

Chapter 2

Inductive Inference Systems for Learning Classes of Algorithmically Generated Sets and Structures

Valentina S. Harizanov

Abstract. Computability theorists have extensively studied sets whose elements can be enumerated by Turing machines. These sets, also called *computably enumerable* sets, can be identified with their Gödel codes. Although each Turing machine has a unique Gödel code, different Turing machines can enumerate the same set. Thus, knowing a computably enumerable set means knowing one of its infinitely many Gödel codes. In the approach to learning theory stemming from E.M. Gold's seminal paper [9], an inductive inference learner for a computably enumerable set A is a system or a device, usually algorithmic, which when successively (one by one) fed data for A outputs a sequence of Gödel codes (one by one) that at certain point stabilize at codes correct for A . The convergence is called semantic or *behaviorally correct*, unless the same code for A is eventually output, in which case it is also called syntactic or *explanatory*. There are classes of sets that are semantically inferable, but not syntactically inferable.

Here, we are also concerned with generalizing inductive inference from sets, which are collections of distinct elements that are mutually independent, to mathematical structures in which various elements may be interrelated. This study was recently initiated by F. Stephan and Yu.

Ventsov. For example, they systematically investigated inductive inference of the ideals of computable rings. With F. Stephan we continued this line of research by studying inductive inference of computably enumerable vector subspaces and related structures.

In particular, we showed how different convergence criteria interact with different ways of supplying data to the learner. Positive data for a set A are its elements, while negative data for A are the elements of its complement. Inference from *text* means that only positive data are supplied to the learner. Moreover, in the limit, all positive data are given. Inference from *switching* means that changes from positive to negative data or *vice versa* are allowed, but if there are only finitely many such changes, then in the limit all data of the eventually requested type (either positive or negative) are supplied. Inference from an *informant* means that positive and negative data are supplied alternately, but in the limit all data are supplied. For sets, inference from switching is more restrictive than inference from an informant, but more powerful than inference from text. On the other hand, for example, the class of computably enumerable vector spaces over an infinite field syntactically inferable from text does not change if we allow semantic convergence, or inference by switching, but not both at the same time. Many classes of inferable algebraic structures have nice algebraic characterizations, at least when learning from text or from switching is considered. On the other hand, we do not know of such characterizations for learning from an informant.

2.1 Computably Enumerable Languages

In theoretical computer science, inductive inference introduced by Gold [9] in 1967 is a framework for learning theory. Learning is viewed as a dialogue between a learner and a teacher. The learner is trying to learn a family of computably enumerable sets of natural numbers. A set of natural numbers A to be learned can be viewed as coding a language L by identifying in an algorithmic manner the grammatically correct sentences of L with the elements of the set A .

A set is *computably enumerable* if there is an algorithm, a computable function, which generates it by enumerating (listing) its elements. Computably enumerable sets are often abbreviated by c.e. They are also called recursively enumerable and abbreviated by r.e. In other words, a language is c.e. if there is an algorithmic grammar that generates all correct sentences. These algorithmic grammars and the languages they generate are called *unrestricted* in Chomsky's hierarchy, and are further classified according to restrictiveness of the grammars as regular, context-free, and context-sensitive.

Regular languages have the most restrictive grammars and are context-free. However, there are context-free languages that are not regular. Similarly,

context-free languages are context-sensitive, and not all context-sensitive languages are context-free. There are unrestricted languages that are not context-sensitive. These classes of languages also have machine equivalents. While regular languages coincide with languages recognized by finite automata, context-free languages coincide with those recognized by push-down automata. Context-sensitive languages are the same as languages recognized by linear-bounded Turing machines. An unrestricted language can be characterized as the domain of a partial computable function. That is, for such a language there is a Turing machine that on any input x halts (converges) if x is a correct sentence, and computes forever (diverges) if x is an incorrect sentence.

