class, C, distribution, D, and loss function, ℓ, if S_ϵ is such that the size of of any ϵ-net of C, with respect to D and ℓ, must be at least S_ϵ, then the number of generations required for any evolutionary mechanism in the sense of Valiant (without recombination) requires $\Omega(\log(S_\epsilon)/\log(n/\epsilon))$ generations.*

Remark 5.4. *If the evolutionary process uses randomness, we use a standard probabilistic method argument to show that there exists a polynomial length string, which when interpreted as random bits is guaranteed to result in successful evolution. The failure probability, ϵ, may be reduced to be smaller than $1/2^n$, by repeating the evolutionary process several times and taking the best one.*

On the other hand, evolution with recombination allows many more bits of information received per generation (because of polynomial-sized population). We now show some explicit lower bounds for the class of conjunctions and homogeneous linear separators.

Lower Bound for Monotone Conjunctions

Let $\mathcal{X} = \{-1, 1\}^n$ and let $\mathbf{e}_i = (-1, \ldots, -1, 1, -1, \ldots, -1)$ be the vector that has -1 in all positions except the i^{th} position. Consider, the simple distribution, D_1, over $\{-1, 1\}^n$, where $\Pr_{D_1}(\mathbf{e}_i) = 1/n$, for $1 \leq i \leq n$. Then, note that any two different monotone conjunctions differ on at least one of the $\mathbf{e}_i$s. Suppose that the loss function is the classification error, ℓ_c, and that the functions in the ϵ-net are required to be boolean. Then, if C is the class of monotone conjunctions over $\{-1, 1\}^n$, for

$\epsilon < 1/(2n)$, any ϵ-net of C under distribution, D_1, and loss function, ℓ_c, must have size 2^n. This is because no two monotone conjunctions can have the same function in the ϵ-net (if $f \neq g$ are two monotone conjunctions, $\mathrm{L}_{f,D}(g) \geq 1/n$). Thus, we get Corollary 5.4 below; compare with Corollary 5.2.

Corollary 5.4. *The class of monotone conjunctions under distribution, D_1 (defined above), and with classification error, ℓ_c, as loss function, requires $\Omega(n/\log(n))$ generations for evolution in Valiant's model (without recombination).*

Lower Bound for Homogeneous Linear Separator

Let $\mathcal{U}_n$ be the uniform distribution over the unit sphere in n dimensions. It is well known that the ϵ-net for homogeneous linear separators, with respect to classification error, ℓ_c, and distribution, $\mathcal{U}_n$, has size at least $(1/\epsilon)^n$. Thus, we can prove Corollary 5.5; compare with Corollary 5.3 (note that uniform distribution over the unit sphere is a radially symmetric distribution).

Corollary 5.5. *The class of homogeneous linear separators in $\mathbb{R}^n$, H_n, under the uniform distribution over the unit sphere, $\mathcal{U}_n$, and with classification error, ℓ_c, as loss function, requires $\Omega(n/\log(n/\epsilon))$ generations for evolution in Valiant's model (without recombination).*

5.7 Discussion

The results in this chapter suggest that in principle recombination can accelerate evolution through parallelism. The results should be understood as demonstrating that for certain definitions of selection rules and recombination, polylogarithmic generations are sufficient for evolution whenever a suitable parallel learning algorithm exists. In this section, we discuss in some detail the choices made in the model of recombination.

The recombinator operator defined here simultaneously captures both mutation and recombination. In a sexually reproducing species, only mutations that occur on the germ line matter and natural selection never acts on mutations alone, but simultaneously also on recombination. We have defined selection of a descendant based on performance comparison with its parents. We take the view that if the parents were able to survive, descendants having performance no worse than their parents should also be able to survive. This definition is different from those common in population genetics where survival probability is determined by relative fitness, thus capturing competition within individuals in a species. However, the mapping from performance (or loss) with respect to a functionality and fitness as defined by the average number of progeny of a particular genotype that survive may be an extremely complicated one. Indeed, mapping of fitness to functional traits of organisms in a quantifiable manner seems to be a very difficult problem.

Also, the sharp thresholds we use to decide beneficial vs neutral, optimal vs suboptimal, feasible vs infeasible do seem unnatural. However, the only thing that can be said with certainty is that mapping of performance with respect to a certain functionality and fitness (as in survival of genotype) should be monotonically related. Sharp threshold offer the advantage that they are robust to any such monotone mapping.

The population model (with respect to selection) we proposed in this chapter is most realistic when the population is large and distributed over a large area, selective advantage is weak and similar for roughly all beneficial mutations. If selective forces for some mutations are strong, then these mutation are likely to spread in the population quickly, with or without recombination. Maynard Smith [34, 35] has a detailed discussion regarding conditions when sexual reproduction and recombination may accelerate evolution.

Chapter 6

Limitations on Evolvability

In this chapter, we study some of the limitations that must exist on classes of functions that are evolvable within *reasonable* (polynomially-bounded) resources. Some of the limitations arise due to purely statistical reasons, *i.e.* not enough information may be obtained through the processes of mutation (or recombination) and selection to evolve a function that is (almost) accurate. These results use the fact that any concept class that is evolvable is also learnable in the statistical query framework (and in some cases the CSQ framework). These results are part of existing literature and have been summarized in Section 6.3.

The question we mainly focus on is: are there *strictly* computational limitations on evolvability? Alternatively, if the evolutionary process is allowed to use unbounded *computational* resources, but only polynomially bounded *statistical* (or information-theoretic) resources, are more concept classes evolvable? We show that the answer to this question is indeed positive. These results are presented in the framework of statistical query (SQ) and correlational statistical query (CSQ) learning.

In Section 6.1, we review the SQ and CSQ models and delineate the computational and statistical aspects of these models. In Section 6.2, we precisely define the statistical and computational constraints imposed on the evolutionary process in Valiant's model. We show that there is a direct correspondence between computational limitations on evolvability and computational limitations on SQ/CSQ learning. This follows from the reduction from learning algorithms to evolutionary mechanisms described in Section 3.8.1. Section 6.4 contains proofs of the main results. In Section 6.5, we quantify computational resources that are sufficient for learning in the SQ framework, whenever statistically efficient learning is possible.

6.1 Statistical Query Learning Models

In this section, we describe several variations of SQ models considered in this chapter. We recall notation described in Chapter 2. In this chapter, we focus only on *boolean* concept classes, so we repeat the definitions of statistical query learning (only for boolean functions) in the interest of clarity.

Let $\mathcal{X}$ be the instance space and suppose that n characterizes the representation size of each element $x \in \mathcal{X}$ (e. g. $\mathcal{X} = \{-1, 1\}^n$ or $\mathcal{X} = \mathbb{R}^n$). A concept class, C, over $\mathcal{X}$ is a subset of boolean functions defined over $\mathcal{X}$, where each concept, $c \in C$, is represented as a binary string; it is required that there exist an efficient Turing machine that outputs $c(x)$, given $c \in C$ and $x \in \mathcal{X}$ as inputs (cf. [23]). Let D be a distribution over $\mathcal{X}$. In the SQ model [22], the learning algorithm has access to a statistical query oracle, $\mathsf{STAT}(f, D)$, to which it can make a query of the form (ψ, τ), where $\psi : \mathcal{X} \times \{-1, 1\} \to [-1, 1]$ is the query function and τ is the (inverse polynomial) tolerance[1] The oracle responds with a value v such that $|\mathbb{E}_D[\psi(x, f(x))] - v| \leq \tau$, where $f \in C$ is the target concept. The goal of the learning algorithm is to output a hypothesis, $h : \mathcal{X} \rightarrow \{-1, 1\}$, such that $\mathrm{err}_D(h, c) = \Pr_{x \sim D}[h(x) \neq c(x)] \leq \epsilon$.

We use the following characterization of the SQ model due to [7] (see also [12]): A statistical query $\psi : \mathcal{X} \times \{-1, 1\} \rightarrow [-1, 1]$ is said to be *target-independent* if $\psi(x, b) \equiv \psi^{\mathrm{ti}}(x)$ for some function $\psi^{\mathrm{ti}} : \mathcal{X} \rightarrow [-1, 1]$. A statistical query is said to be *correlational* if $\psi(x, y) \equiv y\psi^{\mathrm{cor}}(x)$ for some function $\psi^{\mathrm{cor}} : \mathcal{X} \rightarrow [-1, 1]$. [7] showed that any statistical query (ψ, τ) (in Kearns' model) can be replaced by two queries, one of which is target-independent and the other correlational, each with tolerance $\tau/2$. We denote an oracle that accepts only target-independent or correlational queries as $\mathsf{SQ}\text{-}\mathcal{O}(f, D)$.

Formally, let $\mathsf{SQ}\text{-}\mathcal{O}(f, D)$ denote the statistical query oracle, which on receiving a target-independent query $(\psi^{\mathrm{ti}}, \tau)$ (where $\psi^{\mathrm{ti}} : \mathcal{X} \rightarrow [-1, 1]$), responds with a value v such that $|\mathbb{E}_D[\psi^{\mathrm{ti}}(x)] - v| \leq \tau$ and on receiving a correlational statistical query $(\psi^{\mathrm{cor}}, \tau)$ (where $\psi^{\mathrm{cor}} : \mathcal{X} \rightarrow [-1, 1]$), responds with a value, v, such that $|\mathbb{E}_{x \sim D}[\psi^{\mathrm{cor}}(x) f(x)] - v| \leq \tau$, where f is the target concept.

6.1.1 Distribution-Specific SQ Learning

The first framework we consider is distribution-specific SQ learning. It is required that the running time of the algorithm is polynomial in the parameters n and $1/\epsilon$

[1]We have referred to τ as the approximation parameter in the rest of this thesis. This was to avoid confusion with the tolerance function in the context of evolvability. However, in this chapter we use the more conventional notation in learning theory, and refer to τ as the tolerance parameter.

and also that the queries made by the algorithm are efficiently evaluable[2] and use a tolerance parameter τ, that is lower-bounded by some inverse polynomial in n and $1/\epsilon$.

