On Evolvability: The Swapping Algorithm, Product Distributions, and Covariance

Dimitrios I. Diochnos[1] György Turán[1,2]

[1] Dept. of Mathematics, Statistics, and Computer Science,
University of Illinois at Chicago, Chicago IL 60607, USA
[2] Research Group on Artificial Intelligence of the
Hungarian Academy of Sciences, University of Szeged

Abstract

Valiant recently introduced a learning theoretic framework for evolution, and showed that his swapping algorithm evolves monotone conjunctions efficiently over the uniform distribution. We continue the study of the swapping algorithm for monotone conjunctions. A modified presentation is given for the uniform distribution, which leads to a characterization of best approximations, a simplified analysis and improved complexity bounds. It is shown that for product distributions a similar characterization does not hold, and there may be local optima of the fitness function. However, the characterization holds if the correlation fitness function is replaced by covariance. Evolvability results are given for product distributions using the covariance fitness function, assuming either arbitrary tolerances, or a non-degeneracy condition for the distribution and a size bound on the target.

Key words: learning, evolution.

1 Introduction

A model combining evolution and learning was introduced recently by Valiant [14]. It assumes that some functionality is evolving over time. The process of evolution is modelled by updating the representation of the current hypothesis, based on its performance for training examples. Performance is measured by the correlation of the hypothesis and the target. Updating is done using a randomized local search in a neighborhood of the current representation. The objective is to evolve a hypothesis with a close to optimal performance.

As a paradigmatic example, Valiant [14] showed that monotone conjunctions of Boolean variables with the uniform probability distribution over the training examples are evolvable. Monotone conjunctions are a basic concept class for learning theory, which has been studied from several different aspects [7, 8, 10]. Valiant's algorithm, which is referred to as the *swapping* algorithm in this paper, considers *mutations* obtained by swapping a variable for another one, and adding and deleting a variable[1], and chooses randomly among beneficial mutations (or among neutral ones if there are no beneficial mutations).

Valiant also established a connection between the model and learning with *statistical queries* (Kearns [7], see also [8]), and studied different versions such as evolution with and without initialization. Valiant noted that concept learning problems have been studied before in the framework of genetic and evolutionary algorithms (e.g., Ros [11]), but his model is different (more restrictive) in its emphasis on fitness functions which depend on the training examples *only* through their performance, and *not* on the training instances themselves. His model excludes, e.g., looking at which bits are on or off in the training examples.

Feldman [2, 3] gave general results on the model and its different variants, focusing on the relationship to statistical queries. He showed that statistical query algorithms of a certain type can be translated

[1] These mutations may be viewed as swapping a variable with the constant 1.

into evolution algorithms. The translation, as noted by Feldman, does not lead to the most efficient or natural[2] evolution algorithms in general. This is the case with monotone conjunctions: even though their evolvability follows from Feldman's result, it is still of interest to find simple and efficient evolution procedures for this class. Michael [9] showed that decision lists are evolvable under the uniform distribution using the Fourier representation.

In general, exploring the performance of *simple* evolution algorithms is an interesting direction of research; hopefully, leading to new design and analysis techniques for efficient evolution algorithms. The swapping algorithm, in particular, appears to be a basic evolutionary procedure (mutating features in and out of the current hypothesis) and it exhibits interesting behavior. Thus its performance over distributions other than uniform deserves more detailed study.

In this paper we continue the study of the swapping algorithm for evolving monotone conjunctions. A modified presentation of the algorithm for the uniform distribution is given, leading to a simplified analysis and an improved complexity bound (Theorem 4.4). We give a simple characterization of best approximations by short hypotheses, which is implicit in the analysis of the algorithm.

We then consider the swapping algorithm for *product distributions*. Product distributions generalize the uniform distribution, and they are studied in learning theory in the context of extending learnability results from the uniform distribution, usually under non-degeneracy conditions (see, e.g. [4, 5, 6, 12]). We show that the characterization of best approximations does not hold for product distributions in general, and that the fitness function may have local optima.

It is shown that the picture changes if we replace the correlation fitness function with *covariance*. (Using fitness functions other than correlation has also been considered by Feldman [3] and Michael [9]; the fitness functions discussed in those papers are different from covariance.) In this case there is a characterization of best approximations similar to the uniform distribution with correlation. This leads to two positive results for the evolvability of monotone conjunctions under product distributions.

Theorem 6.3 shows that in the *unbounded-precision* model of evolution, the swapping algorithm using covariance as the fitness function, is an efficient algorithm for monotone conjunctions over *arbitrary* product distributions. Thus this result applies to a very simply defined (though clearly not realistic) evolution model, and analyzes a very simple and natural evolution algorithm (swaps using covariance) over a whole class of distributions (product distributions without any restrictions). Therefore, it may be of interest as an initial example motivating further models and algorithms, such as the introduction of short and long hypotheses in order to work with polynomial sample size estimates of performances. Theorem 6.4 shows that the swapping algorithm works if the target is short and the underlying product distribution is μ-nondegenerate.

The paper is structured as follows. Section 2 has an informal description of the swapping algorithm. As we are focusing on a single algorithm and its variants, we do not need to define the evolution model in general. The description of the swapping algorithm and some additional material given in Section 3 contain the details of the model that are necessary for the rest of the paper. Section 4 contains the analysis of the swapping algorithm for the case of uniform distribution. The performance of the swapping algorithm for product distributions is discussed in Section 5. In Section 6 we turn to the swapping algorithm using covariance as fitness function. Finally, Section 7 contains some further remarks and open problems.

2 An Informal Description of the Swapping Algorithm

Given a set of Boolean variables $x_1, \ldots, x_n$, we assume that there is an unknown *target* c, a monotone conjunction of some of these variables. The possible *hypotheses* h are of the same class. The truth values *true* and *false* are represented by 1 and -1. The *performance* of a hypothesis h is

$$\mathrm{Perf}_{\mathcal{U}_n}(h, c) = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} h(x) \cdot c(x), \tag{1}$$

[2]Of course, we do not use the term 'natural' here to suggest any actual connection with evolutionary processes in nature.

called the *correlation* of h and c. Here $\mathcal{U}_n$ denotes the uniform distribution over $\{0,1\}^n$. The evolution process starts with an initial hypothesis h_0, and produces a sequence of hypotheses using a random walk type procedure on the set of monotone conjunctions.

Each hypothesis h is assigned a *fitness value*, called the *performance* of h. The walk is performed by picking randomly a hypothesis h' from the *neighborhood* of the current hypothesis h which seems to be more fit (beneficial) compared to h, or is about as fit (neutral) as h. Details are given in Section 3.

