Classification of Predicates and Languages

Rolf Wiehagen

FB Informatik, Uni. Kaiserslautern, P.O. Box 3049, 67653 Kaiserslautern, wiehagen@informatik.uni-kl.de

Carl H. Smith

Department of Computer Science, University of Maryland, College Park,MD 20912, smith@cs.umd.edu

Thomas Zeugmann

Inst. f. Theoretische Informatik, TH Darmstadt, 64283 Darmstadt, zeugmann@iti.informatik.th-darmstadt.de

Abstract

We study the classification of recursive predicates and languages. In particular, we compare the classification of predicates and languages with the classification of arbitrary recursive functions and with learning. Moreover, we refine our investigations by introducing classification with a bounded number of mind changes and establish a new hierarchy. Furthermore, we introduce *multi–classification* and characterize it. Finally, we study the classification of families of languages that have attracted a lot of attention in learning theory.

1 Introduction

Learning and *classification* have attracted considerable attention by computer scientists, both in theory and practice. *Inductive inference* is an important aspect of learning that has been widely studied (cf. Angluin and Smith (1983, 1987)). The inductive inference problem is to take finite samples of some target concept and to *generalize* an algorithm that can produce all other samples of the same concept. Hence, inductive inference may be regarded as the most general framework to study the *generalization problem* (cf. Michalski et al. (1983)). The *classification problem* may be described as follows: Given a number of, usually finite, choices, one takes finite samples of a target concept and has to find out algorithmically to which of the possible choices the concept belongs to (cf. Duda and Hart (1973)).

Recently, the problem of classification has been compared with the inductive inference problem in a recursion theoretic setting (cf. Wiehagen

and Smith (1992)). In that paper a new formalization of classification was introduced. As it turned out, despite some obvious similarities, the two notions are distinct and warrant further study.

The aim of the present paper is fourfold. First, we apply the previously developed formalism to investigate the classification of $\{0,1\}$ valued recursive functions. These functions are often called *predicates* as they represent binary decisions on the input. By utilizing the isomorphism between strings of symbols and the natural numbers, the predicates over the natural numbers also represent *formal languages*. Hence, we simultaneously study the classification of predicates and languages. On the one hand, we are interested in learning the differences and similarities between the classification of predicates and arbitrary recursive functions. On the other hand, we enrich this study by considering the following two cases. In the first case we are satisfied if the classification algorithm produces any *one* of the possibly many correct answers. In the second case we require the classification algorithm to produce *all* of the correct classifications (cf. Definition 3). Second, we compare the power of classification algorithms that are allowed to produce only a single guess (finite classification, cf. Definition 2, the $c = 0$ case) with those that may change their mind a predetermined fixed number of times (cf. Definition 2, the $c \in \mathbb{N}$ case). Third, we introduce the notion of *consistent* classification (cf. Definition 4) and compare it with general classification. Finally, we study classification of families of languages that have received considerable attention in learning theory.

2 Technical Preliminaries

By $\mathbb{N} = \{0,1,2,...\}$ we denote the set of all natural numbers. Members of $\mathbb{N}$ will serve as names for programs. The function computed by program i will be denoted by φ_i. Most reasonable ways of assigning names to programs results in a list $\varphi_0, \varphi_1, \ldots$ called an *acceptable programming system* (cf. Machtey and Young (1978)). By $\mathcal{R}$ we denote the class of all recursive functions. The class of $\{0,1\}$ valued recursive functions, our model of both predicates and languages, is denoted by $\mathcal{R}_{0,1}$. A set is *recursively enumerable* (r.e.) iff it is the domain of some φ_i. Subset is denoted by $\subseteq$ and $\subset$ denotes *proper* subset. For a function $f \in \mathcal{R}$ and $n \in \mathbb{N}$, let $f^n = \mathrm{cod}(f(0), f(1), ..., f(n))$, where cod denotes a computable and bijective mapping from the set $\mathbb{N}^*$ of all finite sequences of natural numbers onto $\mathbb{N}$. Sometimes, for the sake of simplicity of notation, we identify $\alpha \in \mathbb{N}^*$ with $\mathrm{cod}(\alpha)$, for α a finite function.

