Learning Concepts Incrementally With Bounded Data Mining*

JOHN CASE
Department of CIS
University of Delaware
Newark
DE 19716, USA
case@cis.udel.edu

SANJAY JAIN
Department of ISCS
National University of Singapore
Lower Kent Ridge Road
Singapore 119260, Rep. of Singapore
sanjay@iscs.nus.sg

STEFFEN LANGE
FB Mathematik und Informatik
HTWK Leipzig
Postfach 30066
04251 Leipzig, Germany
steffen@informatik.htwk-leipzig.de

THOMAS ZEUGMANN
Department of Informatics
Kyushu University
Fukuoka 812–81, Japan
thomas@i.kyushu-u.ac.jp

Abstract

Important refinements of incremental concept learning from positive data *considerably restricting the accessibility of input data* are studied. Let c be any concept; every infinite sequence of elements exhausting c is called *positive presentation* of c. In all learning models considered the learning machine computes a sequence of hypotheses about the target concept from a positive presentation of it. With *iterative* learning, the learning machine, in making a conjecture, has access only to its previous conjecture and the latest datum coming in. In k-*bounded example-memory* inference (k is *a priori* fixed) the learner is allowed to access, in making a conjecture, only its previous hypothesis, its memory of up to k data items it has already seen, and the latest datum coming in. In the case of k-*feedback* identification, the learning machine, in making a conjecture, has access only to its previous conjecture, the latest datum coming in, *and*, on the basis of this information, it can compute k items and query the database of previous data to find out, for each of the k items, whether or not it is in the database (k is again *a priori* fixed). In all cases, the sequence of conjectures has to converge to a hypothesis correctly describing the target concept.

Our results are manyfold. An infinite hierarchy of successively more powerful feedback learners in dependence on the number k of queries allowed to be asked is established. However, the hierarchy collapses to 1-feedback inference if only indexed families of *infinite* concepts are considered, and, moreover, its learning power is then equal to unrestricted incremental learning; however, the hierarchy remains infinite for concept classes of only *infinite* r.e. concepts. Both k-feedback inference and k-bounded example-memory identification are more powerful than iterative learning but, surprisingly, incomparable to one another. Furthermore, there *are* cases where redundancy in the hypothesis space is shown to be a resource increasing the learning power of iterative learners. Finally, of the important class of unions up to k pattern languages is shown to be *iteratively* inferable.

*An expansion of the present paper, with all proofs included, appears as a technical report (cf. Case *et al.* (1997)).

1. Introduction

The present paper derives its motivation to a certain extent from the rapidly emerging field of knowledge discovery in databases (abbr. KDD). Historically, there is a variety of names including data mining, knowledge extraction, information discovery, data pattern processing, information harvesting, and data archeology all referring to finding useful information that has not been known before about the data. Throughout this paper we shall use the term *KDD* for the *overall process* of discovering useful knowledge from data and *data mining* to refer to the particular subprocess of applying specific algorithms for learning something useful from the data. Thus, the additional steps such as data presentation, data selection, incorporating prior knowledge, and defining the semantics of the results obtained belong to KDD (cf., e.g., Fayyad *et al.* (1996b)). Prominent examples of KDD applications in health care and finance include Matheus *et al.* (1996) and Kloesgen (1995). The importance of KDD research finds its explanation in the fact that the data collected in various fields such as biology, finance, retail, astronomy, medicine are extremely rapidly growing, while our ability to analyze those data has not kept up proportionally.

KDD mainly combines techniques originating from machine learning, knowledge acquisition and knowledge representation, artificial intelligence, pattern recognition, statistics, data visualization, and databases to automatically extract new interrelations, knowledge, patterns and the like from *huge* collections of data. Usually, the data are available from massive data sets collected, for example, by scientific instruments (cf., e.g., Fayyad *et al.* (1996a)), by scientists all over the world (as in the human genome project), or in databases that have been built for other purposes than a current purpose.

We shall be mainly concerned with the extraction of *concepts* in the data mining process. Thereby, we emphasize the aspect of working with *huge* data sets. For example, in Fayyad *et al.* (1996a) the SKICAT-system is described which operates on 3 terabytes of image data originating from approximately 2 billion sky objects which had to be classified. If huge data sets are around, no learning algorithm can use all the data or even large portions of it simultaneously for computing hypotheses about concepts represented by the data. Different methods have been proposed for overcoming the difficulties caused by huge data sets. For example, *sampling* may be a method of choice. That is, instead of doing the discovery process on all the data, one starts with significantly smaller samples, finds the regularities in it, and uses the different portions of the overall data to verify what one has found. Clearly, a major problem involved concerns the choice of the right sampling size. One way proposed to solve this problem as well as other problems related to huge data sets is *interaction* and *iteration* (cf., e.g., Brachman and Anand (1996) and Fayyad *et al.* (1996b)). That is, the whole data mining process is iterated a few times, thereby allowing human interaction until a satisfactory interpretation of the data is found.

Looking at data mining from the perspective described above, it becomes a true limiting process. That means, the actual result of the data mining algorithm application run on a sample is tested versus (some of) the remaining data. Then, if, for any reason whatever, a current hypothesis is not acceptable, the sample may be enlarged (or replaced) and the algorithm is run again. Since the data set is extremely large, clearly not all data can be validated in a prespecified amount of time. Thus, from a theoretical point of view, it is appropriate to look at the data mining process as an *ongoing, incremental* one.

In the present theoretical study, then, we focus on *important refinements or restrictions of* Gold's (1967) model of learning *in the limit* grammars for concepts from positive instances.[1] Gold's (1967) model itself makes the unrealistic assumption that the learner has access to samples of increasingly growing size. Therefore, we investigate refinements that *considerably restrict the accessibility of input data.* In particular, we deal with so-called *iterative* learning, *bounded example-memory* inference, and *feedback* identification (cf. Definitions 3, 4, and 5, respectively). Each of these models formalizes a kind of *incremental learning.* In each of these models we imagine a stream of positive data coming in about a concept and that the data that arrived in the past sit in a database which can get very very large. Intuitively, with *iterative* learning, the learning machine, in making a conjecture, has access to its previous conjecture and the latest data item coming in — *period.* In *bounded example-memory* inference, the learning machine, in making a conjecture, has access to its previous conjecture, its *memory* of *up to* k data items it has seen, and a new data item. Hence, a bounded example-memory machine wanting to memorize a *new* data item it's just seen, if it's already remembering k previous data items, must *forget* one of the previous k items in its memory to make room for the new one! In the case of *feedback* identification, the learning machine, in making a conjecture, has access to its previous conjecture, the latest data item coming in, *and*, on the basis of this information, it can compute k items and query the database of previous data to find out, for each of the k items, whether or not it is in the database. For some extremely large databases, a query about whether an item is in there can be very expensive, so, in such cases, feedback identification is interesting when the bound k is small.

