Too Much Information Can be Too Much for Learning Efficiently

Rolf Wiehagen[*]
Humboldt–Universität
Institut für Theoretische Informatik
PSF 1297
O–1086 Berlin
wiehagen@hubinf.uucp

Thomas Zeugmann
TH Darmstadt
Institut für Theoretische Informatik
Alexanderstr. 10
W–6100 Darmstadt
zeugmann@iti.informatik.th-darmstadt.de

Abstract

In designing learning algorithms it seems quite reasonable to construct them in a way such that all data the algorithm already has obtained are correctly and completely reflected in the description the algorithm outputs on these data. However, this approach may totally fail, i.e., it may lead to the unsolvability of the learning problem, or it may exclude any efficient solution of it. In particular, we present a natural learning problem and prove that it can be solved in polynomial time if and only if the algorithm is allowed to ignore data.

1. Introduction

The phenomenon of learning has attracted much attention of researchers in various fields. When dealing with learning computer scientists are mainly interested in studying the question whether or not learning problems may be solved algorithmically. Nowadays, algorithmic learning theory is a rapidly emerging science (cf. Angluin and Smith (1983, 1987), Osherson, Stob and Weinstein (1986) and the references therein). Nevertheless, despite the enormous progress having been made since the pioneering papers of Solomonoff (1965) and of Gold (1965, 1967), there are still many problems that deserve special attention. The global question we shall deal with may be posed as follows: Are all data of equal importance a learning algorithm is fed with?

First we study this question in the setting of inductive inference. Then we ask whether the insight obtained may be important when one has to solve learning problems that are somehow closer to potential applications. But let us explain all these things in some more detail.

[*] On leave at TH Darmstadt

One main problem of algorithmic learning theory consists in synthesizing "global descriptions" for the objects to be learnt from examples. Thus, one goal is the following. Let f be any computable function from N into N. Given more and more examples $f(0), f(1), ..., f(n), ...$ a learning strategy is required to produce a sequence of hypotheses $h_0, h_1, ..., h_n, ...$ the limit of which is a correct global description of the function f, i.e., a program that computes f. Since at any stage n of this learning process the strategy knows exclusively the examples $f(0), f(1), ..., f(n)$, it seems reasonable to construct the hypothesis h_n in a way such that for any $x \leq n$ the "hypothesis function" g described by h_n is defined and computes the value $f(x)$. Such a hypothesis is called *consistent.* In other words, a hypothesis is consistent if and only if all information obtained so far about the unknown object is completely and correctly encoded in this hypothesis. Otherwise, a hypothesis is said to be *inconsistent.* Consequently, if the hypothesis h_n above is inconsistent, then there must be an $x \leq n$ such that $g(x) \neq f(x)$. Note that there are two possible reasons for g to differ from f on argument x; namely, $g(x)$ may be not defined, or the value $g(x)$ is defined and does not equal $f(x)$. Hence, if a hypothesis is inconsistent then it is not only wrong at all but it is wrong on an argument for which the learning strategy does already know the correct value. At first glance we are tempted to totally exclude strategies producing inconsistent hypotheses ¿from our considerations. It might seem that *consistent strategies*, i.e., strategies that produce always consistent hypotheses, are the only reasonable learning devices.

Surprisingly enough this is a misleading impression. As it turns out, in a sense learning seems to be *the art of knowing what to overlook.* Barzdin (1974) first announced that there are classes of recursive functions that can be inferred in the limit but only by strategies working inconsistently. This result directly yields the following questions:

(1) Why does it make sense to output inconsistent hypotheses?
(2) What kind of data, if any, the strategy should overlook?

As we shall see below the first question finds its preliminary answer in the fact that, roughly speaking, there is no algorithm detecting whether or not a hypothesis is consistent. Consequently, in general a strategy has no chance to effectively verify the consistency of its previous guess with the new data it has been fed with. On the other hand, a strategy cannot overcome this drawback by simply searching any consistent hypothesis, since it has to converge in the limit, too. Therefore, in order to be really successful, intuitively speaking, a strategy cannot take care whether all the information it is provided with is actually correctly reflected by its current hypotheses. Answering the second question is more complicated. However, an intuitively satisfying answer is provided by a characterization of identification in the limit in terms of computable numberings (cf. Wiehagen (1978b), Theorem 8). This theorem actually states that a class U of recursive functions can be learnt in the limit iff there are a space of hypotheses containing for each function at least one program, and a computable "discrimination" function d such that for any two programs i and j the value $d(i, j)$ is an upper bound for an argument x on which program i behaves different than program j does. The key observation used in constructing the strategy that infers any function from U is the following. Let i be the strategy's last guess and let $f(0), ..., f(n)$ be the data now fed to it. If the strategy finds a program j such that for all inputs $x \leq d(i, j)$ its output equals $f(x)$, then program i *cannot be a correct* *one* for function f. Then the strategy changes its mind from i to $i+1$. In other words, the strategy uses the data it receives for the purpose to find a *proof* for the *incorrectness* of its actual hypothesis via some global property the space of all hypotheses possesses.

Summarizing the above discussion we see that the main reason for the superiority of inconsistent strategies in the setting of inductive inference of recursive functions is caused by the undecidability of consistency. However, it remained open whether this phenomenon is of fundamental epistemological importance but of almost no pratical relevance. Dealing with the latter problem requires a different approach. It might be well conceivable that consistency is always decidable if one restricts itself to learning problems that are of interest with respect to potential applications. Consequently, the superiority of inconsistent strategies in this setting, if any, can only be established in terms of complexity theory. What we present in the sequel is a partial solution to this problem. As it turned out, there are "natural" learning problems having the following properties: First, consistency is decidable, and second, they can be solved by a polynomial–time strategy if and only if the strategy may work inconsistently.

