Stochastic Finite Learning of The Pattern Languages

Peter Rossmanith (rossmani@in.tum.de)
Institut für Informatik, Technische Universität München, 80290 München, Germany

Thomas Zeugmann (thomas@i.kyushu-u.ac.jp)
Department of Informatics, Kyushu University, Kasuga 816-8580, Japan

October 10, 1999

Abstract. The present paper proposes a new learning model – called *stochastic finite learning* – and shows the whole class of pattern languages to be learnable within this model.

This main result is achieved by providing a new and improved average-case analysis of the Lange–Wiehagen (1991) algorithm learning the class of all pattern languages in the limit from positive data. The complexity measure chosen is the *total learning time*, i.e., the overall time taken by the algorithm until *convergence*. The expectation of the total learning time is carefully analyzed and *exponentially* shrinking tail bounds for it are established for a large class of probability distributions. For every pattern π containing k different variables it is shown that Lange and Wiehagen's algorithm possesses an expected total learning time of $O\Big(\hat{\alpha}^k(\log_{1/\beta}(k)+2)E[\Lambda]\Big)$, where $\hat{\alpha}$ and β are two easy computable parameters arising naturally from the underlying probability distributions, and $E[\Lambda]$ is the expected example string length.

Finally, assuming a bit of *domain knowledge* concerning the underlying class of probability distributions, it is shown how to convert learning in the limit into *stochastic finite learning*.

Keywords: Inductive Learning, Pattern Languages, Average-Case Analysis, Learning in the Limit, Stochastic Finite Learning

1. Introduction

Suppose you have to deal with a learning problem of the following kind. You are given a concept class $\mathcal{C}$. On the one hand, it is known that $\mathcal{C}$ is not PAC learnable. On the other hand, your concept class has been proved to be learnable within Gold's [7] learning in the limit model. Here, a learner is successively fed data about the concept to be learned and it is computing a sequence of hypotheses about the target object. However, the only knowledge you have about this sequence is its convergence in the limit to a hypothesis correctly describing the target concept. Therefore, you *never* know whether 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. However, if the learner is additionally known to be *conservative* and *rearrangement independent* the following general strategy allows one to *efficiently* transform learning in the limit into stochastic finite learning. First, one has to choose a class of admissible probability distributions. Next, the expected total learning time for each concept in the considered concept class must be finite. If these assumptions are fulfilled then the probability to exceed the expected total learning time by a factor of t is exponentially small in t (cf. Corollary 1). Finally, a bit of additional domain knowledge about the underlying class of probability distributions nicely buys a learner behaving as follows. Again, it 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 $1-\delta$.

The learning scenario just described above is referred to as *stochastic finite learning* (cf. Section 4, Definition 2). 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 learning time has been met. Let C be the total number of examples read until this event has happened. 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 rapidly the probability of failure. At this point the exponentially shrinking tail bounds for the expected total learning time come into play. That is, the stochastic finite learner continues to simulate the limit learner until it has read and processed a total of $C\cdot\lceil\log\frac{1}{\delta}\rceil$ many examples. Then it outputs the last hypothesis computed in the simulation, and stops. This strategy works, since the probability of failure is halved each time a new sample of C many examples has been read and processed. Thus, when the stochastic finite learner stops, the limit learner has converged to a correct hypothesis with probability at least $1-\delta$.

Note that this strategy also works, if we do not have exponentially shrinking tail bounds for the expected total learning time but still know that this expectation is finite for all concepts in $\mathcal{C}$. However, in this case the desired confidence would rely on Markov's inequality, and thus the sample complexity would increase by a factor of $1/\delta$. On the other hand, in our setting the increase of the sample complexity is bounded by $\log(1/\delta)$, and therefore, the original efficiency of the limit learner is almost preserved.

Our model of stochastic finite learning differs to a certain extend 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.* [8]).

Furthermore, in the uniform PAC model as introduced in Valiant [26] 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.* [3] and Haussler *et al.* [10]). 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. [10]). 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 show.

In the present paper, we exemplify this approach by considering the class of all pattern languages (PAT for short). Our research derives its motivation from the fact that PAT is a prominent and important concept class that can be learned from positive data. Moreover, learning algorithms for pattern languages have already found interesting applications in a variety of domains such as molecular biology and data bases (cf., e.g., Salomaa [22, 23], and Shinohara and Arikawa [25] for an overview). Recently, Mitchell *et al.* [16] 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 [24] 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 consisting of all canonical patterns (*Pat* for short). Since the membership problem for this hypothesis space is $\mathcal{NP}$-complete, it is *not* polynomially evaluable (cf. [1]).

In contrast, Kearns and Pitt [12] 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. Additionally, the length of substitution strings has been required to be polynomially related to the length of the target pattern. Finally, their algoritm uses as hypothesis space all unions of polynomially many patterns that have k or fewer variables[1]. 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.

We present a stochastic finite learner for *PAT* from positive data only[2] that uses *Pat* as hypothesis space. For achieving this goal, we had to restrict the class of all (product) probability distributions to a subclass that has an arbitrary but fixed bound on two parameters arising naturally. In essence, that means at least two letters from the underlying probability distribution over the alphabet of constants have a *known* lower bound on their probability. In general, its running time is exponential in the length of the target pattern. However, if we assume the additional prior knowledge that all target patterns have at most k distinct variables as done by Kearns and Pitt [12] then the total learning time is *linearly* bounded in the *expected* length of sample strings fed to the learner. Now, the constant depends only exponentially

[1] 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.

[2] *PAT* is learnable in the limit from positive data but there is no deterministic learner that finitely infers *PAT* from positive data. In contrast, if learning from both positive and negative data is considered then *PAT is finitely learnable* by a deterministic algorithm (cf. [14]). Thus, negative data facilitate pattern language learning. Since finite learning is an idealized version of the PAC model, it is natural to ask, why then, Schapire [24] result is true. The different outcome in the two models is caused by the way in which data are treated. A PAC learner has to find a suitable approximation from *every* sample that is sufficiently large while a finite learner has the freedom to wait until the “informative” examples have been delivered.

on the number k of different variables occurring in the target pattern (cf. Corollary 3).

Thus, a smaller class of admissible probability distributions yields both a gain in efficiency and correctness. It is therefore natural to ask how reasonable it is to assume our class of admissible probability distributions, especially with respect to potential applications. There is, however, no unique answer to this question. The crucial point is that shortest examples must have a non-zero probability. There are application domains where this assumption is violated, for example, when learning data structures (cf. [25]). In other potential application domains the situation may be different. Nevertheless, further research is necessary to meet practical demands, e.g., handling noisy data.

An important ingredient to our learner is the Lange-Wiehagen algorithm (LWA for short) that infers *PAT* from positive data. We generalize and improve the average-case analysis of this algorithm performed by Zeugmann [27]. Finally, we shortly summarize the improvements obtained concerning the average-case analysis compared to the analysis undertaken in [27].

(1) Let Λ be the length of random positive examples according to some distribution. Zeugmann's [27] estimate of the expected total learning time contains the variance of Λ as a factor. While this variance is small for many distribution it is sometimes infinity and sometimes very big. The new analysis does not use variances or other higher moments of Λ at all. Thus it turns out that the expected total learning time is bounded iff the expectation of Λ is bounded.

