Stochastic Finite Learning

Thomas Zeugmann [1]

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

Abstract. Recently, we have a developed a learning model, called *stochastic finite learning*, that makes a connection between concepts from PAC learning and inductive inference learning models.
The motivation for this work is as follows. Within Gold's (1967) model of learning in the limit many important learning problems can be formalized and it can be shown that they are algorithmically solvable *in principle*. However, since a limit learner is only supposed to converge, one never knows at any particular learning stage whether or not it has already been successful. Such an uncertainty may be not acceptable in many applications.
The present paper surveys the new approach to overcome this uncertainty that potentially has a wide range of applicability.

1. Introduction

Let us assume that we have to deal with the following problem. We are given a concept class $\mathcal{C}$ and should design a learner for it. Next, suppose we already know or could prove $\mathcal{C}$ not to be PAC learnable. But it can be shown that $\mathcal{C}$ is learnable within Gold's [6] model of learning in the limit. Consequently, there is a learner behaving as follows. When fed any of the data sequences allowed in this model, it converges in the limit to a hypothesis correctly describing the target concept. Nothing more is known. Since it is either undecidable whether or not the sequence of hypotheses has already converged or, even if it decidable, it practically infeasible to do so, one never knows whether or not the learner has already converged. But such an *uncertainty* may not be tolerable in many applications. So, how can we recover?

In general, there may be no way to overcome this uncertainty at all, or at least not efficiently. But if the learner satisfies some additional assumptions outlined below then there is a rather general way to transform learning in the limit into *stochastic finite learning*. Within this survey, we aim to describe our method developed and to exemplify it by using some well-known concept classes.

It should also be noted that our approach may be beneficial even in case that the considered concept class is PAC learnable.

2. Learning in the Limit

Gold's [6] 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 $\text{range}(t) =_{df} \{x_j \mid j \in \mathbb{N}\} = c$. Then t is said to be a *positive presentation* or, synonymously, a *text* for c. By $text(c)$ 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 $t_y = x_0, \dots, x_y$, 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.

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 LIM we denote the collection of all concepts classes $\mathcal{C}$ that are learnable in the limit from text. Note that instead of LIM sometimes EX is used.

It can be shown that many interesting concepts classes are learnable in the limit from text and informant, respectively. However, it should be noted that there are also some major *points of concern*. First, in the definition given above, the limit learner has always access to the whole actual initial segment of the data sequence provided. Clearly, in practice such a learner would run out of space pretty soon. This problem can be overcome be considering *incremental* learning algorithms. We do not develop this topic here any further. Instead, the reader is referred to [3, 13].

Second, since a limit learner is only supposed to converge, one never knows at any particular learning stage whether or not it has already been successful. Such an uncertainty may be not acceptable in many applications. Clearly, one could modify the definition of learning in the limit by adding the requirement that convergence should be *decidable*. Then one would arrive at finite learning (cf. Gold [6]). However, the power of finite learning is much smaller than that of learning in the limit. Finally, it seems difficult to define an appropriate measure for the complexity of learning in the limit (cf. Pitt [15]).

Note that the latter points of concern are really serious, since a major task of algorithmic learning theory consists in providing *performance guarantees*. Thus, it is only natural to ask why we not simply refrain to consider learning the limit. Alternatively, we could try to use the well-known PAC model (cf. [20]). Our answer is twofold. During the last decade many interesting concept classes have been proven to be *not* PAC learnable. Moreover, even if a concept class is PAC learnable, the performance guarantees provided by this model are often much too pessimistic. So, how can we recover?

Though in general there may be no way to recover, we recently discovered a rather general method that allows one to overcome the difficulties described above. In the remaining part of this paper, we outline this approach. The key ingredient is an *average-case* analysis of the learning algorithms considered as initiated by Zeugmann [22]. As it turned out, the most promising complexity measure to be considered is the so called *total learning time*, i.e., the overall time taken by a learner until convergence. Formally, the total learning time is defined as follows (cf. Daley and Smith [4]). 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. [6]). 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 language from text t then the total learning time is *infinite*.

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. Clearly, if this viewpoint is adopted, then our definitions of learning and of the total learning time seem inappropriate. Note however, that there may be no way at all to decide whether or not the current hypothesis is not correct for the latest piece of data received. But even if one can decide whether or not the latest piece of data obtained is correctly reflected by the current hypothesis, such a test may be computationally infeasible. For example, consider the well-known class of all pattern languages (cf. Angluin [1]). The membership problem for the pattern languages is $\mathcal{NP}$-complete. Thus, directly testing consistency would immediately lead to a non-polynomial update time unless $\mathcal{P} = \mathcal{NP}$. Moreover, at least for the informant case there is no polynomial time update learning algorithm at all that infers the whole class of pattern languages (cf. [21]). Thus, adopting the definition of the total learning time given above is reasonable.

