Can Learning in the Limit be Done Efficiently ?

Thomas Zeugmann [1]

Institut für Theoretische Informatik, Universität zu Lübeck, Wallstraße 40, 23560 Lübeck, Germany
thomas@tcs.uni-luebeck.de

Abstract. Inductive inference can be considered as one of the fundamental paradigms of algorithmic learning theory. We survey results recently obtained and show their impact to potential applications.
Since the main focus is put on the efficiency of learning, we also deal with postulates of naturalness and their impact to the efficiency of limit learners. In particular, we look at the learnability of the class of all pattern languages and ask whether or not one can design a learner within the paradigm of learning in the limit that is nevertheless efficient.
For achieving this goal, we deal with iterative learning and its interplay with the hypothesis spaces allowed. This interplay has also a severe impact to postulates of naturalness satisfiable by any learner.
Finally, since a limit learner is only supposed to converge in the limit, one never knows at any particular learning stage whether or not the learner did already succeed. The resulting uncertainty may be prohibitive in many applications. We survey results to resolve this problem by outlining a new learning model, called *stochastic finite learning.* Though pattern languages can neither be finitely inferred from positive data nor PAC-learned, our approach can be extended to a stochastic finite learner that *exactly* infers all pattern languages from positive data with *high confidence.*

1. Introduction

Inductive inference can be considered as one of the fundamental paradigms of algorithmic learning theory. In particular, inductive inference of recursive functions and of recursively enumerable languages have been studied intensively within the last four decades (cf., e.g., [3, 4, 30, 16]). The basic model considered within this framework is learning in the limit which can be informally described as follows. The learner receives more and more data about the target and maps these data to hypotheses. Of special interest is the investigation of scenarios in which the sequence of hypotheses *stabilizes* to an *accurate* and *finite* description (e.g. a grammar, a program) of the target. Clearly, then some form of learning must have taken place. Here by data we mean either any infinite sequence of pairs argument-value (in case of learning recursive functions) such that all arguments appear eventually or any infinite sequence of all members of the target language (in case of language learning from positive data). Alternatively, one can also study language learning from both positive and negative data.

Most of the work done in the field has been aimed at the following goals: showing what general collections of function classes or language classes are learnable, characterizing those collections of classes that can be learned, studying the impact of several postulates on the behavior of learners to their learning power, and dealing with the influence of various parameters to the efficiency of learning. However, defining an appropriate measure for the complexity of learning in the limit has turned out to be quite difficult (cf. Pitt [31]). Moreover, whenever learning in the limit is done, in general one never knows whether or not the learner has already converged. This is caused by the fact that it is either undecidable at all whether or not convergence already occurred. But even if it is decidable, it is practically infeasible to do so. Thus, there is always an uncertainty which may not be tolerable in many applications of learning.

Therefore, different learning models have been proposed. In particular, Valiant's [46] model of *probably approximately correct* (abbr. PAC) learning has been very influential. As a matter of fact, this model puts strong emphasis on the efficiency of learning and avoids the problem of convergence at all. In the PAC model, the learner receives a finite labeled sample of the target concepts and outputs, with high probability, a hypothesis that is approximately correct. The sample is drawn with respect to an unknown probability distribution and the error of as well as the confidence in the hypothesis are measured with respect to this distribution, too. Thus, if a class is PAC learnable, one obtains nice performance guarantees. Unfortunately, many interesting concept classes are not PAC learnable.

Consequently, one has to look for other models of learning or one is back to learning in the limit. So, let us assume that learning in the limit is our method of choice. What we would like to present in this survey is a rather general way to transform learning in the limit into *stochastic finite learning.* It should also be noted that our ideas may be beneficial even in case that the considered concept class is PAC learnable.

Furthermore, we aim to outline how a thorough study of limit learnability of concept classes may nicely contribute to support our new approach. We exemplify the research undertaken by mainly looking at the class of all pattern languages introduced by Angluin [1]. As Salomaa [37] has put it "Patterns are everywhere" and thus we believe that our research is worth the effort undertaken.

There are several problems that have to be addressed when dealing with the learnability of pattern languages. First, the nice thing about patterns is that they are very intuitive. Therefore, it seems desirable to design learners outputting pattern as their hypotheses. Unfortunately, membership is known to be $\mathcal{NP}$-complete for the pattern languages (cf. [1]). Thus, many of the usual approaches used in machine learning will directly lead to infeasible learning algorithms. As a consequence, we shall ask what kind of appropriate hypothesis spaces can be used at all to learn the pattern languages, and what are the appropriate learning strategies.

In particular, we shall deal with the problem of redundancy in the hypothesis space chosen, with consistency, conservativeness, and iterative learning. Here consistency means that the intermediate hypotheses output by the learner do correctly reflect the data seen so far. Conservativeness addresses the problem to avoid overgeneralization, i.e., preventing the learner from guessing a proper superset of the target language. These requirements are naturally arising desiderata, but this does not mean that they can be fulfilled. With *iterative* learning, the learning machine, in making a conjecture, has access to its previous conjecture and the latest data item coming in. Iterative learning is also a natural requirement whenever learning in the limit is concerned, since no practical learner can process at every learning stage all examples provided so far, it may even not be able to store them.

Finally, we address the question how efficient the overall learning process can be performed, and how we can get rid of the uncertainty of not knowing whether or not the learner has already converged.

2. Preliminaries

Unspecified notation follows Rogers [35]. By $\mathbb{N} = \{0, 1, 2, \ldots\}$ we denote the set of all natural numbers. We set $\mathbb{N}^+ = \mathbb{N} \setminus \{0\}$. The cardinality of a set S is denoted by $|S|$. Let $\emptyset$, $\in$, $\subset$, $\subseteq$, $\supset$, and $\supseteq$, denote the empty set, element of, proper subset, subset, proper superset, and superset, respectively.

Let $\varphi_0, \varphi_1, \varphi_2, \ldots$ denote any fixed *acceptable programming system* for all (and only) the partial recursive functions over $\mathbb{N}$ (cf. Rogers [35]). Then φ_k is the partial recursive function computed by *program* k .

In the following subsection we define the main learning models considered within this paper.

2.1. *Learning in the Limit*

Gold's [12] model of learning in the limit allows one to formalize a rather general class of learning problems, i.e., learning from examples. For defining this model we assume any recursively enumerable set $\mathcal{X}$ and refer to it as the *learning domain.* By $\wp(\mathcal{X})$ we denote the power set of $\mathcal{X}$. Let $\mathcal{C} \subseteq \wp(\mathcal{X})$, and let $c \in \mathcal{C}$ be non-empty; then we refer to $\mathcal{C}$ and c as a *concept class* and a *concept*, respectively. Let c be a concept, and let $t = (x_j)_{j\in\mathbb{N}}$ be any infinite sequence of elements $x_j \in c$ such that $\mathrm{range}(t) =_{df} \{x_j \mid j \in \mathbb{N}\} = c$. Then t is said to be a *positive presentation* or, synonymously, a *text* for c . By $\mathit{text}(c)$ we denote the set of all positive presentations for c . Moreover, let t be a positive presentation, and let $y \in \mathbb{N}$. Then, we set $t_y = x_0, \ldots, x_y$, i.e., t_y is the initial segment of t of length $y + 1$, and $t_y^+ =_{df} \{x_j \mid j \leq y\}$. We refer to t_y^+ as the *content* of t_y .

Furthermore, let $\sigma = x_0, \ldots, x_{n-1}$ be any finite sequence. Then we use $|\sigma|$ to denote the *length* n of σ , and let $\mathrm{content}(\sigma)$ and σ^+ , respectively, denote the content of σ . Additionally, let t be a text and let τ be a finite sequence; then we use $\sigma \diamond t$ and $\sigma \diamond \tau$ to denote the sequence obtained by *concatenating* σ onto the front of t and τ , respectively.

Alternatively, one can also consider *complete presentations* or, synonymously, *informants.* Let c be a concept; then any sequence $i = (x_j, b_j)_{j\in\mathbb{N}}$ of labeled examples, where $b_j \in \{+,-\}$ such that $\{x_j \mid j \in \mathbb{N}\} = \mathcal{X}$ and $i^+ = \{x_j \mid (x_j, b_j) = (x_j, +),\ j \in \mathbb{N}\} = c$ and $i^- = \{x_j \mid (x_j, b_j) = (x_j, -),\ j \in \mathbb{N}\} = \mathcal{X} \setminus c$ is called an informant for c . For the sake of presentation, the following definitions are only given for the text case, the generalization to the informant case should be obvious. We sometimes use the term *data sequence* to refer to both text and informant, respectively.