Definition 6.1 (Distribution-Specific SQ Learning). *Let $\mathcal{X}$ be the instance space (with representation size n), D a distribution over $\mathcal{X}$ and C a concept class over $\mathcal{X}$. We say that C is* distribution-specific *SQ learnable with respect to distribution D, if there exists a randomized algorithm,* Alg*, that for every $\epsilon, \delta > 0$, every target concept $f \in C$, with access to oracle* $\mathsf{SQ}\text{-}\mathcal{O}(f, D)$*, outputs with probability at least $1 - \delta$, a hypothesis, h, such that $\mathrm{err}_D(h, f) \leq \epsilon$. Furthermore, the running time of the algorithm must be polynomial in n and $1/\epsilon$ and $1/\delta$ and the queries made to the oracle and the output hypothesis must be polynomially evaluable and have a tolerance τ that is lower-bounded by an inverse polynomial in n, $1/\epsilon$.*

To distinguish the computational complexity and information-theoretic (or statistical) complexity of SQ learning, we use the notion of *query complexity*. The query complexity for learning concept class C is the minimum number of queries required by any (possibly unbounded) algorithm to learn C in the SQ model. It is worthwhile to point that when defining query complexity, it is not required that the queries be efficiently evaluable and need not even have a small representation.

Definition 6.2 (Distribution-Specific SQ Query Complexity). *Let $\mathcal{X}$ be the instance space (with representation size n), D a distribution over $\mathcal{X}$ and C a concept class over $\mathcal{X}$. We say that the* query complexity *of learning C under distribution, D,*

[2]By this we mean that given a description of a query function ψ and it's input x, there exists an efficient Turing machine that outputs $\psi(x)$.

to accuracy ϵ, is bounded by q if there exists a (possibly computationally unbounded) algorithm that for every concept $f \in C$, makes at most q queries to oracle $\mathsf{SQ}\text{-}\mathcal{O}(f, D)$ outputs a hypothesis, h, such that $\mathrm{err}_D(h, f) \leq \epsilon$. The tolerance, τ, for the queries must be lower-bounded by an inverse polynomial in n and $1/\epsilon$.

Distribution-Specific Weak SQ Learning

In the case of weak learning, the learning algorithm is required only to output a hypothesis whose error is at most $1/2 - \gamma$, where $1/\gamma$ is bounded by a polynomial in n. The definitions of distribution-specific weak SQ learning and distribution-specific weak query complexity are identical to the definitions above (except for the requirement on the error of the output hypothesis).

6.1.2 Distribution-Independent SQ Learning

The second framework we consider is distribution-independent SQ learning, where the same learning algorithm is required to output an accurate hypothesis for all distributions. As in the case of distribution-specific learning, the requirement is that the running time of the algorithm be polynomial in n and $1/\epsilon$, the queries made by the algorithm be polynomially evaluable and use a tolerance parameter τ, that is lower-bounded by some inverse polynomial in n and $1/\epsilon$.

Definition 6.3 (Distribution-Independent SQ Learning)**.** *Let $\mathcal{X}$ be the instance space (with representation size n) and C a concept class over $\mathcal{X}$. We say that C is* distribution-independently *SQ learnable if there exists a randomized algorithm,* Alg*, that for every $\epsilon, \delta > 0$, every target concept $f \in C$ and for every distribution, D, over*

$\mathcal{X}$, with access to oracle, $\mathsf{SQ}\text{-}\mathcal{O}(f, D)$, outputs with probability at least $1 - \delta$, a hypothesis, h, such that $\mathrm{err}_D(h, f) \leq \epsilon$. Furthermore, the running time of the algorithm must be polynomial in n and $1/\epsilon$ and $1/\delta$ and the queries made to the oracle and the output hypothesis must be polynomially evaluable and have a tolerance, τ, that is lower-bounded by an inverse polynomial in n, $1/\epsilon$.

As in the case of distribution-specific SQ learning, to distinguish between computational complexity and information-theoretic (or statistical) complexity of learning, we use the notion of distribution-independent SQ query complexity. We re-iterate that the queries themselves are not required to be polynomially evaluable and need not even have a polynomial-size representation. However, the number of queries made and the inverse of the tolerance parameter, τ, must be bounded by some polynomial in n and $1/\epsilon$.

Definition 6.4 (Distribution-Independent SQ Query Complexity)**.** *Let $\mathcal{X}$ be the instance space (with representation size n) and C a concept class over $\mathcal{X}$. We say that the distribution-independent* query complexity *of learning C to accuracy ϵ is bounded by q, if there exists a (possibly computationally unbounded) algorithm that for every concept $f \in C$, every distribution D over $\mathcal{X}$ makes at most q queries to oracle $\mathsf{SQ}\text{-}\mathcal{O}(f, D)$ outputs a hypothesis, h, such that $\mathrm{err}_D(h, f) \leq \epsilon$. The tolerance τ for the queries must be lower-bounded by an inverse polynomial in n and $1/\epsilon$.*

In the case of distribution-independent SQ learning, weak and strong learning are equivalent (cf. [1]), hence we do not consider the distribution-independent weak learning model.

Access to Random Examples

In the distribution-independent SQ learning setting, we also consider the variant where the learning algorithm has access to random *unlabeled* examples from the distribution, D. In the case of distribution-specific SQ learning, this model is not particularly interesting, since we may assume instead that the learning algorithm has a reasonably large (polynomial size) sample provided as (non-uniform) advice. The lower bounds shown in Section 6.4 hold even in the model where the learning algorithms have access to random unlabeled examples from the distribution. The upper bound proved in Section 6.5 only holds if the learning algorithm has access to such examples from the distribution.

6.1.3 Correlational Statistical Query Learning

The correlational statistical query (CSQ) learning model was introduced by Feldman [12] and he showed that this model is equivalent to Valiant's evolution model (with boolean ideal functions and boolean representations). In the CSQ model, the learner is only allowed to make statistical queries that are *correlational*. Let (ψ, τ) be a query, where $\psi : \mathcal{X} \rightarrow [-1, 1]$ is the query function and τ is the (inverse polynomial) tolerance factor. A CSQ oracle, $\mathsf{CSQ}\text{-}\mathcal{O}(f, D)$, responds with a value v such that $|E_{x \sim D}[\psi(x) f(x)] - v| \leq \tau$, where $f \in C$ is the target concept.

Distribution-specific CSQ learning and distribution-specific SQ learning are essentially equivalent, as long as the learning algorithm has a random sample from the distribution as non-uniform advice[3]. Thus, we do not consider the case of distribution-

[3]See [12] for more details.

specific CSQ learning separately. Also, in the case of CSQ learning, it does not make sense to consider models which have access to random unlabeled examples, since this would make it the same as SQ learning.

Definition 6.5 (Distribution-Independent CSQ Learning). *Let $\mathcal{X}$ be the instance space (with representation size n) and C a concept class over $\mathcal{X}$. We say that C is* distribution-independently *CSQ learnable if there exists a randomized algorithm,* Alg*, that for every $\epsilon, \delta > 0$, every target concept, $f \in C$ and for every distribution, D over $\mathcal{X}$, with access to oracle, $\mathsf{CSQ}\text{-}\mathcal{O}(f, D)$, outputs with probability at least $1 - \delta$, a hypothesis, h, such that $\mathrm{err}_D(h, f) \leq \epsilon$. Furthermore, the running time of the algorithm must be polynomial in n, $1/\epsilon$ and $1/\delta$ and the queries made to the oracle and the output hypothesis must be polynomially evaluable and have a tolerance τ, that is lower-bounded by a polynomial in n, $1/\epsilon$.*

As in the previous cases, one can define the *distribution-independent query complexity* of CSQ learning. This captures the information-theoretic complexity of CSQ learning.

Definition 6.6 (Distribution-Independent CSQ Query Complexity). *Let $\mathcal{X}$ be the instance space (with representation size n) and C a concept class over $\mathcal{X}$. We say that the distribution-independent* query complexity *of learning C to accuracy ϵ is bounded by q, if there exists a (possibly computationally unbounded) algorithm that for every concept, $f \in C$, every distribution, D over $\mathcal{X}$, makes at most q queries to oracle, $\mathsf{CSQ}\text{-}\mathcal{O}(f, D)$ outputs a hypothesis, h, such that $\mathrm{err}_D(h, f) \leq \epsilon$. The tolerance, τ, for the queries must be lower-bounded by an inverse polynomial in n and $1/\epsilon$.*

6.1.4 Notation

Finally, we introduce some additional notation used in the rest of this chapter.

Note that all binary strings are strings over $\{-1, 1\}$ and boolean functions have range $\{-1, 1\}$. Let $[k]$ denote the set $\{1, 2, \ldots, k\}$ and for any k bit string, z, let $S(z) = \{i \mid z_i = -1\} \subseteq [k]$. Define $\mathsf{MAJ}_z : \{-1, 1\}^k \to \{-1, 1\}$ to be the majority function over $S(z)$. Formally for $x \in \{-1, 1\}^k$, $\mathsf{MAJ}_z(x) = -1$ if $|S(x) \cap S(z)|/|S(z)| \geq 1/2$ and $\mathsf{MAJ}_z(x) = 1$ otherwise. Define $\mathsf{PAR}_z : \{-1, 1\}^k \to \{-1, 1\}$ as the parity function over the bits in $S(z)$. Formally, for $x \in \{-1, 1\}^k$, $\mathsf{PAR}_z(x) = -1$ if $|S(x) \cap S(z)|$ is odd and $\mathsf{PAR}_z(x) = 1$ if $|S(x) \cap S(z)|$ is even, i.e. $\mathsf{PAR}_z(x) = \prod_{i \in S(z)} x_i$. Define $\mathsf{OR}_z : \{-1, 1\}^k \to \{-1, 1\}$ as the OR function over the bits in $S(z)$. Formally, for $x \in \{-1, 1\}^k$, $\mathsf{OR}_z(x) = -1$ if $|S(x) \cap S(z)| \geq 1$ and $\mathsf{OR}_z(x) = 1$ otherwise.

Fourier Analysis

Under the uniform distribution over $\{-1, 1\}^n$, the set of parity functions, $\langle \mathsf{PAR}_z \rangle_{z \in \{-1,1\}^n}$ forms an orthonormal basis (Fourier basis) for real-valued functions defined over $\{-1, 1\}^n$. For a function $f : \{-1, 1\}^n \to \mathbb{R}$, $\hat{f}(S(z)) = \mathbb{E}_{x \sim U_n}[f(x)\mathsf{PAR}_z(x)]$ is the Fourier coefficient of f corresponding to the subset $S(z)$; here U_n denotes the uniform distribution over $\{-1, 1\}^n$. Fourier analysis has been used extensively for learning with respect to the uniform distribution (for a survey see [28]). In particular, Kushilevitz and Mansour showed that with blackbox access to a function f, all Fourier coefficients of f with magnitude at least θ, can be obtained in time $\text{poly}(n, 1/\theta)$ [25]. We refer to this as the KM algorithm and use it frequently in our proofs.