(2) We present exponentially fast shrinking tail bounds for the expected total learning time. Previous work did not study tail bounds at all.

(3) The new analysis presents the bound on the expected total learning time as a simple formula that contains only three parameters: α, β, and $E[\Lambda]$. The parameters α and β are very simple to compute for each probability distribution.

(4) The new analysis is slightly tighter: For the "uniform" distribution, e.g., the upper bound is by a factor of $k^2|\pi|$ smaller, where $|\pi|$ is the length of the pattern and k again the number of different variables occurring in π. We also give all bounds as exact formulas without hiding constant factors in a big O. These constants, however, are not the best possible in order to keep the proofs simple.

(5) Besides the total learning time we give separate estimates on the number of iterations until convergence, the number of union operations performed by the LWA, and the time spent in union operations. The union operation is the most difficult part in the LWA. On the one hand, we provide a new algorithm for computing the union operation

that achieves linear time, while all previously known algorithms take quadratic time. Nevertheless, our algorithm needs quadratic space. Thus, these extra estimates help to judge whether this optimization is worthwhile. It turns out that except for rather pathological distributions the time spent for union operations is quite small even when each union operation takes quadratic time. Hence, if space is a serious matter of concern, one can still stick to the naïve implementation without increasing the overall time bound too much.

2. Preliminaries

Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ be the set of all natural numbers, and let $\mathbb{N}^+ = \mathbb{N} \setminus \{0\}$. For all real numbers x we define $\lfloor x \rfloor$, the *floor function*, to be the greatest integer less than or equal to x. Furthermore, by $\lceil x \rceil$ we denote the *ceiling function*, i.e., the least integer greater than or equal to x.

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 $\mathcal{A}$ (cf. Hopcroft and Ullman [11]). 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. By $|\mathcal{A}|$ we denote the cardinality of $\mathcal{A}$. 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 $0 \le j < k-1$, 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.

Let $\pi \in Pat$, $1 \le i \le |\pi|$; we use $\pi(i)$ to denote the i-th symbol in π. If $\pi(i) \in \mathcal{A}$, then we refer to $\pi(i)$ as to a *constant*; otherwise $\pi(i) \in X$, and we refer to $\pi(i)$ as to a *variable*. Analogously, by $s(i)$ we denote the i-th symbol in s for every string $s \in \mathcal{A}^+$ and all $i = 1, \ldots, |s|$. By $\#\mathrm{var}(\pi)$ we denote the number of different variables occurring in π, and by $\#_{x_i}(\pi)$ we denote the number of occurrences of variable x_i in π. If $\#\mathrm{var}(\pi) = k$, then we refer to π as to a *k-variable pattern*. Let $k \in \mathbb{N}$, 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 $\pi[x_0/u_0, \dots, x_{k-1}/u_{k-1}]$ the string $w \in \mathcal{A}^+$ obtained by substituting u_j for each occurrence of x_j, $j = 0, \dots, k-1$, in the pattern π. For example, let $\pi = 0x_01x_1x_0$. Then $\pi[x_0/10, x_1/01] = 01010110$. The tuple $(u_0, \dots, u_{k-1})$ is called a *substitution*. Furthermore, if $|u_0| = \cdots = |u_{k-1}| = 1$, then we refer to $(u_0, \dots, u_{k-1})$ as to a *shortest substitution*. Let $\pi \in Pat_k$; we define the *language generated by pattern* π by $L(\pi) = \{\pi[x_0/u_0, \dots, x_{k-1}/u_{k-1}] \mid u_0, \dots, u_{k-1} \in \mathcal{A}^+\}$. 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 $\mathcal{A}$. Note that for every $L \in PAT$ there is precisely one pattern $\pi \in Pat$ such that $L = L(\pi)$ (cf. Angluin [1]).

We are interested in *inductive inference*, which means to gradually learn a concept from successively growing sequences of examples. If L is a language to be identified, a sequence $t = (s_1, s_2, s_3, \dots)$ is called a *text* for L if $L = \{s_1, s_2, s_3, \dots\}$ (cf. [7]). However, in practical applications, the requirement to exhaust the language to be learned will be hardly fulfilled. We therefore *omit* this assumption here. Instead, we generalize the notion of text to the case that the sequence $t = s_1, s_2, s_3, \dots$ contains "enough" information to recognize the target pattern. As for the LWA, "enough" can be made precise by requesting that sufficiently many shortest strings appear in the text. We shall come back to this point when defining admissible probability distributions. Let t be a text, and $n \in \mathbb{N}^+$; then we set $t_n = s_1, \dots, s_n$, and we refer to t_n as the initial segment of t of length n.

As introduced by Gold [7] an *inductive inference machine* 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 languages described by the hypotheses. A hypothesis is said to describe a language L iff $L = h$.

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

(1) for all $n \in \mathbb{N}^+$, $M(t_n)$ is defined,

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

In the case of pattern languages and the k-variable pattern languages the hypothesis space is Pat and Pat_k, respectively. Moreover, we avoid defining a particular enumerations of Pat and Pat_k. Instead, our learning algorithms will directly output pattern as hypotheses.

Whenever one deals with the average case analysis of algorithms one has to consider probability distributions over the relevant input domain. For learning from text, we have the following scenario. 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, …, 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, $D(u_0, \ldots, u_{k-1}) = \prod_{j=0}^{k-1} d(u_j)$. As a special case of a regular product distribution we sometimes 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 $|u| = \ell$.

Note, however, that most of our results can be generalized to larger classes of distributions. Finally, we can provide the announced specification of what is meant by "enough" information. We call a regular distribution *admissible* if $d(a) > 0$ for at least two different elements $a \in \mathcal{A}$.

Following Daley and Smith [4] we define the total learning time as follows. Let M be any IIM that learns all the pattern languages. Then, for every $L \in PAT$ and every text t for L, 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. [7]). Furthermore, we define $Conv(M,t) = \infty$ if M does not learn the target language 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. As for the pattern languages, the membership problem is $\mathcal{NP}$-complete (cf. Angluin [1]). Thus, testing consistency would immediately lead to a non-polynomial update time unless $\mathcal{P} = \mathcal{NP}$. Finally, it should be mentioned that defining an appropriate complexity measure for learning in the limit is a difficult problem. We refer the reader to Pitt [18] for a more detailed discussion.

Assuming any fixed admissible probability distribution D as described above, we aim to evaluate the *expectation* of $TT(M, t)$ with respect to D which we refer to as the *expected total learning time*.

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.

Finally, we recall the LWA. The LWA works as follows. Let h_n be the hypothesis computed after reading $s_1, \ldots, s_n$, i.e., $h_n = M(s_1, \ldots, s_n)$. 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. Hence, the LWA is quite simple and the update time will be very fast for these two possibilities.

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

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

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.