All c.e. sets can be simultaneously algorithmically listed by systematically coding all Turing machines. Let

$$P_0, P_1, P_2, \dots, P_e, \dots$$

be a fixed algorithmic (computable) enumeration of all Turing machines. For a Turing machine P_e , let φ_e be the unary partial function it computes. Then

$$\varphi_0, \varphi_1, \varphi_2, \dots, \varphi_e, \dots$$

is an algorithmic enumeration of all unary partial computable functions. It can be shown that a set is c.e. if and only if there is a partial computable function whose domain is that set. In other words, the set of all outputs of an algorithm is the set of all inputs on which another algorithm halts.

The domain of a partial computable function φ_e is the c.e. set denoted by W_e . We call the index e its Gödel code. Thus,

$$W_0, W_1, W_2, \dots, W_e, \dots$$

is an algorithmic enumeration of all c.e. sets. We can also think of this enumeration as an enumeration of all Chomsky's unrestricted languages. Every partial computable function and every c.e. set have infinitely many Gödel codes, because for every Turing machine there are infinitely many distinct Turing machines that have the same effect, that is, compute the same function. Hence knowing a c.e. language is knowing one of its infinitely many Gödel codes. Similarly, every Chomsky's language is generated by infinitely many grammars. We say that such grammars are *equivalent*.

In the next subsection, we introduce an important subclass of c.e. languages: decidable, or computable, languages. In Section 2.2, we discuss learning classes of c.e. sets using positive and negative information, and also different convergence criteria. In Section 2.3, we generalize learning to classes of c.e. algebraic structures. (We can think of sets as being special algebraic structures with the identity relations only.) Many subsections contain detailed proofs with explanations of the relevant computability theoretic and algebraic facts.

2.1.1 Computable and Noncomputable Sets

Some c.e. sets are *computable*, also called *recursive* or *decidable*. More precisely, a set is computable if and only if both the set and its complement are c.e. From

these two computable (algorithmic) enumerations, we can devise an algorithm (decision procedure) for recognizing the elements of the set and, equivalently, the nonelements of the set. That is, a set is computable exactly when it has a computable characteristic function. The *characteristic function* of a set A is the function that outputs 1 on the elements of the set A , and outputs 0 on the elements of the complement of A . Every context-sensitive language is computable.

Clearly, the complement of a computable set is computable. On the other hand, there are c.e. sets whose complements are not c.e., namely noncomputable c.e. sets. For example, the *halting set* K is c.e., but not computable. The set K consists of all inputs e on which the Turing machine with Gödel code e halts. Equivalently, K consists of those e that appear in W_e :

$$K = \{e : P_e(e) \text{ halts}\} = \{e : \varphi_e(e) \downarrow\} = \{e : e \in W_e\}.$$

The set K is c.e. because it is enumerated by the procedure that simultaneously runs

$$P_0(0), P_1(1), \dots, P_e(e), \dots,$$

and enumerates those e for which $P_e(e)$ converges, as soon as the convergence occurs. Here, simultaneously means that, say, at each step we add (activate) a new Turing machine and also run all activated machines for an additional computational step.

The complement of K , the *divergence set* \overline{K} , is not c.e. If \overline{K} were c.e., then for some e_0 ,

$$\overline{K} = W_{e_0} = \{e : e \notin W_e\}.$$

Hence, for this particular e_0 , we have

$$e_0 \in \overline{K} \Leftrightarrow e_0 \in W_{e_0} \Leftrightarrow e_0 \notin W_{e_0},$$

which is a contradiction.

Computable sets can also be viewed as c.e. sets whose elements are algorithmically enumerated in an increasing order. Finite initial segments of such an enumeration give additional negative information that certain “small” elements that have not been enumerated so far will never be enumerated into the set.

A sequence of computable sets A_0, A_1, A_2, \dots is *uniformly computable* iff there is a binary computable function f such that for every i , the unary function g defined by $g(x) = f(x, i)$ is the characteristic function of A_i . We also write $A_i(x) = f(x, i)$. For more on computability theory, see Odifreddi [19] and Soare [23].