Of course the $k = 0$ cases of bounded example-memory inference and feedback identification are just iterative learning.

Next we summarize informally our main results.

Theorems 2 and 3 imply that, for each $k > 0$, there are concept classes of infinite r.e. languages which can be learned by some feedback machine using no more than k queries of the database, but *no* feedback machine can learn these classes if it's restricted to no more than $k - 1$ queries.[2] Hence, each additional, possibly expensive dip into the database buys more concept learning power. Theorem 2 is a consequence of Theorem 3, and the proof of the latter is non-trivial. However, the feedback hierarchy collapses to its first level if only *indexable classes* of *infinite* concepts are to be learned (cf. Theorem 4).

A bounded example-memory machine can remember *its choice of* k items from the data, and it can *choose* to forget some old items so as to remember

[1] The sub-focus on learning *grammars*, or, equivalently, recognizers (cf. Hopcroft and Ullman (1969)), for concepts from *positive* instances nicely models the situation where the database flags or contains *examples* of the concept to be learned and doesn't flag or contain the non-examples.

[2] That the concepts in the concept classes witnessing this hierarchy are all *infinite* languages is also interesting and for two reasons: 1. It is arguable that all natural languages are infinite, and 2. many language learning *un*solvability results *depend strongly* on including the finite languages (cf. Gold (1967), Case (1996)). Ditto for other results below, namely, Theorems 6 and 7, which are witnessed by concept classes containing only infinite concepts.

some new ones. On the other hand, at each point, the feedback machine can query the database about *its choice of* k things each being or not being in the database. A bounded example-memory machine chooses which k items to *memorize* as being in the database, and the feedback machine can decide which k items to *lookup* to see if they are in the database. There are apparent similarities between these two kinds of learning machines, yet Theorems 6 and 7 show that in very strong senses, for each of these two models, there are concept class domains where that model is competent and the other is not!

Theorem 8 shows that, even in fairly concrete contexts, with iterative learning, *redundancy* in the hypothesis space increases learning power.

Angluin's (1980a) *pattern languages* are learnable from positive data, and they (and finite unions thereof) have been extensively studied and applied to molecular biology and to the learning of interesting special classes of logic programs (see the references in Section 3.4 below). Theorem 9 implies that, for each $k > 0$, the concept class consisting of all unions of at most k pattern languages is learnable (from positive data) by an iterative machine!

Because of space limitations, we have omitted most proofs, but they can be found in Case *et al.* (1997).

2. Preliminaries

Unspecified notation follows Rogers (1967). In addition to or in contrast with Rogers (1967) we use the following. By $\mathbb{N} = \{0, 1, 2, \dots\}$ we denote the set of all natural numbers. We set $\mathbb{N}^+ = \mathbb{N} \setminus \{0\}$. The cardinality of a set S is denoted by $|S|$. Let $\emptyset$, $\in$, $\subset$, $\subseteq$, $\supset$, and $\supseteq$, denote the empty set, element of, proper subset, subset, proper superset, and superset, respectively. Let S_1, S_2 be any sets; then we write $S_1 \triangle S_2$ to denote the symmetric difference of S_1 and S_2, i.e., $S_1 \triangle S_2 = (S_1 \setminus S_2) \cup (S_2 \setminus S_1)$. Additionally, for any sets S_1 and S_2 and $a \in \mathbb{N} \cup \{*\}$ we write $S_1 =^a S_2$ provided $|S_1 \triangle S_2| \leq a$, where $a = *$ means that the symmetric difference is finite. By $\max S$ and $\min S$ we denote the maximum and minimum of a set S, respectively, where, by convention, $\max \emptyset = 0$ and $\min \emptyset = \infty$.

By $\langle \cdot, \cdot \rangle \colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ we denote *Cantor's pairing function.*[3] Moreover, we let π_1 and π_2 denote the corresponding *projection functions* over $\mathbb{N}$ to the first and second components, respectively. That is, $\pi_1(\langle x, y \rangle) = x$ and $\pi_2(\langle x, y \rangle) = y$ for all $x, y \in \mathbb{N}$.

Let $\varphi_0, \varphi_1, \varphi_2, \dots$ denote any fixed standard *programming system* for all (and only) the partial recursive functions over $\mathbb{N}$, and let $\Phi_0, \Phi_1, \Phi_2, \dots$ be any associated *complexity measure* (cf. Blum (1967)). Then φ_k is the partial recursive function computed by *program* k. Furthermore, let $k, x \in \mathbb{N}$; if $\varphi_k(x)$ is defined (abbr. $\varphi_k(x)\downarrow$) then we also say that $\varphi_k(x)$ *converges*; otherwise $\varphi_k(x)$ *diverges*.

Any recursively enumerable set $\mathcal{X}$ is called a *learning domain*. By $\wp(\mathcal{X})$ we denote the power set of $\mathcal{X}$. Let $\mathcal{C} \subseteq \wp(\mathcal{X})$, and let $c \in \mathcal{C}$; then we refer to $\mathcal{C}$ and c as a *concept class* and a *concept*, respectively. Let c be a concept, and let $T = x_0, x_1, x_2, \dots$ an infinite sequence of elements $x_i \in c \cup \{\#\}$ such that $\text{range}(T) =_{df} \{x_k \mid x_k \neq \#,\ k \in \mathbb{N}\} = c$. 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 be a number. Then, T_y denotes the initial segment of T of length $y + 1$, and $T_y^+ =_{df} \{x_k \mid x_k \neq \#,\ k \leq y\}$. We refer to T_y^+ as the *content* of T_y. Intuitively, the #'s represent pauses in the positive presentation of the data of a concept c. Furthermore, let $\sigma = x_0, \dots, x_{n-1}$ be any finite sequence. Then we use $|\sigma|$ to denote the *length* n of σ. Additionally, let T be a text and let τ be a finite sequence; then we use $\sigma \diamond T$ and $\sigma \diamond \tau$ to denote the sequence obtained by *concatenating* σ onto the front of T and τ, respectively. By *SEQ* we denote the set of all finite sequences of elements from $\mathcal{X} \cup \{\#\}$.

As a special case, we often consider the scenario $\mathcal{X} = \mathbb{N}$, and $\mathcal{C} = \mathcal{E}$, where $\mathcal{E}$ denotes the collection of all recursively enumerable sets W_i, $i \in \mathbb{N}$, of natural numbers. These sets W_i can be described as $W_i = \text{domain}(\varphi_i)$. Thus, we also say that W_i is accepted, recognized (or, equivalently, generated) by the φ-program i. Hence, we also refer to the index i of W_i as a *grammar* for W_i.