Hence the inconsistency phenomenon does survive also in domains where consistency *is* decidable. Moreover, the reason for the eventual superiority of inconsistent strategies is in both settings in some sense the same. In both cases the learning algorithms cannot handle the consistency problem. On the one hand this inability has been caused by the provable absense of any algorithm solving it, while, on the other hand, the consistency problem is computationally intractable, provided $\mathcal{P} \neq \mathcal{NP}$. As far as we know the result above is the first one proving the existence of learning problems that cannot be solved in polynomial time by any consistent strategy in a setting where consistency is decidable. Moreover, in our opinion it strongly recommends to take seriously inconsistent learning strategies into consideration. This requires both, the elaboration of "intelligent" inconsistent techniques as well as finding criteria with the help of which one can decide whether or not inconsistent strategies are appropriate. The inconsistent technique we have applied consists in overlooking or ignoring data unnecessary for the strategy in order to fulfil its learning task. Of course, this might not be the only such technique.

The paper is structured as follows. Section 2 presents notation and definitions. Section 3 deals with the inconsistency phenomenon in the setting of inductive inference of recursive functions. The problem whether the inconsistency phenomenon is of any relevance in the world of polynomial time computability is affirmatively exemplified in Section 4. In Section 5 we discuss the results obtained and present open problems. All references are given in Section 6.

2. Preliminaries

Unspecified notations follow Rogers (1967). $N = \{0, 1, 2, ...\}$ denotes the set of all natural numbers. The set of all finite sequences of natural numbers is denoted by N^*. The classes of all partial recursive and recursive functions of one, two arguments over N are denoted by P, P^2, R, and R^2, respectively. $R_{0,1}$ denotes the set of all $0-1$ valued recursive functions. Sometimes it will be suitable to identify a recursive function with the sequence of its values, e.g., let $\alpha = (a_0, ..., a_k) \in N^*$, $j \in N$, and $p \in R_{0,1}$; then we write αjp to denote the function f for which $f(x) = a_x$, if $x \leq k$, $f(k+1) = j$, and $f(x) = p(x-k-1)$, if $x \geq k+2$. Furthermore, let $g \in P$; we write $\alpha \sqsubset$ iff α is a prefix of the sequence of values associated with g, i.e., for any $x \leq k$, $g(x)$ is defined and $g(x) = a_x$. If $U \subseteq R$, then we denote by $[U]$ the set of all finite prefixes of functions from U.

Any function $\psi \in P^2$ is called a numbering. Moreover, let $\psi \in P^2$, then we write ψ_i instead of $\lambda x \psi(i,x)$ and set $P_\psi = \{\psi_i \mid i \in N\}$ as well as $R_\psi = P_\psi \cap R$. Consequently, if $f \in P_\psi$, then there is a number i such that $f = \psi_i$. A numbering $\varphi \in P^2$ is called a Gödel numbering (cf. Rogers (1967)) iff $P_\varphi = P$, and for any numbering $\psi \in P^2$, there is a $c \in R$ such that $\psi_i = \varphi_{c(i)}$ for all $i \in N$. *Göd* denotes the set of all Gödel numberings.

Using a fixed encoding $\langle ... \rangle$ of N^* onto N we write f^n instead of $\langle ((f(o), ..., f(n))) \rangle$, for any $n \in N$, $f \in R$. Now we define some concepts of learning.

Definition 1. (Gold, 1965)
Let $U \subseteq R$ and let $\psi \in P^2$. The class U is said to be learnable in the limit with respect to ψ iff there is a strategy $S \in P$ such that for any function $f \in U$,

(1) for any $n \in N$, $S(f^n)$ is defined,

(2) there is a $j \in N$ such that $\psi_j = f$ and, for almost all n, $S(f^n) = j$.

If U is learnable in the limit with respect to ψ by strategy S we write $U \in LIM_\psi(S)$. Let $LIM_\psi = \{U \mid U$ is learnable in the limit w.r.t. $\psi\}$, and let $LIM = \bigcup_{\psi \in P^2} LIM_\psi$.

Note that $LIM_\varphi = LIM$ for any Gödel numbering φ. Next we formally define consistent learning.

Definition 2. (Barzdin, 1974)
Let $U \subseteq R$ and let $\psi \in P^2$. The class U is called consistently learnable in the limit with respect to ψ iff there is a strategy $S \in P$ such that

(1) $U \in LIM_\psi(S)$

(2) $\psi_{S(f^n)}(x) = f(x)$ for all $n \in N$ and $x \leq n$.

$CONS_\psi(S)$, $CONS_\psi$ and $CONS$ are defined analogously as above.

Intuitively speaking, a consistent strategy does correctly reflect all the data it has already seen. Note that $CONS_\varphi = CONS$ for any Gödel numbering φ. If a strategy does not always work consistently, we call it inconsistent.

3. The Inconsistency Phenomenon

The inconsistency phenomenon has been discovered independently by Barzdin (1974) and the Blums (1975). They observed that there are classes of recursive functions that are inferrable in the limit but which cannot be learnt by any consistent strategy. Since both papers do not contain a proof of this assertion, we present here a proof from Wiehagen (1976) actually showing a somewhat stronger result than the one formulated in the next theorem. We shall discuss this issue below.