2.2 Inductive Inference of Computably Enumerable Languages: Identification in the Limit

Although we are mainly interested in algorithmic learners, which can be thought of as Turing machines or computable functions, a learner can also be general. A

general learner can be thought of as a function (not necessarily algorithmic) that maps finite sequences of numbers (coding sentences in a language) to numbers (Gödel codes for c.e. languages). A learner receives an infinite stream of data x_0, x_1, x_2, \dots of a c.e. set A to be learned, and outputs a sequence of hypothesized codes

$$e_0, e_1, e_2, \dots, e_n, \dots,$$

which, in case the learner successfully learns A , converges to a ‘description’ of the set A . In addition to Gödel codes, the learner is also allowed to output a special non-numerical symbol, say ‘?’.

2.2.1 Learning from Text

One way to feed data for a c.e. set (language) to the learner is via a text. A *text* t for a set A is any infinite sequence of its elements

$$t = a_0, a_1, a_2, \dots,$$

possibly with repetitions, such that $A = \{a_0, a_1, a_2, \dots\}$. The set A is the *content* of t . Hence every member (positive datum) of A appears in t at least once, but no nonmember of A (negative datum) is in t . If A is finite, then some element must appear infinitely often in t . One-element sets have only one text, while any set with at least two elements has uncountably many texts. A learner M *converges on* t if after every finite sequence of data a_0, \dots, a_n the learner outputs a code e_n , in symbols $M(a_0, \dots, a_n) = e_n$, such that eventually (i.e., after finitely many steps) the outputs e_n code the correct language. This means that for some step n , we have

$$A = W_{e_n} = W_{e_{n+1}} = W_{e_{n+2}} = \dots$$

Although the codes $e_n, e_{n+1}, e_{n+2}, \dots$ all generate the same set, they do not have to be the same—in fact, they can all be distinct.

A learner learns A *from text* if the learner converges on any text for A . A learner learns a partial computable function φ if it learns its graph $\{(x, \varphi(x)) : \varphi(x) \text{ halts}\}$. Partial computable functions have c.e. graphs. A learner learns a class \mathcal{L} of languages if it learns every language A in \mathcal{L} . Clearly, if the learner learns a class \mathcal{L} , then it also learns every subclass $\mathcal{L}_0 \subseteq \mathcal{L}$. In general case, we do not care what happens with languages that are not in the class to be learned. On the other hand, we call a learner for \mathcal{L} *confident* when it always converges to some hypothesis, even when given a text for a language that does not belong to the class \mathcal{L} . However, the learner does not have to converge accurately on the languages that are not in \mathcal{L} . Not every learnable class of c.e. languages can be learned by a confident learner. A learner for a class \mathcal{L} is called *class-comprising* if the learner always guesses a language in \mathcal{L} .

A class consisting of a single language is always learnable since there is a learning algorithm that always outputs the same index for the single set in the class. An example of an infinite learnable class is the class of all finite sets of

natural numbers. It is learnable since the learner can guess that the language consists exactly of the elements that are seen up to that point. Since every language in the class is finite, the learner will be correct in the limit. However, it can be shown that this learner cannot be confident (see [21]). The class of all sets missing exactly one element is also learnable. The learner guesses that the missing element is the least number not seen so far. Again, such learning strategy will be correct in the limit.

For a sequence σ , the range of σ , $rng(\sigma)$, is the set of all elements of σ . We use the symbol $\hat{}$ for concatenation of sequences. A learner M from text is *consistent* (Angluin [1]) if for every sequence σ of data given to the learner, the learner guesses a c.e. set with code $M(\sigma)$, which contains $rng(\sigma)$, that is, $W_{M(\sigma)} \supseteq rng(\sigma)$. A learner M from text is *conservative* (Angluin [1]) if whenever $M(\sigma \hat{\tau}) \neq M(\sigma)$, then it must be that $ran(\sigma \hat{\tau}) \not\subseteq W_{M(\sigma)}$. Thus, a conservative learner makes only justified changes of its hypotheses. While conservatism is not restrictive for general learning, it does restrict algorithmic learning.

