Robust Learning Aided by Context*

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

Sanjay Jain
Department of Information Systems
and Computer Science
National University of Singapore
Singapore 119260, Republic of Singapore
sanjay@iscs.nus.edu.sg

Matthias Ott†
Institut für Logik, Komplexität
und Deduktionssysteme
Universität Karlsruhe
76128 Karlsruhe, Germany
m_ott@ira.uka.de

Arun Sharma‡
School of Computer Science and Engineering
University of New South Wales
Sydney 2052, Australia
arun@cse.unsw.edu.au

Frank Stephan§
Mathematisches Institut
Universität Heidelberg
69120 Heidelberg, Germany
fstephan@math.uni-heidelberg.de

Abstract

Empirical studies of *multitask* learning provide some evidence that the performance of a learning system on its intended targets improves by presenting to the learning system related tasks, also called *contexts*, as additional input. Angluin, Gasarch, and Smith, as well as Kinber, Smith, Velauthapillai, and Wiehagen have provided mathematical justification for this phenomenon in the inductive inference framework. However, their proofs rely heavily on self-referential coding tricks, that is, they directly code the solution of the learning problem into the context. Fulk has shown that for the **Ex**- and **Bc**-anomaly hierarchies, such results, which rely on self-referential coding tricks, may not hold *robustly*. In this work we analyze robust versions of learning aided by context and show that — in contrast to Fulk's result above — the *robust versions* of these learning notions are still very powerful. Also, studied is the difficulty of the functional dependence between the intended target tasks and useful associated contexts.

1 Introduction

There is empirical evidence that in many cases performance of learning systems improves when they are modified to learn auxiliary, "related" tasks (called *contexts*) in addition to the primary tasks of interest [6, 7, 19]. For example, an experimental system to predict the value of German *Daimler* stock performed better when it was modified to track *simultaneously* the German stock-index DAX [2]. The value of the Daimler stock here is the primary or target concept and the value of the DAX—a related concept—provides useful auxiliary context. The additional task of recognizing road stripes was able to improve empirically the performance of a system learning to steer a car to follow the road [7]. Other examples where multitask learning has successfully been applied to real world problems appear in [9, 20, 23, 27].

Importantly, these empirical phenomena of such *context sensitivity* in machine learning [19] is also supported, for example, by *mathematical existence theorems* for this phenomena (and variants) in the inductive inference framework [1]. More technical theoretical work appears in [15, 17]. For a Bayesian PAC-style approach to multitask learning see [4].

The theoretical papers, [1, 15], provide theorems in the inductive inference framework, witnessing situations in which learnability *absolutely* (not just empirically) passes from impossible to possible in the presence of suitable auxiliary contexts to be learned. These theorems are proved there by means of self-referential coding tricks, where, in effect, correct hypotheses for the pri-

*This work was carried out while J. Case, S. Jain, M. Ott, and F. Stephan were visiting the School of Computer Science and Engineering at the University of New South Wales.

†Supported by the Deutsche Forschungsgemeinschaft (DFG) Graduiertenkolleg "Beherrschbarkeit komplexer Systeme" (GRK 209/2-96).

‡Supported by Australian Research Council Grant A49600456.

§Supported by the Deutsche Forschungsgemeinschaft (DFG) Grant Am 60/9-2.

mary tasks are coded into the auxiliary contexts. The use of such coding has justifiably been criticized on the grounds of resorting to artificial tricks. In the present paper we attempt to address this criticism and analyze several notions of learning in the presence of context.

Based on a suggestion of Bārzdiņš, Fulk [12] proposed a stricter notion of *robust identification* with a view to avoiding self-referential coding tricks. He showed that several important results like the **Ex**- and **Bc**-anomaly hierarchies, which had been established using self-referential coding tricks [8], did not hold robustly. While it was earlier believed that robust identification avoids all self-referential coding tricks, Jain, Smith, and Wiehagen [14] have recently shown that it only avoids certain kinds of coding tricks. This result notwithstanding, establishing robust analogs of results demonstrating advantages of learning in the presence of context considerably strengthens their claim. We employ Fulk's notion to *robust identification* to show that for several, partially new, models of learning from context, their robust analog is still more powerful than conventional identification. In Section 5, we present results about the problem of *finding useful auxiliary contexts* to enable the *robust* learning of classes of functions which might not be learnable without such contexts.

Before we proceed formally, we devote the rest of this section to a discussion of robustness and of various models of learning in the presence of context considered in this paper.

1.1 Robust Identification

In this section we introduce the notion of robust identification and discuss its effectiveness and limitations in avoiding self-referential coding tricks. We begin with a definition of **Ex**-identification.

A machine M **Ex**-*identifies* a computable function f just in case M, fed the graph of f, outputs a sequence of programs eventually converging to a program for f [5, 8]. A class of functions S is **Ex**-*identifiable* just in case there is a machine that **Ex**-identifies each member of S.

Here is a particularly simple example of a self-referential coding trick. Let $\mathcal{SD} = \{$ computable $f \mid f(0)$ is a program for $f\}$. Clearly, $\mathcal{SD}$ is **Ex**-identifiable since a machine on $f \in \mathcal{SD}$ need only wait for the value $f(0)$ and output it.[1] However, the **Ex**-identification of $\mathcal{SD}$ severely depends on programs for its members being coded into the values (at zero) of those members.

In the 1970's, Bārzdiņš was concerned, among other things, with how to formulate that an existence result in function learnability was proved to hold *without resort to such self-referential coding tricks*. He, in effect, reasoned that instead of the self-referential witness, one should construct a function class S and then show that the desired result holds for *any* class S' which can be obtained from S by applying a general recursive operator to all functions in S. The idea was that a suitable general recursive operator would irretrievably scramble the coding tricks embedded in the self-referential class. For example, for $\mathcal{SD}$ itself, consider the operator Ψ^L such that, for all partial functions ψ, for all x, $\Psi^L(\psi)(x) = \psi(x+1)$. Ψ^L essentially "shifts the partial function ψ to the left." It is easy to see that $\Psi^L(\mathcal{SD}) = REC$, the class of *all* computable functions. It is well known that REC is not **Ex**-identifiable; hence, Ψ^L transforms the identifiable class $\mathcal{SD}$ into an unidentifiable class REC by removing the self-referential information from $\mathcal{SD}$ that made it identifiable.

Motivated by the above proposal by Bārzdiņš, Fulk [12] defined a class $S \subseteq REC$ to be *robustly* **Ex**-*identifiable* just in case for all general recursive operators Ψ, $\Psi(S)$ is **Ex**-identifiable. Thus, the class $\mathcal{SD}$ is **Ex**-identifiable but not robustly **Ex**-identifiable. Fulk also showed that other important results in function learning like the **Ex**- and **Bc**-anomaly hierarchies, which had been established using self-referential coding tricks [8], did not hold robustly. On the other hand, Jain, Smith, and Wiehagen [14] have recently shown that the mind change hierarchy holds robustly. So, in some sense, results that hold robustly may be considered "strong" as they appear to hold without resorting to coding tricks. However, as the following discussion demonstrates, robustness avoids only certain kinds of coding tricks.

Consider the class

$$\mathcal{C} = \{f \mid (\exists x)[f(x) \neq 0] \text{ and } \min\{x \mid f(x) \neq 0\} \text{ is a program for } f\}.$$

Certainly, $\mathcal{C}$ is defined by a self-referential trick. However, as shown in [14], $\mathcal{C}$ is *robustly* learnable, by the following argument. Fix a general recursive operator Θ. Let $g \in \Theta(\mathcal{C})$ be given. We write f_0 for the constant 0-function. If $g = \Theta(f_0)$ then every program for $\Theta(f_0)$ is also a program for g. Otherwise, if $g \neq \Theta(f_0)$, a learner will eventually find an x with $g(x) \neq \Theta(f_0)(x)$. Having this information, the learner can effectively compute an n such that g is inconsistent with $\Theta(0^n)$. This implies that n is an upper bound for the minimal program for any $f \in \mathcal{C}$, with $\Theta(f) = g$. That is, there exists a program $e \leq n$ such that $\varphi_e \in \mathcal{C}$, and $g = \Theta(\varphi_e)$. Computing programs for all such possible $\Theta(\varphi_e)$, $e \leq n$, yields an upper bound for a program for g. But it is well known that one can **Ex**-identify a computable function, in our case g, when an upper bound on one of its programs is known [11].

Thus, though $\mathcal{C}$ is defined by a self-referential class, $\mathcal{C}$ is *robustly* learnable. This, in particular, refutes Bārzdiņš' (and other's) belief, that a suitable general recursive operator can destroy every kind of self-referential coding trick. Rather, as already noted in [14], robustness rules out "purely numerical" coding tricks like that of $\mathcal{SD}$, but it still allows "topological" coding tricks as present in the class $\mathcal{C}$.