Next, one has to assume an appropriate class of probability distributions $\mathcal{D}$ over the relevant learning domain. Ideally, this class should be parameterized. In what follows, it is assumed that the data sequences are drawn at random with respect to any of the distributions from $\mathcal{D}$. Next, it is advantageous to 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 of the average-case analysis consists now in determining the expectation $E[CONV]$. Clearly, $E[CONV]$ should be finite for all concepts $c \in \mathcal{C}$ and all distributions $D \in \mathcal{D}$.

Next, Markov's inequality provides us with the following tail bounds:

$$\Pr(CONV \geq t \cdot E[CONV]) \leq \frac{1}{t} .$$

Having reached this stage, learning in the limit can be transformed into *stochastic finite learning.*

3. Stochastic Finite Learning

A stochastic finite learner is successively fed data about the target concept. Note that these data are generated randomly with respect to one of the probability distributions from the class of underlying probability distributions. 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 $1 - \delta$.

Some more remarks are mandatory here. The description given above 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 can be done by using again Markov's inequality, i.e., increasing the sample complexity by a factor of $1/\delta$ results in a confidence of $1-\delta$ for having reached the stage of convergence.

Finally, 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 know 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 point to some examples at the end of this section.

Next, we formally define stochastic finite learning.

DEFINITION 2. 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 our model of stochastic finite learning differs in several ways from the PAC model. First, it is not completely distribution independent. Thus, from that perspective, this variant is weaker than the PAC-model. But the hypothesis computed is exactly correct with high probability. That is, we do *not* measure the quality of the hypothesis with respect to the underlying probability distribution. For seeing the difference, consider a target concept class $\mathcal{C}$ over some fixed learning domain $\mathcal{X}$ containing the concept $\mathcal{X}$. Furthermore, suppose $\mathcal{X}$ to be learnable from positive data only. Thus, it is reasonable to assume that all negative examples have probability 0. Consequently, a PAC learner could simply output a finite description for $\mathcal{X}$ independent of the particular $c \in \mathcal{C}$ to be learned. It would even learn from 0 examples without any error (where error is measured as usual in the PAC model). But this may be hardly what one really wants to learn. In contrast, a stochastic finite learner would always output a hypothesis h such that $c = h$ with any desired probability. Last but not least, in the PAC 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 (cf. [2]). In contrast, a stochastic finite learner decides itself how many examples it wishes to read.

Finally, if the limit learner M is additionally known to be *conservative* and *rearrangement independent* much more efficient stochastic finite learners are available. A learner is said to be rearrangement independent if its output depends exclusively on the range and length of its input (cf. [12] and the references therein). Furthermore, a learner is conservative if it exclusively performs mind changes that can be justified by an inconsistency of the abandoned hypothesis with the data received so far (cf. [1] for a formal definition). In this case, we could even prove *exponentially* shrinking tail bounds for $E[CONV]$.

THEOREM 1. (ROSSMANITH AND ZEUGMANN [19].) *Let $CONV$ be the sample complexity of a conservative and rearrangement-independent learning algorithm. Then*

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

Having this theorem, the sample complexity of the resulting stochastic finite learner is only by a factor of $\log(1/\delta)$ larger than that of the original limit learner. Consequently, Theorem 1 puts the importance of conservative and rearrangement-independent 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 [12]).

Since the distribution of $CONV$ decreases geometrically, all higher moments of $CONV$ exists, too. Thus, instead of applying Theorem 1 directly, one can hope for further improvements by applying even sharper tail bounds using for example Chebyshev's inequality.

Our model of stochastic finite learning differs to a certain extent from the PAC model. First, it is *not* completely distribution independent, since 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. But with respect to the quality of its hypotheses, it is stronger than the PAC model by requiring the output to be *probably exactly correct* rather than *probably approximately correct*. That is, we do *not* measure the quality of the hypothesis with respect to the underlying probability distribution. Note that exact identification with high confidence has been considered within the PAC paradigm, too (cf., e.g., Goldman *et al.* [7]).

Furthermore, in the uniform PAC model as introduced in Valiant [20] 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.* [2] and Haussler *et al.* [9]). Provided no upper bound for the concept complexity of the target concept is given, such PAC learners decide themselves how many examples they wishes to read (cf. [9]). 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.