Theorem 1. (Barzdin, 1974)
$CONS \subset LIM$

Proof. Let $U_\varphi = \{f \in R \mid f = \alpha j p,\ \alpha \in N^*,\ j \geq 2,\ p \in R_{0,1},\ \varphi_j = f\}$, where $\varphi \in Göd$. Obviously, $U_\varphi \in LIM(S)$, where $S(f^n)$ is equal to the last value $f(x) \geq 2$ from $(f(0), ..., f(n))$ and 0, if no such value exists. For the purpose to prove that $U_\varphi \notin CONS$ we need the following claim.

Claim: Let $\varphi \in Göd$. For any $\alpha \in N^*$, there is a $f \in U_\varphi$ such that $\alpha \sqsubset f$.

Indeed, by an implicit use of the Recursion Theorem (cf. Rogers (1967)) it is easy to see that for any $\alpha \in N^*$ and any $p \in R_{0,1}$ there is a $j \geq 2$ such that $\varphi_j = \alpha j p$.

Now, suppose the converse, i.e., there is a strategy $S \in P$ such that $U_\varphi \in CONS_\varphi(S)$. By the claim we get that $S \in R$ and that for any $\alpha \in N^*$, $\alpha \sqsubset \varphi_{S(\alpha)}$. Thus, on any $\alpha \in N^*$, S always produces a consistent guess. Then, again by an implicit use of the Recursion Theorem, let $j \geq 2$ be any φ–number of the function f defined as follows: $f(0) = j$, and for any $n \in N$,

$$f(n+1) = \begin{cases} 0 & , \quad if \quad S(f^n 0) \neq S(f^n) \\ 1 & , \quad if \quad S(f^n 0) = S(f^n)\ and S(f^n 1) \neq S(f^n) \end{cases}$$

In accordance with the claim and the assumption that S works consistently one straightforwardly verifies that $S(f^n 0) \neq S(f^n)$ or $S(f^n 1) \neq S(f^n)$ for any $n \in N$. Therefore the function f is everywhere defined and we have $f \in U_\varphi$. On the other hand, the strategy S changes its mind infinitely often when successively fed with f, a contradiction to $U_\varphi \in CONS_\varphi(S)$.

q.e.d.

Note that the class U_φ from the proof of Theorem 1 is even *iteratively* learnable in the limit. We call a class U of recursive functions iteratively learnable iff there is a strategy that learns any $f \in U$ as follows: In step n the strategy exclusively gets its previous guess produced in step $n-1$ as well as the new value $f(n)$. By IT we denote the collection of all classes U which can be learnt iteratively. In Wiehagen (1976) $CONS \subset IT \subset LIM$ has been proved. Recent papers give new evidence for the power of iterative learning (cf. e.g. Porat and Feldman (1988), Lange and Wiehagen (1991), Lange and Zeugmann (1992)).

A closer look to the proof above shows that, in general, a strategy attempting to consistently learn functions has to overcome two difficulties. First, it should avoid to change too often its current guess that is eventually no longer consistent, to a definitely consistent hypothesis, since this behavior may force the strategy to diverge. Second, trusting that its current guess is consistent may eventually lead to an actually inconsistent hypothesis, since the strategy cannot effectively prove consistency. Indeed, it turns out that a class U is consistently learnable iff there is a suitable space of hypotheses ψ such that the consistency problem restricted to U and ψ is effectively decidable. More precisely, let $U \subseteq R$ and let ψ be any numbering. We say that U–consistency is decidable with respect to ψ iff there is a predicate $cons \in P^2$ such that for any $\alpha \sqsubset U$ and any $i \in N$, $cons(\alpha, i)$ is defined and $cons(\alpha, i) = 1$ if and only if $\alpha \sqsubset \psi_i$.

We say that consistency with respect to ψ is decidable iff there is a predicate $cons \in R^2$ such that for any $\alpha \in N^*$ and for any $i \in N$, $cons(\alpha, i) = 1$ if and only if $\alpha \sqsubset \psi_i$.

Theorem 2. *$U \in CONS$ iff there is a numbering $\psi \in P^2$ such that*

(1) $U \subseteq P_\psi$

(2) U–consistency with respect to ψ is decidable.

Theorem 2 is a consequence of Theorem 9 from Wiehagen (1978b). Note that for an arbitrary Gödel numbering, U-consistency is undecidable for any non–empty class $U \subseteq R$.

Furthermore, for many classes $U \in CONS$, the halting problem with respect to the numbering ψ from Theorem 2 is *undecidable*. More exactly, let $NUM = \{U \mid U \subseteq R$ and there is $\psi \in R^2$ such that $U \subseteq P_\psi\}$ denote the family of all classes of recursive functions that are contained in some recursively enumerable class of recursive functions. Then we know that $NUM \subset CONS$ (cf. Wiehagen (1976)). Furthermore, it turns out that for any class $U \notin NUM$, there are only spaces of hypotheses ψ such that $U \subseteq P_\psi$ implies the undecidabilty of the halting problem with respect to ψ (cf. Lemma 3 below). Moreover, it is justified to call the latter property a "bad" one, since conversely the decidability of the halting problem easily implies the consistent learnability in the limit of *all* functions from R_ψ with respect to ψ.