Next, we formalize the modes of identification and classification mentioned in the introduction. As in Gold (1967) we define an *inductive inference machine* (abbr. IIM) to be an algorithmic device which works as follows: The IIM takes as its input larger and larger initial segments of the

graph of a function and it either requests the next function value, or it first outputs a hypothesis, i.e., a name of a program, and then it requests the next function value.

A classification machine (abbr. CM) takes as input the graph of a function (as IIMs do) and it either requests the next function value, or it first outputs an integer chosen ¿from a finite set, and then it requests the next function value.

Let M be an IIM or a CM. Furthermore, let i and j be two consecutive hypotheses produced by M. We say that M changes its mind, or synonymously, M performs a mind change, iff $i \neq j$. When dealing with mind changes, it is technically much more convenient to require the IIMs to behave as follows. Let f be a recursive function. If M on $f(0), ..., f(n)$ outputs its first guess, then it has to output a hypothesis at any subsequent step. It is easy to see that any IIM M may be straightforwardly converted into an IIM $\check{M}$ behaving as required such that both machines produce the same sequence of mind changes.

We start with the formalization of learning. The following definition is due to Gold (1967) and Blum and Blum (1975).

Definition 1. *Let $U \subseteq \mathcal{R}$ and let $c \in \mathbb{N} \cup \{*\}$. The class U is said to be learnable with at most c mind changes iff there is an IIM M such that for all $f \in U$*

(1) *there is a j such that $\varphi_j = f$ and $M(f^n) = j$ for almost all $n \in \mathbb{N}$,*

(2) *M, when successively fed with $f(0)$, $f(1)$, ... performs at most c ($c = *$ means at most finitely many) mind changes, i.e., $card(\{n \mid M(f^n) \neq M(f^{n+1})\}) \leq c$.*

If U can be learned by an IIM M with at most c mind changes, then we write $U \in EX_c(M)$. The class of all sets of recursive functions that are learnable with at most c mind changes is denoted by EX_c, an abbreviation for *explains* as a program for f can be regarded as an explanation of the set of examples constituting the graph of f (cf. Case and Smith (1983)). If U can be learned with 0 mind changes, then we also say that U is *finitely* learnable. Moreover, we set $FIN =_{df} EX_0$. If $c = *$, then we usually omit the lower index and simply say U can be learned.

Next, we formalize classification of finitely many sets.

Definition 2. *Let $S_0, ..., S_{k-1} \subseteq \mathcal{R}$ and let $S = S_0 \cup ... \cup S_{k-1}$. Furthermore, let $c \in \mathbb{N} \cup \{*\}$. Then $(S_0, ..., S_{k-1})$ is said to be classifiable with at most c mind changes iff there is a CM M such that for all $f \in S$*

(1) *for all $n \in \mathbb{N}$, whenever M, on input f^n, outputs a hypothesis j, then $j \in \{0, ..., k-1\}$,*

(2) *there is a j such that $f \in S_j$ and $M(f^n) = j$ for almost all $n \in \mathbb{N}$,*

(3) *M, when successively fed $f(0)$, $f(1)$, ... performs at most c ($c = *$ means at most finitely many) mind changes, i.e., $card(\{n \mid M(f^n) \neq M(f^{n+1})\}) \leq c$.*

If $(S_0, ..., S_{k-1})$ is classified by a CM with at most c mind changes, then we write $(S_0, ..., S_{k-1}) \in CL_k^c(M)$. By CL_k^c we denote the collection of all k–tuples of sets that are classifiable with at most c mind changes, i.e.,

$$CL_k^c = \{(S_0, ..., S_{k-1}) \mid \exists \text{CM } M\ [(S_0, ..., S_{k-1}) \in CL_k^c(M)]\}.$$

Moreover, we set $CL^c =_{df} \bigcup_{k \geq 2} CL_k^c$. If $c = *$, then we usually omit the upper index, and say simply that $(S_0, ..., S_{k-1})$ is classifiable. Furthermore, if $c = 0$, then we also say that $(S_0, ..., S_{k-1})$ is *finitely classifiable*, and set $FCL_k =_{df} CL_k^0$. Finally, we set $FCL =_{df} \bigcup_{k \geq 2} FCL_k$.