4. Learning Conjunctive Concepts

In this section we exemplify the general approach outlined above by using the class of all concepts describable by a monomial. For all details omitted the reader is referred to Reischuk and Zeugmann [17].

To define the classes of concepts we deal with in this paper let $\mathcal{L}_n = \{x_1, \bar{x}_1, x_2, \bar{x}_2 \dots, x_n, \bar{x}_n\}$ be a set of literals. x_i is a *positive* literal and $\bar{x}_i$ a *negative* one. A conjunction of literals defines a ***monomial***. For a monomial m let $\#(m)$ denote its length, that is the number of literals in it.

m describes a subset $L(m)$ of $\mathcal{X}_n$, in other words a concept, in the obvious way: the concept contains exactly those binary vectors for which the monomial evaluates to 1, that is $L(m) := \{b \in \mathcal{X}_n \mid m(b) = 1\}$. The collection of objects we are going to learn is the set $\mathcal{C}_n$ of all concepts that are describable by monomials over $\mathcal{X}_n$. There are two trivial concepts, the empty subset and $\mathcal{X}_n$ itself. $\mathcal{X}_n$, which will also be called "TRUE", can be represented by the empty monomial. The concept "FALSE" has several descriptions. To avoid ambiguity, we always represent "FALSE" by the monomial $x_1\bar{x}_1 \dots x_n\bar{x}_n$. Furthermore, we often identify the set of all monomials over $\mathcal{L}_n$ and the concept class $\mathcal{C}_n$. Note that $|\mathcal{C}_n| = 3^n + 1$.

For the concept class $\mathcal{C}_n$ we choose as hypothesis space the set of all monomials over $\mathcal{L}_n$. We shall distinguish learning from positive data only and learning from both positive and negative data. Note that when considering learning from positive data only, one cannot decide whether or not the learner has already converged. When learning from positive and negative data is considered, the stage of convergence is *decidable*, but one would have to read the data sequence until all Boolean vectors did appear. Thus, for any interesting n decidability is practically infeasible.

The learner used is Haussler's [8] Wholist algorithm. Note that this learner computes a new hypothesis by using only the most recent example received and his old hypothesis. Such learners are called *iterative* (cf. [13]). For the sake of completeness, we include the Wholist learning algorithm here. By $data(c)$ we denote any data sequence for the concept c.

Algorithm $\mathcal{P}$**:** On input sequence $\langle (b_1, c(b_1)), (b_2, c(b_2)), \dots \rangle$ do the following:

Initialize $h_0 := x_1\bar{x}_1 \dots x_n\bar{x}_n$.
for $i = 1, 2, \dots$ **do**
let h_{i-1} denote $\mathcal{P}$'s internal hypothesis produced before receiving $(b_i, c(b_i))$;
when receiving $(b_i, c(b_i))$ test whether or not $h_{i-1}(b_i) = c(b_i)$.
if $h_{i-1}(b_i) = c(b_i)$ **then** set $h_i := h_{i-1}$ and output h_i.
  else for $j := 1$ **to** n **do**
    if $b_i^j = 1$ **then** delete $\bar{x}_j$ in h_{i-1} **else** delete x_j in h_{i-1};
  let h_i be the resulting monomial and output h_i.
end.

Now, it is easy to see that Algorithm $\mathcal{P}$ learns the set of all monomials in the limit. If the target monomial is the concept "FALSE", then Algorithm $\mathcal{P}$ immediately converges. Thus, we call "FALSE" the minimal concept. If the target concept contains precisely n literals, then one positive example suffices. This positive example is unique.

In the general case, we call the literals in a monomial m ***relevant***. All other literals in $\mathcal{L}_n$ are said to be ***irrelevant*** for m. There are $2n - \#(m)$ irrelevant literals. We call bit i **relevant** for m if x_i or $\bar{x}_i$ is relevant for m. By $\boldsymbol{k} = \boldsymbol{k(m)} = n - \#(m)$ we denote the number of irrelevant bits.

First, we consider learning from positive data. To avoid the trivial cases, we let $c = L(m) \in \mathcal{C}_n$ be a concept with monomial $m = \bigwedge_{j=1}^{\#(m)} \ell_{i_j}$ such that $k = k(m) = n - \#(m) > 0$. There are 2^k positive examples for c. For the sake of presentation, we assume these examples to be ***binomially distributed***. That is, in a random positive example all entries corresponding to irrelevant bits are selected independently of each other. With some probability p this will be a 1, and with probability $q := 1 - p$ a 0. We shall consider only nontrivial distributions where $0 < p < 1$. Note that otherwise the data sequence does not contain all positive examples. We aim to compute the expected number of examples taken by $\mathcal{P}$ until convergence. Again we use $CONV$ to denote a random variable counting the number of examples read till convergence.