Furthermore, we sometimes consider the scenario that indexed families of recursive languages have to be learned (cf. Angluin (1980b)). Let Σ be any finite alphabet of symbols, and let $\mathcal{X}$ be the free monoid over Σ, i.e., $\mathcal{X} = \Sigma^*$. As usual, we refer to subsets $L \subseteq \mathcal{X}$ as to languages. A class of non-empty recursive languages $\mathcal{L}$ is said to be an *indexed family* provided there are an effective enumeration $L_0, L_1, L_2, \dots$ of all and only the languages in $\mathcal{L}$ and a recursive function f such that for all $j \in \mathbb{N}$ and all strings $x \in \mathcal{X}$ we have

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

Since the paper of Angluin (1980b) learning of indexed families of languages has attracted much attention (cf., e.g., Zeugmann and Lange (1995)). Mainly, this seems due to the fact that most of the established language families such as regular languages, context-free languages, context-sensitive languages, and pattern languages are indexed families.

Essentially from Gold (1967) we define an *inductive inference machine* (abbr. *IIM*), or simply a learning machine, to be an algorithmic mapping from *SEQ* to

[3] This function is easily computable, 1-1, and onto (cf. Rogers (1967)).

$\mathbb{N} \cup \{?\}$. Intuitively, we interpret the output of a learning machine with respect to a suitably chosen hypothesis space $\mathcal{H}$. The output "?" is uniformly interpreted as "no conjecture." We always take as a hypothesis space a recursively enumerable family $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ of concepts (construed as sets or languages), where the j in h_j is thought of a numerical name for some finite description or computer program for h_j. Moreover, let c be a concept, let h_j be a hypothesis, and let $a \in \mathbb{N} \cup \{*\}$; then we write $c =^a h_j$ iff $|c \triangle h_j| \leq a$. That is, if $a \in \mathbb{N}$, then h_j describes c up to at most a anomalies. The $*$ is used to express any finite number of anomalies. We let M, with or without decorations, range over learning machines.

Let T be a positive presentation, and let $y \in \mathbb{N}$. The sequence $(M(T_y))_{y\in\mathbb{N}}$ is said to *converge* to the number j iff in $(M(T_y))_{y\in\mathbb{N}}$ all but finitely many terms are equal to j.

Now we define some models of learning. We start with Gold's (1967) unrestricted learning in the limit (and some variants). Then we will present the definitions of the models which more usefully restrict access to the database.

DEFINITION 1 (*Gold* (1967)). *Let $\mathcal{C}$ be a concept class, let c be a concept, let $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ be a hypothesis space, and let $a \in \mathbb{N} \cup \{*\}$. An IIM M $TxtEx^a_{\mathcal{H}}$-infers c iff, for every $T \in text(c)$, there exists a $j \in \mathbb{N}$ such that the sequence $(M(T_y))_{y\in\mathbb{N}}$ converges to j and $c =^a h_j$.*

M $TxtEx^a_{\mathcal{H}}$-infers $\mathcal{C}$ iff M $TxtEx^a_{\mathcal{H}}$-infers c, for each $c \in \mathcal{C}$.

Let $TxtEx^a_{\mathcal{H}}$ denote the collection of all concept classes $\mathcal{C}$ for which there is an IIM M such that M $TxtEx^a_{\mathcal{H}}$-infers $\mathcal{C}$.

$TxtEx^a$ denotes the collection of all concept classes $\mathcal{C}$ for which there are an IIM M and a hypothesis space $\mathcal{H}$ such that M $TxtEx^a_{\mathcal{H}}$-infers $\mathcal{C}$.

The a represents the number of mistakes or anomalies allowed in the final conjectures (cf. Case and Smith (1983)), with $a = 0$ being Gold's (1967) original case where no mistakes are allowed. If $a = 0$, we usually omit the upper index, e.g., we write $TxtEx$ instead of $TxtEx^0$. We adopt this convention in the definitions of the learning types below.

Since, by the definition of convergence, only finitely many data about c were seen by the IIM up to the (unknown) point of convergence, whenever an IIM infers the concept c, some form of learning must have taken place. For this reason, hereinafter the terms *infer*, *learn*, and *identify* are used interchangeably.

For $TxtEx^a_{\mathcal{H}}$-inference, a learner has to converge to a *single* description for the target to be inferred. However, it is imaginable that humans do not converge to a single grammar when learning their mother tongue. Instead, we may learn a small number of *equivalent* grammars each of which is easier to apply than the others in quite different situations. This speculation directly suggests the following definition.

DEFINITION 2 (*Case and Smith* (1983)). *Let $\mathcal{C}$ be a concept class, let c be a concept, let $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ be a hypothesis space, and let $a \in \mathbb{N} \cup \{*\}$. An IIM M $TxtFex^a_{\mathcal{H}}$-infers c iff, for every $T \in text(c)$, there exists a nonempty finite set D such that $c =^a h_j$, for all $j \in D$ and $M(T_y) \in D$, for all but finitely many y.*

M $TxtFex^a_{\mathcal{H}}$-infers $\mathcal{C}$ iff M $TxtFex^a_{\mathcal{H}}$-infers c, for each $c \in \mathcal{C}$.

Let $TxtFex^a_{\mathcal{H}}$ denote the collection of all concept classes $\mathcal{C}$ for which there is an IIM M such that M $TxtFex^a_{\mathcal{H}}$-infers $\mathcal{C}$.

$TxtFex^a$ denotes the collection of all concept classes $\mathcal{C}$ for which there are an IIM M and a hypothesis space $\mathcal{H}$ such that M $TxtFex^a_{\mathcal{H}}$-infers $\mathcal{C}$.

The following theorem clarifies the relation between Gold's (1967) classical learning in the limit and *TxtFex*-inference. The assertion remains true even if the learner is only allowed to *vacillate* between up to 2 descriptions, i.e., in the case $|D| \leq 2$ (cf. Case (1988; 1996)).

Theorem 1 (Osherson *et al.* (1986); *Case* (1988; 1996)). *$TxtEx^a \subset TxtFex^a$, for all $a \in \mathbb{N} \cup \{*\}$.*

Looking at the above definitions, we see that an IIM M has always access to the whole history of the learning process, i.e., in order to compute its actual guess M is fed all examples seen so far. In contrast to that, next we define *iterative IIMs* and a natural generalization of them called *bounded example-memory IIMs*. An iterative IIM is only allowed to use its last guess and the next element in the positive presentation of the target concept for computing its actual guess. Conceptionally, an iterative IIM M defines a sequence $(M_n)_{n\in N}$ of machines each of which takes as its input the output of its predecessor.

DEFINITION 3 (*Wiehagen* (1976)). *Let $\mathcal{C}$ be a concept class, let c be a concept, let $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ be a hypothesis space, and let $a \in \mathbb{N} \cup \{*\}$. An IIM M $TxtItEx^a_{\mathcal{H}}$–infers c iff for every $T = (x_j)_{j\in\mathbb{N}} \in text(c)$ the following conditions are satisfied:*

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

Finally, M $TxtItEx^a_{\mathcal{H}}$–infers $\mathcal{C}$ iff, for each $c \in \mathcal{C}$, M $TxtItEx^a_{\mathcal{H}}$–infers c.

