A New Composition Theorem for Learning Algorithms

Nader H. Bshouty*

Abstract

We present a new approach to the composition of learning algorithms (in various models) for classes of constant VC-dimension into learning algorithms for more complicated classes. We prove that if a class $\mathcal{C}$ is learnable in time t from a hypothesis class $\mathcal{H}$ of constant VC-dimension then the class $\mathcal{C}^*$ of all functions F of the form $F = f(g_1, \ldots, g_m)$, where f is any function and $g_1, \ldots, g_m \in \mathcal{C}$, is learnable in time polynomial in t and m. We also use a simple argument to prove that the composition theorem cannot be extended to classes with a nonconstant VC-dimension.

A composition theorem for the exact learning model (with equivalence queries only) is proven in [BBK97] *only* for classes $\mathcal{C}$ of constant VC-dimension that have *constant space* learning algorithms. Constant space algorithms are hard to find and have large complexity. Our algorithm is simple and has a complexity lower than the algorithm in [BBK97].

We then show how to change a PAC-learning algorithm of $\mathcal{C}$ from $\mathcal{H}$ to an SQ-learning algorithm and to a PAC-learning algorithm for $\mathcal{C}^*$ with malicious noise that achieves the optimal error rate $\eta/(1-\eta) + \beta$ for any β. This, in particular, shows that if a class of constant VC-dimension is PAC-learnable from a class of constant VC-dimension then it is SQ-learnable and PAC-learnable with malicious noise. We apply this result for SQ-learning and PAC-learning with malicious noise a general class of geometric objects. This class includes the set of all geometric objects in the constant dimensional space that are bounded by m algebraic surfaces of constant degree (for example, hyperplanes, spheres, etc.). This result generalizes all the results known from the literature about learning geometric objects in the SQ-learning and PAC-learning models with malicious noise.

1 Introduction

The Composition Theorem for learning algorithms [BBK97] shows that if a class of concepts $\mathcal{C}$ is exactly learnable (from equivalence queries only) in time t using *constant space* from a class of concepts $\mathcal{H}$, (and therefore $\mathcal{H}$ must have a constant VC-dimension) then the class $\mathcal{C}^*$ of all functions of the form $f(g_1, \ldots, g_m)$ is exactly learnable in time polynomial in t and m where f is any function and $g_1, \ldots, g_m \in \mathcal{C}$. This result is applied in [BBK97] to prove the exact learnability of geometric classes from equivalence queries only. It is shown that algebraic surfaces of constant degree (for example, hyperplanes, spheres, or any $p(x_1, \ldots, x_d) = 0$ for a constant degree polynomial p) over a constant dimensional space d are constant space learnable. Therefore, by applying the composition theorem, any geometric object bounded by constant degree surfaces in the constant dimensional space is exactly learnable. Space bounded exact learning algorithms are hard to find and have large complexity.

In this paper we prove the Composition Theorem for a broader set of classes $\mathcal{C}$ and for other learning models. We first show that if a class of concepts $\mathcal{C}$ is exactly learnable in time t by a hypothesis class $\mathcal{H}$ of constant VC-dimension then the class $\mathcal{C}^*$ is learnable in time polynomial in t and m. We then show that a much weaker condition can be placed on $\mathcal{C}$ and $\mathcal{H}$ to ensure learnability of $\mathcal{C}^*$. For example, if $\mathcal{C}$ is PAC-learnable (this is weaker than exact learning) from a constant VC-dimension hypothesis class $\mathcal{H}$ then the randomized Halving algorithm for $\mathcal{C}$ (here the hypothesis class is not of constant VC-dimension) can be changed to an algorithm for $\mathcal{C}^*$. We also show that the composition theorem cannot be extended to classes with nonconstant VC-dimension.

We then investigate SQ-learning model and the PAC-learning with malicious noise model. We show that if $\mathcal{C}$ is PAC-learnable in time t by a hypothesis class $\mathcal{H}$ of constant VC-dimension then the class $\mathcal{C}^*$ is SQ-learnable and PAC-learnable with malicious noise $\eta = \epsilon/(1+\epsilon) + \Delta$ in time polynomial in t and $1/\Delta$. In particular, if a class $\mathcal{C}$ of constant VC-dimension is learnable from a constant VC-dimension hypothesis class then it is SQ-learnable and PAC-learnable with malicious noise. Here the malicious noise model is the one defined in [CFSS97] which is weaker than the one defined in [KL93]. In this model the learner makes one request for m examples. The teacher (adversary) chooses m examples $\{(x_i, F(x_i))\}$ according to the distribution D and then marks each example $(x_i, F(x_i))$ with probability η. The teacher then replaces each marked example by an arbitrary pair (a, b).

One type of concept classes which has attracted considerable attention (in the exact model as well as in other models) is that of geometric concept classes. In the case of exact learning we consider a discretized domain of R_n^d (i.e., the set of points of the form $\{0, 1, \ldots, n\}^d$, for some n) and the concepts considered are of geometric nature such as axis-parallel boxes (e.g., [MT89, CM92, Aue93]) and geometric objects bounded by constant degree algebraic surphaces (e.g. [BGMST96]). We apply our composition theorem to the class of constant degree algebraic surfaces to get the

*Department of Computer Science, University of Calgary, Calgary, Alberta, Canada. E-mail: bshouty@cpsc.ucalgary.ca. http://www.cpsc.ucalgary.ca/~bshouty/home.html.
and Department of Computer Science, Technion, Haifa 32000, Israel. E-mail: bshouty@cs.technion.ac.il.

STOC '98 Dallas Texas USA

first polynomial time SQ-learning algorithm and PAC-learning algorithm with malicious noise for the class of all geometric objects in the constant dimensional space bounded by constant degree algebraic surfaces.