The first example received forces $\mathcal{P}$ to delete precisely n of the $2n$ literals in h_0. Thus, this example always plays a *special* role. Note that the resulting hypothesis h_1 depends on b_1, but the number k of literals that remain to be deleted from h_1 until convergence is *independent* of b_1. Using tail bound techniques, we can show the following theorem.

THEOREM 2. *Let $c = L(m)$ be a non-minimal concept in $\mathcal{C}_n$, and let the positive examples for c be binomially distributed with parameter p. Define $\psi := \min\{\frac{1}{1-p}, \frac{1}{p}\}$. Then the expected number of positive examples needed by algorithm $\mathcal{P}$ until convergence can be bounded by $E[CONV] \le \lceil \log_\psi k(m) \rceil + 3$.*

Proof. Let $k := k(m)$. The first positive example contains ν times a 1 and $k - \nu$ many 0 with probability $\binom{k}{\nu} p^\nu q^{k-\nu}$ at the positions not corresponding to a literal in the target monomial m. Now, assuming any such vector, we easily see that h_1 contains ν positive irrelevant literals and $k - \nu$ negative literals. Therefore, in order to achieve convergence, the algorithm $\mathcal{P}$ now needs positive examples that contain at least one 0 for each positive irrelevant literal and at least one 1 for each negative irrelevant literal. Thus, the probability that at least one irrelevant literal survives μ subsequent positive examples is bounded by $\nu q^\mu + (k-\nu)p^\mu$. Therefore,

$$\Pr[CONV - 1 > \mu] \le \sum_{\nu=0}^{k} \binom{k}{\nu} p^\nu q^{k-\nu} \cdot \left(\nu q^\mu + (k-\nu)p^\mu\right) .$$

Next, we derive a closed formula for the sum given above.

Claim 1. $\sum_{\nu=0}^{k} \binom{k}{\nu} p^{\nu} q^{k-\nu} \cdot \nu = kp$ and $\sum_{\nu=0}^{k} \binom{k}{\nu} p^{\nu} q^{k-\nu} \cdot (k-\nu) = kq$

The first equality can be shown as follows.

$$\begin{aligned}
\sum_{\nu=0}^{k} \binom{k}{\nu} p^{\nu} q^{k-\nu} \cdot \nu &= \sum_{\nu=1}^{k} \binom{k}{\nu} p^{\nu} q^{k-\nu} \cdot \nu = \sum_{\nu=0}^{k-1} \binom{k}{\nu+1} p^{\nu+1} q^{k-1-\nu} \cdot (\nu+1) \\
&= \sum_{\nu=0}^{k-1} k \cdot \binom{k-1}{\nu} p^{\nu+1} q^{k-(\nu+1)} \\
&= k\, p \cdot \sum_{\nu=0}^{k-1} \binom{k-1}{\nu} p^{\nu} q^{(k-1)-\nu} \\
&= k\, p \cdot (p+q)^{k-1} = k\, p\, .
\end{aligned}$$

The other equality can be proved analogously, which yields Claim 1.

Now, proceeding as above, we obtain

$$\begin{aligned}
E[CONV - 1\,] &= \sum_{\mu=0}^{\infty} \Pr[CONV - 1 > \mu] \\
&\le \lambda + \sum_{\mu=\lambda}^{\infty} \sum_{\nu=0}^{k} \binom{k}{\nu} p^{\nu} q^{k-\nu} \cdot \bigl(\nu q^{\mu} + (k-\nu) p^{\mu}\bigr) \\
&= \lambda + \sum_{\mu=\lambda}^{\infty} \sum_{\nu=0}^{k} \binom{k}{\nu} p^{\nu} q^{k-\nu} \cdot \nu q^{\mu} + \sum_{\mu=\lambda}^{\infty} \sum_{\nu=0}^{k} \binom{k}{\nu} p^{\nu} q^{k-\nu} \cdot (k-\nu) p^{\mu} \\
&= \lambda + \sum_{\mu=\lambda}^{\infty} q^{\mu} \cdot \underbrace{\sum_{\nu=0}^{k} \binom{k}{\nu} p^{\nu} q^{k-\nu} \nu}_{=kp \quad \text{by Claim 1}} + \sum_{\mu=\lambda}^{\infty} p^{\mu} \cdot \underbrace{\sum_{\nu=0}^{k} \binom{k}{\nu} p^{\nu} q^{k-\nu} \cdot (k-\nu)}_{=kq \quad \text{by Claim 1}} \\
&= \lambda + kp \cdot \sum_{\mu=\lambda}^{\infty} q^{\mu} + kq \cdot \sum_{\mu=\lambda}^{\infty} p^{\mu} = \lambda + k \cdot [q^{\lambda} + p^{\lambda}] \\
&\le \lambda + k\, 2\, \psi^{-\lambda}\, .
\end{aligned}$$