Figure 1 displays the union operation for $\pi = 01x_0x_121x_0x_201x_0x_1$ and $s = 120021010212$. Since the letters in the first and second column are different to one another, and since the first and second column differ, too, we get $\varrho(1) = x_0$ and $\varrho(2) = x_1$. Next, $\pi(3) = x_0$ and $\pi(4) = x_1$, and thus ϱ must also contain different variables at positions 3 and 4. Consequently, these variables get renamed, i.e., $\varrho(3) = x_2$ and $\varrho(4) = x_3$. The letters in the 5th and 6th column are identical, hence $\varrho(5) = 2$ and $\varrho(6) = 1$ (cf. the first case in the definition of the union operation). In the 7th column, we have x_0 and 0 and this column is equal to to the third column. Therefore, the second case in the definition of the union operation applies and $\varrho(7) = x_2$. Now, $\varrho(8) = x_4$ and $\varrho(9) = 0$ are obvious. The 10th column is identical to the second one, thus $\varrho(10) = x_1$. Next, we have x_0 and 1 while both the third and 7th column contain x_0 and 0. Therefore, a new variable has to be introduced and $\varrho(11) = x_5$ (cf. the third case in the definition of the union operation). Analogously, the x_1 in the 12th column has to be distinguished from the x_1 in 4th column resulting in $\varrho(12) = x_6$.

π	0	1	x_0	x_1	2	1	x_0	x_2	0	1	x_0	x_1
s	1	2	0	0	2	1	0	1	0	2	1	2
$\varrho \cup s$	x_0	x_1	x_2	x_3	2	1	x_2	x_4	0	x_1	x_5	x_6

Figure 1

Obviously, the union operation can be computed in time $O(|\pi|^2)$, i.e., in quadratic time. We finish this section by providing a linear-time algorithm computing the union operation. The only crucial part is to determine whether or not there is some $k < i$ with $\varrho(k) = x_j$, $s(k) = s(i)$, and $\pi(k) = \pi(i)$. The new algorithm uses an array $I =$

$\{1, \ldots, |s|\}^{\mathcal{A}\times(\mathcal{A}\cup\{x_0,\ldots,x_{|\pi|-1}\})}$ for finding the correct k, if any, in *constant* time. The array I is *partially* initialized by $I_{s(i),\pi(i)} = j$, where j is the smallest number such that $s(i) = s(j)$ and $\pi(i) = \pi(j)$. Then, for each position i, the algorithm checks whether $I_{s(i),\pi(i)} = i$. Suppose it is, thus $s(i), \pi(i)$ did not occur left of i. Hence, it remains to check whether or not $\pi(i) = s(i)$ and $\varrho(i)$ is either the constant $s(i)$ or a new variable. If $I_{s(i),\pi(i)} \neq i$, then $s(i), \pi(i)$ did occur left of i. Hence, in this case it suffices to output $\varrho(j)$ where $j = I_{s(i),\pi(i)}$.

THEOREM 1. *The union operation can be computed in linear time.*

Proof. The following algorithm constructs $\varrho = \pi \cup s$ in linear time.

***Algorithm* 1**

Input: A pattern π and a string $s \in \mathcal{A}^+$ such that $|\pi| = |s|$.
Output: $\pi \cup s$
Method:
for $i = |s|, \ldots, 1$ **do** $I_{s(i),\pi(i)} \leftarrow i$;
$m \leftarrow 0$;
for $i = 1, \ldots, |s|$ **do** $j \leftarrow I_{s(i),\pi(i)}$
  if $i = j$ **then**
    if $\pi(i) = s(i)$ **then** $\varrho(i) \leftarrow \pi(i)$
    else $\varrho(i) \leftarrow x_m$; $m \leftarrow m + 1$ **fi**
  else $\varrho(i) = \varrho(j)$ **fi**
od

The correctness of this algorithm can be easily proved inductively by formalizing the argument given above. We omit the details.

3. The Average-Case Analysis

Following [27] we perform the desired analysis in dependence on the number k of different variables occurring in the target pattern π. If $k = 0$, then the LWA immediately converges. Therefore, in the following we assume $k \in \mathbb{N}^+$, and $\pi \in Pat_k$. Taking into account that $|w| \geq |\pi|$ for every $w \in L(\pi)$, it is obvious that the LWA can only converge if it has been fed sufficiently many strings from $L(\pi)$ having minimal length. Therefore let

$$L(\pi)_{min} = \{w \mid w \in L(\pi),\ |w| = |\pi|\}.$$

Zeugmann [27] found an exact formula for the minimum number of examples that the LWA needs to converge:

PROPOSITION 1 (Zeugmann [27]). *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.*

Clearly, in order to match this bound all examples must have been drawn from $L(\pi)_{min}$. In the worst case there is no upper bound on the number of examples.

For analyzing the *average-case* behavior of the LWA, in the following we let $t = s_1, s_2, s_3, \ldots$ range over all randomly generated texts with respect to some arbitrarily fixed admissible probability distribution D. Then the stage of convergence is a random variable which we denote by C. Note that the distribution of C depends on π and on D. We introduce several more random variables. By Λ_i we denote the length of the example string s_i, i.e., $\Lambda_i = |s_i|$. Since all Λ_i are independent and identically distributed, we can assume that the random variable has the same distribution as Λ. We will use Λ when talking about the length of an example when the number of the example is not important. Particularly, we will often use the expected length of a random example $E[\Lambda]$.

Let T be the *total length* of examples processed until convergence, i.e.,

$$T = \Lambda_1 + \Lambda_2 + \ldots + \Lambda_C.$$

Whether the LWA converges on $s_1, \ldots, s_r$ depends only on those examples s_i with $s_i \in L(\pi)_{min}$. Let $r \in \mathbb{N}^+$; by M_r we denote the number of minimum length examples among the first r strings, i.e.,

$$M_r = |\{\, i \,|\, 1 \le i \le r \text{ and } \Lambda_i = |\pi|\,\}|.$$

In particular, M_C is the number of minimum length examples read until convergence. We assume that reading and processing one character takes exactly one time step in the LWA unless union operations are performed. Disregarding the union operations, the total learning time is then T. The number of union operations until convergence is denoted by U. The time spent in union operations until convergence is V. The total learning time is therefore $TT = T + V$. We assume that computing $\varrho \cup s$ takes at most $c \cdot |\rho|$ steps, where c is a constant that depends on the implementation of the union operation.

We will express all estimates with the help of the following parameters: $E[\Lambda]$, c, α and β. 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.,

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

The parameter α and β are therefore quite easy to compute even for complicated distributions since they depend only on $|\mathcal{A}|$ point probabilities. We can also compute $E[\Lambda]$ for a pattern π from d quite easily.

Let $u = (u_0, \ldots, u_{k-1})$ be any substitution. Because of

$$|\pi[x_0/u_0, \ldots, x_{k-1}/u_{k-1}]| = |\pi| + \sum_{i=0}^{k-1} \#_{x_i}(\pi)(|u_i| - 1),$$

we have $E[\Lambda] = |\pi| + v(E[|u|] - 1)$, where v is the total number of variable occurrences in π, i.e., $v = \sum_{i=0}^{k-1} \#_{x_i}(\pi)$.