An *inductive inference machine* (abbr. IIM) is an algorithm that takes as input larger and larger initial segments of a text and outputs, after each input, a hypothesis from a prespecified *hypothesis space* $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$. The indices j are regarded as suitable finite encodings of the concepts described by the hypotheses. A hypothesis is said to describe a concept c iff $c = h$.

Definition 1. Let $\mathcal{C}$ be any concept class, and let $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ be a hypothesis space for it. $\mathcal{C}$ is called *learnable in the limit from text* iff there is an IIM M such that for every $c \in \mathcal{X}$ and every text t for c ,

(1) for all $n \in \mathbb{N}^+$, $M(t_n)$ is defined,
(2) there is a j such that $c = h_j$ and for all but finitely many $n \in \mathbb{N}^+$, $M(t_n) = j$.

By *LimTxt* we denote the collection of all concepts classes $\mathcal{C}$ that are learnable in the limit from text[1]. Note that instead of *LimTxt* sometimes *TxtEx* is used.

Note that Definition 1 does not contain any requirement concerning efficiency. Before we are going to deal with efficiency, we want to point to another crucial parameter of our learning model, i.e., the hypothesis space $\mathcal{H}$. Since our goal is *algorithmic* learning, we can consider the special case that $\mathcal{X} = \mathbb{N}$ and let $\mathcal{C}$ be any subset of the collection of all recursively enumerable sets over $\mathbb{N}$. Let $W_i = \text{domain}\varphi_i$. In this case, $(W_j)_{j\in\mathbb{N}}$ is the most general hypothesis space.

Within this setting many learning problems can be described. Moreover, this setting has been used to study the general capabilities of different learning models which can be obtained by suitable modifications of Definition 1. There are numerous papers performing studies along this line of research (cf., e.g., [16, 30] and the references therein). On the one hand, the results obtained considerably broaden our general understanding of algorithmic learning. On the other hand, one has also to ask what kind of consequences one may derive from these results for practical learning problems. This is a non-trivial question, since the setting of learning recursively enumerable languages is very rich. Thus, it is conceivable

[1] If learning from informant is considered we use *LimInf* to denote the collection of all concepts classes $\mathcal{C}$ that are learnable in the limit from informant.

that several of the phenomena observed hold in this setting due to the fact too many sets are recursively enumerable and there are no counterparts within the world of efficient computability.

As a first step to address this question we mainly consider the scenario that *indexable concept* classes with uniformly decidable membership have to be learned (cf. Angluin [2]). A class of non-empty concepts $\mathcal{C}$ is said to be an ***indexable class*** with uniformly decidable membership provided there are an effective enumeration $c_0, c_1, c_2, ...$ of all and only the concepts in $\mathcal{C}$ and a recursive function f such that for all $j \in \mathbb{N}$ and all elements $x \in \mathcal{X}$ we have

$$f(j,x) = \begin{cases} 1, & \text{if} \quad x \in c_j, \\ 0, & \text{otherwise.} \end{cases}$$

In the following we refer to indexable classes with uniformly decidable membership as to indexable classes for short. Furthermore, we call any enumeration $(c_j)_{j\in\mathbb{N}}$ of $\mathcal{C}$ with uniformly decidable membership problem an *indexed family.*

Since the paper of Angluin [2] learning of indexable concept classes has attracted much attention (cf., e.g., Zeugmann and Lange [51]). Let us shortly provide some-well known indexable classes. Let Σ be any finite alphabet of symbols, and let $\mathcal{X}$ be the free monoid over Σ, i.e., $\mathcal{X} = \Sigma^*$. We set $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$, where λ denotes the empty string. As usual, we refer to subsets $L \subseteq \mathcal{X}$ as languages. Then the set of all regular languages, context-free languages, and context-sensitive languages are indexable classes.

Next, let $X_n = \{0,1\}^n$ be the set of all n-bit Boolean vectors. We consider $\mathcal{X} = \bigcup_{n\geq 1} X_n$ as learning domain. Then, the set of all concepts expressible as a monomial, a k-CNF, a k-DNF, and a k-decision list form indexable classes.

When learning indexable classes $\mathcal{C}$, it is generally assumed that the hypothesis space $\mathcal{H}$ has to be an indexed family, too. We distinguish *class preserving learning* and *class comprising learning* defined by $\mathcal{C} = \texttt{range}(\mathcal{H})$ and $\mathcal{C} \subseteq \texttt{range}(\mathcal{H})$, respectively. When dealing with class preserving learning, one has the freedom to choose as hypothesis space a possibly *different enumeration* of the target class $\mathcal{C}$. In contrast, when class comprising learning is concerned, the hypothesis space may enumerate, additionally, languages not belonging to $\mathcal{C}$. Note that, in general, one has to allow class comprising hypothesis spaces to obtain the maximum possible learning power (cf. Lange and Zeugmann [20, 22]). Finally, we call an hypothesis space *redundant* if it is larger than necessary, i.e., there is at least one hypothesis in $\mathcal{H}$ not describing any concept from the target class or one concept possesses at least two different descriptions in $\mathcal{H}$. Thus, *non-redundant* hypothesis spaces are as small as possible.

Formally, a hypothesis space $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ is *non-redundant* for some target concept class $\mathcal{C}$ iff $\texttt{range}(\mathcal{H}) = \mathcal{C}$ and $h_i \neq h_j$ for all $i, j \in \mathbb{N}$ with $i \neq j$. Otherwise, $\mathcal{H}$ is a *redundant* hypothesis space for $\mathcal{C}$.

Next, let us come back to the issue of efficiency. Looking at Definition 1 we see that an IIM M has always access to the whole history of the learning process, i.e., in order to compute its actual guess M is fed all examples seen so far. In contrast to that, next we define *iterative IIMs*. An iterative IIM is only allowed to use its last guess and the next element in the positive presentation of the target concept for computing its actual guess. Conceptionally, an iterative IIM M defines a sequence $(M_n)_{n\in\mathbb{N}}$ of machines each of which takes as its input the output of its predecessor.

DEFINITION 2 (Wiehagen [47]). Let $\mathcal{C}$ be a concept class, let c be a concept, let $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ be a hypothesis space, and let $a \in \mathbb{N}\cup\{*\}$. *An IIM* M $ItLimTxt_{\mathcal{H}}$ *-infers* c iff for every $t = (x_j)_{j\in\mathbb{N}} \in text(c)$ the following conditions are satisfied:

(1) for all $n \in \mathbb{N}$, $M_n(T)$ is defined, where $M_0(T) =_{df} M(x_0)$ and for all $n \geq 0:$ $M_{n+1}(T) =_{df} M(M_n(T), x_{n+1})$,
(2) the sequence $(M_n(T))_{n\in\mathbb{N}}$ converges to a number j such that $c = h_j$.

Finally, M $ItLimTxt_{\mathcal{H}}$ *-infers* $\mathcal{C}$ iff, for each $c \in \mathcal{C}$, M $ItLimTxt_{\mathcal{H}}$-infers c.

In the latter definition $M_n(t)$ denotes the $(n+1)$ th hypothesis output by M when successively fed the text t. Thus, it is justified to make the following convention. Let $\sigma = x_0, \ldots, x_n$ be any finite sequence of elements over the relevant learning domain. Moreover, let $\mathcal{C}$ be any concept class over $\mathcal{X}$, and let M be any IIM that iteratively learns $\mathcal{C}$. Then we denote by $M_y(\sigma)$ the $(y+1)$ th hypothesis output by M when successively fed σ provided $y \leq n$, and there exists a concept $c \in \mathcal{C}$ with $\sigma^+ \subseteq c$. Furthermore, we let $M_*(\sigma)$ denote $M_{|\sigma|-1}(\sigma)$.

Moreover, whenever learning a concept class from text, a major problem one has to deal with is avoiding or detecting *overgeneralization*. An overgeneralization occurs if the learner is guessing a superconcept of the target concept. Clearly, such an overgeneralized guess cannot be detected by using the incoming positive data only. Therefore, one may be tempted to disallow overgeneralized guesses at all. Learners behaving thus are called conservative. Intuitively speaking a conservative IIM maintains its actual hypothesis at least as long as it has not seen data contradicting it. More formally, an IIM M is said to be ***conservative*** iff for all concepts c in the target class $\mathcal{C}$ and all texts t for c the condition $M(t_y) \neq M(t_{y+z})$ then $t^+_{y+z} \not\subseteq h_{M(t_y)}$ is fulfilled.