Finally, choosing $\lambda = \lceil \log_{\psi} k \rceil$ gives the statement of the theorem. ∎

Now, we can easily determine the expected total learning time.

Corollary 3. *For every binomially distributed text with parameter* $0 < p < 1$ *the average total learning time of algorithm* $\mathcal{P}$ *for concepts in* $\mathcal{C}_n$ *with* μ *literals is at most* $O(n(\log(n-\mu+2))$.

The Wholist algorithm possesses the two favorable properties needed to apply Theorem 1, i.e., it is rearrangement-independent and conservative. Thus, we can conclude

$\Pr[CONV > 2t \cdot E[CONV] \le 2^{-t}$ for all $t \in \mathbb{N}$.

Next, we turn our attention to the design of a stochastic finite learner. We study the case that the positive examples are binomially distributed with parameter p. But we do not require precise knowledge about the underlying distribution. Instead, we reasonably assume that *prior knowledge* is provided by parameters p_{low} and p_{up} such that $p_{low} \le p \le p_{up}$ for the true parameter p. Binomial distributions fulfilling this requirement are called $\boldsymbol{(p_{low}, p_{up})}$ ***–admissible distributions***. Let $\boldsymbol{\mathcal{D}_n[p_{low}, p_{up}]}$ denote the set of such distributions on $\mathcal{X}_n$.

If bounds p_{low} and p_{up} are available, the Wholist algorithm can be transformed into a stochastic finite learner inferring all concepts with high confidence.

THEOREM 4. *Let* $0 < p_{low} \le p_{up} < 1$ *and* $\psi := \min\{\frac{1}{1-p_{low}}, \frac{1}{p_{up}}\}$. *Then* $(\mathcal{C}_n, \mathcal{D}_n[p_{low}, p_{up}])$ *is stochastically finitely learnable with high confidence from positive presentations. To achieve* δ *-confidence no more than* $O(\log_2 1/\delta \cdot \log_\psi n)$ *many examples are necessary.*

Proof. The learner is based on the Wholist algorithm and a counter for the number of examples already processed. If the Wholist algorithm is run for $\tau := \lceil \log_\psi n \rceil + 3$ many examples Theorem 2 implies that τ is at least as large as the expected convergence stage $E[CONV]$.

In order to achieve the desired confidence, the learner sets $\gamma := \lceil \log \frac{1}{\delta} \rceil$ and runs the Wholist algorithm for a total of $2\,\gamma \cdot \tau$ examples. The algorithm outputs the last hypothesis $h_{2\,\gamma \cdot \tau}$ produced by the Wholist algorithm and stops thereafter. The reliability follows from the tail bounds established in Theorem 1. ∎

4.1. *Average-case Analysis for Learning in the Limit from Informant*

Finally, let us consider how the results obtained so far translate to the case of learning from informant. Since the Wholist algorithm does not learn anything from negative examples, one may expect that it behaves much poorer in this setting. Let us first investigate the uniform distribution over $\mathcal{X}_n$. Again, we have the trivial cases that the target concept is "FALSE" or m is a monomial without irrelevant bits. In the first case, no example is needed at all, while in the latter one, there is only one positive example having probability 2^{-n}. Thus the expected number of examples needed until successful learning is $2^n = 2^{\#(m)}$.