Some care is needed in the specification of the probability distribution over beneficial and neutral hypotheses. Moreover, there is a distinction between *short* and *long* conjunctions, and the neighborhoods they induce. Valiant uses a threshold value $q = \mathcal{O}(\log(n/\varepsilon))$ for this distinction. Section 4.2 has details.

Valiant showed that if this algorithm runs for $O(n \log(n/\varepsilon))$ stages, and evaluates performances using total sample size $O((n/\varepsilon)^6)$ and different tolerances for short, resp. long conjunctions, then with probability at least $1-\varepsilon$ it finds a hypothesis h with $\mathrm{Perf}_{\mathcal{U}_n}(h,c) \geqslant 1-\varepsilon$.

3 Preliminaries

The neighborhood N of a conjunction h is the set of conjunctions that arise by *adding* a variable, *removing* a variable, or *swapping* a variable with another one, plus the conjunction itself[3]. The conjunctions that arise by adding a variable form the neighborhood N^+, the conjunctions that arise by dropping a variable form the neighborhood N^-, and the conjunctions that arise by swapping a variable form the neighborhood N^{+-}. In other words we have $N = N^- \cup N^+ \cup N^{+-} \cup \{h\}$. As an example, let our current hypothesis be $h = x_1 \wedge x_2$, and $n = 3$. Then, $N^- = \{x_1, x_2\}$, $N^+ = \{x_1 \wedge x_2 \wedge x_3\}$, and $N^{+-} = \{x_3 \wedge x_2, x_1 \wedge x_3\}$. Note that $|N| = O(n^2)$ in general.

Similarity between two conjunctions h and c in an underlying distribution $\mathcal{D}_n$ is measured by the *performance* function[4] $\mathrm{Perf}_{\mathcal{D}_n}(h,c)$ which is evaluated approximately, by drawing a random sample S and computing $\frac{1}{|S|}\sum_{x\in S} h(x)\cdot c(x)$. The goal of the evolution process is to *evolve* a hypothesis h such that:

$$\mathbf{Pr}\left[\mathrm{Perf}_{\mathcal{D}_n}(h,c) < \mathrm{Perf}_{\mathcal{D}_n}(c,c) - \varepsilon\right] < \delta. \tag{2}$$

The accuracy parameter ε and the confidence δ are treated as one in [14].

Given a target c, we split the neighborhood in 3 parts by the *increase* in performance that they offer. There are *beneficial*, *neutral*, and *deleterious* mutations. In particular, for a given neighborhood N and real constant t (*tolerance*) we are interested in the sets

$$\begin{cases} \mathrm{Bene} &= N \cap \left\{h' \mid \mathrm{Perf}_{\mathcal{D}_n}(h',c) \geqslant \mathrm{Perf}_{\mathcal{D}_n}(h,c) + t\right\} \\ \mathrm{Neut} &= N \cap \left\{h' \mid \mathrm{Perf}_{\mathcal{D}_n}(h',c) \geqslant \mathrm{Perf}_{\mathcal{D}_n}(h,c) - t\right\} \setminus \mathrm{Bene}. \end{cases} \tag{3}$$

A mutation is *deleterious* if it is neither beneficial nor neutral.

The size (or length) |h| of a conjunction h is the number of variables it contains. Given a target conjunction c and a size q, we will be interested in the best size q approximation of c.

Definition 3.1 (Best q-Approximation)**.** A hypothesis h is called a best q-approximation of c if $|h| \leqslant q$ and $\forall h' \neq h, |h'| \leqslant q : \mathrm{Perf}_{\mathcal{D}_n}(h',c) \leqslant \mathrm{Perf}_{\mathcal{D}_n}(h,c)$.

Note that the best approximation is not necessarily unique.

In this paper the following performance functions are considered; the first one is used in [14] and the second one is the *covariance* of h and c[5]:

$$\mathrm{Perf}_{\mathcal{D}_n}(h,c) = \sum_{x\in\{0,1\}^n} h(x)c(x)\mathcal{D}_n(x) = \mathbf{E}[h\cdot c] = 1 - 2\cdot\mathbf{Pr}[h \neq c] \tag{4}$$

$$\mathbf{Cov}[h,c] = \mathrm{Perf}_{\mathcal{D}_n}(h,c) - \mathbf{E}[h]\cdot\mathbf{E}[c]. \tag{5}$$

[3] As h will be clear from the context, we write N instead of N(h).

[4] See the end of this section for the specific performance functions considered in this paper. For simplicity, we keep the notation Perf for a specific performance function.

[5] A related performance function, not considered here, is the *correlation coefficient*.

4 Monotone Conjunctions under the Uniform Distribution

Given a target conjunction c and a hypothesis conjunction h, the performance of h with respect to c can be found by counting truth assignments. Let

$$h = \bigwedge_{i=1}^{m} x_i \wedge \bigwedge_{\ell=1}^{r} y_\ell \quad \text{and} \quad c = \bigwedge_{i=1}^{m} x_i \wedge \bigwedge_{k=1}^{u} w_k. \tag{6}$$

Thus the x's are *mutual* variables, the y's are *redundant* variables in h, and the w's are *undiscovered*, or *missing* variables in c. Variables in the target c are called *good*, and variables not in the target c are called *bad*.

The probability of the error region is $(2^r + 2^u - 2)2^{-m-r-u}$ and so

$$\mathrm{Perf}_{\mathcal{U}_n}(h, c) = 1 - 2^{1-m-u} - 2^{1-m-r} + 2^{2-m-r-u}. \tag{7}$$

For a fixed threshold value q, a conjunction h is *short* (resp., *long*), if $|h| \leqslant q$ (resp., $|h| > q$). The following lemma and its corollary show that if the target conjunction is long then every long hypothesis has good performance, as both the target and the hypothesis are false on most instances.

Lemma 4.1 (Performance Lower Bound)**.** *If $|h| \geqslant q$ and $|c| \geqslant q + 1$ then $Perf_{\mathcal{U}_n}(h, c) > 1 - 3 \cdot 2^{-q}$.*

Corollary 4.2. *Let $q \geqslant \lg(3/\varepsilon)$. If $|h| \geqslant q$, $|c| \geqslant q + 1$ then $Perf_{\mathcal{U}_n}(h, c) > 1 - \varepsilon$.*

4.1 Properties of the Local Search Procedure

Local search, when switching to h' from h, is guided by the quantity

$$\Delta = \mathrm{Perf}_{\mathcal{U}_n}(h', c) - \mathrm{Perf}_{\mathcal{U}_n}(h, c). \tag{8}$$

We analyze Δ using (7). The analysis is summarized in Figure 1, where the node *good* represents good variables and the node *bad* represents bad variables. Note that Δ depends only on the type of mutation performed and on the values of the parameters m, u and r; in fact, as the analysis shows, it depends on the size of the hypothesis $|h| = m + r$ and on the number u of undiscovered variables.

