A Survey of Robust Learning

Sanjay Jain[1]

School of Computing
National University of Singapore
Singapore 119260
sanjay@comp.nus.edu.sg

Abstract. A class of objects is said to be robustly learnable if not only this class itself is learnable but all of its computable "transformations" are also learnable. We study robust learning within the framework of inductive inference. A class of recursive functions is said to be robustly learnable under a learning criterion **I** iff all of its images under operators of a suitable kind are learnable under the criterion **I**. In this paper we will do a survey of some of the results in the area.

1 Introduction

Consider the following basic model for learning functions. A learner is fed the graph of a function f, one element at a time. The learner, as it is receiving the graph of the function, outputs a sequence of programs as its hypotheses (the program output at any particular time may be thought of as machine's conjecture about how f may be computed). The learner is said to identify (learn) f, just in case the sequence of programs converges to a program for f. The learner identifies a class $\mathcal{C}$ of functions iff it identifies every function f from the class $\mathcal{C}$. A class $\mathcal{C}$ is identifiable (learnable), if some learner identifies the class.

The above is essentially the paradigm of identification in the limit (called **Ex**-identification) introduced by Gold [Gol67] (see the formal definitions in Section 2). In this paper we will be concerned only about learners which are computable (and called learning or inductive inference machines) and will mostly focus on learnability of total functions (though we will briefly consider learnability of partial functions).

The theory of learning in the limit has been a focus of study by several researchers over the last four decades. Researchers have studied modifications of the model discussed above by relaxing and constraining it in various ways. We direct the reader to [JORS99] for an introduction to the area.

Let us consider an example. The class of polynomials can be **Ex**-identified as follows. The learner, on data $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, outputs the smallest degree polynomial f, such that $f(x_i) = y_i$, for $1 \leq i \leq n$. It can be easily verified that such a learner **Ex**-identifies all polynomials. Gold [Gol67] also showed that every recursively enumerable class of recursive functions can be **Ex**-identified. This can be done as follows. Suppose $\mathcal{C}$ is a recursively enumerable class of recursive functions. Let $p_0, p_1, \ldots$ be an effective sequence of programs which compute exactly the functions in $\mathcal{C}$. Now consider the following machine **M**. On

any finite input data, **M** outputs p_i, where i is the least number such that the function computed by program p_i is consistent with the input data. It is easy to verify that **M** acting as above will **Ex**-identify each function in $\mathcal{C}$. The above technique is called *identification by enumeration*.

A question one might ask about learnable classes is: how stable is the property that a class is learnable? That is, whether the class is still learnable if one makes small and not so small "transformations" of the class. This ofcourse depends on the class being learned, and the transformations considered. One may call a class robustly learnable if all of its transformations (of certain type) are learnable.

In this paper we will do a survey of robust learning within the paradigm of inductive inference. As this is a survey paper, the results presented here are from various papers listed in the references. Most of the motivation below is from the paper [JSW98].

Let us now consider some motivations for studying robust learning. Recall the identification by enumeration strategy we considered earlier for learning any recursively enumerable class of functions. The naturalness of this strategy led Gold to conjecture that *any* class of functions which can be **Ex**-identified, can also be **Ex**-identified using identification by enumeration — that is, every **Ex**-identifiable class is contained in a recursively enumerable class of functions.

The above conjecture of Gold was shown to be false by Bārzdiņš [Bār71a]. This was done using the following class of functions: $\mathbf{SD} = \{f \mid f(0) \text{ is a program for } f\}$. It is easy to construct a machine that **Ex**-identifies **SD**. On the other hand, it can be shown that **SD** is not contained in any recursively enumerable class of recursive functions.

In Gold's defense, **SD** and other self-referential classes used to refute his conjecture seem artificial. In these classes, programs for the input function are directly "coded" in the graph of the function (for example as $f(0)$ in case of **SD** above). To circumvent such artificial self-referential counter-examples, Bārzdiņš made a more sophisticated version of Gold's conjecture. He argued that if a class of functions can be identified only because of the presence of self-referential property, then there would be an effective transformation that can transform the class into an unidentifiable one. To see this, in the context of the class **SD** above, consider the operator Θ defined as follows: $\Theta(f) = g$, where $g(x) = f(x+1)$. It can be shown that $\Theta(\mathbf{SD}) = \{\Theta(f) \mid f \in \mathbf{SD}\} = \mathcal{R}$, the class of all the recursive functions. Since $\mathcal{R}$ is not **Ex**-identifiable [Gol67], $\Theta(\mathbf{SD})$ is not **Ex**-identifiable. Essentially, the idea is that if a learning machine can find the self-referential information, so can an effective transformation, which can then remove such information from the graph. The notion of an effective transformation can be made precise in several ways. In the present paper we will mostly be using general recursive operators, i.e. effective and total mappings from total functions to total functions (see Definition 6).

Informally, Bārzdiņš' conjecture can be stated as follows. If all the projections of a class of functions under all general recursive operators are identifiable (or, in other words, if the class is identifiable *robustly*), then the class is contained in a recursively enumerable class of recursive functions (and thus, identifiable

by enumeration). [Zeu86] first considered Bārzdiņš' conjecture. In this paper, total effective operators (see [Rog67]) rather than general recursive operators were used. Bārzdiņš' conjecture was then verified for several learning criteria, namely $\mathbf{Ex}_0$-identification (in which learner is allowed to make only one conjecture (which must be correct), see Definition 1), and **Reliable**-identification (here the learning machine is not allowed to converge on a total function, which cannot be learned by the machine, see Definition 15). Fulk [Ful90] showed that Bārzdiņš' conjecture as stated above (for general recursive operators) is false by exhibiting a class of functions which is robustly **Ex**-identifiable, but not contained in any recursively enumerable class of recursive functions.