6.2 Computational Limitations on Evolvability

Recall from Chapter 3, that an evolutionary mechanism is defined as a 5-tuple, $\mathcal{E}M = (R, \mathsf{Mut}, \tau, s, t)$. We required that all components of an evolutionary mechanism be polynomially bounded. Of these the statistical (polynomial) bounds are the following:

- The approximation parameter, τ, indicates how accurate an estimate of the true expected loss of a representation, r, may be obtained by accessing the $\widehat{\mathrm{L}}_{f,D}(\cdot,\cdot)$ oracle. We require that this is be bounded by a polynomial in n and $1/\epsilon$. We required that the tolerance function, t, may be polynomially sandwiched. However, allowing the tolerance function to superpolynomially small is of no advantage if $1/\tau$ is polynomially bounded.

- The size function, s, determines the size of the neighborhood of some representation, r, that is explored at each evolutionary step.

- The number of generations (or evolutionary steps) must be at most some polynomial in n and $1/\epsilon$.

The statistical bounds are also computational ones. On the other hand, there are also purely computational polynomial bounds imposed upon the model:

- That the representations in R have size bounded by a polynomial in n and $1/\epsilon$ and also that these representations be polynomially evaluable.

- The mutator be a polynomial-time randomized Turing machine.

- That the size function, s, and tolerance function, t, be efficiently evaluable.

The question we consider is: if the statistical bounds on the model of evolvability are retained, but it is allowed unbounded computational power, are more concept classes evolvable?

We relate the above question to that of computational complexity vs query complexity in the statistical query models. The reduction from learning algorithms to evolutionary mechanism in Section 3.8.1 allows us to observe the following:

1. A computationally efficient statistical query algorithm leads to a computationally and statistically polynomially bounded evolutionary mechanism.

2. A statistical query algorithm with polynomial query complexity can be reduced to an evolutionary mechanism that is polynomially bounded in the statistical properties, but may otherwise be computationally unbounded. This is because the number of queries made by the algorithm translate into the size of the neighborhood and the number of generations. On the other hand, evaluating the representations requires simulating the statistical query algorithm (which may be unbounded), thus the evolutionary mechanism may use possibly unbounded computation. Note that the converse is also true, *i.e.* if an evolutionary mechanism is statistically bounded, then it may be simulated in the SQ framework with polynomial query complexity.

Thus, in the rest of the chapter, we use the language of statistical query frameworks and separate concept classes that are efficiently learnable from those which have polynomial query complexity.

6.3 Statistical Lower Bounds

We summarize some statistical lower bounds that are known in the statistical query framework. Kearns in his original paper on statistical query learning had already demonstrated that parity functions require an exponential number of queries for learning [22]. Blum et al. proved that the number of statistical queries required for weak learning a concept class, C, over some distribution, D, is characterized by a relatively simple combinatorial parameter of C, called the *statistical query dimension* [6]. The statistical query dimension measures the maximum number of *nearly uncorrelated* (relative to distribution D) functions in C. These bounds for weak learning were strengthened and extended to other variants by Bshouty and Feldman [7], Blum, Kalai and Wasserman [5], Yang [42], and Feldman [12].

More recently, Simon [32] described a characterization of strong SQ learning with respect to a fixed distribution, D. Simpler and stronger characterizations were subsequently derived by Feldman [13] and Szörényi [37]. All of these characterizations are in the fixed distribution setting, and the corresponding question in the distribution-independent case is still open. Using these characterizations, Feldman, Lee, and Servedio [17] have shown statistical lower-bounds for learning shallow (in particular, depth-3) monotone formulas.

All of these results are based on information-theoretic arguments and hence, are unconditional.

6.4 Computational Lower Bounds

For the sake of completeness, we first state the result that for weak distribution-specific SQ/CSQ learning, the query complexity and computational complexity is essentially the same. This fact was observed by Blum et al. [6] and follows easily from their characterization of SQ learning. Next, we show that for distribution-specific (strong) SQ learning, distribution-independent SQ learning and distribution-independent (strong) CSQ learning, the query complexity is significantly different from the computational complexity. In the case of distribution-specific (strong) SQ learning and distribution-independent SQ learning, we show that there exists a concept class that has polynomial query complexity, but cannot be efficiently learned in the respective SQ model unless $\mathsf{RP} = \mathsf{NP}$. In the case of distribution-independent CSQ learning, the separation is based on a stronger assumption: we show that there is a concept class C with polynomial query complexity (of CSQ learning), but cannot be learned efficiently unless every problem in $\mathsf{W[P]}$ has a randomized fixed parameter tractable algorithm.

6.4.1 Weak Distribution-Specific SQ/CSQ Learning

Blum et al. [6] showed that weak distribution-specific SQ learnability of a concept class is characterized by a combinatorial parameter, the statistical query dimension – SQ-DIM(C, D), which characterizes the number of nearly uncorrelated concepts in C and observed that this implies the equivalence of query complexity and computational complexity in this model of learning. A short proof sketch of the following theorem is provided for completeness.

Theorem 6.1. *If a concept class C is weakly SQ learnable over a distribution D then there exists a polynomial-size circuit that weakly SQ learns C over a distribution D.*

Proof. SQ-DIM(C, D, γ) is the size of the largest subset $S \subseteq C$, such that for every $c_1, c_2 \in S$, $\mathrm{err}_D(c_1, c_2) = \Pr_{x \sim D}[c_1(x) \neq c_2(x)] \geq 1/2 - \gamma$. A simple weak-learning algorithm just tries every concept from S [4], and at least one of them will have error less than $1/2 - \gamma$ with the target function f (since S is the largest such subset, if this weren't the case adding the target function f to S would give a larger subset with the same property). Blum et al. [6] showed that SQ-DIM(C, D, γ) is polynomially related to the query complexity of weak SQ learning C under D to accuracy $1/2 - \gamma$. □

6.4.2 Strong Distribution-Specific SQ/CSQ Learning

Let $\phi \in \{-1, 1\}^m$ denote a 3-CNF formula over n variables (encoded as a string). Suppose ϕ is satisfiable and let $\zeta(\phi)$ denote the lexicographically first satisfying assignment of ϕ. Throughout this section $b \in \{-1, 1\}$, $x \in \{-1, 1\}^m$ and $x' \in \{-1, 1\}^n$ and let bxx' denote the $m + n + 1$ bit string obtained by concatenating b, x and x'. For $\phi \in \{-1, 1\}^m$, where ϕ is a satisfiable 3-CNF formula, define the function $f_{\phi,y} : \{-1, 1\}^{m+n+1} \to \{-1, 1\}$ as follows:

$$f_{\phi,y}(bxx') = \begin{cases} \mathsf{MAJ}_\phi(x) & \text{if } b = 1 \\ \mathsf{PAR}_y(x') & \text{if } b = -1 \end{cases}$$

In other words, $f_{\phi,y}$ is a function that over one half of the domain is the majority function, MAJ_ϕ, and over the other half of the domain is the parity function, PAR_y.

[4]Note that this is a non-uniform algorithm, since the set S needs to be given as advice to the algorithm

Note that the function $f_{\phi,y}$ is efficiently computable given the representation (ϕ, y). Define C_1 to be the following concept class:

$$C_1 = \{f_{\phi,\zeta(\phi)} \mid \phi \text{ is satisfiable}\}.$$

Theorems 6.2 and 6.3 show that the query complexity of C_1 is polynomial, but unless $\mathsf{RP} = \mathsf{NP}$ there is no polynomial time SQ algorithm for learning C_1. To prove that the query complexity is polynomial, the key idea is that the learning algorithm only needs to (proper) learn majorities in the SQ model, which is easy. The learning algorithm can recover ϕ and solve for $\zeta(\phi)$ (possibly using unbounded computation). Thus, $f_{\phi,\zeta(\phi)}$ can be exactly SQ learned using only polynomially many queries. On the other hand, we show that an efficient SQ learning algorithm for C_1 can be used to recover a satisfying assignment of the 3-CNF formula ϕ. The key point to note here is that parities are essentially invisible to statistical queries and hence the only way to learn C_1 is to obtain $\zeta(\phi)$ using ϕ, which is not possible unless $\mathsf{RP} = \mathsf{NP}$.

Theorem 6.2. *The query complexity of SQ learning C_1 with respect to the uniform distribution U is at most m.*

Proof. Let $f_{\phi,\zeta(\phi)} \in C_1$ be the target function. We show how to obtain ϕ using statistical queries. For $i \in \{1, \ldots, m\}$, define the function $\psi_i : \{-1,1\}^{m+n+1} \times \{-1,1\} \to [-1,1]$ as follows:

$$\psi_i(bxx', y) = \begin{cases} 0 & \text{if } b = -1 \\ x_i y & \text{if } b = 1 \end{cases}$$

Then, observe that $\mathbb{E}_{U_{m+n+1}}[\psi(bxy, f_{\phi,\zeta(\phi)}(bxy))] = (1/2)\mathbb{E}_{U_m}[x_i \mathsf{MAJ}_\phi(x)]$. It is well known (see for example [31]) that if $\phi_i = -1$ (i.e. the i^{th} bit is part of the majority function) then $\mathbb{E}_{U_m}[x_i \mathsf{MAJ}_\phi(x)] = \Omega(1/\sqrt{m})$ and 0 otherwise. Hence, by setting $\tau = \Theta(1/\sqrt{m})$, the query (ψ_i, τ) reveals the bit ϕ_i. Thus, using at most $m = O(n^3)$ statistical queries, we obtain ϕ. Now, it is easy to obtain (possibly using unbounded computation) the value $\zeta(\phi)$ and thus obtain the function, $f_{\phi,\zeta(\phi)}$. □

Theorem 6.3. *C_1 is not* efficiently *SQ learnable under the uniform distribution unless* $\mathsf{RP} = \mathsf{NP}$*.*

Proof. Suppose to the contrary that Alg is a (possibly randomized) algorithm that learns C_1 to error at most 0.1 (in fact, to any value noticeably lower than 1/4) in polynomial time. We show that using Alg it is possible (with high probability) to find a satisfying assignment to any 3-CNF formula ϕ, if one exists. Thus, failure to find a satisfying assignment implies that ϕ is unsatisfiable.

Let ϕ be a 3-CNF instance. Suppose ϕ is a satisfiable, so that $f_{\phi,\zeta(\phi)} \in C_1$; we show that in this case a solution to ϕ can be obtained with high probability. Suppose Alg makes q statistical queries each with tolerance τ to learn C_1. We show that we can simulate any statistical query (ψ, τ) with respect to $f_{\phi,\zeta(\phi)}$ efficiently. The queries made by Alg to the oracle $\mathsf{SQ}\text{-}\mathcal{O}$ may be target-independent or correlational. Below, we consider the two cases:

1. Let $(\psi^{\mathrm{ti}}, \tau)$ be a *target-independent* query; we need to return an (additive) τ-approximation to the value $\mathbb{E}_{U_{m+n+1}}[\psi(bxx')]$. This is easily achieved by drawing a sample of size $\tilde{O}(1/\tau^2)$ from the uniform distribution and returning the empirical estimate.