Organization: In section 2 we will give some preliminary background information. In section 3 we present our composition theorem for the exact learning model and in section 4 we present the theorem for the SQ-learning model and for the PAC-learning model with malicious noise.

2 Preliminaries

2.1 Dual Class and VC Dimension

Let $X = \{x_1, \ldots, x_m\}$ be a set and $\mathcal{H} \subseteq \{0,1\}^X$ be a class of boolean functions over X. Define the *dual* class $\mathcal{H}^\perp \subseteq \{0,1\}^{\mathcal{H}}$ to be the class of boolean functions $x_i^\perp$ where for every $h \in \mathcal{H}$ we have $x_i^\perp(h) = h(x_i)$.

It is convenient to think of a class $\mathcal{H}$ as a matrix M where each row of M corresponds to a function $h \in \mathcal{H}$ and each of its columns corresponds to a function $x^\perp \in \mathcal{H}^\perp$. The class $\mathcal{H}^\perp$ is the class represented by the transposed matrix M^T. The VC-dimension of M, denoted VC-dim($\mathcal{H}$), is the maximal number of columns d in which all of the 2^d combinations of 0's and 1's appear. The following claim relates the VC-dim of $\mathcal{H}$ and $\mathcal{H}^\perp$ is well known (see, for example, [BBK97]).

Claim 1 *For every class $\mathcal{H}$,* VC-dim($\mathcal{H}$) $\geq \lfloor \log$ VC-dim($\mathcal{H}^\perp$)$\rfloor$.

2.2 The Learning Models

In the exact learning [A88, L88] there is a boolean function F, called the *target function*, which is a member of a class of functions $\mathcal{C}$ defined over the domain X. An exact learning algorithm is given access to an equivalence query and the goal of the learning algorithm is to find and output a formula H that is logically equivalent to F. To ask an *equivalence query*, the learning algorithm supplies any function $H \in \mathcal{H}$ as input to an *equivalence oracle* and the reply of the oracle is either "YES", signifying that H is logically equivalent to F, or a *counterexample*, which is an assignment b such that $H(b) \neq F(b)$. For our algorithms we will represent a query to this oracle as a procedure EQ(H). We say that a class of boolean functions $\mathcal{C}$ is *exactly learnable* from $\mathcal{H}$ in polynomial time if for any $F \in \mathcal{C}$ over X there is an algorithm that is given access to an equivalence oracle and outputs a hypothesis $H \in \mathcal{H}$ that is logically equivalent to F using time polynomial in $\log|X|$ and the size of F (in some representation).

In the PAC-learning [Val84] there is an unknown distribution D defined over the domain set X. The learning algorithms in this model are given access to an *examples oracle* which, upon request, supplies the algorithm with a labeled example $(a, F(a))$ where a is drawn according to D and F is the unknown target formula. We say that a class of boolean functions $\mathcal{C}$ is PAC-learnable from $\mathcal{H}$ in polynomial time if there is an algorithm given access to an examples oracle such that for any $F \in \mathcal{C}$ over X, any $0 < \epsilon, \delta < 1$ and any distribution D on X, the algorithm runs in time polynomial in $\log|X|$, the size of F, $1/\delta$ and $1/\epsilon$ and with probability at least $1-\delta$ outputs a hypothesis $H \in \mathcal{H}$ that is ϵ-close to F with respect to the distribution D, i.e.,

$$\Pr_D[F(x) \neq H(x)] \leq \epsilon.$$

In the PAC-learning model with a malicious noise rate η, when the learning algorithm asks for an example from the example oracle, the oracle will, with probability $1-\eta$, provide the learning algorithm an example $(a, F(a))$ where a is chosen from X according to the distribution D and with probability η it will provide the learning algorithm any labeled example, i.e., (b, c) where c may not be $F(b)$. It is known [KL93] that in this model the desired accuracy cannot be less than $\epsilon = \eta/(1-\eta)$.

In the SQ-learning model [K93] the learner can ask a *statistical query* or ask for unlabeled examples. To ask a statistical query the learning algorithm supplies any polynomial time computable function $G : X \times \{0,1\} \to \{0,1\}$ and a constant $\tau > 1/poly$ to the *statistical oracle* and the reply of the oracle is a real number ξ such that

$$|E_D[G(x, F(x))] - \xi| \leq \tau.$$

When the algorithm asks for an unlabeled example the oracle returns an example $x \in X$ choosen according to the distribution D. An SQ-learning algorithm is defined the same as a PAC-learning algorithm except the algorithm has access to a different oracle.

In all of the above models we assume that the classes $\mathcal{C}$ and $\mathcal{H}$ are classes of formulas. We also assume that $\mathcal{C}$ and $\mathcal{H}$ are decidable in polynomial time, that is, there is an algorithm that decides whether a formula h is or isn't in $\mathcal{H}$. Notice that it is possible to have a formula h that is not in $\mathcal{H}$ but is logically equivalent to a formula in $\mathcal{H}$. The answer to decidability in this case would be "NO".

It is known from [A88] that if a class is exactly learnable in polynomial time from equivalence queries then it is PAC-learnable in polynomial time. It is also known from [K93] that if a class is SQ-learnable then it is PAC-learnable.

2.3 Halfspaces and Algebraic Surfaces

For an integer d we call the domain $X_n = \{0, 1, \ldots, n\}^d$ the *d-dimensional discretized space*. We will consider boolean functions in $\{0,1\}^{X_n}$, in particular, the class of *Halfspaces* over X_n which is the set of functions of the form

$$h(x) = \begin{cases} 1 & c_1x_1 + \cdots + c_dx_d \geq b \\ 0 & c_1x_1 + \cdots + c_dx_d < b \end{cases}$$

where $c_1, \ldots, c_d, b$ are integers.