The above discussion notwithstanding, there is clear merit in showing that a result holds robustly. In this work we follow the flavor of Jain, Smith, and Wiehagen [14] and show, for several models of learning aided by

[1] And it's a very large class of computable functions. Blum and Blum [5] essentially show that such classes contain a finite variant of each computable function!

context, that many interesting existence theorems even *hold robustly*.

1.2 Models of Learning Aided by Context

We now describe the models of learning in the presence of context and the results presented in this paper. To aid in our discussion, we first define the notion of **Bc**-identification.

A machine M **Bc**-*identifies* computable f just in case M, fed the graph of f, outputs a sequence of programs and beyond some point in this sequence all the programs compute f [3, 8].

In Section 3 we consider essentially the model of Kinber, Smith, Velauthapillai, and Wiehagen [15]. They defined this notion using finite learning, that is, **Ex**-style learning without any mind changes. We directly introduce the notion for **Ex**-style learning (which, thus, contains finite learning as a special case): a learner M is said to (a,b)**Ex**-*identify* a set of b pairwise distinct functions just in case M, fed graphs of the b functions simultaneously, **Ex**-identifies at least a of them. [15] showed that for all a, there is a class of functions S which cannot even be **Bc**-identified but which is $(a,a+1)$**Ex**-identifiable with 0 mind changes. As we show in Section 3 below, this result also holds for *robust* $(a,a+1)$**Ex**-learning, although no longer with 0 mind changes. However, an only slightly weaker version of this result also holds robustly for finite learning: there is a class which is robustly $(a,a+2)$**Ex**-learnable with 0 mind changes, but which is not in **Bc**.

The above model of parallel learning may be viewed as learning from arbitrary context. No distinction is made between which function is the target concept and which function provides the context. Let $R \subseteq REC \times REC$ be given. Intuitively, for $(f,g) \in R$, f is the target function and g is the context. We say the class R is **ConEx**-*identifiable* if there exists a machine which, upon being fed graphs of $(f,g) \in R$ (suitably marked as target and context), converges in the limit to a program for f. Now, we define a class of functions $S \subseteq REC$ to be **SelEx**-*identifiable* if there exists a mapping $C\colon S \to S$ such that $\{(f,C(f)) \mid f \in S\}$ is **ConEx**-identifiable. Here, C may be viewed as a *context mapping* for the concept class S. Of course, the freedom to choose any computable context is very powerful since, then, even REC can be **SelEx**-identified with 0 mind changes. To see this, just consider a mapping C that maps each $f \in REC$ to a computable function g such that $g(0)$ codes a program for f. Then, after reading $(f(0), g(0))$, a machine need only output $g(0)$. Of course, this natural proof resorts to a purely numerical coding trick. Nevertheless, as we show in Theorem 4.4 of Section 4, the class REC is even *robustly* **SelEx**-identified, although no longer with 0 mind changes!

The model of **SelEx**-identification is similar to the parallel learning model of Angluin, Gasarch, and Smith [1] where they require the learner to output also a program for the context g. Our Theorem 4.5 in Section 4 is a *robust* version of their [1, Theorem 6].

Though REC is *robustly* **SelEx**-learnable, the appropriate context mappings may be uncomputable with unpleasant Turing complexity. For this reason, we also investigate the nature of the appropriate context mapping to gain some understanding of the functional dependence between the target (primary task) and the context. In particular, we look for example classes that are **SelEx**-identifiable or robustly **SelEx**-identifiable but are not **Ex**-identifiable and are such that the context mapping may be more feasible. We consider two approaches to implement the context mappings: operators, which work on values of the target function, and program mappings, which work on programs for the target function. As a sample result we are able to show that if the functional dependence between the target function and the context is "too high" then the presence of context is not of much help as the class is learnable without any context.

2 Preliminaries

The set of natural numbers, i.e., the set of the nonnegative integers, is denoted by ω. If $A \subseteq \omega^n$, we write $A|_i = \{x_i \mid (x_1,\dots,x_i,\dots,x_n) \in A\}$ for the projection to the i-th component. We are using an acceptable programming system $\varphi_0, \varphi_1, \dots$ for the class of all partial computable functions [24, 26]. $MinInd(f) = \min\{e \mid \varphi_e = f\}$ is the minimal index of a partial computable function f with respect to this programming system. REC denotes the set of all (total) computable functions; $REC_{0,1}$ denotes the class of all $\{0,1\}$-valued functions from REC. A class $S \subseteq REC$ is *computably enumerable* if S is empty or $S = \{\varphi_{h(i)} \mid i \in \omega\}$ for some $h \in REC$.

$Seq = \omega^*$ is the set of all *finite* sequences from ω. For strings $\sigma, \tau \in Seq \cup \omega^\omega$, $\sigma \preceq \tau$ means that σ is an initial segment of τ. $|a_1 \dots a_n| = n$ denotes the length of a string $a_1 \dots a_n \in Seq$. Total functions $f\colon \omega \to \omega$ are identified with the infinite string $f(0)f(1)\dots \in \omega^\omega$. We write $f[n]$ for the initial segment $f(0)\dots f(n-1)$ of a total function f. For sets $D \subseteq S \subseteq \omega^\omega$ we say that D is a *dense* subset of S, if $(\forall f \in S)(\forall n)(\exists g \in D)[f[n] \preceq g]$.

Let $\mathcal{P}$ denote the class of all partial functions mapping ω to ω. For $\varphi, \psi \in \mathcal{P}$ we write $\varphi \subseteq \psi$ if $(\forall x)[\varphi(x)\downarrow \implies \psi(x)\downarrow = \varphi(x)]$. Mappings $\Theta : \mathcal{P} \to \mathcal{P}$ are called operators. An operator Θ is *recursive* if, for all finite functions α, one can effectively enumerate all (x,y) with $\Theta(\alpha)(x)\downarrow = y$, and furthermore, Θ is

- monotone, that is, $(\forall \varphi, \psi \in \mathcal{P})[\varphi \subseteq \psi \implies \Theta(\varphi) \subseteq \Theta(\psi)]$, and
- compact, that is, $(\forall \varphi \in \mathcal{P})[\Theta(\varphi)(x)\downarrow = y \implies (\exists \alpha \subseteq \varphi)[\alpha \text{ finite and } \Theta(\alpha)(x)\downarrow = y]]$.

An operator $\Theta : \mathcal{P} \to \mathcal{P}$ is *general* if $\Theta(f)$ is total for all total functions f. For every general recursive operator C there exists a general recursive operator C' such that for all total f, $C'(f) = C(f)$, and

(*) for all τ, $\{x \mid C'(\tau)(x)\downarrow\}$ is finite, and a canonical index [25] for it can be effectively determined from τ.

Note that such a C' can easily be constructed by "slowing down" C appropriately. Since we are only interested in the properties of operators on total functions, we may restrict our attention to general recursive operators satisfying condition $(*)$. Let $\Theta_0, \Theta_1, \ldots$ be a (noneffective) listing of all general recursive operators satisfying condition $(*)$.

The quantifier $(\overset{\infty}{\forall} n)$ abbreviates $(\exists m)(\forall n \geq m)$. *Learning machines* are typically total Turing machines which compute some mapping $Seq^m \to (\omega \cup \{?\})^n$. Intuitively, output of ? by M indicates that it has not made up its mind about the hypothesis. It is not necessary to consider ? when one is considering **Ex** or **Bc** identification, but it is useful when one considers number of mind changes. **Bc** is the class of all subsets S of REC such that there exists a learning machine $M : Seq \to \omega \cup \{?\}$ with

$$(\forall f \in S)(\overset{\infty}{\forall} n)[\varphi_{M(f[n])} = f].$$

S is in **Ex** if there exists a learning machine M such that

$$(\forall f \in S)(\exists e)[\varphi_e = f \wedge (\overset{\infty}{\forall} n)[\varphi_{M(f[n])} = e]].$$

M makes a mind change at stage $n+1$ on input f if $? \neq M(f[n]) \neq M(f[n+1])$. A learner M learns S *finitely* iff for every $f \in S$, M, fed graph of f, **Ex**-learns f without any mind changes, that is, M outputs only one program (not counting initial ?s), and this program is correct for f. **Fin** denotes the collection of all finitely learnable classes, that is, the classes which are **Ex**-learnable *without any mind changes.* It is well known that **Fin** $\subset$ **Ex** $\subset$ **Bc** and $REC_{0,1} \notin$ **Bc** (see, e.g., [22]).

