Machine Learning 33, 87–104 (1998)
© 1998 Kluwer Academic Publishers. Manufactured in The Netherlands.

Self-Directed Learning and Its Relation to the VC-Dimension and to Teacher-Directed Learning

SHAI BEN-DAVID shai@cs.technion.ac.il
NADAV EIRON nadav@cs.technion.ac.il
Computer Science Department, Technion, Haifa 32000, Israel

Editor: David Haussler

Abstract. We study the *self-directed* (SD) *learning* model. In this model a learner chooses examples, guesses their classification and receives immediate feedback indicating the correctness of its guesses. We consider several fundamental questions concerning this model: the parameters of a task that determine the cost of learning, the computational complexity of a student, and the relationship between this model and the *teacher-directed* (TD) learning model. We answer the open problem of relating the cost of self-directed learning to the VC-dimension by showing that no such relation exists. Furthermore, we refute the conjecture that for the intersection-closed case, the cost of self-directed learning is bounded by the VC-dimension. We also show that the cost of SD learning may be arbitrarily higher that that of TD learning.

Finally, we discuss the number of queries needed for learning in this model and its relationship to the number of mistakes the student incurs. We prove a trade-off formula showing that an algorithm that makes fewer queries throughout its learning process, necessarily suffers a higher number of mistakes.

Keywords: mistake-bound learning, self-directed learning, VC-dimension, teacher-directed learning

1. Introduction

Since defined by Littlestone (Littlestone, 1988, 1989), the *mistake-bound* model of learning has attracted a considerable amount of attention (e.g., Littlestone, 1988, 1989; Littlestone & Warmuth, 1994; Blum, 1990, 1992; Maass, 1991; Chen & Maass, 1994; Helmbold, Littlestone, & Long, 1992; Goldman, Rivest, & Schapire, 1993; Goldman & Sloan, 1994; Ben-David, Kushilevitz, & Mansour, 1995; Rivest & Yin, 1995; Yin, 1995; Frances & Litman, 1995). In this model the learner has to make predictions on the next instance based on the previous instances that it has already seen and their labels. The quantity that the learner should try to minimize is the number of mistakes it makes along this process.

Several variants of this model were considered, allowing the learner various degrees of freedom in choosing the instances presented to him. These variants include (in increasing power of the learner):

Some of the results in this paper appeared in Ben-David, Eiron, & Kushilevitz (1995).

- The *on-line* model (Littlestone, 1988, 1989), in which the sequence of instances is chosen by an adversary (the teacher) and the instances are presented to the learner one-by-one.
- The *worst sequence* (off-line) model (Ben-David, Kushilevitz, & Mansour, 1995), in which the sequence of instances is still chosen by the adversary but the whole sequence (without the labels) is presented to the learner before the prediction process starts.
- The *best sequence* (off-line) model (Ben-David, Kushilevitz, & Mansour, 1995), in which the whole sequence of instances is chosen by the learner before the prediction process starts.[1]
- The *self-directed* model (Goldman, Rivest, & Schapire, 1993; Goldman & Sloan, 1994), in which the learner may choose the sequence of instances *adaptively*; i.e., each instance is chosen only after seeing the labels of all previous instances.

Another related model that was studied (Goldman, Rivest, & Schapire, 1993; Goldman & Kearns, 1995; Rivest & Yin, 1995) is the *teacher-directed* model. In this model, a helpful teacher, who knows the target function, presents labeled examples to the learner. The teaching process is over once only a single concept in $\mathcal{C}$ is consistent with the examples presented.

Denote by $M_{\text{on-line}}(\mathcal{C})$, $M_{\text{worst}}(\mathcal{C})$, $M_{\text{best}}(\mathcal{C})$, $M_{\text{sd}}(\mathcal{C})$ and $M_{\text{td}}(\mathcal{C})$ the number of mistakes made by the best learning algorithm in the on-line, worst sequence, best sequence, self-directed and teacher-directed models, respectively, on the worst target concept in a concept class $\mathcal{C}$.

Goldman & Sloan (1994) consider the relations between M_{sd} and the VC-dimension. They give examples where the VC-dimension is 2 and $M_{\text{sd}} = 3$ (this is generalized in Ben-David, Kushilevitz, & Mansour (1995) where examples of classes with VC-dimension d and $M_{\text{sd}} = d + 1$ are presented). They ask as an open problem whether there exists a constant α such that $M_{\text{sd}} \leq \alpha \cdot VCdim$, for all concept classes. We answer this open problem by showing that no such constant exists. In fact, we prove a much stronger statement: For any d and n, there exists a concept class $\mathcal{C}_n^d$ having VC-dimension d and self-directed learning complexity $M_{\text{sd}}(\mathcal{C}_n^d) = \Omega(n)$. Furthermore, Goldman & Sloan (1994, Conjecture 11) conjectured that for so-called “intersection-closed” concept classes $M_{\text{sd}}(\mathcal{C}) \leq VCdim(\mathcal{C})$. We disprove this conjecture by presenting for every even d an intersection-closed concept class with VC-dimension d, and $M_{\text{sd}} = \frac{3}{2}d$.

Previous work (Goldman, 1990; Goldman & Kearns, 1995; Goldman, Rivest, & Schapire, 1993; Goldman & Sloan, 1994) presented many classes for which self-directed learning is more efficient than teacher-directed learning. We show that there are some classes, $\mathcal{C}_n^d$, which are hard for the self-directed learner to learn, yet easy to teach by a helpful teacher in the teacher-directed model. We will show that while $M_{\text{sd}}(\mathcal{C}_n^d) = \Omega(n)$, $M_{\text{td}}(\mathcal{C}_n^d) = d + 1$. This result improves upon the results of (Rivest & Yin, 1995), where cryptographic assumptions and constraints on the computational power of the self-directed learner are required to obtain similar results.

In Section 5 we offer a generic algorithm for the self-directed learning task. This algorithm is a natural adaptation of Littlestone’s Halving Algorithm. We discuss the mistake-bound performance of this algorithm and then refine it and define a family of generic self-directed algorithms having increasingly better mistake-bound performance. These algorithms have polynomial running time (in $|\mathcal{C}|$) and for many natural concept classes, including classes for which self-directed learning is strictly better than best sequence off-line learning, they achieve the optimal self-directed bound. On the other hand, we show that there are concept classes for which the number of mistakes these algorithms make can be arbitrarily higher than the optimum.

We conclude, in Section 6, with a preliminary discussion of the issue of the number of *queries* made by a self-directed algorithm. We prove a trade-off formula showing that lowering the mistake bound of a self-directed algorithm necessarily results in an increase in the number of queries it has to go through. Loosely speaking, if the learner wishes to reduce the number of mistaken guesses it makes in the process of learning, it inevitably has to pay by asking many questions for which it already has a pretty good idea of what the answers will be.

2. Preliminaries

2.1. Basic definitions

In this section we formally present the mistake-bound learning model in all its variants. We follow definitions given in (Littlestone, 1988, 1989; Goldman, Rivest, & Schapire, 1993; Goldman & Sloan, 1994; Ben-David, Kushilevitz, & Mansour, 1995).[2] We restrict our definitions to finite instance spaces. However, it should be noted, that these definitions may be extended to infinite instance spaces.