Let C be the class of all halfspaces over $\{0, 1, \ldots, n\}^d$. We define C^r to be the class of all functions $f(g_1, \ldots, g_r)$ where f is any boolean function and $g_1, \ldots, g_r \in C$. Notice that any geometric object that is bounded by r hyperplanes is in C^r. Denote $C^* = \cup_r C^r$. The size of a geometric object G in C^* will be the minimal r such that $G \in C^r$, i.e., the minimal number of hyperplanes that bound G.

An algebraic surface of degree k is defined by a degree-k multivariate polynomial in $x_1, \ldots, x_d$. If $P(x)$ is such a polynomial then the corresponding function f gives the value 1 to every x such that $P(x) \geq 0$ and the value 0 otherwise. (Obviously a halfspace is a special case of such a surface with degree $k = 1$.) Let $\mathcal{C}$ be the class of all degree-k algebraic surfaces over $\{0, 1, \ldots, n\}^d$. The class $\mathcal{C}^*$ is called the class of degree-k semi-algebraic functions (over R^d).

For the PAC-model and SQ-model all the results of the paper are also true when the domain is $X = \{x \mid 0 \leq x \leq n\}^d$.

3 The Composition Theorem for the Exact Learning Model

In this section we present our main tool which is a reduction that constructs an algorithm that learns any combination of concepts in the above classes using any learning algorithm for any concept class of constant VC-dimension.

Let $\mathcal{C}$ be a class of boolean functions $g : X \to \{0,1\}^n$. Define the class $\mathcal{C}^*$ to be the set of all boolean functions that can be represented as $f(g_1, \ldots, g_m)$ where f is any boolean function,

$m \geq 0$ and $g_i \in \mathcal{C}$ for $i = 1, \ldots, m$. We define the size s of $f(g_1, \ldots, g_m)$ to be m.

Let P and N be set of points in X. We say that a boolean function $h : X \to \{0,1\}^n$ is *consistent* with (P, N) if h takes value 1 on all the points of P and value 0 on all the points of N. For two classes $(\mathcal{C}, \mathcal{H})$ we say that $(\mathcal{C}, \mathcal{H})$ has a *consistency algorithm* if there is an algorithm that takes (P, N) as input, runs in time polynomial in $|P \cup N|$ and with probability at least $1/2$ satisfies the following. If there is a function in $\mathcal{C}$ that is consistent with (P, N) the algorithm outputs "YES" and some $h \in \mathcal{H}$ that is consistent with (P, N). If there is no $h \in \mathcal{H}$ that is consistent with (P, N) then the algorithm outputs "NO". Notice that the algorithm can output anything if there is an $h \in \mathcal{H}$ that is consistent with (P, N) but no member of $\mathcal{C}$ is consistent with (P, N).

First we notice a relationship between exact learning algorithms and consistency algorithms.

Claim 2 *If $\mathcal{C}$ is PAC-learnable (or exact learnable) from $\mathcal{H}$ with equivalence queries then $(\mathcal{C}, \mathcal{H})$ has a consistency algorithm.*

Proof: Let ALG be the algorithm that learns $\mathcal{C}$ from $\mathcal{H}$. We can change ALG to an algorithm CON that is a consistency algorithm for $(\mathcal{C}, \mathcal{H})$ as follows. Algorithm CON runs algorithm ALG and simulate the distribution $D(x) = 1/|P \cup N|$ for $x \in P \cup N$ and $D(x) = 0$ for all other x. The value of ϵ is $1/(|P \cup N| + 1)$ and $\delta = 1/2$. If there is a consistent hypothesis in $\mathcal{C}$ then with probability at least $1/2$ we will get a consistent $h \in \mathcal{H}$. If there is no consistent in $\mathcal{H}$ the algorithm will get stuck, run more than it should, output an inconsistent h or it will output an $h \notin \mathcal{H}$. All those can be verified in polynomial time. □

Let Q be a set of points and CON be a consistency algorithm for $(\mathcal{C}, \mathcal{H})$. We would like to find the set

$$S_{\text{CON}}(Q) = \{P \mid \text{CON}(P, Q \backslash P) = \text{"YES"}\}.$$

This is the set of all possible splittings of Q into two sets $\{P, Q \backslash P\}$ where CON answer "YES" for $(P, Q \backslash P)$. One way to generate all the elements of $S_{\text{CON}}(Q)$ is using the following recursive approach. To find $S_{\text{CON}}(R \cup \{x\})$ we take all $P \in S_{\text{CON}}(R)$ and run CON on $(P \cup \{x\}, R \backslash P)$ and $(P, (R \backslash P) \cup \{x\})$. Then we include in $S_{\text{CON}}(R \cup \{x\})$ all the pairs for which CON answers "YES". Notice that the algorithm will answer "YES" for at least one of the pairs for a given P. This recursive execution has a tree structure of depth $|Q|$ and width at most $|S_{\text{CON}}(Q)|$. So the time complexity of this algorithm is bounded by

$$|Q| \cdot |S_{\text{CON}}(Q)| \cdot t$$

where t is the running time of CON. We now give a bound on the value of $|S_{\text{CON}}(Q)|$.

Lemma 1 *For any set of points Q,*

$$|S_{\text{CON}}(Q)| \leq |Q|^{\text{VC-dim}(\mathcal{H})}.$$

Proof: Let $Q = \{x_1, x_2, \cdots, x_{|Q|}\}$. Notice that the number of Ps in $S_{\text{CON}}(Q)$ is at most the number of the combinations

$$F = \{(h(x_1), \ldots, h(x_{|Q|})) \mid h \in \mathcal{H}\}.$$

By Sauer's Lemma the number of distinct elements in F is at most $|Q|^{\text{VC-dim}(\mathcal{H})}$. □

The algorithm in Figure 1 takes an algorithm CON and creates a set of all $((P, Q \backslash P), h)$ such that $P \in S_{\text{CON}}(Q)$ and h is consistent with $(P, Q \backslash P)$. We are now ready to state and prove our main result.