2. Let ψ^{cor} be the *correlational* query. In this case, we need to return an (additive) τ-approximation to the value $\mathbb{E}_{U_{m+n+1}}[\psi^{\mathrm{cor}}(bxx')f_{\phi,\zeta(\phi)}(bxx')]$. For $b \in \{-1, 1\}$,

define $\psi_b^{\mathrm{cor}}(xx') \equiv \psi^{\mathrm{cor}}(bxx')$. Then,

$$\mathbb{E}_{U_{m+n+1}}[\psi^{\mathrm{cor}}(bxx')f_{\phi,\zeta(\phi)}(bxx')] = \\ \frac{1}{2}\mathbb{E}_{U_{m+n}}[\psi_1^{\mathrm{cor}}(xx')\mathsf{MAJ}_\phi(x)] + \frac{1}{2}\mathbb{E}_{U_{m+n}}[\psi_{-1}^{\mathrm{cor}}(xx')\mathsf{PAR}_{\zeta(\phi)}(x')]$$

It suffices to find τ-approximations to both the terms in the above expression. To obtain a τ-approximate estimate of $\mathbb{E}_{U_{m+n}}[\psi_1^{\mathrm{cor}}(xx')\mathsf{MAJ}_\phi(x)]$, as in the earlier case, we can draw a sample of size $\tilde{O}(1/\tau^2)$ from U_{m+n} and return the empirical estimate (since we can efficiently compute the functions ψ_1^{cor} and MAJ_ϕ).

We show that either 0 is a τ-approximation to $\mathbb{E}_{U_{m+n}}[\psi_{-1}^{\mathrm{cor}}(xx')\mathsf{PAR}_{\zeta(\phi)}(x')]$ or we find a satisfying assignment to ϕ using Fourier analysis. Observe that $\mathbb{E}_{U_{m+n}}[\psi_{-1}^{\mathrm{cor}}(xx')\mathsf{PAR}_{\zeta(\phi)}(x')]$ is simply the Fourier coefficient of ψ_{-1}^{cor} corresponding to $\zeta(\phi)$ (or actually the set $S(\zeta(\phi)) = \{m + i \mid \zeta(\phi)_i = -1\} \subseteq [m + n]$). We know that all Fourier coefficients of ψ_{-1}^{cor} of magnitude larger than $\tau/2$ can be estimated to accuracy $\tau/4$ using the KM algorithm in time poly$(n, 1/\tau)$ (see Section 6.1.4 or [25]). Furthermore, the number of such coefficients is polynomial in n, $1/\tau$. We check whether any such coefficient (interpreted as a string of length n) is a satisfying assignment of ϕ. If we find an assignment, we are done; if not we know that the coefficient $|\widehat{\psi_{-1}^{\mathrm{cor}}}(S(\zeta(\phi)))| \le \tau$, since $\zeta(\phi)$ is a solution to ϕ and we would have identified it as such, had it been in the list of heavy coefficients. Thus, 0 is an (additive) τ-approximate estimate to the term $\mathbb{E}_{U_{m+n}}[\psi_{-1}^{\mathrm{cor}}(xx')\mathsf{PAR}_{\zeta(\phi)}(x')]$.

Thus, we have shown that we can either find a satisfying assignment to ϕ or simulate the SQ-$\mathcal{O}$ oracle response satisfactorily to all queries made by algorithm

Alg. In the latter case, the algorithm outputs h such that $\mathrm{err}_{U_{m+n+1}}(h, f_{\phi,\zeta(\phi)}) \leq 0.1$, i.e. $\mathbb{E}_{U_{m+n+1}}[h(bxx')f_{\phi,\zeta(\phi)}(bxx')] \geq 4/5$. Let $h_b(xx') \equiv h(bxx')$, then

$$\mathbb{E}_{U_{m+n+1}}[h(bxx')f_{\phi,\zeta(\phi)}(bxx')] = \frac{1}{2}\mathbb{E}_{U_{m+n}}[h_1(xx')\mathsf{MAJ}_\phi(x)]+\frac{1}{2}\mathbb{E}_{U_{m+n}}[h_{-1}(xx')\mathsf{PAR}_{\zeta(\phi)}(x')]$$

The above equation implies that $\mathbb{E}_{U_{m+n}}[h_{-1}(xx')\mathsf{PAR}_{\zeta(\phi)}(x')] = \hat{h}_{-1}(S(\zeta(\phi))) \geq 3/5$, where $\hat{h}_{-1}(S(\zeta(\phi)))$ is the Fourier coefficient of h_{-1} corresponding to the set $S(\zeta(\phi))$. Thus identifying all large coefficients of h_{-1}, by the KM algorithm, and checking whether any of the coefficients (when interpreted as a string of length n) satisfies ϕ, a satisfying assignment of ϕ is obtained (since $\zeta(\phi)$ has a large Fourier coefficient).

Thus, if ϕ is satisfiable, using Alg it is possible to find, with high probability, a satisfying assignment to ϕ. If we fail to find the satisfying assignment, then ϕ is unsatisfiable. Hence, an algorithm to efficiently SQ learn C_1 does not exist unless $\mathsf{RP} = \mathsf{NP}$. □

6.4.3 Strong Distribution-Independent SQ Learning

In this section, we consider the distribution-independent SQ learning model. As in the case of distribution-specific SQ/CSQ learning, we construct a concept class, C_2, such that C_2 is distribution-independently SQ learnable, but not *efficiently* distribution-independently SQ-learnable unless $\mathsf{RP} = \mathsf{NP}$.

Using the notation from Section 6.4.2 define $g_{\phi,y}$ as:

$$g_{\phi,y}(xx') = \begin{cases} PAR_y(x') & x = \phi \\ 1 & \text{otherwise} \end{cases}$$

Thus, $g_{\phi,y}$ is the function that equals $\mathsf{PAR}_y(x')$ on the part of the domain that has ϕ as the prefix and is the constant function 1 otherwise. Define the concept class C_2

as follows:

$$C_2 = \{g_{\phi,\zeta(\phi)} \mid \phi \text{ is satisfiable}\}.$$

First, we show that the distribution-independent query complexity of SQ learning C_2 is bounded by a polynomial in n. The key idea is that either the constant function 1 is an accurate predictor (if the distribution has almost no mass on points that have ϕ as a prefix), or else it is possible to recover the 3-CNF formula ϕ using statistical queries, and then (using possibly unbounded computation) the assignment $\zeta(\phi)$ can be obtained to learn $g_{\phi,\zeta(\phi)}$ exactly. On the contrary, we show that C_2 cannot be efficiently learned in the distribution-independent SQ model unless $\mathsf{RP} = \mathsf{NP}$. As in the previous case, we show that an efficient SQ algorithm for learning C_2 can be used to find a satisfying assignment to any 3-CNF formula ϕ, if it exists. These results are proved formally as Theorems 6.4 and 6.5.

Theorem 6.4. *The distribution-independent query complexity of SQ learning C_2 is at most $2m+1$.*

Proof. Let $g_{\phi,\zeta(\phi)}$ be the target function, D the target distribution and let $\epsilon > 0$ be the target error rate. We first test if the hypothesis, the constant 1 function, is ϵ-accurate. This can be tested using a single correlational statistical query $(1, \epsilon/4)$. If the value returned is at least $1 - 3\epsilon/4$, then $\mathbb{E}_D[g_{\phi,\zeta(\phi)}(xx')] \geq 1 - \epsilon$, i.e. the constant 1 hypothesis is ϵ-accurate. If not, we know that $g_{\phi,\zeta(\phi)}$ is -1 on at least $\epsilon/4$ fraction of the domain (under the target distribution D).

Now, suppose that $\Pr_D[g_{\phi,\zeta(\phi)}(xx') = -1] \geq \epsilon/4$. For $i = 1, \ldots, m$, define $\psi_i : \{-1,1\}^{m+n} \rightarrow [-1,1]$ as the following function: $\psi_i(xx') = 1$, if $x_i = 1$ and $\psi_i(xx') = 0$ otherwise.

Consider the following expectation,

$$\mathbb{E}_D[\psi_i(xx') - g_{\phi,\zeta(\phi)}(xx')\psi_i(xx')]$$

If $\phi_i = -1$ (i.e. the i^{th} bit of the representation of the 3-CNF formula ϕ is -1), then $g_{\phi,\zeta(\phi)}(xx') = 1$ for all points where $\psi_i(xx') \neq 0$. This is because $g_{\phi,\zeta(\phi)}$ is the constant 1 function on points which do not have ϕ as a prefix, and if $\psi_i(xx') \neq 0$, then $x_i = 1 \neq \phi_i$. Thus, for all points $\psi_i(xx') = g_{\phi,\zeta(\phi)}(xx')\psi_i(xx')$ and hence the value of the above expectation is exactly 0.

On the other hand, if $\phi_i = 1$, then whenever $g_{\phi,\zeta(\phi)}(xx') = -1$, $\psi_i(xx') = 1$. When $g_{\phi,\zeta(\phi)}(xx') = 1$, $\psi_i(xx') - g_{\phi,\zeta(\phi)}(xx')\psi_i(xx') = 0$. Recall that $\Pr_D[g_{\phi,\zeta(\phi)}(xx') = -1] \geq \epsilon/4$, thus the above expectation is at least $\epsilon/2$.

As $\mathbb{E}_D[\psi_i(xx') - g_{\phi,\zeta(\phi)}(xx')\psi_i(xx')] = \mathbb{E}_D[\psi_i(xx')] - \mathbb{E}_D[g_{\phi,\zeta(\phi)}(xx')\psi_i(xx')]$, an $\epsilon/8$ accurate estimate to the above expectation can be obtained by making a target independent query $(\psi_i, \epsilon/16)$ and a correlational query $(\psi_i, \epsilon/16)$. Thus, by looking at the query responses it is possible to determine whether $\phi_i = 1$ or $\phi_i = -1$.

Using $2m$ queries, each bit of ϕ can be determined, and then $\zeta(\phi)$ can be obtained, if necessary by brute force, to output $g_{\phi,\zeta(\phi)}$. □

Theorem 6.5. *C_2 is not* efficiently *distribution-independently SQ learnable unless* $\mathsf{RP} = \mathsf{NP}$.