Another property of learners quite often found in the literature is consistency. Informally, a learner is called consistent if all its intermediate hypotheses do correctly reflect the data seen so far. More formally, an IIM M is said to be ***consistent*** iff $t^+_x \subseteq h_{M(t_x)}$ for all $x \in \mathbb{N}$ and every text t for every concept c in the target class $\mathcal{C}$.

Whenever one talks about the efficiency of learning besides the storage needed by the learner one has also to consider the time complexity of the learner. When talking about the time complexity of learning, it does not suffice to consider the time needed to compute the actual guess. What really counts in applications is the overall time needed until successful learning. Therefore, following Daley and Smith [10] we define the total learning time as follows.

Let $\mathcal{C}$ be any concept class, and let M be any IIM that learns $\mathcal{C}$ in the limit. Then, for every $c \in \mathcal{C}$ and every text t for c, let

$$\begin{aligned} Conv(M,t) =_{df} & \text{ the least number } m \in \mathbb{N}^+ \\ & \text{ such that for all } n \geq m,\ M(t_n) = M(t_m) \end{aligned}$$

denote the *stage of convergence* of M on t (cf. [12]). Note that $Conv(M,t) = \infty$ if M does not learn the target concept from its text t. Moreover, by $T_M(t_n)$ we denote the time to compute $M(t_n)$. We measure this time as a function of the length of the input and call it the *update time*. Finally, the total learning time taken by the IIM M on successive input t is defined as

$$TT(M,t) =_{df} \sum_{n=1}^{Conv(M,t)} T_M(t_n).$$

Clearly, if M does not learn the target concept from text t then the total learning time is *infinite*.

Two more remarks are in order here. First, it has been argued elsewhere that within the learning in the limit paradigm a learning algorithm is invoked only when the current hypothesis has some problem with the latest observed data. However, such a viewpoint implicitly assumes that membership in the target concept is decidable in time polynomial in the length of the actual input. This may be not case. Thus, directly testing consistency would immediately lead to a non-polynomial update time provided membership is not known to be in $\mathcal{P}$.

Second, Pitt [31] addresses the question with respect to what parameter one should measure the total learning time. In the definition given above this parameter is the length of all examples seen so far. Clearly, now one could try to play with this parameter by waiting for a large enough input before declaring success. However, when dealing with the learnability of non-trivial concept classes, in the worst-case the total learning time will be anyhow unbounded. Thus, it does not make much sense to deal with the worst-case. Instead, we shall study the *expected* total learning time. In such a setting one cannot simply wait for long enough inputs. Therefore, using the definition of total learning time given above seems to be reasonable.

Next, we define important concept classes which we are going to consider throughout this survey.

2.2. *The Pattern Languages*

Following Angluin [1] we define patterns and pattern languages as follows. Let $\mathcal{A} = \{0, 1, \ldots\}$ be any non-empty finite alphabet containing at least two elements. By $\mathcal{A}^*$ we denote the free monoid over $\mathcal{A}$. The set of all finite non-null strings of symbols from $\mathcal{A}$ is denoted by $\mathcal{A}^+$, i.e., $\mathcal{A}^+ = \mathcal{A}^* \setminus \{\lambda\}$, where λ denotes the empty string. Let $X = \{x_i \mid i \in \mathbb{N}\}$ be an infinite set of variables such that $\mathcal{A} \cap X = \emptyset$. *Patterns* are non-empty strings over $\mathcal{A} \cup X$, e.g., 01, $0x_0111$, $1x_0x_00x_1x_2x_0$ are patterns. The length of a string $s \in \mathcal{A}^*$ and of a pattern π is denoted by $|s|$ and $|\pi|$, respectively. A pattern π is in *canonical form* provided that if k is the number of different variables in π then the variables occurring in π are precisely $x_0, \ldots, x_{k-1}$. Moreover, for every j with $0 \leq j < k-1$, the leftmost occurrence of x_j in π is left to the leftmost occurrence of x_{j+1}. The examples given above are patterns in canonical form. In the sequel we assume, without loss of generality, that all patterns are in canonical form. By Pat we denote the set of all patterns in canonical form.

If k is the number of different variables in π then we refer to π as to a k*-variable pattern.* By Pat_k we denote the set of all k*-variable patterns.* Furthermore, let $\pi \in Pat_k$, and let $u_0, \ldots, u_{k-1} \in \mathcal{A}^+$; then we denote by $\pi[x_0/u_0, \ldots, x_{k-1}/u_{k-1}]$ the string $w \in \mathcal{A}^+$ obtained by substituting u_j for each occurrence of x_j, $j = 0, \ldots, k-1$, in the pattern π. For example, let $\pi = 0x_01x_1x_0$. Then $\pi[x_0/10, x_1/01] = 01010110$. The tuple $(u_0, \ldots, u_{k-1})$ is called a *substitution.* Furthermore, if $|u_0| = \cdots = |u_{k-1}| = 1$, then we refer to $(u_0, \ldots, u_{k-1})$ as to a *shortest substitution.* Let $\pi \in Pat_k$; we define the *language generated by pattern* π by $L(\pi) = \{\pi[x_0/u_0, \ldots, x_{k-1}/u_{k-1}] \mid u_0, \ldots, u_{k-1} \in \mathcal{A}^+\}$. By PAT_k we denote the set of all k*-variable pattern languages.* Finally, $PAT = \bigcup_{k\in\mathbb{N}} PAT_k$ denotes the set of all pattern languages over $\mathcal{A}$.

Furthermore, we let Q range over finite sets of patterns and define $L(Q) = \bigcup_{\pi\in Q} L(\pi)$, i.e., the union of all pattern languages generated by patterns from Q. Moreover, we use $Pat(k)$ and $PAT(k)$ to denote the family of all unions of at most k canonical patterns and the family of all unions of at most k pattern languages, respectively. That is, $Pat(k) = \{Q \mid Q \subseteq Pat,\ |Q| \leq k\}$ and $PAT(k) = \{L \mid (\exists Q \in Pat(k))[L = L(Q)]\}$. Finally, let $L \subseteq \mathcal{A}^+$ be a language, and let $k \in \mathbb{N}^+$; we define $Club(L,k) = \{Q \mid |Q| \leq k,\ L \subseteq L(Q),\ (\forall Q')[Q' \subset Q \Rightarrow L \not\subseteq L(Q')]\}$. *Club* stands for **c**onsistent **l**east **u**pper **b**ounds.

The pattern languages have been intensively investigated (cf., e.g., Salomaa [37, 38], and Shinohara and Arikawa [43] for an overview). Nix [29] as well as Shinohara and Arikawa [43] outlined interesting applications of pattern inference algorithms. For example, pattern language learning algorithms have been successfully applied for solving problems in molecular biology (cf., e.g., Shimozono *et al.* [39], Shinohara and Arikawa [43]).

As it turned out, pattern languages and finite unions of pattern languages are subclasses of Smullyan's [45] elementary formal systems (Abbr. EFS). Arikawa *et al.* [5] have shown that EFS can also be treated as a logic programming language over strings. Recently, the techniques for learning finite unions of pattern languages have been extended to show the learnability of various subclasses of EFS (cf. Shinohara [42]). The investigations of the learnability of subclasses of EFSs are interesting because they yield corresponding results about the learnability of subclasses of logic programs. Hence, these results are also of relevance for Inductive Logic Programming (ILP) [28, 23, 8, 24]. Miyano *et al.* [26] intensively studied the polynomial-time learnability of EFSs.

Therefore, we may consider the learnability of pattern languages and of unions thereof as a nice test bed for seeing what kind of results one may obtain by considering the corresponding learning problems within the setting of learning in the limit.

3. Results

Within this section we ask whether or not the pattern languages and finite unions thereof can be learned efficiently. The principal learnability of the pattern languages from text with respect to the hypothesis space Pat has been established by Angluin [1]. However, her algorithm is based on computing descriptive patterns for the data seen so far. Here a pattern π is said to be descriptive (for the set S of strings contained in the input provided so far) if π can generate all strings contained in S and no other pattern with this property generates a proper subset of the language generated by π. Since no efficient algorithm is known for computing descriptive patterns, and finding a descriptive pattern of *maximum* length is $\mathcal{NP}$ -hard, its update time is practically *intractable.*

There are also serious difficulties when trying to learn the pattern languages within the PAC model introduced by Valiant [46]. In the original model, the sample complexity depends exclusively on the VC dimension of the target concept class and the error and confidence parameters ε and δ, respectively. Recently, Mitchell *et al.* [25] have shown that even the class of all one-variable pattern languages has infinite VC dimension. Consequently, even this special subclass of PAT is not uniformly PAC learnable. Moreover, Schapire [40] has shown that pattern languages are not PAC learnable in the generalized model provided $\mathcal{P}/poly \neq \mathcal{NP}/poly$ with respect to every hypothesis space for PAT that is uniformly polynomially evaluable. Though this result highlights the difficulty of PAC learning PAT it has no clear application to the setting considered in this paper, since we aim to learn PAT with respect to the hypothesis space Pat . Since the membership problem for this hypothesis space is $\mathcal{NP}$-complete, it is *not* polynomially evaluable (cf. [1]).