Algorithm $S_{\text{CON}}(Q)$.

$\mathcal{S} \leftarrow \{((\emptyset, \emptyset), 0)\}$.
Let $Q = \{x_1, \ldots, x_q\}$.
For $i = 1$ to q do
  $\mathcal{T} \leftarrow \emptyset$.
  For all $((P, N), h) \in \mathcal{S}$
    Run CON$(P \cup \{x_i\}, N)$ and if the
    answer is ("YES", h') do $\mathcal{T} \leftarrow \mathcal{T} \cup \{((P \cup \{x_i\}, N), h')\}$
    Run CON$(P, N \cup \{x_i\})$ and if the
    answer is ("YES", h') do $\mathcal{T} \leftarrow \mathcal{T} \cup \{((P, N \cup \{x_i\}), h')\}$
  $\mathcal{S} \leftarrow \mathcal{T}$.
Output($\mathcal{S}$)

Figure 1: *An algorithm for generating $S_{\text{CON}}(Q)$.*

Theorem 2 *Let $\mathcal{C}$ and $\mathcal{H}$ be classes of boolean functions over domain X where $\mathcal{C} \subseteq \mathcal{H}$. If the concept class $\mathcal{C}$ is learnable from $\mathcal{H}$ using q equivalence queries in time T then the concept class $\mathcal{C}^*$ is learnable using*

$$p = (2mq)^{\text{VC-dim}(\mathcal{H})\text{VC-dim}(\mathcal{H}^\perp)}$$

equivalence queries and poly(p, T) time where m is the size of the target, i.e., number of functions in $\mathcal{C}$ on which the target function is based.

Let ALG be the algorithm that learns $\mathcal{C}$ from $\mathcal{H}$ and CON be a consistency algorithm for $(\mathcal{C}, \mathcal{H})$ that is generated from ALG using Claim 2. Let $F = f(g_1, \ldots, g_m)$ be the target function $g_1, \ldots, g_m \in \mathcal{C}$. Consider the algorithm ALG* in Figure 2. To understand the algorithm and prove its correctness and complexity we prove the following claims. Some of these claims are from [BBK97].

Claim 3 *[BBK97]. For every t (i.e., t is the number of hypotheses h_t), the number of entries $y \in \{0,1\}^t$ such that $M(y) \neq \star$ is at most*

$$t^{\text{VC-dim}(\mathcal{H}^\perp)}.$$

Proof: Notice that $M(y) \neq \star$ implies that $y = (h_1(x), \ldots, h_t(x))$ for some x in A. Therefore, the number of non-star entries of M is at most the number of different values of $(h_1(x), \ldots, h_t(x))$ over all x. These are simply vectors in the space $\mathcal{H}^\perp$. Using Sauer's Lemma the result follows. □

The next claim shows that if in the collection of our hypotheses we have the correct functions $g_1, \ldots, g_m$ then the protocol will halt.

Claim 4 *[BBK97]. If $g_1, \ldots, g_m \in \{h_1, \ldots, h_t\}$ then the algorithm will only execute steps 5-7 several times (at most t^v where $v = \text{VC-dim}(\mathcal{H}^\perp)$) and then it will get the answer "yes" to one of its equivalence queries (in which case the algorithm halts).*

Proof: We will show that if $g_1, \ldots, g_m \in \{h_1, \ldots, h_t\}$ then for every counterexample a we have $M(h_1(a), \ldots, h_t(a)) = \star$ (hence, in this case, the condition in step 7 holds and the condition in step 8 does not). This implies that the algorithm will execute only steps 5-7 (see the condition in 7). Suppose $M(h_1(a), \ldots, h_t(a)) \neq \star$. Then (by the description of the algorithm) there is an assignment

Algorithm ALG*.

1. $Q \leftarrow \emptyset$.
2. Let $\mathcal{S}_{\text{CON}}(Q) = \{((P_1, N_1), h_1), \ldots, ((P_t, N_t), h_t)\}$.
3. Define a table $M(y_1, \ldots, y_t) = \star$ for $(y_1, \ldots, y_t) \in \{0,1\}^t$
4. $A \leftarrow \emptyset$.
5. Define a hypothesis

$$H(y_1, \ldots, y_t) = \begin{cases} M(y_1, \ldots, y_t) & \text{if } M(y_1, \ldots, y_t) \neq \star \\ 0 & \text{otherwise} \end{cases}.$$

6. Ask $EQ(H(h_1(x), \cdots, h_t(x))) \rightarrow a$. If the answer is "yes" then return $H(h_1(x), \cdots, h_t(x))$ and Halt.
7. If $M(h_1(a), \ldots, h_t(a)) = \star$ then set $M(h_1(a), \ldots, h_t(a))$ to 1, set $A \leftarrow A \cup \{a\}$, and goto 5.
8. If $M(h_1(a), \ldots, h_t(a)) \neq \star$ then there exists $a' \in A$ such that

$$(h_1(a), \ldots, h_t(a)) = (h_1(a'), \ldots, h_t(a')).$$

9. $Q \leftarrow Q \cup \{a, a'\}$.
10. goto 2.

Figure 2: Algorithm ALG* for learning $\mathcal{C}^\star$.