Proof. Suppose to the contrary and let Alg be a (possibly randomized) algorithm that efficiently learns C_2 in the distribution-independent SQ model. We show that if ϕ is a satisfiable 3-CNF formula then, using Alg, a satisfying assignment can be constructed with high probability.

Let ϕ be a 3-CNF formula that is satisfiable, so that $g_{\phi,\zeta(\phi)} \in C_2$. Let D_2 be the distribution defined as follows: $D_2(xx') = 2^{-n}$ if $x = \phi$, $D_2(xx') = 0$ otherwise; thus, D_2 is the uniform distribution on strings of the form $\phi x'$.

Let $g_{\phi,\zeta(\phi)}$ be the target concept from C_2 and D_2 the target distribution. Suppose $\epsilon \leq 1/4$. We run Alg to learn $g_{\phi,\zeta(\phi)}$. We need to show that we can simulate the queries made by Alg to the oracle $\mathsf{SQ}\text{-}\mathcal{O}(g_{\phi,\zeta(\phi)}, D_2)$.

As in the proof of Theorem 6.3, response to a target-independent query can be simulated by drawing a sample from D_2 of size $\tilde{O}(1/\tau^2)$ and returning the empirical estimate. In the case of correlational queries also, the main idea is similar to that used in the proof of Theorem 6.3. Let $(\psi^{\mathrm{cor}}, \tau)$ be a correlational query, define $\psi^{\mathrm{cor}}_\phi : \{-1,1\}^n \to [-1,1]$ to be the function $\psi^{\mathrm{cor}}_\phi(x') = \psi^{\mathrm{cor}}(\phi x')$. Thus, $\mathbb{E}_{D_2}[\psi^{\mathrm{cor}}(xx')g_{\phi,\zeta(\phi)}(xx')] = \mathbb{E}_{U_n}[\psi^{\mathrm{cor}}_\phi(x')\mathsf{PAR}_{\zeta(\phi)}(x')]$. This is just the Fourier coefficient of ψ^{cor}_ϕ on the subset $S(\zeta(\phi))$. Thus, we obtain all large (of magnitude greater than $\tau/2$) Fourier coefficients of ψ^{cor}_ϕ and check whether any of them (i.e. their string representations of length n) are a satisfying assignment to ϕ. If not, then 0 is valid (τ-approximate) answer to the query $(\psi^{\mathrm{cor}}, \tau)$.

Thus, we can simulate access to the $\mathsf{SQ}\text{-}\mathcal{O}(g_{\phi,\zeta(\phi)}, D_2)$ oracle to Alg or else we find a satisfying assignment to ϕ. Suppose we don't find a satisfying assignment to ϕ and Alg runs to completion, then for the output hypothesis, h, $\mathrm{err}_{D_2}(h, g_{\phi,\zeta(\phi)}) \leq 1/4$ or equivalently, $\mathbb{E}_{D_2}[h(xx')g_{\phi,\zeta(\phi)}(xx')] \geq 1/2$. Again define $h_\phi(x') = h(\phi x')$, so that $\mathbb{E}_{U_n}[h_\phi(x')\mathsf{PAR}_{\zeta(\phi)}(x')] \geq 1/2$. Thus, looking at the heavy Fourier coefficients of h_ϕ reveals a satisfying assignment to ϕ. The above algorithm works correctly with high probability.

If we are unable to find a satisfying assignment of ϕ, then we report ϕ as being unsatisfiable. Thus, we get a randomized polytime algorithm for 3-CNF. $\square$

6.4.4 Strong Distribution-Independent CSQ Learning

Showing a separation between the computational complexity and query complexity of distribution-independent CSQ learning is significantly more involved. The separation in this case is based on a stronger assumption: $\mathsf{W[P]}$ does not have randomized fixed parameter tractable algorithms. A *fixed parameter tractable algorithm* for a decision problem (x, k) is allowed to take running time $f(k)p(|x|)$ where p is a polynomial and f is an arbitrary function. A complete problem for $\mathsf{W[P]}$ is weighted circuit satisfiability, i.e. given a circuit ϕ and parameter k, does there exist a satisfying assignment of Hamming weight k? It is widely believed that $\mathsf{W[P]}$ does not have randomized fixed-parameter tractable algorithms and such an algorithm would also imply a subexponential time algorithm for circuit satisfiability (see [11]).

The construction relies on Feldman's recent result [15], where he shows that the class of disjunctions cannot be learned (for information theoretic reasons) in the distribution-independent CSQ model. The class of disjunctions on the other hand is weakly learnable in the distribution-independent CSQ model [12]. Unlike in the case of distribution-independent SQ model, this fact is required[5] because any algorithm that only uses *correlational* statistical queries can only get information about the distribution by first finding some function that is (at least weakly) correlated with the target function under that distribution.

[5]Note that the class of parities is not weakly learnable in the SQ model.

Let $\phi \in \{-1,1\}^m$ denote a circuit (represented as a string) with n input variables. For some parameter ℓ, let $\zeta(\phi)$ denote the lexicographically first satisfying assignment of Hamming weight ℓ. Let $n' = 3\ell n$ and let $\mathsf{Enc} : \{-1,1\}^n \to \{-1,1\}^{n'}$ be an encoding such that for any string $s \in \{-1,1\}^n$ with ℓ "-1" bits, $\mathsf{Enc}(s) \in \{-1,1\}^{n'}$ has 3ℓ "-1" bits. Furthermore, recovering any ℓ of these 3ℓ "-1" bits of $\mathsf{Enc}(s)$ allows us to reconstruct s. Such encodings can be constructed using Reed-Solomon codes and are defined below. We will explain shortly the necessity for these codes for our construction. Let $\xi(\phi) = \mathsf{Enc}(\zeta(\phi))$. Let $y \in \{-1,1\}^{n'}$ (recall that $n' = 3\ell n$), let $x \in \{-1,1\}^m$, $x' \in \{-1,1\}^{n'}$ and define $c_{\phi,y} : \{-1,1\}^{m+n'} \to \{-1,1\}$ as follows:

$$c_{\phi,y}(xx') = \begin{cases} \mathsf{OR}_y(x') & \text{if } x = \phi \\ 1 & \text{otherwise} \end{cases}$$

Define the concept class.

$$C_3 = \{c_{\phi,\xi(\phi)} \mid \phi \text{ has a satisfying assignment of Hamming weight at most } \ell\}.$$

Theorem 6.6 shows that the query complexity of CSQ learning C_3 is polynomial. This can be proved using the fact that OR is weakly learnable and by modifying Feldman's singleton learning algorithm [14]. This enables us to recover ϕ and the lexicographically first satisfying assignment of ϕ can be easily constructed (using unbounded computation). On the other hand, Theorem 6.7 shows that an efficient CSQ algorithm for learning C_3, implies a poly$(2^\ell, n)$ time for the weighted-circuit-SAT problem (given (ϕ, ℓ), does there exist a satisfying assignment for ϕ of Hamming weight ℓ?). The reduction requires us to set the accuracy of the learning algorithm to $O(2^{-\ell})$ and also allows us to only recover one-third of the bits of the hidden OR.

For this reason we need to use an OR that uses $\xi(\phi) = \mathsf{Enc}(\zeta(\phi))$ rather than $\zeta(\phi)$. Recovering a third of the bits of $\xi(\phi)$ is enough to reconstruct $\zeta(\phi)$.

We now provide more details required for the proof. We first show that the class of disjunctions is weakly learnable in the CSQ model (distribution-independently).

Weak Distribution-independent CSQ Learning Disjunctions

Let $\mathsf{DISJ}_n = \{\mathsf{OR}_z \mid z \in \{-1, 1\}^n\}$ be the class of disjunctions over n variables. Let $x \in \{-1, 1\}^n$ and let $x_1, x_2, \ldots, x_n$ be the input bits. Let $\mathcal{W} = \{-1, x_1, x_2, \ldots, x_n\}$ be a set of $n+1$ functions, where -1 is the constant function that is -1 everywhere, and x_i is the function $w(x) = x_i$. The following simple lemma shows that for every $z \in \{-1, 1\}^n$ and every distribution D over $\{-1, 1\}^n$, there exists $w \in \mathcal{W}$ such that $\mathbb{E}_D[\mathsf{OR}_z(x)w(x)] \geq 1/(2n)$. Thus, this implies that the class DISJ_n is efficiently weakly distribution-independently CSQ learnable. Feldman [12] gives a proof of this lemma, but we include a proof for completeness.

Lemma 6.1. *For every $\mathsf{OR}_z \in \mathsf{DISJ}_n$ and every distribution D over $\{-1, 1\}^n$, there exists $w \in \mathcal{W}$ such that $\mathbb{E}_D[\mathsf{OR}_z(x)w(x)] \geq 1/(2n)$.*

Proof. For a string $z \in \{-1, 1\}^n$, recall that $S(z) = \{i \mid z_i = -1\}$. Let $\beta_z(x) = \sum_{i \in S(z)} x_i - |S(z)| + 1$. Then observe that $\mathsf{OR}_z(x) = \text{sign}(\beta_z(x))$, since $\mathsf{OR}_z(x) = 1$ if all x_i such that $i \in S(z)$ are 1, in which case $\beta_z(x) = \sum_{i \in S(z)} x_i - |S(z)| + 1 = 1$, otherwise $\beta_z(x) = \sum_{i \in S(z)} x_i - |S(z)| + 1$ is at most -1.

Note that $\beta_z(x) = \sum_{i \in S(z)} x_i - |S(z)| + 1$ is always an odd integer, and hence $|\beta_z(x)| \geq 1$ for all x. Thus, for all x, $\beta_z(x)\,\text{sign}(\beta_z(x)) \geq 1$.

Then for any distribution D over $\{-1,1\}^n$ we have,

$$\mathbb{E}_{x\sim D}[\beta_z(x)\mathsf{OR}_z(x)] = \mathbb{E}_{x\sim D}[\beta_z(x)\,\mathrm{sign}(\beta_z(x))] \geq 1$$

Hence, either $\mathbb{E}_D[(-1)\cdot \mathsf{OR}_z(x)] \geq 1/(2(|S(z)|-1))$ or there exists $i \in S(z)$ such that $\mathbb{E}_D[x_i\mathsf{OR}_z(x)] \geq 1/(2|S(z)|)$. □

Encoding Sparse Strings

We give here a simple implementation of the encoding of sparse strings described above. Let s be a string of length n that contains at most ℓ "-1"-bits. We want to encode s as a string, $\mathsf{Enc}(s)$, of length $3\ell n$ that has at most 3ℓ "-1"-bits such that identifying any ℓ of the 3ℓ positions that have "-1" suffice to recover the string s. For a string s' of length $3\ell n$, let $\mathsf{Dec}(s')$ denote the (unique) string of length n, such that $\mathsf{Enc}(\mathsf{Dec}(s')) = s'$, or the null string if no such string exists.