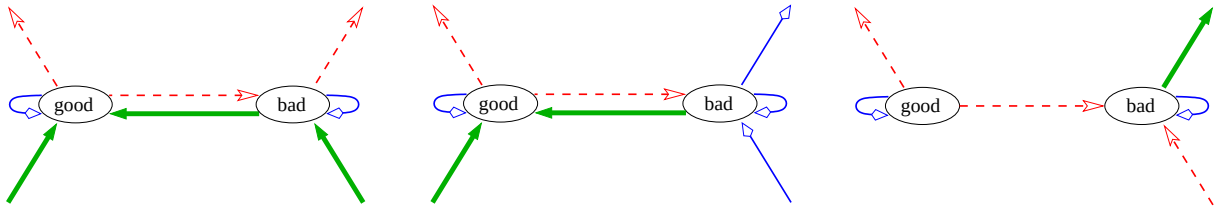
Comparing $h' \in N^+$ with h. We introduce a variable z in the hypothesis h. If z is good, $\Delta = 2^{-|h|} > 0$. If z is bad, $\Delta = 2^{-|h|}(1 - 2^{1-u})$.

Comparing $h' \in N^-$ with h. We remove a variable z from the hypothesis h. If z is good, $\Delta = -2^{1-|h|} < 0$. If z is bad, $\Delta = -2^{1-|h|}(1 - 2^{1-u})$.

Comparing $h' \in N^{+-}$ with h. Replacing a good with a bad variable gives $\Delta = -2^{1-|h|-u}$. Replacing a good with a good, or a bad with a bad variable gives $\Delta = 0$. Replacing a bad with a good variable gives $\Delta = 2^{2-|h|-u}$.

Correlation produces a perhaps unexpected phenomenon already in the case of the uniform distribution: adding a bad variable can result in Δ being positive, 0 or negative, depending on the number of undiscovered variables.

Now we turn to characterizing the best bounded size approximations of concepts, implicit in the analysis of the swapping algorithm. The existence of such characterizations seems to be related to efficient evolvability and so it may be of interest to formulate it explicitly. Such a characterization does *not* hold for product distributions in general, as noted in the next section. However, as shown in Section 6, the analogous characterization *does* hold for every product distribution if the fitness function is changed from correlation to covariance.



(a) $\mathfrak{u} \geqslant 2$ (b) $\mathfrak{u} = 1$ (c) $\mathfrak{u} = 0$

Figure 1: Arrows pointing towards the nodes indicate additions of variables and arrows pointing away from the nodes indicate removals of variables. Note that this is consistent with arrows indicating the swapping of variables. Thick solid lines indicate $\Delta > 0$, simple lines indicate $\Delta = 0$, and dashed lines indicate $\Delta < 0$. Usually Figure 1a applies. When only one good variable is missing we have the case shown in Figure 1b. Once all good variables are discovered, Figure 1c applies; hence two arrows disappear. Note that an arrow with $\Delta > 0$ may correspond to a beneficial *or* neutral mutation, depending on the value of the tolerance $\mathtt{t}$.

Theorem 4.3 (Structure of Best Approximations)**.** *The best* $\mathtt{q}$*-approximation of a target* c *is* c *if* $|c| \leqslant \mathtt{q}$*, or any hypothesis formed by* $\mathtt{q}$ *good variables if* $|c| > \mathtt{q}$*.*

Proof. The claim follows directly from the definitions if $|\mathrm{c}| \leqslant \mathtt{q}$. Let $|\mathrm{c}| > \mathtt{q}$. Let h be a hypothesis consisting of $\mathtt{q}$ good variables. Then both deleting a variable or swapping a good variable for a bad one decrease performance. Thus h cannot be improved among hypotheses of size at most $\mathtt{q}$. If h has fewer than $\mathtt{q}$ variables then it can be improved by adding a good variable. If h has $\mathtt{q}$ variables but it contains a bad variable then its performance can be improved by swapping a bad variable for a good one. Hence every hypothesis other than the ones described in the theorem can be improved among hypotheses of size at most $\mathtt{q}$. □

4.2 Evolving Monotone Conjunctions under the Uniform Distribution

The core of the algorithm for evolving monotone conjunctions outlined in Section 2 is composed by the `Mutator` function, presented in Algorithm 1. The role of `Mutator` is, given a current hypothesis h, to produce a new hypothesis h' which has better performance than h if Bene is nonempty, or else a hypothesis h' with about the same performance as h, in which case h' arises from h by a neutral mutation. Hence, during the evolution, we have $\mathtt{g}$ calls to `Mutator` throughout a sequence of $\mathtt{g}$ generations. We pass some slightly different parameters in the `Mutator` from those defined in [14], to avoid ambiguity. Hence, `Mutator` receives as input $\mathtt{q}$, the maximum allowed size for the approximation, $\mathtt{s}_{\mathrm{M},1}$, the sample size used for all the empirical estimates of the performance of each conjunction of size up to $\mathtt{q}$, $\mathtt{s}_{\mathrm{M},2}$ the sample size used for conjunctions of length greater than $\mathtt{q}$, and the current hypothesis h. We view conjunctions as objects that have two extra attributes, their *weight* and the *value* of their performance. `GetPerformance` returns the value of the performance, previously assigned by `SetPerformance`. The performance of the initial hypothesis has been determined by another similar call to the `SetPerformance` function with the appropriate sample size. Weights are assigned via `SetWeight`. Hence, `SetWeight` assigns the same weight to all members of $\{\mathrm{h}\} \cup \mathsf{N}^- \cup \mathsf{N}^+$ so that they add up to 1/2, and the same weight to all the members of N^{+-} so that they add up to 1/2. Finally, `RandomSelect` computes the sum W of weights of the conjunctions in the set that it has as argument, and returns a hypothesis h' from that set with probability $w_{\mathrm{h}'}/W$, where $w_{\mathrm{h}'}$ is the weight of h'.