In the next definition we consider the situation that the sets $S_0, ..., S_{k-1}$ are not necessarily disjoint. Looking at potential applications, it might be highly desirable not to obtain only *one* index of a set the target function f belongs to, but the indices of *all* those sets that contain f. For example, consider the case of automated medical diagnosis. In this case, we would certainly desire to be aware of all the diseases that manifest the observed symptoms. In our formalism, each disease is set S_i and the functions to be classified map symptoms to *present* or *not present*.

Definition 3. *Let $S_0, ..., S_{k-1} \subseteq \mathcal{R}$, and let $S = S_0 \cup \ ... \ \cup S_{k-1}$. Furthermore, let $ALL = \{0, ..., k-1\}$. Then $(S_0, ..., S_{k-1})$ is said to be multi–classifiable iff there is a CM M such that for all $f \in S$*

(1) *for all $n \in \mathbb{N}$, whenever M, on input f^n, outputs a hypothesis HYP, then $HYP \subseteq ALL$,*

(2) *there is a non-empty set $SUB \subseteq ALL$ such that*

   (a) $M(f^n) = SUB$ for almost all n,

   (b) $f \in S_j$ for all $j \in SUB$,

   (c) $f \notin S_m$ for all $m \in ALL \backslash SUB$.

We write $(S_0, ..., S_{k-1}) \in \mathit{Multi\text{–}CL}_k(M)$ if $(S_0, ..., S_{k-1})$ is multi–classified by a CM M. By $\mathit{Multi\text{–}CL}_k$ we denote the collection of all k–tuples of sets that are multi–classifiable. We set $\mathit{Multi\text{–}CL} =_{df} \bigcup_{k \geq 2} \mathit{Multi\text{–}CL}_k$.

Finally, we introduce *consistent* classification. The main intention is as follows. Since potential users of a CM M never know whether M has successfully finished its classification task, they might want to be sure that the hypotheses they receive do correctly reflect the information the CM has been fed with.

Definition 4. *Let $S_0, ..., S_{k-1} \subseteq \mathcal{R}$ and let $S = S_0 \cup \ ... \ \cup S_{k-1}$. Then $(S_0, ..., S_{k-1})$ is said to be consistently classifiable iff there is a CM M such that*

(1) $(S_0, ..., S_{k-1}) \in CL_k(M)$,

(2) *for all* $f \in S$, *if* $M(f^n) = i$, *then there must be a function* $g \in S_i$ *such that* f *and* g *coincide up to* n.

We write $(S_0, ..., S_{k-1}) \in$ *Cons–*$CL_k(M)$ if $(S_0, ..., S_{k-1})$ is consistently classifiable by a CM M. By *Cons–*CL_k we denote the collection of all k–tuples of sets that are consistently classifiable. Finally, we set *Cons–*$CL =_{df} \bigcup_{k \geq 2}$ *Cons–*CL_k.

3 Classification of Predicates versus Classification of Arbitrary Functions

In this section we compare the classification of $\{0,1\}$ valued functions with the classification of functions in general. Looking at learning there are several results establishing major differences between the learnability of classes of predicates and the inferability of arbitrary classes of recursive functions (cf. Blum and Blum (1975), Zeugmann (1983, 1988), Osherson, Stob and Weinstein (1986)). Moreover, Freivalds, Kinber and Wiehagen (1992) discovered that consistent learning of predicates considerably differs from consistent identification of arbitrary recursive functions. Hence, it is only natural to ask whether there are differences between the classification of predicates and arbitrary recursive functions.

Clearly, if a collection of sets of $\{0,1\}$ valued functions is classifiable, then it is classifiable as a collection of sets of arbitrary recursive functions. To consider the converse direction, we need the following notation. Let $S \subseteq \mathcal{R}$; then we use $\rho(S)$ to denote the restriction of S to predicates, i.e., $\rho(S) = S \cap \mathcal{R}_{0,1}$.

Theorem 5. *There is a collection of pairwise disjoint sets* S_0, S_1, S_2 *such that*

(1) $\mathcal{R} = S_0 \cup S_1 \cup S_2$,

(2) $(S_0, S_1, S_2) \notin CL$,

(3) $(\rho(S_0), \rho(S_1), \rho(S_2)) \in CL_3^2$.