For simplicity, let n be a power of 2, say $n = 2^k$. Given $s \in \{-1,1\}^n$ with at most ℓ "-1" bits, do the following: Identify the set $S = \{i \mid s_i = -1\}$; notice that $|S| = \ell$. We use the Reed-Solomon code to encode the elements of S using a set T, $|T| = 3\ell$ such that identifying any subset of T of size ℓ allows us to recover S. This is done by interpreting $i \in S$ as elements of the field $\mathbb{F}_{2^k}$ and constructing a polynomial of degree $\ell-1$, using elements of S as the coefficients. The set T contains an evaluation of this polynomial at 3ℓ different points in $\mathbb{F}_{2^k}$. Clearly, identifying any ℓ elements of T is enough to perform interpolation and hence obtain S. Now, we can encode T using a string of length $3\ell n$, with at most 3ℓ, "-1" bits as follows: Let $T = \{t_1, \ldots, t_{3\ell}\}$ and consider the string $s' = \mathsf{Enc}(s)$ as 3ℓ blocks of length n. In the i^{th} block, only the t_i^{th} bit is -1 and the rest are all 1. Notice, that although t_i are technically elements of

$\mathbb{F}_{2^k}$, they can be interpreted as integers less than n. Thus identifying the positions of any ℓ, "-1" bits of s' allows for decoding and recovering s. Denote by $\mathsf{Dec}(s')$ the string s, if s' is any string that has at least ℓ "-1" bits and must have been a (corrupted) version of $\mathsf{Enc}(s)$.

We can now prove the main result, stated as Theorems 6.6 and 6.7.

Theorem 6.6. *The distribution-independent CSQ query complexity of C_3 is at most* $\mathrm{poly}(n, 1/\epsilon)$

Proof. Let $c_{\phi,\xi(\phi)}$ be the target concept from C_3 and let $\epsilon > 0$ be the target error rate. We first test the hypothesis that is constant 1 everywhere. This can be tested using the correlational query $(1, \epsilon/4)$, where 1 is the constant 1 function. If the query response is greater than $1 - 3\epsilon/4$, then $\mathbb{E}_D[c_{\phi,\xi(\phi)}(xx')] \geq 1 - \epsilon$ and hence the constant 1 function is an ϵ-accurate hypothesis and we are done. Otherwise, $\Pr_D[c_{\phi,\xi(\phi)}(xx') = -1] \geq \epsilon/4$.

Let ϕ be the encoding of the 3-CNF-SAT formula corresponding to the target function $c_{\phi,\xi(\phi)}$. Let D_1 be the marginal distribution over the first m bits of the target distribution D. Suppose $h : \{-1, 1\}^m \to \{-1, 1\}$ is a function satisfying the two properties: (i) $h(\phi) = 1$, and (ii) $\Pr_{D_1}[h(x) = 1 \wedge x \neq \phi] \leq \epsilon/(100n)$.

Let D_2 be the distribution D conditioned on the first m bits being ϕ, i.e. $D_2(x') = D(\phi x')/D_1(\phi)$. Let $\mathcal{W}$ be as in Lemma 6.1 and let $w \in \mathcal{W}$ be such that $\mathbb{E}_{D_2}[\mathsf{OR}_{\xi(\phi)}(x')w(x')] \geq 1/(2n)$. Then, define the function $h^w(xx') = w(x')$ if $h(x) = 1$ and $h^w(xx') = 0$ otherwise. Note that,

$$\begin{aligned}\mathbb{E}_D[h^w(xx')c_{\phi,\xi(\phi)}(xx')] &\geq \Pr_{D_1}[x=\phi]\mathbb{E}_{D_2}[\mathsf{OR}_{\xi(\phi)}(x')w(x')] - \Pr_{D_1}[h^w(xx')=1 \wedge x \neq \phi] \\ &\geq \frac{\epsilon}{4}\cdot\frac{1}{2n} - \frac{\epsilon}{100n}\end{aligned}$$

Now define $h_i^w : \{-1,1\}^{m+n} \to [-1,1]$ to be the function, where $h_i^w(xx') = w(x')$ if $h(x) = 1$ and $x_i = 1$, and $h_i^w(xx') = 0$ otherwise. Now note that if $\phi_i = 1$, $\mathbb{E}_D[h_i^w(xx)c_{\phi,\xi(\phi)}(xx')] \geq \epsilon/(8n) - \epsilon/(100n)$, as in the previous case. On the other hand if $\phi_i = -1$, then $\mathbb{E}_D[h_i^w(xx')c_{\phi,\xi(\phi)}(xx')] \leq \epsilon/(100n)$.

This gap between the expectations in the two cases is large enough that the response to the correlational statistical query $(h_i^w, \epsilon/(100n))$ distinguishes the case when $\phi = 1$ and $\phi = -1$. Thus m such correlational queries can be used to exactly determine ϕ and then (possibly using unbounded computation) $\xi(\phi)$ may be determined to identify $c_{\phi,\xi(\phi)}$.

Now, suppose we did not know that h satisfied the properties (i) and (ii), mentioned above. We could still carry out the operations described above to come up with a candidate $\tilde{\phi}$ and guess $c_{\tilde{\phi},\xi(\tilde{\phi})}$ to be the target concept. We can then simply make the correlational query $(c_{\tilde{\phi},\xi(\tilde{\phi})}, \epsilon/4)$ to check whether $c_{\tilde{\phi},\xi(\tilde{\phi})}$ is an ϵ-accurate hypothesis. Note that if h did indeed satisfy the properties (i) and (ii), then $\tilde{\phi} = \phi$.

The last part required to complete the proof is to show that it is easy to construct a random hypothesis h that satisfies properties (i) and (ii) with non-negligible (inverse polynomial) probability. Then, several such hypotheses may be generated and each tested until the right one (or one that is good enough) is found. But, this is exactly what Feldman's algorithm for CSQ learning singletons does [14]. □

Theorem 6.7. *C_3 is not* efficiently *distribution-independently CSQ learnable, unless there exists a randomized algorithm that determines whether or not a given circuit ϕ, has a satisfying assignment of Hamming weight at most ℓ in time* $\mathrm{poly}(2^\ell, n)$.

Proof. Suppose that there exists an efficient algorithm, Alg, that distribution-independently CSQ learns C_3. Let ϕ be a circuit formula. We show that if ϕ has a satisfying assignment of Hamming weight at most ℓ, then using Alg we can find such a solution, with high probability.

Let $z = \xi(\phi)$, $S(z) = \{i \mid z_i = -1\}$ and suppose that $|S(z)| = k$, where $k \leq 3\ell$. Note that $\zeta(\phi)$ has Hamming weight at most ℓ. Then, the function OR_z can be expressed as the following polynomial.

$$\begin{aligned}\mathsf{OR}_z(x') &= -1 + 2 \prod_{i \in S(z)} \frac{1 + x'_i}{2} \\ &= -1 + 2^{-k+1} \sum_{T \subseteq S(z)} \chi_T(x')\end{aligned}$$

where $\chi_T(x')$ is the parity function over T. Let t_z be the polynomial,

$$t_z(x') = -1 + 2^{-k+1} + 2^{-k+1} \sum_{\substack{T \subseteq S(z) \\ |T| > k/3}} \chi_T(x')$$

Define D_z to be the distribution where $D_z(x') = |t_z(x')|/(\sum_{x'} |t_z(x')|)$. [?] showed that $\mathrm{sign}(t_z(x')) = \mathsf{OR}_z(x')$, and hence for all x', $D_z(x')\mathsf{OR}_z(x') = U_{n'}(x')t_z(x')$, where $U_{n'}$ is the uniform distribution over n' bits.

Define D to be the distribution over $\{-1, 1\}^{m+n'}$, where $D(xx') = D_z(x')$ if $x = \phi$ and $D(xx') = 0$ if $x \neq \phi$. Now, we run algorithm Alg to learn C_3 to accuracy $\epsilon = 2^{-k-2}$, where D is the target distribution and $c_{\phi,\xi(\phi)}$ is the target concept. We need to show that we can simulate oracle $\mathsf{CSQ}\text{-}\mathcal{O}$ for any query (ψ, τ). Let $\psi_\phi : \{-1,1\}^{n'} \rightarrow [-1,1]$ be the function where $\psi_\phi(x') = \psi(\phi x')$.

Note that,

$$\mathbb{E}_D[\psi(xx')c_{\phi,\xi(\phi)}(xx')] = \mathbb{E}_{D_z}[\psi_\phi(x')\mathsf{OR}_z(x')] = \mathbb{E}_{U_{n'}}[\psi_\phi(x')t_z(x')]$$

Then observe that,

$$\mathbb{E}_{U_{n'}}[\psi_\phi(x')t_z(x')] = (-1 + 2^{-k+1})\widehat{\psi_\phi}(\emptyset) + 2^{-k+1} \sum_{\substack{T \subseteq S(z) \\ |T| > k/3}} \widehat{\psi_\phi}(T)$$

Note that the only Fourier coefficients of ψ_ϕ that matter are those corresponding to the empty set and sets $T \subseteq S(z)$ such that $|T| \geq k/3$. There are at most 2^k subsets of $S(z)$. Using the KM algorithm, we can identify in time polynomial in $2^k, n, 1/\tau$, all Fourier coefficients of ψ_ϕ whose magnitude is at least $\tau/2^k$. Now if there exists a subset $T \subseteq S(z)$ such that $|\widehat{\psi_\phi}| \geq \tau/2^{-k}$ and $|T| > k/3$, then it will be in the list of coefficients obtained above. But note that T can be converted into a string of length n', say $\sigma(T)$, such that $\mathsf{Dec}(\sigma(T)) = \zeta(\phi)$ which is a satisfying assignment of ϕ. Thus, for each heavy (magnitude $\geq \tau/2^k$) Fourier coefficient of ψ_ϕ, we check if we get a satisfying assignment to ϕ. If not, then 0 is a valid answer (τ-approximate) to the query (ψ, τ).