THEOREM 5. *Let* $c = L(m) \in \mathcal{C}_n$ *be a nontrivial concept. If a data sequence for* c *is generated from the uniform distribution on the learning domain by independent draws the expected number of examples needed by algorithm* $\boldsymbol{\mathcal{P}}$ *until convergence is bounded by*

$$E[CONV] \;\le\; 2^{\#(m)}\,(\lceil \log_2 k(m) \rceil + 3)\;.$$

Proof. Let $CONV+$ be a random variable for the number of positive examples needed until convergence. Every positive example is preceded by a possibly empty block of negative examples. Thus, we can partition the initial segment of any randomly drawn informant read until convergence into $CONV+$ many blocks B_j containing a certain number of negative examples followed by precisely one positive example. Let Λ_j be a random variable for the length of block B_j . Then $CONV = \Lambda_1 + \Lambda_2 + \cdots + \Lambda_{CONV+}$, where the Λ_j are independently identically distributed. In order to compute the distribution of Λ_j , it suffices to calculate the probabilities to draw a negative and a positive example, respectively. Since the overall number of positive examples for c is 2^k with $k = k(m)$, the probability to generate a positive example is 2^{k-n} . Hence, the probability to draw a negative example is $1 - 2^{k-n}$. Consequently,

$$\Pr[\Lambda_j = \mu + 1] \;=\; \left(1 - 2^{k-n}\right)^{\mu} \cdot 2^{k-n} \;.$$

Therefore,

$$\begin{aligned}
E[CONV] &= E[\Lambda_1 + \Lambda_2 + \cdots + \Lambda_{CONV+}] \\
&= \sum_{\zeta=0}^{\infty} E[\Lambda_1 + \Lambda_2 + \cdots + \Lambda_\zeta \mid CONV+ = \zeta] \;\cdot\; \Pr[CONV+ = \zeta] \\
&= \sum_{\zeta=0}^{\infty} \zeta \;\cdot\; E[\Lambda_1] \;\cdot\; \Pr[CONV+ = \zeta] \\
&= E[CONV+] \;\cdot\; E[\Lambda_1]
\end{aligned}$$

By Markov's inequality, we have $E[CONV+] \le \lceil \log_2 k \rceil + 3$, and thus it remains to estimate $E[\Lambda_1]$. A simple calculation shows

LEMMA 1. For every $0 < a < 1$ holds:

$$\sum_{\mu=0}^{\infty} (\mu + 1) \cdot a^{\mu} \;=\; (1 - a)^{-2} \;.$$

Using this estimation we can conclude

$$\begin{aligned}
E[\Lambda_1] &= \sum_{\mu=0}^{\infty} (\mu + 1) \;\cdot\; \Pr[\Lambda_1 = \mu + 1] \\
&= 2^{k-n} \sum_{\mu=0}^{\infty} (\mu + 1) \cdot \left(1 - 2^{k-n}\right)^{\mu} \;=\; 2^{n-k} \;,
\end{aligned}$$

and thus the theorem follows. ∎

Hence, as long as the length of m is constant, and therefore $k(m) = n - O(1)$, we still achieve an expected total learning time of order $n \log n$. But if $\#(m)$ grows linearly the expected total learning becomes exponential. On the other hand, if there are many relevant literals then even h_0 may be considered as a not too bad *approximation* for c. Consequently, it is natural at this point to introduce an error parameter $\varepsilon \in (0,1)$ as in the PAC model, and to ask whether one can achieve an expected sample complexity for computing an ε-approximation that is bounded by a function depending on $\log n$ and $1/\varepsilon$.

To answer this question, let us formally define $error_m(h_j) = D(L(h_j) \triangle L(m))$ to be the error made by hypothesis h_j with respect to monomial m. Here $L(h_j) \triangle L(m)$ stands for the symmetric difference of $L(h_j)$ and $L(m)$ and D for the underlying probability distribution with respect to which the examples are drawn. Note that by Theorem 1 we can conclude $error_m(h_j) = D(L(m) \setminus L(h_j))$. We call h_j an $\boldsymbol{\varepsilon}$**−*approximation*** for m if $error_m(h_j) \leq \varepsilon$. Finally, we redefine the stage of convergence. Let $d = (d_j)_{j \in \mathbb{N}^+}$ be an informant for $L(m)$, then

$$\boldsymbol{CONV_\varepsilon(d)} =_{df}$$
$$\text{the least number } j \text{ such that } error_m(P(d[i])) \leq \varepsilon \text{ for all } i \geq j.$$

Note that once the Wholist algorithm has reached an ε-approximate hypothesis all further hypotheses will also be at least that close to the target monomial.

The following theorem gives an affirmative answer to the question posed above.

