Massive Online Teaching to Bounded Learners

Brendan Juba* Harvard and MIT

Ryan Williams† Stanford

August 30, 2012

Abstract

We consider a model of teaching in which the learners are consistent and have bounded state, but are otherwise arbitrary. The teacher is non-interactive and "massively open": the teacher broadcasts a sequence of examples of an arbitrary target concept, intended for every possible on-line learning algorithm to learn from. We focus on the problem of designing interesting teachers: sequences of examples that allow all capable and consistent learners to *efficiently* learn concepts, regardless of the underlying algorithm used by the learner. We use two measures of efficiency: the number of mistakes made by the worst-case learner, and the maximum length of the example sequence needed for the worst-case learner. Our results are summarized as follows:

- Given a uniform random sequence of examples of an n-bit concept function, learners (capable of consistently learning the concept) with $s(n)$ bits of state are guaranteed to make only $O(n \cdot s(n))$ mistakes and exactly learn the concept, with high probability. This theorem has interesting corollaries; for instance, every concept c has a sequence of examples can teach c to all capable consistent on-line learners implementable with $s(n)$-size circuits, such that every learner makes only $\tilde{O}(s(n)^2)$ mistakes. That is, all resource-bounded algorithms capable of consistently learning a concept can be simultaneously taught that concept with few mistakes, on a single example sequence.

  We also show how to efficiently generate such a sequence of examples on-line: using Nisan's pseudorandom generator, each example in the sequence can be generated with polynomial-time overhead per example, with an $O(n \cdot s(n))$-bit initial seed.
- To justify our use of randomness, we prove that any non-trivial derandomization of our sequences would imply circuit lower bounds. For instance, if there is a deterministic $2^{n^{O(1)}}$ time algorithm that generates a sequence of examples, such that all consistent and capable polynomial-size circuit learners learn the all-zeroes concept *with less than* 2^n *mistakes*, then $\mathsf{EXP} \not\subset \mathsf{P/poly}$.
- We present examples illustrating that the key differences in our model – our focus on mistakes rather than the total number of examples, and our use of a state bound – must be considered together to obtain our results.
- We show that for every consistent $s(n)$-state bounded learner $\mathcal{A}$, and every n-bit concept that $\mathcal{A}$ is capable of learning, there is a custom "tutoring" sequence of only $O(n \cdot s(n))$ examples that teaches $\mathcal{A}$ the concept. That is, in principle, there are *no slow learners*, only *bad teachers*: if a state-bounded learner is capable of learning a concept at all, then it can always be taught that concept quickly via some short sequence of examples.

*Supported by ONR grant number N000141210358 and NSF Grant CCF-0939370.

†Supported in part by a David Morgenthaler II Faculty Fellowship, and NSF Grant CCF-1212372 (Exploiting Duality between Algorithms and Complexity).

1 Introduction

In 1960, Hans Freudenthal [10] proposed a non-interactive message called LINCOS, which attempts to teach the "whole bulk of our knowledge" in a way that can be easily understood by any intelligent being. Using regularities in the message itself, the message presents a long sequence of examples for each concept, starting with basic concepts such as equality (of natural numbers), gradually building up a vocabulary of arithmetic, and then proceeding onwards to more complex concepts using the existing vocabulary. Freudenthal's intent was to broadcast this message into outer space, to teach alien civilizations about humanity.

Inspired by this remarkable (but perhaps a bit zany) work, we ask a more grounded question: can concepts be taught to *every* capable and bounded computational device in a non-interactive way? This sort of question is "dual" to the usual type of problem that arises so often in TCS, where we wish to design an algorithm that is "useful" on a large class of data. Here we have a problem of *data design*: presenting an arrangement of data that is useful to a large class of algorithms. We wish to design, once and for all, a sequence of inputs and outputs so that no matter *who* or *what* observes the behavior (including future observers we haven't conceived of), the observer can efficiently discover the structure of the underlying process. Is this possible, even in principle? What are the constraints involved?

To properly formalize these questions, we consider a new model of teaching in which the learners are of limited computational complexity (and consistent), but otherwise arbitrary. A vast array of teaching models already exist in the literature (cf. Section 1.2); most of these works have either considered models in which either the learner is assumed to be more sophisticated, or the teacher and learner are designed as a pair. The closest relative to our study is the well-known work on teaching dimension by Goldman and Kearns [11] and Shinohara and Miyano [26], which studies the number of examples needed to identify each concept in a given class. The model we consider retains a complexity-theoretic "universal" flavor along the lines of Goldreich, Juba, and Sudan [13]: we wish to design sequences of examples for efficiently teaching an *unknown* but capable bounded-state learner. Our work can be seen as a complexity-theoretic extension of the work on teaching dimension, focused on efficient learners. (See Section 2 for more comparison.)

1.1 Our Results

We first observe that the number of examples required to teach some very simple concepts for very simple learners can be maximally large. Let AC^0-LINEAR be the class of on-line learning algorithms whose functionality is computable by AC^0 circuits of size $O(n)$; more precisely, the learner's prediction and hypothesis update functions are computable with AC^0 circuits of $O(n)$ size (as a consequence, the learner stores its hypotheses with at most $O(n)$ bits of state).

Theorem 1.1 (Teaching weak learners requires many examples) *There is a concept class $\mathcal{H}$ such that for all n, there is a concept $h_n \in \mathcal{H}$ such that, for every sequence s_n of n-bit examples that allow all AC^0-LINEAR learners to identify c, the length of s_n equals 2^n.*

Hence we cannot measure learning efficiency by the number of examples, without further constraints. Furthermore, we observe that existing results on PAC-learning readily imply methods for *approximately* teaching concepts to many bounded learners. However, if we instead focus on exact learning, and consider the number of *mistakes* made by learners, we enter interesting territory. We find that there are simple, efficient (per round, on-line) universal teaching strategies for learners with bounded state. Informally, we say that a learning algorithm is $s(n)$*-bounded* if, in learning any n-bit concept function, the maximum length of any of its state descriptions is $s(n)$. (Full definitions will come later.)

Theorem 1.2 (Exact learning with few mistakes from random examples) *For all concepts $h : \{0,1\}^n \to Y$ and $\delta > 0$, every consistent $s(n)$-bounded $\mathcal{A}$ for h learns h and makes at most $O(s(n)(n + \log \frac{1}{\delta}))$ mistakes with probability at least $1 - \delta$, when the instances of length n are chosen uniformly at random.*

Theorem 1.2 has interesting corollaries; for example, any concept that is consistently learnable by some subset of the $s(n)$-size circuits can be learned all such circuits simultaneously, with $\tilde{O}(s(n)^2)$ mistakes by any learner, just by broadcasting uniform random examples (Corollary 3.1). Our use of the probabilistic method begs the question of whether such a sequence can be generated efficiently, with little to no randomness. As a partial answer, we confirm that Nisan's pseudorandom generator [21] for space-bounded computation can be used to obtain an efficient teaching strategy that uses few bits of randomness.