$a' \in A$ such that $(h_1(a), \ldots, h_t(a)) = (h_1(a'), \ldots, h_t(a'))$. On the other hand since a is a counterexample then (by the way H is defined) $f(g_1(a), \ldots, g_m(a)) \neq f(g_1(a'), \ldots, g_m(a'))$. However, since we have $h_i(a) = h_i(a')$ for all i and because $g_1, \ldots, g_m \in \{h_1, \ldots, h_t\}$ then in particular $g_i(a) = g_i(a')$ for every i. Therefore $f(g_1(a), \ldots, g_m(a)) = f(g_1(a'), \ldots, g_m(a'))$. A contradiction. □

The next claim shows that the number of times that the main loop (steps 2-10) is performed is at most qm.

Claim 5 *We have* $|Q| \leq 2qm$.

Proof: We will show that after executing steps 2-9 r times we will have the following. For every g_i there is a hypothesis $h_{u(i)}$ that is equivalent to the hypothesis that we would get from running algorithm ALG for at least r_i phases for the target g_i and $r = r_1 + \cdots + r_m$. By "step" we mean the running of the algorithm until a new equivalence query is asked. We show the above by showing that executing steps 2-9 is equivalent to running the algorithm ALG at least one more step for some g_i. This will imply that steps 2-9 are executed at most qm times and therefore $|Q| \leq 2qm$.

To show the above, suppose we get to step 8 and we have $M(h_1(a), \ldots, h_t(a)) = \xi \neq \star$. There is a' such that

$$(h_1(a), \ldots, h_t(a)) = (h_1(a'), \ldots, h_t(a'))$$

and

$$f(g_1(a), \ldots, g_m(a)) \neq f(g_1(a'), \ldots, g_m(a')).$$

From the latter we must have $g_i(a) \neq g_i(a')$ for some i. Now because $h_{u(i)}(a) = h_{u(i)}(a')$ we have that either a or a' is a counterexample for $h_{u(i)}$. Therefore using this counterexample one of the algorithms CON $(P_{u(i)} \cup \{w\}, N_{u(i)})$ or CON$(P_{u(i)}, N_{u(i)} \cup \{w\})$ for some $w \in \{a, a'\}$ will generate the hypothesis generated by ALG in one more steps. □

Now by Lemma 1 and Claim 5 we have

Claim 6 *We have*

$$t \leq (2mq)^{\text{VC-dim}(\mathcal{H})}.$$

Now since $t \leq (2mq)^{\text{VC-dim}(\mathcal{H})}$, the size of the table is at most

$$t^{\text{VC-dim}(\mathcal{H}^\perp)} \leq (2mq)^{\text{VC-dim}(\mathcal{H})\text{VC-dim}(\mathcal{H}^\perp)}$$

as stated in Theorem 2.□

To learn a geometric object bounded by m hyperplanes in the d dimensional space of the $[0, n]^d$ lattice the algorithm complexity is

$$(2m \log n)^{(d+1)^2}.$$

The algorithm in [BBK97] has complexity (at least)

$$(2m \log^d n)^{O(d^3)}.$$

We now can prove a more general theorem.

Theorem 3 *Let $\mathcal{C}$ and $\mathcal{H}$ and $\mathcal{G}$ be classes of boolean functions over the domain X where $\mathcal{C} \subseteq \mathcal{H} \subseteq \mathcal{G}$. Suppose we have the following*

1. *There is an algorithm A that with an input (P, N) runs in polynomial time in $|P \cup N|$ and decides whether there is a consistent hypothesis $h \in \mathcal{C}$ with (P, N) or if there is no hypothesis $h \in \mathcal{H}$ that is consistent with (P, N).*
2. *For every polynomial number of functions $g_1, \ldots, g_m \in \mathcal{G}$ the number of distinct vectors $(g_1(x), \ldots, g_m(x))$ over all $x \in X$ is $\Gamma(m)$.*

If the concept class $\mathcal{C}$ is learnable from $\mathcal{G}$ using q equivalence queries in time T then the concept class $\mathcal{C}^\star$ is learnable using

$$p = \Gamma\left((2mq)^{\text{VC-dim}(\mathcal{H})}\right)$$

equivalence queries and poly(p, T) time where m is the size of the target, i.e., number of functions in $\mathcal{C}$ on which the target function is based.

Proof: We follow the steps of the algorithm ALG* in figure 2. Step 2 in the algorithm cannot be simulated. Instead we use algorithm A to generate the set $\{(P_1, N_1), \ldots, (P_t, N_t)\}$ and then run the algorithm that learns $\mathcal{C}$ from $\mathcal{G}$ to find hypotheses $g_1, \ldots, g_t$ consistent with $(P_1, N_1), \ldots, (P_t, N_t)$, respectively.

As in the proof of Theorem 1 we have $|Q| \leq 2qm$ and $t \leq (2mq)^{\text{VC-dim}(\mathcal{H})}$. Now the number of possible entries in the table will be at most $\Gamma(t)$. □

Notice that $\mathcal{G}$ may have a nonconstant VC-dimension. If these conditions are true and Γ is polynomial then $\mathcal{C}^\star$ is learnable in polynomial time.

We now show the following.