Let $\mathcal{X}$ be any finite set, and let $\mathcal{C}$ be a collection of boolean functions defined over $\mathcal{X}$ (i.e., $\mathcal{C} \subseteq \{0,1\}^{\mathcal{X}}$). We refer to $\mathcal{X}$ as the *instance space* and to $\mathcal{C}$ as the *concept class*. An *on-line learning algorithm with respect to a concept class* $\mathcal{C}$ is an algorithm $\mathcal{A}$ that works in steps as follows: In the ith step the algorithm is presented with a new element $x_i \in \mathcal{X}$. It then outputs its prediction p_i and in response it gets the true value $c_t(x_i)$, where $c_t \in \mathcal{C}$ denotes the *target* function. The prediction p_i may depend on the values it has seen so far (i.e., $c_t(x_1), \ldots, c_t(x_{i-1})$) and, of course, on the concept class $\mathcal{C}$. The process continues until all the elements of $\mathcal{X}$ have been presented. Let $\sigma = x_1, x_2, \ldots, x_n$ denote the order in which the elements of $\mathcal{X}$ are presented to the learning algorithm. Denote by $M(\mathcal{A}, \sigma, c_t)$ the number of mistakes made by the algorithm $\mathcal{A}$ on a sequence σ and target function $c_t \in \mathcal{C}$ (i.e., the number of elements for which $p_i \neq c_t(x_i)$). Define the mistake bound of the on-line algorithm as

$$M(\mathcal{A}) \stackrel{\triangle}{=} \max_{\sigma, c_t \in \mathcal{C}} M(\mathcal{A}, \sigma, c_t).$$

Finally, let

$$M_{\text{on-line}}(\mathcal{C}) \stackrel{\triangle}{=} \min_{\mathcal{A}} M(\mathcal{A}) = \min_{\mathcal{A}} \max_{\sigma, c_t \in \mathcal{C}} M(\mathcal{A}, \sigma, c_t).$$

An *off-line learning algorithm* is an algorithm $\mathcal{A}$ that is given (in advance) the actual sequence σ as an input. The learning process remains unchanged (except that each prediction

p_i can now depend on σ and not only on $\mathcal{C}$). Denote by $M(\mathcal{A}[\sigma], c_t)$ the number of mistakes made by an off-line algorithm $\mathcal{A}$ on a sequence σ and a target c_t. Define

$$M(\mathcal{A}[\sigma]) \triangleq \max_{c_t \in \mathcal{C}} M(\mathcal{A}[\sigma], c_t).$$

We are interested in the *best* and *worst* sequences. We would like to view the $M_{\text{best}}(\mathcal{C})$ as letting the learner optimally choose (without knowing the target concept) the sequence of instances presented to it. That is, we define:

$$M_{\text{best}}(\mathcal{C}) \triangleq \min_{\mathcal{A}} \min_{\sigma} M(\mathcal{A}[\sigma]).$$

Similarly, the worst sequence model can be thought of as letting an adversary choose the sequence, so we may define:

$$M_{\text{worst}}(\mathcal{C}) \triangleq \min_{\mathcal{A}} \max_{\sigma} M(\mathcal{A}[\sigma]).$$

A *self-directed learning algorithm* $\mathcal{A}$ is one that chooses its sequence adaptively; hence, the sequence may depend on the classifications given to previous instances (and so, indirectly, on the target function). Denote by $M_{\text{sd}}(\mathcal{A}, c_t)$ the number of mistakes made by a self-directed learning algorithm $\mathcal{A}$ on a target function $c_t \in \mathcal{C}$. Define

$$M_{\text{sd}}(\mathcal{A}) \triangleq \max_{c_t \in \mathcal{C}} M_{\text{sd}}(\mathcal{A}, c_t)$$

and

$$M_{\text{sd}}(\mathcal{C}) \triangleq \min_{\mathcal{A}} M_{\text{sd}}(\mathcal{A}) = \min_{\mathcal{A}} \max_{c_t \in \mathcal{C}} M_{\text{sd}}(\mathcal{A}, c_t).$$

The *teacher-directed model* (Goldman, Rivest, & Schapire, 1993) is concerned with a different setting. In this model, a teacher presents labeled examples (i.e., pairs of the form $(x, c_t(x))$) to a student. The teaching is said to be successful once only a single concept in the concept class $\mathcal{C}$ is consistent with the labeled examples presented by the teacher.

For the teacher-directed model, let $\mathcal{A}$ denote the teacher's strategy for choosing examples. Denote by $M_{\text{td}}(\mathcal{A}, c_t)$ the number of examples $\mathcal{A}$ will present when attempting to teach the concept c_t. We define:

$$M_{\text{td}}(\mathcal{A}) \triangleq \max_{c_t \in \mathcal{C}} M_{\text{td}}(\mathcal{A}, c_t)$$

and

$$M_{\text{td}}(\mathcal{C}) \triangleq \min_{\mathcal{A}} M_{\text{td}}(\mathcal{A}) = \min_{\mathcal{A}} \max_{c_t \in \mathcal{C}} M_{\text{td}}(\mathcal{A}, c_t).$$

(Alternatively, for every $c \in \mathcal{C}$ let $n(c)$ be the minimal size of a subset of the domain on which no other concept in $\mathcal{C}$ agrees with c. It is easy to see that $M_{\mathrm{td}}(\mathcal{C}) = \max_{c\in\mathcal{C}} n(c)$).

Note that, unlike the previous models, in this model the teacher is considered to be 'helpful' rather than 'adversarial'. That is, the learning complexity is defined as the number of examples needed for learning using the teacher that *minimizes* the the sample size (although we still use the worst-case measure over all the possible targets and all the possible consistent learners).

The following is a simple consequence of the definitions:

Observation 1. For any finite $\mathcal{X}, \mathcal{C}$,

$$\begin{aligned} & M_{\mathrm{sd}}(\mathcal{C}) \\ \le\ & M_{\mathrm{best}}(\mathcal{C}) \\ \le\ & M_{\mathrm{worst}}(\mathcal{C}) \\ \le\ & M_{\text{on-line}}(\mathcal{C}). \end{aligned}$$

There are no such immediate relations between M_{sd} and M_{td}. On one hand, as the M_{td} parameter counts the number of needed *examples* needed to allow *any* consistent student to find the target, and the M_{sd} counts only *mistakes* made by the *optimal* student, there are classes for which $M_{\mathrm{sd}}(\mathcal{C}) < M_{\mathrm{td}}(\mathcal{C})$ (such as Monotone Monomials). On the other hand, as the agent that drives the learning process in the teacher-directed model knows what the target is, it may get an advantage over the self-directed learning process. In Section 4 below, we construct an example of a class $\mathcal{C}$ for which $M_{\mathrm{sd}}(\mathcal{C}) > M_{\mathrm{td}}(\mathcal{C})$.

2.2. *Labeled trees and their rank*

Each of the first four models presented above has a combinatorial characterization in terms of the *rank* of a certain family of trees. Littlestone (1988, 1989) investigates this characterization for the original on-line model, while Ben-David, Kushilevitz, & Mansour (1995) give similar results for the worst-sequence and best-sequence models.

In this subsection we review these characterizations and provide a similar characterization of the M_{sd} parameter. A variant of this characterization appears in Goldman & Sloan (1994).

We start with the definition of the *rank* of a binary tree (see, e.g., Cormen, Leiserson, & Rivest, 1990; Ehrenfeucht & Haussler, 1989; Blum, 1992), and later present similar characterization for the self-directed model.

For a binary tree T, if T is empty then $\operatorname{rank}(T) = -1$. Otherwise, let T_L be its left subtree and T_R be its right subtree. Then,

$$\operatorname{rank}(T) = \begin{cases} \max\{\operatorname{rank}(T_L), \operatorname{rank}(T_R)\} & \text{if } \operatorname{rank}(T_L) \neq \operatorname{rank}(T_R) \\ \operatorname{rank}(T_L) + 1 & \text{otherwise.} \end{cases}$$

For example, the rank of a leaf is 0.