Since Gold [Gol67] many criteria of inference have been proposed by researchers all over the world, see for example [AS83, BB75, CS83, Fre91, KW80, JORS99]. These criteria have usually been accompanied by proofs showing the differences between the new and old criteria of inference. The proof techniques used to show separations between the criteria often involve classes with self-referential properties. The same criticism of the class **SD** above applies to such separations. Thus, it is interesting to study whether these separations hold robustly.

Herman Weyl [Wey52] described the famous Erlangen program on founding geometry algebraically due to Felix Klein as follows: "If you are to find deep properties of some object, consider all the natural transformations that preserve your object (i.e. under which the object remains invariant)." Since general recursive operators (or other types of operators) can be looked upon as natural transformations, it is interesting to consider robust identification from a purely philosophical point of view. [AF96] consider a problem in a sense dual to ours. They search for general recursive operators which map all learnable classes to learnable classes.

The paper is organized as follows. In section 2 we consider notations and basic definitions from learning theory. In section 3 we consider results directly related to Bārzdiņš' Conjecture. Section 4 deals with results about robust separations. In section 5 we study uniform robust learning. Section 6 considers a generalization of robust learning known as hyper robust learning. Section 7 considers other possible operators, besides the general recursive operators, which may be used for transformations.

2 Notation and Preliminaries

Recursion-theoretic concepts not explained below are treated in [Rog67]. N denotes the set of natural numbers. $*$ denotes a non-member of N and is assumed to satisfy $(\forall n)[n < * < \infty]$. Let $\in, \subseteq, \subset, \supseteq, \supset$, respectively denote the membership, subset, proper subset, superset and proper superset relations for sets. The empty set is denoted by $\emptyset$. We let $\mathrm{card}(S)$ denote the cardinality of the set S. So "$\mathrm{card}(S) \leq *$" means that $\mathrm{card}(S)$ is finite. The minimum and maximum of a set S are denoted by $\min(S)$ and $\max(S)$, respectively. We take $\max(\emptyset)$ to be 0 and $\min(\emptyset)$ to be ∞.

$\langle \cdot, \cdot \rangle$ denotes a 1-1 computable mapping from pairs of natural numbers onto natural numbers. π_1, π_2 are the corresponding projection functions. $\langle \cdot, \cdot \rangle$ is extended to n-tuples of natural numbers in a natural way. Λ denotes the empty function. η, with or without decorations[1], ranges over partial functions. If η_1 and η_2 are both undefined on input x, then, we take $\eta_1(x) = \eta_2(x)$. We say that $\eta_1 \subseteq \eta_2$ iff for all x in domain of η_1, $\eta_1(x) = \eta_2(x)$. For $a \in N \cup \{*\}$, $\eta_1 =^a \eta_2$ means that $\text{card}(\{x \mid \eta_1(x) \neq \eta_2(x)\}) \leq a$. $\eta_1 \neq^a \eta_2$ means that $\neg[\eta_1 =^a \eta_2]$. If $\eta =^a f$, then we often call a program for η as an a-error program for f. We let $\text{domain}(\eta)$ and $\text{range}(\eta)$ respectively denote the domain and range of the partial function η. $\eta(x)\downarrow$ denotes that $\eta(x)$ is defined. $\eta(x)\uparrow$ denotes that $\eta(x)$ is undefined. We say that a partial function η is *consistent* with η' iff for all $x \in \text{domain}(\eta) \cap \text{domain}(\eta')$, $\eta(x) = \eta(x')$. η is *inconsistent* with η' iff there exists an x such that $\eta(x)\downarrow \neq \eta'(x)\downarrow$.

f, g, h, F and G, with or without decorations, range over total functions. $\mathcal{R}$ denotes the class of all *recursive* functions, i.e., total computable functions with arguments and values from N. $\mathcal{C}$ and $\mathcal{S}$, with or without decorations, range over subsets of $\mathcal{R}$. $\mathcal{P}$ denotes the class of all *partial recursive* functions over N. φ denotes a *fixed* acceptable programming system [Rog67]. φ_i denotes the partial recursive function computed by program i in the φ-system. Note that in this paper all programs are interpreted with respect to the φ-system. We let Φ be an arbitrary Blum complexity measure [Blu67] associated with the acceptable programming system φ; many such measures exist for any acceptable programming system [Blu67].

A class $\mathcal{C} \subseteq \mathcal{R}$ is said to be recursively enumerable (r.e.) iff there exists an r.e. set X such that $\mathcal{C} = \{\varphi_i \mid i \in X\}$. For any non-empty recursively enumerable class $\mathcal{C}$, there exists a recursive function f such that $\mathcal{C} = \{\varphi_{f(i)} \mid i \in N\}$. Zero is the everywhere 0 function, i.e., $\text{Zero}(x) = 0$, for all $x \in N$.

2.1 Function Identification

We first describe inductive inference machines. We assume, without loss of generality, that the graph of a function is fed to a machine in canonical order. For $f \in \mathcal{R}$ and $n \in N$, we let $f[n]$ denote the finite initial segment $\{(x, f(x)) \mid x < n\}$. Clearly, $f[0]$ denotes the empty segment. SEG denotes the set of all finite initial segments, $\{f[n] \mid f \in \mathcal{R} \wedge n \in N\}$. We let σ and τ, with or without decorations, range over SEG. Let $|\sigma|$ denote the length of σ. We often identify (partial) functions with their graphs. Thus for example, for $\sigma = f[n]$ and for $x < n$, $\sigma(x)$ denotes $f(x)$. An *inductive inference machine* (IIM) [Gol67] is an algorithmic device that computes a mapping from SEG into $N \cup \{?\}$. Intuitively, "?" above denotes the case when the machine may not wish to make a conjecture. Although it is not necessary to consider learners that issue "?" for identification in the limit, it becomes useful when the number of mind changes a learner can make is bounded (see Definition 1 below). In this paper, we assume, without loss of generality, that once an IIM has issued a conjecture on some initial segment