The algorithm, Alg, outputs a hypothesis h. Let $h_\phi(x') = h(\phi x')$. Note that

$$\begin{aligned}\mathbb{E}_D[h(xx')c_{\phi,\xi(\phi)}(xx')] &= \mathbb{E}_{U_{n'}}[h_\phi(x')t_z(x')] \\ &= (-1 + 2^{-k+1})\widehat{h_\phi}(\emptyset) + 2^{-k+1} \sum_{\substack{T \subseteq S(z) \\ |T| > k/3}} \widehat{h_\phi}(T) \geq 1 - 2\epsilon = 1 - 2^{-k-1}\ .\end{aligned}$$

This means that

$$\sum_{\substack{T \subseteq S(z) \\ |T| > k/3}} \widehat{h_\phi}(T) \geq 1/2.$$

Thus, as for the queries, identifying and decoding all large (magnitude $\geq 0.1/2^k$) Fourier coefficients of h_ϕ reveals a satisfying assignment of ϕ of Hamming weight at most ℓ. $\square$

6.5 Upper Bounds

In this section, we consider the following question: how much computational power is sufficient for learning in these models, given that the query complexity is polynomial?

In the setting where the learning algorithm has access to i.i.d. unlabeled examples from the underlying distribution, we show that an NP-oracle suffices for learning. We show that if the query complexity for a class C is polynomial, then there exists a polynomial-time algorithm that with access to random unlabeled examples from the distribution and with access to an NP-oracle learns C. We use Szörényi's characterization of SQ learning, where he shows that any algorithm that makes consistent queries from the class C, learns C. We require an additional natural condition, $C \in \mathsf{P}$, i.e. given c as a bit string, there is a polynomial time algorithm that determines whether or not c is a valid representation of a concept in C.

Definition 6.7 (Consistent Learner [37]). *Let $\langle(\phi_i, \tau_i)\rangle_{i \geq i}$ be the queries made by an SQ learning algorithm* Alg *and let $\langle v_i \rangle_{i \geq 1}$ be the responses of the SQ oracle. Algorithm* Alg *is said to be* consistent *if for every $j < i$, $|\mathbb{E}_D[\phi_j(x)\phi_i(x)] - v_j| \leq \tau_j$.*

Szörényi [37] proved the following result.

Theorem 6.8 ([37]). *Let q be the query complexity of SQ learning concept class C with respect to distribution D, then there exists τ, such that $1/\tau$ is bounded by* $\text{poly}(q, n, 1/\epsilon)$ *and any* consistent *algorithm that makes queries of the form (c, τ), where $c \in C$, eventually makes a query of the form (c', τ), where* $\text{err}(c') \leq \epsilon/2$. *The total number of queries made by the algorithm is at most* $\text{poly}(q, n, 1/\epsilon)$.

As a corollary of this result, we can show that an NP-oracle suffices for statistical query learning, when the learning algorithm also has access to unlabeled examples from the underlying distribution. The key idea is that it is possible to find queries from C that are consistent with the previous query responses by using a large enough sample and with access to an NP-oracle. The following theorem follows easily from Theorem 6.8.

Theorem 6.9. *Let $q(C, D, \epsilon)$ be the query complexity of SQ learning concept class $C \in \mathsf{P}$ with respect to distribution D to accuracy ϵ. Then there exists an algorithm that for every target function $f \in C$, for every distribution D, with access to random examples from distribution D, oracle $\mathsf{CSQ}\text{-}\mathcal{O}(f, D)$[6] and an NP-oracle, outputs $c' \in C$, such that* $\text{err}_D(c', f) \leq \epsilon$. *The running time of the algorithm is* $\text{poly}(q(C, D, \epsilon), n, 1/\epsilon)$.

Proof. Let C be a concept class and assume that every $c \in C$ has a representation that uses at most $s(n, 1/\epsilon)$ bits for some polynomial s. Let $m = (16s(n, \epsilon^{-1})/\tau(n, \epsilon^{-1}))^2 \log(1/\delta)$. Then a random sample of size m from distribution D satisfies the following,

$$\forall c_1, c_2 \in C, |\mathbb{E}_D[c_1(x)c_2(x)] - \frac{1}{m}\sum_{k=1}^{m} c_1(x_k)c_2(x_k)| \leq \tau/4$$

[6]Since we are assuming that our algorithm has access to i.i.d. random examples from the distribution an oracle that only responds to correlational statistical queries is sufficient.

Now, consider the following algorithm. First make any query $(c_1, \tau/4)$ and receive response v_1. Note that v_1 is also a valid query response for the query (c_1, τ). Given queries $(c_1, \tau/4), \ldots, (c_{i-1}, \tau/4)$ with responses $v_1, \ldots, v_{i-1}$. Find c_i such that for every $j < i$ it holds simultaneously that,

$$\left| \frac{1}{m} \sum_{k=1}^{m} c_i(x_k) c_j(x_k) - v_j \right| \leq \tau/2 \tag{6.1}$$

Now it is easy to see that such a c_i exists because the true target concept f satisfies this. It is also easy to see that such a c_i can be identified easily using an NP-oracle, since c_i has a polynomial-size representation (thus obtaining c_i one bit at a time), and so the fact that $c_i \in C$ and the relations (6.1) can be verified easily in polynomial time.

An algorithm that makes queries $(c_1, \tau), (c_2, \tau), \ldots$ and receives responses $v_1, v_2, \ldots$ is *consistent*. Hence there will be some $t = \text{poly}(q(C, D, \epsilon)n, 1/\epsilon)$ such that $\text{err}_D(c_t, f) \leq \epsilon/2$. $\square$

Remark 6.1. *We note that the concept classes C_1, C_2, and C_3 defined respectively in Sections 6.4.2, 6.4.3, and 6.4.4 are actually not recognized by a polynomial time Turing machine. This is because given a string of the form $(\phi, \zeta(\phi))$, it is not possible to verify that $\zeta(\phi)$ is indeed the lexicographically first satisfying assignment to ϕ unless* $\mathsf{P} = \mathsf{NP}$. *We note however that even then these classes can be learned with access to an* NP*-oracle because $C_1, C_2, C_3 \in \mathsf{P}^{\mathsf{NP}}$, i.e. with access to an* NP*-oracle, the lexicographically first satisfying assignments can be constructed (one bit at a time).*

Remark 6.2. *Under stronger cryptographic assumptions, we can construct classes $C_1', C_2', C_3' \in \mathsf{P}$ that are also not efficiently learnable in the respective statistical query* *models. The functions constructed can be of the form* $(s(z), z)$*, where* $s(z)$ *is easy to find information and* s *is a one-way permutation (that is cannot be inverted efficiently). An additional implication of such constructions is average-case computational hardness: learning is hard for most functions in* $C_1'/C_2'/C_3'$.

Appendix A

Equivalence between CSQ and ZCSQ learning

We prove the following Theorem.

Theorem A.1. *For some concept class, C, and distribution, D, suppose that there is a parallel CSQ algorithm for learning C, that takes T parallel-time steps, uses p processors and makes queries to the oracle, $\mathsf{CSQ}\text{-}\mathcal{O}$, with approximation parameter 8τ. Then, there is a parallel algorithm that only queries a ZCSQ oracle, uses $T\log(1/(8\tau)) + 1$ parallel time-steps and uses polynomially many processors.*

Proof. Fix some ideal function, $f \in C$, and distribution, D. We consider a $\mathsf{CSQ}^{\leq}(\cdot,\cdot,\cdot)$ oracle, that takes a query, (ϕ, τ, θ) and responds as follows:

$$\mathsf{CSQ}^{\leq}(\phi,\tau,\theta) = \begin{cases} 1 & \text{if } \mathbb{E}_{x\sim D}[\phi(x)f(x)] \leq \theta - \tau \\ 0 & \text{if } \mathbb{E}_{x\sim D}[\phi(x)f(x)] \geq \theta + \tau \\ 1 \text{ or } 0 & \text{otherwise} \end{cases}$$

First, we use Feldman's result that shows that a query, $(\phi, 8\tau)$, to the oracle, $\mathsf{CSQ}\text{-}\mathcal{O}$, can be simulated by $\log(1/(8\tau)) + 1$ queries, to the oracle, $\mathsf{CSQ}^{\leq}(\cdot, \cdot, \cdot)$. Furthermore, each query to the $\mathsf{CSQ}^{\leq}(\cdot, \cdot, \cdot)$ oracle is of the form $(\phi, 4\tau, \theta)$, where $|\theta| \geq 4\tau$. (See also Lemma 3.1.)

Next, suppose that there exists a function, g, such that $\mathbb{E}_{x\sim D}[g(x)f(x)] = \alpha$. Assume that the function, g, is efficiently evaluable, $1/\alpha$ is polynomially bounded and that g and α are known. Then, consider the query, $((\alpha\phi - \theta g)/2, \alpha\tau)$, made to a ZCSQ oracle. We claim that the response, $\mathsf{ZCSQ}((\alpha\phi - \theta g)/2, \alpha\tau)$ is a valid response to the query, $(\phi, 4\tau, \theta)$, made to a $\mathsf{CSQ}^{\leq}(\cdot, \cdot, \cdot)$ oracle. This is because, $\mathbb{E}_{x\sim D}[(\alpha\phi(x) - \theta g(x))f(x)/2] = (\alpha/2)\mathbb{E}_{x\sim D}[\phi(x)f(x)] - (\theta/2)\mathbb{E}_{x\sim D}[g(x)f(x)] = (\alpha/2)(\mathbb{E}_{x\sim D}[\phi(x)f(x)] - \theta)$.

Feldman showed that if some concept class, C, is efficiently CSQ learnable, then there exists some polynomially bounded d, and efficiently constructible (and evaluable) functions, $g_1, g_2, \ldots, g_d$, such that for every $f \in C$, there exists g_i such that $\mathbb{E}_{x\sim D}[g_i(x)f(x)] \geq 1/d$ (see Theorem 5.2 [12]). However, we still are left with the problem of identifying such a g_i and the value $\mathbb{E}_{x\sim D}[g_i(x)f(x)]$. We simply guess some index $i \in \{1, \ldots, d\}$ and $\alpha' \in \{j\tau/d \mid j = 1, \ldots, d/\tau\}$. Let i be such that if $g' = g_i$, then $\mathbb{E}_{x\sim D}[g'(x)f(x)] = \alpha \geq 1/d$ and let α' be such that $\alpha' = j\tau/d$, and α' is the smallest number of this form that is at least α.