Theorem 1.3 (Few mistakes from low randomness) *Using a block length of $\Theta(s(n) + n + \log \frac{1}{\delta})$ (and $k = O(n + \log \frac{1}{\delta})$), Nisan's pseudorandom generator produces a sequence of $2^{O(n)}$ random bits for which with probability $1 - \delta$ over the seed $h_1, \ldots, h_k, x$, any consistent learning algorithm that is $s(n)$-bounded on a given concept exactly identifies that concept and makes at most $O(s(n)(n + \log \frac{1}{\delta}))$ mistakes.*

So using $O(s(n) \cdot n)$ random bits, we can generate $2^{O(n)}$ examples such that any learning algorithm consistent on h can be taught h with at most $O(s(n) \cdot n)$ mistakes on the examples, with high probability. This still does not answer the question of whether we can eliminate randomness entirely. We show that a *deterministic* exponential time algorithm for generating a sequence with similar properties would imply strong circuit lower bounds for EXP.

Theorem 1.4 (Deterministic sequences teaching all learners implies circuit lower bounds) *Let $\mathcal{F}$ be a class of functions from $\mathbb{N}$ to $\mathbb{N}$. Suppose there is a deterministic $t(n)$ time algorithm M such that for all $s(n) \in \mathcal{F}$ and all sufficiently large n, $M(1^n)$ prints a sequence S of examples of the empty concept $h : \{0,1\}^n \to \{0\}$ such that every consistent $s(n)$-size circuit learner $\mathcal{A}$ for h learns h and makes less than 2^n mistakes on the sequence S. Then there are problems solvable in $t(n)$ time that do not have circuits of size $s(n)$, for all $s(n) \in \mathcal{F}$ and almost every n.*

To illustrate, let $\mathcal{F}$ contain all functions of the form $f(n) = 2^{\delta n}$ and let $t(n) = 2^{O(n)}$. Suppose there is an $\varepsilon > 0$ and a $2^{O(n)}$ time algorithm which (for all n) prints a sequence such that all consistent learners implementable with $2^{\varepsilon n}$-size circuits learn the empty concept with less than 2^n mistakes. Then Theorem 1.4 implies E lacks $2^{\varepsilon n}$ size circuits (almost everywhere), which by Impagliazzo and Wigderson [15] implies $\mathsf{P} = \mathsf{BPP}$. (In contrast, Theorem 1.2 says that a uniform random sequence would teach all such learners with less than $2^{3\varepsilon n}$ mistakes.) For another example, let $\mathcal{F}$ contain all polynomials and let $t(n) = 2^{n^k}$ for a fixed k. Then any 2^{n^k} time algorithm which prints a sequence teaching all polynomial-size circuit learners the empty concept with less than 2^n mistakes implies $\mathsf{EXP} \not\subset \mathsf{P/poly}$. Hence our teaching model provides another case where non-trivial derandomization of a simple random process implies circuit lower bounds [14, 17]. Theorem 1.4 also clarifies why we consider our "data design" problem to be dual to algorithm design: positive solutions to the data design problem entail negative solutions to some circuit design problems.

Another natural question is whether we could remove the dependence on the state size $s(n)$ in the above bounds. We prove that this dependency is inherent:

Theorem 1.5 (Learners with large space can make many mistakes) *For every integer $s \in [n, 2^n]$, there is a concept class $\mathcal{H}$, a concept $h \in \mathcal{H}$ and a consistent on-line learning algorithm $\mathcal{A}$ for $\mathcal{H}$, computed by a uniform family of AC^0 circuits of size $O(s \cdot n)$ using s-bit states, such that for every sequence of examples of c, $\mathcal{A}$ makes at least $s - 1$ mistakes before identifying c.*

Finally, we consider a slightly relaxed objective. Suppose we have chosen both a concept h and a particular algorithm $\mathcal{A}$ capable of learning the concept. Can $\mathcal{A}$ be taught h quickly? A variant of our analysis in Theorem 3.1 establishes that there is always a short sequence of examples that forces any given state-bounded learner to identify any concept it is capable of learning:

Theorem 1.6 (Short sequences of examples for exact identification) *For every concept class $\mathcal{H}$, every $h \in \mathcal{H}$, and every consistent $s(n)$-bounded learner $\mathcal{A}$ for $\mathcal{H}$, there is a sequence of examples of length $O(n \cdot s(n))$ after which $\mathcal{A}$ identifies h.*

1.2 Existing models of teaching

Although we will try to review the main threads of research here, we advise the interested reader to consult the recent survey of Balbach and Zeugmann [6]. The oldest model of teaching, described by Freivalds et al. [22] is a variant of inductive inference (featuring little to no emphasis on computational complexity) in which the learner is given a carefully selected set of "good instances." Balbach and Zeugmann [4] considered a different sort of inductive-inference model in which a neighborhood structure is imposed on the space of concepts, and they study the structural properties of such learning—for example, they consider a variant in which the teacher may view the learner's hypothesis.

The highly influential notion of teaching dimension was independently introduced and studied by Shinohara and Miyano [26] and by Goldman and Kearns [11]. While we will discuss the relationship of our work to teaching dimension further in Section 2, a brief summary is warranted here, as it motivated most subsequent work. The underlying goal is to select minimum sets of examples so that a target concept can be uniquely identified out of the class of possible concepts; the maximum size of such a set (over all concepts in the class) is called the teaching dimension of that concept class. (The definition may also be cast in terms of consistent learning, a point we return to in Section 2.) Although the definition is highly natural and has spawned much interesting work, its consequences are unfortunately counterintuitive. For instance, the set $\mathcal{CONJ}$ of non-contradictory conjunctions over n variables has polynomial-size teaching dimension, but when the empty (always false) concept is included in $\mathcal{CONJ}$, the teaching dimension jumps to 2^n, i.e., *every* example must be presented to the learner.

In follow-up work, Anthony et al. [2] consider (among other results) the *average* teaching dimension (under the uniform measure over concepts), where the above undesirable effect disappears. In a similar vein, Balbach and Zeugmann [5] consider an "average" learner modeled by a MDP that chooses its concept at random from the space of consistent concepts; they also restrict the number of states in the MDP.

A different direction concerns models in which the teacher and learner are designed together. The main obstacle in such a model is that somehow the teacher must "honestly" teach the class to the learner, and not "collude" by coordinating an encoding of the concepts, and thus directly sending an encoding of the target concept, e.g., by either requiring that the learner still operate correctly with an adversarial teacher (as in Jackson and Tomkins [16]), by requiring that the learner still identify the concept correctly when (e.g.) additional examples are inserted into the teaching set (as in Goldman and Mathias [12] and Mathias [20]), or by giving the teacher and learner different representations of the concept (as in Angluin and Kri kis [1]).

Another recent direction assumes a powerfully helpful teacher which provides a *minimal* set of examples necessary for learning a concept, so the learner can eliminate competing concepts based on the size of the teaching set [3, 28]. Naturally, the strange behavior of the learner on the contradictory concept also vanishes in this model (and in fact, this concept has a teaching set of size $O(1)$), but only under the troublesome assumption that the learner "knows" the optimal size of teaching sets—troublesome because Servedio [25]

showed that the optimal size of teaching sets (in the traditional sense) is NP-hard to compute, so it is doubtful that these more sophisticated notions of teaching dimension could be computed easily.