Next, we define the relevant types of trees.

Definition 1. Let $\mathcal{X}$ denote some domain set and $\mathcal{C} \subseteq \{0, 1\}^{\mathcal{X}}$ be as above.

- An $\mathcal{X}$-labeled tree is a pair, (T, F), where T is a binary tree and F a function labeling the nodes of T by elements of $\mathcal{X}$. We require the labeling F to be unique on any path, i.e., on any path from the root of the tree to a leaf, no two nodes are labeled with the same point from $\mathcal{X}$.
- A branch $(t_1, \ldots, t_n)$ realizes a function

  $$h : \{F(t_1), \ldots, F(t_n)\} \mapsto \{0, 1\}$$

  if for all $1 \le i < n$, t_{i+1} is a left son of t_i if and only if $h(F(t_i)) = 1$. Note that the values of F on leaves of T do not affect the realization of h, and therefore, for the purposes of this work, may be ignored.
- An $\mathcal{X}$-labeled tree is a $\mathcal{C}$-tree if the set of functions it realizes is exactly $\mathcal{C}$.
- Let $\mathcal{T}_{\mathcal{X}}^{\mathcal{C}}$ denote the set of all $\mathcal{C}$-trees.
- For a sequence $\sigma = (x_1, \ldots, x_n)$ of elements of $\mathcal{X}$, let $T_\sigma^{\mathcal{C}}$ denote the labeled tree that is defined by starting with a complete binary tree, T', with $n + 1$ levels, setting $F(t) = x_k$ for every node t on the kth level of T', and restricting the tree to the branches that are consistent with $\mathcal{C}$.

Note, that using this notation, a class $\mathcal{C}$ shatters the set of elements of a sequence, σ, if and only if $T_\sigma^{\mathcal{C}}$ is a complete binary tree. We can therefore conclude that, for any class $\mathcal{C}$,

$$VCdim(\mathcal{C}) = \max\left\{\text{rank}\left(T_\sigma^{\mathcal{C}}\right) : T_\sigma^{\mathcal{C}} \in \mathcal{T}_{\mathcal{X}}^{\mathcal{C}} \text{ and } T_\sigma^{\mathcal{C}} \text{ is a complete binary tree}\right\}.$$

(We shall usually omit the superscript $\mathcal{C}$ when it is clear from the context.)

With the above notation it is now easy to state the combinatorial characterization of the mistake-bound models:

Theorem 1 (Littlestone, 1988, 1989). *For all $\mathcal{X}$ and $\mathcal{C}$ as above*,

$$M_{\text{on-line}}(\mathcal{C}) = \max\left\{\text{rank}(T) : T \in \mathcal{T}_{\mathcal{X}}^{\mathcal{C}}\right\}.$$

Theorem 2 (Ben-David, Kushilevitz, & Mansour, 1995). *For all $\mathcal{X}$ and $\mathcal{C}$ above*,

$$M_{\text{worst}}(\mathcal{C}) = \max\left\{\text{rank}(T) : T = T_\sigma^{\mathcal{C}}, \quad \sigma \textit{ is an ordering of } \mathcal{X}\right\}.$$

Theorem 3 (Ben-David, Kushilevitz, & Mansour, 1995). *For all $\mathcal{X}$ and $\mathcal{C}$ as above*,

$$M_{\text{best}}(\mathcal{C}) = \min\left\{\text{rank}(T) : T = T_\sigma^{\mathcal{C}}, \quad \sigma \textit{ is an ordering of } \mathcal{X}\right\}.$$

Finally, we get a characterization in similar terms of the $M_{\text{sd}}(\mathcal{C})$ parameter. A variant of this characterization appears in Goldman & Sloan (1994).

Theorem 4. *For all $\mathcal{X}$ and $\mathcal{C}$ as above*,

$$M_{\mathrm{sd}}(\mathcal{C}) = \min\left\{\mathrm{rank}(T) : T \in \mathcal{T}_{\mathcal{X}}^{\mathcal{C}}\right\}.$$

Proof: Consider the tree T whose rank is the minimal one in $\mathcal{T}_{\mathcal{X}}^{\mathcal{C}}$. We will show that $M_{\mathrm{sd}}(\mathcal{C})$ is at most the rank of T. For this, we present an appropriate algorithm that makes use of this tree. At each point, the learner asks for the instance which is the current node in the tree. In addition, he predicts according to the subtree of the current node whose rank is higher (arbitrarily, if the ranks of the two subtrees are equal). The true classification determines the child of the current node from which the learner needs to proceed. It follows from the definition of rank that whenever the algorithm makes a mistake the remaining subtree has rank which is strictly smaller than the previous one.

For the other direction, given a strategy for the learner that makes at most $M_{\mathrm{sd}}(\mathcal{C})$ mistakes we can construct a tree T that describes this strategy. Namely, at each point the instances that the learner will ask at the next stage given the possible classifications of the current instance determine the two children of the current node. Now, if the rank of T was more than $M_{\mathrm{sd}}(\mathcal{C})$ then this gives the adversary a strategy to fool the learner: at each node classify the current instance according to the subtree with higher rank. If the ranks of both subtrees are equal then on any answer by the algorithm the adversary says the opposite. By the definition of rank, this gives rank(T) mistakes. Hence, rank(T) is at most $M_{\mathrm{sd}}(\mathcal{C})$ and certainly the minimum over all trees can only be smaller. □

3. Self-directed complexity and the VC-dimension

We now turn our attention to the relationship between the self-directed complexity and the VC-dimension. We follow the following standard definition:

Definition 2. A concept class $\mathcal{C} \subseteq 2^{\mathcal{X}}$ is said to *shatter* a set $A \subseteq \mathcal{X}$ if the following holds:

$$\forall a \subseteq A\, \exists c \in \mathcal{C} \quad \text{s.t. } a = A \cap c.$$

The *VC-dimension* (or VCdim for short) of $\mathcal{C}$ is defined as the cardinality of the largest subset of $\mathcal{X}$ that $\mathcal{C}$ shatters, or ∞ if $\mathcal{C}$ shatters sets of unbounded cardinality.

Experience in other learnability models leads one to expect that the information complexity of learning a concept class is closely related to the combinatorial complexity of that class as measured by the VC-dimension. Goldman & Sloan (1994) showed that the VC-dimension of a concept class does not impose any lower bounds on its self-directed complexity. They presented concept classes of arbitrarily large VC-dimension that could be learned by a self-directed algorithm making only one mistake (in fact, even M_{best} of these classes is just 1). They asked whether an upper bound on the self-directed complexity can be derived from the VC-dimension of a class alone. We show that no such upper bound exists. Namely, there are concept classes of arbitrarily large self-directed complexity, yet their VC-dimension may be fixed at any $d \geq 3$.

3.1. Arbitrary concept classes

Definition 3. A concept class $\mathcal{C}$ over a domain $\mathcal{X}$ is said to contain a copy of a class $\mathcal{C}'$ over a domain $\mathcal{X}'$ if there exist functions $f : \mathcal{X}' \mapsto \mathcal{X}$ and $g : \mathcal{C}' \mapsto \mathcal{C}$ such that for all $x \in \mathcal{X}'$, $c \in \mathcal{C}'$, $c(x) = g(c)(f(x))$.

It is immediate to see that if a class $\mathcal{C}$ contains a copy of another class $\mathcal{C}'$ then $M_{\mathrm{sd}}(\mathcal{C}) \geq M_{\mathrm{sd}}(\mathcal{C}')$.