In our analysis we will use often the *median* of a random variable. If X is a random variable then μX is a median of X iff

$$\Pr(X \geq \mu X) \geq 1/2 \text{ and } \Pr(X \leq \mu X) \geq 1/2.$$

A nonempty set of medians exists for each random variable and consists either of a single real number or of a closed real interval. We will denote the smallest median of X by μX, since this choice gives the best upper bounds.

Next, we present the main results and compare them to Zeugmann's [27] analysis. His distribution independent results read as follows in our notation:

PROPOSITION 2 (Zeugmann [27], Theorem 8).
$E[TT] = O(E[C] \cdot (V[\Lambda] + E^2[\Lambda]))$.

The variance of Λ is herein denoted by $V[\Lambda]$. The parameters $V[\Lambda]$ and $E[\Lambda]$ have to be computed for a given distribution. For $E[C]$ he gives an estimate with the help of $E[M_C]$ and $E[T_j]$, which is the expected time to receive the first j pairwise different elements from $L(\pi)_{min}$:

PROPOSITION 3 (Zeugmann [27], Theorem 8).

$$E[C] \le E[M_C]\cdot\max\left\{E[T_1],\ \frac{1}{2}\sum_{j=1}^{2} E[T_j],\ \ldots,\ \frac{1}{|\mathcal{A}|^{k-1}+1}\sum_{j=1}^{|\mathcal{A}|^{k-1}+1} E[T_j]\right\}$$

He then proceeds to estimate these parameters for the *uniform distribution*, where $d(u) = 2^{-|u|}|\mathcal{A}|^{-|u|}$. Here his estimate is as follows:

PROPOSITION 4 (Zeugmann [27], Theorem 11).

$E[TT] = O(2^k k^2 |\pi|^2 \log_{|\mathcal{A}|}(k|\mathcal{A}|))$ *for the uniform distribution.*

The main difference to our analysis are the parameters that have to be evaluated for a given distribution. Instead of $E[\Lambda]$, $V[\Lambda]$, $E[M_C]$, $E[T_j]$, and $|\mathcal{A}|$ we use α, β, and $E[\Lambda]$, which are easier to obtain. Setting $\hat{\alpha} = 1/\alpha$, we can estimate the total learning time as follows:

THEOREM 2. $E[TT] = O(\hat{\alpha}^k(\log_{1/\beta}(k) + 2)E[\Lambda])$.

Theorem 2 may look complicated, but it is rather simple to evaluate. Moreover, the variance of Λ is not used at all. Take for example some distribution with $\Pr(|u| = 2^i) = 3 \cdot 4^{-i-1}$ and $\Pr(|u| = n) = 0$ if n is not a power of 2. Then $E[\Lambda] \le (3/2)|\pi|$, but $V[\Lambda] = \infty$. Hence, Proposition 2 just says $E[TT] \le \infty$. Since $\hat{\alpha} = 4/3$, Theorem 2 yields a very good upper bound, i.e., $E[TT] = O((4/3)^k(\log_{1/\beta}(k) + 2)|\pi|)$. Even if $V[\Lambda]$ exists it can be much bigger than $E^2[\Lambda]$.

Next, we insert the parameter for the uniform distribution into Theorem 2. For the uniform distribution we get $\hat{\alpha} = 2$, $\beta = 1/|\mathcal{A}|$, and $E[\Lambda] \le 2|\pi|$.

THEOREM 3. $E[TT] = O(2^k|\pi|\log_{|\mathcal{A}|}(k))$ *for the uniform distribution.*

This estimate is slightly better than Proposition 4.

We continue by investigating other expected values of interest. Often time is the most precious resource and then we have to minimize the total learning time. The number of examples until convergence can also be interesting, if the gathering of examples is expensive. Then the average number of examples is the critical parameter and we are interested in $E[C]$.

THEOREM 4. $E[C] = O(\hat{\alpha}^k \cdot \log_{1/\beta}(k))$.

If we compare Theorems 2 and 4 it turns out that in many cases the total learning time is by a factor of about $E[\Lambda]$ larger than the number of examples read until convergence. This is about the same time an algorithm uses that just reads $E[C]$ random positive examples.

We can even get a better understanding of the behavior if we examine the union operations by themselves. Is it worthwhile to optimize the computation of $w \cup \pi$? It turns out that union operations are responsible only for a small part of the overall computation time. Recall that U is the number of union operations and V is the time spent in union operations.

THEOREM 5.

(1) $E[U] = O(\hat{\alpha}k + \log_{1/\beta}(k))$

(2) $E[V] = O(\hat{\alpha}kE[\Lambda] + \log_{1/\beta}(k)|\pi|)$ *provided the union operation is performed by Algorithm 1,*

(3) $E[V] = O(\hat{\alpha}kE^2[\Lambda] + \log_{1/\beta}(k)|\pi|^2)$ *if the union operation is performed by the naïve algorithm.*

Consequently, if space is a serious matter of concern, e.g., if the patterns to be learned are very long, one may easily trade a bit more time by using the naïve, quadratic time algorithm instead of Algorithm 1 above.

3.1. Tail Bounds

Finally we have to ask whether the average total learning time is sufficient for judging the LWA. The expected value of a random variable is only one aspect of its distribution. In general we might also be interested on how often the learning time exceeds the average substantially. Again this is a question motivated mainly by practical considerations. Equivalently we can ask, how well the distribution is concentrated around its expected value. Often this question is answered by estimating the *variance*, which enables the use of Chebyshev's inequality. If the variance is not available, Markov's inequality provides us with (worse) tail bounds:

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

Markov's inequality is quite general but produces only weak bounds. The next theorem gives much better tail bounds for a large class of learning algorithms including the LWA. The point here is, that the

LWA possesses two additional desirable properties, i.e., it is *rearrangement-independent* (cf. [27]) and *conservative*. A learner is said to be rearrangement-independent, if its outputs depend only on the range and length of its input (cf. [6]). Conservative learners maintain their actual hypotheses at least as long as they have not seen data contradicting them (cf. Angluin [2]).

THEOREM 6. *Let X be the sample complexity of a conservative and rearrangement-independent learning algorithm. Then* $\Pr(X \geq t \cdot \mu X) \leq 2^{-t}$ *for all* $t \in \mathbb{N}$.

Proof. We divide the text $(s_1, s_2, \ldots)$ into blocks of length μX. The probability that the algorithm converges after reading any of the blocks is then at least $1/2$. Since the algorithm is rearrangement-independent the order of the blocks does not matter and since the algorithm is conservative it does not change its hypothesis after computing once the right pattern.

COROLLARY 1. *Let X be the sample complexity of a conservative and rearrangement-independent learning algorithm. Then* $\Pr(X \geq 2t \cdot E[X]) \leq 2^{-t}$ *for all* $t \in \mathbb{N}$.

Proof. Since $\mu X \leq 2E[X]$ for every positive random variable X by the Markov inequality, we get immediately that $\Pr(X \geq 2t \cdot E[X]) \leq 2^{-t}$ for every $t \in \mathbb{N}$.

Theorem 6 and Corollary 1 put 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 [15]).

Since the distribution of X decreases geometrically, all higher moments of X exist and can be bounded by a polynomial in μX. The next two theorems establish this for the expected value and the variance of X. Similar results hold for higher moments.