3 Learning From Arbitrary Contexts

A very restricted form of learning aided by context arises when we require that the learning machine be successful with *any* context from the concept class under consideration. In this case it is most natural (as argued below) to look at the learning problem in a symmetric manner, that is, we do not distinguish between the target function and the context. Instead, we treat each input function with the same importance and try to learn programs for each of them (but may only be successful on some of the input functions). However, in this case we do have to require that the input functions are pairwise different; otherwise, we do not get a different learning notion, since the ordinary **Ex**-learning problem would reduce to such a learning type. The resulting learning notion, which we formally introduce in the next definition, has essentially already been introduced and studied by Kinber, Smith, Velauthapillai and Wiehagen [15, 16] (see also the work of Kummer and Stephan [17]).

Definition 3.1 $S \subseteq REC$ is in (a,b)**Ex** if there exists a learning machine M such that for all pairwise distinct $f_1, \ldots, f_b \in S$:

$$(\exists i_1 < \ldots < i_a)(\exists e_1, \ldots, e_a)(\forall j, 1 \leq j \leq a)$$
$$[\varphi_{e_j} = f_{i_j} \wedge (\overset{\infty}{\forall} n)[M(f_1[n], \ldots, f_b[n])|_{i_j} = e_j]].$$

In the literature, so far only the very restrictive finite identification variant of (a,b)**Ex** has been studied, in which the learner has to correctly infer a out of b given functions *without any mind changes.* For this version, it was shown in [15] that, for all a, there is a class S of functions which is not in **Bc** but is $(a, a+1)$**Ex**-learnable without any mind changes. Thus, presenting $a+1$ functions of a non-**Bc**-learnable class in parallel may allow finite learnability of at least a of the $a+1$ functions. This result appears to provide a very strong case the usefulness of parallel learnability. However, the proof of this result uses a purely numerical coding trick. More precisely, each non-empty finite subset F of S contains one function which holds programs for all other functions of F in its values. Thus, it is interesting to see whether this result also holds for the following *robust* version of learning with arbitrary context.

Definition 3.2 $S \subseteq REC$ is in (a,b)**RobEx** if $\Theta(S) \in (a,b)$**Ex** for all general recursive operators Θ.

As the next theorem shows there are still classes in $(a, a+1)$**RobEx** $-$ **Bc**, that is, the existence result from [15] holds robustly, although no longer with 0 mind changes.[2] In the proof of Theorem 3.4, and at several other places, we will use the following consequence of the result of Freivalds and Wiehagen, that REC, the class of all the computable functions, can be identified in the **Ex** sense, if one is given an upper bound on the minimal program for the input function in addition to the graph of the input function [11] (see also [13]).

Fact 3.3 (Freivalds and Wiehagen [11]) *Let $S \subseteq REC$. If there exists a learning machine M such that*

$$(\forall f \in S)(\exists c \geq MinInd(f))(\overset{\infty}{\forall} n)[M(f[n]) = c],$$

then S is in **Ex**.

Theorem 3.4 $(a, a+1)$**RobEx** $\not\subseteq$ **Bc** *for all $a \in \omega$.*

Proof. Let $M_0, M_1, \ldots$ be an enumeration of all learning machines. We inductively define functions $g_0, g_1, \ldots$ and sequences $\sigma_0, \sigma_1, \ldots$ below. Suppose we have defined g_i, σ_i, for $i < n$. Then define g_n and σ_n as follows:

(1) Choose g_n such that, (a) for $i \leq n$, M_i does not **Bc**-infer g_n, and (b) if $n > 0$, then $g_n \succeq \sigma_{n-1}$.

(2) Choose $\sigma_n \preceq g_n$ such that, (a) for all $m \leq n$, for all $x \leq MinInd(g_n)$, $\Theta_m(\sigma_n)(x)\downarrow$ and (b) if $n > 0$, then $\sigma_n \succeq \sigma_{n-1}$.

[2] This result can be "improved" to show that there are classes that are not in **Bc**, but which can be robustly $(a, a+2)$-finite identified (i.e., with 0 mind changes). Furthermore, one can show that there are classes that are not in **Bc**, but which can be robustly $(a, a+1)$-finite identified if one is willing to tolerate a finite number of errors in the output programs. It is open at present whether $(a, a+1)$**RobFin** $-$ **Bc** $\neq \emptyset$, where **RobFin** is **RobEx** with 0-mind changes.

Now let $S = \{g_n \mid n \in \omega\}$. By construction M_i does not **Bc**-identify g_n, for $n \geq i$. Thus, $S \notin$ **Bc**.

We now show $S \in (a, a+1)$**RobEx** for $a = 1$. The generalization to arbitrary $a \in \omega$ is straightforward. Suppose an arbitrary general recursive operator Θ_k is given. We need to show that $\{\Theta_k(g_n) \mid n \in \omega\} \in (1,2)$**Ex**. Note that (a,b)**Ex** is closed under union with finite sets. So, by Fact 3.3, it suffices to construct a machine M such that, for all i and j satisfying $i, j \geq k$, and $\Theta_k(g_i) \neq \Theta_k(g_j)$, $M(\Theta_k(g_i), \Theta_k(g_j))\downarrow \geq \min\{MinInd(\Theta_k(g_i)), MinInd(\Theta_k(g_j))\}$.

Let e_r be a program, obtained effectively from r, for $\Theta_k(\varphi_r)$. Define M as follows.

$$M(f_1[n], f_2[n]) = \begin{cases} 0, & \text{if } f_1[n] = f_2[n]; \\ \max\{e_r \mid r \leq y\}, & \text{if } y = \min\{x \mid f_1(x) \neq f_2(x)\}. \end{cases}$$

We claim that for all i and j such that $i, j \geq k$, and $\Theta_k(g_i) \neq \Theta_k(g_j)$, $M(\Theta_k(g_i), \Theta_k(g_j))\downarrow \geq \min\{MinInd(\Theta_k(g_i)), MinInd(\Theta_k(g_j))\}$. To see this, suppose $i, j \geq k$, $f_1 = \Theta_k(g_i)$, $f_2 = \Theta_k(g_j)$, and $f_1 \neq f_2$. Let $r = \min\{i, j\}$. Thus, $\sigma_r \preceq g_i$ and $\sigma_r \preceq g_j$. It follows by (2) above that, for all $x \leq MinInd(g_r)$, $f_1(x) = f_2(x)$. Thus, $\min\{x \mid f_1(x) \neq f_2(x)\} \geq MinInd(g_r)$. It follows that $M(f_1, f_2) \geq \max\{e_{r'} \mid r' \leq MinInd(g_r)\}$. Thus, $M(f_1, f_2) \geq \min\{MinInd(f_1), MinInd(f_2)\}$. Theorem follows. □

In addition to **Bc**, one can also show for all other inference types **IT**, which do not contain a cone $\{f \in REC \mid \sigma \preceq f\}$ for any σ, that $(a, a+1)$**Ex** contains classes which are not in **IT**. This is achieved by suitably modifying (1) in the just above proof. For example, for all non-high sets A there exist $(a, a+1)$**RobEx**-inferable classes which are not in **Ex**[**A**] (see [10, 18]).[3] However, along the lines of [15] it follows that (b,b)**Ex** = **Ex**, in particular, (b,b)**RobEx** = **RobEx** for all $b \geq 1$. Thus, it is not possible to improve Theorem 3.4 to (b,b)**RobEx**-learning.

Furthermore, one may wonder whether it is possible to guarantee that an (a,b)**Ex**-leaner always correctly infers, say, the first of the b input functions. This means that we declare the first function as the target function and all other functions as context. However, one can show that this yields exactly the class **Ex**, by choosing the context functions always from a set $F = \{g_1, g_2, \ldots, g_b\}$ of cardinality b. Then, on any input function f, we simulate the (a,b)**Ex**-learner on $(f, h_{i_1}, \ldots, h_{i_{b-1}})$, where $Y = \{h_{i_1}, \ldots, h_{i_{b-1}}\}$ is a subset of $F - \{f\}$ containing $b-1$ functions. Thus, variants of learning with arbitrary context, where a target function is designated, do not increase the learning power compared to an ordinary **Ex**-learner.

4 Learning From Selected Contexts

In Section 3 we have established that an arbitrary context may be enough to increase the robust learning ability, if one is willing to pay the price of not learning at most one of the input functions. Of course, on intuitive grounds, it is to be expected that the learning power can be further increased if the context given to the learner is not arbitrary, but is carefully *selected*. In order to formally define such a notion of learning from *selected* context, we first introduce the notion of asymmetric learning from context. This notion is asymmetric since, in contrast to Section 3, here we distinguish between the target function and the context:

Definition 4.1 $P \subseteq REC \times REC$ is in **ConEx** if there exists a learning machine M that for all $(f,g) \in P$:

$$(\exists e)[\varphi_e = f \wedge (\stackrel{\infty}{\forall} n)[M(f[n], g[n]) = e]].$$

For $(f,g) \in P$ we call f the *target* and g the *context* function.