Lemma 3. *Let $U \subseteq R$ and let $U \notin NUM$. Then, for any numbering $\psi \in P^2$ satisfying $U \subseteq P_\psi$, the halting problem with respect to ψ is undecidable.*

Proof. Let $U \subseteq R$ and let $U \notin NUM$. Furthermore, let $\psi \in P^2$ be any numbering such that $U \subseteq P_\psi$. Suppose, the halting problem with respect to ψ is decidable. Hence there is a function $h \in R^2$ such that for any $i, x \in N$, $h(i, x) = 1$ iff $\psi_i(x)$ is defined. Then define a numbering $\tilde{\psi}$ by effectively filling out the "gaps" in ψ as follows:

$$\tilde{\psi}_i(x) = \begin{cases} \psi_i(x) & , \quad if \quad h(i,x) = 1 \\ 0 & , \quad otherwise \end{cases}$$

Obviously, for any $i \in N$, if $\psi_i \in R$ then $\tilde{\psi}_i = \psi_i$. Hence $U \subseteq P_{\tilde{\psi}}$. However, $\tilde{\psi} \in R^2$ and consequently, $U \subseteq P_{\tilde{\psi}}$ does imply $U \in NUM$, a contradiction.

q.e.d.

Therefore, Theorem 2 offers a possibility to compensate the undecidability of the halting problem with respect to ψ by the decidability of U–consistency. One can even show that there are classes $U \in CONS \setminus NUM$ such that $[U] = N^*$. By Theorem 2 and Lemma 3 this yields the existence of numberings ψ with undecidable halting problem and decidable consistency problem. Note that the latter assertion is a pure numbering theoretical result proved by learning theoretical observations (though it can also be proved directly).

Lemma 4. *There is a class $U \subseteq R$ such that*

(1) $U \in CONS$

(2) $U \notin NUM$

(3) $[U] = N^*$

Sketch of proof. Let (φ, Φ) be any complexity measure (cf. Blum (1967)). We modify Φ to $\hat{\Phi}$ in a way such that

1. $(\varphi, \hat{\Phi})$ is again a complexity measure,

2. for any $\alpha \in N^*$, there is an $i \in N$ such that $\varphi_i \in R$ and $\alpha \sqsubset \hat{\Phi}_i$.

Now let $U = \{\hat{\Phi}_i \mid \varphi_i \in R\}$. Then $U \in CONS_{\hat{\Phi}}$, since for any $i, x, y \in N$ the predicate "$\hat{\Phi}_i(x) = y$" is uniformly decidable. Furthermore, $U \notin NUM$, since otherwise $R \in NUM$. Finally, $[U] = N^*$ by construction.

q.e.d.

Corollary 5. *There is a numbering ψ with $[R_\psi] = N^*$ such that*

(1) The halting problem with respect to ψ is undecidable.

(2) The consistency problem with respect to ψ is decidable.

Proof. Let $U \subseteq R$ be a class in accordance with Lemma 4. Furthermore, let ψ be a numbering chosen accordingly to Theorem 2. Then (1) is an immediate consequence of Lemma 3, and (2) follows from Theorem 2, assertion (2), taking into account that, by construction, $[R_\psi] = N^*$.

q.e.d.

Finally in this section we want to point out that a natural weakening of consistency does not suffice to learn any class inferrable in the limit.

Definition 3. (Wiehagen, 1978a)
Let $U \in R$, $\psi \in P^2$. U is called conformly learnable in the limit with respect to ψ iff there is a strategy $S \in P$ such that

(1) $U \in LIM_\psi(S)$,

(2) for any function $f \in U$, any $n, x \in N$ such that $x \leq n$, either $\psi_{S(f^n)}(x) = f(x)$ or $\psi_{S(f^n)}(x)$ is not defined.

Intuitively speaking, a conform strategy is never allowed to output a hypothesis that indeed computes a wrong value at an argument x which is less than the length of the initial segment seen so far. By $CONF_\psi$ we denote the collection of classes $U \subseteq R$ that are conformly identifiable with respect to ψ. Finally, let $CONF$ denote the union of all families $CONF_\psi$ where $\psi \in P^2$. The following theorem is due to Wiehagen (1978a).

Theorem 6.
$CONS \subset CONF \subset LIM$

The reader is encouraged to consult Jantke and Beick (1981), Zeugmann (1983) and Fulk (1989) for further investigation concerning consistent and conform identification.

4. Inconsistency and Polynomial Time

The results presented in the previous section may lead to the impression that the inconsistency phenomenon may be far beyond any practical relevance, since it has been established in a setting where both, the halting and the consistency problem are undecidable in general. Therefore now we ask whether the inconsistency phenomenon does survive in more realistic settings, i.e., in settings where consistency is decidable. Moreover, in order to be consequently realistic we additionally restrict ourselves to deal exclusively with learning strategies that are polynomial time computable. Then, of course, the superiority of inconsistent strategies, if any, has to be established in terms of complexity theory. We present a learning problem in a realistic setting which is generally consistently solvable but which *cannot be solved consistently in polynomial time*, unless $\mathcal{P} \neq \mathcal{NP}$. The desired goal is achieved by elaborating an algorithm *inconsistently* solving the same learning problem in *polynomial time.* As far as we know this is the first learning problem for which the announced properties are rigorously proved. In our opinion, this result gives strong evidence of taking seriously inconsistent learning strategies into consideration.