The resulting learning types $TxtItEx^a_{\mathcal{H}}$ and $TxtItEx^a$ are analogously defined as above.

In the latter definition $M_n(T)$ denotes the $(n+1)$th hypothesis output by M when successively fed the positive presentation T. Thus, it is justified to make the following convention. Let $\sigma = x_0, \dots, x_n$ be any finite sequence of elements over the relevant learning domain.

Moreover, let $\mathcal{C}$ be any concept class over $\mathcal{X}$, and let M be any IIM that iteratively learns $\mathcal{C}$. Then we denote by $M_y(\sigma)$ the $(y+1)$th hypothesis output by M when successively fed σ provided $y \leq n$, and there exists a concept $c \in \mathcal{C}$ with $\sigma^+ \subseteq c$. We adopt this convention to the learning types defined below.

Within the following definition we consider a natural relaxation of iterative learning which we call *bounded example-memory* inference.[4] Now, an IIM M is allowed to memorize an *a priori* bounded number of the examples it already has had access to during the learning process. Again, M defines a sequence $(M_n)_{n\in\mathbb{N}}$ of machines each of which takes as input the output of its predecessor. Thus, a bounded example-memory IIM has to output a hypothesis as well as a subset of the set of examples seen so far.

DEFINITION 4 (*Lange and Zeugmann* (1996)). *Let $k \in \mathbb{N}$, let $\mathcal{C}$ be a concept class, let c be a concept, let $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ be a hypothesis space, and let $a \in \mathbb{N} \cup \{*\}$. An IIM M $TxtBem^k Ex^a_{\mathcal{H}}$–infers c iff for every $T = (x_j)_{j\in\mathbb{N}} \in text(c)$ the following conditions are satisfied:*

(1) *for all $n \in \mathbb{N}$, $M_n(T)$ is defined, where $M_0(T) =_{df} M(x_0) = \langle j_0, S_0\rangle$ such that $S_0 \subseteq T_0^+$ and $|S_0| \leq k$, and for all $n \geq 0$: $M_{n+1}(T) =_{df} M(M_n(T), x_{n+1}) = \langle j_{n+1}, S_{n+1}\rangle$ such that $S_{n+1} \subseteq S_n \cup \{x_{n+1}\}$ and $|S_{n+1}| \leq k$,*

(2) *the j_n in the sequence $(\langle j_n, S_n\rangle)_{n\in\mathbb{N}}$ of M's guesses converges to a $j \in \mathbb{N}$ with $c =^a h_j$.*

Finally, M $TxtBem^k Ex^a_{\mathcal{H}}$–infers $\mathcal{C}$ iff, for each $c \in \mathcal{C}$, M $TxtBem^k Ex^a_{\mathcal{H}}$–infers c.

For every $k \in \mathbb{N}$, the resulting learning types $TxtBem^k Ex^a_{\mathcal{H}}$ and $TxtBem^k Ex^a$ are analogously defined as above. Clearly, by definition, $TxtItEx^a = TxtBem^0 Ex^a$, for all $a \in \mathbb{N} \cup \{*\}$.

Finally, we define learning by *feedback* IIMs. The idea of feedback learning goes back to Wiehagen (1976) who considered it in the setting of inductive inference of recursive functions. Lange and Zeugmann (1996) adapted the concept of feedback learning to inference from positive data. Here, we *generalize* this definition. Informally, a feedback IIM M is an iterative IIM that is additionally allowed to make a bounded number of a particular type of query. In each learning Stage $n+1$, M has access to the actual input x_{n+1}, and its previous guess j_n. However, M is additionally allowed to compute queries from x_{n+1} and j_n. Each query concerns the history of the learning process. Let $k \in \mathbb{N}$; then a *k-feedback learner* computes a k-tuple of elements $(y_1, \ldots, y_k) \in \mathcal{X}^k$ and gets a k-tuple of "YES/NO" answers such that the ith component of the answer is 1, if $y_i \in T_n^+$ and it's 0, otherwise. Hence, M can just ask whether or not k particular strings have already been presented in previous learning stages.

[4] Our definition is a variant of one found in Osherson, Stob and Weinstein (1986) and Fulk *et al.* (1994). Case *et al.* (1997), Subsection 3.5 gives full details about the relation between both notions.

DEFINITION 5. *Let $k \in \mathbb{N}$, let $\mathcal{C}$ be a concept class, let c be a concept, let $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ be a hypothesis space, and let $a \in \mathbb{N} \cup \{*\}$. Moreover, let $Q_k\colon \mathbb{N} \times \mathcal{X} \to \mathcal{X}^k$, be a computable total mapping. An IIM M $TxtFb^k Ex^a_{\mathcal{H}}$–infers c iff for every positive presentation $T = (x_j)_{j\in\mathbb{N}} \in text(c)$ the following conditions are satisfied: below $A^n_k\colon \mathcal{X}^k \to \{0,1\}^k$ denotes the answer to the queries (based on whether the corresponding queried elements appear in T_n or not).*

(1) *for all $n \in \mathbb{N}$, $M_n(T)$ is defined, where $M_0(T) =_{df} M(x_0)$ and for all $n \geq 0$: $M_{n+1}(T) =_{df} M(M_n(T), A^n_k(Q_k(M_n(T), x_{n+1})), x_{n+1})$,*

(2) *the sequence $(M_n(T))_{n\in\mathbb{N}}$ converges to a number j such that $c =^a h_j$ provided that A^n_k truthfully answers the questions computed by Q_k (i.e. the j-th component of $A^n_k(Q_k(M_n(T), x_{n+1}))$ is 1 iff the j-th component of $Q_k(M_n(T), x_{n+1})$ appears in T_n.)*

Finally, M $TxtFb^k Ex^a_{\mathcal{H}}$–infers $\mathcal{C}$ iff there is computable mapping Q_k as described above such that, for each $c \in \mathcal{C}$, M $TxtFb^k Ex^a_{\mathcal{H}}$–identifies c.

The resulting learning types $TxtFb^k Ex^a_{\mathcal{H}}$ and $TxtFb^k Ex^a$ are defined analogously as above.

Finally, we extend Definitions 3 through 5 to the *Fex* case analogously to the generalization of $TxtEx^a_{\mathcal{H}}$ to $TxtFex^a_{\mathcal{H}}$ (cf. Definition 1 and 2). The resulting learning types are denoted by $TxtItFex^a_{\mathcal{H}}$, $TxtBem^k Fex^a_{\mathcal{H}}$, and $TxtFbEx^a_{\mathcal{H}}$. Moreover, for the sake of notation, we shall use the following convention for learning machines corresponding to Definitions 3 through 5 as well as to $TxtItFex^a_{\mathcal{H}}$, $TxtBem^k Fex^a_{\mathcal{H}}$, and $TxtFbEx^a_{\mathcal{H}}$. Let τ be any finite sequence; then we let $M_*(\tau)$ denote $M_{|\tau|-1}(\tau)$.