THEOREM 6. *Let $c = L(m) \in \mathcal{C}_n$ be a nontrivial concept. Assuming that examples are drawn independently from the uniform distribution, the expected number of examples needed by algorithm $\boldsymbol{\mathcal{P}}$ until converging to an ε-approximation for c can be bounded by*

$$E[CONV_\varepsilon] \;\leq\; \frac{1}{\varepsilon} \cdot (\lceil \log_2 k(m) \rceil + 3) \;.$$

Proof. It holds $error_m(h_0) = 2^{k(m)-n}$, since h_0 misclassifies exactly the positive examples. Therefore, if $error_m(h_0) \leq \varepsilon$, we are already done. Now suppose $error_m(h_0) > \varepsilon$. Consequently, $1/\varepsilon > 2^{n-k(m)}$, and thus the bound stated in the theorem is larger than $2^{n-k(m)}(\lceil \log_2 k(m) \rceil + 3)$, which, by Theorem 5 is the expected number of examples needed until convergence to a correct hypothesis. ∎

Thus, additional knowledge concerning the underlying probability distribution pays off again. Applying Theorem 1 and modifying the stochastic finite learner presented above *mutatis mutandis*, we get a learner identifying ε-approximations for all concepts in $\mathcal{C}_n$ stochastically with high confidence using $O(\frac{1}{\varepsilon} \cdot \log \frac{1}{\delta} \cdot \log n)$ many examples. Comparing this bound with the sample complexity given in the PAC model, we see that it is reduced exponentially, i.e., instead of a factor n now we have the factor $\log n$.

Finally, we can generalize the last results to the case that the data sequences are binomially distributed for some parameter $p \in (0,1)$. This means that any particular vector containing ν times a 1 and $n - \nu$ a 0 has probability

$p^\nu(1-p)^{n-\nu}$ since a 1 is drawn with probability p and a 0 with probability $1-p$. First, Theorem 5 generalizes as follows.

THEOREM 7. *Let $c = L(m) \in \mathcal{C}_n$ be a nontrivial concept. Let m contain precisely π positive literals and τ negative literals. If the labeled examples for c are independently binomially distributed with parameter p and $\psi := \min\{\frac{1}{1-p}, \frac{1}{p}\}$ then the expected number of examples needed by algorithm $\boldsymbol{\mathcal{P}}$ until convergence can be bounded by*

$$E[CONV] \;\le\; \frac{1}{p^\pi(1-p)^\tau}\left(\lceil \log_\psi k(m)\rceil + 3\right) .$$

Proof. Assuming the same notation as in the proof of Theorem 5, it is easy to see that we only have to recompute $E[\Lambda_1]$, and thus $\Pr[\Lambda_1 = \mu+1]$, too. Since m contains precisely π positive literals and τ negative literals, the probability to draw a positive example is clearly $p^\pi(1-p)^\tau$, and thus the probability to randomly draw a negative example is $1-p^\pi(1-p)^\tau$. Consequently,

$$\Pr[\Lambda_1 = \mu+1] \;=\; (1-p^\pi(1-p)^\tau)^\mu \cdot p^\pi(1-p)^\tau ,$$

and Lemma 1 gives $E[\Lambda_1] = \frac{1}{p^\pi(1-p)^\tau}$. ∎

Theorem 6 directly translates into the setting of binomially distributed inputs, too.

THEOREM 8. *Let $c = L(m) \in \mathcal{C}_n$ be a nontrivial concept. Assume that the examples are drawn with respect to a binomial distribution with parameter p, and let $\psi = \min\{\frac{1}{1-p}, \frac{1}{p}\}$. Then the expected number of examples needed by algorithm $\boldsymbol{\mathcal{P}}$ until converging to an ε-approximation for c can be bounded by*

$$E[CONV] \;\le\; \frac{1}{\varepsilon}\cdot\left(\lceil \log_\psi k(m)\rceil + 3\right) .$$

Finally, one can also get stochastic finite approximations with high confidence from informant with an exponentially smaller sample complexity.

THEOREM 9. *Let $0 < p_{low} \le p_{up} < 1$ and $\psi := \min\{\frac{1}{1-p_{low}}, \frac{1}{p_{up}}\}$. For $(\mathcal{C}_n, \mathcal{D}_n[p_{low}, p_{up}])$ ε-approximations are stochastically finitely learnable with δ-confidence from informant for any $\varepsilon, \delta \in (0,1)$.*

For this purpose, $O(\frac{1}{\varepsilon}\cdot\log_2 1/\delta\cdot\log_\psi n)$ many examples suffice.

5. Learning the Pattern Languages

Finally, we take a look at the class of all pattern languages. The results presented below have been obtained by Rossmanith and Zeugmann [19] and we refer the reader to this paper for all omitted details.

This class is not PAC learnable with respect to every hypothesis space having a membership problem uniformly decidable in polynomial time.

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 $\mathcal{A}$.