Lemma 1. *For every pair* (d, n) *of natural numbers such that* $d \geq 3$, *there exists a concept class* $\mathcal{C}_n^d$ *such that*:

1. $\mathcal{C}_n^d$ *is a class of subsets of* $\{1, \ldots, (3^n)d\}$.
2. $|\mathcal{C}_n^d| = 3^n 2^d$.
3. $VCdim(\mathcal{C}_n^d) = d$.
4. *For every point* x *in the domain of* $\mathcal{C}_n^d$, *and for every label of* x, *the set of concepts in* $\mathcal{C}_n^d$ *which are consistent with this labeling of* x *contains a copy of* $\mathcal{C}_{n-1}^d$.

Proof: Let us start by constructing the classes $\mathcal{C}_n^d$. We shall represent classes, $\mathcal{C}$, as matrices over $\{0, 1\}$ whose columns stand for the elements of $\mathcal{X}$ ($\mathcal{C}$'s domain) and rows stand for the concepts of $\mathcal{C}$. The entry (i, j) of this matrix is 1 iff the jth element of the domain is a member of the ith concept in $\mathcal{C}$.

Let A be any 3×3 matrix over binary matrices. We require A to have, in each row and each column, a matrix that is all 1's and a matrix that is all 0's. In the following example, such a matrix is shown, where the matrices used on the diagonal will be defined later.

?	1	0
0	?	1
1	0	?

Let $\mathcal{C}_0^d$ be a $2^d \times d$ matrix whose rows are all the $\{0, 1\}$-valued vectors of length d. For every $n > 0$, the matrix $\mathcal{C}_n^d$ is obtained by taking the matrix A over matrices that are the same size as $\mathcal{C}_{n-1}^d$ and using $\mathcal{C}_{n-1}^d$ for the diagonal of A. We later refer to $\mathcal{C}_n^d$ as either a matrix of 3×3 blocks (denoted $A(i, j)$) or as a $2^d 3^n \times d3^n$ binary matrix.

That is,

$$\begin{array}{|c|c|c|}\hline \mathcal{C}_{n-1}^d & \begin{matrix}1 & \cdots & 1\\ \vdots & \ddots & \vdots\\ 1 & \cdots & 1\end{matrix} & \begin{matrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{matrix} \\ \hline \begin{matrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{matrix} & \mathcal{C}_{n-1}^d & \begin{matrix}1 & \cdots & 1\\ \vdots & \ddots & \vdots\\ 1 & \cdots & 1\end{matrix} \\ \hline \begin{matrix}1 & \cdots & 1\\ \vdots & \ddots & \vdots\\ 1 & \cdots & 1\end{matrix} & \begin{matrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{matrix} & \mathcal{C}_{n-1}^d \\ \hline \end{array}$$

Let us show that property (4) holds. Assume that x belongs to column j_x of blocks in the matrix. By the definition of A, one of the blocks $A(i, j_x)$ has the same label as x in each entry (since A had a 1 and a 0 off the diagonal in every column). This implies that all concepts that fall within the ith row of blocks in the matrix are consistent with the labeling for x, but these include the block $A(i, i)$ which contains a copy of $\mathcal{C}^d_{n-1}$.

To prove property (3), define the *Robust VC-dimension* of a set $\mathcal{C}$ of subsets of $\mathcal{X}$, $rVCdim(\mathcal{C})$, as the VC-dimension of $\mathcal{C}'$ where $\mathcal{C}' = \mathcal{C} \cup \{\emptyset, \mathcal{X}\}$. Observe that, for every concept class $\mathcal{C}$, $rVCdim(\mathcal{C}) \geq VCdim(\mathcal{C})$, and $rVCdim(\mathcal{C}) = rVCdim(\mathcal{C}')$. We now show, by induction on n, that $rVCdim(\mathcal{C}^d_n) \leq d$.

$\mathcal{C}^d_0$ is defined over a domain of size d so it cannot have a robust VC-dimension that is greater than d. Now, examine a subset $\mathcal{Y}$ of size $d + 1 \geq 4$ of the domain of $\mathcal{C}^d_n$. By the induction hypothesis, if all the elements of $\mathcal{Y}$ fall within the same column of blocks in the matrix then $\mathcal{Y}$ is not shattered. Otherwise, $\mathcal{Y}$ contains elements from at least two different columns of blocks in the matrix. Since $|\mathcal{Y}| \geq 4$, at least two of these elements, y_1 and y_2, belong to the same column of blocks, j_y $(1 \leq j_y \leq 3)$. For every concept where the labeling of y_1 is different from that of y_2, the concept must come from the j_yth row of blocks, but that forces a single label for each element in $\mathcal{Y}$ that is not in the j_yth column of blocks, and so $\mathcal{Y}$ cannot be shattered. Furthermore, adding the rows of all 1's and all 0's will not make $\mathcal{Y}$ shattered, as the missing subsets of $\mathcal{Y}$ are those where the labeling of y_1 is different from that of y_2. Note that the last argument cannot be stated for the standard definition of the VC-dimension, as the row of all 1's and the row of all 0's was added. □

Corollary 5. *The self-directed complexity of a concept class is not bounded by any function of its VC-dimension. Namely, for every n and $d \geq 3$, there exist a concept class whose VC-dimension is d, yet its $M_{\rm sd}$ exceeds n.*

Proof: Apply Lemma 1. Note that property (4) of the lemma implies that, for all $n \geq 1$ and d, $M_{\rm sd}(\mathcal{C}^d_n) \geq M_{\rm sd}(\mathcal{C}^d_{n-1}) + 1$. It can be easily verified for all d, $M_{\rm sd}(\mathcal{C}^d_0) = d$, as for $M_{\rm sd}(\mathcal{C}^d_0)$ an adversary can force the learner to err on every point $x \in \mathcal{X}$. It follows that, for all n and d, $M_{\rm sd}(\mathcal{C}^d_n) \geq d + n$. The corollary is now established by recalling property (3) of the lemma, namely that for every n and $d \geq 3$, $VCdim(\mathcal{C}^d_n) = d$. □

3.2. Intersection-closed classes

We also consider Conjecture 11 of (Goldman & Sloan, 1994) which states that for *intersection-closed* concept classes, $\mathcal{C}$, $M_{\rm sd}(\mathcal{C}) \leq VCdim(\mathcal{C})$.[3] We provide a counterexample to this conjecture by showing the existence of concept classes $\mathcal{C}_d$ that are intersection-closed, yet, $VCdim(\mathcal{C}_d) = 2d$ and $M_{\rm sd}(\mathcal{C}_d) = 3d$.

Example 1. For every integer d, we show a concept class $\mathcal{C}_d$ so that $VCdim(\mathcal{C}_d) = 2d$, $\mathcal{C}_d$ is intersection-closed, and $M_{\rm sd}(\mathcal{C}_d) = 3d$.

We start with $d = 1$. Let $\mathcal{C}_1$ be the set of all segments on the circumference of a discrete circle (i.e., n many points equally spaced on the circumference of a two-dimensional circle, with $n \geq 6$), where the length of each segment is less than half of the circumference. It is clear that $\mathcal{C}_1$ is intersection-closed, and $VCdim(\mathcal{C}_1) = 2$.