The concept **ConEx** is related to the notion of parallel learning studied by Angluin, Gasarch and Smith [1]. The main difference being that in the learning type from [1], the learning machine was required to infer programs for both input functions, not just the target. In Theorem 4.5 we will also present a robust version of one of the results from [1] concerning parallel learning.

The next definition formally introduces the notion of learning in the presence of a *selected context*:

Definition 4.2 $S \subseteq REC$ is in **SelEx** if there exists a mapping $C : S \to S$ such that the class $S^C := \{(f, C(f)) \mid f \in S\}$ is in **ConEx**. C is called a context mapping for S.

As was discussed in the introductory section, using a purely numerical coding trick, one can easily see that the freedom of carefully selecting a context yields extreme increases in learning power. Indeed, the entire class REC is in **SelEx** without any mind changes.

However, if we consider the robust version of **SelEx**, as specified in Definition 4.3 below, we can still show that freely selecting a context makes it possible to learn the class of all computable functions.

Definition 4.3 $P \subseteq REC \times REC$ is in **RobConEx** if the class $\Theta(P) := \{(\Theta(f), \Theta(g)) \mid (f,g) \in P\}$ is in **ConEx** for all general recursive operators Θ.

$S \subseteq REC$ is in **RobSelEx** if there exists a context mapping $C : S \to S$ such that the class S^C is in **RobConEx**.

Theorem 4.4 *If $S \subseteq REC$ contains a dense, computably enumerable subclass, then $S \in$* **RobSelEx**. *In particular, $REC \in$* **RobSelEx**.

Proof. The proof is based on similar ideas as that of Theorem 3.4. Let $f_0, f_1, \ldots$ be a (not necessarily computable) listing of the functions in S, and $\{\varphi_{h(i)}\}_{i \in \omega}$, with $h \in REC$, be a dense subset of S. For each n choose a sequence $\sigma_n \preceq f_n$ such that

$$(\forall m \leq n)(\forall x \leq MinInd(f_n))[\Theta_m(\sigma_n)(x)\downarrow]. \quad (1)$$

We define the context mapping $C : S \to S$ by $C(f_n) = \varphi_{h(i)}$ for the least i such that $\sigma_n \preceq \varphi_{h(i)}$.

---
[3] A is *high* iff $K' \leq_T A'$.

We want to show that S^C is in **RobConEx**. Let an arbitrary general recursive operator Θ_k be given. We need to show that $\{(\Theta_k(f), \Theta_k(C(f))) \mid f \in S\} \in$ **ConEx**. Note that **ConEx** is closed under union with finite sets. So, by Fact 3.3, it suffices to construct a machine M such that, for all $n \geq k$, (i) if $\Theta_k(f_n) = \Theta_k(C(f_n))$, then $M(\Theta_k(f_n), \Theta_k(C(f_n)))\downarrow$ to a program for $\Theta_k(f_n) = \Theta_k(C(f_n))$, and (ii) if $\Theta_k(f_n) \neq \Theta_k(C(f_n))$, then $M(\Theta_k(f_n), \Theta_k(C(f_n)))\downarrow \geq MinInd(\Theta_k(f_n))$.

Let e_r be a program, obtained effectively from r, for $\Theta_k(\varphi_r)$. Define M as follows.

$$M(f[n], g[n]) = \begin{cases} e_{h(r)} & \text{if } f[n] = g[n], \text{ and} \\ & r = \min\{r' \mid g[n] \preceq \Theta_k(\varphi_{h(r')})\}; \\ \max\{e_r \mid r \leq y\} & \text{if } y = \min\{x \mid f(x) \neq g(x)\}. \end{cases}$$

We claim that for all $n \geq k$ (i) if $\Theta_k(f_n) = \Theta_k(C(f_n))$, then $M(\Theta_k(f_n), \Theta_k(C(f_n)))\downarrow$ to a program for $\Theta_k(f_n) = \Theta_k(C(f_n))$, and (ii) if $\Theta_k(f_n) \neq \Theta_k(C(f_n))$, then $M(\Theta_k(f_n), \Theta_k(C(f_n)))\downarrow \geq MinInd(\Theta_k(f_n))$.

To see this, consider any $n \geq k$. If $\Theta_k(f_n) = \Theta_k(C(f_n))$, then in particular, we will have $\Theta_k(f_n) \in \{\Theta_k(\varphi_{h(r')}) \mid r' \in \omega\}$. Thus, first clause in the definition of M ensures that M **ConEx**-identifies $(\Theta_k(f_n), \Theta_k(C(f_n)))$. If $\Theta_k(f_n) \neq \Theta_k(C(f_n))$, then by definition of σ_n and $C(f_n)$, we have $\sigma_n \preceq f_n$ and $\sigma_n \preceq C(f_n)$. Thus by (1) we have that $\min\{x \mid \Theta_k(f_n)(x) \neq \Theta_k(C(f_n))(x)\} \geq MinInd(f_n)$. Thus by second clause in the definition of M it follows that $M(\Theta_k(f_n), \Theta_k(C(f_n))) \geq MinInd(\Theta_k(f_n))$. Theorem follows. □

Theorem 4.4 can be improved in several ways. First, one can show that the mapping $C : REC \to REC$, which provides a context for each function in REC, can actually be chosen $1:1$ and onto. Furthermore, this $1:1$ and onto mapping can be constructed in such a way that not only the target functions but also the context functions can be robustly learned in parallel. In order to state this result we let **ParEx** denote the variant of **ConEx** from Definition 4.1, where the learning machine is replaced by a machine $M : Seq^2 \to \omega^2$ such that M converges on each $(f, g) \in P$ to a pair of programs (i, j) with $\varphi_i = f$ and $\varphi_j = g$. **ParEx** coincides exactly with the 2-ary parallel learning type as defined in [1]. Analogously to the other robust variants, we let **RobParEx** contain all classes $P \in S^2$ such that $\{\Theta(f), \Theta(g) \mid (f, g) \in P\} \in$ **ParEx** for all general recursive operators Θ. Thus, the following theorem provides a robust version of [1, Theorem 6].

Theorem 4.5 *There exists a class $P \subseteq REC \times REC$ such that $P|_1 = P|_2 = REC$, but $P \in$* **RobParEx**.

Proof. Let $(\varphi_{u(i)})_{i\in\omega}$, $u \in REC$, be an effective enumeration without repetitions of $\mathcal{F} = \{\sigma 0^\omega \mid \sigma \in \omega^*\}$, that is, $\mathcal{F} = \{\varphi_{u(i)} \mid i \in \omega\}$ and $(\forall i, j)[i \neq j \implies \varphi_{u(i)} \neq \varphi_{u(j)}]$. We inductively define a partial mapping C from REC into REC. In the (non-effective) construction we identify $C = \bigcup_{e\in\omega} C_e$ with its graph. $dom(C) = \{x \mid C(x)\downarrow\}$ denotes the domain, and $rg(C) = \{C(x) \mid x \in dom(C)\}$ denotes the range of a partial mapping C.

Stage 0: $C_0 = \emptyset$.

Stage $e + 1$:
If φ_e is total and $\varphi_e \notin dom(C_e) \cup rg(C_e)$ then:

Let i_e be the smallest i such that
(1) $\varphi_{u(i)} \notin dom(C_e) \cup rg(C_e) \cup \{\varphi_e\}$,
(2) $\Theta_m(\varphi_e)(x) = \Theta_m(\varphi_{u(i)})(x)$ for all $m, x \leq e$.
Let $C_{e+1} = C_e \cup \{(\varphi_e, \varphi_{u(i_e)})\}$.

Otherwise, let $C_{e+1} = C_e$.

From the definition, we immediately get the following facts:

- $dom(C) \cap rg(C) = \emptyset$,
- $dom(C) \cup rg(C) = REC$,
- C is $1:1$.