We can also allow blanks (pause symbols) in the texts—these are pauses in the presentation of data (see [3]). More precisely, we fix a new symbol, \sharp , and allow any number of occurrences of this symbol in a text. This does not change learnability properties, while allowing the teacher not to give any data at certain steps. Furthermore, we can also assign a text to the empty language:

$\sharp, \sharp, \sharp, \sharp, \dots$

However, we agree that for a sequence σ that might contain the symbol \sharp , the set $rng(\sigma)$ does not contain \sharp .

If we have a class of computable sets, then we may also want to consider learning characteristic functions for these sets. That is, instead of guessing codes for enumerating algorithms, the learner tries to guess a correct index for the characteristic function of the set to be learned.

2.2.2 Different Convergence Criteria

Gold [9] introduced a strong convergence criterion for identification in the limit. Assume that the learner's sequence of hypotheses for a c.e. set A is

$$e_0, e_1, e_2, \dots, e_n, e_{n+1}, e_{n+2}, \dots$$

The strong convergence requires that after finitely many steps, the hypotheses are the *same* and correct:

$$e_0, e_1, e_2, \dots, e, e, e, \dots, \text{ and}$$

$$A = W_e.$$

This convergence is also called *syntactic* convergence. The corresponding learning is often called *intensional* or *explanatory* learning, and is abbreviated by *EX*.

A weaker notion of convergence, introduced by Case and Lynes [4] and independently by Osherson and Weinstein [20], allows the hypotheses to be *distinct*,

although they must be correct after finitely many steps. That is, there is some n such that

$$A = W_{e_n} = W_{e_{n+1}} = W_{e_{n+2}} = \dots$$

This convergence is also called *semantic* convergence. The corresponding learning is often called *extensional* or *behaviorally correct* learning, and is abbreviated by *BC*.

It can be shown that the class of all computable functions is *EX*-learnable from text by a general learner. On a given finite subset of the graph of a computable function to be learned, the learner guesses that it is the c.e. set with the least index, which contains this finite set. On the other hand, Gold [9] proved that the class of all computable functions is not *EX*-learnable from text by an algorithmic learner. (See the introduction to the volume.)

The identification in the limit definition does not require that the learner signals or confirms convergence. However, it yields the following result for a general *EX*-learner, due to L. Blum and M. Blum [3].

Proposition 2.1 (*L. Blum and M. Blum*) *If an EX-learner can learn a c.e. set A from text, then there is a finite sequence σ of elements of A , called a locking sequence for A , onto which the learner ‘locks’ its conjecture for A . That is, if the learner outputs e after seeing σ , then $A = W_e$, and after seeing any sequence of elements from A extending σ , the learner outputs e again.*

For example, Proposition 2.1 implies that the class \mathcal{L} of all finite sets enlarged by adding the set \mathbb{N} of all natural numbers is not *EX*-learnable. The reason is that no learner will be able to distinguish between the set \mathbb{N} and the finite set given by the locking sequence for \mathbb{N} , which also belongs to the class \mathcal{L} . A similar example of a class that is not *EX*-learnable is the collection consisting of an infinite c.e. set together with all of its finite subsets. Hence, as established by Gold [9], for example, the class of all regular languages, or the class of all computable languages is not *EX*-learnable.

The following result in [1] gives an important characterization of *EX*-learnability from text for a general learner.

Theorem 2.2 (*Angluin*) *Let \mathcal{L} be a class of c.e. sets. Then \mathcal{L} is EX-learnable from text if and only if for every A in \mathcal{L} , there is a finite set $D \subseteq A$ such that for no U in \mathcal{L} we can have*

$$D \subseteq U \subset A.$$

Proof. Assume that the class \mathcal{L} is *EX*-learnable and that A is in \mathcal{L} . Then the corresponding set D will be the set of all elements of the locking sequence σ for A . Now if U is the language in \mathcal{L} containing D and contained in A , then for any text for U extending σ , the learner will guess A , so $U = A$ (thus, U cannot be properly contained in A).