3. Results

In this section we present our results. In the next subsection, we deal with feedback learning. Our aim is twofold. On the one hand, we investigate the learning power of feedback inference in dependence on k, i.e., the number of strings that may be simultaneously queried. On the other hand, we compare feedback identification with the other learning models introduced, varying the error parameter too (cf. Subsection 3.2). In subsequent subsections we study iterative learning: in Subsection 3.3, the efficacy of redundant hypotheses for iterative learning and, in Subsection 3.4, the iterative learning of finite unions of pattern languages.

3.1. *Feedback Inference*

The next theorem establishes a new infinite hierarchy of successively more powerful feedback learners in dependence on the number k of database queries allowed to be asked simultaneously.[5]

Theorem 2. *$TxtFb^{k-1}Ex \subset TxtFb^{k}Ex$, for all $k \in \mathbb{N}^+$.*

Theorem 3 below not only provides the hierarchy of Theorem 2, but it says that, for suitable concept domains, the feedback learning power of $k+1$ queries of the database, where a *single, correct* grammar is found in the limit, *beats* the feedback learning power of k queries, even when *finitely many grammars* each with *finitely many anomalies* are allowed in the limit.

Theorem 3. *$TxtFb^{k+1}Ex \setminus TxtFb^{k}Fex^{*} \neq \emptyset$, for all $k \in \mathbb{N}$. Moreover this separation can be witnessed by a class consisting of only infinite languages.*

Theorem 3 above nicely contrasts with the following result.

Theorem 4. *Let $\mathcal{L}$ be any* indexed *family consisting of only infinite languages. Then, $\mathcal{L} \in TxtFex$ implies $\mathcal{L} \in TxtFb^{1}Ex$.*

Hence, in the case of *indexed* families of infinite languages, the hierarchy of Theorem 2 collapses for $k \geq 2$; furthermore, again, for *indexed* families of infinite languages, the *expansion* of Gold's model, which not only has unrestricted access to the database, but which also allows *finitely many correct grammars* output in the limit, achieves no more learning power than *feedback* identification with only *one* query of the database.

Next, we compare feedback inference and $TxtFex^{a}$-identification in dependence on the number of anomalies allowed.

Theorem 5. *$TxtFb^{0}Ex^{a+1} \setminus TxtFex^{a} \neq \emptyset$, for all $a \in \mathbb{N}$.*

Hence, for some concept domains, the model of *iterative* learning, where we tolerate $a+1$ anomalies in the single final grammar, is competent, but the expanded Gold model, where we allow unlimited access to the database and finitely many grammars in the limit each with no more than a anomalies, is not. A little extra anomaly tolerance nicely buys, in such cases, no need to remember any past database history or to query it!

3.2. *Feedback Inference versus Bounded Example-Memory Learning*

As promised in the introductory section, the next two theorems show that, for each of these two models of bounded example-memory inference and feedback identification, there are concept class domains where that model is competent and the other is not!

Theorem 6. *$TxtFb^{1}Ex \setminus TxtBem^{k}Fex^{*} \neq \emptyset$, for all $k \in \mathbb{N}$. Moreover this separation can be witnessed by a class consisting of only infinite languages.*

Theorem 6 says that, for suitable concept domains, the feedback learning power of *one* query of the database, where a *single, correct* grammar is found in the limit, *beats* the bounded example-memory learning power of memorizing k database items, even when *finitely many grammars* each with *finitely many anomalies* are allowed in the limit.

Theorem 7. *$TxtBem^{1}Ex \setminus TxtFb^{k}Ex^{*} \neq \emptyset$, for all $k \in \mathbb{N}$. Moreover this separation can be witnessed by a class consisting of only infinite languages.*

Theorem 7 says that, for suitable concept domains, the bounded example-memory learning power of memorizing *one* item from the database history *beats* the feedback learning power of k queries of the database, even when the final grammar is allowed to have *finitely many anomalies.* It is currently open whether $TxtFb^{k}Ex^{*}$ in Theorem 7 can be replaced by $TxtFb^{k}Fex^{*}$.

3.3. *Iterative Learning*

In this subsection we show that *redundancy* in the hypothesis space may considerably increase the learning power of iterative learners. Interestingly, it turns out that, redundancy may serve as a tool exploited by the iterative learner allowing it to *overgeneralize* in learning stages before convergence. Here, overgeneralization refers to the situation in which the learner outputs a description for a proper superset of the target concept. Furthermore, this phenomenon can be already observed at the fairly concrete level of indexed families.

Theorem 8. *There are an indexed family $\mathcal{L}$ and a redundant hypothesis space $\mathcal{H}$ for it such that $\mathcal{L} \in TxtItEx_{\mathcal{H}} \setminus TxtItEx_{\mathcal{L}}$*

3.4. *The Pattern Languages*

The pattern languages (defined two paragraphs below) were formally introduced by Angluin (1980a) and have been widely investigated (cf., e.g., Salomaa (1994a; 1994b), and Shinohara and Arikawa (1995) for an overview). Moreover, Angluin (1980a) proved that the class of all pattern languages is learnable in the limit from positive data. Subsequently, Nix (1983) as well as Shinohara and Arikawa (1995) outlined interesting applications of pattern inference algorithms. For example, pattern language learning algorithms have been successfully applied for solving problems in molecular biology (cf., e.g. Shimozono *et al.* (1994), Shinohara and Arikawa (1995)).

Pattern languages and finite unions of pattern languages turn out to be subclasses of Smullyan's (1961)) elementary formal systems (EFS). Arikawa *et al.* (1992) have shown that EFS can also be treated as

[5] It follows from Fulk *et al.* (1994) and Lange and Zeugmann (1996) that there is an infinite hierarchy of successively more powerful bounded example-memory learners in dependence on the number k of items that can be memorized.

a logic programming language over strings. Recently, the techniques for learning finite unions of pattern languages have been extended to show the learnability of various subclasses of EFS (cf. Shinohara (1991)). From a theoretical point of view, investigations of the learnability of subclasses of EFS are important because they yield corresponding results about the learnability of subclasses of logic programs. Arimura and Shinohara (1994) have used the insight gained from the learnability of EFS subclasses to show that a class of linearly covering logic programs with local variables is identifiable in the limit from only positive data. More recently, using similar techniques, Krishna-Rao (1996) has established the learnability from only positive data of an even larger class of logic programs. These results have consequences for Inductive Logic Programming.[6]

Patterns and pattern languages are defined as follows (cf. Angluin (1980a)). 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 (1969)). The set of all finite non-null strings of symbols from $\mathcal{A}$ is denoted by $\mathcal{A}^+$, i.e., $\mathcal{A}^+ = \mathcal{A}^* \setminus \{\varepsilon\}$, where ε denotes the empty string. By $|\mathcal{A}|$ we denote the cardinality of $\mathcal{A}$. Furthermore, 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. 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} in π. 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.