Theorem 4 *Let $\mathcal{C}$ be a class that is PAC-learnable from a class of constant VC-dimension $\mathcal{H}$. If $\mathcal{C}$ is exactly learnable from $\mathcal{H}^\star$ in polynomial time then $\mathcal{C}^\star$ is learnable in polynomial time.*

Proof: By Claim 2 we have the first condition of Theorem 3. To prove that condition (2) is also true with polynomial Γ, let $g_1, \ldots, g_m$ be in $\mathcal{H}^*$ and m be polynomial (in the size of the target and $\log |X|$). Since $g_i \in \mathcal{H}^*$ and is of polynomial size (because the algorithm runs in polynomial time) we have $g_i = f_i(g_{i,1}, \ldots, g_{i,l_i})$ where l_i is polynomial and $g_{i,j} \in \mathcal{H}$. Now the number of distinct vectors $(g_1(x), \ldots, g_m(x))$ for $x \in X$ is at most the number of distinct vectors $(g_{i,j}(x))_{i,j}$ for $x \in X$. Since $g_{i,j} \in \mathcal{H}$ and $\mathcal{H}$ is of constant VC-dimension then we know that this number is polynomial in $\sum_i l_i m$ and therefore is polynomial. □

Notice here that $\mathcal{H}^*$ may not be of constant VC-dimension. One of the algorithms that learns $\mathcal{C}$ using hypotheses from $\mathcal{C}^*$ is the randomized Halving algorithm. At each step of the randomized Halving algorithm the algorithm randomly chooses a polynomial number of consistent hypotheses and asks equivalence queries with their majority. So the hypothesis class is $\mathcal{C}^*$.

3.1 Lower Bound

In this short subsection we show that the Composition Theorem cannot be extended to a class with a nonconstant VC-dimension. To prove the lower bound we use a simple argument.

Theorem 5 *For every d there is $\mathcal{C}$ of VC-dimension d such that learning $F \in \mathcal{C}^*$ will require at least $(m/d)^d$ equivalence queries where $m = size(F)$.*

Proof: It is known that there is a class of VC-dimension d and m functions $g_1, \ldots, g_m$ such that the number of distinct vectors $(g_1(x), \ldots, g_m(x))$ is at least

$$\sum_{i=1}^{d} \binom{m}{i} \geq \left(\frac{m}{d}\right)^d.$$

Now this means that the class $\{f(g_1, \ldots, g_m) \mid f\} \subset \mathcal{C}^*$ has VC-dimension $(m/d)^d$ and therefore we need at least $(m/d)^d$ equivalence queries to learn it. □

4 The Composition Theorem for Other Learning Models

Let ALG be a PAC-learning algorithm that learns $\mathcal{C}$ from $\mathcal{H}$. As was shown in Claim 2 using this algorithm we can also build a consistency algorithm CON for $(\mathcal{C}, \mathcal{H})$.

Let $h_1, \ldots, h_t \in \mathcal{H}$. We define the set of domains $\mathcal{W}(h_1, \ldots, h_t)$ to be the set of all $W_a = \{x \mid (h_1(x), \ldots, h_t(x)) = a\}$ for $a \in \{0,1\}^t$. Notice that $\mathcal{W}(h_1, \ldots, h_t)$ is a partition of the domain X and

$$|\mathcal{W}(h_1, \ldots, h_t)| \leq t^{\text{VC-dim}(\mathcal{H}^\perp)}.$$

Now the algorithm for learning $F \in \mathcal{C}^*$ is in Figure 3.

In the first step the algorithm takes r examples and then in step 2 finds all possible splittings of the examples using the algorithm in Figure 1. Each splitting $(P_i, Q \backslash P_i)$ defines a hypothesis $h_i \in \mathcal{H}$ that is consistent with $(P_i, Q \backslash P_i)$. The set of all hypothesis $\{h_1, \ldots, h_t\}$ defines at most $t^{\text{VC-dim}(\mathcal{H}^\perp)}$ domains. The domains are all

$$W_a = \{x \mid (h_1(x), \ldots, h_t(x)) = a\}$$

that are not empty. We then take another round of examples and define the value of each domain to be the majority of the labels of the examples that fall in this domain. Domains that contains no examples are given a value 0.

Notice that these steps can be done in the SQ-learning model. The first step does not use the labels of the examples. In the second step we can find the expectation of the sign in each domain W_a using

$$\mathrm{E}[F \mid x \in W_a] = \frac{\mathrm{E}[h_1^{a_1} \cdots h_t^{a_t} F]}{\mathrm{E}[h_1^{a_1} \cdots h_t^{a_t}]}$$

where $h_i^{a_i}$ is h_i if $a_i = 1$ and $\neg h_i$ otherwise. Now if $\mathrm{E}[h_1^{a_1} \cdots h_t^{a_t}]$ is "too small" we can disregard this domain. If $\mathrm{E}[F \mid x \in W_a] \geq 1/2$ then we define $H(a) = 1$. Otherwise, we define $H(a) = 0$.

Learning $\mathcal{C}^*$.

1. Get r examples E. Let $Q \leftarrow \{x \mid (x, y) \in E\}$.
2. Let $S_{CON}(Q) = \{((P_1, Q \backslash P_1), h_1), \ldots, ((P_t, Q \backslash P_t), h_t)\}$.
3. Get s examples R.
4. Define $H(y_1, \ldots, y_t)$ as follows: For every y if $W_y \in \mathcal{W}(h_1, \ldots, h_t)$ is not empty then find all examples $R_y \subset R \cap W_y$ and define
$$H(y) = Maj_{(z, F(z)) \in R_y}(F(z)).$$
If R_y or W_y is empty then define $H(y) = 0$
5. Return $H(h_1, \ldots, h_t)$.

Figure 3: Algorithm for learning $\mathcal{C}^*$

4.1 Analysis

We now give the analysis for the PAC model with malicious noise η. The analysis for the SQ-model is very similar and we leave it to the reader. We first give Chernoff bounds

Lemma 6 (Chernoff) *Let X_i be independent random variables all with mean μ such that for all i, $X_i \in \{0,1\}$. Then for any ξ*

$$\Pr\left[\frac{1}{n}\sum_{i=1}^{n} X_i \geq (1+\xi)\mu\right] \leq e^{-\xi^2 n\mu/3}$$

and

$$\Pr\left[\frac{1}{n}\sum_{i=1}^{n} X_i \leq (1-\xi)\mu\right] \leq e^{-\xi^2 n\mu/2}.$$

The VC-dimension result is

Lemma 7 (BEHW89) *Let $\mathcal{C}$ be class of functions and $\mathcal{H} \supset \mathcal{C}$. Then the Occam algorithm (that take M examples and find a consistent hypothesis from $\mathcal{H}$) with*

$$M(\epsilon, \delta) = \frac{4}{\epsilon}\left(2 \cdot \text{VC-dim}(\mathcal{H}) \log \frac{13}{\epsilon} + \log \frac{2}{\delta}\right)$$

examples is a PAC-learning algorithm for $\mathcal{C}$.