1.3 Work on compression in learning and other structural learning themes

Our main results can be seen as a variant on the theme that learning algorithms which compress the data well must necessarily learn efficiently. At the most basic level, we note in Section 3 that a standard hypothesis-size analysis in PAC-learning establishes the existence of short teaching sequences for approximate concept identification. A more general result for the PAC model, starting from an arbitrary compression scheme, was the content of the well-known work by Blumer et al. [7] on Occam's razor. The difference here is that we start with a stronger algorithm – a consistent on-line learning algorithm rather than a consistent batch algorithm (or mere "compression scheme") – but then show that the given algorithm learns exceptionally well, obtaining *exact* rather than approximate identification in a mistake-bounded model. The "moral" that compression implies generalization is certainly a common thread across these results. Relationships between compression schemes and learnability were also considered by Littlestone and Warmuth [19], and further developed by Floyd and Warmuth [9], but there the learner was restricted to remember a list of examples that "compressed" the given sample; Floyd [8] also studied on-line algorithms in a similar vein. Our learners are not restricted to store information in any particular form.

Our analysis, which considers the performance of an on-line learning algorithm on i.i.d. examples from an arbitrary distribution, is thus also vaguely related to constructions that convert on-line learning algorithms to batch PAC-learning algorithms (cf. [18, 24]). Here, we analyze an on-line model and obtain mistake bounds for exact identification from random examples, rather than extracting an approximately good hypothesis in the batch model from an (on-line) algorithm.

2 Preliminaries and definitions

We first recall some standard terminology for on-line learning algorithms.

Definition 2.1 (On-line learning algorithms) *An* on-line learning algorithm *is specified by a pair of algorithms,* EVAL *and* UPDATE*, and an initial* state $\sigma_0 \in \{0,1\}^*$ *which may depend on the instance length* n*. Learning of a* target concept $h : \{0,1\}^n \to Y$ *from a* concept class $\mathcal{H}$ *proceeds in a sequence of* trials*: given the current* state *of the algorithm* $\sigma \in \{0,1\}^*$ *and the current* instance $x \in \{0,1\}^*$*,* $\texttt{EVAL}(\sigma, x)$ *produces a* prediction $p \in Y$ *(typically Boolean). The subsequent state of the algorithm is output by* $\texttt{UPDATE}(\sigma, x, p, h(x))$ *for a* reinforcement *value* $h(x)$*. If* $p \neq h(x)$*, then we say that the algorithm has made a* mistake*. The* mistake bound *of an algorithm for a concept class* $\mathcal{H}$ *and fixed size* n *is given by the maximum number of mistakes over all* $h \in \mathcal{H}$ *and (infinite) sequences of instances of length* n*.*

Let $\mathcal{C}$ *be a class of algorithms. We say that an on-line learning algorithm is* $\mathcal{C}$-bounded *if the algorithms* EVAL *and* UPDATE *are given by algorithms from* $\mathcal{C}$*. If* $\mathcal{C}$ *is a class of non-uniform circuits, we require that for each* $n \in \mathbb{N}$ *that the algorithms* EVAL *and* UPDATE *on examples of length* n *are given by circuits from* $\mathcal{C}$ *having* n*-bit inputs.*

We say that a learning algorithm is consistent on $\mathcal{H}$ *if, for every* n *and every* $h \in \mathcal{H}$*, after every sequence of examples* $x_1, \ldots, x_t \in \{0,1\}^n$ *of* h*, the algorithm is in a state* σ *such that* $\texttt{EVAL}(\sigma, x_i) = h(x_i)$ *for all* i*.*

We say that the algorithm has identified *or* learned h *if the algorithm is in a state* σ *such that for every subsequent sequence of examples* $x_1, \ldots, x_t \in \{0,1\}^n$ *the algorithm correctly predicts* $h(x_i)$ *for all* i*.*

We say that the algorithm is a learner for h *if there exists a sequence of examples after which the algorithm has identified* h*.*

The most relevant learning concept to our present work is that of teaching dimension.

Definition 2.2 *Let $\mathcal{H}$ be a concept class. A* teaching sequence *for an $h \in \mathcal{H}$ is a list of examples such that all consistent learners for $\mathcal{H}$ identify h after receiving those examples. The* minimum teaching length *for an $h \in \mathcal{H}$ is the minimum number n of examples in any teaching sequence for h. The* teaching dimension *of $\mathcal{H}$ is the maximum over all $h \in \mathcal{H}$ of the minimum teaching length for h.*

Therefore, the teaching dimension specifies the minimum number of examples needed for any consistent learner to distinguish any concept in $\mathcal{H}$ from all other concepts. This concept is quite similar to the notion of teaching that we consider; the two major differences are that we only consider learners with bounded state complexity, and we will focus on the number of mistakes rather than the number of examples in order to achieve meaningful generic bounds. The tutoring sequences considered in Section 4 are different from the teaching sequences in the above definition, in that they are tailored to a specific learner and they take into account the learner's complexity.

As noted earlier, the teaching dimension model [11, 26] has some counterintuitive properties, namely that some very simple concepts are maximally hard to teach, i.e., it requires specifying the entire domain. An example we will use is:

Definition 2.3 (Singletons and empty concept) *For each $y \in \{0,1\}^n$, the* singleton *concept for y is given by the function that evaluates to 1 at y and 0 everywhere else; the class of singletons is thus the set of all singleton concepts. The* empty *concept is the constant all-0 function.*

We define $\mathcal{S}$ to be the concept class consisting of all singleton concepts and the empty concept, for all n.

3 Teaching consistent learning algorithms of bounded complexity

Can we find sequences of examples that force every simple consistent learning algorithm to identify the target hypothesis? For *approximate* identification (i.e., standard PAC guarantees) this is immediate: the standard counting analysis shows that for learners that find consistent hypotheses of size at most $s(n)$, presenting $O(\frac{1}{\epsilon}(s(n) + \log \frac{1}{\delta}))$ random examples suffices to guarantee that any such learner arrives at a $1-\epsilon$-accurate hypothesis with probability $1-\delta$. Our hope is that the introduction of a "simple" requirement on the learners might reduce the complexity of (exact) teaching. In one sense, we can show that this *cannot* be true. The concept class $\mathcal{S}$ from the previous section, which is hard to learn in the usual (Goldman-Kearns-Shinohara-Miyano) teaching model, remains hard under severe complexity restrictions on learners. Recall from the Introduction that AC^0-LINEAR is the class of on-line learning algorithms whose functionality is computable by AC^0 circuits of size $O(n)$.

Reminder of Theorem 1.1 *For all n, let $h_n : \{0,1\}^n \to \{0,1\}$ be the all-zeroes concept (i.e., the empty concept). For every sequence s_n of n-bit examples that allow all AC^0-LINEAR learners to identify h_n, the length of s_n equals 2^n.*

Note that the length of the sequence cannot exceed 2^n: once all examples have been seen, the concept is surely identified. Hence we only have to prove a lower bound of at least 2^n.