The setting we want to deal with is the learnability of pattern languages introduced by Angluin (1980). Subsequently, Shinohara (1982) dealt with polynomial time learnability of subclasses of pattern languages. Nix (1983) outlined interesting applications of pattern inference algorithms. Recently, Kearns and Pitt (1989) as well as Ko, Marron and Tzeng (1990) studied intensively the learnabilty of pattern languages in the PAC–learning model.

So let us define what are a pattern and a pattern language. Let $\Sigma = \{a, b, ..\}$ be any non–empty finite alphabet containing at least two elements. Furthermore, let $X = \{x_i \mid i \in N\}$ be an infinite set of variables such that $\Sigma \cap X = \emptyset$. *Patterns* are non–empty strings from $\Sigma \cup X$, e.g., ab, ax_1ccc, $bx_1x_1cx_2x_2$ are patterns. $L(p)$, the language generated by pattern p is the set of strings which can be obtained by substituting non–null strings from Σ^* for the variables of the pattern p. Thus $aabbb$ is generable from pattern ax_1x_2b, while $aabba$ is not. Pat and PAT denote the set of all patterns and of all pattern languagues, respectively. In order to deal with the learnability of pattern languages we have to specify from what information the learning strategies should do their task. Following Gold (1967) we distinguish between learning from text and from informant. Formally, let $L \subseteq \Sigma^*$; we call a mapping $I : N \to \Sigma^* \times \{+, -\}$ *informant* of L iff

(1) For any $w \in \Sigma^*$, there are an $n \in N$ and $\lambda \in \{+, -\}$ such that $I(n) = (w, \lambda)$.

(2) For any $n \in N,\ w \in \Sigma^*,$ and $\lambda \in \{+, -\}$, if $I(n) = (w, \lambda)$ then $w \in L$ iff $\lambda = +$.

Let *Info(L)* denote the set of all informants of L. Furthermore, for $I \in \mathit{Info(L)}$ and $n \in N$ let $I^n = \mathit{cod}(I(0), ..., I(n))$, where *cod* denotes an effective and bijective mapping from the set of all finite sequences of elements from $\Sigma^* \times \{+, -\}$ onto N. Finally, we set $I_n = \{w \in \Sigma^* \mid \text{there are } i \leq n,\ \lambda \in \{+, -\} \text{s.t. } I(n) = (w, \lambda)\}$, and $I_n^+ = I_n \cap L,\ I_n^- = I_n \setminus I_n^+$.

Any mapping T from N onto L is called a *text* for L. By *Text(L)* we denote the set of all texts for L. The sets T_n, T_n^+ as well as T_n^- are defined analogously as above.

Intuitively, a text for L generates the language L– without any information concerning the complement of L, whereas an informant of L decides L by informing the strategy whether or not any word from Σ^* belongs to L. Note that we allow a text and an informant to be non–effective.

Definition 4. *PAT is called learnable in the limit from informant (abbr.* $PAT \in LIM - INF$*) iff there is an effective strategy S from N into Pat such that for any*

$L \in PAT$ and any $I \in$ Info(L),

(1) for any $n \in N$, $S(I^n)$ is defined, and

(2) there is a $p \in PAT$ such that $L(p) = L$ and for almost all $n \in N$, $S(I^n) = p$.

Definition 5. *PAT is called consistently learnable in the limit from informant (abbr. $PAT \in CONS - INF$) iff there is an effective strategy S from N into Pat such that*

(1) $PAT \in LIM - INF$ by S

(2) for any $L \in PAT$, $I \in$ Info(L) and $n \in N$, $I_n^+ \subseteq L(S(I^n))$ and $I_n^- \cap L(S(I^n)) = \emptyset$.

Note that a consistent learning strategy is required to correctly reflect both, the positive as well as the negative data it has already seen. Next we sharpen Definition 5 by additionally requiring polynomial time computability of S.

Definition 6. *PAT is called (consistently) learnable in the limit ¿from informant in polynomial time (abbr. $PAT \in$ Poly $- LIM - INF$ ($PAT \in$ Poly $- CONS - INF$)) iff there are a strategy S and a polynomial pol such that*

(1) $PAT \in LIM - INF$ ($PAT \in CONS - INF$) by S, and

(2) for any $L \in PAT$, $I \in$ Info(L) and any $n \in N$, time to compute $S(I^n) \leq pol(length(I^n))$.

Learning from text is analogously defined in replacing everywhere "informant" by "text". However, one point should be stated more precisely, i.e., consistent learning on text does only require consistency with the data contained in the text. In order to have an example illuminating the difference we could define a strategy that initially outputs x_1. Since $L(x_1)$ contains any string over Σ but the empty one, this hypothesis is consistent on text for any finite input. However, since the strategy has to converge, it cannot maintain this hypothesis ad infinitum. Finally, we use $LIM - TXT$, $CONS - TXT$ as well as $Poly - CONS - TXT$ to denote the corresponding learning types on text.

Now we can precisely state the result mentioned above.

Theorem 7.

(1) $PAT \in CONS - INF$

(2) If $\mathcal{P} \neq \mathcal{NP}$, then $PAT \notin$ Poly $- CONS - INF$.