[1] Decorations are subscripts, superscripts, primes, and the like.

of a function, it outputs a conjecture on all extensions of that initial segment. This is without loss of generality because a machine wishing to emit "?" after making a conjecture can instead be thought of as repeating its previous conjecture. We let $\mathbf{M}$, with or without decorations, range over learning machines. Since the set of all finite initial segments, SEG, can be coded onto N, we can view these machines as taking natural numbers as input and emitting natural numbers or ?'s as output. We say that $\mathbf{M}(f)$ converges to i (written: $\mathbf{M}(f)\downarrow = i$) iff $(\forall^{\infty} n)[\mathbf{M}(f[n]) = i]$; $\mathbf{M}(f)$ is undefined if no such i exists.

Definition 1. [Gol67, BB75, CS83] Let $a, b \in N \cup \{*\}$. Let $f \in \mathcal{R}$, and $\mathcal{S} \subseteq \mathcal{R}$.

(a) $\mathbf{M}$ $\mathbf{Ex}_b^a$*-identifies* f (written: $f \in \mathbf{Ex}_b^a(\mathbf{M})$) just in case there exists an a-error program i for f such that $\mathbf{M}(f)\downarrow = i$ and $\mathrm{card}(\{n \mid ? \neq \mathbf{M}(f[n]) \neq \mathbf{M}(f[n+1])\}) \leq b$ (i.e., $\mathbf{M}$ makes no more than b mind changes on f).
(b) $\mathbf{M}$ $\mathbf{Ex}_b^a$*-identifies* $\mathcal{S}$ iff $\mathbf{M}$ $\mathbf{Ex}_b^a$-identifies each $f \in \mathcal{S}$.
(c) $\mathbf{Ex}_b^a = \{\mathcal{S} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathcal{S} \subseteq \mathbf{Ex}_b^a(\mathbf{M})]\}$.

Note that in part (a) above, change of conjecture from ? to some $i \in N$ is not considered a mind change.

We often write $\mathbf{Ex}_b$ for $\mathbf{Ex}_b^0$, $\mathbf{Ex}^a$ for $\mathbf{Ex}_*^a$, and $\mathbf{Ex}$ for $\mathbf{Ex}_*^0$. $\mathbf{Ex}_0$ is also referred to as *finite* identification. By the definition of convergence, only finitely many data points from a function f have been observed by an IIM $\mathbf{M}$ at the (unknown) point of convergence. Hence, some form of learning must take place in order for $\mathbf{M}$ to learn f. For this reason, hereafter the terms *identify*, *learn* and *infer* are used interchangeably.

Definition 2. [Bār74, CS83] Let $a \in N \cup \{*\}$. Let $f \in \mathcal{R}$.

(a) $\mathbf{M}$ $\mathbf{Bc}^a$*-identifies* f (written: $f \in \mathbf{Bc}^a(\mathbf{M})$) iff, for all but finitely many $n \in N$, $\mathbf{M}(f[n])$ is an a-error program for f.
(b) $\mathbf{M}$ $\mathbf{Bc}^a$*-identifies* $\mathcal{S}$ iff $\mathbf{M}$ $\mathbf{Bc}^a$-identifies each $f \in \mathcal{S}$.
(c) $\mathbf{Bc}^a = \{\mathcal{S} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathcal{S} \subseteq \mathbf{Bc}^a(\mathbf{M})]\}$.

We often write $\mathbf{Bc}$ for $\mathbf{Bc}^0$.

Definition 3. $\mathbf{NUM} = \{\mathcal{C} \mid (\exists \mathcal{C}' \mid \mathcal{C} \subseteq \mathcal{C}' \subseteq \mathcal{R})[\mathcal{C}' \text{ is recursively enumerable}]\}$.

Some relationships between the above criteria are summarized in the following theorem.

Theorem 4. [CS83, BB75, Bār71a, Gol67]
(a) Let $b \in N \cup \{*\}$. Then, $\mathbf{Ex}_b = \mathbf{Ex}_b^0 \subset \mathbf{Ex}_b^1 \subset \mathbf{Ex}_b^2 \subset \cdots \subset \mathbf{Ex}_b^*$.
(b) Let $a \in N \cup \{*\}$. Then, $\mathbf{Ex}_0^a \subset \mathbf{Ex}_1^a \subset \mathbf{Ex}_2^a \subset \cdots \subset \mathbf{Ex}_*^a$.
(c) Let $a, b, c, d \in N \cup \{*\}$. Then, $\mathbf{Ex}_b^a \subseteq \mathbf{Ex}_d^c$ iff $a \leq c$ and $b \leq d$.
(d) $\mathbf{NUM} \subseteq \mathbf{Ex}$.
(e) $\mathbf{Ex}_0 - \mathbf{NUM} \neq \emptyset$.
(f) $\mathbf{NUM} - \bigcup_{m \in N} \mathbf{Ex}_m \neq \emptyset$.
(g) $\mathbf{Ex}_*^* \subset \mathbf{Bc} = \mathbf{Bc}^0 \subset \mathbf{Bc}^1 \subset \mathbf{Bc}^2 \subset \cdots \subset \mathbf{Bc}^*$.
(h) $\mathcal{R} \in \mathbf{Bc}^*$.

We let $\mathbf{I}$ and $\mathbf{J}$ range over identification criteria defined above.