The length of a string $s \in \mathcal{A}^*$ and of a pattern π is denoted by $|s|$ and $|\pi|$, respectively. By $\#\text{var}(\pi)$ 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 $\#\text{var}(\pi) = k$, then we refer to π as a *k-variable pattern*. Let $k \in \mathbb{N}$, by Pat_k we denote the set of all *k-variable patterns*.

Now let $\pi \in Pat_k$, and let $u_0, \ldots, u_{k-1} \in \mathcal{A}^+$. Then we denote by $\pi[u_0/x_0, \ldots, u_{k-1}/x_{k-1}]$ the string $s \in \mathcal{A}^+$ obtained by substituting u_j for each occurrence of x_j, $j = 0, \ldots, k-1$, in the pattern π. The tuple $(u_0, \ldots, u_{k-1})$ is called *substitution*. For every $\pi \in Pat_k$ we define the *language generated by pattern* π by $L(\pi) = \{\pi[u_0/x_0, \ldots, u_{k-1}/x_{k-1}] \mid u_0, \ldots, u_{k-1} \in \mathcal{A}^+\}$.[7] By PAT_k we denote the set of all *k-variable pattern languages*. Finally, $PAT = \bigcup_{k \in \mathbb{N}} PAT_k$ denotes the set of all *pattern languages* over $\mathcal{A}$.

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

As already mentioned above, the class PAT is $TxtEx_{Pat}$-learnable from positive data (cf. Angluin (1980a)). Subsequently, Lange and Wiehagen (1991) showed PAT to be $TxtItEx_{Pat}$-inferable. As for unions, the first result goes back to Shinohara (1983) who proved the class of all unions of at most two pattern languages to be in $TxtEx_{Pat(2)}$. Wright (1989) extended this result to $PAT(k) \in TxtEx_{Pat(k)}$ for all $k \ge 1$. Moreover, Theorem 4.2 in Shinohara and Arimura's (1996) together with a lemma from Blum and Blum's (1975) shows that $\bigcup_{k \in \mathbb{N}} PAT(k)$ is not $TxtEx_{\mathcal{H}}$-inferable for every hypothesis space $\mathcal{H}$. However, nothing was known previous to the present paper concerning the *incremental* learnability of $PAT(k)$. We resolve this problem by showing the strongest possible result (Theorem 9 below): each $PAT(k)$ is *iteratively* learnable!

PROPOSITION 1.

(1) *For all $L \subseteq \mathcal{A}^+$, $k \in \mathbb{N}^+$, $Club(L, k)$ is finite.*

(2) *If $L \in PAT(k)$, then $Club(L, k)$ is nonempty and contains Q, such that $L(Q) = L$.*

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

LEMMA 1. *Let $k \in \mathbb{N}^+$, and let $L \subseteq \mathcal{A}^+$ be any language. Suppose $T = s_0, s_1, \ldots$ is a text for L. Let L_n below denote $\{s_i \mid i \le n\}$. Then,*

(1) *$Club(L_0, k)$ can be effectively obtained from s_0, and $Club(L_{n+1}, k)$ can be effectively obtained from $Club(L_n, k)$ and s_{n+1} (* note the iterative nature *).*

(2) *The sequence $Club(L_0, k)$, $Club(L_1, k)$, ... converges to $Club(L, k)$.*

Proof. (1): Fix $k \ge 1$, and suppose $T = s_0, s_1 \ldots, s_n, s_{n+1}, \ldots$ is a text for L. Furthermore, let $\mathcal{S}_0 = \{\{\pi\} \mid s_0 \in L(\pi)\}$. We proceed inductively; for $n \ge 0$, we define $\mathcal{S}'_{n+1} = \{Q \in \mathcal{S}_n \mid s_{n+1} \in L(Q)\} \cup \{Q \cup \{\pi\} \mid Q \in \mathcal{S}_n \wedge s_{n+1} \notin L(Q) \wedge |Q| < k \wedge s_{n+1} \in L(\pi)\}$, and then $\mathcal{S}_{n+1} = \{Q \in \mathcal{S}'_{n+1} \mid (\forall Q' \in \mathcal{S}'_{n+1})[Q' \not\subset Q]\}$.

[6] We are grateful to Arun Sharma for bringing to our fuller attention these potential applications to ILP of learning special cases of pattern languages and finite unions of pattern languages.

[7] We study so-called *non-erasing* substitutions. It is also possible to consider *erasing* substitutions where variables may be replaced by empty strings, leading to a different class of languages (cf. Filé (1988)).

Note that $\mathcal{S}_0$ can be effectively obtained from s_0, since every pattern π with $s_0 \in L(\pi)$ must satisfy $|\pi| \le |s_0|$. Thus, there are only finitely many candidate patterns π with $s_0 \in L(\pi)$ which can be effectively constructed. Since membership is uniformly decidable, we are done. Furthermore, using the same argument, $\mathcal{S}_{n+1}$ can be effectively obtained from $\mathcal{S}_n$ and s_{n+1}, too. Also it is easy to verify, by induction on n, that $\mathcal{S}_n = \mathit{Club}(L_n, k)$. Thus, (1) is satisfied.

(2): Consider a tree $\mathcal{T}$ formed mimicking the above construction of $\mathcal{S}_n$ as follows. The nodes of $\mathcal{T}$ will be labeled either "empty" or by a pattern. The root is labeled "empty". The children of any node in the tree (and their labels) are defined as follows. Suppose the node, v, is at distance n from the root. Let Q denote the set of patterns formed by collecting the labels on the path from root to v (ignoring the "empty" labels). Children of v are defined as follows:

(a) If $s_n \in L(Q)$, then v has only one child with label "empty."

(b) If $s_n \notin L(Q)$, and $|Q| = k$, then v has no children.

(c) If $s_n \notin L(Q)$, and $|Q| < k$, then v has children with labels π, where $s_n \in L(\pi)$ (the number of children is equal to the number of patterns π such that $s_n \in L(\pi)$).

Suppose $\mathcal{U}_n = \{Q \mid (\exists v$ at a distance $n+1$ from root $)[Q =$ the set of patterns formed by collecting the labels on the path from root to v (ignoring the "empty" labels) $]\}$. Then it is easy to verify using induction that $\mathcal{S}_n = \{Q \in \mathcal{U}_n \mid (\forall Q' \in \mathcal{U}_n)[Q' \not\subset Q]\}$.