Let us show an adversary strategy that will force any algorithm to make at least three mistakes on $\mathcal{C}_1$. The first time the algorithm makes a prediction, the adversary will declare a "mistake". We now show that the concepts that are consistent with that answer have a pair of points that they shatter. Moreover, the adversary can force the algorithm to err before having to determine the label of each of these points. If the learner predicted $(x, 0)$ (for some point x on the circle), then the two neighboring points of x (one on each side) are such a pair (since the target may be a segment of 1, 2 or 3 points). If the learner predicted $(x, 1)$, then the adversary (now committed to a 0 on x) marks to itself the point that is antipodal to x as being in the target. The same argument implies that the two neighbours of this marked point constitute a pair as required. (It is easy to see that $\mathcal{C}_1$ can indeed be learned with three mistakes.)

We construct $\mathcal{C}_d$ by using d disjoint copies of $\mathcal{C}_1$, i.e., d circles with a concept being a set that contains one segment on each circle. It is easy to see that $VCdim(\mathcal{C}_d) = 2d$ and that the above strategy, when duplicated d times, once for each circle, will force $3d$ mistakes.

4. Teacher-directed complexity

As stated in Rivest & Yin (1995), most "natural" concept classes are easier to learn in the self-directed model than in the teacher-directed model. This raises the question of whether the teacher's knowledge of the target function may be of any help to a "smart" self-directed learner. Rivest & Yin (1995) give an example where $M_{\text{td}}(\mathcal{C}) < M_{\text{sd}}(\mathcal{C})$ for the case where the self-directed learner is limited, in each step, to polynomial time in the size of each example (not in the size of the concept class). Their example relies on some cryptographic assumptions. We present here an example where teacher-directed learning will require less examples than the number of mistakes self-directed learner will make, without imposing any limitations on the self-directed learner.

Lemma 2. *For the classes $\mathcal{C}_n^d$ of Lemma* 1, $M_{\text{td}}(\mathcal{C}_n^d) = d + 1$.

Proof: We present a strategy for the teacher that will allow any consistent learner to infer the target functions after receiving no more than $d + 1$ examples. Let the teacher use the following algorithm:

- For a target c_t present the d coordinates on the diagonal of the matrix (the coordinates for which concepts in the same block as c_t may have any behavior).
- If the target concept labels all these d points identically (all 0's or all 1's), present one more example from another block of the same innermost matrix as the one the target is in (i.e., another block that is in the same copy of $\mathcal{C}_1^d$ as the d examples already given to the learner) that will differentiate it from the case where the first d examples were taken from a block that is all 1's or all 0's (for example, if the target is taken from the first third of a copy of $\mathcal{C}_1^d$, and for the first d coordinates its values are all 0's, the teacher will present an example that falls into the third block column of the same copy of $\mathcal{C}_1^d$ whose

value will be 0, as rows whose first third is all 0's and belong to the second block-row of $\mathcal{C}_1^d$ will have a 1 in that position).

It is immediate that there will be only one concept in $\mathcal{C}_n^d$ that is consistent with the first $d+1$ examples presented according to the above strategy. $\square$

Corollary 6. *The self-directed complexity of a concept class is not bounded by any function of its teacher-directed complexity. Namely, for every n and $d \geq 3$, there exist a concept class whose teacher-directed complexity is d, yet its M_{sd} exceeds n.*

5. Generic self-directed learning algorithms

In this section we address the existence of generic learning algorithms for a given family of learning problems. Do there exist efficient algorithms that, given a concept class as an input, output a 'good' learning algorithm for it?

The meaning of 'efficiency' is not immediately clear in the context of such algorithms. Usually, a learning algorithm is considered to be efficient if its running time is polynomial in the size of an example and in the size of the description of the target concept. That is, an algorithm for a finite class $\mathcal{C}$ over a finite domain $\mathcal{X}$ is efficient if its running time is polynomial in $(\log(|\mathcal{X}|), \log(|\mathcal{C}|))$. In the 'generic learning' task, the class $\mathcal{C}$ is also a part of the input, so we expect the running time of the algorithm to depend upon the size of a description of $\mathcal{C}$ as well. Clearly, the choice of representation for the input concept classes plays a crucial role. One simple option is to represent each input concept class as a binary matrix. More precisely, a concept class corresponds to a binary matrix in which the (i, j)th entry is 1 if and only if the jth element of $\mathcal{X}$ is labeled 1 by the ith concept in $\mathcal{C}$. Papadimitriou & Yannakakis (1993) provide evidence that, using this matrix representation, the problem of finding the VC-dimension of a class cannot be decided in polynomial time. Frances & Litman (1995) show that the VC-dimension problem is reducible to any generic mistake-bound-optimal On-Line learning algorithm. In other words, any generic On-Line learning algorithm, that makes the optimal number of mistakes on every concept class, runs in time that is, up to a polynomial factor, larger than that of some algorithm for the VC-dimension search problem (when the whole concept class, as a binary matrix, is considered a part of the algorithm's input). Regretfully, we do not have similar results concerning the complexity of generic self-directed learning algorithms. On the other hand, we cannot offer an efficient generic self-directed learning algorithm that will make the optimal number of mistakes on any concept class. We conjecture that such algorithms do not exist.

In this section we present an efficient generic algorithm for self-directed learning. This algorithm follows the basic ideas of the Halving algorithm (Littlestone, 1988, 1989). Our algorithm receives as input the concept class from which the target is chosen and then runs as a learning algorithm for this class. Our generic algorithm is efficient in the sense that its running time for a class $\mathcal{C}$ over some domain $\mathcal{X}$ (and any target $t \in \mathcal{C}$) is polynomial in $(|\mathcal{C}|, |\mathcal{X}|)$. While we show that for some classes it does achieve optimal learning complexity, we also show that there exist classes on which it does make more than the optimal possible number of mistakes.

Algorithm 1.

```
1. Initialization: Set $i \leftarrow 0$ and $\mathcal{C}_{\text{cons}} \leftarrow \mathcal{C}$.
2. Let
       $\mathcal{X}' \triangleq \left\{x \in \mathcal{X} \mid \mathcal{C}_{\text{cons}}^{(x=0)} \neq \emptyset \wedge \mathcal{C}_{\text{cons}}^{(x=1)} \neq \emptyset\right\}$
   Find $x_i \in \mathcal{X}'$ such that
       $\left|\left|\mathcal{C}_{\text{cons}}^{(x_i=0)}\right| - \left|\mathcal{C}_{\text{cons}}^{(x_i=1)}\right|\right| = \max_{x \in \mathcal{X}'}\left|\left|\mathcal{C}_{\text{cons}}^{(x=0)}\right| - \left|\mathcal{C}_{\text{cons}}^{(x=1)}\right|\right|$
   Set $p_i$ such that $\mathcal{C}_{\text{cons}}^{(x_i=p_i)} \geq \mathcal{C}_{\text{cons}}^{(x_i=1-p_i)}$.
3. Predict $p_i$ on $x_i$.
4. After being given $c_t(x_i)$, set $\mathcal{C}_{\text{cons}} \leftarrow \mathcal{C}_{\text{cons}}^{(x_i=c_t(x_i))}$ and set $i \leftarrow i+1$.
5. If $|\mathcal{C}_{\text{cons}}| = 1$ then $\mathcal{C}_{\text{cons}} = \{c_t\}$, and we are done. Otherwise, go to step 2.
```

This algorithm is based on the same principal as the Halving algorithm (Littlestone, 1988, 1989), that is, keeping the set of all concepts that are consistent with the examples received so far and predicting a value that most of the concepts are consistent with. The only difference is that this algorithm also chooses the next point on which to predict by choosing a point x_i for which the difference between the number of consistent concepts that have $x_i \in c$ and those who have $x_i \notin c$ is the greatest. This is a heuristic that is based on the hope that making a mistake on such a sample point will be very profitable to the learner, as it will shrink the set of consistent concepts by as much as possible in a single step.