The theorem has the following corollary.

Corollary 6. *For any* $n \geq 3$ *there is a collection of pairwise disjoint sets exhausting* $\mathcal{R}$ *that is not in* CL_n, *but the collection of their restrictions to predicates is in* CL_n.

The latter results show that there might be interesting differences between the classification of predicates and arbitrary functions. Later on, we shall point out some more differences. In the next section we study the power of classification algorithms with respect to the allowed number of mind changes.

4 Finite Classification versus Classification and Learnability

Our next theorem in particular states that even the classification of two sets might be too complex to be done by a finitely working CM.

Theorem 7. *For any $k > 1$ there are pairwise disjoint sets $S_0, \cdots, S_{k-1}$ such that*

(1) $S_0 \cup S_1 \cup \cdots \cup S_{k-1} = \mathcal{R}_{0,1}$,

(2) $(S_0, S_1, \cdots, S_{k-1}) \in CL \setminus FCL$.

In contrast to Theorem 7, it is possible to split $\mathcal{R}_{0,1}$ into disjoint, finitely classifiable sets.

Theorem 8. *For any $k > 1$ there are pairwise disjoint sets $S_0, \cdots, S_{k-1}$ such that*

(1) $S_0 \cup S_1 \cup \cdots \cup S_{k-1} = \mathcal{R}_{0,1}$,

(2) $(S_0, S_1, \cdots, S_{k-1}) \in FCL$.

Next we compare finite classification and learnability. If a classification machine outputs a hypothesis i, then we know almost *all* about the corresponding set S_i, since there are only finitely many sets $S_0, ..., S_{k-1}$. On the other hand, if an IIM outputs a hypothesis i, then we know almost *nothing* about the corresponding function φ_i. Hence, at first glance it seems much easier to disprove a CM than to fool an IIM. However, the situation is much more subtle, as the following theorems show.

Theorem 9. *For any $S_0 \subset \mathcal{R}_{0,1}$ with $S_0 \in FIN$, there is a S_1 such that*

(1) $S_1 \subset \mathcal{R}_{0,1}$,

(2) $S_1 \in FIN$,

(3) $S_0 \cap S_1 = \emptyset$,

(4) $(S_0, S_1) \in FCL$.

The latter theorem allows the following interpretation. Any "easily" learnable class can be completed to an "easily" classifiable pair. Thus, sometimes learning and classification are *not* very distinct. On the other hand, we have the following:

Theorem 10. *For any $S_0 \subset \mathcal{R}_{0,1}$ with $S_0 \in FIN$, there is a S_1 such that*

(1) $S_1 \subset \mathcal{R}_{0,1}$,

(2) $S_1 \notin EX$,

(3) $S_0 \cap S_1 = \emptyset$,

(4) $(S_0, S_1) \in FCL$.

5 Bounding the Number of Mind Changes

For finite classification (FCL) a CM is not allowed to change its conjecture at all. For standard classification (CL) a CM is allowed to change its conjecture an arbitrary finite number of times. The precise number of mind changes has not to be determined in advance. In the study of inductive inference, a mind change hierarchy was discovered by fixing in advance a particular number of mind changes that an IIM was allowed to make (cf. Case and Smith (1983)). In the sequel we present a hierarchy for classification based on a fixed number of allowed mind changes.

Theorem 11. *For any k there are sets S_0, S_1, $\cdots$, S_{k+1} such that*

(1) $S_0 \cup S_1 \cup \cdots \cup S_{k+1} = \mathcal{R}_{0,1}$,

(2) $(S_0, \cdots, S_{k+1}) \in CL^{k+1} \setminus CL^k$.

Moreover, it is also possible to prove a hierarchy in the number of mind changes that is not related to the number of sets to be classified.

Theorem 12. *For any $c \in \mathbb{N}$ there exist pairwise disjoint sets $S_0, S_1 \subseteq \mathcal{R}_{0,1}$ such that $(S_0, S_1) \in CL_2^{c+1} \setminus CL_2^c$.*

6 Classification versus Multi–Classification and Consistent Classification

In this section we compare the power of classification, multi–classification and consistent classification. In particular, we are interested in learning whether or not multi–classification or consistent classification is harder to achieve than ordinary one. Since neither multi–classification nor consistent classification has been studied in the framework present here, our goal is twofold. First we are interested in general results concerning the classification power of these classification modes. Second, we ask whether or not the results obtained do extend to the classification of predicates.