In contrast, Kearns and Pitt [18] have established a PAC learning algorithm for the class of all k-variable pattern languages. Positive examples are generated with respect to arbitrary product distributions while negative examples are allowed to be generated with respect to any distribution. In their algorithm the length of substitution strings is required to be polynomially related to the length of the target pattern. Finally, they use as hypothesis space all unions of polynomially many patterns that have k or fewer variables[2]. The overall learning time of their PAC learning algorithm is polynomial in the length of the target pattern, the bound for the maximum length of substitution strings, $1/\varepsilon$, $1/\delta$, and $|\mathcal{A}|$. The constant in the running time achieved depends *doubly exponential* on k, and thus, their algorithm becomes rapidly impractical when k increases.

[2] More precisely, the number of allowed unions is at most $poly(|\pi|, s, 1/\varepsilon, 1/\delta, |\mathcal{A}|)$, where π is the target pattern, s the bound on the length on substitution strings, ε and δ are the usual error and confidence parameter, respectively, and $\mathcal{A}$ is the alphabet of constants over which the patterns are defined.

Finally, Lange and Wiehagen [19] have proposed an inconsistent but iterative and conservative algorithm that learns PAT with respect to Pat. We shall study this algorithm below in much more detail.

But before doing it, we aim to figure out under which circumstances iterative learning of PAT is possible at all. A first answer is given by the following theorems from Case *et al.* [9]. Note that Pat is a non-redundant hypothesis space for PAT.

Theorem 1 (Case *et al.* [9]). *Let $\mathcal{C}$ be any concept class, and let $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ be any non-redundant hypothesis space for $\mathcal{C}$. Then, every IIM M that $ItLimTxt_{\mathcal{H}}$-infers $\mathcal{C}$ is conservative.*

Proof. Suppose the converse, i.e., there are a concept $c \in \mathcal{C}$, a text $t = (x_j)_{j\in\mathbb{N}} \in text(c)$, and a $y \in \mathbb{N}$ such that, for $j = M_*(t_y)$ and $k = M_*(t_{y+1}) = M(j, x_{y+1})$, both $j \neq k$ and $t^+_{y+1} \subseteq h_j$ are satisfied. The latter implies $x_{y+1} \in h_j$, and thus we may consider the following text $\tilde{t} \in text(h_j)$. Let $\hat{t} = (\hat{x}_j)_{j\in\mathbb{N}}$ be any text for h_j and let $\tilde{t} = \hat{x}_0, x_{y+1}, \hat{x}_1, x_{y+1}, \hat{x}_2, \ldots$ Since M has to learn h_j from $\tilde{t}$ there must be a $z \in \mathbb{N}$ such that $M_*(\tilde{t}_{z+r}) = j$ for all $r \geq 0$. But $M_*(\tilde{t}_{2z+1}) = M(j, x_{y+1}) = k$, a contradiction. □

Next, we point to another peculiarity of PAT, i.e., it meets the superset condition defined as follows. Let $\mathcal{C}$ be any indexable class. $\mathcal{C}$ meets the *superset condition* if, for all $c, c' \in \mathcal{C}$, there is some $\hat{c} \in \mathcal{C}$ being a superset of both c and c'.

Theorem 2. (Case *et al.* [9]). *Let $\mathcal{C}$ be any indexable class meeting the superset condition, and let $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ be any non-redundant hypothesis space for $\mathcal{C}$. Then, every consistent IIM M that $ItLimTxt_{\mathcal{H}}$-infers $\mathcal{C}$ may be used to decide the inclusion problem for $\mathcal{H}$.*

Proof. Let $\mathcal{X}$ be the underlying learning domain, and let $(w_j)_{j\in\mathbb{N}}$ be an effective enumeration of all elements in $\mathcal{X}$. Then, for every $i \in \mathbb{N}$, $t^i = (x^i_j)_{j\in\mathbb{N}}$ is the following computable text for h_i. Let z be the least index such that $w_z \in h_i$. Recall that, by definition, $h_i \neq \emptyset$, since $\mathcal{H}$ is an indexed family, and thus w_z must exist. Then, for all $j \in \mathbb{N}$, we set $x^i_j = w_j$, if $w_j \in h_i$, and $x^i_j = w_z$, otherwise.

We claim that the following algorithm Inc decides, for all $i, k \in \mathbb{N}$, whether or not $h_i \subseteq h_k$.

Algorithm *Inc*: "On input $i, k \in \mathbb{N}$ do the following:

```
    Determine the least $y \in \mathbb{N}$ with $i = M_*(t^i_y)$. Test whether or not $t^{i,+}_y \subseteq h_k$.
    In case it is, output 'Yes,' and stop. Otherwise, output 'No,' and stop."
```

Clearly, since $\mathcal{H}$ is an indexed family and t^i is a computable text, Inc is an algorithm. Moreover, M learns h_i on every text for it, and $\mathcal{H}$ is a non-redundant hypothesis space. Hence, M has to converge on text t^i to i, and therefore Inc has to terminate.

It remains to verify the correctness of Inc. Let $i, k \in \mathbb{N}$.

Clearly, if Inc outputs 'No,' a string $s \in h_i \setminus h_k$ has been found, and $h_i \not\subseteq h_k$ follows.

Next, consider the case that Inc outputs 'Yes.' Suppose to the contrary that $h_i \not\subseteq h_k$. Then, there is some $s \in h_i \setminus h_k$. Now, consider M when fed the text $t = t^i_y \diamond t^k$. Since $t^{i,+}_y \subseteq h_k$, t is a text for h_k. Since M learns h_k, there is some $r \in \mathbb{N}$ such that $k = M_*(t^i_y \diamond t^k_r)$. By assumption, there are some $\hat{c} \in \mathcal{C}$ with $h_i \cup h_k \subseteq \hat{c}$, and some text $\hat{t}$ for $\hat{c}$ having the initial segment $t^i_y \diamond s \diamond t^k_r$. By Theorem 1, M is conservative. Since $s \in h_i$ and $i = M_*(\hat{t}_y)$, we obtain $M_*(\hat{t}_{y+1}) = M(i, s) = i$. Consequently, $M_*(t^i_y \diamond s \diamond t^k_r) = M_*(t^i_y \diamond t^k_r)$. Finally, since $s \in \hat{t}^+_{y+r+2}$, $k = M_*(t^i_y \diamond t^k_r)$, and $s \notin h_k$, M fails to consistently learn $\hat{c}$ from text $\hat{t}$, a contradiction. This proves the theorem. $\square$

Taking into account that the inclusion problem for Pat is undecidable (cf. Jiang *et al.* [17] and that PAT meets the superset condition, since $L(x_0) = \mathcal{A}^+$, by Theorem 2, we immediately arrive at the following corollary.

Corollary 3 (Case *et al.* [9]). *If an IIM M $ItLimTxt_{Pat}$-learns PAT then M is inconsistent.*

As a matter of fact, the latter corollary generalizes to *all* non-redundant hypothesis spaces for PAT. All the ingredients to prove this can be found in Zeugmann *et al.* [52]. Consequently, if one wishes to learn the pattern languages or unions of pattern languages iteratively, then either redundant hypothesis spaces or inconsistent learners cannot be avoided.

As for unions, the first result goes back to Shinohara [41] who proved the class of all unions of at most two pattern languages to be in $LimTxt_{Pat(2)}$. Wright [49] extended this result to $PAT(k) \in LimTxt_{Pat(k)}$ for all $k \geq 1$. Moreover, Theorem 4.2 in Shinohara and Arimura's [44] together with a lemma from Blum and Blum [6] shows that $\bigcup_{k\in\mathbb{N}} PAT(k)$ is not $LimTxt_{\mathcal{H}}$-inferable for every hypothesis space $\mathcal{H}$.

The iterative learnability of $PAT(k)$ has been established by Case *et al.* [9]. Our learner is also consistent. Thus, the hypothesis space used had to be designed to be *redundant.* We only sketch the proof here.

Theorem 4.