THEOREM 7. *Let X be the sample complexity of a conservative and rearrangement-independent learning algorithm. Then* $E[X] \leq 2\mu X$.

Proof.

$$E[X] = \sum_{i=1}^{\infty} \Pr(X \geq i) \;\leq\; \sum_{i=1}^{\infty} 2^{-\lfloor i/\mu X \rfloor} \leq \sum_{i=0}^{\infty} \sum_{j=0}^{\mu X - 1} 2^{-\left\lfloor \frac{i\mu X + j}{\mu X} \right\rfloor}$$

$$= \sum_{i=0}^{\infty} \mu X \cdot 2^{-i} = 2\mu X$$

THEOREM 8. *Let X be the sample complexity of a conservative and rearrangement-independent learning algorithm. Then* $V[X] \leq 2\mu X(3\mu X - E[X]) \leq 5(\mu X)^2$.

Proof.

$$\begin{aligned}
V[X] &= E[X^2] - E^2[X] = \sum_{i=0}^{\infty} i^2 \cdot \Pr(X = i) - E^2[X] \\
&= \sum_{i=0}^{\infty} i^2 \cdot (\Pr(X \geq i) - \Pr(X \geq i+1)) - E^2[X] \\
&= \underbrace{\sum_{i=0}^{\infty} i^2 \Pr(X \geq i) - \sum_{i=0}^{\infty} (i+1)^2 \Pr(X \geq i+1)}_{=0} \\
&\quad + \sum_{i=0}^{\infty} (2i+1) \Pr(X \geq i+1) - E^2[X] \\
&= \sum_{i=1}^{\infty} (2i-1) \cdot \Pr(X \geq i) - E[X] \sum_{i=1}^{\infty} \Pr(X \geq i) \\
&= \sum_{i=1}^{\infty} (2i - 1 - E[X]) \Pr(X \geq i) \\
&\leq \sum_{i=1}^{\infty} (2i - 1 - E[X]) \cdot 2^{-\lfloor i/\mu X \rfloor} \text{ (by Theorem 6)} \\
&\leq \sum_{i=0}^{\infty} \sum_{j=0}^{\mu X - 1} (2(i\mu X + j) - 1 - E[X]) \cdot 2^{-\left\lfloor \frac{i\mu X + j}{\mu X} \right\rfloor} + 1 + E[X] \\
&= \sum_{i=0}^{\infty} \mu X (2(i+1)\mu X - E[X] - 2) \cdot 2^{-i} + 1 + E[X] \\
&\leq 2\mu X (3\mu X - E[X] - 2) + 1 + 2\mu X \\
&\leq 2\mu X (3\mu X - E[X]) \leq 5(\mu X)^2
\end{aligned}$$

3.2. The Sample Complexity

In this section we estimate the sample complexity. While being of interest itself, whenever acquiring examples is expensive, $E[C]$ is also an important ingredient in the estimation of the total learning time. In estimating $E[C]$, we first need $E[M_C]$, which we get from tail bounds of M_C given in Lemma 1 below. In the following, we adopt the convention that $\binom{k}{2} = 0$ provided $k = 1$.

LEMMA 1. $\Pr(M_C > m) = \Pr(C > r \mid M_r = m) \le \binom{k}{2}\beta^m + k\beta^{m/2}$ *for all* $m, r \in \mathbb{N}^+$ *with* $r \ge m$.

Proof. For proving the equality, first note that $M_C > m$ holds if and only if $C > r$ *under the condition* that $M_r = m$. Since changing the order of examples that yield no convergence cannot force the learner to converge, we get

$$\Pr(M_C > m) = \Pr(M_C > m \mid M_r = m) .$$

Next we prove the inequality. Without loss of generality, let $S_r = \{s_1, \ldots, s_m\}$, i.e., $m = r$. Additionally, we can make the assumption that all strings $s_i \in S_r$ have length k, since we need to consider only shortest strings for M_C and we can assume that $\pi = x_0 x_1 \ldots x_{k-1}$ (cf. [27]). For $1 \le j \le k$ let $c_j = s_0(j)s_1(j)\ldots s_{m-1}(j)$ be the jth *column* of a matrix whose rows are $s_1, \ldots, s_m$.

The LWA computes the hypothesis π on input S_r iff no column is constant and there are no identical columns. The probability that c_j is constant is at most $\beta^{m/2}$, since $m/2$ pairs have to be identical. But this short argument works only for even m.

The probability that a column is constant, i.e., the probability that m independent random substitution strings are identical under the condition that each length is exactly 1, is

$$\sum_{a \in \mathcal{A}} d(a)^m \Bigg/ \left(\sum_{a \in \mathcal{A}} d(a)\right)^m .$$

In the following we show that this quantity is at most $\beta^{m/2}$. We start with the following inequality obtained by the multinomial theorem.

$$\sum_{a \in \mathcal{A}} d(a)^m = \sum_{a \in \mathcal{A}} d(a)^{2 \cdot \frac{m}{2}} \le \left(\sum_{a \in \mathcal{A}} d(a)^2\right)^{m/2}$$

Dividing both sides by $\left(\sum_{a \in \mathcal{A}} d(a)\right)^m$ yields

$$\frac{\sum_{a \in \mathcal{A}} d(a)^m}{\left(\sum_{a \in \mathcal{A}} d(a)\right)^m} \le \frac{\left(\sum_{a \in \mathcal{A}} d(a)^2\right)^{m/2}}{\left(\left(\sum_{a \in \mathcal{A}} d(a)\right)^2\right)^{m/2}}$$

$$= \left(\sum_{a \in \mathcal{A}} d(a)^2 \Big/ \left(\sum_{a \in \mathcal{A}} d(a) \right)^2 \right)^{m/2} = \beta^{m/2}$$

Consequently, the probability that at least one of the k columns is constant is then at most $k\beta^{m/2}$. Thus, for $k = 1$, we are already done.

Next, assume $k \geq 2$. We estimate the probability that there are at least two identical columns. The probability that $c_i = c_j$ is β^m provided $i \neq j$. Therefore, the probability that some columns are equal is at most $\binom{k}{2}\beta^m$. Finally, putting it all together, we see that the probability of having at least one constant column *or* at least two identical columns is at most $\binom{k}{2}\beta^m + k\beta^{m/2}$.

Inserting the above tail bounds into the definition of the expected value yields an upper bound on $E[M_C]$.