(3) $PAT \in$ Poly $- LIM - INF$

Sketch of proof. Assertion (1) is proved in applying Gold's (1967) enumeration technique. Therefore, let $(p_i)_{i \in N}$ be any fixed effective enumeration of *Pat*. Let $L \in PAT$, let $I \in$ *Info(L)* be any informant, and let $n \in N$. Define $S(I^n)$ to be the first pattern p in the enumeration of *Pat* satisfying $I_n^+ \subseteq L(p)$ and $I_n^- \cap L(p) = \emptyset$. Since membership for pattern languages is uniformly decidable (cf. Angluin (1980)), S is computable. Due to the definition of S, consistency is obvious. Moreover, the strategy converges to the first pattern in the enumeration that generates the language L to be learnt. Note that S cannot be computed in polynomial time, unless $\mathcal{P} = \mathcal{NP}$, since membership for pattern languages is $\mathcal{NP}$–complete (cf. Angluin (1980)).

Next we have to show that there is no strategy at all consistently learning PAT that is computable in polynomial time, if $\mathcal{P} \neq \mathcal{NP}$. This part of the proof is done by showing the $\mathcal{NP}$–hardness of an appropriate problem defined below. For any information concerning reducibility as well as $\mathcal{NP}$–complete problems the reader is referred to Garey and Johnson (1979). First we define the following decision problem SEP. Let $W^+,\ W^- \subseteq \Sigma^*$. We say that $W^+,\ W^-$ are *separable* iff there is a pattern p such that $W^+ \subseteq L(p)$ and $W^- \cap L(p) = \emptyset$. SEP denotes the problem of *deciding* whether any $W^+,\ W^- \subseteq \Sigma^*$ are separable. Moreover, by $CSEP$ we denote the problem of *constructing* a separating pattern p for any given $W^+,\ W^-$ that are separable. The proof of assertion (2) is completed via the following lemmata.

Lemma A. (Ko, Marron, Tzeng, 1990)
$3 - SAT$ *is polynomial time reducible to* SEP.

Lemma B. $CSEP$ *is* $\mathcal{NP}$*–hard.*

Proof of Lemma B. Let $C3 - SAT$ denote the problem to construct a satisfying assignment to any satisfiable instance from $3 - SAT$.
Claim 1. $C3 - SAT \in \mathcal{P}$ implies $3 - SAT \in \mathcal{P}$

Assume there is an algorithm $\mathcal{A}$ having a running time that is bounded by some polynomial *pol* in the length of its input, and that, moreover, on input C returns a satisfying assignment of C, if C is satisfiable. Now let C be any instance of $3 - SAT$. Start $\mathcal{A}$ on input C. Since any polynomial is time constructable, we we may combine $\mathcal{A}$ with a clock, i.e., we can efficiently stop $\mathcal{A}$ on input C after at most $length(C)$ steps of computation. Then two cases are possible. Either $\mathcal{A}$ returns nothing. Consequently, C cannot be satisfiable. Otherwise $\mathcal{A}$ outputs an assignment *ass* within the given time bound. Then one can check in polynomial time whether or not *ass* indeed satisfies C. In case it does, we know that C is satisfiable. In case it does not C is again not satisfiable, since otherwise $\mathcal{A}$ fails.

Note that we *cannot* prove the $\mathcal{NP}$–hardness of $CSEP$ in the same manner as in showing claim 1, since membership for pattern languages is $\mathcal{NP}$–complete, i.e., one cannot check in polynomial time whether a pattern eventually returned on input (W^+, W^-) does indeed separate these sets. However, we overcome this difficulty in showing the following claim.
Claim 2. $CSEP \in \mathcal{P}$ implies $C3 - SAT \in \mathcal{P}$

In accordance with Lemma A let *red* be any polynomial time reduction of $3 - SAT$ to SEP. Suppose, there is an algorithm $\mathcal{B}$ solving $CSEP$ in polynomial time. Now let C be any satisfiable instance of $3 - SAT$. The wanted satisfying assignment may be computed as follows. First, compute $red(C) = (W^+, W^-)$. Since C is satisfiable, we get that (W^+, W^-) are separable. Next compute $p = \mathcal{B}(W^+, W^-)$. Finally, let *ass* be the assignment contructed to p in the proof of the "only–if" direction of Lemma A. Since *red* is computable in time bounded by a polynomial in the length of C, the length of

(W^+, W^-) is bounded by a polynomial in the length of C, too. Consequently, ass is polynomial time computable. Hence, $C3 - SAT \in \mathcal{P}$, if $CSEP \in \mathcal{P}$.

Finally, Claim 1 and 2 directly yield Lemma B.

The proof of assertion (2) is completed by showing the next claim.
Claim 3. $PAT \in Poly - CONS - INF$ implies $CSEP \in \mathcal{P}$.

Suppose $PAT \in Poly - CONS - INF$ by some strategy S. Let W^+, W^- be any two separable sets, and let p be any pattern separating them. Let $I \in Info(L(p))$ be an arbitrary informant such that, for some n, $I_n^+ = W^+$ and $I_n^- = W^-$. In accordance with the definition of separabilty, I obviously exists. Consequently, $S(I^n)$ has to be defined, and furthermore, $q = S(I^n)$ has to be a pattern separating W^+, W^-, too. Finally, S is polynomial time computable. Hence we get $CSEP \in \mathcal{P}$.

It remains to prove assertion (3). In Lange and Wiehagen (1991) $PAT \in Poly - LIM - TXT$ has been shown. The corresponding strategy witnessing $PAT \in Poly - LIM - TXT$ works by "overlooking" data, i.e., it ignores all but the actually shortest strings of the language to be learnt. It turns out that sufficiently many really shortest strings of a pattern language do suffice to learn it. From these remaining strings a hypothesis is generated in time that is even polynomial in the length of these strings. However, it is most of the time inconsistent, while being correct in the limit. Let S denote the strategy from Lange and Wiehagen (1991) proving $PAT \in Poly - LIM - TXT$. We define a strategy $\tilde{S}$ witnessing $PAT \in Poly - LIM - INF$ as follws: On any input I^n we set $\tilde{S}(I^n) = S(I_n^+)$. This proves the theorem.