Let $F = f(g_1, \ldots, g_m)$ be the target function where $g_i \in \mathcal{C}$. We will show that the algorithm in Figure 3 achieves an error $\eta/(1-\eta) + \beta$ in time polynomial in $1/\beta$, $1/\delta$, $1/\epsilon$, m and $1/(1-\eta)$.

Let

$$\lambda = \beta\eta/16,$$

and

$$r = \frac{8m}{(1-\eta)\lambda}\left(2 \cdot \text{VC-dim}(\mathcal{H}) \log \frac{13m}{\lambda} + \log \frac{6m}{\delta}\right)$$

By Chernoff bound since $|Q| = r \geq \frac{s}{1-\eta} \log \frac{3}{\delta}$ and the probability that $x \in Q$ is chosen according to a distribution D is $1-\eta$ we have: With probability at least $1-\delta/3$, $(1-\eta)r/2$ of the examples in Q are chosen according to the distribution D.

Let $((P_{j_i}, Q \backslash P_{j_i}), h_{j_i})$ be the triple for which g_i is consistent on $(P_{j_i}, Q \backslash P_{j_i})$. Since h_{j_i} is also consistent on $(P_{j_i}, Q \backslash P_{j_i})$ and since, with probability at least $1-\delta/3$,

$$\frac{(1-\eta)r}{2} = \frac{4m}{\lambda}\left(2 \cdot \text{VC-dim}(\mathcal{H}) \log \frac{13m}{\lambda} + \log \frac{6m}{\delta}\right)$$

of the examples in Q came from the distribution D, by Lemma 7 we have with probability at least $1-\delta/(3m)$

$$\Pr_D[h_{j_i} \neq g_i] \leq \frac{\lambda}{m}.$$

So assuming we know h_{j_i} and we know f, with probability at least $1-\delta/3$ we have

$$\Pr_D[f(g_1, \ldots, g_m) \neq f(h_{j_1}, \ldots, h_{j_m})] \leq \lambda.$$

Let T be the set of all points in which $f(g_1, \ldots, g_m)$ and $f(h_{j_1}, \ldots, h_{j_m})$ disagree. We have

$$D(T) \leq \lambda.$$

Notice that $\mathcal{W}(h_1, \ldots, h_t)$ is a subpartition of the partition $\mathcal{W}(h_{j_1}, \ldots, h_{j_m})$. Since the algorithm do not know h_{j_i} and f it will learn a new $\hat{f}$ such that $\hat{f}(h_1, \ldots, h_t)$ is good approximation of $f(g_1, \ldots, g_m)$. Now the algorithm learns $\hat{f}$ for different values of $(h_1, \ldots, h_t)$. Let $\mathcal{W}$ be the set of all nonempty domains W_a. By Lemma 1 the number of nonempty domains in $\mathcal{W}$ is at most

$$d \leq t^{\text{VC-dim}(\mathcal{H})} \leq r^{\text{VC-dim}(\mathcal{H})\text{VC-dim}(\mathcal{H}^\perp)}.$$

Now we define the *important domains* $\mathcal{I} \subset \mathcal{W}$ to be the set of all domains W_a such that

$$D(W_a) \geq \lambda/d.$$

We will show that those domains that are not important have small weight. We have

$$\sum_{D(W_a) < \lambda/d} D(W_a) \leq \frac{\lambda}{d} d \leq \lambda.$$

Therefore we may ignore the nonimportant domains.

In the second round we choose

$$s = \frac{2d}{\lambda^3(1-\eta)}\left(\log \frac{9}{\delta} + \log d\right)$$

examples. For the sake of analysis, we assume that we have three phases. In the first phase the teacher chooses s examples E_1 of the target function. In the second phase the teacher chooses each example in E_1 with probability $1-\eta$ and puts it in a set E_2. In the third phase the teacher chooses $|E_1 \backslash E_2|$ arbitrary new labeled examples E_3 and then sends $E_2 \cup E_3$ to the learner.

Now we will define for each important domain W_a four integers $(t_a, t_a^{out}, t_a^{in}, t_a^{err})$. The integer t_a is the number of examples in $E_2 \cap W_a$. The integer t_a^{out} is the number of examples in $(E_1 \backslash E_2) \cap W_a$ and t_a^{in} is the number of examples in $E_3 \cap W_a$. The integer t_a^{err} is the number of examples in $E_2 \cap T \cap W_a$. Let $t^{in} = \sum_a t_a^{in}$ and $t^{err} = \sum_a t_a^{err}$.

Now since each important domain has distribution greater than λ/d, by Chernoff bound we have

$$\begin{aligned}\Pr\left[(\forall W_a \in \mathcal{I})\ \frac{t_a}{s} \leq (1-\lambda)(1-\eta)D(W_a)\right] &\leq \\ \sum_{W_a \in \mathcal{I}} e^{-\lambda^2 s(1-\eta)D(W_a)/2} &\leq \\ de^{\lambda^3 s(1-\eta)/(2d)} &\leq \frac{\delta}{9}.\end{aligned}$$

Using again Chernoff bound we can show that with probability at least $1-2\delta/9$ we have $t^{in} \leq (1+\lambda)\eta s$, and $t^{err} \leq 2\lambda s$.