Since the number of non-empty labels on any path of the tree is bounded by k, using König's Lemma we have that the number of nodes with non empty label must be finite. Thus the sequence $\mathcal{U}_0, \mathcal{U}_1, \ldots$ converges. Hence the sequence $\mathcal{S}_0 = \mathit{Club}(L_0, k), \mathcal{S}_1 = \mathit{Club}(L_1, k), \ldots$ converges, to say $\mathcal{S}$. Now, for all $Q \in \mathcal{S}$, for all n, $L_n \subseteq L(Q)$. Thus, for all $Q \in \mathcal{S}$, $L \subseteq L(Q)$. Also, for all $Q \in \mathcal{S}$ and $Q' \subset Q$, for all but finitely many n, $L_n \not\subseteq L(Q')$. Thus for all $Q \in \mathcal{S}$ and $Q' \subset Q$, $L \not\subseteq L(Q')$. It follows that $\mathcal{S} = \mathit{Club}(L, k)$. Thus, Part (2) of Lemma follows. ∎

Theorem 9. *$PAT(k) \in TxtItEx$ for all $k \ge 1$.*

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

Let $L \in PAT(k)$, and let $T = s_0, s_1, \ldots$ be a text for L. The desired IIM M is defined as follows. Initially, we set $M_0(T) = M(s_0) = \mathrm{pd}(g(\mathit{Club}(\{s_0\}, k)), \mathrm{cn}(\mathit{Club}(\{s_0\}, k)))$. Furthermore, for all $n > 0$, we set $M_{n+1}(T) = M(M_n(T), s_{n+1}) = \mathrm{pd}(g(\mathit{Club}(\{s_0, \ldots, s_n\}, k)), \mathrm{cn}(\mathit{Club}(\{s_0, \ldots, s_n\}, k)))$. Using Lemma 1 it is easy to verify that $M_{n+1}(T) = M(M_n(T), s_{n+1})$ can be obtained effectively from $M_n(T)$ and s_{n+1}. Thus, M *TxtItEx*-identifies $PAT(k)$. ∎

4. Conclusions and Future Directions

We studied refinements of concept learning in the limit from positive data that considerably restrict the accessibility of input data. Our research derived its motivation from the rapidly emerging field of data mining. Here, huge data sets are a fact of life, and any practical learning system has to deal with the high cost of querying a huge database. Given this, a systematic study of incremental learning is important for gaining a better understanding of how *different* restrictions to the accessibility of input data do affect the resulting *inference capabilities* of the corresponding learning models. The study undertaken extends, in various directions, previous work done by Osherson *et al.* (1986), Fulk *et al.* (1994) and Lange and Zeugmann (1996).

First, the class of all unions of at most k pattern languages has been shown to be iteratively learnable. Moreover, we proved redundancy in the hypothesis space to be a resource extending the learning power of iterative learners in fairly concrete contexts. As a matter of fact, the hypothesis space used in showing Theorem 9 is highly redundant, too. Moreover, we strongly conjecture this redundancy to be necessary, i.e., no *iterative learner* can identify all unions of at most k pattern languages with repsect to a 1–1 hypothesis space. Clearly, once the principal learnability has been established, complexity becomes a central issue. Thus, further research should address the problem of designing *time efficient* iterative learners for $PAT(k)$. This problem is even unsolved for $k = 1$. On the one hand, Lange and Wiehagen (1991) designed an iterative pattern learner having *polynomial update time*. Nevertheless, the *expected total learning time*, i.e., the overall time needed until convergence is exponential in the number of different variables occurring in the target pattern for inputs drawn with respect to the uniform distribution (cf. Zeugmann (1996)).

Second, we considerably generalized the model of feedback inference introduced in Lange and Zeugmann (1996) by allowing the feedback learner to ask k queries simultaneously. Though at first glance it may seem that asking simultaneously for k elements and memorizing k carefully selected data items may be traded one to another, we rigorously proved the resulting learning types to be advantageous in very different scenarios (cf. Theorem 6 and 7). Consequently, there is no unique way to design superior incremental learning algorithms. Therefore, the comparison of k-feedback learning and k-bounded example-memory inference deserves special interest, and future research should address the problem under what circumstances which model is preferable. Characterizations may serve as suitable tool for accomplishing this goal (cf., e.g., Angluin (1980b), Blum and Blum (1975), Zeugmann *et al.* (1995)), and Baliga *et al.* (1996).

Furthermore, for concept learning from extraordinarily large databases, by Theorems 6 and 7, *in some cases, but not in others*, one can avoid the high cost of querying the whole database by remembering a small but judiciously chosen "cache" of database items. We would like to find generally useful techniques (or characterizations as mentioned above) for positing what to store in inexpensive database-cache memory and/or which minimal set of expensive queries to use.

Feedback identification and bounded example-memory inference have been considered in the general context of classes of recursively enumerable concepts rather than uniformly recursives ones as done in Lange and Zeugmann (1996). As our Theorem 4 shows, there are subtle differences. Furthermore, a closer look at the proof of Theorem 4 directly yields the interesting problem whether or not allowing a learner to ask simultaneously k questions instead of querying one data item per time may speed-up the learning process.

A further generalization can be obtained by allowing a k-feedback learner to ask its queries *sequentially*, i.e., the next query is additionally allowed to depend on the answers to its previous queries. Interestingly, our theorems hold if we use this definition for k-feedback learning in place of the *parallel* queries one we actually do use. It is, however, currently open whether this possible change in the meaning of k-feedback learning enables learning of classes not learnable using our original definition.

Next, we discuss possible extensions of the incremental learning model considered. A natural relaxation of the constraint to fix k *a priori* can be obtained by using the notion of constructive ordinals as done by Freivalds and Smith (1993) for mind changes. Intuitively, the paramenter k is now specified to be a constructive ordinal, and the bounded example-memory learner as well as a feedback machine can change their mind of how many data items to store and to ask for, respectively, in dependence on k. Furthermore, future research should examine a hybrid model which permits *both* memorizing a database-cache of k_1 items from the database *and* k_2 queries of the database, where, again, k_1 and k_2 may be specified as constructive ordinals.

Moreover, it would also be interesting to extend this and the topics of the present paper to probabilistic learning machines. This branch of learning theory has recently seen as variety of surprising results (cf., e.g., Jain and Sharma (1995), Meyer (1995; 1997)), and thus, one may expect further interesting insight into the power of probabilism by combining it with incremental learning.

Finally, while the research presented in the present paper clarified what are the strength and limitations of incremental learning, further investigations are necessary dealing with the impact of incremental inference on the complexity of the resulting learner. First results along this line are established in Wiehagen and Zeugmann (1994), and we shall see what the future brings concerning this interesting topic.

References

Angluin D. 1980a. Finding patterns common to a set of strings. *J. Comput. System Sci.*, 21:46–62.

Angluin D. 1980b. Inductive inference of formal languages from positive data. *Inform. and Control*, 45:117–135.

Arikawa S., Shinohara T., and Yamamoto A. 1992. Learning elementary formal systems. *Theoret. Comput. Sci.*, 95:97–113.

Arimura H. and Shinohara T. 1994. Inductive inference of Prolog programs with linear data dependency from positive data. In *Proc. Information Modeling and Knowledge Bases V*, pp. 365–375, IOS Press.