Our pattern learner will directly output patterns as hypotheses. Note that this hypothesis space is not admissible for PAC learning unless $\mathcal{P} = \mathcal{NP}$. It uses Lange and Wiehagen's [11] pattern language learner as a main ingredient. We consider here learning from positive data only. Again, first an average-case analysis has to be performed. 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 $(u_0, \ldots, u_{k-1}) \in \mathcal{A}^+ \times \cdots \times \mathcal{A}^+$ we denote by $D(u_0, \ldots, u_{k-1})$ the probability that variable x_0 is substituted by u_0, variable x_1 is substituted by u_1, $\ldots$, 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, $0 \leq j \leq k-1$, 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.,

$D(u_0, \ldots, u_{k-1}) = \prod_{j=0}^{k-1} D_j(u_j)$. 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, $D(u_0, \ldots, u_{k-1}) = \prod_{j=0}^{k-1} d(u_j)$. We call a regular distribution *admissible* if $d(a) > 0$ for at least two different elements $a \in \mathcal{A}$.

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 $u_0, \ldots, u_{k-1}$ 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.,

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

Then we can prove the following bound for the stage of convergence for every target pattern from Pat_k .

THEOREM 10. $E[CONV] = O\left(\frac{1}{\alpha^k} \cdot \log_{1/\beta}(k)\right)$ *for all* $k \geq 2$.

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

The transformation of Lange and Wiehagen's [11] pattern language learner into a stochastic finite learner is the performed *mutatis mutandis*. The prior knowledge needed is now assumed concerning a lower bound for α and an upper bound for β . More precisely, we have the following theorem.

THEOREM 11. *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* $D \in \mathcal{D}$. *Then* $(PAT, \mathcal{D})$ *is stochastically finitely learnable with high confidence from text.*

The latter theorem allows a nice corollary which we state next. Making the same assumption as done by Kearns and Pitt [10], 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 [10] 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. [18]).

COROLLARY 12. *Let α_*, $\beta_* \in (0,1)$. 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. *)*

Further results have been obtained for the class of all one-variable pattern languages. Note that even this class is not PAC learnable as shown in [14]. For the class of all one-variable pattern languages our stochastic finite learner has a total learning time that is essentially linear in the length of the target pattern (cf. [16, 18]). Moreover, this stochastic finite learner works for all meaningful probability distributions.

These results provide at least some evidence that our new approach may lead to a new generation of learning algorithms that combine a feasible total learning time and proven performance guarantees.

References

1. D. Angluin, Finding Patterns common to a Set of Strings, *Journal of Computer and System Sciences* **21** (1980), 46–62.
2. A. Blumer, A. Ehrenfeucht, D. Haussler and M. Warmuth, Learnability and the Vapnik-Chervonenkis Dimension, *Journal of the ACM* **36** (1989), 929–965.
3. 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.
4. R. Daley and C.H. Smith. On the Complexity of Inductive Inference. *Information and Control* **69** (1986), 12–40.
5. 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.
6. E.M. Gold, Language identification in the limit, *Information and Control* **10** (1967), 447–474.
7. 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.

8. D. Haussler, Bias, version spaces and Valiant's learning framework. "Proc. 8th National Conference on Artificial Intelligence" (pp. 564–569). Morgan Kaufmann, 1987.
9. D. Haussler, M. Kearns, N. Littlestone and M.K. Warmuth, Equivalence of models for polynomial learnability. *Information and Computation* **95** (1991), 129–161.
10. M. Kearns L. Pitt, A polynomial-time algorithm for learning k –variable pattern languages from examples. "Proc. Second Annual ACM Workshop on Computational Learning Theory" (pp. 57–71). Morgan Kaufmann, 1989.
11. S. Lange and R. Wiehagen, Polynomial-time inference of arbitrary pattern languages. *New Generation Computing* **8** (1991), 361–370.
12. S. Lange and T. Zeugmann, Set-driven and Rearrangement-independent Learning of Recursive Languages, *Mathematical Systems Theory* **29** (1996), 599–634.
13. S. Lange and T. Zeugmann, Incremental Learning from Positive Data, *Journal of Computer and System Sciences* **53**(1996), 88–103.
14. 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.
15. 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.
16. 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.
17. 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.
18. 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.
19. P. Rossmanith and T. Zeugmann. Stochastic Finite Learning of the Pattern Languages, *Machine Learning* **44**, No. 1-2, 2001, 67–91.
20. L.G. Valiant, A Theory of the Learnable, *Communications of the ACM* **27** (1984), 1134–1142.
21. 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.
22. 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.