Then, we note that in fact, the query response $\mathsf{ZCSQ}((\alpha'\phi - \theta g')/2, \alpha'\tau))$ is a valid response to the query, $(\phi, 4\tau, \theta)$ to the oracle, $\mathsf{CSQ}^{\leq}(\cdot, \cdot, \cdot)$. To see this see the following – suppose $\mathbb{E}_{x\sim D}[\phi(x)f(x)] \geq \theta + 4\tau$. Then,

$$\begin{aligned}
\mathbb{E}_{x\sim D}[(\alpha'\phi(x) - \theta g'(x))f(x)/2] &= \frac{\alpha'}{2}\mathbb{E}_{x\sim D}[\phi(x)f(x)] - \frac{\theta\alpha}{2} \\
&\geq \frac{\alpha'}{2}\mathbb{E}_{x\sim D}[\phi(x)f(x)] - \frac{\alpha'\theta + \alpha'\tau}{2} \quad \text{Since } |\alpha' - \alpha| \leq \alpha'\tau \\
&= \frac{\alpha'}{2}(\mathbb{E}_{x\sim D}[\phi(x)f(x)] - \theta) - \alpha'\tau \\
&\geq \frac{4\tau\alpha'}{2} - \alpha'\tau = \alpha'\tau
\end{aligned}$$

A similar argument shows that when $\mathbb{E}_{x\sim D}[\phi(x)f(x)] \leq \theta - 4\tau$, $\mathbb{E}_{x\sim D}[(\alpha'\phi(x) - \theta g'(x))f(x)/2] \leq -\alpha'\tau$. Thus, a $\mathsf{CSQ}^{\leq}(\cdot,\cdot,\cdot)$ oracle may be simulated by a ZCSQ oracle, if such g' and α' are known.

Note that there are only d^2/τ possible combinations $(g_i, j\tau/d)$, for $i \in \{1,\ldots,d\}$ and $j \in \{1,\ldots,d/\tau\}$. We run d^2/τ copies of the parallel CSQ algorithm, and at least one of these, the one that is simulated using the correct (g', α'), is guaranteed to produce a hypothesis, h, such that $\Pr_{x\sim D}[f(x) \neq h(x)] \leq \epsilon$, or alternatively, $\mathbb{E}_{x\sim D}[f(x)h(x)] \geq 1 - 2\epsilon$. Note that the hypotheses produced by other copies of the parallel algorithm may be incorrect.

Suppose that the d^2/τ hypotheses are $h_1, \ldots, h_{d^2/\tau}$, and at least one, say h^*, of them is guaranteed to be such that $\mathbb{E}_{x\sim D}[f(x)h^*(x)] \geq 1 - 2\epsilon$. We say that h_i beats h_j, if $\mathbb{E}[h_i(x)f(x)] \geq \mathbb{E}[h_j(x)f(x)]$, or $\mathbb{E}[(h_i(x) - h_j(x))f(x)] \geq 0$. Note that if the ZCSQ query, $((h_j - h_i)/2, \tau)$, receives response 1, we can be sure that $\mathbb{E}[h_i(x)f(x)] \geq \mathbb{E}[h_j(x)f(x)] - 2\tau$. We compare all the d^4/τ^2 possible pairs, and define the rank of h_i to be the number of j, for which the ZCSQ oracle returned 1 on the query, $((h_j - h_i)/2, \tau)$. Let h be the hypothesis that such that has highest rank. Recall that h^* is the hypothesis that satisfies, $\mathbb{E}_{x\sim D}[h^*(x)f(x)] \geq 1 - 2\epsilon$. Now, either

$h = h^*$, or one of the two must hold: (i) the ZCSQ query, $((h^* - h)/2, \tau)$ returned 1, or (ii) there exists some h', such that the two ZCSQ queries, $((h' - h)/2, \tau)$ and $((h^* - h')/2, \tau)$ both returned 1.

In the first case, it is obvious that $\mathbb{E}_{x \sim D}[h(x)f(x)] \geq \mathbb{E}_{x \sim D}[h^*(x)f(x)] - 2\tau \geq 1 - 2(\epsilon + \tau)$. In the second case, $\mathbb{E}_{x \sim D}[h(x)f(x)] \geq \mathbb{E}_{x \sim D}[h'(x)f(x)] - 2\tau \geq \mathbb{E}_{x \sim D}[h^*(x)f(x)] - 4\tau \geq 1 - 2(\epsilon + 2\tau)$. The result follows by rescaling ϵ appropriately and outputting h.

Note that the simulation of the parallel CSQ algorithm, for each of the d^2/τ pairs, can be done in parallel (on pd^2/τ processors – note that each copy of the parallel CSQ algorithm requires p processors). This takes $T \log(1/(8\tau))$ parallel time-steps. Then, the d^4/τ^2 comparison queries can be made in parallel as well, on d^4/τ^2 processors in 1 parallel time-step. Note that in our model of parallel computation, since each processor can broadcast at every parallel time-step, the hypothesis, h, that had highest ranked can be output at this stage. Thus, the total number of parallel time-steps required is $\log(1/(8\tau)) + 1$. The total number of processors required is $\max\{pd^2/\tau, d^4/\tau^2\}$. □

Bibliography

[1] J. Aslam and S. Decatur. General bounds on statistical query learning and pac learning with noise via hypothesis boosting. *Information and Computation*, 141(2):85–118, 1998.

[2] P. L. Bartlett. Learning with a slowly changing distribution. In *Proceedings of the Conference on Learning Theory (COLT)*, 1992.

[3] N. H. Barton and B. Charlesworth. Why sex and recombination? *Science*, 281:1986–1990, 1998.

[4] R. D. Barve and P. M. Long. On the complexity of learning from drifting distributions. *Information and Computation*, 138(2):101–123, 1997.

[5] 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.

[6] Avrim Blum, Merrick Furst, Jeffery Jackson, Michael J. Kearns, Yishay Mansour, and Steven Rudich. Weakly learning dnf and characterizing statistical query learning using fourier analysis. In *Proceedings of the Symposium on the Theory of Computation (STOC)*, 1994.

[7] Nader Bshouty and Vitaly Feldman. Extended. *Journal of Machine Learning Research (JMLR)*, 2002.

[8] Charles Darwin. *On the Origin of Species by Means of Natural Selection.* John Murray, 1859.

[9] Sanjoy Dasgupta. Coarse sample complexity bounds for active learning. In *Neural Information Processing Systems (NIPS)*, 2005.

[10] D. Diochnos and G. Turán. On evolvability: The swapping algorithm, product distributions, and covariance. In *Symposium on Stochastic Algorithms, Foundations and Applications*, 2009.

[11] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness i: Basic results. *SIAM Journal on Computing*, 24(4):873–921, 1995.

[12] Vitaly Feldman. Evolution from learning algorithms. In *Proceedings of the Symposium on the Theory of Computation (STOC)*, 2008.

[13] Vitaly Feldman. Characterization of statistical query learning. In *Proceedings of the Symposium on the Theory of Computation (STOC)*, 2009.

[14] Vitaly Feldman. Robustness of evolvability. In *Proceedings of the Conference on Learning Theory (COLT)*, 2009.

[15] Vitaly Feldman. Distribution-independent evolution of linear threshold functions. In *Proceedings of the Conference on Learning Theory (COLT)*, 2011.

[16] Vitaly Feldman and Varun Kanade. Computational bounds on statistical query learning. In *Proceedings of the Conference on Learning Theory (COLT)*, 2012.

[17] Vitaly Feldman, Homin Lee, and Rocco Servedio. Shallow. In *Proceedings of the Conference on Learning Theory (COLT)*, 2011.

[18] R. A. Fisher. *The genetical theory of natural selection*. Clarendon Press, 1930.

[19] D. P. Helmbold and P. M. Long. Tracking drifting concepts by minimizing disagreements. *Machine Learning*, 14(1):27–46, 1994.

[20] Varun Kanade. Evolution with recombination. In *Proceedings of IEEE Conference on the Foundations of Computer Science (FOCS)*, 2011.

[21] Varun Kanade, Jennifer Wortman Vaughan, and Leslie G. Valiant. Evolution with drifting targets. In *Proceedings of the Conference on Learning Theory (COLT)*, 2010.

[22] Michael J. Kearns. Sq learning. *Journal of Computing*, 1998.

[23] Michael J. Kearns and Umesh Vazirani. *Computational Learning Theory*. The MIT Press, 1994.

[24] A. Kuh, T. Petsche, and R. Rivest. Incrementally learning time-varying half-planes. In *Neural Information Processing Systems (NIPS)*, 1991.

[25] E. Kushilevitz and Y. Mansour. Learning decision trees using the fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, 1993.

[26] A. Livnat, C. Papadimitriou, J. Dushoff, and M. W. Feldman. A mixability theory for the role of sex in evolution. *Proceedings of the National Academy of Science*, 105(50):19803–19808, 2008.

[27] A. Livnat, C. Papadimitriou, N. Pippenger, and M. W. Feldman. Sex, mixability, and modularity. *Proceedings of the National Academy of Science*, 107(4):1452–1457, 2010.

[28] Y. Mansour. Learning boolean functions via the fourier transform. In *Theoretical Advances in Neural Computation and Learning*, pages 391–424. Kluwer, 1994.

[29] L. Michael. Evolvability via the fourier transform, 2010. To appear in *Theoretical Computer Science*.

[30] H. J. Muller. Some genetic aspects of sex. *The American Naturalist*, 66(703):118–138, 1932.

[31] R. O'Donnell. *Computational Applications of Noise Sensitivity.* PhD thesis, MIT, 2003.

[32] H. Simon. A characterization of strong learnability in the statistical query model. In *Proceedings of the Symposium on Theoretical Aspects of Computer Science*, pages 393–404, 2007.

[33] J. Maynard Smith. *The Theory of Evolution.* Penguin Books, 1958.

[34] J. Maynard Smith. What use is sex? *Journal of Theoretical Biology*, 30(2):319 – 335, 1971.

[35] J. Maynard Smith. *The Evolution of Sex.* Cambridge University Press, Cambridge, 1978.

[36] J. Maynard Smith. The evolution of recombination. *The evolution of sex : an examination of current ideas (edited by R. E. Michod and B. R. Levin)*, pages 106–125, 1988.

[37] B. Szörényi. Characterizing statistical query learning : Simplified notions and proofs. In *Proceedings of Algorithmic Learning Theory*, pages 186–200, 2009.

[38] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 1984.

[39] Leslie G. Valiant. Evolvability. *Journal of the ACM*, 2009. Earlier version appears as *Leslie G. Valiant. Evolvability. ECCC Technical Report TR06-120.*

[40] Paul Valiant. Evolvability of real-valued functions. In *Proceedings of Innovations in Theoretical Computer Science (ITCS)*, 2012.

[41] S. Wright. Evolution in mendelian populations. *Genetics*, 16:97–159, 1931.

[42] Ke Yang. New lower bounds for statistical query learning. *Journal of Computing*, 70(4):485–509, 2005.