2.2 Operators

Definition 5. [Rog67] A *recursive operator* is a computable total mapping, Θ, from (possibly partial) functions to (possibly partial) functions, which satisfies the following properties:

(a) Monotonicity: For all functions η, η', if $\eta \subseteq \eta'$ then $\Theta(\eta) \subseteq \Theta(\eta')$.
(b) Compactness: For all η, if $(x, y) \in \Theta(\eta)$, then there exists a finite function $\alpha \subseteq \eta$ such that $(x, y) \in \Theta(\alpha)$.
(c) Recursiveness: For all finite functions α, one can effectively enumerate (in α) all $(x, y) \in \Theta(\alpha)$.

Definition 6. [Rog67] A recursive operator Θ is called *general recursive* iff Θ maps all total functions to total functions.

For each recursive operator Θ, we can effectively (from Θ) find a recursive operator Θ' such that,

(d) for each finite function α, $\Theta'(\alpha)$ is finite, and its canonical index can be effectively determined from α, and
(e) for all total functions f, $\Theta'(f) = \Theta(f)$.

This allows us to get a nice effective sequence of recursive operators.

Proposition 7. *[JSW98] There exists an effective enumeration, $\Theta_0, \Theta_1, \cdots$ of recursive operators satisfying condition (d) above such that, for all recursive operators Θ, there exists an $i \in N$ satisfying:*

$$\textit{for all total functions } f,\ \Theta(f) = \Theta_i(f).$$

Since we will be mainly concerned with the properties of operators on total functions, for diagonalization purposes, one can restrict attention to operators in the above enumeration $\Theta_0, \Theta_1, \ldots$.

Definition 8. [Ful90, JSW98] Let $\mathbf{I}$, be an identification criterion.
$\mathbf{RobustI} = \{\mathcal{C} \mid (\forall \text{ general recursive operators } \Theta)[\Theta(\mathcal{C}) \in \mathbf{I}]\}$.

Proposition 9. *Suppose* $\mathbf{I}$ *and* $\mathbf{J}$ *are identification criteria. Suppose* $\mathbf{I} \subseteq \mathbf{J}$. *Then,* $\mathbf{RobustI} \subseteq \mathbf{RobustJ}$.

3 Bārzdiņš' Conjecture for specific criteria

Formally one may state Bārzdiņš' Conjecture (for transformations being general recursive operators) for an identification criterion $\mathbf{I}$ as follows:

Conjecture 10. (Bārzdiņš) Suppose $\mathcal{C} \subseteq \mathcal{R}$. If for all general recursive operators Θ, $\Theta(\mathcal{C}) \in \mathbf{I}$, then $\mathcal{C} \in \mathbf{NUM}$.

Note that Bārzdiņš himself didn't explicitly refer to the kind of operators or transformations to be considered. So one may have different versions of the conjecture for each possible kind of operators considered. In this section we will concentrate on general recursive operators.

First few results below show that Bārzdiņš' Conjecture holds for several natural criteria. First we note that any class in **NUM** is robustly learnable.

Theorem 11. *(Folklore)* **RobustNUM** = **NUM**.

Proof. Suppose $\mathcal{C} \in$ **NUM**. Let $\mathcal{C}'$ be recursively enumerable such that $\mathcal{C} \subseteq \mathcal{C}'$. Then for any general recursive operator Θ, $\Theta(\mathcal{C}) \subseteq \Theta(\mathcal{C}')$, and $\Theta(\mathcal{C}')$ is also recursively enumerable. Thus, $\Theta(\mathcal{C}) \in$ **NUM**. ∎

Based on Popper's refutability principle, the following criteria of inference was considered in [CNM79, CJNM94]

Definition 12. [CNM79, CJNM94] **M** *is* Popperian *iff for all* $f[n]$, $\mathbf{M}(f[n])$ *is a program for a total function.*

PEx = $\{\mathcal{C} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathbf{M}$ *is Popperian, and* $\mathcal{C} \subseteq \mathbf{Ex}(\mathbf{M})]\}$.

It was shown in [BF72, CS83] that **PEx** = **NUM**. Thus,

Theorem 13. **RobustPEx** = **PEx** = **NUM**.

Similar result holds for **NV** (in which machine is supposed to predict the next value instead of coming up with a program to compute f, [Bār71b, BB75]) since **NV** = **NUM** ([Bār71b, BF72, CS83]).

Zeugmann [Zeu86] showed the following theorem using total effective operators instead of general recursive operators. [JSW98] showed (using essentially the same proof as in [Zeu86]) it for general recursive operators.

Theorem 14. *[Zeu86, JSW98] If* $\mathcal{C} \in$ **RobustEx**$_0$*, then* $\mathcal{C}$ *is finite (and thus in* **NUM***).*

Intuitively, a machine is reliable if it does not converge on a function it does not identify.

Definition 15. [Min76, BB75] **M** is *reliable* if for all total f, $\mathbf{M}(f)\downarrow \Rightarrow$ **M** **Ex**-identifies f.

M Reliable-*identifies* $\mathcal{C}$, iff **M** is reliable and **M** **Ex**-identifies $\mathcal{C}$.

Reliable = $\{\mathcal{C} \mid (\exists \mathbf{M})[\mathbf{M}$ **Reliable**-identifies $\mathcal{C}]\}$.

Theorem 16. *[Zeu86]* **RobustReliableEx** ⊆ **NUM**.

[KS89] showed that there exists a class in **RobustNUM** which is not recursively enumerable. Fulk [Ful90] showed that there exists a class in **RobustEx** which is not in **NUM**.

Theorem 17. *[Ful90]* **RobustEx** − **NUM** ≠ ∅.

Thus, Bārzdiņš' Conjecture does not hold for **Ex**-identification. [JSW98] actually showed that even self-referential classes of certain kind are robustly learnable.