Proof. Consider the following class of learners $\mathcal{L}$: for each $z \in \{0,1\}^n$, the initial hypothesis for $L_z \in \mathcal{L}$ is the singleton concept for z, which is used for prediction until either L_z makes a mistake on z, or L_z makes a mistake on some other y. In the former case, L_z switches to the empty concept; in the latter case,

L_z switches to the singleton concept for y. Finally, if the current hypothesis is the empty concept, and L_z makes a mistake on some y, then L_z switches to the singleton for y.

Observe that for all $z \in \{0,1\}^n$, the learner L_z is consistent for the concept class $\mathcal{S}$, and every L_z can be implemented with linear size depth-2 circuits which output states of length at most $n+1$. The first n bits of the state are used to represent a singleton concept, and the last bit of the state is 1 if and only if the current hypothesis is the empty concept. The algorithm `UPDATE` for L_z works as follows: if the label of the current example is 1, then the first n bits of state are switched to the current example and the last bit of state is set to 0. If the label is 0 and the current example equals z, then the last bit of state is set to 1. This behavior can be easily implemented with an OR of ANDs of $O(n)$ size. The algorithm `EVAL` just tests if the input equals the first n bits of state (and that the last bit of state is 0), which can be done with a linear size depth-2 circuit.

Note that for any sequence s_n of length less than 2^n, there must be some $z \in \{0,1\}^n$ that does not appear in the sequence. Then the learner L_z, after reading the sequence s_n, still has not identified the empty concept (it will make a mistake on z). □

Rather than being discouraged, let us consider other measures of the complexity of teaching. Notice that

1. The learners L_z constructed above only make at most two mistakes.
2. For every learner L_z, there is a one-example sequence after which L_z identifies the target concept.

In the following sections, we show that some version of these nice properties hold of low-complexity learners *in general* for state-bounded learners, and for arbitrary concepts that they can learn.

3.1 Sequences guaranteeing few mistakes for state-bounded learners

Another measure of the quality of a teaching strategy is the number of mistakes that the learners make. Here, we show how to construct strategies for which *every* consistent learner with bounded state makes at most a polynomial number of mistakes in its space bound and the size of the examples.

Definition 3.1 *Let $\mathcal{A} = (\texttt{EVAL}, \texttt{UPDATE})$ be an on-line learning algorithm, and let $s : \mathbb{N} \to \mathbb{N}$. $\mathcal{A}$ is* $s(n)$-bounded *if, on all length-n instances, the length of the initial state of $\mathcal{A}$ is at most $s(n)$, and all outputs of* `UPDATE` *(on all length-n instances and all states of length at most $s(n)$) have length at most $s(n)$.*

That is, an $s(n)$-bounded learner always stores its hypothesis and its general summary of the n-bit examples it has seen, using at most $s(n)$ bits.

The core of our analysis is the following theorem, giving a mistake bound for consistent on-line learning algorithms when presented with i.i.d. examples (as opposed to adversarial). A key point is that the quality of the final hypothesis improves *exponentially* with the number of mistakes—this will ensure that we can get exact identification of a concept in a polynomial number of mistakes.

Theorem 3.1 (Random examples teach consistent bounded learners with few mistakes) *For every concept $h : \{0,1\}^n \to Y$, $\varepsilon > 0$, $\delta > 0$, distribution D over examples of length n, and consistent $s(n)$-bounded learner $\mathcal{A}$ for h, the following holds. Given random labeled examples drawn from D, after $\mathcal{A}$ makes at most $O(s(n)(\log \frac{1}{\varepsilon} + \log \frac{1}{\delta}))$ mistakes on the examples, the hypothesis of $\mathcal{A}$ agrees with h on all but an ε-fraction of D, with probability at least $1 - \delta$.*

To prove the theorem, we need a couple of useful definitions.

Definition 3.2 *Given an $s(n)$-bounded learning algorithm $\mathcal{A}$, and a concept $h : \{0,1\}^n \to Y$, we define the* configuration graph of $\mathcal{A}$ on h *to be a directed graph in which the vertices correspond to the $O(2^{s(n)})$*

states of the algorithm, and there is an edge from a state σ_i to a state σ_j labeled by $x \in \{0,1\}^n$ provided that $\texttt{UPDATE}(\sigma_i, x, \texttt{EVAL}(\sigma_i, x), h(x)) = \sigma_j$.

For a given concept h and algorithm $\mathcal{A}$, we say that a state σ knows (the label of) *$x \in \{0,1\}^n$ if there is a path from the initial state σ_0 to σ such that some edge in the path is labeled by x.*

The key property of our "knowledge" definition is the following:

Proposition 1 *For any on-line learning algorithm that is consistent on h and is in a state σ that knows an instance x, we have $\texttt{EVAL}(\sigma, x) = h(x)$. Furthermore, all states $\tilde{\sigma}$ reachable from σ must also know x.*

Next, we show that after making $s(n)$ mistakes, the fraction of instances not known by the algorithm drops by a constant fraction. For a distribution D, we say that a subset $S \subseteq D$ is a *ρ-fraction of D* provided that $\Pr_{x \in D}[x \in S] = \rho$.

Lemma 3.1 *Suppose that the algorithm $\mathcal{A}$ is in a state σ_t that knows all but a ρ-fraction of D. Then, after a sequence of trials in which instances are drawn from D and $\mathcal{A}$ makes $s(n)$ mistakes, $\mathcal{A}$ enters a state σ_{t+1} that knows all but a $(3/4 \cdot \rho)$-fraction of D, with probability at least $1 - 2^{-s(n)}$.*

Proof. Consider any state $\tilde{\sigma}$ that is reachable from a path from σ_t on which $s(n)$ mistakes occur, and that knows less than a $(1 - 3/4 \cdot \rho)$-fraction of D. Since σ_t knows a $(1-\rho)$-fraction of D, and all mistakes made by $\mathcal{A}$ starting from σ_t must fall in the ρ-fraction that σ_t does not know, this means that the $s(n)$ mistakes leading to $\tilde{\sigma}$ must have all been drawn from a set that has conditional probability at most $1/4$, out of the instances that σ_t does not know. The probability that we hit such a set of conditional probability at most $1/4$ for $s(n)$ times (for our $s(n)$ mistakes) is at most $2^{-2s(n)}$. Therefore the probability that we reach $\tilde{\sigma}$ from σ_t (assuming we make $s(n)$ mistakes) is at most $2^{-2s(n)}$. Taking the union bound over all (at most $2^{s(n)}$) such states $\tilde{\sigma}$, the probability that we reach some state that does not know at least a $(1 - 3/4 \cdot \rho)$-fraction of D is at most $2^{-s(n)}$. □

The final ingredient in the proof of Theorem 3.1 is the following probabilistic inequality, which can be derived from Chernoff-Hoeffding bounds:

Theorem 3.2 *Let $X_1, \ldots, X_t$ be independent Bernoulli random variables such that $E[X_i] \geq 1/2$ for all i, let $\delta > 0$, and let $s \in \mathbb{N}$. Then for $t = 3s + 3\log(1/\delta)$ trials, $\Pr[\sum_{i=1}^{t} X_i < s] \leq \delta$.*