LEMMA 2. $E[M_C] \leq \dfrac{2\ln(k) + 3}{\ln(1/\beta)} + 2.$

Proof. M_C is the number of shortest strings read until convergence. By Lemma 1 we have $\Pr(M_C > m) \leq \binom{k}{2}\beta^m + k\beta^{m/2}$.

$$\begin{aligned} E[M_C] &= \sum_{m=0}^{\infty} \Pr(M_C > m) \\ &\leq \ell + \sum_{m=l}^{\infty} \left(\binom{k}{2}\beta^m + k\beta^{m/2} \right) \\ &= \ell + \binom{k}{2} \frac{\beta^\ell}{1-\beta} + k \frac{\sqrt{\beta}^\ell}{1-\sqrt{\beta}} \end{aligned}$$

for each natural number ℓ. We choose $\ell = \lceil 2\log_{1/\beta}(k) \rceil + 1$, which yields when inserted into the above inequality

$$E[M_C] \quad \leq \quad \ell + \frac{\beta}{1-\beta} + \frac{\sqrt{\beta}}{1-\sqrt{\beta}}.$$

The lemma now follows from the inequality

$$\frac{\beta}{1-\beta} + \frac{\sqrt{\beta}}{1-\sqrt{\beta}} \quad \leq \quad \frac{3}{\ln(1/\beta)} \ ,$$

which can be proved by standard methods from calculus.

The expression given in Lemma 2 looks a bit complicated and we therefore continue by further estimating it. However, if $k = 1$, then $E[M_C] \leq 1/\ln(1/\beta)$ which cannot be simplified. Therefore, in the following corollary we assume $k \geq 2$.

COROLLARY 2. *$E[M_C] \leq 7\log_{1/\beta}(k) + 2 = O(\log_{1/\beta}(k))$ for all $k \geq 2$.*

Proof. By Lemma 2 we have

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

Using $3 < 5\ln(k)$, we obtain $2\ln(k) + 3 \leq 7\ln(k)$ and hence,

$$\frac{2\ln(k)+3}{\ln(1/\beta)} \leq 7\log_{1/\beta}(k) = O(\log_{1/\beta}(k))$$

Thus, the constant hidden in the O-notation is quite moderat and for larger k even smaller constants can be obtained. For avoiding a further case distinction between $k = 1$ and $k \geq 2$, we assume in the following $k \geq 2$ whenever the O-notation is used.

Now, we already know the expectation for the number of strings from $L(\pi)_{min}$ the LWA has to read until convergence. Our next major goal is to establish an upper bound on the *overall* number of examples to be read on average by the LWA until convergence. This is done by the next theorem.

THEOREM 9. *$E[C] = \hat{\alpha}^k E[M_C] = O(\hat{\alpha}^k \log_{1/\beta}(k))$.*

Proof. The LWA converges after reading exactly C example strings. Among these examples are M_C many of minimum length. Every string of minimum length is preceded by a possibly empty block of strings whose length is bigger than $|\pi|$. Hence, we can partition the initial segment of any randomly drawn text read until convergence into M_C many blocks B_j containing a certain number of strings s with $|s| > |\pi|$ followed by precisely one string of minimum length. Let G_j be a random variable for the number of examples in blocks B_j. Then $C = G_1 + G_2 + \cdots + G_{M_C}$. It is easy to compute the distribution of G_i:

$$\Pr(G_i = m+1) = \Pr(\Lambda > |\pi|)^m \Pr(\Lambda = |\pi|) = (1-\alpha^k)^m \alpha^k \qquad (1)$$

Of course, all G_i are identically distributed and independent. The expected value of C is therefore

$$\begin{aligned} E[C] &= E[G_1 + \cdots + G_{M_C}] \\ &= \sum_{m=0}^{\infty} E[G_1 + \cdots + G_m \mid M_C = m] \cdot \Pr(M_C = m) \\ &= \sum_{m=0}^{\infty} m \cdot E[G_1] \cdot \Pr(M_C = m) \\ &= E[M_C] \cdot E[G_1] \qquad (2) \end{aligned}$$

The expected value of G_1 is

$$\begin{aligned} E[G_1] &= \sum_{m=0}^{\infty}(m+1)\cdot \Pr(G_1 = m+1) \\ &= \sum_{m=0}^{\infty} m(1-\alpha^k)^m\alpha^k + \sum_{m=0}^{\infty}(1-\alpha^k)^m\alpha^k \\ &= \frac{1-\alpha^k}{\alpha^k} + 1 = \frac{1}{\alpha^k} \end{aligned} \tag{3}$$

Combining (2) and (3) proves the first equality of the theorem, and the second one follows by Corollary 2.

3.3. The Length of the Text until Convergence

Now, we are almost done. For establishing the main theorem, i.e., the expected total learning time, it suffices to calculate the expected length of a randomly generated text until the LWA converges.

LEMMA 3. *Let $m \geq 1$. Then $E[\Lambda_1 \mid G_1 = m] = (E[\Lambda] - \alpha^k)/(1-\alpha^k)$.*

Proof. This proof is based on $\Pr(\Lambda_1 = i \mid G_1 = m) = \Pr(\Lambda_1 \mid \Lambda_1 > |\pi|)$, which intuitively holds because for Λ_1 the condition $G_1 = m$ means that the first block of non-minimum length strings is not empty and this holds iff Λ_1 has not minimum length. More formally note that $G_1 = m$ is equivalent to $\Lambda_i > |\pi|$ for $1 \leq i \leq m$ and $\Lambda_{m+1} = |\pi|$ and therefore

$$E[\Lambda_1 \mid G_1 = m] = E[\Lambda_1 \mid \Lambda_1 > |\pi| \wedge \cdots \wedge \Lambda_m > |\pi| \wedge \Lambda_{m+1} = |\pi|]\ .$$

Since all Λ_i are independent it boils down to $E[\Lambda_1 \mid G_1 = m] = E[\Lambda_1 \mid \Lambda_1 > |\pi|]$. Now it is easy to compute $E[\Lambda_1 \mid G_1 = m]$:

$$\begin{aligned} & E[\Lambda_1 \mid G_1 = m] \\ &= \sum_{i=|\pi|+1}^{\infty} i\cdot \Pr(\Lambda_1 = i \mid G_1 = m) \\ &= \sum_{i=|\pi|+1}^{\infty} i\cdot \Pr(\Lambda_1 = i \mid \Lambda_1 > |\pi|) \ = \sum_{i=|\pi|+1}^{\infty} i\cdot \frac{\Pr(\Lambda_1 = i)}{\Pr(\Lambda_1 > |\pi|)} \\ &= \frac{E[\Lambda_1]}{\Pr(\Lambda_1 > |\pi|)} - \frac{\Pr(\Lambda_1 = |\pi|)}{\Pr(\Lambda_1 > |\pi|)} \ = \ \frac{E[\Lambda] - \alpha^k}{1-\alpha^k} \end{aligned}$$

THEOREM 10.
$E[T] = E[M_C] \cdot (|\pi| + \hat{\alpha}^k (E[\Lambda] - 1)) = O(\hat{\alpha}^k (\log_{1/\beta}(k) + 2) E[\Lambda])$.