Theorem 18. *[JSW98] Let* $\mathcal{C} = \{f \in \mathcal{R} \mid \varphi_{\min(\{x \mid f(x) \neq 0\})} = f\}$. *Then,* $\mathcal{C} \in \mathbf{RobustEx} - \mathbf{NUM}$.

Proof. (from [JSW98]) We first show that $\mathcal{C} \notin \mathbf{NUM}$. Suppose by way of contradiction, $\mathcal{C}'$ is a recursively enumerable superset of $\mathcal{C}$. Let f be a recursive function such that $\mathcal{C}' = \{\varphi_{f(i)} \mid i \in N\}$. Now by implicit use of recursion theorem [Rog67], there exists an e such that,

$$\varphi_e(x) = \begin{cases} 0, & \text{if } x < e; \\ 1, & \text{if } x = e; \\ \varphi_{f(x-e-1)}(x) + 1, & \text{if } x > e. \end{cases}$$

Now $\varphi_e \in \mathcal{C}$. However, for each i, $\varphi_e(i+e+1) = 1 + \varphi_{f(i)}(i+e+1)$, thus $\varphi_e \neq \varphi_{f(i)}$. It follows that $\varphi_e \notin \mathcal{C}'$, contradicting $\mathcal{C} \subseteq \mathcal{C}'$. Thus, $\mathcal{C}$ is not in **NUM**.

We now show that, for any general recursive operator Θ, $\Theta(\mathcal{C}) \in \mathbf{Ex}$.

Suppose Θ is a general recursive operator. Let z be a program for Θ(Zero). Let ProgUnion be a recursive function, mapping finite sets P of programs to programs, such that $\varphi_{\text{ProgUnion}(P)}$ may be defined as follows. Running on input x, program ProgUnion(P) searches for a y, using a fixed dovetailing procedure, such that $(x, y) \in \bigcup_{i \in P} \Theta(\varphi_i)$; if and when such a y is found, $\varphi_{\text{ProgUnion}(P)}(x)$ is defined to be y. Define **M** as follows.

M(σ)
1. If Θ(Zero) is consistent with σ, then output z, the program for Θ(Zero).
2. Otherwise let n be the least number such that Θ(Zero[n]) is inconsistent with σ.
   (* Note that this implies that the input function, if from $\Theta(\mathcal{C})$, is one of $\Theta(\varphi_i)$, $i \leq n$. *)
3. Let $P = \{i \mid i \leq n \ \wedge \ \Theta(\varphi_i)$ (as enumerated in $|\sigma|$ steps) is consistent with $\sigma\}$.
4. Output ProgUnion(P).

End

Now consider any input function $f \in \Theta(\mathcal{C})$. If f is consistent with Θ(Zero), then **M** **Ex**-identifies f due to step 1. If f is not consistent with Θ(Zero), then let n be least number such that Θ(Zero[n]) is inconsistent with f. Now f must be one of $\Theta(\varphi_i)$, $i \leq n$, due to the definition of $\mathcal{C}$. Thus steps 3, 4 ensure that **M** **Ex**-identifies f.

The theorem follows. ∎

The above theorem can in some sense be considered as second order refutation of Bārzdiņš' Conjecture, since not only does it refute Bārzdiņš' Conjecture,

it does so using 'self-referential' classes, for which it was widely believed that Bārzdiņš' Conjecture is true.

Recently [CJSW00] have also considered the Bārzdiņš' Conjecture for some other criteria of learning such as consistent learning. Here the answer depends on which type of consistency is considered. We refer the reader to the above paper for details.

4 Robust Separations

As mentioned in the introduction, many of the separations among learning criteria in inductive inference use self referential tricks. So in a similar spirit to Bārzdiņš' Conjecture, it would be interesting to consider which of these separation results hold robustly. Besides this, other factors as discussed in the introduction, also justify studies about whether the separations in inductive inference hold robustly or not. First we see the following two results from [Ful90] which show that some of the hierarchies of inductive inference do not hold robustly.

Theorem 19. *[Ful90] For all $a \in N \cup \{*\}$,* $\mathbf{RobustEx}^a = \mathbf{RobustEx}$.

Proof. For any function f, let $f_{cyl}(\langle x, y\rangle) = f(x)$. For any class $\mathcal{C} \subseteq \mathcal{R}$, let $\mathcal{C}_{cyl} = \{f_{cyl} \mid f \in \mathcal{C}\}$.

The following three facts can be easily shown.

(1) If $\mathcal{C}_{cyl} \in \mathbf{Ex}^a$, then $\mathcal{C}_{cyl} \in \mathbf{Ex}$.

(2) $\mathcal{C}_{cyl} \in \mathbf{Ex}$ iff $\mathcal{C} \in \mathbf{Ex}$.

(3) There exists a general recursive operator which maps $\mathcal{C}$ to $\mathcal{C}_{cyl}$.

Now suppose $\mathcal{S} \in \mathbf{RobustEx}^a$. We then show that $\mathcal{S} \in \mathbf{RobustEx}$. To see this, consider any general recursive operator Θ. Thus, $\mathcal{C} = \Theta(\mathcal{S}) \in \mathbf{Ex}^a$. Thus $\mathcal{C}_{cyl} \in \mathbf{Ex}^a$, using (3) and the fact that general recursive operators are closed under composition. Thus, $\mathcal{C}_{cyl} \in \mathbf{Ex}$ using (1). Hence, $\mathcal{C} = \Theta(\mathcal{S}) \in \mathbf{Ex}$ using (2). Since, Θ was arbitrary general recursive operator, we have that $\mathcal{S} \in \mathbf{RobustEx}$. ∎

Theorem 20. *[Ful90] For all $a \in N$,* $\mathbf{RobustBc}^a = \mathbf{RobustBc}$.