Our next theorem compares multi–classification and ordinary classification in the finite case.

Theorem 13. *Multi–$FCL_2 \subset FCL_2$*

Next we aim to characterize multi–classification in terms of standard classification. For that purpose, we introduce the notion of a *mosaic*.

Let $S_0, ..., S_{k-1} \subseteq \mathcal{R}$ and $S = S_0 \cup \ ... \ \cup S_{k-1}$. Let $ALL = \{0, ..., k-1\}$ and $SUB \subseteq ALL$. Then let $S_{SUB} = \bigcap_{j \in SUB} S_j \setminus \bigcup_{i \in \ ALL \setminus SUB} S_i$. Clearly, S_{SUB} is the set of all $f \in S$ that belong to any of the classes S_j, $j \in SUB$,

and to none of the classes S_i, $i \in ALL \backslash SUB$. Let $mosaic(S_0, ..., S_{k-1})$ be the tuple of all sets S_{SUB}, where $SUB \subseteq ALL$.

Obviously, $mosaic(S_0, ..., S_{k-1})$ is a 2^k–tuple of pairwise disjoint sets the union of which is S. Note that some components of $mosaic(S_0, ...S_{k-1})$ may be empty.

Finally, let $mosaic(S_0, ..., S_{k-1}) = (T_0, ..., T_{2^k-1})$. Moreover, without loss of generality we assume that the T_j's, $0 \leq j \leq 2^k - 1$, are ordered in such a way that from the index j the corresponding set SUB can be computed such that $S_{SUB} = T_j$ and vice versa. Formally, this can be realized by a one-to-one mapping m from the set of all subsets SUB of ALL onto the set $\{0, ...2^k - 1\}$ such that for any $SUB \subseteq ALL$, $S_{SUB} = T_{m(SUB)}$.

We have obtained the following characterization.

Theorem 14. *For all $k \geq 2$ and all sets $S_0, ..., S_{k-1} \subseteq \mathcal{R}$ we have: $(S_0, ..., S_{k-1}) \in$ Multi–CL if and only if $mosaic(S_0, ..., S_{k-1}) \in CL$.*

Finally in this section we relate consistent classification to ordinary one.

Theorem 15. $Cons\text{–}CL \subset CL$

Note that all theorems in this section remain valid if one considers exclusively the classification of predicates.

7 Classification of Languages

In this section first we examine the classification of the regular languages (cf. Lewis and Papadimitriou (1981)). The regular languages can be modeled as predicates as well. This is done by fixing an isomorphism between strings over the alphabet of the regular language and the natural numbers. Then a (regular) language represented by a $\{0, 1\}$ valued function f is the set of strings that correspond to the natural numbers in the set $\{x \,|\, f(x) = 1\}$. The details of this encoding will be suppressed as much as possible without sacrificing clarity. By a *positive example* we mean a string that corresponds to a value x such that $f(x) = 1$. Similarly, by a *negative example* we mean a string that corresponds to a value x such that $f(x) = 0$.

Theorem 16. *It is impossible to classify an arbitrary language as being either regular or not regular.*

By way of contrast with Theorem 16, it is possible to separate arbitrarily large subsets of the regular languages ¿from the rest.

Theorem 17. *Let n be an arbitrary natural number. Let S_0 be the set of all regular languages that are recognized by some n-state finite automaton. Let S_1 be all the languages not in S_0. Then $(S_0, S_1) \in CL$.*

The last theorem relates the learnability of indexed families of languages to their classifiability. In particular, the next theorem establishes a condition under what circumstances a classification algorithm can be derived from a given learning algorithm. In general, it is impossible to convert a learning algorithm into a classification machine, as previously obtained results show (cf. Wiehagen and Smith (1992)). Due to lack of space, we can only sketch the next result. $\mathcal{L} = (L_j)_{j \in \mathbb{N}}$ is said to be an indexed family, if all languages L_j are non–empty, and there is an algorithm that uniformly decides membership in L_j for all strings over the underlying alphabet and all $j \in \mathbb{N}$. The inferability of indexed families has attracted a lot of attention in learning theory (cf. Lange and Zeugmann (1993) and the references therein). Of particular interest is the learnability of indexed families from positive data. We have obtained the following theorem.