It is easy to see that the running time of this algorithm is polynomial in the size of its input (the input is $\mathcal{C}$ and each step requires $O(|\mathcal{C}| \cdot |\mathcal{X}|)$ time).

While in the worst case this algorithm performs no better than the Halving algorithm, for many "natural" problems it gives optimal performance, even for classes where $M_{\text{best}}(\mathcal{C}) > M_{\text{sd}}(\mathcal{C})$. We present here two examples for which Algorithm 1 is optimal. These include the class of monotone monomials and the concept class presented in (Ben-David, Eiron, & Kushilevitz, 1995, Theorem 9), for which self-directed learning is better than best sequence learning.

Example 2 (*Monotone monomials*). It is known that $M_{\text{sd}}(\mathcal{C}_{\text{MM}}) = 1$ (Goldman & Sloan, 1994).

That result is achieved by predicting negative on all assignments to n variables, starting with the assignment that assigns false to every variable, moving to all assignments that assign true to a single variable, then to those that assign true to two variables, etc.

Claim 1. *Algorithm* 1 *will make one mistake on* $\mathcal{C}_{\text{MM}}$.

Proof: All there is to show is that at every stage there is a variable x_i such that $|\mathcal{C}_{\text{cons}}^{(x_i=1)}| = 1$. This will ensure that the algorithm will predict false on that x_i, and if mistaken, the target will be known to it.

Let x be an assignment that assigns "true" to a minimal number k, of variables, among all assignments that still satisfy some concepts in $\mathcal{C}_{\text{cons}}$. It is clear that such an assignment satisfies a single monotone monomial in $\mathcal{C}_{\text{cons}}$ (if there are two monotone monomials that x

satisfies, at least one of them contains less than k variables, as there is surely just a single concept with k variables that is satisfied by x. This contradicts the assumption that all assignments that assign true to less than k variables do not satisfy any concept in $\mathcal{C}_{\text{cons}}$). It follows that x satisfies the above condition, and Algorithm 1 will make just one mistake on $\mathcal{C}_{\text{MM}}$. $\square$

Note that for the class of monotone monomials, an optimal Best Sequence algorithm will also make a single mistake. This proof can be easily extended to the class of monomials, where the optimal mistake bound is 2.

The following example shows that Algorithm 1 exploits the power of self-directed learning, showing that it is sometimes optimal even for classes where $M_{\text{best}} > M_{\text{sd}}$.

Example 3 (*Power of Adaptiveness*). In the proof of (Ben-David, Eiron, & Kushilevitz, 1995, Theorem 9) a set of concept classes for which $M_{\text{sd}}(\mathcal{C}) = 2$ but $M_{\text{best}} = \Omega(\log n)$ is introduced.

These concept classes are defined as follows: The concept class $\mathcal{C}$ consists of $2 \cdot 2^d$ functions $\mathcal{C} = \{f_1, \ldots, f_{2^d}, g_1, \ldots, g_{2^d}\}$ defined over $\mathcal{X} = \{z, x_1, \ldots, x_{2^d}, y_1, \ldots, y_{2^d}\}$. Each function f_i is defined as follows: $f_i(z) = 0$; $f_i(x_i) = 1$; $f_i(x_j) = 0$ for all $j \neq i$ (i.e., there is a single x_i which is assigned 1 and hence can be observed as an indicator for the corresponding function f_i). The variables $y_1, \ldots, y_{2^d}$ are partitioned into d "blocks" each of size $2^d/d$, where variables in the same block are equivalent with respect to $f_1, \ldots, f_{2^d}$. Each of the 2^d functions f_i gets one of the 2^d possible behaviors on these d blocks. The functions $g_1, \ldots, g_{2^d}$ are defined similarly by switching the roles of x's and y's. More precisely, $g_i(z) = 1$; $g_i(y_i) = 1$; $g_i(y_j) = 0$ for all $j \neq i$ (i.e., this time y_i serves as an indicator for the corresponding function g_i). Again, the variables $x_1, \ldots, x_{2^d}$ are partitioned into d "blocks" of $2^d/d$ equivalent variables. Each of the 2^d functions g_i gets one of the 2^d possible behaviors on these d blocks.

Claim 2. *For this class. Algorithm* 1 *will make at most* $M_{\text{sd}}(\mathcal{C}) = 2$ *mistakes.*

Proof: At first, Algorithm 1 will pick any of the x_i's or the y_i's (but not z) and predict 0 for it (because exactly $1 + 2^{d-1}$ of the concepts include each of these points, and the majority does not). This will continue until the algorithm is first mistaken. Note that if the algorithm make a correct prediction on a point x_i, the part of the concept class that will remain consistent will have all the targets of the form f_j, except for f_i, and exactly half of the concepts of the form g_j. The algorithm will now continue to predict on points of the form x_j (not y_j) until being mistaken, as most of the concepts that were found to be inconsistent had an equal number of 1's and 0's for each of the x_j's (so the number of 1's was cut by almost a half), while having only a single 1 for any y_j.

Without loss of generality, let us assume that Algorithm 1 makes its first mistake on x_i. Now, the concepts that are still consistent with Algorithm 1's observations are: f_i and some of the g_j's. Note that for z, there is only a single consistent function that has $f_i(z) = 0$. This means that if Algorithm 1 will make a mistake before predicting on z, it will be its last mistake (as the point on which it will predict will have just one concept that assigns the "wrong" value to it, or Algorithm 1 would have tried z before). Once Algorithm 1 predicts on z, and is not mistaken (again, if it is mistaken, the only consistent function that will remain is f_i, and no more mistakes will be made), the only functions that remain consistent are of the form g_j. For all these functions there are now points y_j that have a single function that assigns 1 to them, so Algorithm 1 will predict on all these points before trying any others, and it will, finally, be mistaken on one of them, leaving, once again, a single consistent function.

All in all, Algorithm 1 will make no more than two mistakes: One on x_i and the other one, either on z or on one of the y_j's. This shows that Algorithm 1 actually exploits the power of adaptiveness. □

However, it can also be easily shown that Algorithm 1 is not always optimal.

Theorem 7. *For every integer $k \geq 1$ and $d > 3 + k$ there exists a concept class $\mathcal{C}_k^d$ such that $M_{\text{sd}}(\mathcal{C}_k^d) = d$ while Algorithm 1 can be forced to make $d + k$ mistakes on $\mathcal{C}_k^d$.*

Proof: Let us describe the concept classes $\mathcal{C}_k^d$. These concept classes are defined over an instance space $\mathcal{X}$ of $k + n \cdot 2^k$ points, where $n \triangleq 2^{\frac{d}{2}+1}$ and $\mathcal{X} = \{y_1, \ldots, y_k, x_1, \ldots, x_{n \cdot 2^k}\}$.

$\mathcal{C}_1^d$ is constructed as follows: The first 2^d concepts assign 0 to y_1, while the other $n^2 - n + 2$ concepts assign 1 to y_1. The x_i's are set to be duplicated blocks of all 2^d binary vectors of length d for the first 2^d concepts, and are set to the class of all segments on the circumference of a discrete n-circle for the rest of the concepts (there are $n^2 - n + 2$ segments on the circumference of a discrete n-circle).