Note that the neighborhoods and the tolerances are different for short and long hypotheses, where a hypothesis h is short if $|\mathrm{h}| \leqslant \mathtt{q}$, and

$$\mathtt{q} = \left\lceil \lg \frac{3}{\varepsilon} \right\rceil . \tag{9}$$

Algorithm 1: The Mutator Function under the Uniform Distribution

Input: $q \in \mathbb{N}^*$, samples $s_{M,1}$, samples $s_{M,2}$, a hypothesis h, a target c.
Output: a new hypothesis h'

if $|h| > 0$ **then** Generate N^- **else** $N^- \leftarrow \emptyset$;
if $|h| < q$ **then** Generate N^+ **else** $N^+ \leftarrow \emptyset$;
if $|h| \leqslant q$ **then** Generate N^{+-} **else** $N^{+-} \leftarrow \emptyset$;
$v_b \leftarrow$ `GetPerformance(`h`)`;
Initialize Bene, Neutral to $\emptyset$;
if $|h| \leqslant q$ **then** $t \leftarrow 2^{-2q}$ **else** $t \leftarrow 2^{1-q}$; `/* set tolerance */`
for $x \in N^+, N^-, N^{+-}$ **do**
  `SetWeight(`x, h, N^+, N^-, N^{+-}`)`;
  if $|x| \leqslant q$ **then** `SetPerformance(`x, c, $s_{M,1}$`)`; `/*` $s_{M,1}$ `examples */`
  else `SetPerformance(`x, c, $s_{M,2}$`)`; `/*` $s_{M,2}$ `examples */`
  $v_x \leftarrow$ `GetPerformance(`x`)`;
  if $v_x \geqslant v_b + t$ **then** Bene $\leftarrow$ Bene $\cup \{x\}$;
  else if $v_x \geqslant v_b - t$ **then** Neutral $\leftarrow$ Neutral $\cup \{x\}$;
`SetWeight(`h, h, N^+, N^-, N^{+-}`)`;
Neutral $\leftarrow$ Neutral $\cup \{h\}$;
if Bene $\neq \emptyset$ **then return** `RandomSelect(`Bene`)`;
else return `RandomSelect(`Neutral`)`;

Theorem 4.4. *For every target conjunction c and every initial hypothesis h_0 it holds that after $\mathcal{O}\left(q + |h_0| \ln \frac{1}{\delta}\right)$ iterations, each iteration evaluating the performance of $\mathcal{O}(nq)$ hypotheses, and each performance being evaluated using sample size $\mathcal{O}\left(\left(\frac{1}{\varepsilon}\right)^4 \left(\ln n + \ln \frac{1}{\delta} + \ln \frac{1}{\varepsilon}\right)\right)$ per iteration, equation (2) is satisfied.*

Proof. The analysis depends on the size of the target and the initial hypothesis.

Short Initial Hypothesis and Short Target. Note first that for any hypothesis h and for any target c such that $|h| = m + r \leqslant q$ and $|c| = m + u \leqslant q$, for the *non-zero* values of the quantity Δ it holds that $|\Delta| \geqslant 2^{1-m-r-u} \geqslant 2^{1-(m+r)-(m+u)} = 2^{1-(|h|+|c|)} \geqslant 2^{1-2q}$. Tolerance for short hypotheses is $t = \frac{1}{2}2^{1-2q} = 2^{-2q}$. Hence as long as the estimate of the performance is within t of its exact value, beneficial mutations are identified as beneficial. Therefore it is sufficient to analyze the signs of Δ along the arrows in Figure 1. Note that deleting a good variable is always deleterious, and so u is non-increasing.

If there are at least two undiscovered variables (i.e., $u \geqslant 2$, corresponding to Figure 1a), then beneficial mutations can only *add* or *swap* variables. Each swap increases the number of good variables, and so after $|c| - 1$ many swaps there is at most one undiscovered variable. Hence, as long as $u \geqslant 2$, there can be at most $q - |h_0|$ additions and at most $|c| - 1$ swaps.

If there is one undiscovered variable (i.e., $u = 1$, corresponding to Figure 1b), then, in 1 step, the first beneficial mutation brings this variable into the hypothesis, and all variables become discovered.

If all variables are discovered (i.e., $u = 0$, corresponding to Figure 1c) then beneficial mutations are those which delete bad variables from h_0. After we get to the target, there are no beneficial mutations, and the only neutral mutation is the target itself, hence there is no change. Thus the number of steps until getting to the target is at most $q - |c|$.

Summing up the above, the total number of steps is at most $2q - |h_0| \leqslant 2q$.

Short Initial Hypothesis and Long Target. As long as $|h| < q$, we have $u \geqslant 2$, corresponding to Figure 1a. Therefore adding any variable is beneficial. Note that replacing a bad variable by a good one may or may not be so, depending on the size of c. The same analysis as above implies that after at most $2q$ beneficial mutations we reach a hypothesis of size q.

If $|c| \geqslant q+2$ then $u \geqslant 2$ continues to hold, and so all beneficial or neutral mutations will keep hypothesis size at q. However, by Corollary 4.2, all those hypotheses have performance at least $1-\varepsilon$.

If $|c| = q+1$ then after reaching level q there is one undiscovered variable, corresponding to Figure 1b. Swaps of bad variables for good ones are beneficial. Combining these mutations with the ones needed to reach level q, we can bound the *total* number of steps until reaching a hypothesis of q good variables by $2q$ (using the same argument as above). After that, there are only neutral mutations swapping a good variable with another good one, and again all those hypotheses have performance at least $1-\varepsilon$.

As a summary, if we start from a short hypothesis and all the empirical tests perform as expected, then, we are *always* at a good hypothesis after $2q$ iterations. This will not be the case when we start from a long hypothesis.

Long Initial Hypothesis. For long hypotheses the neighborhood consists of hypotheses obtained by deleting a variable, and the hypothesis itself. We set the tolerance in such a way that every hypothesis in the neighborhood is neutral. This guarantees that with high probability in $\mathcal{O}\left(|h_0| \ln \frac{1}{\delta}\right)$ iterations we arrive at a hypothesis of size at most q, and from then on we can apply the analysis of the previous two cases. The model assumes that staying at a hypothesis is always a neutral mutation, hence it is possible to end up in a hypothesis of size bigger than q.

Computing sample sizes is done by standard Chernoff bound arguments and the details are omitted. □

5 Monotone Conjunctions under Product Distributions Using Correlation

A *product distribution* over $\{0,1\}^n$ is specified by the probabilities $p = (p_1, \ldots, p_n)$, where p_i is the probability of setting the variable x_i to 1. The probability of a truth assignment $(a_1, \ldots, a_n) \in \{0,1\}^n$ is $\prod_{i=1}^{n} p_i^{a_i} \cdot (1-p_i)^{1-a_i}$. For the uniform distribution U_n the probabilities are $p_1 = \ldots = p_n = 1/2$. We write $\mathcal{P}_n$ to denote a fixed product distribution, omitting p for simplicity.