We set

$$P = \{(f, C(f)), (C(f), f) \mid f \in dom(C)\}.$$

Obviously, it holds $P|_1 = P|_2 = REC$. We want to prove $P \in$ **RobParEx**. So, let an arbitrary general recursive operator Θ_k be given. Part of the proof is based on similar ideas as previous proofs:

1. It suffices to show that $P' = \{(\Theta_k(f), \Theta_k(g)) \mid (f, g) \in P, k \leq MinInd(f), k \leq MinInd(g)\}$ is in **Ex**.
2. It suffices to infer an upper bound for the minimal index of both input functions by Fact 3.3.
3. Assume that the input functions $f = \Theta_k(\varphi_e)$ and $g = \Theta_k(C(\varphi_e))$ with $\varphi_e \in dom(C), e \geq k$, are given. If $f = g$ then both function f, g are in $\Theta_k(\mathcal{F})$. In this case we can infer a program for f and g using "learning by enumeration". If $f \neq g$ then, by condition (2), $x' = \mu x[f(x) \neq g(x)]$ is an upper bound on e, from which one can compute an upper bound on $MinInd(f)$. So, it remains to show how to find an upper bound on $MinInd(C(f))$ given an upper bound on $MinInd(f)$.

In order to prove this last point, we consider, for $e, s \in \omega$, the *computable* set $I(e, s)$ of all i such that

$$(\forall m \leq e)(\forall x \leq e)$$
$$[\Theta(\varphi_{e,s})(x)\downarrow \implies \Theta(\varphi_{e,s})(x) = \Theta_m(\varphi_{u(i)})(x)].$$

$I(e, s)$ is infinite for all e, s. Assume that $\varphi_{e'}$ is in $dom(C)$. Then, $i_{e'}$ is in $I(e', s)$ for all $s \in \omega$. Choose an s' such that $\Theta(\varphi_{e',s'})(x)\downarrow$ for all $x \leq e'$. Then, for $s \geq s'$, every $i \in I(e', s)$ satisfies condition (2). Since

$|dom(C_{e'}) \cup rg(C_{e'})| \leq 2e'$ and $(\varphi_{u(i)})_{i\in\omega}$ is an enumeration without repetitions, we get, for all $s \geq s'$,

$$|\{i \in I(e', s) \mid i < i_{e'}\}| \leq 2e'.$$

Let

$$c(e, s) = \mu i[|\{j \in I(e, s) \mid j \leq i\}| = 2e + 1].$$

This implies $i_{e'} \leq c(e', s)$ for all $s \geq s'$

Note that $c(e, s)$ is computable, and converges for all $e \in \omega$ if s tends to ∞, that is, $(\forall e)(\exists c')(\overset{\infty}{\forall} s)[c(e, s) = c']$. Thus, if we have an upper bound $y \geq e'$, then the function

$$u'(y, s) = \max\{u(i) \mid e \leq y, i \leq c(e, s)\}$$

converges to an upper bound on $MinInd(C(\varphi_{e'}))$ if s tends to ∞.

In order to formulate the learning algorithm for P' choose functions $v, w \in REC$ with

$$\begin{aligned} \Theta_k(\mathcal{F}) &= \{\varphi_{v(e)} \mid e \in \omega\}, \text{ and} \\ \varphi_{w(e)} &= \Theta_k(\varphi_e), \text{ for all } e \in \omega. \end{aligned}$$

Now, the following algorithm infers an upper bound on $MinInd(f)$ and $MinInd(g)$ for all $(f, g) \in P'$:

> Input $(f[n], g[n])$.
>
> If $f[n] = g[n]$ then output $v(\mu i[f[n] \preceq \varphi_{v(i)}])$.
>
> If $f[n] \neq g[n]$ then let $x' = \mu x[f(x) \neq g(x)]$, compute $w' = \max\{w(i) \mid i \leq x'\}$, and output $w' + u'(x', n)$.

□

Our results demonstrate that learning with a selected context is a very powerful learning notion, in particular, it renders the entire class of computable functions learnable. However, it should be noted that in this particular case, the high learning power also results from the very large function space, namely REC, from which a context can be selected. We required in Definition 4.2 that for each class $S \subseteq REC$, the context which we associate with each $f \in REC$, is also chosen from the set S. So, if one considers proper subsets $S \subset REC$, it may happen that one loses learning power, just because the space of possible contexts is reduced. Indeed, one can show that even the non-robust version **SelEx** of learning from selected contexts does not contain all subsets of REC. Due to the result from Theorem 4.4 that every class with a dense, computably enumerable subclass is in **RobSelEx**, it is actually not easy to construct such a class which is not in **SelEx**.[4]

Theorem 4.6 $(\exists S \subseteq REC)[S \notin \mathbf{SelEx}]$.

The proof of Theorem 4.6 is based on the notion of trees. We briefly recall some basics of this concept (see [21] for more details). Here, we call a mapping $T : \{0, 1\}^* \to \{0, 1\}^*$ a *tree*, if

- T is total computable,
- $(\forall \sigma, \tau)[\sigma \preceq \tau \implies T(\sigma) \subseteq T(\tau)]$ and
- $(\forall \sigma)[T(\sigma 0)$ and $T(\sigma 1)$ are incomparable$]$.

σ is a *node* of T (or *in* T) if $(\exists \tau)[\sigma \preceq T(\tau)]$. Correspondingly, a tree Q is a *subtree* of T ($Q \subseteq T$) if $(\forall \sigma)(\exists \tau)[Q(\sigma) \preceq T(\tau)]$. A total function f is a *branch* of a tree T (or, f is *on* T) if f is computable and $(\forall n)(\exists \sigma)[f[n] \preceq T(\sigma)]$. Note that $REC_{0,1}$ is "effectively isomorphic" to the set of branches of an arbitrary tree T via the mapping $f \mapsto \bigcup_{n\in\omega} T(f[n])$. In other words, every tree has "sufficiently many" branches. This implies, in particular, the following corollary:

Corollary 4.7 *If T is a tree, then*

$$\{f \mid f \text{ on } T\} \in \mathbf{Ex} \text{ iff } REC_{0,1} \in \mathbf{Ex}.$$

First, we show two lemmata in order to isolate the essential technical steps in the proof of Theorem 4.6. We write $M(f, g) \not\to h$ if M, on input (f, g), does not **Ex**-converge to h, that is,

$$(\forall e)[(\overset{\infty}{\forall} n)[M(f[n], g[n]) = e] \implies \varphi_e \neq h].$$

Lemma 4.8 *Let a tree T, a learning machine M and a computable function $g \in REC$ be given. Then there exists a subtree $Q \subseteq T$ such that*

$$(\forall f \text{ on } Q)[M(f, g) \not\to f \text{ and } M(f, f) \not\to f].$$

That is, for every f on T, neither g nor f itself provide a suitable context for f with respect to the learning machine M.

Proof. We first construct a subtree $Q \subseteq T$ which diagonalizes against the context g. For this, we distinguish two cases:

Case 1: $(\forall \sigma$ in $T)(\exists \tau$ in $T, \tau \succ \sigma)[M(\sigma, g[|\sigma|]) \neq M(\tau, g[|\tau|])]$. Then Q is defined inductively. We start with $Q(\epsilon) = \epsilon$. Assume that $Q(\sigma)$ is already defined. In order to determine $Q(\sigma 0)$ and $Q(\sigma 1)$ we search for the smallest two incomparable strings $\tau_0, \tau_1 \in T$ such that, for $i = 0, 1$,

- $Q(\sigma) \preceq \tau_i$, and
- $M(Q(\sigma), g[|Q(\sigma)|]) \neq M(\tau_i, g[|\tau_i|])$.

Note that τ_1 and τ_2 always exist by hypothesis. Now, we set $Q(\sigma i) = \tau_i$ for $i = 1, 2$. By construction, for every f on Q, M makes infinitely many mind changes on input (f, g), that is, $M(f, g) \not\to f$.

Case 2: $(\exists \sigma$ in $T)(\forall \tau$ in $T, \tau \succ \sigma)[M(\sigma, g[|\sigma|]) = M(\tau, g[|\tau|])]$. Then we choose a $\tau \succ \sigma$ such that $\varphi_{M(\sigma, g[|\sigma|])}$ is inconsistent with τ, and let Q be the subtree below τ, that is, $Q(\sigma) = T(\tau\sigma)$ for all σ. Thus, for all g on Q, on input (f, g), M converges to a program e with $\varphi_e \neq f$, that is, $M(f, g) \not\to f$.

In order to diagonalize also against the context f for all branches f of the tree, we reapply the construction on Q, but replace, this time, each term of the form $M(\eta, g[|\eta|])$ with the term $M(\eta, \eta)$. □

[4] Furthermore, this result represents a rather unusual phenomenon, since in inductive inference most learning types are closed with respect to subclasses.

Lemma 4.9 *Let a tree T and a learning machine M be given. Then there exists a branch f on T and a subtree $Q \subseteq T$ such that*

$$(\forall g \text{ on } Q)[M(f,g) \not\to f].$$

That is, no g on Q provides a suitable context for f with respect to the learning machine M.

Proof. Similarly to the proof of Lemma 4.8 we distinguish two cases.