Note that $\mathcal{R} \in \mathbf{Bc}^*$, and hence $\mathbf{RobustBc}^*$. It was shown in [Jai99, OS99] that **Bc** and **Ex** are separated robustly.

Theorem 21. *[Jai99, OS99]* $\mathbf{RobustBc} - \mathbf{Ex} \neq \emptyset$.

Though the anomaly hierarchy does not stand robustly, the mind change hierarchy does stand robustly, as shown by the following theorem from [JSW98].

Theorem 22. *[JSW98]* $\mathbf{RobustEx}_{n+1} - \mathbf{Ex}_n \neq \emptyset$.

Smith [Smi82] studied learning by a team of machines, and showed a hierarchy based on number of members in a team. Next theorem shows that team hierarchy for learning holds robustly.

Definition 23. [Smi82] (a) $\mathbf{M}'_1, \mathbf{M}'_2, \ldots, \mathbf{M}'_n$, $\mathbf{Team}_n\mathbf{Ex}$-identifies f (written $f \in \mathbf{Team}_n\mathbf{Ex}(\mathbf{M}'_1, \ldots, \mathbf{M}'_n)$, iff at least one of the n machines, $\mathbf{Ex}$-identifies f.
(b) $\mathbf{M}'_1, \mathbf{M}'_2, \ldots, \mathbf{M}'_n$, $\mathbf{Team}_n\mathbf{Ex}$-identifies $\mathcal{C}$, iff $\mathcal{C} \subseteq \mathbf{Team}_n\mathbf{Ex}(\mathbf{M}'_1, \mathbf{M}'_2, \ldots, \mathbf{M}'_n)$.
(c) $\mathbf{Team}_n\mathbf{Ex} = \{\mathcal{C} \mid (\exists \mathbf{M}'_1, \ldots, \mathbf{M}'_n)[\mathcal{C} \subseteq \mathbf{Team}_n\mathbf{Ex}(\mathbf{M}'_1, \ldots, \mathbf{M}'_n)]\}$.

Theorem 24. *[JSW98]* $\mathbf{RobustTeam}_{n+1}\mathbf{Ex} - \mathbf{Team}_n\mathbf{Ex} \neq \emptyset$.

Similar result holds for Team $\mathbf{Bc}$ learning.

Robustness idea can be similarly applied to several other notions in learning. [CJSW00] consider robust learning for criteria involving consistency. [CJO+98] apply robustness to learning with context.

5 Uniform Robust Learning

Quite often it may be useful to consider not only whether one can robustly identify a class, but whether one can effectively get a machine to learn every "transformed" class, using the description of the transformation. For example, if one can learn a certain type of geometric figure, one might expect to be able to learn its transformations via some operations such as rotation, scaling etc, effectively from the parameters of the transformations. This motivates the consideration of uniform robust learning.

In our context, uniform robust learning means that the images (under general recursive operators) of $\mathcal{C}$ are not only learnable, but one can effectively find a machine to learn them. For this strengthened version of robust learning, there have been both positive and negative results. For example, for $\mathbf{Ex}$-learning with a bounded number of mind changes, uniformly robust learning is possible only for finite classes (with restricted cardinality). On the other hand, for standard $\mathbf{Ex}$-learning uniform robust learning seems be achievable in several cases.

Definition 25. [JSW98] A class $\mathcal{C} \in \mathbf{UniformRobustEx}$, iff there exists a recursive function g such that, for all e such that Θ_e is general recursive, $\Theta_e(\mathcal{C}) \subseteq \mathbf{Ex}(\mathbf{M}_{g(e)})$.

$\mathbf{UniformRobustI}$ can be defined similarly for other criteria of inference. Following theorem shows that uniform robust learning is very restrictive for learning with bounded mind changes.

Theorem 26. *[JSW98]* $\mathcal{C} \in \mathbf{UniformRobustEx}_n \Rightarrow \mathcal{C}$ *is finite.*

In fact it can be shown [CJSW00] that $\mathcal{C} \in \mathbf{UniformRobustEx}$ iff $\mathrm{card}(\mathcal{C}) < 2^{n+1}$. Thus, robust and uniform robust $\mathbf{Ex}_n$-learning are separated.

Clearly, since for any general recursive operator Θ and any $\mathcal{C} \in \mathbf{NUM}$, one can effectively (in index for Θ and effective enumeration of $\mathcal{C}$) find an enumeration of $\Theta(\mathcal{C})$, one has

Theorem 27. *[JSW98]* $\mathbf{NUM} = \mathbf{UniformRobustNUM}$.

The next two theorems show the strength of uniform robust **Ex**-learning. Recall the self referential class considered in Theorem 18. The following theorem shows that it is uniformly robustly learnable.

Theorem 28. *[JSW98]* $\mathcal{C} = \{f \in \mathcal{R} \mid \varphi_{\min(\{x \mid f(x) \neq 0\})} = f\} \in$ **UniformRobustEx**.

It is not known at present whether robust and uniform robust **Ex**-learning are separated.

The next theorem shows the power of uniform robust learning (with respect to Bārzdiņš' Conjecture) for learning partial functions.

A *text* is a mapping from N to $N^2 \cup \{\#\}$. A text T is for partial function η if $\text{range}(T) - \{\#\} = \eta$.

Definition 29. [BB75]

(a) **M BlumEx**-*identifies* η (written: $\eta \in$ **BlumEx**(**M**) iff for each text T for η, there is an i such that $\mathbf{M}(T)\downarrow = i$ and $\eta \subseteq \varphi_i$.
(b) **M BlumEx**-identifies a class $\mathcal{S}$ of partial functions, iff **M BlumEx**-identifies each $\eta \in \mathcal{S}$.