Let us consider a target c and a hypothesis h as in (6). Let $\textsc{index}(z)$ be a function that returns the set of indices of the participating variables in a hypothesis z. We define the sets $\mathfrak{M} = \textsc{index}(h) \cap \textsc{index}(c)$, $\mathfrak{R} = \textsc{index}(h) \setminus \mathfrak{M}$, and $\mathfrak{U} = \textsc{index}(c) \setminus \mathfrak{M}$. We can now define

$$M = \prod_{i \in \mathfrak{M}} p_i, \qquad R = \prod_{\ell \in \mathfrak{R}} p_\ell, \qquad \text{and} \qquad U = \prod_{k \in \mathfrak{U}} p_k.$$

Finally, set $|\mathfrak{M}| = m$, $|\mathfrak{R}| = r$, and $|\mathfrak{U}| = u$. Then (7) generalizes to

$$\mathrm{Perf}_{\mathcal{P}_n}(h, c) = 1 - 2M(R + U - 2RU). \tag{10}$$

We impose some conditions on the p_i's in the product distribution.

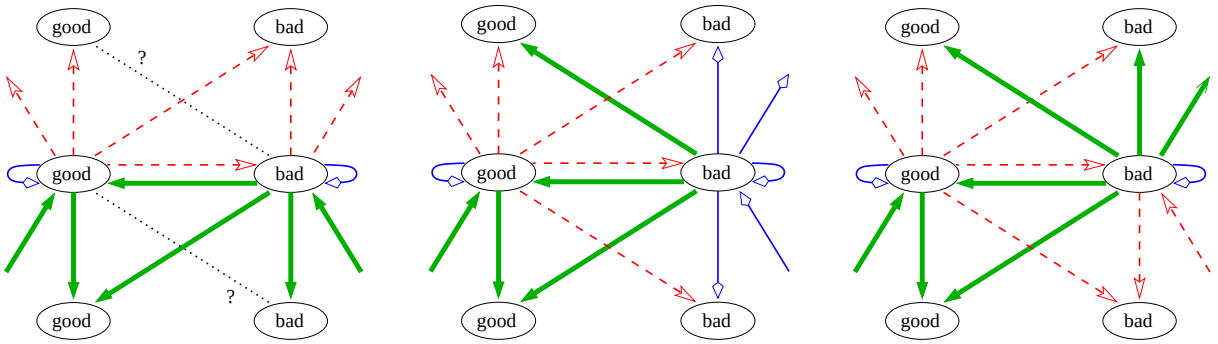
Definition 5.1 (Nondegenerate Product Distribution)**.** A product distribution $\mathcal{P}_n$ given by $p = (p_1, \ldots, p_n)$ is μ-nondegenerate if

- $\min\{p_z, 1-p_z\} \geqslant \mu$ for every variable z
- the difference of any two members of the multiset $\{p_1, 1-p_1, \ldots, p_n, 1-p_n\}$ is zero, or has absolute value at least μ.

The following Lemma and its Corollary are analogous to Lem. 4.1 and Cor. 4.2.

Lemma 5.2 (Performance Lower Bound)**.** *Let a hypothesis h such that $|h| \geqslant q-1$ and a target c such that $|c| \geqslant q+1$. Then, $Perf_{\mathcal{P}_n}(h, c) > 1 - 6.2 \cdot e^{-\mu q}$.*

Corollary 5.3. *Let $q \geqslant \frac{1}{\mu} \ln\left(\frac{6.2}{\varepsilon}\right)$, $|h| \geqslant q-1$, $|c| \geqslant q+1 \Rightarrow Perf_{\mathcal{U}_n}(h, c) > 1-\varepsilon$.*



(a) $\mathsf{U} < 1/2$ (b) $\mathsf{U} = 1/2$ (c) $\mathsf{U} > 1/2$

Figure 2: The style and the directions of arrows have the same interpretation as in Figure 1. The middle layer represents variables that have the same probability of being satisfied under the distribution; i.e. $\mathsf{p}_{\text{good}} = \mathsf{p}_{\text{bad}}$. A node x that is one level above another one y indicates higher probability of satisfying the variable x; i.e. $\mathsf{p}_\mathsf{x} > \mathsf{p}_\mathsf{y}$. Here we distinguish the three basic cases for U; for two arrows in the first case we have a ? to indicate that Δ can not be determined by simply distinguishing cases for U.

5.1 Properties of the Local Search Procedure

We want to generalize the results of Section 4.1 by looking at the quantity

$$\Delta = \text{Perf}_{\mathcal{P}_n}\left(h', c\right) - \text{Perf}_{\mathcal{P}_n}\left(h, c\right) \tag{11}$$

which corresponds to (8). We use (10) for the different types of mutations.

The signs of Δ depend on the ordering of the probabilities p_i. A variable x_i is *smaller* (resp., *larger*) than a variable x_j if $\mathsf{p}_i < \mathsf{p}_j$ (resp., $\mathsf{x}_i > \mathsf{x}_j$). If $\mathsf{p}_i = \mathsf{p}_j$ then x_i and x_j are *equivalent*. Analyzing Δ, we draw the different cases in Figure 2. However, when $\mathsf{U} < 1/2$, two arrows can not be determined. These cases refer to mutations where we replace a bad variable with a bigger good one, or a good variable with a smaller bad one. Both mutations depend on the distribution; the latter has $\Delta = -2\mathsf{MR}(\mathsf{p}_{\mathbf{in}}/\mathsf{p}_{\mathbf{out}} - 1 + 2\mathsf{U}(1 - \mathsf{p}_{\mathbf{in}}))$, where **out** is a good variable and **in** is the bad smaller variable. One application of this equation is that the Structure Theorem 4.3 does not hold under product distributions. The other application is the construction of local optima; example follows.

Example 1. *Let $\mathcal{P}_n$ be a distribution such that $\mathsf{p}_1 = \mathsf{p}_2 = 1/3$, and the rest of the $n - 2$ variables are satisfied with probability $1/10$. Set the target $c = \mathsf{x}_1 \wedge \mathsf{x}_2$. A hypothesis h formed by q bad variables has performance $Perf_{\mathcal{P}_n}(h, c) = 1 - 2\mathbf{Pr}[\text{error region}] < 1 - 2/9 = 7/9$. Note that, for the nonzero values of Δ, it holds $|\Delta| \geqslant 2\mu^{q+2}$. Hence, by setting the tolerance $t = \mu^{q+2}$, and the accuracy on the empirical tests on conjunctions of size at most q, equal to $\epsilon_M = t = \mu^{q+2}$, all the arrows in the diagrams can be determined precisely.*

Starting from $h_0 = \emptyset$, there are sequences of beneficial mutations in which the algorithm inserts a bad variable in each step, e.g. $h_0 = \emptyset \rightsquigarrow h_1 = \mathsf{x}_3 \rightsquigarrow \ldots \rightsquigarrow h_q = \bigwedge_{\ell=3}^{q+2} \mathsf{x}_\ell$. This is a local optimum, since swapping a bad variable with a good one yields $\Delta < 0$. Note that $\mu = 1/10, q = \lceil 10 \ln(62) \rceil = 42$, and for $\varepsilon = 1/10$ the algorithm is stuck in a hypothesis with $Perf_{\mathcal{P}_n}(h_q, c) < 1 - \varepsilon$.