Theorem 18. *Let $m \geq 1$ and let $\mathcal{L}$ be any indexed family over some fixed alphabet that can be learned from positive data with at most m mind changes. Then there exists a partition of $\mathcal{L}$ into $m+1$ pairwise disjoint classes $\mathcal{L}_0, ..., \mathcal{L}_m$ such that*

(1) $\mathcal{L}_0 \cup ... \cup \mathcal{L}_m = \mathcal{L}$,

(2) $(\mathcal{L}_0, ..., \mathcal{L}_m) \in CL_{m+1}$.

Finally, all proofs of the theorems stated above can be found in Wiehagen, Smith and Zeugmann (1993).

Acknowledgment

The first author was supported by the German Ministry for Research and Technology (BMFT) under grant no. 01 IW 101. The second author was supported in part by NSF Grant 9020079.

8 References

1. Angluin, D., and Smith, C.H. (1983). Inductive inference: theory and methods, *Computing Surveys* **15**, 237 - 269.
2. Angluin, D., and Smith, C.H. (1987). Formal inductive inference, *in* "Encyclopedia of Artificial Intelligence" (St.C. Shapiro, Ed.), Vol. 1, pp. 409 - 418, Wiley-Interscience Publication, New York.
3. Blum, L., and Blum, M. (1975). Toward a mathematical theory of inductive inference, *Information and Control* **28**, 122 - 155.

4. Case, J., and Smith, C.H. (1983). Comparison of identification criteria for machine inductive inference, *Theoretical Computer Science* **25**, 193 - 220.
5. Duda, R., and Hart, P. (1973). *Pattern Classification and Scene Analysis*, Wiley Interscience.
6. Freivalds, R., Kinber, E.B., and Wiehagen, R. (1992). Convergently versus divergently incorrect hypotheses in inductive inference, GOSLER Report 05/92, January 1992, Fachbereich Mathematik und Informatik, TH Leipzig.
7. Freivalds, R.V., and Wiehagen, R. (1979). Inductive inference with additional information, *Journal of Information Processing and Cybernetics (EIK)* **15**, 179 - 184.
8. Gold, M.E. (1967). Language identification in the limit, *Information and Control* **10**, 447 - 474.
9. Jantke, K.P., and Beick, H.R. (1981). Combining postulates of naturalness in inductive inference, *Journal of Information Processing and Cybernetics (EIK)* **17**, 465 - 484.
10. Lange, S., and Zeugmann, T. (1993). Learning recursive languages with bounded mind changes, *International Journal of Foundations of Computer Science*, **4**, Vol. 2, 157 - 178.
11. Lewis, H., and Papadimitriou, C. (1981). *Elements of the Theory of Computation*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.
12. Machtey, M., und Young, P. (1978). *An Introduction to the General Theory of Algorithms*, North-Holland, New York.
13. Michalski, R.S., Carbonell, J.G., and Mitchell, T.M. (1983). *Machine Learning*, Tioga Publishing Co., Palo Alto, CA.
14. Osherson, D., Stob, M., and Weinstein, S. (1986). *Systems that Learn, An Introduction to Learning Theory for Cognitive and Computer Scientists*, MIT-Press, Cambridge, Massachusetts.
15. Wiehagen, R., and Smith, C. (1992). Classification versus generalization, *in* "Proceedings 5th Annual ACM Workshop on Computational Learning Theory," Pittsburgh, July 1992, pp. 224 - 230, ACM Press, New York.
16. Wiehagen, R., Smith, C, and Zeugmann, T. (1993). Classifying recursive predicates and languages, GOSLER Report 21/93, Dezember 1993, Fachbereich Mathematik und Informatik, TH Leipzig.
17. Zeugmann, T. (1983). A-posteriori characterizations in inductive inference of recursive functions, *Journal of Information Processing and Cybernetics (EIK)* **19**, 559 - 594.
18. Zeugmann, T. (1988). On the power of recursive optimizers, *Theoretical Computer Science* **62**, 289 - 310.