The proof of Theorem 3.2 is in Appendix A. Now we can complete the proof of Theorem 3.1:

Proof of Theorem 3.1. Let $\mathcal{A}$ and h be as in the theorem statement. Until $\mathcal{A}$ has correctly labeled every instance correctly (in which case we are done), $\mathcal{A}$ reduces the fraction of D that it does not know by a $(3/4)$-factor, after every sequence of examples from D in which it makes $s(n)$ mistakes, with probability at least $1 - 2^{-s(n)}$ (independently on each such sequence). This $1 - 2^{-s(n)}$ probability event only has to occur $u = (\log 1/\varepsilon)/\log(4/3)$ times, in order for the fraction of D that $\mathcal{A}$ does not know to drop below ε. Letting $t = 3 \cdot u + 3\log(1/\delta)$ and applying Theorem 3.2, we find that after $t \cdot s(n) = O(s(n)(\log 1/\delta + \log 1/\varepsilon))$ mistakes, the probability that $\mathcal{A}$ knows all but an ε-fraction of D is at least $1 - \delta$. □

Note that there are several examples in the literature of consistent on-line learning algorithms with polynomially bounded state, such as the learning algorithms we have described for learning singletons, learning constant-degree polynomials over F_2, and the elimination algorithm for learning conjunctions (analyzed by Valiant [27] in the original work on PAC-learning) which also lends to learning k-CNF formulas.

Exact learning from Theorem 3.1. It is now straightforward to construct the desired teaching strategies for state-bounded consistent learners: we only need to present the learner with i.i.d. examples from a distribution for which the minimum probabilities are relatively large (at most exponentially small). Of course, the uniform distribution meets these needs optimally:

Reminder of Theorem 1.2 *For every concept $h : \{0,1\}^n \to Y$ and $\delta > 0$, every consistent $s(n)$-bounded $\mathcal{A}$ for h makes at most $O(s(n)(n + \log \frac{1}{\delta}))$ mistakes with probability at least $1 - \delta$, when the instances of length n are chosen uniformly at random.*

Proof. Since the instance space has size 2^n, once the algorithm is in a state that is correct on all but a ε-measure set under the uniform distribution for $\varepsilon < 2^{-n}$, the algorithm must label every instance correctly. The claim thus follows immediately from Theorem 3.1. □

Theorem 1.2 only guarantees that the sequence of examples selected by the strategy works for a particular learner with high probability. Of course, when the probability is exponentially close to 1, we can obtain a *fixed* sequence that guarantees a polynomial mistake bound for *all* of the learners from some finite class. Let $\mathsf{SIZE}[s(n)]$ denote the class of all circuit families $\{C_n\}$ such that C_n has bounded fan-in and $s(n)$ size.

Corollary 3.1 *For every concept $h : \{0,1\}^n \to Y$ and every size bound $s(n) \geq n$, there is a sequence of examples of h such that every $\mathsf{SIZE}[s(n)]$-bounded learner for h makes at most $O(s(n)^2 \log s(n))$ mistakes.*

Proof. First, note that if the learning algorithm is computable by a circuit of size $s(n)$, its states must have length at most $s(n)$. Second, note that there are at most $S = s(n)^{O(s(n))}$ circuits of size $s(n)$. Therefore, by taking $\delta < 1/S$, we find by Theorem 1.2 that a random sequence of examples guarantees that every consistent learner for h with states of size $s(n)$ makes at most $O(s(n)(n+\log S))$ mistakes with probability $> 1 - 1/S$. By taking a union bound over all S circuits, we find that every circuit makes a number of mistakes not exceeding $O(s(n)(n + \log S)) \leq O(s(n)^2 \log s(n))$ with nonzero probability, hence some sequence of examples suffices. □

Finally, before moving on, we note that some polynomial dependence on the space bound (hypothesis size) is essentially inevitable in the mistake bound of any teaching sequence for learners given any reasonable complexity bound. Recall that $\mathcal{S}$ is the class consisting of all singleton and empty concepts.

Reminder of Theorem 1.5 *For every integer $s \in [n, 2^n]$, there is a consistent on-line learning algorithm $\mathcal{A}$ for the concept class $\mathcal{S}$ computed by a uniform family of AC^0 circuits of size $O(s \cdot n)$ using s-bit states, such that for every sequence of examples of the empty concept, $\mathcal{A}$ makes at least $s-1$ mistakes before identifying the empty concept.*

Proof. We describe our "adversarial" learning algorithm $\mathcal{A}$. Fix a lexicographic ordering on strings, and divide the space of $\{0,1\}^n$ into $s-1$ intervals of equal length in which the first $s-2$ intervals all have length $\lfloor 2^n/(s-1) \rfloor$ (and the final interval contains the rest). If $\mathcal{A}$ ever sees an example y such that $h(y) = 1$, then it switches to the singleton for y as its concept. Otherwise, $\mathcal{A}$ represents its concept by a bit-vector of length $s-1$, corresponding to each of these intervals. Initially the bits are all set to 1, and if $\mathcal{A}$ sees an example falling in an interval with its corresponding bit set to 1, `EVAL` predicts 1, otherwise it predicts 0. When the `UPDATE` algorithm sees an example z for which $h(z) = 0$ in an interval with a bit previously set to 1, that bit is set to 0. (All other bits remain unchanged.) We observe that this $\mathcal{A}$ is consistent on the class of singletons with the empty concept—once it sees a 1, it always predicts the singleton correctly (and likewise correctly labels all of the other points it previously saw labeled 0), and until that point, every example it sees labeled

0 will subsequently be predicted to be 0 (along with the rest of the interval it belongs to). We also observe that the lexicographic comparisons can be carried out in uniform AC^0, and that the hypotheses of $\mathcal{A}$ can be represented in s bits as required. Finally, observe that until $\mathcal{A}$ has received an example from each of the $s-1$ intervals, $\mathcal{A}$ does not identify the empty concept. □

On the efficiency of teaching. Although we noted in Theorem 1.1 that teaching concepts such as even the singletons and empty concept may involve presenting exponentially many examples, there is a sense in which the teaching sequences of Theorem 1.2 *are* efficiently generated. Namely, if we consider an on-line communication model such as the one introduced by Goldreich, Juba, and Sudan [13], then as our teaching strategy simply chooses an n-bit example uniformly at random on each round, our strategy is a (randomized) linear-time on-line universal teaching strategy (for space-bounded consistent learners) that achieves polynomial error complexity (in n and the space bound).