Case 1: $(\exists f \text{ on } T)(\forall \sigma)(\exists \tau \succ \sigma)[M(f[|\sigma|]), \sigma) \neq M(f[|\tau|], \tau)]$. In this case we can construct Q similarly as in case 1 of Lemma 4.8.

Case 2: $(\forall f \text{ on } T)(\exists \sigma)(\forall \tau \succ \sigma)[M(f[|\sigma|]), \sigma) = M(f[|\tau|], \tau)]$. Let $W(f)$ be the set of all witnesses σ to f on T such that the above formula holds, that is,

$$(\forall \tau \succ \sigma)[M(f[|\sigma|]), \sigma) = M(f[|\tau|], \tau)].$$

Furthermore, we set $u(f,\sigma) = M(f[|\sigma|]), \sigma)$ for all $\sigma \in W(f)$.

Assume that, for all f on T and $\sigma \in W(f)$, we have $MinInd(f) \leq u(f,\sigma)$. Then, we can, for all f on T, infer an upper bound on $MinInd(f)$ in the limit by the following algorithm:

> *Let $\sigma = \epsilon$ and $h = M(f[0], \sigma)$. On input $f[n]$ check whether there exists a $\tau \in T \cap \{0,1\}^n$ such that $\sigma \preceq \tau$ and $h \neq M(f[n], \tau)$. If so, update $\sigma = \tau$ and $h = M(f[n], \tau)$. Output h.*

Thus, by Fact 3.3 it follows $\{f \mid f \text{ on } T\} \in \mathbf{Ex}$, and thus, $REC_{0,1} \in \mathbf{Ex}$ by Corollary 4.7. We have a contradiction.

Hence, there exists an f on T and a $\sigma' \in W(f)$ such that $u(f,\sigma') < MinInd(f)$. Now, this f witnesses our claim together with the subtree Q of T below σ', that is, the subtree Q with $Q(\sigma) = T(\sigma'\sigma)$ for all σ. □

Proof of Theorem 4.6. We define a sequence of trees $T_0 \supseteq T_1 \supseteq T_2 \ldots$ and a sequence of functions f_i on T_i, $i \in \omega$, using Lemmata 4.8 and 4.9. Let $M_0, M_1, \ldots$ be an enumeration of all learning machines.

Stage 0*:* $T_0(\sigma) = \sigma$ for all σ.

Stage $s+1$*:* We define f_s and T_{s+1}.

1. By applying Lemma 4.8 s times on the functions $f_0, \ldots, f_{s-1}$ determine a tree $Q_s \subseteq T_s$ such that
   $(\forall i < s)(\forall f \text{ on } Q_s)$
   $[M_s(f,f_i) \not\to f \text{ and } M_s(f,f) \not\to f].$
2. By applying Lemma 4.9 determine a function f_s on Q_s and a tree $T_{s+1} \subseteq T_s$ such that
   $$(\forall g \text{ on } T_{s+1})[M_s(f_s, g) \not\to f_s]$$

We claim $S = \{f_i \mid i \in \omega\} \notin \mathbf{SelEx}$. Assume by way of contradiction that $S \in \mathbf{SelEx}$ as witnessed by M_s. Thus, there is a function $f_i \in S$ such that M_s on input (f_s, f_i) converges to a program for f_s. However, if $i \leq s$ then $M(f_s, f_i) \not\to f_s$ by Lemma 4.8, since f_s is on Q_s. Otherwise, if $i > s$ then $M(f_s, f_i) \not\to f_s$ by Lemma 4.9, since f_i is on $T_i \subseteq T_{s+1}$. Contradiction. □

5 Measuring The Functional Dependence

In Section 4 we have shown that there are very hard learning problems which become learnable when a suitably selected context is supplied to the learner. In this section we analyze the possible functional dependence between the target functions and the contexts in such examples. In particular, we attempt to find examples which are only learnable with a context, but so that the functional dependence between the target function and the context is manageable. The functional dependence is measured from a computational theoretic point of view, that is, we are looking for examples in $\mathbf{RobSelEx} - \mathbf{Ex}$ and $\mathbf{SelEx} - \mathbf{Ex}$, such that the problem of implementing a suitable context mapping has low Turing complexity. We will consider two types of implementations for context mappings: *operators*, which work on (the values of) the target functions, and *program mappings*, which work on programs for the target functions.

First we consider context mappings $C\colon S \to S$ which are implementable by operators. Here, one can show that, if the functional dependence between the target function and the context is too manageable then the multitask problem will not have the desired property, that is, the learnability with the help of a selected context will imply the learnability of the target functions without any context.

This can easily be seen if one assumes that a context mapping $S \to S$ is implemented by a general recursive operator C. Then, let s be a computable function from *Seq* to *Seq* such that, for all total functions f, $s(f[n]) \subseteq C(f)$, and $\bigcup_{n\in\omega} s(f[n]) = C(f)$. Note that such an s exists by condition $(*)$ mentioned in Section 2, and the discussion around it. This implies that the machine defined by $N(f[n]) = M(f[\,|s(f[n])|\,], s(f[n]))$ $\mathbf{Ex}$-infers every $f \in S$.

In the case of robust learning, this observation can even be surprisingly generalized to the result that for all classes $S \subseteq REC$, which are closed under finite variants and robustly learnable from selected contexts, the existence of a *general continuous operator* implementing a context mapping is enough to guarantee the robust learnability of S itself!

An operator Ψ, not necessarily computable, is *continuous* (by definition) iff it is compact, that is, $(\forall f)(\forall x)(\exists \sigma \preceq f)[\Psi(\sigma)(x) = \Psi(f)(x)]$, and monotone, that is, $(\forall \sigma,\tau)(\forall x)[\sigma \preceq \tau \wedge \Psi(\sigma)(x)\downarrow \implies \Psi(\tau)(x)\downarrow = \Psi(\sigma)(x)]$. As noted in Section 1 above, the *general continuous operators* are the continuous operators which map all total functions into total functions.

Theorem 5.1 *If $S \subseteq REC$ is closed under finite variants and* $\mathbf{RobSelEx}$*-learnable as witnessed by a (not necessarily computable) general continuous operator C: $S \to S$, then $S \in \mathbf{RobEx}$.*

Proof. If $C(f) = f$ for all $f \in S$, then $S \in \mathbf{RobEx}$ is obvious. So, assume that there exists a function $f' \in S$

and an x with $C(f')(x) \neq f'(x)$. Choose a $\sigma \preceq f'$ such that $(\forall y \leq x)[C(f')(y) = C(\sigma)(y)]$. Since C is continuous, it follows for all total functions f,

$$\sigma \prec f \implies \sigma \not\prec C(f).$$

Let $\tau_0, \tau_1, \ldots$ be an effective enumeration of $\omega^{|\sigma|+1}$. We choose a general recursive operator Γ with

$$\Gamma(f) = \begin{cases} \tau_a f(|\sigma|+1) f(|\sigma|+2) \ldots & \text{if } \sigma a \prec f, \\ 0^\omega & \text{if } \sigma \not\prec f, \end{cases}$$

for all total functions f. Note that

$$S = \{\Gamma(f) \mid f \in S, \sigma \prec f\},$$

since S is closed under finite variants.

Now, let an arbitrary general recursive operator Θ be given. Then, $\Psi = \Theta \circ \Gamma$ is also general recursive. Thus, there exists a machine M which **ConEx**-learns $\{(\Psi(f), \Psi(C(f))) \mid f \in S\}$. In particular, M **ConEx**-infers the set

$$\begin{aligned} & \{(\Psi(f), \Psi(C(f))) \mid f \in S, \sigma \prec f\} \\ = \; & \{(\Theta(\Gamma(f)), \Theta(\Gamma(C(f))) \mid f \in S, \sigma \prec f\} \\ = \; & \{(\Theta(f), \Theta(0^\omega)) \mid f \in S\}. \end{aligned}$$

It follows that $\Theta(S)$ is **Ex**-identifiable, and hence, $S \in$ **RobEx**. $\square$

Since each operator, which is general recursive relative to some oracle A, is already general continuous, we can thus not hope to find examples in **RobSelEx** $-$ **Ex** such that the context mapping can be implemented by a general A-recursive operator, no matter how complex is the oracle A!

However, for the *non*-robust version such examples exist for all suitably non-trivial oracles A, i.e., for all A such that $\mathbf{Ex[A]} - \mathbf{Ex} \neq \emptyset$. Such A's exist in abundance by [10, 18].