(1) *$Club(L, k)$ is finite for all $L \subseteq \mathcal{A}^+$ and all $k \in \mathbb{N}^+$,*
(2) *If $L \in PAT(k)$, then $Club(L, k)$ is non-empty and contains a set Q, such that $L(Q) = L$.*

Proof. Part (2) is obvious. Part (1) is easy for finite L. For infinite L, it follows from the lemma below.

Lemma 1. *Let $k \in \mathbb{N}^+$, let $L \subseteq \mathcal{A}^+$ be any language, and suppose $t = (s_j)_{j\in\mathbb{N}} \in text(L)$. Then,*

(1) *$Club(t^+_0, k)$ can be obtained effectively from s_0, and $Club(t^+_{n+1}, k)$ is effectively obtainable from $Club(t^+_n, k)$ and s_{n+1} (* note the iterative nature *).*

(2) *The sequence* $\mathit{Club}(t_0^+, k)$, $\mathit{Club}(t_1^+, k)$, $\ldots$ *converges to* $\mathit{Club}(L, k)$.

Putting it all together, one directly gets the following theorem.

THEOREM 5. *For all* $k \geq 1$, $\mathit{PAT}(k) \in \mathit{ItLimTxt}$.

Proof. Let $\mathrm{can}(\cdot)$, be some computable bijection from finite classes of finite sets of patterns onto $\mathbb{N}$. Let pad be a 1–1 padding function such that, for all $x, y \in \mathbb{N}$, $W_{\mathrm{pad}x,y} = W_x$. For a finite class $\mathcal{S}$ of sets of patterns, let $g(\mathcal{S})$ denote a grammar obtained, effectively from $\mathcal{S}$, for $\bigcap_{Q\in\mathcal{S}} L(Q)$.

Let $L \in \mathit{PAT}(k)$, and let $t = (s_j)_{j\in\mathbb{N}} \in \mathit{text}(L)$. The desired IIM M is defined as follows. We set $M_0(t) = M(s_0) = \mathrm{pad}g(\mathit{Club}(t_0^+, k)), \mathrm{can}(\mathit{Club}(t_0^+, k))$, and for all $n > 0$, let

$$\begin{aligned} M_{n+1}(t) &= M(M_n(t), s_{n+1}) \\ &= \mathrm{pad}g(\mathit{Club}(t_{n+1}^+, k)), \mathrm{can}(\mathit{Club}(t_{n+1}^+, k)) \end{aligned}$$

Using Lemma 1 it is easy to verify that $M_{n+1}(t) = M(M_n(t), s_{n+1})$ can be obtained effectively from $M_n(t)$ and s_{n+1}. Therefore, M $\mathit{ItLimTxt}$-identifies $\mathit{PAT}(k)$. □

So far, the general theory provided substantial insight into the iterative learnability of the pattern languages. But still, we do not know anything about the number of examples needed until successful learning and the total amount of time to process them. Therefore, we address this problem in the following subsection.

3.1. *Stochastic Finite Learning*

As we have already mentioned, it does not make much sense to study the worst-case behavior of learning algorithms with respect to their total learning time. The reason for this phenomenon should be clear, since an arbitrary text may provide the information needed for learning very late. Therefore, in the following we always assume a class $\mathcal{D}$ of admissible probability distributions over the relevant learning domain. Ideally, this class should be parameterized. Then, the data fed to learner are generated randomly with respect to one of the probability distributions from the class $\mathcal{D}$ of underlying probability distributions. Furthermore, we introduce a random variable CONV for the stage of convergence. Note that CONV can be also interpreted as the total number of examples read by the IIM M until convergence. The first major step to be performed consists now in determining the expectation $E[\mathit{CONV}]$. Clearly, $E[\mathit{CONV}]$ should be finite for all concepts $c \in \mathcal{C}$ and all distributions $D \in \mathcal{D}$. Second, one has to deal with tail bounds for $E[\mathit{CONV}]$. The easiest way to perform this step is to use Markov's inequality, i.e., we always know that

$$\Pr(\mathit{CONV} \geq t \cdot E[\mathit{CONV}]) \leq \frac{1}{t} \text{ for all } t \in \mathbb{N}^+ .$$

However, quite often one can obtain much better tail bounds. If the underlying learner is known to be *conservative* and *rearrangement-independent* we always get *exponentially* shrinking tail bounds. A learner is said to be rearrangement-independent if its output depends exclusively on the range and length of its input (cf. [21] and the references therein). These tail bounds are established by the following theorem.

Theorem 6. (Rossmanith and Zeugmann [36].) *Let $CONV$ be the sample complexity of a conservative and rearrangement-independent learning algorithm. Then*

$$\Pr(CONV) \geq 2t \cdot E[CONV]) \leq 2^{-t} \text{ for all } t \in \mathbb{N} \ .$$

Theorem 6 puts the importance of rearrangement-independent and conservative learners into the right perspective. As long as the learnability of indexed families is concerned, these results have a wide range of potential applications, since every conservative learner can be transformed into a learner that is both conservative and rearrangement-independent provided the hypothesis space is appropriately chosen (cf. Lange and Zeugmann [21]).

Furthermore, since the distribution of $CONV$ decreases geometrically for all conservative and rearrangement-independent learning algorithms, all higher moments of $CONV$ exist in this case, too. Thus, instead of applying Theorem 6 directly, one can hope for further improvements by applying even sharper tail bounds using for example Chebyshev's inequality.

Additionally, the learner takes a confidence parameter δ as input. But in contrast to learning in the limit, the learner itself decides how many examples it wants to read. Then it computes a hypothesis, outputs it and stops. The hypothesis output is correct for the target with probability at least $1-\delta$.

The explanation given so far explains how it works, but not why it does. Intuitively, the stochastic finite learner simulates the limit learner until an upper bound for twice the expected total number of examples needed until convergence has been met. Assuming this to be true, by Markov's inequality the limit learner has now converged with probability $1/2$. All what is left, is to decrease the probability of failure. This is done by using the tail bounds for $CONV$. Applying Theorem 6, one easily sees that increasing the sample complexity by a factor of $O(\log\frac{1}{\delta})$ results in a probability of $1-\delta$ for having reached the stage of convergence. If Theorem 6 is not applicable, one can still use Markov's inequality but then the sample complexity needed will increase by a factor of $1/\delta$.

It remains to explain how the stochastic finite learner can calculate the upper bound for $E[CONV]$. This is precisely the point where we need the parameterization of the class $\mathcal{D}$ of underlying probability distributions. Since in general, it is not known which distribution from $\mathcal{D}$ has been chosen, one has to assume a bit of *prior knowledge* or *domain knowledge* provided by suitable upper and/or lower bounds for the parameters involved. A more serious difficulty is to incorporate the unknown target concept into this estimate. This step depends on the concrete learning problem on hand, and requires some extra effort. We shall exemplify it below.

Now we are ready to formally define stochastic finite learning.

Definition 3 ([33, 34, 36]). Let $\mathcal{D}$ be a set of probability distributions on the learning domain, $\mathcal{C}$ a concept class, $\mathcal{H}$ a hypothesis space for $\mathcal{C}$, and $\delta \in (0,1)$. $(\mathcal{C},\mathcal{D})$ is said to be *stochastically finitely learnable with δ-confidence* with respect to $\mathcal{H}$ iff there is an IIM M that for every $c \in \mathcal{C}$ and every $D \in \mathcal{D}$ performs as follows. Given any random data sequence θ for c generated according to D, M stops after having seen a finite number of examples and outputs a single hypothesis $h \in \mathcal{H}$. With probability at least $1-\delta$ (with respect to distribution D) h has to be correct, that is $c = h$.

If stochastic finite learning can be achieved with δ-confidence for every $\delta > 0$ then we say that $(\mathcal{C},\mathcal{D})$ can be learned stochastically finite *with high confidence*.

Note that there are subtle differences between our model and PAC learning. By its definition, stochastic finite learning is not completely distribution independent. A bit of *additional knowledge* concerning the underlying probability distributions is required. Thus, from that perspective, stochastic finite learning is weaker than the PAC-model. On the other hand, we do *not* measure the quality of the hypothesis with respect to the underlying probability distribution. Instead, we require the hypothesis computed to be exactly correct with high probability. Note that exact identification with high confidence has been considered within the PAC paradigm, too (cf., e.g., Goldman *et al.* [13]). Conversely, we also can easily relax the requirement to learn *probably exactly correct* but whenever possible we shall not do it.