It can be easily verified that the class of segments of a discrete circle can be learned by a self-directed algorithm with at most three mistakes. It is also clear that the class of all 2^d binary vectors of length d cannot be learned in less than d mistakes. This implies that an optimal self-directed algorithm for $\mathcal{C}_1^d$ will first predict 0 on y_1. If it is correct, it can be forced to make d more mistakes, and if it is wrong, it will make no more than three more mistakes, for a maximum total of d mistakes. Algorithm 1 on the other hand, will also guess first on y_1 (as all x's have an equal number of 0's and 1's), but it will predict 1 (as there are just $2^d = \frac{n^2}{4}$ concepts for which y_1 is 0, while there are $n^2 - n + 2$ concepts for which it is 1). This will force up to $d + 1$ mistakes on $\mathcal{C}_1^d$ as required.

Given $\mathcal{C}_k^d$ we construct $\mathcal{C}_{k+1}^d$ as follows:

- The first part of $\mathcal{C}_{k+1}^d$ will assign 0 to y_{k+1}, be the same as $\mathcal{C}_k^d$ for all other y's, and will have two exact duplicates of each x_i from $\mathcal{C}_k^d$ for its x's (there are twice as many x's now).
- The second part of $\mathcal{C}_{k+1}^d$ will assign 1 to y_{k+1}, assign 0's and 1's arbitrarily to the rest of the y's, making sure that on the total, each of $y_1, \ldots, y_k$ has an equal number of 0's and 1's (since this part is more than double in size than the first part, such an assignment can be found), and will assign to the x's all segments on a discrete $n \cdot 2^{k-1}$-circle.

It can be easily verified that under the assumptions for d and k, this construction leaves $M_{\text{sd}}(\mathcal{C}_{k+1}^d) = d$ but Algorithm 1 will first predict 1 on y_{k+1}, and if mistaken, will make one more mistake for the class $\mathcal{C}_{k+1}^d$ than it did for $\mathcal{C}_k^d$. □

Having demonstrated classes for which Algorithm 1 is not optimal, it is only natural to try to improve it. The first step that comes to mind is to enhance the algorithm's look-ahead capability. We thus go on and define a family of algorithms based on the same heuristics as Algorithm 1. Each of these algorithms is based on approximating the number of mistakes it will do for any remaining subset of the concept class by using the previous algorithm to learn the subset. More formally, we define the following sequence of algorithms:

Algorithm 2. *Let* $\mathcal{A}_1$ *be Algorithm* 1. *Define* $\mathcal{A}_{n+1}$ *as follows*:

1. *Initialization*: *Set* $i \leftarrow 0$ *and* $\mathcal{C}_{\text{cons}} \leftarrow \mathcal{C}$
2. *Find* $x_i \in \mathcal{X}$ *and* $p_i \in \{0, 1\}$ *such that*

$$M_{\text{sd}}\big(\mathcal{A}_n, \mathcal{C}_{\text{cons}}^{(x_i=1-p_i)}\big) = \min_{x\in\mathcal{X},\delta\in\{0,1\}} M_{\text{sd}}\big(\mathcal{A}_n, \mathcal{C}_{\text{cons}}^{(x=\delta)}\big)$$

   Predict p_i *on the above* x_i. *After being given* $c_t(x_i)$, *set* $\mathcal{C}_{\text{cons}} \leftarrow \mathcal{C}_{\text{cons}}^{(x_i=c_t(x_i))}$ *and set* $i \leftarrow i+1$
3. *If* $|\mathcal{C}_{\text{cons}}| = 1$ *then* $\mathcal{C}_{\text{cons}} = \{c_t\}$, *and we are done. Otherwise*, *go to step* 2.

Considering the classes $\mathcal{C}_k^d$ used in the proof of Theorem 7, it is immediate to realize that the algorithm $\mathcal{A}_{n+1}$ will be optimal for all the classes $\mathcal{C}_k^d$ for which $k \leq n$. It follows that the sequence of algorithms, $\{\mathcal{A}_n : n \in \mathrm{N}\}$ is of strictly increasing self-directed learning capabilities (in the sense that for every n, the algorithm $\mathcal{A}_{n+1}$ makes on every class at most the number of mistakes made on this class by $\mathcal{A}_n$, and on some classes, $\mathcal{A}_{n+1}$ makes strictly fewer mistakes than $\mathcal{A}_n$).

6. On the *query complexity* of self-directed learning

The information complexity of on-line learning tasks is commonly measured by either the number of student's queries or by the number of mistakes made by the student. The query complexity measure is usually used in models in which the learning communication is student-driven, while the mistake bound measure is more common in models with teacher-driven learning communication.

The self-directed learning model may be viewed as a mixture of these two types of models. On one hand, it is naturally presented in terms of a student-driven learning scenario; yet, on the other hand, it uses mistake counting for its definition of learning complexity.

A fully student-driven version of the self-directed learning model may be obtained by changing its complexity measure to the number of student queries (rather than mistakes). The resulting model may be viewed as a restricted version of the Maass-Turan *Partial Equivalence Query* (PEQ) model (Maass & Turan, 1992). In that model, the student presents hypotheses that are *partial* functions from $\mathcal{X}$ to $\{0, 1\}$. In response to its partial-function query, the student gets a counterexample from the query's domain (that is, a point $x \in \mathcal{X}$ on which the query function is defined and its value differs from that of the target function). By restricting the student to asking only hypotheses whose domains are singletons, one gets the self-directed learning scenario.[4]

Another way to view this model is as Membership Queries Only learning (MQ). In the MQ model the student queries the teacher by presenting a point of the domain set and gets, in response, the value of the target function on that point. The student is charged by the number of queries it makes. This is exactly the model one gets by the restriction to singletons of the PEQ model (and ignoring the value that the student's hypothesis assigns to the domain point). The main difference between the MQ (or the restricted PEQ) model and the self-directed model is in their definition of learning complexity. On one hand, by charging for queries, rather than mistake, the SD model becomes the restricted PEQ or the MQ model, while on the other hand, it is not hard to see that as far as the mistake bound measure is concerned, the (restricted or unrestricted) PEQ and the self-directed learning models are equivalent.

Therefore, it is natural to investigate the query complexity of self-directed learning, in addition to its mistake-bound complexity.[5] As a first step in this direction, we have the following trade-off formula between the number of queries and the number of mistakes of any self-directed learning algorithm. Yin (1995) addressed this issue independently (and at the same time with the conference version of our work (Ben-David, Eiron, & Kushilevitz, 1995)). She proves a similar trade-off formula and provides several examples of classes for which there exist learning algorithms that almost meet the query complexity bounds implied by the formula. Our proof below is much simpler than the one in Yin (1995).

Definition 4 (*Self-directed query complexity*). Given a self-directed learning problem $(\mathcal{C}, \mathcal{X})$ and an algorithm $\mathcal{A}$ for it, let $q_{\mathcal{A}}$ denote the maximum, over all possible targets $t \in \mathcal{C}$, of the number of queries made by the algorithm $\mathcal{A}$ during the learning protocol when applied to the target t.