Baliga G., Case J., and Jain S. 1996. Synthesizing enumeration techniques for language learning. In *Proc. 9th Annual Conference on Computational Learning Theory*, pp. 169–180. ACM Press.

Blum M. 1967. A machine independent theory of the complexity of recursive functions. *J. ACM*, 14:322–336.

Brachman R. and Anand T. 1996. The process of knowledge discovery in databases: A human centered approach. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy, Eds., *Advances in Knowledge Discovery and Data Mining*, pp. 37–58, AAAI Press.

Blum M. and Blum L. 1975. Toward a mathematical theory of inductive inference. *Inform. and Control* 28:125–155.

Case J. 1988. The power of vacillation. In *Proc. 1st Workshop on Computational Learning Theory*, pp. 133–142, Morgan Kaufmann Publishers Inc.

Case J. 1996. The power of vacillation in language learning. Technical Report LP-96-08, Logic, Philosophy and Linguistics Series of the Institute for Logic, Language and Computation, University of Amsterdam. To appear revised in *SIAM J. Computing.*

Case J., Jain S., Lange S. and Zeugmann T. 1997. Incremental concept learning for bounded data mining. Technical Report DOI-TR-136, Department of Informatics, Kyushu University, Fukuoka, Japan (URL: ftp://ftp.i.kyushu-u.ac.jp/tr/trcs136.ps.gz).

Case J. and Smith C.H. 1983. Comparison of identification criteria for machine inductive inference. *Theoret. Computer Sci.*, 25:193–220.

Fayyad U.M., Djorgovski S.G., and Weir N. 1996a. Automating the analysis and cataloging of sky surveys. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy, Eds., *Advances in*

Knowledge Discovery and Data Mining, pp. 471–494, AAAI Press.

Fayyad U.M., Piatetsky-Shapiro G., and Smyth P. 1996b. From data mining to knowledge discovery: An overview. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy, Eds., *Advances in Knowledge Discovery and Data Mining*, pp. 1–34, AAAI Press.

Filé G. 1988. The relation of two patterns with comparable languages. In *Proc. 5th Ann. Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science 294, pp. 184–192, Springer-Verlag.

Freivalds R. and Smith C.H. 1993. On the role of procrastination for machine learning. *Inform. and Comput.*, 107:237–271.

Fulk M., Jain S. and Osherson D.N. 1994. Open problems in systems that learn. *J. Comput. System Sci.*, 49:589–604.

Gold E.M. 1967. Language identification in the limit. *Inform. and Control*, 10:447–474.

Hopcroft J.E. and Ullman J.D. 1969. *Formal Languages and their Relation to Automata.* Addison-Wesley Publishing Company.

Jain S. and Sharma A. 1995. On identification by teams and probabilistic machines. In *Algorithmic Learning for Knowledge-Based Systems*, Lecture Notes in Artificial Intelligence 961, pp. 108–145, Springer-Verlag.

Kloesgen W. 1995. Efficient discovery of interesting statements in databases. *J. Intell. Inform. Systems*, 4:53–69.

Lange S. and Wiehagen R. 1991. Polynomial-time inference of arbitrary pattern languages. *New Generation Computing*, 8:361–370.

Lange S. and Zeugmann T. 1996. Incremental learning from positive data. *J. Comput. System Sci.*, 53:88–103.

Matheus C.J., Piatetsky-Shapiro G., and McNeil D. 1996. Selecting and reporting what is interesting. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy, Eds., *Advances in Knowledge Discovery and Data Mining*, pp. 495–515, AAAI Press.

Meyer L. 1995. Probabilistic language learning under monotonicity constraints. In *Proc. 6th Int. Workshop on Algorithmic Learning Theory*, Lecture Notes in Artificial Intelligence 997, pp. 169–184, Springer-Verlag.

Meyer L. 1997. Monotonic and dual monotonic probabilistic language learning of indexed families with high probability. In *Proc. 3rd European Conference on Computational Learning Theory*, Lecture Notes in Artificial Intelligence 1208, pp. 66–78, Springer-Verlag.

Nix R.P. 1983. Editing by examples. Technical Report 280, Department of Computer Science, Yale University, New Haven, USA.

Osherson D., Stob M., and Weinstein S. 1986. *Systems that learn, An introduction to learning theory for cognitive and computer scientists.* MIT Press.

Rao K. 1996. A class of Prolog programs inferable from positive data. In *Proc. 7th Int. Workshop on Algorithmic Learning Theory*, Lecture Notes in Artificial Intelligence 1160, pp. 272–284, Springer-Verlag.

Rogers H. 1967. *Theory of Recursive Functions and Effective Computability.* McGraw Hill, New York. Reprinted, MIT Press 1987.

Salomaa A. 1994a. Patterns. EATCS Bull. 54:46–62.

Salomaa A. 1994b. Return to patterns. EATCS Bull. 55:144–157.

Shimozono S., Shinohara A., Shinohara T., Miyano S., Kuhara S., and Arikawa S. 1994. Knowledge acquisition from amino acid sequences by machine learning system BONSAI. *Trans. Information Processing Society of Japan*, 35:2009–2018.

Shinohara T. 1983. Inferring unions of two pattern languages. *Bull. Informat. Cybern.*, 20:83–88.

Shinohara T. 1991. Inductive inference of monotonic formal systems from positive data. *New Generation Computing*, 8:371–384.

Shinohara T. and Arikawa A. 1995. Pattern inference. In *Algorithmic Learning for Knowledge-Based Systems*, Lecture Notes in Artificial Intelligence 961, pp. 259–291, Springer-Verlag.

Shinohara S. and Arimura H. 1996. Inductive inference of unbounded unions of pattern languages from positive data. In *Proc. 7th Int. Workshop on Algorithmic Learning Theory*, Lecture Notes in Artificial Intelligence 1160, pp. 256–271, Springer-Verlag.

Smullyan R. 1961. *Theory of Formal Systems, Annals of Mathematical Studies, No. 47.* Princeton, NJ.

Wiehagen R. 1976. Limes-Erkennung rekursiver Funktionen durch spezielle Strategien. *J. Inform. Process. Cybern. (EIK)*, 12:93–99.

Wiehagen R. and Zeugmann T. 1994. Ignoring data may be the only way to learn efficiently. *J. Exp. & Theoret. Artif. Intellig.*, 6:131–144.

Wright K. 1989. Identification of unions of languages drawn from an identifiable class. In *Proc. 2nd Workshop on Computational Learning Theory*, pp. 328–333, Morgan Kaufmann Publishers Inc.

Zeugmann T. 1996. 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.* to appear.

Zeugmann T. and Lange S. 1995. A guided tour across the boundaries of learning recursive languages. In *Algorithmic Learning for Knowledge-Based Systems*, Lecture Notes in Artificial Intelligence 961, pp. 190–258, Springer-Verlag.

Zeugmann T., Lange S., and Kapur S. 1995. Characterizations of monotonic and dual monotonic language learning. *Inform. and Comput.* 120:155–173.