Note that $\tilde{S}$ even works semi–consistently, since $I_n^- \cap L(\tilde{S}(I^n)) = \emptyset$ is valid for all $n \in N$. Moreover, $\tilde{S}$ works iteratively as S does.

q.e.d.

At this point some remarks are mandatory. It should be mentioned that any consistent strategy S, independently of how complex it is, may be trivially converted into an inconsistent one that works in quadratic time. This is done as follows. On input I^n, one simulates S on input I^1, I^2,...,I^n no more than n steps, and outputs $S(I^k)$, where k is the largest number $y \leq n$ for which $S(I^y)$ is computable within at most n steps.

However, it is obvious that this simulation technique does not yield any advantage. It does neither increase the efficiency of the learning algorithm, if one sums up all steps of computation until the learning task is successfully solved; nor does it enlarge the learning power. What we are looking for are "intelligent" inconsistent techniques. In our opinion, Lange and Wiehagen's (1991) refined strategy behaves thus by the following reasons. First, it avoids membership tests at all. Second, it computes its current hypothesis iteratively. Third, the test whether or not it should eventually change its mind is extremely simple and may be executed in linear time. Moreover, the algorithm yielding an eventually new hypothesis performs exclusively syntactical or formal manipulations over strings.

Finally, let $Poly - CONS - LEXINF$ ($Poly - CONS - LEXTXT$) be the learning type obtained from $Poly - CONS - INF$ ($Poly - CONS - TXT$) by restricting the information presentation from any informant $I \in$ *Info(L)* (any text $T \in$ *Text(L)*) to the lexicographically ordered one (cf. Definition 6). Furthermore, let S be a strategy such that $PAT \in LIM - INF$ ($LIM - TXT$) by S. Then, for any $L \in PAT$ and $D \in$ *Info(L)*

$\cup$ *Text(L)*, let

$$\textit{Conv}(S, D) = \text{the least number } m \text{ such that for all } n \geq m,\ S(D^n) = S(D^m)$$

denote the *stage of convergence* of S on D.

We have been surprised to obtain the following theorem. It actually states that the inconsistent learning strategy of Lange and Wiehagen (1991) may behave both, i.e., consistently and efficiently, if it receives the crucial information on the language to be learnt in an appropriate order.

Theorem 8

(1) There are a strategy $\tilde{S}$ and a polynomial pol such that

(i) $PAT \in Poly - CONS - LEXINF$ by $\tilde{S}$,

(ii) for any $p \in Pat$, there is an $I_p \in$ Info$(L(p))$ such that

- *$\tilde{S}$ works consistently on I_p,*
- *$\Sigma_{n=0}^{Conv(S,I_p)}$time to compute $\tilde{S}(I_p^n) \leq pol(length(p))$.*

(2) There are a strategy S and a polynomial pol such that

(i) $PAT \in Poly - CONS - LEXTXT$ by S,

(ii) for any $p \in Pat$, there is a $T_p \in$ Text$(L(p))$ such that

- *S works consistently on T_p,*
- *$\Sigma_{n=0}^{Conv(S,T_p)}$time to compute $S(T_p^n)) \leq pol(length(p))$.*

Sketch of proof. Assertion (1), part (i) is proved using the strategy $\tilde{S}$ from the proof of Theorem 7, assertion (3) above. Then part (i) follows by Lemma 2 of Lange and Wiehagen (1991). Part (ii) directly follows from Theorem 2, assertion (3) of Lange and Wiehagen (1991).

The first part of assertion (2) is an immediate consequence of the proof of Theorem 1 in Lange and Wiehagen (1991), while the second follows as above.

q.e.d.

We conjecture that Theorem 7 remains valid for $CONS - TXT$. Assertion (1) has been proved by Angluin (1980). As already mentioned above, the strategy from Lange and Wiehagen (1991) directly yields (3). Consequently, the only part not yet proved is (2). One reason why (2) seems to be easier provable for informant than for text is the following. A strategy working consistently on informant has to work "hard" at any step of the learning process, while a consistent strategy on text may output x_1 for a while.

5. Conclusions

We have investigated the problem of consistent versus inconsistent learning. In spite of the remarkable power of consistent learning it turns out that this power is not universal.

There are learning problems which can be exclusively solved by inconsistent strategies, i.e., by strategies that do temporarily incorrectly reflect the behavior of the unknown object on data for which the correct behavior of the object is already known at the current stage of the learning process. This phenomenon has been investigated in a "highly theoretical" setting, namely in inductive inference of recursive functions. In this setting the seemingly senseless work of inconsistent strategies could be completely explained by the undecidability of consistency.

However, it turned out that the inconsistency phenomenon is also valid in more realistic situations, namely in domains where consistency is always decidable and the learning strategies have to work in polynomial time. The reason is quite anologous to that in the setting of arbitrary recursive functions. Providing $\mathcal{P} \neq \mathcal{NP}$, the $\mathcal{NP}$-hardness of problems can prevent learning strategies to produce consistent hypotheses in polynomial time.

Moreover, we conjecture that an analogous effect can be shown for incremental learning of *finite* classes of *finite* objects such as Boolean functions.