Theorem 5.2 *Let $S \in \mathbf{Ex[A]} - \mathbf{Ex}$ such that S contains all almost constant functions. Then S is* **SelEx**-*learnable as witnessed by some general A-recursive operator.*

Proof. Let $S \in \mathbf{Ex[A]} - \mathbf{Ex}$ as witnessed by the oracle learning machine M^A. Without loss of generality, we can assume that M^A is total. We define a general A-computable operator C by

$$C(f)(n) = M^A(f[n]) \text{ for all total functions } f.$$

Thus, for all $f \in S$, there exists an e with

$$\varphi_e = f \text{ and } (\overset{\infty}{\forall} n)[C(f)(n) = e].$$

Since $C(f)$ is almost constant, $C(f)$ is in S. Consider the learning machine N with $N(\epsilon, \epsilon) = 0$ and $N(\sigma, \tau) = \tau(n)$ for $\sigma, \tau \in \omega^{n+1}$. It follows immediately that N **ConEx**-learns the set $\{(f, C(f)) \mid f \in S\}$. Hence, S is in **SelEx** via the context mapping C. $\square$

We now turn our attention to context mappings which are implementable by program mappings. One can show that the context mapping $C\colon REC \to REC$ constructed in Theorem 4.4 in Section 4 above is computable relative to K', that is, there exists a partial K'-computable function $h\colon \omega \to \omega$ with

$$(\forall e)[\varphi_e \in REC \implies [h(e)\downarrow \;\wedge\; \varphi_{h(e)} = C(\varphi_e)]].$$

Thus, K' provides an upper bound on the Turing degree of context mappings for classes in **RobSelEx** $-$ **Ex**. However, if one wants to reduce this upper bound, the problem arises that these program mappings are generally not invariant with respect to different indices of the same function [21].[5] And transforming an arbitrary program mapping into an invariant (or extensional) one generally requires an oracle of degree K'. It is convenient, then, in the sequel, to use the following *equivalent* definition for **SelEx** and **RobSelEx** instead of Definition 4.2:

Definition 5.3 $S \subseteq REC$ is in **SelEx** if there exists a class $S' \subseteq S \times S$ in **ConEx** and a partial *program mapping* $h : \omega \to \omega$ such that, for every $\varphi_e \in S$, $h(e)\downarrow$ and $(\varphi_e, \varphi_{h(e)}) \in S'$. If, furthermore, S' can be chosen from **RobConEx**, then we say that S is in **RobSelEx**.

Recall that there are no classes $S \in$ **RobSelEx** $-$ **Ex** such that the corresponding context mapping is implementable by any, even noncomputable, general continuous operator. In contrast, the following interesting theorem shows that the class $REC_{0,1}$ is in **RobSelEx** as witnessed by a program mapping h which is *computable, i.e., which requires no oracle to compute.*

Theorem 5.4 *$REC_{0,1}$ is* **RobSelEx**-*learnable as witnessed by a computable program mapping.*

Proof. We define a computable program mapping $h \in REC$ by

$$\varphi_{h(e)}(x) = \begin{cases} \varphi_e(x) & \text{if } x < e, \\ \max\{0, 1 - \varphi_e(e)\} & \text{if } x = e \text{ and } \varphi_e(e)\downarrow, \\ \uparrow & \text{if } x = e \text{ and } \varphi_e(e)\uparrow, \\ 0 & \text{if } x > e. \end{cases}$$

Let $\mathcal{F} = \{\sigma 0^\omega \mid \sigma \in \{0,1\}^*\}$. Note that for all $\varphi_e \in REC_{0,1}$ we have

$$\varphi_{h(e)} \in \mathcal{F} \text{ and } \varphi_e[e] \preceq \varphi_{h(e)}.$$

We want to prove that h is a program mapping witnessing $REC_{0,1} \in$ **RobSelEx** according to Definition 5.3. So, let an arbitrary general recursive operator Θ be given. We will exploit the well known fact that, for every n, one can effectively compute a number $l(n)$ such that $\Theta(\sigma)(x)\downarrow$ for all $x \leq n$ and all $\sigma \in \{0,1\}^{l(n)}$. This can be seen, for example, by considering the *computable* binary tree[6]

$$T = \{\sigma \mid (\exists x \leq n)[\Theta(\sigma)(x)\uparrow]\}.$$

[5] They are not extensional in the terminology of [25].

[6] For convenience, here, we use a way to define trees which is formally different from that in Theorem 4.6, but does not change the essential nature of this concept [21]: A subset $T \subseteq \{0,1\}^*$ is a tree if it is closed under initial segments.

If T is infinite, then, by Königs Lemma, T contains an infinite branch $f \in \{0,1\}^\omega$. But this implies $\Theta(f)(x)\uparrow$ for some $x < n$, which contradicts the fact that Θ is general. Thus, actually, T is finite and $l(n)$ can be computed by $l(n) = \mu m[T \cap \{0,1\}^m = \emptyset]$.

We will now define a learning machine M, which infers, on input $(\Theta(\varphi_e), \Theta(\varphi_{h(e)}))$ with $\varphi_e \in REC_{0,1}$, an upper bound for $MinInd(\Theta(\varphi_e))$ in the limit. This implies our claim by Fact 3.3.

We choose functions $u, v \in REC$ such that

$$\begin{aligned} \Theta(\mathcal{F}) &= \{\varphi_{u(e)} \mid e \in \omega\}, \text{ and} \\ \varphi_{v(e)} &= \Theta(\varphi_e), \text{ for all } e \in \omega. \end{aligned}$$

The learning machine M works as follows:

Input $(f[n], g[n])$.

If $f[n] = g[n]$ then output $u(\mu i[f[n] \preceq \varphi_{u(i)}])$.

If $f[n] \neq g[n]$ then let $x' = \mu x[f(x) \neq g(x)]$ and output $\max\{v(i) \mid i \leq l(x')\}$.

Let an arbitrary function $\varphi_e \in REC_{0,1}$ be given and consider the input functions $f = \Theta(\varphi_e)$ and $g = \Theta(\varphi_{h(e)})$. Clearly, if $f = g$ then $f \in \Theta(\mathcal{F})$. In this case M will, in fact, infer a program e' for f by line 2. Otherwise, M converges to $e' = \max\{v(i) \mid i \leq l(x')\}$ where $x' = \mu x[f(x) \neq g(x)]$. Recall that $\Theta(\sigma)(x)\downarrow$ for all $x \leq x'$ and $\sigma \in \{0,1\}^{l(x')}$. Assume $l(x') < e$. Then, we get

$$f(x') = \Theta(\varphi_e[e])(x')\downarrow = \Theta(\varphi_{h(e)}[e])(x')\downarrow = g(x'),$$

which is is a contradiction. Thus, $e \leq l(x')$ holds. This implies $e' \geq v(e) \geq MinInd(f)$. □

On the other hand, one can also show that there is no upper bound on program mappings implementing context mappings for classes in **RobSelEx** − **Ex**:

Theorem 5.5 *For all oracles A, there is a class $S \in$ **RobSelEx** − **Ex** such that for every partial program mapping $h : \omega \to \omega$ witnessing $S \in$ **RobSelEx** it holds that A is Turing reducible to h.*

Proof. Let an arbitrary oracle A be given. We construct strings $\sigma_0, \sigma_1, \ldots$ and $\eta_0, \eta_1, \ldots$, as well as computable functions $f_0, f_1, \ldots$ satisfying the following conditions for all $n \in \omega$:

(1) $f_n \succeq \sigma_n$ such that M_m on input (f_n, f_k) does not converge to (a program for) f_n for all $k, m \leq n$,

{so, if $M_m(f_n, f_k)$ converges to f_n, this implies $k > n$ and $m > n$}

(2) $\sigma_n \preceq \eta_n \preceq f_n$ such that $\Theta_k(\eta_n)(x)\downarrow$ for all $x \leq MinInd(f_n), k \leq n$,

(3) $\sigma_{n+1} = \eta_n a A(0) \ldots A(n) \tau_0 \ldots \tau_n$ such that $a \neq f_n(|\eta_n|)$ and the strings $\tau_0, \ldots, \tau_n$ establish, if possible, $\Theta_i(f_n) \neq \Theta_i(f_{n+1})$, for $i = 0, \ldots, n$, as follows:

Let $\rho_i = \eta_n a A(0) \ldots A(n) \tau_0 \ldots \tau_{i-1}$.

If $(\forall g \succ \rho_i)[\Theta_i(g) = \Theta_i(f_n)]$ then let $\tau_i = \epsilon$.

If $(\exists \tau, x)[\Theta_i(\rho_i \tau)(x)\downarrow \neq \Theta_i(f_n)\downarrow]$ then let $\tau_i = \tau$.

Now we set $S = \{f_n \mid n \in \omega\}$. Clearly, S is not in **Ex** by condition (1).