Proof. We can write the length of text read until convergence as $T = T_1 + T_2 + \cdots + T_{M_C} + |\pi| M_C$. Exactly M_C strings of length $|\pi|$ are read; all other strings are longer and are contained in blocks in front of those minimum length strings. The ith blocks contains G_i strings and we denote the total length of these G_i strings by T_i. In order to get $E[T]$ we start by computing $E[T_1]$.

$$
\begin{aligned}
E[T_1] &= \sum_{m=0}^{\infty} E[\Lambda_1 + \cdots + \Lambda_m \mid G_1 = m] \cdot \Pr(G_1 = m) \\
&= \sum_{m=1}^{\infty} m \cdot E[\Lambda_1 \mid G_1 = m] \cdot \Pr(G_1 = m) \\
&= \sum_{m=1}^{\infty} m \cdot \frac{E[\Lambda] - \alpha^k}{1 - \alpha^k} \cdot (1 - \alpha^k)^m \alpha^k \quad \text{(by Lemma 3 and (1))} \\
&= (E[\Lambda] - \alpha^k) \alpha^k \sum_{m=1}^{\infty} m (1 - \alpha^k)^{m-1} \\
&= \hat{\alpha}^k E[\Lambda] - 1
\end{aligned}
$$

Now it is easy to estimate $E[T]$. We use that T_1 and M_C are independent.

$$
\begin{aligned}
E[T] - |\pi| E[M_C] &= E[T_1 + \cdots + T_{M_C}] \\
&= \sum_{m=0}^{\infty} m \cdot E[T_1] \cdot \Pr(M_C = m) = E[M_C] \cdot E[T_1]
\end{aligned}
$$

and thus $E[T] = E[M_C](|\pi| + \hat{\alpha}^k E[\Lambda] - 1)$. Finally insert the estimation of $E[M_C]$ from Corollary 2.

4. Stochastic Finite Learning with High Confidence

Now we are ready to introduce our new learning model. For defining it in its whole generality, we assume any fixed learning domain and any concept class $\mathcal{C}$ over it. The information given to the learner can be either texts as defined above or both positive and negative data. In the latter case, it is assumed that all examples are labeled with respect to their containment in the target concept. In the definition below, we refer to both types of information sequences as *data sequences*.

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 finitely *with high confidence*.

Note that the learner in the definition above takes δ as additional input.

Next, we show how the LWA can be transformed into a stochastic finite learner that identifies all the pattern languages from text with high confidence. To do this, we assume a bit of prior knowledge about the class of admissible distributions that may actually be used to generate the random texts.

THEOREM 11. *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 $d \in \mathcal{D}$. 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. Note that d induces an admissible probability distribution D. 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 $|\tau_n| \geq |\tau_{n+1}|$ for all $n \in \mathbb{N}^+$.

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

$$C = \hat{\alpha}_*^{|\tau|} \cdot \left(\frac{2\ln(|\tau|)+3}{\ln(1/\beta)} + 2\right) \qquad (A)$$

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 9. On the one hand, this follows from our assumptions about the allowed α and β as well as from the fact that $|\tau| \geq |\pi|$ for every hypothesis output. On the other hand, the learner does not know k, but the estimate $\#var(\pi) \leq |\pi|$ is sufficient. Therefore, after having processed C many examples the LWA has already converged on average. The desired confidence is then an immediate consequence of Corollary 1.

The latter theorem allows a nice corollary which we state next. Making the same assumption as done by Kearns and Pitt [12], 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 [12] 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.

COROLLARY 3. *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 $E(d)$ finite for all distributions $d \in \mathcal{D}$. Furthermore, let $k \in \mathbb{N}^+$ 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 $O(\hat{\alpha}_*^k(\log_{1/\beta_*}(k) + 2)E[\Lambda]\log_2(1/\delta))$.*
(Note that $\hat{\alpha}_*^k$ and $\log_{1/\beta_*}(k)$ now are constants. *)*

Proof. The learner works precisely as in the proof of Theorem 11 except that (A) is replaced by

$$C = \hat{\alpha}_*^k \cdot \left(\frac{2\ln(k) + 3}{\ln(1/\beta)} + 2\right) \qquad (A')$$

The correctness follows as above by Theorem 9 and Corollary 1, since the target belongs to Pat_k. The running time is a direct consequence of Theorem 10 and the choice of γ.

One more remark is mandatory here. The learners described above can be made more efficiently by using even better tail bounds. We therefore continue to establish some more tail bounds. Note that each of these bounds has a special range where it outperforms the other ones. Hence, the concrete choice in an actual implementation of the algorithm above depends on the precise values of α and β. Since these values are usually not known precisely, it is advantageous to take the minimum of all three.

LEMMA 4. $\Pr(M_r \geq er\alpha^k) \leq e^{-r\alpha^k}$ *and* $\Pr(M_r \leq \frac{1}{2}r\alpha^k) \leq (e/2)^{-r\alpha^k/2}$.

Proof. The expected value of M_r is $r\alpha^k$, since $\Pr(\Lambda = k) = \alpha^k$. Chernoff bounds [9, (12)] yield

$$\Pr(M_r \geq er\alpha^k) \leq \left(\frac{r\alpha^k}{er\alpha^k}\right)^{er\alpha^k} e^{er\alpha^k - r\alpha^k} = e^{-r\alpha^k}$$

and

$$\Pr(M_r \leq \tfrac{1}{2}r\alpha^k) \leq \left(\frac{r\alpha^k}{r\alpha^k/2}\right)^{r\alpha^k/2} e^{r\alpha^k/2 - r\alpha^k} = (e/2)^{-r\alpha^k/2}.$$

THEOREM 12. $\Pr(C > r) \leq k^2\beta^{\frac{1}{4}r\alpha^k} + (e/2)^{-r\alpha^k/2}$.

Proof. We split $\Pr(C > r)$ into a sum of conditional probabilities according to the conditions $M_r \geq m$ and $M_r < m$ for a well chosen parameter m.

In the following we use the first part of Lemma 1.

$$\begin{aligned}
\Pr(C > r) &= \sum_{i=0}^{\infty} \Pr(C > r \mid M_r = m) \Pr(M_r = i) \\
&= \Pr(M_r \leq m) + \sum_{i=0}^{\infty} \Pr(C > r \mid M_r = i) \Pr(M_r = i) \\
&= \Pr(M_r \leq m) + \sum_{i=m+1}^{\infty} \Pr(M_C > i) \Pr(M_r = i) \\
&\leq \Pr(M_r \leq m) + \Pr(M_C > m)
\end{aligned}$$

We choose $m = \frac{1}{2}r\alpha^k$ and get

$$\Pr(C > r) \leq k^2\beta^{\frac{1}{4}r\alpha^k} + (e/2)^{-r\alpha^k/2}$$

by Lemma 1 and 4.

THEOREM 13. $\Pr(C > r) \le k^2 e^{-\alpha^k(1-\beta)r}$.

Proof. Using $\Pr(C > r \mid M_r = m) = \Pr(M_C > m \mid M_r = m) = \Pr(M_C > m)$ we can write $\Pr(C > r)$ as a sum of products:

$$\Pr(C > r) = \sum_{m=0}^{r} \Pr(M_C > m) \cdot \Pr(M_r = m).$$

Now $\Pr(M_C > m) \le k^2 \beta^{m/2}$ by Lemma 1 and $\Pr(M_r = m) = \binom{r}{m}\alpha^{km}(1-\alpha^k)^{r-m}$, since M_r has a binomial distribution with parameters α^k and $1-\alpha^k$. Using these estimates we get immediately

$$\begin{aligned}
\Pr(C > r) &\le \sum_{m=0}^{r} \binom{r}{m} k^2 \beta^{m/2} \alpha^{km} (1-\alpha^k)^{r-m} \\
&= k^2 \Big(\sqrt{\beta}\alpha^k + 1 - \alpha^k\Big)^r \\
&= k^2 \Big(1 - \alpha^k(1-\sqrt{\beta})\Big)^r \\
&\le k^2 \Big(\frac{1}{e}\Big)^{\alpha^k(1-\sqrt{\beta})r}
\end{aligned}$$

and the theorem is proved.