Theorem 8. *For all $\mathcal{X}$, $\mathcal{C}$ and for every self-directed learning algorithm $\mathcal{A}$, let $m_{\mathcal{A}}$ denote $M_{\text{sd}}(\mathcal{A})$ then,*
1. $\log(|\mathcal{C}|) \leq m_{\mathcal{A}} \cdot \log(q_{\mathcal{A}})$.
2. $|\mathcal{C}| \leq \sum_{i=0}^{m_{\mathcal{A}}} \binom{q_{\mathcal{A}}}{i}$

Proof: Note that part 1 of the theorem follows from part 2 (use n^d as an upper bound to $\sum_{i=0}^{d} \binom{n}{i}$) and take the logarithm of both sides of the inequality). Part 2 of the theorem is implied by the following simple argument: Given a self-directed learning algorithm $\mathcal{A}$ for a class $\mathcal{C}$, each concept in $\mathcal{C}$ is uniquely determined by the set of queries on which the algorithm errs when the teacher's responses correspond to this concept. The theorem follows by noting that this set of queries is, in turn, determined by the set of their indices in the sequence of $\mathcal{A}$'s queries. □

7. Conclusions and open problems

In this paper we discussed some combinatorial problems related to the information complexity of self-directed learning. We have fully resolved the question of the relation between the self-directed complexity of a class and its VC-dimension by showing that none of these parameters imposes any bound on the other. We have also obtained a similar result concerning the relation between the self-directed and the teacher-directed complexity of a concept class. Concerning the family of intersection-closed classes, we have shown that there exist such classes for which the self-directed complexity can exceed the VC-dimension by a ratio of $3/2$. We do not know whether this result is the best possible, namely,

Open Problem 1. *Is it true that, for every intersection-closed class $\mathcal{C}$, $M_{\text{sd}}(\mathcal{C}) \leq 3/2$ $(VCdim(\mathcal{C}))$?*

Rather than addressing the computational complexity of the self-directed learning problem for specific classes, we discussed, in Section 5, the existence of a fixed 'generic' algorithm that will learn every concept class. Frances & Litman (1995) addressed the analogous problem for the On-Line learning model and showed that the problem of computing the VC-dimension of a class is reducible, via the problem of computing the On-Line mistake bound of a class, to the existence of a generic On-Line algorithm that is mistake-bound-optimal for all classes. This results is a strong indication to the computational hardness of generic mistake-bound-optimal On-Line learning algorithms. Regretfully, we do not have a similar result for self-directed learning. It is quite easy to see that, like the case for On-Line learning, self-directed generic optimal learning is poly-time equivalent to the task of calculating $M_{\text{sd}}(\mathcal{C})$ (on input $\mathcal{C}$ in its matrix representation). The main open problem along this line of research is whether a result like the Frances-Litman reduction holds for self-directed learning, namely,

Open Problem 2. *Does there exist an algorithm that, on input $\mathcal{C}$, calculates the VC-dimension of $\mathcal{C}$ in polynomial time (in the binary matrix representation of $\mathcal{C}$) using an oracle that provides the self-directed mistake bound of classes?*

Notes

1. In fact, the models defined in Ben-David, Kushilevitz, & Mansour (1995) allow restricting the instances to some subset S of the instance space. To simplify the presentation we omit this generalization from this paper.
2. Ben-David, Kushilevitz, & Mansour (1995) restrict the learner to a subset $S \subseteq \mathcal{X}$ of the instance space. For clarity, our definition follows the more standard definition of Littlestone (1988, 1989). However, it can easily be extended to include this additional ingredient.
3. A Class $\mathcal{C}$ is called *intersection-closed* if for every two concepts $c_1, c_2 \in \mathcal{C}$ also $c_1 \cap c_2 \in \mathcal{C}$.
4. Note that, without this restriction, the PEQ model is strictly stronger than the SD model. For an example, consider a concept class of n-many singletons. Clearly, both a PEQ and a self-directed learning algorithm can learn it with only one mistake. As for queries, PEQ can settle for only one query, while for any self-directed learning algorithm there is a concept in the class on which it will be forced to make $n - 1$ queries. When Goldman & Sloan (1994, Section 7) discuss the relation of the SD model to the PEQ model, they state the converse statement, namely, that the self directed learning model is more powerful than the PEQ model. One should note, however, that their claim is only due to the bias inflicted by using a different complexity measure for each of these models (they compare the number of *mistakes* made by an SD learner to the number of *queries* made by a PEQ learner).
5. Goldman & Sloan (1994) state, in the definition of the self-directed model, that the student should eventually ask every single member of the domain. For the purposes of our discussion, we adopt the approach of Maass & Turan (1992) and view the learning process as complete once only one member of the concept class remains consistent with the data collected so far.

References

Angluin, D. (1988). Queries and concept learning. *Machine Learning*, *2*, 319–342.

Blum, A. (1990). Separating distribution-free and mistake-bound learning models over the Boolean domain. *Proceedings of FOCS90* (pp. 211–218).

Blum, A. (1992). Learning Boolean functions in an infinite attribute space. *Machine Learning*, *9*. Also, *Proceedings of STOC90* (pp. 64–72), 1990.

Blum, A. (1992). Rank-r decision trees are a subclass of r-decision lists. *Information Processing Letters*, *42*, 183–185.

Ben-David, S., Eiron, N., & Kushilevitz, E. (1995). On self-directed learning. *Proceedings of COLT95* (pp. 136–143).

Ben-David, S., Kushilevitz, E., & Mansour, Y. (1995). Online learning versus offline learning. *Second European Conference on Computational Learning Theory (Euro COLT)*. In P. Vitanyi (Ed.), Springer-Verlag, Lecture Notes in Artificial Intelligence (Vol. 904, pp. 38–52).

Cormen, T.H., Leiserson, C.E., & Rivest, R.L. (1990). *Algorithms*. MIT Press.

Chen, Z., & Maass, W. (1994). On-line learning of rectangles and unions of rectangles. *Machine Learning*, *17*(2/3), 23–50.

Ehrenfeucht, A., & Haussler, D. (1989). Learning decision trees from random examples. *Information and Computation*, *82*, 231–246.

Frances, M., & Litman, A. (1995). *Optimal mistake bound learning is hard* (Technical Report No. 855). Department of Computer Science, The Technion, Israel Institute of Technology.

Goldman, S. (1990). *Learning binary relations, total orders, and read-once formulas.* Ph.D. thesis, MIT Department of Electrical Engineering and Computer Science.

Goldman, S., & Kearns, M. (1995). On the complexity of teaching. *J. of Comput. Syst. Sci.*, *50*(1), 20–31.

Goldman, S.A., Rivest, R.L., & Schapire, R.E. (1993). Learning binary relations and total orders. *SIAM J. of Computing*, *22*(5), 1006–1034.

Goldman, S.A., & Sloan, R.H. (1994). The power of self-directed learning. *Machine Learning*, *14*, 271–294.

Helmbold, D.P., Littlestone, N., & Long, P.M. (1992). Apple tasting and nearly one-sided learning. *Proceedings of FOCS92* (493–502).

Littlestone, N. (1988). Learning when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, *2*, 285–318.

Littlestone, N. (1989). *Mistake bounds and logarithmic linear-threshold learning algorithms*. Ph.D. thesis, U.C. Santa Cruz.

Littlestone, N., & Warmuth, M.K. (1994). The weighted majority algorithm. *Information and Computation*, *108*(2), 212–261.

Maass, W. (1991). On-line learning with an oblivious environment and the power of randomization. *Proceedings of COLT91* (pp. 167–175).

Maass, W., & Turan, G. (1992). Lower bound methods and separation results for on-line learning models. *Machine Learning*, *9*, 107–145.

Papadimitriou, C.H., & Yannakakis, M. (1993). On limited nondeterminism and the complexity of the VC dimension. *Proceedings of the Eighth Annual Structure in Complexity Theory Conference* (pp. 12–18).

Rivest, R.L., & Yin, Y.L. (1995). Being taught can be faster than asking questions. *Proceedings of COLT95* (pp. 144–151).

Yin, Y.L. (1995). Reducing the number of queries in self-directed learning. *Proceedings of COLT95* (pp. 128–135).

Received
Accepted
Final manuscript