Theorem 30. *[JSW98] There exist a class* $\mathcal{C} \subseteq \mathcal{R}$, $\mathcal{C} \notin$ **NUM** *and a recursive function* G *such that, for all* k, $\Theta_k(\mathcal{C}) \subseteq$ **BlumEx**($\mathbf{M}_{G(k)}$).

6 Hyper Robust Learning

Ott and Stephan [OS99] considered a generalized version of robust learning and uniform robust learning. In this they required that the image of $\mathcal{C}$ under every primitive recursive operator be learnable by the *same* machine. The choice of primitive recursive operators above is not crucial, as one can choose any large enough recursively enumerable class of general recursive operators. First let us consider the definition of primitive recursive operator.

A general recursive operator Θ is said to be *primitive recursive*, if there exist two primitive recursive functions g and h such that $\Theta(f)(x) = g(x, f(0)f(1)\cdots f(h(x)))$, for all x.

Definition 31. $[\mathcal{S}] = \{\Theta(f) \mid f \in \mathcal{S}, \Theta \text{ is primitive recursive }\}$.

We now consider hyper robust learning.

Definition 32. [OS99] $\mathcal{S}$ is *hyper robustly* **I** *learnable* (written $\mathcal{S} \in$ **HyperRobustI**) if $[\mathcal{S}] \in$ **I**.

The following theorem shows that choice of primitive recursive operators in the definition of hyper robust learning is not crucial, but we could choose any larger recursively enumerable class of operators.

Theorem 33. *[OS99] Let* $\Theta^0, \Theta^1, \ldots$ *be any recursively enumerable class of general recursive operators. Suppose* $[\mathcal{S}] \in$ **HyperRobustEx**. *Then there exists a* **M** *such that* **M Ex**-*identifies* $\{\Theta^i(f) \mid f \in \mathcal{C}, i \in N\}$.

The above result also holds for **Bc**-learning instead of **Ex**-learning. Clearly, by above theorem **HyperRobustEx** $\subseteq$ **RobustEx**. It can also be shown that **HyperRobustEx** $\subseteq$ **UniformRobustEx**.

Definition 34. $\mathcal{S}$ is said to be *bounded* iff there exists a recursive h such that for all $f \in \mathcal{S}$, for all but finitely many x, $f(x) \leq h(x)$.

Theorem 35. *[OS99] If* $\mathcal{S} \in$ **HyperRobustEx**, *then* $\mathcal{S}$ *is bounded.*

We now consider an interesting connection between classes which are robustly identifiable and closed under finite variants, with classes that are hyperrobustly identifiable.

Theorem 36. *[OS99] Suppose* $\mathcal{S}$ *is closed under finite variants. Then* $\mathcal{S} \in$ **HyperRobustEx** *iff* $\mathcal{S} \in$ **RobustEx**.

The following theorem shows that hyper robust learning satisfies Bārzdiņš' Conjecture.

Theorem 37. *[OS99]* **HyperRobustEx** = **NUM**.

Corollary 38. *[OS99] If* $\mathcal{S}$ *is closed under finite variants and* $\mathcal{S} \in$ **RobustEx**, *then* $\mathcal{S} \in$ **NUM**.

Thus, in some sense if learnability (in **Ex**-sense) does not depend on "finite variations" and also does not use "coding", then identification by enumeration is the only method of learning.

However, such a result does not hold for **Bc** learning.

Theorem 39. *[OS99]* **HyperRobustBc** $-$ **NUM** $\neq \emptyset$.

A surprising result in hyper robust learning is that, though hyperrobust learning is closed under union, hyperrobust team hierarchy stands. The result is surprising because the reason and motivation for team learning came from the non-union theorem of Blum and Blum [BB75].

Theorem 40. *[OS99]* $\mathbf{HyperRobustTeam}_{n+1}\mathbf{Ex} - \mathbf{HyperRobustTeam}_{n}\mathbf{Ex} \neq \emptyset$.

Similarly, hyper robust team hierarchy for **Bc**-learning also stands (though it is not known at present whether **HyperRobustBc** is closed under union or not).

7 Other robustness criteria

The operators we have considered in this survey have been general recursive operators. As mentioned earlier, other kinds of operators may be used instead of general recursive operators. We briefly consider some of the alternatives. Some of the results of this survey are sensitive to which kind of operators are used. We refer the reader to papers cited for details.

Definition 41. [Rog67] An operator Θ mapping partial functions to partial functions is called *effective*, if there exists a computable function f such that, for all i, $\varphi_{f(i)} = \Theta(\varphi_i)$.

Definition 42. [Rog67] An operator Θ is *total*, if it maps every total function to a total function.

Zeugmann [Zeu86] considered robustness under total effective operators. This was actually the first formalization of robust learning.

Note that reason for considering total operators (in total effective and general recursive operators) was that total operators map total functions to total functions, thus giving us $\Theta(\mathcal{C}) \subseteq \mathcal{R}$.

This suggests the following alternative. One may allow all recursive operators which map all functions in the class $\mathcal{C}$ under consideration to total functions. This gives rise to the following robustness notion.

Definition 43. [JSW98] $\mathbf{RecRobustI} = \{\mathcal{C} \subseteq \mathcal{R} \mid \mathcal{C} \in \mathbf{I} \ \wedge \ (\forall \text{ recursive operators } \Theta \mid \Theta(\mathcal{C}) \subseteq \mathcal{R})[\Theta(\mathcal{C}) \in \mathbf{I}]\}$.

One of the problems with above definition is that **RecRobustI** is not closed under subset! Due to this one has a funny situation that a class may be **RecRobust**-ly learnable, but some of its subsets are not! Since this situation is not desirable, the above robustness notion is not much interesting.

Another possibility is to consider all recursive operators, but only require learnability of total functions in $\Theta(\mathcal{C})$. This gives rise to the following definition.