In any case, we regard the results obtained as giving strong evidence to take fast *inconsistent* strategies seriously into account.

Finally, the presented results do suggest directions of further research such as

1. finding fast inconsistent learning techniques
2. deriving conditions yielding that a given learning problem has no fast consistent solution, but a fast inconsistent one.

Acknowledgement

Substantial part of this work has been performed while the first author was visiting TH Darmstadt. He is grateful indebted to Rüdiger Reischuk and Wolfgang Bibel for providing inspiring working conditions.

Furthermore, both authors would like to acknowledge fruitful discussions with Andreas Jakoby and Christian Schindelhauer.

6. 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. and C.H. Smith, (1983), Inductive Inference: Theory and Methods, Computing Surveys 15, 3, 237 - 269

[3] Angluin, D. and C.H. Smith, (1987), Formal Inductive Inference, In Encyclopedia of Artificial Intelligence, St.C. Shapiro (Ed.), Vol. 1, pp. 409 - 418, Wiley-Interscience Publication, New York

[4] Barzdin, Ya.M., (1974), Inductive Inference of Automata, Functions and Programs, Proc. Int. Congress of Math., Vancouver, pp. 455 - 460

[5] Blum, M., (1967), Machine Independent Theory of Complexity of Recursive Functions, Journal of the ACM 14, 322 -336

[6] Blum, L. and M. Blum, (1975), Toward a Mathematical Theory of Inductive Inference, Information and Control 28, 122 - 155

[7] Fulk, M.,(1988), Saving the Phenomena: Requirements that Inductive Inference Machines Not Contradict Known Data, Information and Computation 79, 193 - 209

[8] Garey, M.R. and D.S. Johnson, (1979), Computers and Intractability, A Guide to the Theory of $\mathcal{NP}$–completness, Freeman and Company, San Francisco

[9] Gold, M.E., (1965), Limiting Recursion, Journal of Symbolic Logic 30, 28 - 48

[10] Gold, M.E., (1967), Language Identification in the Limit, Information and Control 10, 447 - 474

[11] Jantke, K.P. and H.R. Beick, (1981), Combining Postulates of Naturalness in Inductive Inference, Journal of Information Processing and Cybernetics (EIK) 17, 465 - 484

[12] Kearns, M. and L. Pitt, (1989), A Polynomial–time Algorithm for Learning k–variable Pattern Languages From Examples. In Proc. 2nd Annual Workshop on Computational Learning Theory, R. Rivest, D. Haussler, and M.K. Warmuth (Eds.), pp. 57 - 70, Morgan Kaufmann Publishers Inc.

[13] Ko, Ker–I, Marron, A. and W.G. Tzeng, (1990), Learning String Patterns and Tree Patterns From Examples, Proc. 7th Conference on Machine Learning, pp. 384 - 391

[14] Lange, S. and R. Wiehagen, (1991), Polynomial–Time Inference of Arbitrary Pattern Languages, New Generation Computing 8, 361 - 370

[15] Lange, S. and T. Zeugmann, (1991), Monotonic versus Non-monotonic Language Learning, in Proc. 2nd International Workshop on Nonmonotonic and Inductive Logic, December 1991, Reinhardsbrunn, to appear in Lecture Notes in Artificial Intelligence

[16] Lange, S. and T. Zeugmann, (1992), Types of Monotonic Language Learning and Their Characterization, Proc. 5th Annual Workshop on Computational Learning Theory, Morgan Kaufmann Publishers Inc.

[17] Nix, R.P., (1983), Editing by Examples, Yale University, Dept. Computer Science, Technical Report 280

[18] Osherson, D., Stob, M. and S. Weinstein, (1986), Systems that Learn. An Introduction to Learning Theory for Cognitive and Computer Scientists, MIT-Press, Cambridge, Massachusetts

[19] Porat, S. and J.A. Feldman, (1988), Learning Automata from Ordered Examples, in Proc. First Workshop on Computational Learning Theory, D. Haussler and L. Pitt (Eds.), pp. 386 - 396, Morgan Kaufmann Publishers Inc.

[20] Rogers, H.Jr., (1967), Theory of Recursive Functions and Effective Computability, Mc Graw–Hill, New York

[21] Shinohara, T., (1982), Polynomial Time Inference of Extended Regular Pattern Languages, RIMS Symposia on Software Science and Engineering, Kyoto, Lecture Notes in Computer Science 147, pp. 115 - 127, Springer–Verlag

[22] Solomonoff, R., (1964), A Formal Theory of Inductive Inference, Information and Control 7, 1 - 22, 234 - 254

[23] Wiehagen, R., (1976), Limes–Erkennung rekursiver Funktionen durch spezielle Strategien, Journal of Information Processing and Cybernetics (EIK) 12, 93 - 99

[24] Wiehagen, R., (1978a), Zur Theorie der algorithmischen Erkennung, Dissertation B, Humboldt–Universität zu Berlin

[25] Wiehagen, R., (1978b), Characterization Problems in the Theory of Inductive Inference, Proc. 5th Colloquium on Automata, Languages and Programming, Udine, July 17 - 21, G. Ausiello and C. Böhm (Eds.), Lecture Notes in Computer Science 62, pp. 494 - 508, Springer-Verlag

[26] Zeugmann, T., (1983), A–posteriori Characterizations in Inductive Inference of Recursive Functions, Journal of Information Processing and Cybernetics (EIK) 19, 559 - 594