Furthermore, in the uniform PAC model as introduced in Valiant [46] the sample complexity depends exclusively on the VC dimension of the target concept class and the error and confidence parameters ε and δ, respectively. This model has been generalized by allowing the sample size to depend on the concept complexity, too (cf., e.g., Blumer *et al.* [7] and Haussler *et al.* [15]). Provided no upper bound for the concept complexity of the target concept is given, such PAC learners decide themselves how many examples they wish to read (cf. [15]). This feature is also adopted to our setting of stochastic finite learning. However, all variants of PAC learning we are aware of require that all hypotheses from the relevant hypothesis space are uniformly polynomially evaluable. Though this requirement may be necessary in some cases to achieve (efficient) stochastic finite learning, it is not necessary in general as we shall see below.

Next, let us exemplify our model by looking at the concept class of all pattern languages. The results presented below have been obtained by Zeugmann [50] and Rossmanith and Zeugmann [36]. Our stochastic finite learner uses Lange and Wiehagen's [19] pattern language learner as a main ingredient. We consider here learning from positive data only.

Recall that every string of a particular pattern language is generated by at least one substitution. Therefore, it is convenient to consider probability distributions over the set of all possible substitutions. That is, if $\pi \in Pat_k$, then it suffices to consider any probability distribution D over $\underbrace{\mathcal{A}^+ \times \cdots \times \mathcal{A}^+}_{k-\text{times}}$. For

$(u_0, \dots, u_{k-1}) \in \mathcal{A}^+ \times \cdots \times \mathcal{A}^+$ we denote by $D(u_0, \dots, u_{k-1})$ the probability that variable x_0 is substituted by u_0, variable x_1 is substituted by u_1, $\dots$, and variable x_{k-1} is substituted by u_{k-1}.

In particular, we mainly consider a special class of distributions, i.e., *product distributions*. Let $k \in \mathbb{N}^+$, then the class of all product distributions for Pat_k is defined as follows. For each variable x_j, $0 \le j \le k-1$, we assume an arbitrary probability distribution D_j over $\mathcal{A}^+$ on substitution strings. Then we call $D = D_0 \times \cdots \times D_{k-1}$ product distribution over $\mathcal{A}^+ \times \cdots \times \mathcal{A}^+$, i.e., $D(u_0, \dots, u_{k-1}) = \prod_{j=0}^{k-1} D_j(u_j)$. Moreover, we call a product distribution *regular* if $D_0 = \cdots = D_{k-1}$. Throughout this paper, we restrict ourselves to deal with *regular* distributions. We therefore use d to denote the distribution over $\mathcal{A}^+$ on substitution strings, i.e, $D(u_0, \dots, u_{k-1}) = \prod_{j=0}^{k-1} d(u_j)$. We call a regular distribution *admissible* if $d(a) > 0$ for at least two different elements $a \in \mathcal{A}$. As a special case of an admissible distribution we consider the *uniform* distribution over $\mathcal{A}^+$, i.e., $d(u) = 1/(2 \cdot |\mathcal{A}|)^\ell$ for all strings $u \in \mathcal{A}^+$ with $|u| = \ell$.

We will express all estimates with the help of the following parameters: $E[\Lambda]$, α and β, where Λ is a random variable for the length of the examples drawn. α and β are defined below. To get concrete bounds for a concrete implementation one has to obtain c from the algorithm and has to compute $E[\Lambda]$, α, and β from the admissible probability distribution D. Let $u_0, \dots, u_{k-1}$ be independent random variables with distribution d for substitution strings. Whenever the index i of u_i does not matter, we simply write u or u'.

The two parameters α and β are now defined via d. First, α is simply the probability that u has length 1, i.e.,

$$\alpha = \Pr(|u| = 1) = \sum_{a \in \mathcal{A}} d(a).$$

Second, β is the conditional probability that two random strings that get substituted into π are identical under the condition that both have length 1, i.e.,

$$\beta = \Pr\bigl(u = u' \bigm| |u| = |u'| = 1\bigr) = \sum_{a \in calA} d(a)^2 \Big/ \Bigl(\sum_{a \in \mathcal{A}} d(a)\Bigr)^2.$$

Note that we have *omitted* the assumption of a text to exhaust the target language. Instead, we only demand the data sequence fed to the learner to contain "enough" information to recognize the target pattern. The meaning of "enough" is mainly expressed by the parameter α.

The model of computation as well as the representation of patterns we assume is the same as in Angluin [1]. In particular, we assume a random access machine that performs a reasonable menu of operations each in unit time on registers of length $O(\log n)$ bits, where n is the input length.

Lange and Wiehagen's [19] algorithm (abbr. LWA) works as follows. Let h_n be the hypothesis computed after reading $s_1, \dots, s_n$, i.e., $h_n = M(s_1, \dots, s_n)$.

Then $h_1 = s_1$ and for all $n > 1$:

$$h_n = \begin{cases} h_{n-1}, & \text{if } |h_{n-1}| < |s_n| \\ s_n, & \text{if } |h_{n-1}| > |s_n| \\ h_{n-1} \cup s_n, & \text{if } |h_{n-1}| = |s_n| \end{cases}$$

The algorithm computes the new hypothesis only from the latest example and the old hypothesis. If the latest example is longer than the old hypothesis, the example is ignored, i.e., the hypothesis does not change. If the latest example is shorter than the old hypothesis, the old hypothesis is ignored and the new example becomes the new hypothesis. If, however, $|h_{n-1}| = |s_n|$ the new hypothesis is the *union* of h_{n-1} and s_n . The union $\varrho = \pi \cup s$ of a canonical pattern π and a string s of the same length is defined as

$$\varrho(i) = \begin{cases} \pi(i), & \text{if } \pi(i) = s(i) \\ x_j, & \text{if } \pi(i) \neq s(i)\ \&\ \exists k < i : [\varrho(k) = x_j,\ s(k) = s(i), \\ & \qquad \pi(k) = \pi(i)] \\ x_m, & \text{otherwise, where } m = \#var(\varrho(1) \dots \varrho(i-1)) \end{cases}$$

where $\varrho(0) = \lambda$ for notational convenience. Note that the resulting pattern is again canonical.

If the target pattern does not contain any variable then the LWA converges after having read the first example. Hence, this case is trivial and we therefore assume in the following always $k \geq 1$, i.e., the target pattern has to contain at least one variable. Our next theorem analyzes the complexity of the union operation.

Theorem 7 (Rossmanith and Zeugmann [36]). *The union operation can be computed in linear time.*

Furthermore, the following bound for the stage of convergence for every target pattern from Pat_k can be shown.

Theorem 8 (Rossmanith and Zeugmann [36]).
$E[CONV] = O\left(\frac{1}{\alpha^k} \cdot \log_{1/\beta}(k)\right)$ *for all* $k \geq 2$.

Hence, by Theorem 7, the expected total learning time can be estimated by $E[TT] = O\left(\frac{1}{\alpha^k} E[\Lambda] \log_{1/\beta}(k)\right)$ for all $k \geq 2$.

For a better understanding of the bound obtained we evaluate it for the uniform distribution and compare it to the minimum number of examples needed for learning a pattern language via the LWA.

Theorem 9 (Rossmanith and Zeugmann [36]). $E[TT] = O(2^k |\pi| \log_{|\mathcal{A}|}(k))$ *for the uniform distribution and all* $k \geq 2$.

Theorem 10 (Zeugmann [50]). *To learn a pattern* $\pi \in Pat_k$ *the LWA needs exactly* $\lfloor \log_{|\mathcal{A}|}(|\mathcal{A}| + k - 1) \rfloor + 1$ *examples in the best case.*

The main difference between the two bounds just given is the factor 2^k which precisely reflects the time the LWA has to wait until it has seen the first shortest string from the target pattern language. Moreover, in the best-case the LWA is processing shortest examples only. Thus, we introduce M_C to denote the number of minimum length examples read until convergence. Then, one can show that

$$E[M_C] \le \frac{2\ln(k) + 3}{\ln(1/\beta)} + 2 \ .$$

Note that Theorem 8 is shown by using the bound for $E[M_C]$ just given. More precisely, we have $E[CONV] = (1/\alpha^k)E[M_C]$. Now, we are ready to transform the LWA into a stochastic finite learner.

Theorem 11 (Rossmanith and Zeugmann [36]). *Let* $\alpha_*,\ \beta_* \in (0,1)$. *Assume* $\mathcal{D}$ *to be a class of admissible probability distributions over* $\mathcal{A}^+$ *such that* $\alpha \ge \alpha_*$, $\beta \le \beta_*$ *and* $E[\Lambda]$ *finite for all distributions* $D \in \mathcal{D}$. *Then* $(PAT, \mathcal{D})$ *is stochastically finitely learnable with high confidence from text.*

Proof. Let $D \in \mathcal{D}$, and let $\delta \in (0,1)$ be arbitrarily fixed. Furthermore, let $t = s_1, s_2, s_3, \ldots$ be any randomly generated text with respect to D for the target pattern language. The wanted learner M uses the LWA as a subroutine. Additionally, it has a counter for memorizing the number of examples already seen. Now, we exploit the fact that the LWA produces a sequence $(\tau_n)_{n\in\mathbb{N}^+}$ of hypotheses such that $|\tau_n| \ge |\tau_{n+1}|$ for all $n \in \mathbb{N}^+$.