Under the setup of the example above, the algorithm will insert q bad variables in the first q steps, with probability $\Gamma = \prod_{r=0}^{q-1}\left(1 - \frac{2}{n-r}\right) = \frac{(n-q)(n-q-1)}{n(n-1)}$. Requiring $n \geqslant \left\lceil \frac{2q}{\delta} \right\rceil$ we have $\Gamma \geqslant 1 - \delta$. Hence, starting from the empty hypothesis, the algorithm will fail for *any* $\varepsilon < 2/10$, with probability 0.9, if we set $n \geqslant 840$.

5.2 Special Cases

Although for arbitrary targets and arbitrary product distributions we can not guarantee that the algorithm will produce a hypothesis h such that (2) is satisfied, we can however, pinpoint some cases where the algorithm will succeed with the correct setup. These cases are targets of size at most 1 or greater than q, and *heavy* targets; i.e. targets that are satisfied with probability at least 1/2.

6 Covariance as a Fitness Function

The discussion in the previous section shows that there are problems with extending the analysis of the swapping algorithm from the uniform distribution to product distributions. In this section we explore the possibilities of handling product distributions with a different fitness function, covariance, given by (5).

Using the same notation as in (6), and with $\mathfrak{M}, \mathfrak{R}$, and $\mathfrak{U}$ representing the sets of indices as in the previous section, the first term is given by (10). Furthermore,

$$\begin{cases} \mathbf{E}[\mathrm{h}] &= -1 + 2 \cdot \prod_{\mathrm{i} \in \mathfrak{M}} \mathsf{p}_\mathrm{i} \cdot \prod_{\ell \in \mathfrak{R}} \mathsf{p}_\ell &= -1 + 2\mathsf{MR} \\ \mathbf{E}[\mathrm{c}] &= -1 + 2 \cdot \prod_{\mathrm{i} \in \mathfrak{M}} \mathsf{p}_\mathrm{i} \cdot \prod_{\mathrm{k} \in \mathfrak{U}} \mathsf{p}_\mathrm{k} &= -1 + 2\mathsf{MU} \end{cases} \tag{12}$$

Thus from (10) and (12) we get

$$\mathbf{Cov}[\mathrm{h}, \mathrm{c}] = 4\mathsf{MRU}(1 - \mathsf{M}). \tag{13}$$

We use (13) to examine the difference $\Delta = \mathbf{Cov}\left[\mathrm{h}', \mathrm{c}\right] - \mathbf{Cov}[\mathrm{h}, \mathrm{c}]$.

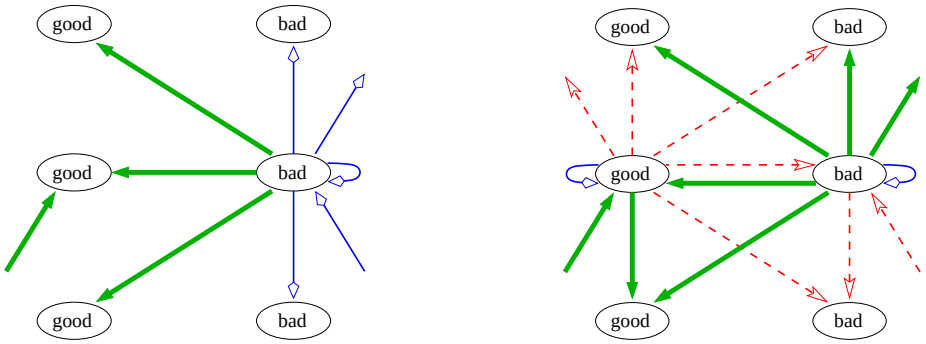
Comparing $\mathbf{h}' \in \mathsf{N}^+$ with h. We introduce a variable z in the hypothesis h. If z is good, then $\Delta = 4\mathsf{M}^2\mathsf{RU}(1 - \mathsf{p}_\mathrm{z}) > 0$. If z is bad, then $\Delta = (\mathsf{p}_\mathrm{z} - 1)\,\mathbf{Cov}[\mathrm{h}, \mathrm{c}] \leqslant 0$. We have equality if $\mathsf{m} = 0$; i.e. $\mathsf{M} = 1$.

Comparing $\mathbf{h}' \in \mathsf{N}^-$ with h. We remove a variable z from the hypothesis h. If z is good, then $\Delta = -4\mathsf{M}^2\mathsf{RU}(1/\mathsf{p}_\mathrm{z} - 1) < 0$. If z is bad, then $\Delta = (1/\mathsf{p}_\mathrm{z} - 1)\mathbf{Cov}[\mathrm{h}, \mathrm{c}] \geqslant 0$. We have equality if $\mathsf{m} = 0$; i.e. $\mathsf{M} = 1$.