Definition 44. $\mathbf{RRobustI} = \{\mathcal{C} \subseteq \mathcal{R} \mid \mathcal{C} \in \mathbf{I} \wedge (\forall \text{ recursive operators } \Theta)[\Theta(\mathcal{C}) \cap \mathcal{R} \in \mathbf{I}]\}$.

Above notion is closed under subset. However, note that the class of constant functions, $\{f \mid (\forall x)[f(x) = f(0)]\} \notin \mathbf{RRobustEx}$. Thus, $\mathbf{NUM} \not\subseteq \mathbf{RRobust}$. This casts some doubts about the interestingness of the above notion.

Alternatively, one may consider learnability of partial functions as done in Definition 29 and Theorem 30, which allows one to consider all recursive operators.

It is not clear which of the above approaches is the "best" if any. Possibly, each approach is justified, if it gives interesting results.

Due to the natural appeal of robust learning we expect that in future many more interesting results on robust learning will be studied.

8 Acknowledgements

Sanjay Jain was supported in part by NUS grant number RP3992710. We acknowledge the various papers on robust learning for the motivations and the results which formed the basis of this survey.

References

[AF96] A. Ambainis and R. Freivalds. Transformations that preserve learnability. In S. Arikawa and A. Sharma, editors, *Algorithmic Learning Theory: Seventh International Workshop (ALT '96)*, volume 1160 of *Lecture Notes in Artificial Intelligence*, pages 299–311. Springer-Verlag, 1996.

[AS83] D. Angluin and C. Smith. A survey of inductive inference: Theory and methods. *Computing Surveys*, 15:237–289, 1983.

[Bār71a] J. Bārzdiņš. *Complexity and Frequency Solution of Some Algorithmically Unsolvable Problems.* PhD thesis, Novosibirsk State University, 1971. In Russian.

[Bār71b] J. Bārzdiņš. Prognostication of automata and functions. *Information Processing*, 1:81–84, 1971.

[Bār74] J. Bārzdiņš. Two theorems on the limiting synthesis of functions. In *Theory of Algorithms and Programs, vol. 1*, pages 82–88. Latvian State University, 1974. In Russian.

[BB75] L. Blum and M. Blum. Toward a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.

[BF72] J. Bārzdiņš and R. Freivalds. On the prediction of general recursive functions. *Soviet Mathematics Doklady*, 13:1224–1228, 1972.

[Blu67] M. Blum. A machine-independent theory of the complexity of recursive functions. *Journal of the ACM*, 14:322–336, 1967.

[CJNM94] J. Case, S. Jain, and S. Ngo Manguelle. Refinements of inductive inference by Popperian and reliable machines. *Kybernetika*, 30:23–52, 1994.

[CJO+98] J. Case, S. Jain, M. Ott, A. Sharma, and F. Stephan. Robust learning aided by context. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 44–55. ACM Press, 1998.

[CJSW00] J Case, S. Jain, F. Stephan, and R. Wiehagen. Robust learning: Rich and poor. Technical report, Centre for Learning Systems and Applications, Department of Computer Science, University of Kaiserslautern, Germany, 2000.

[CNM79] J. Case and S. Ngo Manguelle. Refinements of inductive inference by Popperian machines. Technical Report 152, SUNY/Buffalo, 1979.

[CS83] J. Case and C. Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25:193–220, 1983.

[Fre91] R. Freivalds. Inductive inference of recursive functions: Qualitative theory. In J. Bārzdiņš and D. Bjorner, editors, *Baltic Computer Science*, volume 502 of *Lecture Notes in Computer Science*, pages 77–110. Springer-Verlag, 1991.

[Ful90] M. Fulk. Robust separations in inductive inference. In *31st Annual IEEE Symposium on Foundations of Computer Science*, pages 405–410. IEEE Computer Society Press, 1990.

[Gol67] E. M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[Jai99] S. Jain. Robust behaviorally correct learning. *Information and Computation*, 153(2):238–248, September 1999.

[JORS99] S. Jain, D. Osherson, J. Royer, and A. Sharma. *Systems that Learn: An Introduction to Learning Theory.* MIT Press, Cambridge, Mass., second edition, 1999.

[JSW98] S. Jain, C. Smith, and R. Wiehagen. On the power of learning robustly. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 187–197. ACM Press, 1998.

[KS89] S. Kurtz and C. Smith. On the role of search for learning. In R. Rivest, D. Haussler, and M. Warmuth, editors, *Proceedings of the Second Annual Workshop on Computational Learning Theory*, pages 303–311. Morgan Kaufmann, 1989.

[KW80] R. Klette and R. Wiehagen. Research in the theory of inductive inference by GDR mathematicians – A survey. *Information Sciences*, 22:149–169, 1980.

[Min76] E. Minicozzi. Some natural properties of strong identification in inductive inference. *Theoretical Computer Science*, pages 345–360, 1976.

[OS99] M. Ott and F. Stephan. Avoiding coding tricks by hyperrobust learning. In P. Vitányi, editor, *Fourth European Conference on Computational Learning Theory*, volume 1572 of *Lecture Notes in Artificial Intelligence*, pages 183–197. Springer-Verlag, 1999.

[Rog67] H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967. Reprinted by MIT Press in 1987.

[Smi82] C. Smith. The power of pluralism for automatic program synthesis. *Journal of the ACM*, 29:1144–1165, 1982.

[Wey52] H. Weyl. *Symmetry*. Princeton University Press, 1952.

[Zeu86] T. Zeugmann. On Bārzdiņš' conjecture. In K. P. Jantke, editor, *Analogical and Inductive Inference, Proceedings of the International Workshop*, volume 265 of *Lecture Notes in Computer Science*, pages 220–227. Springer-Verlag, 1986.