The learner runs the LWA until for the first time C many examples have been processed, where

$$C = \left(\tfrac{1}{\alpha_*}\right)^{|\tau|} \cdot \left(\frac{2\ln(|\tau|) + 3}{\ln(1/\beta_*)} + 2\right) \qquad (A)$$

and τ is the actual output made by the LWA.

Finally, in order to achieve the desired confidence, the learner sets $\gamma = \lceil \log \frac{1}{\delta} \rceil$ and runs the LWA for a total of $2 \cdot \gamma \cdot C$ examples. This is the reason we need the counter for the number of examples processed. Now, it outputs the last hypothesis τ produced by the LWA, and stops thereafter.

Clearly, the learner described above is finite. Let L be the target language and let $\pi \in Pat_k$ be the unique pattern such that $L = L(\pi)$. It remains to argue that $L(\pi) = L(\tau)$ with probability at least $1 - \delta$.

First, the bound in (A) is an upper bound for the expected number of examples needed for convergence by the LWA that has been established in Theorem 8 (via the reformulation using $E[M_C]$ given above). On the one hand, this follows from our assumptions about the allowed α and β as well as from the fact that $|\tau| \ge |\pi|$ for every hypothesis output. On the other hand, the learner does not know k, but the estimate $\#var(\pi) \le |\pi|$ is sufficient. Note that we have to use in (A) the bound for $E[M_C]$ given above, since the target pattern may contain zero or one different variables.

Therefore, after having processed C many examples the LWA has already converged on average. The desired confidence is then an immediate consequence of Corollary 6. $\square$

The latter theorem allows a nice corollary which we state next. Making the same assumption as done by Kearns and Pitt [18], i.e., assuming the additional prior knowledge that the target pattern belongs to Pat_k, the complexity of the stochastic finite learner given above can be considerably improved. The resulting learning time is linear in the expected string length, and the constant depending on k grows only exponentially in k in contrast to the doubly exponentially growing constant in Kearns and Pitt's [18] algorithm. Moreover, in contrast to their learner, our algorithm learns from positive data only, and outputs a hypothesis that is correct for the target language with high probability.

Again, for the sake of presentation we shall assume $k \geq 2$. Moreover, if the prior knowledge $k = 1$ is available, then there is also a much better stochastic finite learner for PAT_1 (cf. [34]).

Corollary 12. *Let $\alpha_*, \beta_* \in (0,1)$. Assume $\mathcal{D}$ to be a class of admissible probability distributions over $\mathcal{A}^+$ such that $\alpha \geq \alpha_*$, $\beta \leq \beta_*$ and $E[\Lambda]$ finite for all distributions $D \in \mathcal{D}$. Furthermore, let $k \geq 2$ be arbitrarily fixed. Then there exists a learner M such that*

(1) *M learns $(PAT_k, \mathcal{D})$ stochastically finitely with high confidence from text, and*

(2) *The running time of M is $O\big(\hat{\alpha}_*^k E[\Lambda] \log_{1/\beta_*}(k) \log_2(1/\delta)\big)$.*
(Note that $\hat{\alpha}_*^k$ and $\log_{1/\beta_*}(k)$ now are constants. *)*

4. Conclusions

The present paper surveyed results recently obtained concerning the iterative learnability of the class of all pattern languages and finite unions thereof. In particular, it could be shown that there are strong dependencies between iterative learning, the class of admissible hypothesis spaces and additional requirements to the learner such as consistency, conservativeness and the decidability of the inclusion problem for the hypothesis space chosen. Looking at these results, we have seen that the LWA is in some sense optimal.

Moreover, by analyzing the average-case behavior of Lange and Wiehagen's pattern language learning algorithm with respect to its total learning time and by establishing exponentially shrinking tail bounds for a rather rich class of limit learners, we have been able to transform the LWA into a stochastic finite learner. The price paid is the incorporation of a bit prior knowledge concerning the class of underlying probability distributions. When applied to the class of all k-variable pattern languages, where k is *a priori* known, the resulting total learning time is linear in the expected string length.

Thus, the present paper provides evidence that analyzing the average-case behavior of limit learners with respect to their total learning time may be considered as a promising path towards a new theory of efficient algorithmic learning. Recently obtained results along the same path as outlined in Erlebach *et al.*[11] as well as in Reischuk and Zeugmann [32, 34] provide further support for the fruitfulness of this approach.

In particular, in Reischuk and Zeugmann [32, 34] we have shown that one-variable pattern languages are learnable for basically all meaningful distributions within an optimal linear total learning time on the average. Furthermore, this learner can also be modified to maintain the *incremental* behavior of Lange and Wiehagen's [19] algorithm. Instead of memorizing the pair $(\mathtt{PRE}, \mathtt{SUF})$, it can also store just the two or three examples from which the prefix $\mathtt{PRE}$ and the suffix $\mathtt{SUF}$ of the target pattern has been computed. While it is no longer *iterative*, it is still a *bounded example memory* learner. A bounded example memory learner is essentially an iterative learner that is additionally allowed to memorize an *a priori* bounded number of examples (cf. [9] for a formal definition).

While the one-variable pattern language learner from [34] is highly practical, our stochastic finite learner for the class of all pattern languages is still not good enough for practical purposes. But our results surveyed point to possible directions for potential improvements. However, much more effort seems necessary to design a stochastic finite learner for $PAT(k)$.

Additionally, we have applied our techniques to design a stochastic finite learner for the class of all concepts describable by a monomial which is based on Haussler's [14] Wholist algorithm. Here we have assumed the examples to be binomially distributed. The sample size of our stochastic finite learner is mainly bounded by $\log(1/\delta)\log n$, where δ is again the confidence parameter and n is the dimension of the underlying Boolean learning domain. Thus, the bound obtained is exponentially better than the bound provided within the PAC model.

Our approach also differs from U-learnability introduced by Muggleton [27]. First of all, our learner is fed with positive examples only, while in Muggleton's [27] model examples labeled with respect to their containment in the target language are provided. Next, we do not make any assumption concerning the distribution of the target patterns. Furthermore, we do *not* measure the expected total learning time with respect to a given class of distributions over the targets and a given class of distributions for the sampling process, but exclusively in dependence on the length of the target. Finally, we require exact learning and not approximately correct learning.

References

1. D. Angluin, Finding Patterns common to a Set of Strings, *Journal of Computer and System Sciences* **21**, 1980, 46–62.