Comparing $\mathbf{h}' \in \mathsf{N}^{+-}$ with h. We swap a variable **out** with a variable **in**.
If **out** is good and **in** is good, then $\Delta = 4\mathsf{M}^2\mathsf{RU}(1 - \mathsf{p}_\mathbf{in}/\mathsf{p}_\mathbf{out})$.
If $\mathsf{p}_\mathbf{out} \leqslant \mathsf{p}_\mathbf{in}$, then $\Delta \leqslant 0$, with $\Delta = 0$ if $\mathsf{p}_\mathbf{out} = \mathsf{p}_\mathbf{in}$. If $\mathsf{p}_\mathbf{out} > \mathsf{p}_\mathbf{in} \Rightarrow \Delta > 0$.
If **out** is good and **in** is bad, then $\Delta = 4\mathsf{MRU} \cdot ((\mathsf{p}_\mathbf{in} - 1) + \mathsf{M} \cdot (1 - \mathsf{p}_\mathbf{in}/\mathsf{p}_\mathbf{out}))$. We now examine the quantity $\kappa = (\mathsf{p}_\mathbf{in} - 1) + \mathsf{M} \cdot (1 - \mathsf{p}_\mathbf{in}/\mathsf{p}_\mathbf{out})$:
$\mathsf{p}_\mathbf{out} \leqslant \mathsf{p}_\mathbf{in}$: Then $(1 - \mathsf{p}_\mathbf{in}/\mathsf{p}_\mathbf{out}) \leqslant 0$, and $(\mathsf{p}_\mathbf{in} - 1) < 0$. Therefore $\Delta < 0$.
$\mathsf{p}_\mathbf{out} > \mathsf{p}_\mathbf{in}$: Note $\mathsf{M} \leqslant \mathsf{p}_\mathbf{out}$. Hence, $\kappa < \mathsf{p}_\mathbf{in} - 1 + 1 - \mathsf{p}_\mathbf{in}/\mathsf{p}_\mathbf{out} = \mathsf{p}_\mathbf{in}(1 - 1/\mathsf{p}_\mathbf{out}) < 0$.
If **out** is bad and **in** is bad, then $\Delta = (\mathsf{p}_\mathbf{in}/\mathsf{p}_\mathbf{out} - 1) \cdot \mathbf{Cov}[\mathrm{h}, \mathrm{c}]$ and $\mathbf{Cov}[\mathrm{h}, \mathrm{c}] \geqslant 0$:
$\mathsf{p}_\mathbf{out} \leqslant \mathsf{p}_\mathbf{in}$: In this case, $\Delta \geqslant 0$, and $\Delta = 0$ when $\mathsf{m} = 0$, or $\mathsf{p}_\mathbf{out} = \mathsf{p}_\mathbf{in}$.
$\mathsf{p}_\mathbf{out} > \mathsf{p}_\mathbf{in}$: In this case $\Delta \leqslant 0$, and $\Delta = 0$ when $\mathsf{m} = 0$.
If **out** is bad and **in** is good, then $\Delta = 4\mathsf{MRU}(1/\mathsf{p}_\mathbf{out} - 1 + \mathsf{M}(1 - \mathsf{p}_\mathbf{in}/\mathsf{p}_\mathbf{out}))$. We examine the quantity $\kappa = 1/\mathsf{p}_\mathbf{out} - 1 + \mathsf{M}(1 - \mathsf{p}_\mathbf{in}/\mathsf{p}_\mathbf{out})$:
$\mathsf{p}_\mathbf{out} < \mathsf{p}_\mathbf{in}$: Note $\mathsf{M} \leqslant 1$. Hence, $\kappa > 1/\mathsf{p}_\mathbf{out} - 1 + 1 - \mathsf{p}_\mathbf{in}/\mathsf{p}_\mathbf{out} = (1 - \mathsf{p}_\mathbf{in})/\mathsf{p}_\mathbf{out} > 0$.
$\mathsf{p}_\mathbf{out} \geqslant \mathsf{p}_\mathbf{in}$: In this case $\mathsf{p}_\mathbf{in}/\mathsf{p}_\mathbf{out} \leqslant 1 \Rightarrow \kappa > 0 \Rightarrow \Delta > 0$.

The effects of the different mutations are summarized in Figure 3. Compared to Figure 2, it is remarkably simple and uniform, and can be summarized as follows. If there are some mutual variables (i.e. good) in the hypothesis, then

- $\Delta > 0$ if a good variable is added, a bad variable is deleted, a bad variable is replaced by a good one, a good variable is replaced by a smaller good one, and a bad variable is replaced by a larger bad one,



(a) $M = 1$ (b) $M < 1$

Figure 3: The style and the directions of arrows have the same interpretation as in the previous figures. Similarly, the hierarchy of nodes on levels has the same interpretation. Some arrows are missing in the left picture since there are no good variables in the hypothesis; i.e. $M = 1$.

- $\Delta < 0$ if a good variable is deleted, a bad variable is added, a good variable is replaced by a bad one, a good variable is replaced by a larger good one, and a bad variable is replaced by a smaller bad one,
- $\Delta = 0$ if a good variable is replaced by an equivalent good one, and a bad variable is replaced by an equivalent bad one.

If there are no mutual variables in the hypothesis, then

- $\Delta > 0$ if a good variable is added, or a good variable replaces a bad one.
- $\Delta = 0$ if a bad variable is added, deleted, or replaced by a bad one.

Note that the beneficiality or neutrality of a mutation is *not* determined by these observations; it also depends on the tolerances. Nevertheless, these properties are sufficient for an analogue of Theorem 4.3 on the structure of best approximations to hold for product distributions and the covariance fitness function.

Theorem 6.1 (Structure of Best Approximations)**.** *The best q-approximation of a target c, such that $|c| \geqslant 1$, is c itself if $|c| \leqslant q$, or any hypothesis formed by q good variables, such that the product $\prod_{i=1}^{q} p_i$ is minimized if $|c| > q$.*

As mentioned earlier, the existence of characterizations of best approximations is related to evolvability. This relationship is now illustrated for product distribution and the covariance fitness function. First we introduce an idealized version of evolution algorithms, where beneficiality depends on the precise value of the performance function.

Definition 6.2 (Unbounded-Precision Evolution Algorithm)**.** An evolution algorithm is unbounded-precision if, instead of (3) it uses

$$\begin{cases} \text{Bene} &= N \cap \left\{ h' \mid \text{Perf}_{\mathcal{D}_n}\left(h', c\right) > \text{Perf}_{\mathcal{D}_n}\left(h, c\right) \right\} \\ \text{Neut} &= N \cap \left\{ h' \mid \text{Perf}_{\mathcal{D}_n}\left(h', c\right) = \text{Perf}_{\mathcal{D}_n}\left(h, c\right) \right\} \end{cases}, \tag{14}$$

or, equivalently, arbitrary tolerance to determine which hypotheses are beneficial, neutral or deleterious. All other parts of the definition are unchanged.

Consider the following unbounded-precision evolution algorithm: starting from an arbitrary initial hypothesis, apply beneficial mutations as long as possible. Then beneficial mutations can add a good variable, delete a bad variable, replace a bad variable by a good one, replace a good variable by a smaller good one or replace a bad variable by a larger bad one. The amortized analysis argument of Theorem 6.4 in the next section shows that the number of steps is $O(n^2)$. Hence the following result holds.