3.2 Sequences guaranteeing few mistakes, from Nisan's generator

We have already noted that the teaching sequences we consider must be exponentially long in order to guarantee that *every* consistent algorithm identifies the target concept. Nevertheless, we might hope to improve the construction of such sequences in Theorem 1.2 by reducing the number of random bits they require. As stated, Theorem 1.2 requires exponentially many random bits. However, given that the sequences are uniform-random and the learners have bounded state, one might anticipate that Nisan's pseudorandom generator for space-bounded computation [21] could generate a sequence with similar properties, from a polynomial-length random seed. Although Nisan's analysis says nothing about such a property of the entire string – it only considers the probability that the algorithm ends up in an accepting state – we can confirm that it also generates sequences suitable for our purposes. We prove:

Reminder of Theorem 1.3 *Using a block length of $\Theta(s(n)+n+\log\frac{1}{\delta})$ (and $k=O(n+\log\frac{1}{\delta})$), Nisan's pseudorandom generator produces a sequence of $2^{O(n)}$ random bits for which with probability $1-\delta$ over the seed $h_1,\ldots,h_k,x$, any consistent learning algorithm that is $s(n)$-bounded on a given concept exactly identifies that concept and makes at most $O(s(n)(n+\log\frac{1}{\delta}))$ mistakes.*

Nisan's pseudorandom generator. We recall that Nisan's generator uses a family of pairwise-independent hash functions H taking b bits to b bits, i.e., satisfying the property that for x_1, x_2, y_1, and y_2 in $\{0,1\}^b$, for a uniformly chosen $h \in H$, $\Pr_h[h(x_1)=y_1 \text{ and } h(x_2)=y_2] = 2^{-2b}$; we know that these can be constructed from $O(b)$ random bits by, e.g., multiplying by a random Toeplitz matrix and adding a random vector. The construction is then a recursive construction, in which G_k takes a b-bit seed and k hash functions: $G_0(x)=x$, and

$$G_k(x,h_1,\ldots,h_k) = G_{k-1}(x,h_1,\ldots,h_{k-1}), G_{k-1}(h_k(x),h_1,\ldots,h_{k-1})$$

i.e., concatenating the output of G_{k-1} on x with G_{k-1} on $h_k(x)$. So, the generator stretches $O(kb)$ bits to $b2^k$ bits. We refer to b as the *block length* of the generator (observe that the sequence is a concatenation of blocks of length b obtained by hashing x with the many various subsets of $h_1,\ldots,h_k$).

As is well known, Nisan's generator is also time-efficient in an on-line sense: after the initial choice of seed and k hash functions, the ith block (out of 2^k) of the generator's output may be computed by taking, for each jth bit in the binary representation of i that is a 1, the hash function h_j, and applying them to the seed in order. Thus, by keeping a k-bit counter of the blocks, the sequence of examples can be computed

on-line in time polynomial in n and $s(n)$. So, Theorem 1.3 also yields an efficient on-line universal teaching strategy with polynomial error complexity.

Our starting point is the following lemma encapsulating the abstract version of the analysis of Nisan's generator. For convenience, given $\delta > 0$, for each $k, b \in \mathbb{N}$ and pairs of states i and j of a given (learning) algorithm, we will define events $B_{i,j}^{h_1,\ldots,h_k}$ that $G_k(x, h_1, \ldots, h_k)$ takes state i to state j, and events $A_{i,j}^k$ that a sequence of 2^k uniformly chosen blocks of b bits takes state i to state j.

Lemma 3.2 (Lemma 2 of [21]) *Let any space-$s(n)$ algorithm that reads its input n bits at a time from a read-once input tape be given. Let H be a family of pairwise-independent hash functions on b bits, $\delta > 0$, and $k \in \mathbb{N}$. Then with probability at least $1 - \frac{k2^{6s(n)}}{\delta^2}2^{-b}$ over $h_1, \ldots, h_k$ chosen from H,*

$$\| \left[\Pr[A_{i,j}^k] \right] - \left[\Pr[B_{i,j}^{h_1,\ldots,h_k}] \right] \|_1 \leq (2^k - 1)\delta$$

Now, instead of merely examining the probability that $h_1, \ldots, h_k$ take state i to state j (as Nisan does), we will also examine the probability that the algorithm makes a given number of mistakes, m. Ultimately, we will argue that the *joint distribution* over final states and number of mistakes remains close (in ℓ_1-distance) when Nisan's PRG is substituted for uniform random bits; we do this by noting that the algorithm could be modified to keep count of the number of mistakes that it makes in an n-bit counter (using total space $s(n) + n$) and then the distribution over states of this modified algorithm captures the joint distribution over states of the original algorithm and total number of mistakes. Then by a union bound over the various sources of error, we obtain a general analysis of the quality of Nisan's pseudorandom generator for generating easy sequences of examples:

Theorem 3.3 (Nisan's pseudorandom generator produces easy sequences) *Using a block length of $\Theta(s(n) + n + \log \frac{1}{\delta^*} + \log R)$ rounded to a multiple of n (and $k = \log R/b$), Nisan's pseudorandom generator produces a sequence of R random bits for which with probability $1 - \delta^*$ over the seed $h_1, \ldots, h_k, x$, for any consistent learning algorithm that is space-$s(n)$ bounded on a given concept, the distribution over states and total number of mistakes of the learning algorithm induced by the generator for the concept is δ^*-close to the distribution induced by the uniform distribution over R bits in ℓ_1.*

Proof. Given any consistent space-$s(n)$ bounded learning algorithm A for a given concept, consider the algorithm A' with states given by pairs (σ, m) where σ is a state of A and m is an integer, which simulates A and keeps count of the number of mistakes A makes in the second component; note that since A is consistent, it can never make more than 2^n mistakes (since the concept is then identified) and hence A' uses at most $s(n) + n$ bits of state.

By Lemma 3.2, for this block length, with probability $1 - \delta^*$ over the choice of hash functions and seed, the pseudorandom generator produces a sequence of R random bits such that the statistical distance over final states of A' is δ^*-close to the distribution over states of A' on R uniformly chosen bits; as A' behaves identically to A, by considering the two components of the states of A', we find that the joint distribution over final states of A and total number of mistakes made by A is therefore δ^*-close to the distribution over states and mistakes of A on R uniformly chosen bits. The theorem follows. □

In particular, we can get exact identification sequences, establishing the main theorem of this section:

Proof of Theorem 1.3. By the coupon collector's bound, the expected number of uniformly random examples needed to include every example is $O(n2^n)$, and by Markov's inequality, the probability that this exceeds $\Omega(\frac{1}{\delta}n2^n)$ is at most $\delta/3$. So, using $R = O(\frac{1}{\delta}n^2 2^n)$ random bits, the algorithm enters a state

that knows the entire domain with probability $1-\delta$ (and therefore exactly identifies the concept). By Theorem 1.2, the algorithm makes more than $O(s(n)(n+\log\frac{1}{\delta}))$ mistakes with probability at most $\delta/3$. By a union bound over these two events, we find that on R uniform bits, the algorithm enters a state that identifies the concept and makes at most $O(s(n)(n+\log\frac{1}{\delta}))$ mistakes with probability at least $1-(2/3)\delta$. Therefore, Theorem 3.3 guarantees that for the stated block length, the probability that the algorithm fails to identify the concept or makes more than $O(s(n)(n+\log\frac{1}{\delta}))$ mistakes on the output of the generator is greater by at most $\delta/3$. □

3.3 Deterministic sequences guaranteeing few mistakes implies circuit lower bounds

We now turn to the question of whether the random seed can be removed entirely in generating a sequence which teaches all bounded consistent learners a concept. We show that a deterministic sequence that achieves any mistake bound less than 2^n for all bounded learners implies circuit lower bounds:

Reminder of Theorem 1.4 *Let $\mathcal{F}$ be a class of functions from $\mathbb{N}$ to $\mathbb{N}$. Suppose there is a deterministic $t(n)$ time algorithm M such that for all $s(n) \in \mathcal{F}$ and all sufficiently large n, $M(1^n)$ prints a sequence S of examples of the empty concept $h : \{0,1\}^n \to \{0\}$ such that every consistent $s(n)$-size circuit learner $\mathcal{A}$ for h learns h and makes less than 2^n mistakes on the sequence S. Then there are problems solvable in $t(n)$ time that do not have circuits of size $s(n)$, for all $s(n) \in \mathcal{F}$ and almost every n.*

Proof. By contradiction. Suppose there is a deterministic algorithm that runs in $t(n)$ time and prints a sequence S of examples for the empty concept h with the hypothesized property. Further suppose that for every problem solvable in time $t(n)$ there is an $s(n) \in \mathcal{F}$ such that the problem has circuits of size $s(n)$, for infinitely many input lengths n. First note that all 2^n strings must appear among in S, otherwise some $O(n)$-size circuit learner which learns the class $\mathcal{S}$ of singletons with the empty concept will not be able to distinguish h from some singleton concept (cf. Theorem 1.1). This also entails that $t(n) \geq 2^n$.

We define a "bad" on-line algorithm $\mathcal{A}$ for learning h as follows. First, order the n-bit strings by the order in which they first appear in the sequence S, and re-index the strings as $x_1, x_2, \ldots, x_{2^n}$. (That is, x_1 is the first instance in S, x_2 is the next distinct instance, x_3 is the one after that, and so on. Since every string must appear somewhere in S, this is indeed an ordering on all n-bit strings.)

- The states of $\mathcal{A}$ are the integers from 0 to 2^n, and the initial state is 0.
- Given an example $(x, 0)$ on which a mistake was been made:
  `UPDATE` the state to be the integer i such that $x = x_i$.
- `EVAL`(i, x) predicts 0 if $x = x_j$ for some $j \in [1, i]$, otherwise it predicts 1.

That is, in the initial state 0, all examples are classified as 1, but if $\mathcal{A}$ makes a mistake on example x, x must have the label 1. $\mathcal{A}$ rectifies this by increasing the state to i such that $x = x_i$.

It is easy to verify that, assuming every language in time $t(n)$ has circuits of size $s(n) \in \mathcal{F}$ for infinitely many n, the above `UPDATE` and `EVAL` functions can be implemented with $O(s(n))$-size circuits for some $s(n) \in \mathcal{F}$ and some sufficiently large n – this follows because S can be generated in $t(n)$ time.

Observe that $\mathcal{A}$ is consistent for h: the state i always increases with each mistake, and when we make a mistake on x_i, we increase the state to some $j \geq i$ such that this never happens again. However, on the sequence S, $\mathcal{A}$ makes 2^n mistakes: it predicts every example it sees to have label 1, until all 2^n examples have appeared in S. That is, the state i must equal 2^n in order for $\mathcal{A}$ to correctly label all examples as 0, but when $\mathcal{A}$ receives examples in the sequence S, the state i increases only by 1 for each mistake that is made.

Therefore $\mathcal{A}$ is consistent for the empty concept, and it learns the empty concept, but on the sequence S of examples generated by 1^n, $\mathcal{A}$ makes 2^n mistakes. This is a contradiction. □

Note it is not hard to show that $\mathcal{A}$ actually learns the empty concept from a uniform random sequence of examples with only $O(n)$ mistakes, with high probability, confirming our earlier results. Moreover, it is easy to quickly teach $\mathcal{A}$ the empty concept: simply give it the *last* example $(x_{2^n}, 0)$.

4 Short tutoring sequences

Finally, we consider the problem of generating short sequences of examples tailored to a given learner and given concept that lead the learner to exactly identify the concept. We will informally refer to these as *tutoring sequences*. Although the motivation for considering such sequences is somewhat different from that we discussed for the "massive on-line" teaching models, one can think of it as a communication model that reveals a sense in which a weaker receiver may be easier to communicate with.

The technique from Theorem 3.1 also easily yields a probabilistic (nonconstructive) proof of existence of such sequences, which have *length* that is only polynomially related to the learner's state complexity. This is in stark contrast to the earlier setting, where the lengths of sequences were forced to be exponential.

Reminder of Theorem 1.6 *For every concept class $\mathcal{H}$, every $h \in \mathcal{H}$, and every consistent $s(n)$-bounded learner $\mathcal{A}$ for $\mathcal{H}$, there is a sequence of examples of length $O(n \cdot s(n))$ after which $\mathcal{A}$ identifies h.*

Proof. The argument is quite similar to Theorem 3.1: we again consider the configuration graph of the algorithm, and will argue that the set of examples the algorithm does not know shrinks exponentially. We first state an appropriately modified version of Lemma 3.1 (assuming the uniform distribution, along the lines of Theorem 1.2):

Claim 1 *Suppose that the on-line algorithm is in a state σ_t that knows all but a ρ-fraction of $\{0,1\}^n$. Then, after a sequence of $s(n)$ instances chosen uniformly at random from the set of instances that σ_t does not know, the algorithm enters a state σ_{t+1} that knows all but a $(3/4 \cdot \rho)$-fraction of $\{0,1\}^n$, with probability at least $1 - 2^{-s(n)}$.*

The proof is essentially the same as Lemma 3.1. Now, since the algorithm reaches such a state with nonzero probability, we can in particular fix a sequence of $s(n)$ examples for which the remaining instances decreases by a factor of $3/4$; thus, after $\log_2(4/3) \cdot n$ of these sequences of $s(n)$ examples, the algorithm identifies the target concept as needed. □

5 Conclusion

We have introduced a new model of teaching that attempts to teach all "worst-case learners" a concept with a single sequence of labeled examples, and have established that the number of mistakes made by consistent learners on random examples are only polynomially related to the learner's state complexity. Several interesting questions arise naturally.

The most immediate question is: how tight is the connection between mistakes and state complexity? The lower bound in Theorem 1.5 only provides an $\Omega(s(n))$ lower bound on the number of mistakes, whereas our constructions all achieve mistake bounds of $O(s(n) \cdot n)$. Is this extra factor of n essential (say, for constant probability of success)?

Of course, our results analyze the mistake bound just in terms of the state bound of the (consistent) algorithm. In the standard (batch) PAC model, analyses based on representation sizes have been very useful, e.g., for analyzing Rivest's algorithm for learning decision lists [23]. Our analysis may not be very useful for many learning algorithms, because they do not satisfy such a strong "consistency" condition (it is often the case that an algorithm may initially label an example correctly, and later switch to a hypothesis that labels it incorrectly). Is it possible to generalize the class of learning algorithms further and achieve similar results? Perhaps our work can help give a novel analysis of some interesting on-line learning algorithms.

We have found another setting where uniform random bits do the job, but generating similar bits deterministically would entail circuit lower bounds. Could we build a sequence for ACC-circuit learners in such a way that we separate EXP from ACC? Does $\mathsf{EXP} \not\subset \mathsf{P/poly}$ imply the existence of good deterministic teaching sequences?