Conversely, assume that for every A in \mathcal{L} , there is a finite set $D \subseteq A$ such that for no U in \mathcal{L} we can have $D \subseteq U \subset A$. Choose such D_A for $A \in \mathcal{L}$. Then,

after receiving a finite subsequence σ of the text, the learner outputs the least index of a language A in \mathcal{L} such that D_A is contained in $\text{rng}(\sigma)$, and $\text{rng}(\sigma)$ is contained in A : $D_A \subseteq \text{rng}(\sigma) \subseteq A$. If such A does not exist, the learner outputs an arbitrary fixed index. Note that the learner is not necessarily an algorithmic one.

We will now show that this learner identifies \mathcal{L} . Let i be the least index of a language A in \mathcal{L} , whose text t is fed to the learner. There is a step n by which enough of t is given to the learner—a sequence σ so that $\text{rng}(\sigma)$ contains D_A , but $\text{rng}(\sigma)$ is not contained in any language W_j in \mathcal{L} among W_0, \dots, W_{i-1} such that $A \not\subseteq W_j$. We claim that the learner correctly conjectures W_i at every step after n . The required condition for W_i is satisfied for a corresponding τ :

$$D_A \subseteq \text{rng}(\sigma) \subseteq \text{rng}(\sigma \hat{\ } \tau) \subseteq A.$$

Furthermore, the learner will not conjecture any W_j in \mathcal{L} among W_0, \dots, W_{i-1} , since otherwise, for some σ' extending σ , we have $D_{W_j} \subseteq \text{rng}(\sigma') \subset W_j$. Hence $\text{rng}(\sigma)$ is contained in W_j , so $A \subset W_j$. Therefore,

$$D_{W_j} \subseteq \text{rng}(\sigma') \subseteq A \subset W_j.$$

However, $A \in \mathcal{L}$ and this contradicts the choice of D_{W_j} . ■

2.2.3 *EX-Learnability is More Restrictive than BC-Learnability*

Clearly, *EX*-learnability implies *BC*-learnability. It can be shown, by finding examples, that algorithmic *EX*-learnability is more restrictive than algorithmic *BC*-learnability. We will now present one of first such examples for learning from text (see [21]).

Example 2.3 *Recall that K is the halting set. Let \mathcal{L} be the following class of sets:*

$$\mathcal{L} = \{K \cup D : D \text{ is a finite set of natural numbers}\}.$$

The class \mathcal{L} is algorithmically BC-learnable from text, but not algorithmically EX-learnable from text.

Proof. Let us show that \mathcal{L} is not algorithmically *EX*-learnable. To obtain a contradiction, assume that \mathcal{L} is *EX*-learnable via some algorithmic learner M . Fix a locking sequence σ for K . We will use this locking sequence to show that the complement \bar{K} is c.e. Since K is c.e., we can fix its algorithmic enumeration

$$K = \{k_0, k_1, k_2, \dots\}.$$

Given an input n , if $n \in K$, then $K = K \cup \{n\}$, otherwise, $K \neq K \cup \{n\}$, but the learner still correctly learns $K \cup \{n\}$ because it belongs to \mathcal{L} . More precisely, n

is enumerated in \overline{K} if and only if there is a sufficiently long sequence k_0, \dots, k_m such that the learner M converges on

$$\sigma \hat{\wedge} n \hat{\wedge} k_0 \hat{\wedge} \dots \hat{\wedge} k_m,$$

to an index different from M 's hypothesis on σ . This is a contradiction, since \overline{K} is not c.e.