Theorem 6.3. *The swapping algorithm using the covariance fitness function is an efficient evolution algorithm for monotone conjunctions over product distributions.*

6.1 Evolving Short Monotone Conjunctions under μ-Nondegenerate Product Distributions

The problem with applying the unbounded-precision algorithm to the bounded-precision model is that the presence of the U factor in Δ may make the performance differences superpolynomially small. If we assume that the product distribution is non-degenerate and the target is short then this cannot happen, and an analysis similar to Theorem 4.4 shows that we indeed get an efficient evolution algorithm. In Section 7 we give some remarks on possibilities for handling long targets. We set

$$\mathfrak{q} = \mathcal{O}\left(\frac{1}{\mu}\ln\frac{1}{\varepsilon}\right).$$

Theorem 6.4. *Let $\mathcal{P}_\mathfrak{n}$ be a μ-nondegenerate product distribution. The swapping algorithm, using the covariance fitness function, evolves* non-empty *short ($1 \leqslant |c| \leqslant \mathfrak{q}$) monotone conjunctions starting from an initial hypothesis h_0 in $\mathcal{O}\left(\mathfrak{n}\mathfrak{q} + |h_0|\ln\frac{1}{\delta}\right)$ iterations, each iteration examining the performance of $\mathcal{O}(\mathfrak{n}\mathfrak{q})$ hypotheses, and each performance being evaluated using sample size*

$$\mathcal{O}\left(\left(\frac{1}{\mu}\right)^4\left(\frac{1}{\varepsilon}\right)^{(4/\mu)\ln(1/\mu)}\left(\ln\mathfrak{n} + \ln\frac{1}{\delta} + \ln\frac{1}{\mu} + \ln\frac{1}{\varepsilon}\right)\right).$$

Proof. The analysis of the proof is similar to that of Theorem 4.4.

Short Initial Hypothesis. Again, we are interested in the *non-zero* values of Δ so that, given representative samples, we can characterize precisely all the mutations. It holds that $|\Delta| \geqslant 4\mu^{2\mathfrak{q}+2}$. Therefore, we set the tolerance $\mathfrak{t} = 2\mu^{2\mathfrak{q}+2}$, and require accuracy for the empirical estimates $\epsilon_\mathsf{M} = \mathfrak{t} = 2\mu^{2\mathfrak{q}+2}$.

Without loss of generality, we are in the state of Figure 3b, otherwise, in 1 step a good variable appears in the hypothesis, and we move from Figure 3a to Figure 3b. Throughout the evolution, implicitly, we have two arrays; one with good variables, and one with bad variables. The array of bad variables shrinks in size, while the array of good variables expands in size. Due to mutations that swap good with good variables, and bad with bad variables, each entry of those arrays can take at most $\mathfrak{n}$ different values, since the entries of the bad array increase in value, while the entries of the good array decrease in value; i.e. we have $\mathcal{O}(\mathfrak{n}\mathfrak{q})$ such mutations overall. Mutations that change the parameters $\mathfrak{m}, \mathfrak{r}$ increase the number of good variables or decrease the number of bad variables, hence we can have at most $\mathcal{O}(\mathfrak{q})$ such mutations.

Long Initial Hypothesis. The arguments are similar to those in Theorem 4.4, and in $\mathcal{O}\left(|h_0|\ln\frac{1}{\delta}\right)$ stages the algorithm forms a short hypothesis of size $\mathfrak{q}$. Then, we apply the analysis above. □

7 Further Remarks

We are currently working on completing the results of Section 6, by handling the case of long targets. The problem with long targets is that the term U appearing in Δ can be small. This may lead to the increase in fitness being smaller than the tolerance. The following observation may provide a starting point. It may happen that there are no beneficial mutations, but all mutations are neutral. In this case the evolution process is similar to a random walk on the set of short hypotheses. However, most short hypotheses have maximal size. Therefore, after sufficiently many steps in the random walk, the hypothesis will have maximal size with high probability. Such hypotheses have good performance, and so the evolution process leads to a good hypothesis with high probability.

The evolvability of monotone conjunctions for more general classes of distributions, using the swapping algorithm with the covariance fitness function, or using some other simple evolutionary mechanism, is an

important open problem [14]. There is a similar swapping-type learning algorithm for decision lists, where a single step exchanges two tests in the list [13, 1]. Can such an algorithm be used in the evolution model? A positive answer could give an alternative to Michael's Fourier-based approach [9].

In summary, it appears that from the perspective of learning theory, one of the remarkable features of Valiant's new model of evolvability is that it puts basic, well-understood learning problems in a new light and poses new questions about their learnability. One of these new questions is the performance of basic, simple evolution mechanisms, like the swapping algorithm for monotone conjunctions. The results of this paper suggest that the analysis of these mechanisms may be an interesting challenge.

References

[1] Jorge Castro and José L. Balcázar. Simple PAC Learning of Simple Decision Lists. In *ALT '95*, pages 239–248, London, UK, 1995. Springer-Verlag.

[2] Vitaly Feldman. Evolvability from learning algorithms. In *STOC '08*, pages 619–628, New York, NY, USA, 2008. ACM.

[3] Vitaly Feldman. Robustness of Evolvability. In *COLT 2009*, 2009.

[4] Merrick L. Furst, Jeffrey C. Jackson, and Sean W. Smith. Improved learning of AC0 functions. In *COLT '91*, pages 317–325, 1991.

[5] Thomas Hancock and Yishay Mansour. Learning monotone ku DNF formulas on product distributions. In *COLT '91*, pages 179–183, 1991.

[6] Adam Tauman Kalai and Shang-Hua Teng. Decision trees are PAC-learnable from most product distributions: a smoothed analysis. *CoRR*, abs/0812.0933, 2008.

[7] Michael Kearns. Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6):983–1006, 1998.

[8] Michael J. Kearns and Umesh V. Vazirani. *An introduction to computational learning theory*. MIT Press, Cambridge, MA, USA, 1994.

[9] Loizos Michael. Evolvability via the Fourier Transform, 2009.

[10] Rüdiger Reischuk and Thomas Zeugmann. A Complete and Tight Average-Case Analysis of Learning Monomials. In *STACS'99*, pages 414–423, 1999. Springer.

[11] Johannes P. Ros. Learning Boolean functions with genetic algorithms: A PAC analysis. In *FGA*, pages 257–275, San Mateo, CA, 1993. Morgan Kaufmann.

[12] Rocco A. Servedio. On learning monotone DNF under product distributions. *Inf. Comput.*, 193(1):57–74, 2004.

[13] Hans-Ulrich Simon. Learning decision lists and trees with equivalence-queries. In *EuroCOLT '95*, pages 322–336, London, UK, 1995. Springer-Verlag.

[14] Leslie G. Valiant. Evolvability. In Ludek Kucera and Antonín Kucera, editors, *MFCS*, vol. 4708 of *LNCS*, pages 22–43, 2007. Springer.