2. D. Angluin, Inductive inference of formal languages from positive data, *Information and Control* **45**, 1980, 117–135.
3. D. Angluin and C.H. Smith. Inductive inference: Theory and methods. *Computing Surveys* **15**, No. 3, 1983, 237 - 269.
4. D. Angluin and C.H. Smith. Formal inductive inference. "Encyclopedia of Artificial Intelligence" (St.C. Shapiro, Ed.), Vol. 1, pp. 409 - 418, Wiley-Interscience Publication, New York.
5. S. Arikawa, T. Shinohara and A. Yamamoto, Learning elementary formal systems, *Theoretical Computer Science* **95**, 97–113, 1992.
6. L. Blum and M. Blum, Toward a mathematical theory of inductive inference, *Information and Control* **28**, 125–155, 1975.
7. A. Blumer, A. Ehrenfeucht, D. Haussler and M. Warmuth, Learnability and the Vapnik-Chervonenkis Dimension, *Journal of the ACM* **36** (1989), 929–965.
8. I. Bratko and S. Muggleton, Applications of inductive logic programming, *Communications of the ACM*, 1995.
9. J. Case, S. Jain, S. Lange and T. Zeugmann, Incremental Concept Learning for Bounded Data Mining, *Information and Computation* **152**, No. 1, 1999, 74–110.
10. R. Daley and C.H. Smith. On the Complexity of Inductive Inference. *Information and Control* **69** (1986), 12–40.
11. T. Erlebach, P. Rossmanith, H. Stadtherr, A. Steger and T. Zeugmann, Learning one-variable pattern languages very efficiently on average, in parallel, and by asking queries, *Theoretical Computer Science* **261**, No. 1-2, 2001, 119–156.
12. E.M. Gold, Language identification in the limit, *Information and Control* **10** (1967), 447–474.
13. S.A. Goldman, M.J. Kearns and R.E. Schapire, Exact identification of circuits using fixed points of amplification functions. *SIAM Journal of Computing* **22**, 1993, 705–726.
14. D. Haussler, Bias, version spaces and Valiant's learning framework, "Proc. 8th National Conference on Artificial Intelligence," pp. 564–569, San Mateo, CA: Morgan Kaufmann, 1987.
15. D. Haussler, M. Kearns, N. Littlestone and M.K. Warmuth, Equivalence of models for polynomial learnability. *Information and Computation* **95** (1991), 129–161.
16. S. Jain, D. Osherson, J.S. Royer and A. Sharma, "Systems That Learn: An Introduction to Learning Theory," MIT-Press, Boston, Massachusetts, 1999.
17. T. Jiang, A. Salomaa, K. Salomaa and S. Yu, Inclusion is undecidable for pattern languages, *in* "Proceedings 20th International Colloquium on Automata, Languages and Programming," (A. Lingas, R. Karlsson, and S. Carlsson, Eds.), Lecture Notes in Computer Science, Vol. 700, pp. 301–312, Springer-Verlag, Berlin, 1993.
18. M. Kearns L. Pitt, A polynomial-time algorithm for learning k–variable pattern languages from examples. *in* "Proc. Second Annual ACM Workshop on Computational Learning Theory" (pp. 57–71). San Mateo, CA: Morgan Kaufmann, 1989.
19. S. Lange and R. Wiehagen, Polynomial-time inference of arbitrary pattern languages. *New Generation Computing* **8** (1991), 361–370.
20. S. Lange and T. Zeugmann, Language learning in dependence on the space of hypotheses. *in* "Proc. of the 6th Annual ACM Conference on Computational Learning Theory," (L. Pitt, Ed.), pp. 127–136, ACM Press, New York, 1993.
21. S. Lange and T. Zeugmann, Set-driven and Rearrangement-independent Learning of Recursive Languages, *Mathematical Systems Theory* **29** (1996), 599–634.
22. S. Lange and T. Zeugmann, Incremental Learning from Positive Data, *Journal of Computer and System Sciences* **53**(1996), 88–103.

23. N. Lavrač and S. Džeroski, "Inductive Logic Programming: Techniques and Applications," Ellis Horwood, 1994.
24. T. Mitchell. "Machine Learning," McGraw Hill, 1997.
25. A. Mitchell, A. Sharma, T. Scheffer and F. Stephan, The VC-dimension of Subclasses of Pattern Languages, *in* "Proc. 10th International Conference on Algorithmic Learning Theory," (O. Watanabe and T. Yokomori, Eds.), Lecture Notes in Artificial Intelligence, Vol. 1720, pp. 93 - 105, Springer-Verlag, Berlin, 1999.
26. S. Miyano, A. Shinohara and T. Shinohara, Polynomial-time learning of elementary formal systems, *New Generation Computing*, 18:217–242, 2000.
27. S. Muggleton, Bayesian Inductive Logic Programming, *in* "Proc. 7th Annual ACM Conference on Computational Learning Theory" (M. Warmuth, Ed.), pp. 3–11, ACM Press, New York, 1994.
28. S. Muggleton and L. De Raedt, Inductive logic programming: Theory and methods, *Journal of Logic Programming*, 19/20:669–679, 1994.
29. R.P. Nix, Editing by examples, Yale University, Dept. Computer Science, Technical Report 280, 1983.
30. D.N. Osherson, M. Stob and S. Weinstein, "Systems that Learn, An Introduction to Learning Theory for Cognitive and Computer Scientists," MIT-Press, Cambridge, Massachusetts, 1986.
31. L. Pitt, Inductive Inference, DFAs and Computational Complexity, *in* "Proc. 2nd Int. Workshop on Analogical and Inductive Inference" (K.P. Jantke, Ed.), Lecture Notes in Artificial Intelligence, Vol. 397, pp. 18–44, Springer-Verlag, Berlin, 1989.
32. R. Reischuk and T. Zeugmann, *Learning One- Variable Pattern Languages in Linear Average Time, in* "Proc. 11th Annual Conference on Computational Learning Theory - COLT'98," July 24th - 26th, Madison, pp. 198–208, ACM Press 1998.
33. R. Reischuk and T. Zeugmann, A Complete and Tight Average-Case Analysis of Learning Monomials, *in* "Proc. 16th International Symposium on Theoretical Aspects of Computer Science," (C. Meinel and S. Tison, Eds.), Lecture Notes in Computer Science, Vol. 1563, pp. 414–423, Springer-Verlag , Berlin 1999.
34. R. Reischuk and T. Zeugmann, An Average-Case Optimal One-Variable Pattern Language Learner, *Journal of Computer and System Sciences* **60**, No. 2, 2000, 302–335.
35. H. Rogers, Jr., "Theory of Recursive Functions and Effective Computability," McGraw–Hill, New York, 1967.
36. P. Rossmanith and T. Zeugmann. Stochastic Finite Learning of the Pattern Languages, *Machine Learning* **44**, No. 1-2, 2001, 67–91.
37. Patterns (The Formal Language Theory Column), EATCS Bulletin **54**, 46–62, 1994.
38. Return to patterns (The Formal Language Theory Column), EATCS Bulletin **55**, 144–157, 1994.
39. S. Shimozono, A. Shinohara, T. Shinohara, S. Miyano, S. Kuhara and S. Arikawa, Knowledge acquisition from amino acid sequences by machine learning system BONSAI, *Trans. Information Processing Society of Japan* **35**, 2009–2018, 1994.
40. R.E. Schapire, Pattern languages are not learnable, In M.A. Fulk & J. Case (Eds.), *Proceedings of the Third Annual ACM Workshop on Computational Learning Theory*, (pp. 122–129). San Mateo, CA: Morgan Kaufmann, (1990).
41. T. Shinohara, Inferring unions of two pattern languages, *Bulletin of Informatics and Cybernetics* **20**, 83–88, 1983.
42. T. Shinohara, Inductive inference of monotonic formal systems from positive data, *New Generation Computing* **8**, 371–384, 1991.

43. T.Shinohara and S. Arikawa, Pattern inference, *in* "Algorithmic Learning for Knowledge-Based Systems," (K.P. Jantke and S. Lange, Eds.), Lecture Notes in Artificial Intelligence, Vol. 961, pp. 259–291, Springer-Verlag, Berlin, 1995.
44. T. Shinohara and H. Arimura, Inductive inference of unbounded unions of pattern languages from positive data, *in* "Proceedings 7th International Workshop on Algorithmic Learning Theory," (S. Arikawa and A.K. Sharma, Eds.), Lecture Notes in Artificial Intelligence, Vol. 1160, pp. 256–271, Springer-Verlag, Berlin, 1996.
45. R. Smullyan, "Theory of Formal Systems," Annals of Mathematical Studies, No. 47. Princeton, NJ, 1961.
46. L.G. Valiant, A Theory of the Learnable, *Communications of the ACM* **27** (1984), 1134–1142.
47. R. Wiehagen. Limes-Erkennung rekursiver Funktionen durch spezielle Strategien. *Journal of Information Processing and Cybernetics (EIK)* **12**, 1976, 93–99.
48. R. Wiehagen and T. Zeugmann, Ignoring Data may be the only Way to Learn Efficiently, *Journal of Experimental and Theoretical Artificial Intelligence* **6** (1994), 131–144.
49. K. Wright, Identification of unions of languages drawn from an identifiable class, *in* "Proceedings of the 2nd Workshop on Computational Learning Theory," (R. Rivest, D. Haussler, and M. Warmuth, Eds.), pp. 328–333, San Mateo, CA: Morgan Kaufmann, 1989.
50. T. Zeugmann, Lange and Wiehagen's Pattern Language Learning Algorithm: An Average-case Analysis with respect to its Total Learning Time, *Annals of Mathematics and Artificial Intelligence* **23**, No. 1-2, 1998, 117–145.
51. T. Zeugmann and S. Lange, A guided tour across the boundaries of learning recursive languages, *in* "Algorithmic Learning for Knowledge-Based Systems," (K.P. Jantke and S. Lange, Eds.), Lecture Notes in Artificial Intelligence, Vol. 961, pp. 190–258, Springer-Verlag, Berlin, 1995.
52. T. Zeugmann, S. Lange and S. Kapur, Characterizations of monotonic and dual monotonic language learning, *Information and Computation* **120**, 155–173, 1995.