Acknowledgements. We are grateful to Manuel Blum for having introduced us to LINCOS. We thank Varun Kanade for a useful discussion.

References

[1] Dana Angluin and Mārtiņš Krikis. Learning from different teachers. *Mach. Learn.*, 51(2):137–163, 2003.

[2] Martin Anthony, Graham Brightwell, and John Shawe-Taylor. On specifying Boolean functinos by labelled examples. *Discrete Applied Mathematics*, 61(1):1–25, 1995.

[3] Frank J. Balbach. Measuring teachability using variants of the teaching dimension. *Theoret. Comput. Sci.*, 397(1–3):94–113, 2008.

[4] Frank J. Balbach and Thomas Zeugmann. Teaching learners with restricted mind changes. In *Proc. 16th ALT*, volume 3734, pages 474–489. Springer, 2005.

[5] Frank J. Balbach and Thomas Zeugmann. Teaching randomized learners. In *Proc. 19th COLT*, volume 4005, pages 229–243. Springer, 2008.

[6] Frank J. Balbach and Thomas Zeugmann. Recent developments in algorithmic teaching. In *Proc. 3rd Int'l Conf. on Language and Automata Theory and Applications*, pages 1–18, 2009.

[7] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Inf. Process. Lett.*, 24:377–380, 1987.

[8] Sally Floyd. *On space-bounded learning and the Vapnik-Chervonenkis dimension*. PhD thesis, University of California, Berkeley, Berkeley, CA, 1989.

[9] Sally Floyd and Manfred Warmuth. Sample compression, learnability, and the Vapnik-Chervonenkis dimension. *Mach. Learn.*, 21:269–304, 1995.

[10] Hans Freudenthal. *LINCOS: Design of a language for cosmic intercourse*. North-Holland, Amsterdam, 1960.

[11] Sally A. Goldman and Michael J. Kearns. On the complexity of teaching. *J. Comput. Syst. Sci.*, 50(1):20–31, 1995.

[12] Sally A. Goldman and H. David Mathias. Teaching a smarter learner. *J. Comput. Syst. Sci.*, 52(2):255–267, 1996.

[13] Oded Goldreich, Brendan Juba, and Madhu Sudan. A theory of goal-oriented communication. *J. ACM*, 59(2):8:1–8:65, 2012.

[14] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

[15] Russell Impagliazzo and Avi Wigderson. P=BPP unless E has subexponential circuits: Derandomizing the XOR lemma. In *Proc. 29th STOC*, pages 220–229, 1997.

[16] Jeffrey Jackson and Andrew Tomkins. A computational model of teaching. In *Proc. 5th COLT*, pages 319–326, 1992.

[17] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

[18] Nick Littlestone. From on-line to batch learning. In *Proc. COLT'89*, pages 269–284, 1989.

[19] Nick Littlestone and Manfred K. Warmuth. Relating data compression and learnability. Unpublished manuscript, 1986.

[20] H. David Mathias. A model of interactive teaching. *J. Comput. Syst. Sci.*, 54(3):487–501, 1997.

[21] Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

[22] Rūsiņš Freivalds, Efim B. Kinber, and Rolf Wiehagen. On the power of inductive inference from good examples. *Theoret. Comput. Sci.*, 110(1):131–144, 1993.

[23] Ronald L. Rivest. Learning decision lists. *Mach. Learn.*, 2(3):229–246, 1987.

[24] Dale Schuurmans and Russell Greiner. Sequential PAC learning. In *Proc. 8th COLT*, pages 377–384, 1995.

[25] Rocco A. Servedio. On the limits of efficient teachability. *Inf. Process. Lett.*, 79(6):267–272, 2001.

[26] Ayumi Shinohara and Satoru Miyano. Teachability in computational learning. *New Generation Comput.*, 8(4):337–347, 1991.

[27] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 18(11):1134–1142, 1984.

[28] Sandra Zilles, Steffen Lange, Robert Holte, and Martin Zinkevich. Models of cooperative teaching and learning. *JMLR*, 12:349–384, 2012.

A Appendix

Here we prove:

Reminder of Theorem 3.2 *Let $X_1, \ldots, X_t$ be independent Bernoulli random variables such that $E[X_i] \geq 1/2$ for all i, let $\delta > 0$, and let $s \in \mathbb{N}$. Then for $t = 3s + 3\log(1/\delta)$ trials, $\Pr[\sum_{i=1}^{t} X_i < s] \leq \delta$.*

The theorem will follow from a calculation using Hoeffding's inequality (a.k.a. the additive Chernoff bound).

Theorem A.1 *Let $X_1, \ldots, X_t$ be independent Bernoulli random variables such that $E[X_i] \geq 1/2$ for all i, and $\mathbb{E}[\frac{1}{t}\sum_{i=1}^{t} X_i] = \mu$. Then for any $\epsilon > 0$,*

$$\Pr\left[\frac{1}{t}\sum_{i=1}^{t} X_i < \mu - \epsilon\right] \leq \exp(-2t\epsilon^2)$$

To prove our inequality, we want to compute the number of trials t so that at least s of these X_i come up 1, with probability $1 - \delta$. Naturally, in $2s$ trials, we have at least s in expectation; we will calculate the number of additional trials a needed to guarantee at least s with probability $1 - \delta$. That is, we want a such that the probability that fewer than s out of the $2s + a$ trials come up 1 is at most δ. Plugging in Hoeffding's inequality, we find that a satisfying

$$\exp\left(-2(2s+a)\left(\frac{a}{2(2s+a)}\right)^2\right) = \exp\left(\frac{-a^2}{2(2s+a)}\right) \leq \delta$$

suffices. As $a > 0$ increases, the LHS decreases, so we merely need to find the smallest a that suffices. We note that equality holds when

$$\begin{aligned} \ln\frac{1}{\delta} &= \frac{a^2}{2(2s+a)} \\ 0 &= a^2 - 2a\ln\frac{1}{\delta} - 4s\ln\frac{1}{\delta} \end{aligned}$$

which has the solutions

$$a = \frac{2\ln\frac{1}{\delta} \pm \sqrt{4\ln^2\frac{1}{\delta} + 16s\ln\frac{1}{\delta}}}{2} = \ln\frac{1}{\delta} \pm \sqrt{\ln^2\frac{1}{\delta} + 4s\ln\frac{1}{\delta}}.$$

To simplify the expression for a, we recall that $\sqrt{x^2+y^2} \leq x + y$ and $\sqrt{xy} \leq (x+y)/2$ for non-negative x and y, and obtain:

$$\begin{aligned} a &= \ln\frac{1}{\delta} + \sqrt{\ln^2\frac{1}{\delta} + 4s\ln\frac{1}{\delta}} \\ &\leq \ln\frac{1}{\delta} + \ln\frac{1}{\delta} + 2\sqrt{s\ln\frac{1}{\delta}} \\ &\leq \ln\frac{1}{\delta} + \ln\frac{1}{\delta} + s + \ln\frac{1}{\delta} \end{aligned}$$

Therefore, $t = 2s + a \leq 3s + 3\ln(1/\delta)$ trials suffice.