To see that $S \in$ **RobSelEx** let an arbitrary computable operator Θ_k be given such that, without loss of generality, $\Theta_k(S)$ is is infinite. Since $\Theta_k(S)$ is infinite it follows from condition (3) that

$$(\forall n \geq k)(\forall m > n)[\Theta_k(f_n) \neq \Theta_k(f_m)].$$

Again it suffices to prove that $A = \{\Theta_k(f_n) \mid n \geq k\}$ is in **SelEx**. For this we let

$$A' = \{(\Theta_k(f_n), \Theta_k(f_{n+1})) \mid n \geq k\}.$$

Now, $A' \in$ **ConEx** can be shown similarly as in previous proofs due to condition (2).

Finally, let us assume that the partial program mapping $h : \omega \to \omega$ witnesses $S \in$ **RobSelEx**. So, for all e with $\varphi_e = f_n$ it follows that $h(e)$ is defined. We let $g_n = h(MinInd(f_n))$ for all n. Since S is in **RobSelEx** via h, it holds, in particular, that some machine M_m witnesses $S \in$ **SelEx** via h. For $n \geq m$, we get $g_n = f_k$ for some $k > n$, since $g_n \in S$ and $g_n \notin \{f_0, \ldots, f_n\}$ by condition (1). We inductively define a sequence of indices e_r according to:

$$\begin{aligned} e_0 &= MinInd(f_m), \\ e_{n+1} &= h(e_n). \end{aligned}$$

This implies that e_n is an index of some function f_k with $k \geq m+n$. Now, the following algorithm decides A relative to h:

Input: x.

Compute e_x, e_{x+1}.

Compute the first y such that $\varphi_{e_x}(y) \neq \varphi_{e_{x+1}}(y)$.

Output $A(x) = \varphi_{e_{x+1}}(x + y + 1)$.

□

6 Conclusion

In the present work we investigated a number of models for learning from context in the inductive inference framework, namely, learning from arbitrary context, learning from selected context, and parallel learning. Our positive results showed that for each of these models and their variants, there exist *unlearnable* classes of functions that become learnable by additionally providing to the learner a suitable context, that is, another function from the class. More importantly, all these existence results hold *robustly*, which clearly strengthens their inherent claim. In the process of establishing our results on learning from arbitrary context, we generalized a theorem due to Kinber, Smith, Velauthapillai, and Wiehagen. Another result on parallel learning is a generalization of a theorem due to Angluin, Gasarch, and Smith.

One of the most unexpected findings in the paper could be summed up as follows. The class REC of all computable functions is robustly learnable from selected context. However, somewhat surprisingly, we are able to construct a subclass of *REC* which is not learnable according to even the ordinary notion of learning from selected context.

Finally, we also analyzed the functional dependence between learning tasks and helpful selected contexts. We showed that in general even arbitrary (that is, not necessarily computable) continuous operators are too weak to describe such a dependence. The situation, however, changes if one considers context mappings implementable by program mappings. Here, in some cases, the context mapping can even be implemented by a *computable* program mapping. However, on the other hand, we also showed that in general there does not exist an upper bound on the Turing degree which a program mapping may need to provide useful selected contexts for all tasks in a particular class.

The ordinary (that is nonrobust) variants of our existence theorems can be established using self-referential coding tricks. As discussed in the introduction, the notion of robustness was initially proposed to avoid such coding tricks. However, as shown in [14], robustness, in general, can only avoid "purely numerical" coding tricks, and still allows "topological" self-referential coding to go through. The original proofs of the nonrobust variants of Theorem 3.4 and Theorem 4.5 in [15] and [1], respectively, actually used purely numerical coding tricks. Hence, these proofs do not work in the robust framework. However, a careful analysis of our robustness proofs reveals that numerical coding has been replaced by topological coding. Hence, in addition to the results in [14] our proofs can be seen to provide further evidence of self-referential coding tricks that are able to get around Fulk's notion of robust learning.

Hence, a natural question is if it is possible to invent a learning notion that "avoids all forms of self-referential coding" tricks. A little thought suggests that this may be a bit too much to ask for because if there is to be some kind of learnability at all, then there is certainly some kind of coding. So the more interesting question is perhaps the formulation of a learning model that avoids intuitive/reasonable forms of self-referential coding tricks. Is there some hierarchy of more and more sophisticated coding tricks? And if so, does such a hierarchy interact in any way with the many learnability hierarchies known in inductive inference? We feel answers to these questions will improve our understanding of learnability.

References

[1] D. Angluin, W. I. Gasarch, and C. H. Smith. Training sequences. *Theoretical Computer Science*, 66(3):25–272, 1989.

[2] K. Bartlmae, S. Gutjahr, and G. Nakhaeizadeh. Incorporating prior knowledge about financial markets through neural multitask learning. In *Proceedings of the Fifth International Conference on Neural Networks in the Capital Markets*, 1997.

[3] J. Barzdin. Two theorems on the limiting synthesis of functions. *In Theory of Algorithms and Programs, Latvian State University, Riga*, 210:82–88, 1974.

[4] J. Baxter. A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, 28:7–39, 1997.

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

[6] R. A. Caruana. Multitask connectionist learning. In *Proceedings of the 1993 Connectionist Models Summer School*, pages 372–379, 1993.

[7] R. A. Caruana. Algorithms and applications for multitask learning. In *Proceedings 13th International Conference on Machine Learning*, pages 87–95. Morgan Kaufmann, 1996.

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

[9] T. G. Dietterich, H. Hild, and G. Bakiri. A comparison of ID3 and backpropogation for English text-to-speech mapping. *Machine Learning*, 18(1):51–80, 1995.

[10] L. Fortnow, W. Gasarch, S. Jain, E. Kinber, M. Kummer, S. Kurtz, M. Pleszkoch, T. Slaman, R. Solovay, and F. Stephan. Extremes in the degrees of inferability. *Annals of Pure and Applied Logic*, 66:21–276, 1994.

[11] R. V. Freivalds and R. Wiehagen. Inductive inference with additional information. *Elektronische Informationsverarbeitung und Kybernetik*, 15:179–185, 1979.

[12] M. Fulk. Robust separations in inductive inference. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 405–410, St. Louis, Missouri, 1990.

[13] S. Jain and A. Sharma. Learning with the knowledge of an upper bound on program size. *Information and Computation*, 102(1):118–166, Jan. 1993.

[14] S. Jain, C. H. Smith, and R. Wiehagen. On the power of learning robustly. In *Proceedings of Eleventh Annual Conference on Computational Learning Theory*. ACM Press, New York, NY, 1998.

[15] E. Kinber, C. H. Smith, M. Velauthapillai, and R. Wiehagen. On learning multiple concepts in parallel. In *Proceedings of Sixth Annual Conference on Computational Learning Theory*, pages 175–181, Santa Cruz, CA, USA, 1993. ACM Press.

[16] E. Kinber and R. Wiehagen. Parallel learning - a recursion-theoretic approach. Informatik-Preprint 10, Fachbereich Informatik, Humbold-Universität, 1991.

[17] M. Kummer and F. Stephan. Inclusion problems in parallel learning and games. *Journal of Computer and System Sciences (Special Issue COLT'94)*, 52(3):403–420, 1996.

[18] M. Kummer and F. Stephan. On the structure of degrees of inferability. *Journal of Computer and System Sciences*, 52(2):214–238, Apr. 1996.

[19] S. Matwin and M. Kubat. The role of context in

concept learning. In M. Kubat and G. Widmer, editors, *Proceedings of the ICML-96 Pre-Conference Workshop on Learning in Context-Sensitive Domains, Bari, Italy*, pages 1–5, 1996.

[20] T. Mitchell, R. Caruana, D. Freitag, J. McDermott, and D. Zabowski. Experience with a learning, personal assistant. *Communications of the ACM*, 37:80–91, 1994.

[21] P. Odifreddi. *Classical Recursion Theory*. North-Holland, Amsterdam, 1989.

[22] D. Osherson, M. Stob, and S. Weinstein. *Systems that Learn*. MIT Press, Cambridge, Massachusetts, 1986.

[23] L. Pratt, J. Mostow, and C. Kamm. Direct transfer of learned information among neural networks. In *Proceedings of the 9th National Conference on Artificial Intelligence (AAAI-91)*, 1991.

[24] H. Rogers. Gödel numberings of partial recursive functions. *Journal of Symbolic Logic*, 23:331–341, 1958.

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

[26] J. Royer. *A Connotational Theory of Program Structure*. Lecture Notes in Computer Science 273. Springer-Verlag, 1987.

[27] T. J. Sejnowski and C. Rosenberg. NETtalk: A parallel network that learns to read aloud. Technical Report JHU-EECS-86-01, Johns Hopkins University, 1986.