Finally, we omit the proof of Theorem 5, Assertion (3) due to the lack of space and refer the reader to [21] instead.

5. Conclusions

The present paper dealt with the average-case analysis of Lange and Wiehagen's pattern language learning algorithm with respect to its total learning time. The results presented considerably improved the analysis made by Zeugmann [27]. Clearly, the question arises whether the improvement is worth the effort undertaken to obtain it. This question has been naturally answered by the introduction of our new model of stochastic finite learning. 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.*[5] as well as in Reischuk and Zeugmann [19, 20] provide further support for the fruitfulness of this approach.

Moreover, the approach undertaken may also provide the necessary tools to perform the average-case analysis of a wider variety of learning algorithms. In particular, the newly developed techniques to estimate the average-case behavior by showing very useful tail bounds seem to be generalizable to the large class of conservative and rearrangement-independent limit learning algorithms.

Acknowledgements

Both authors heartily thank the anonymous referees for their careful reading and many valuable comments.

References

1. Angluin, D. (1980). Finding patterns common to a set of strings. *Journal of Computer and System Sciences, 21*, 46–62.
2. Angluin, D. (1980). Inductive inference of formal languages from positive data. *Information and Control, 45*, 117–135.
3. Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M.K. (1989). Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM, 36*, 926–965.
4. Daley, R., & Smith, C.H. (1986). On the complexity of inductive inference. *Information and Control, 69*, 12–40.
5. Erlebach, T., Rossmanith, P., Stadtherr, H., Steger, A., & Zeugmann, T. (1997). Learning one-variable pattern languages very efficiently on average, in parallel, and by asking queries. In M. Li & A. Maruoka (Eds.), *Proceedings of the Eighth International Workshop on Algorithmic Learning Theory*, Lecture Notes in Artificial Intelligence 1316 (pp. 18–44). Berlin: Springer-Verlag.
6. Fulk, M. (1990). Prudence and other restrictions in formal language learning, *Information and Computation, 85*, 1–11.
7. Gold, E.M. (1967). Language identification in the limit. *Information and Control, 10*, 447–474.
8. Goldman, S.A., Kearns, M.J., & Schapire, R.E. (1993). Exact identification of circuits using fixed points of amplification functions. *SIAM Journal of Computing, 22*, 705–726.
9. Hagerup, T., & Rüb, C. (1990). A guided tour of Chernoff bounds. *Information Processing Letters, 33*, 305–308.
10. Haussler, D., Kearns, M., Littlestone, N., & Warmuth, M.K. (1991). Equivalence of models for polynomial learnability. *Information and Computation, 95*, 129–161.
11. Hopcroft, J.E., & Ullman, J.D. (1969). *Formal Languages and their Relation to Automata.* Reading, MA: Addison-Wesley.
12. Kearns, M., & Pitt, L. (1989). A polynomial-time algorithm for learning k–variable pattern languages from examples. In R. Rivest, D. Haussler & M.K. Warmuth (Eds.), *Proceedings of the Second Annual ACM Workshop on Computational Learning Theory* (pp. 57–71). San Mateo, CA: Morgan Kaufmann.

13. Lange, S., & Wiehagen, R. (1991). Polynomial-time inference of arbitrary pattern languages. *New Generation Computing, 8*, 361–370.
14. Lange, S., & Zeugmann, T. (1993). Monotonic versus non-monotonic language learning. In G. Brewka, K.P. Jantke & P.H. Schmitt (Eds.), *Proceedings of the Second International Workshop on Nonmonotonic and Inductive Logic*, Lecture Notes in Artificial Intelligence 659 (pp. 254–269). Berlin: Springer-Verlag.
15. Lange, S., & Zeugmann, T. (1996). Set-driven and rearrangement-independent learning of recursive languages. *Mathematical Systems Theory, 29*, 599–634.
16. Mitchell, A., Scheffer, T., Sharma, A., & Stephan, F. (1999). The VC-dimension of subclasses of pattern languages. In O. Watanabe & T. Yokomori (Eds.), *Proceedings of the Tenth International Conference on Algorithmic Learning Theory*, Lecture Notes in Artificial Intelligence 1720 (pp. 89–100). Berlin: Springer-Verlag.
17. Muggleton, S. (1994). Bayesian inductive logic programming. In W. Cohen & H. Hirsh (Eds.), *Proceedings of the Eleventh International Conference on Machine Learning*, (pp. 371–379). San Mateo, CA: Morgan Kaufmann.
18. Pitt, L. (1989). Inductive inference, DFAs and computational complexity. In K.P. Jantke (Ed.), *Proceedings of the International Workshop on Analogical and Inductive Inference*, Lecture Notes in Artificial Intelligence 397 (pp. 18–44). Berlin: Springer-Verlag.
19. Reischuk, R., & Zeugmann, T. (1998). Learning one-variable pattern languages in linear average time. In P. Bartlett & Y. Mansour (Eds.), *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, (pp. 198–208). New York, NY: ACM Press.
20. Reischuk, R., & Zeugmann, T. (1999). A complete and tight average-case analysis of learning monomials. In C. Meinel and S. Tison (Eds.), *Proceedings of the Sixteenth International Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science 1563 (pp. 414–423). Berlin: Springer-Verlag.
21. Rossmanith, P., & Zeugmann, T. (1998). Learning k-variable pattern languages efficiently stochastically finite on average from positive data. DOI Technical Report DOI-TR-145, Department of Informatics, Kyushu University.
22. Salomaa, A. (1994). Patterns. (The Formal Language Theory Column). *EATCS Bulletin 54*, 46–62.
23. Salomaa, A. (1994). Return to patterns. (The Formal Language Theory Column). *EATCS Bulletin 55*, 144–157.
24. Schapire, R.E. (1990). 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.
25. Shinohara, T., & Arikawa, S. (1995). Pattern inference. In K.P. Jantke & S. Lange (Eds.), *Algorithmic Learning for Knowledge-Based Systems*, Lecture Notes in Artificial Intelligence 961, (pp. 259–291). Berlin: Springer-Verlag.
26. Valiant, L.G. (1984). A theory of the learnable. *Communications of the ACM 27*, 1134–1142.
27. Zeugmann, T. (1998). Lange and Wiehagen's pattern learning algorithm: An average-case analysis with respect to its total learning time. *Annals of Mathematics and Artificial Intelligence, 23* , 117–145.
28. Zeugmann, T., & Lange, S. (1995). A guided tour across the boundaries of learning recursive languages. In K.P. Jantke & S. Lange (Eds.), *Algorithmic*

Learning for Knowledge-Based Systems, Lecture Notes in Artificial Intelligence 961, (pp. 190–258). Berlin: Springer-Verlag.