On the other hand, the class \mathcal{L} is algorithmically BC -learnable from text because, by the s - m - n Theorem of computability theory (Theorem 3.5 in [23]), there is an algorithm which for every finite sequence a_0, \dots, a_n outputs a Gödel code for the c.e. set $K \cup \{a_0, \dots, a_n\}$. The outcome of this algorithm depends on the code for K and on the sequence a_0, \dots, a_n . For some n_0 , the sequence a_0, \dots, a_{n_0} will include a complete text for the finite set D . However, we cannot algorithmically find such an n_0 . ■

BC -learnability is much more powerful than EX -learnability, even when learning with mistakes, or anomalies, is allowed, as showed by Bardzin and independently by Case, Smith, and Harrington (see [5]). Case, Smith, and Harrington established that a specific class of algorithmically BC -learnable computable functions is not algorithmically EX^* -learnable, where EX^* -learnability allows the learner to eventually guess a function finitely different from the one whose data (i.e., elements of the graph) are being fed to the learner.

2.2.4 Positive versus Negative Information. Learning from Text versus Learning from an Informant

So far, we considered only learning from text, that is, when the learner requests only positive data (elements of the set to be learned), and the teacher eventually provides all of them. Learning from text will be abbreviated by *Txt*.

Learning *from an informant* is when the learner alternately requests positive and negative data (negative data are elements of the complement of a set, or incorrect sentences of a language), and the teacher eventually provides every element (with type label 1) of the set to be learned, and every element of its complement (with type label 0). Again, blanks (pauses) are also allowed in the presentation of data. Learning from an informant will be abbreviated by *Inf*.

It can be shown, by finding examples, that learning from text is more restrictive than learning from an informant. For example, the collection \mathcal{L} consisting of \mathbb{N} together with all of its finite subsets can be learned from an informant, but not from text. We already saw that \mathcal{L} is not EX -learnable from text. This class \mathcal{L} is EX -learnable from an informant because the learner, in addition to positive information, also obtains complete negative information in the limit. Thus, the learner guesses that the set to be learned is \mathbb{N} , until the learner sees a nonelement. After that, the learner guesses that the set to be learned is the finite set consisting of the elements given so far. Moreover, Gold [9] showed that the class of all context-sensitive languages is algorithmically learnable from an informant.

The class of all c.e. sets is learnable by a general learner from an informant. On any finite sequence of positive and negative data, the learner's hypothesis is the least Gödel code of a set that is consistent with the given sequence. On the other hand, Gold [9] proved that the class of all c.e. sets is not algorithmically learnable from an informant. Hence general learning from an informant is more powerful than algorithmic learning from an informant.

In [22], Sharma showed that combining learning from an informant with a restrictive convergence requirement that the first numeric (different from ‘?’) hypothesis is already the correct one implies learnability from text. The convergence criterion where the learner is allowed to make only one conjecture, which has to be correct, is known as *finite identification* (see [9]). Since finite identification can be viewed as a model for batch learning, Sharma's result shows that batch learnability from both positive and negative data coincides with incremental learnability (i.e., identification in the limit) from positive data only. For more on learning theory of c.e. languages see Case and Smith [5], Osherson, Stob and Weinstein [21], and Jain, Osherson, Royer and Sharma [13].

2.2.5 Learning from Switching Type of Data

Motoki [17], and later Baliga, Case and Jain [2] considered different ways of supplying finite sets of negative data. For example, there might be a finite set of negative data $F \subseteq \bar{A}$ such that the learner succeeds in learning the set A from F , in addition to a text for A . However, there is algorithmic learner who learns all c.e. sets in this sense. Thus, the following framework that Baliga, Case and Jain introduced in [2] seems more interesting. There is a finite set $F \subseteq \bar{A}$ such that the learner always succeeds in learning the set A from a text for A , plus a text for a set C containing F and disjoint from A , that is, satisfying $F \subseteq C \subseteq \bar{A}$. In any case, Baliga, Case, and Jain established that these finite amounts of negative information result in strong increase of learning power, and also lead to increased speed of learning.

In [12], Jain and Stephan treated positive and negative data symmetrically and introduced several ways of learning from all positive and some negative data, or from all negative and some positive data. We will focus on the following framework in [12] and call the corresponding learning criterion, abbreviated by *Sw*, learning *from switching* (type of data). The learner is allowed to request positive or negative data for A , but when the learner after finitely many switches always requests data of the same type, the teacher eventually gives all elements in A (if the type is positive), or all elements in its complement \bar{A} (if the type is negative). If the learner switches his requests infinitely often, then it still has to learn the language accurately.