Therefore with probability at least $1-\delta/3$ we have

(1) For every important W_a, $(1-\lambda)D(W_a)(1-\eta)s \leq t_a$,

(2) $t^{in} \leq (1+\lambda)\eta s$, and,

(3) $t^{err} \leq 2\lambda s$.

Now by ignoring the nonimportant domains we already have an error λ. Notice that because $\mathcal{W}(h_1, \ldots, h_t)$ is a subpartition of $\mathcal{W}(h_{j_1}, \ldots, h_{j_m})$ the function f has the same value on all the points in $W_a \backslash T$. So if we define $H(W_a)$ to be $f(W_a \backslash T)$ then the only error W_a will contribute is at most $D(T \cap W_a)$. The total error in this case is at most $D(T) \leq \lambda$. The third kind of error is when $H(W_a)$ is defined to be $\neg f(W_a \backslash T)$. This can happen only if $t_a^{in} + t_a^{err} > t_a - t_a^{err}$. So the total error is at most

$$2\lambda + \sum_{t_a^{in} + t_a^{err} > t_a - t_a^{err}} D(W_a).$$

Now

$$\begin{aligned}&\sum_{t_a^{in} + t_a^{err} > t_a - t_a^{err}} D(W_a) \\ &\leq \frac{1}{(1-\lambda)(1-\eta)} \frac{1}{s} \sum_{t_a^{in} + t_a^{err} > t_a - t_a^{err}} t_a \\ &\leq \frac{1}{(1-\lambda)(1-\eta)} \frac{1}{s} \sum_a t_a^{in} + 2t_a^{err} \\ &\leq \frac{1}{(1-\lambda)(1-\eta)}\left(\frac{t^{in}}{s} + 2\frac{t^{err}}{s}\right) \\ &\leq \frac{1}{(1-\lambda)(1-\eta)}((1+\lambda)\eta + 4(1+\lambda)\lambda) \\ &= \frac{\eta}{1-\eta} \cdot \frac{1+\lambda}{1-\lambda}\left(1 + 4\frac{\lambda}{\eta}\right) \\ &\leq \frac{\eta}{1-\eta} + \frac{\beta}{3}.\end{aligned}$$

Therefore, with probability at least $1-\delta$ the error is

$$2\lambda + \left(\frac{\eta}{1-\eta} + \frac{\beta}{3}\right) \leq \frac{\eta}{1-\eta} + \beta.$$

It is also easy to verify that the time complexity is polynomial if VC-dim$(\mathcal{H})$ is constant.

References

[A88] D. Angluin. Queries and Concept Learning. *Machine Learning*, pp 319–342, 2, 4, 1988.

[Aue93] P. Auer, "On-line Learning of Rectangles in Noisy Environments", *Proc. of 6th Annu. ACM Workshop on Comput. Learning Theory*, pages 253–261, 1993.

[BBK97] S. Ben-David, N. H. Bshouty and E. Kushilevitz, A Composition Theorem for Learning Algorithms with Application to Geometric Concepts Classes. *Proc. of the 29th Annual ACM Symposium on the Theory of Computing*, 1997.

[BCH94] N. H. Bshouty, Z. Chen, and S. Homer. "On Learning Discretized Geometric Concepts", *Proc. of 35th Annu. Symp. on Foundations of Computer Science*, pages 54–63, 1994.

[BEHW89] A. Blumer, A. Ehrenfeucht, D. Haussler, M. Warmuth. Learnability and the Vapnik-Chervonenkis Dimension *Journal of the Association for Computing Machinery*, 36 (04), page 929-965, 1989.

[BGMST96] N. H. Bshouty, S. A. Goldman, H. D. Mathias, S. Suri, and H. Tamaki. "Noise-Tolerant Distribution-Free Learning of General Geometric Concepts" *Proc. of the 28th Annu. ACM Symp. on Theory of Computing*, pages 151–160, 1996.

[CM92] Z. Chen and W. Maass. "On-line Learning of Rectangles", In *Proc. of 5th Annu. ACM Workshop on Comput. Learning Theory*, 1992.

[HSW90] D. Helmbold, R. Sloan, and M. K. Warmuth, "Learning Nested Differences of Intersection-Closed Concept Classes", *Machine Learning*, Vol 5, pp. 165-196, 1990.

[K93] M. Kearns "Efficient Noise-Tolerant Learning from Statistical Queries" Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, pp. 392-401, 16-18 May 1993.

[KL93] M. Kearns, M. Li, "Learning in the Presence of Malicious Errors" In: SIAM Journal on Computing, 22 (04), page 807-837 (1993).

[L88] N. Littlestone, "Learning when Irrelevant Attributes Abound: A New Linear-Threshold Algorithm" *Machine Learning*, Vol. 2, pp. 285–318, 1988.

[L89] N. Littlestone, "Mistake bounds and logarithmic linear-threshold learning algorithms" PhD thesis, U.C. Santa Cruz, March 1989.

[MT89] W. Maass and G. Turan. "On the Complexity of Learning from Counterexamples", *Proc. of 30th Annu. Symp. on Foundations of Computer Science*, pages 262–273, 1989.

[Mur71] S. Muroga, "Threshold Logic and its Applications", Wiley Interscience, 1971.

[Sim95] H. U. Simon, "Learning Decision Lists and Trees with Equivalence-Queries", *Proc. of 2nd EuroCOLT*, LNAI, Vol. 904, pp. 322-336, 1995.

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

[W68] H. E. Warren, "Lower Bounds for Approximation by Nonlinear Manifolds", *Transactions of the AMS*, 133, pp. 167-178, 1968.