One motivation for learning from switching comes from computability theory, precisely, from the *n-c.e. sets* for $n \geq 1$ (see p. 58 in [23]). Recall that for a set A , we write $A(x) = 1$ if $x \in A$, and $A(x) = 0$ if $x \notin A$. For an *n-c.e.* set A , the function $A(x)$ is *limit-computable*, $A(x) = \lim_{s \rightarrow \infty} f(x, s)$ for a binary computable function $f(x, s)$, and, assuming that $f(x, 0) = 0$, the limit must be

achieved after at most n changes. Hence 1-c.e. sets are exactly the c.e. sets, and 2-c.e. sets are the set-theoretic differences of two c.e. sets. n -c.e. sets further generalize to α -c.e. sets for arbitrary computable ordinals α .

The following example, due to Jain and Stephan [12], shows that switching type of information provides more learning power than giving positive information only. A set of natural numbers is co-finite if its complement is finite.

Example 2.4 *Let \mathcal{L} be the class of all finite and all co-finite sets.*

- (i) *The class \mathcal{L} is not EX-learnable from text.*
- (ii) *The class \mathcal{L} is EX-learnable from switching.*

Proof. (i) The class \mathcal{L} is not EX-learnable from text since the subclass of \mathcal{L} consisting of all finite sets plus the (trivially co-finite) set of all natural numbers is not EX-learnable from text.

(ii) The following learner can learn \mathcal{L} from switching. If the number of positive data (elements) exceeds the number of negative data (nonelements) given to the learner, then the learner requests a negative datum. Moreover, he conjectures the co-finite set excluding exactly the nonelements given so far. In the other case, the learner requests a positive datum, and conjectures the finite set consisting of all elements given so far. The learner is correct in the limit. ■

Jain and Stephan [12] also showed that learning from switching is weaker than learning from an informant. The following result from [11] gives a general sufficient condition for non-*SwBC*-learnability.

Theorem 2.5 (*Harizanov and Stephan*) *Let \mathcal{L} be a class of c.e. sets. Assume that there is some set A in \mathcal{L} such that for every finite set D , there are U, U' in \mathcal{L} with:*

$$U \subset A \subset U',$$

(*U approximates A from below, and U' approximates A from above*),

$$D \cap U = D \cap U'$$

(*U and U' , and hence A , coincide on D*). *Then the class \mathcal{L} cannot be BC-learned from switching, even by a general learner.*

Proof. Let M be a given general *SwBC*-learner. We will show that M cannot learn \mathcal{L} . More precisely, we will show that there is a way of presenting data, according to the *Sw*-protocol, for a language in \mathcal{L} on which M will be confused and will not be able to *BC*-infer the correct language. We will now describe such a sequence of data given to M .

(i) If the current hypothesis of M is a correct index for A , and there is a finite sequence $\vec{x} = x_1, x_2, \dots, x_k$ of data of some length k , corresponding to the sequence of requests y_1, y_2, \dots, y_k of M ($y_1, y_2, \dots, y_k \in \{0, 1\}$), such that after concatenating the sequence \vec{x} to the current sequence of data presented to M , the hypothesis of M will become an incorrect index for A , then the first

datum x_1 from one of the shortest such sequences will be given to M . (After k steps for the least k , a whole such sequence will be given to M .)

(ii) If the current hypothesis of M is an incorrect index for A , and y is M 's current data type request, then M is given the least x that has not yet appeared in the data sequence and such that $A(x) = y$, where $A(x)$ is the characteristic function of A .

(iii) In the remaining case, we have that all future hypotheses of M , when given appropriate data (positive or negative) consistent with A , result in hypotheses for A . We consider the following two subcases.

(a) If the pair U, U' has not been chosen yet, it will be chosen at this step as follows. Let D be the set of positive data given so far to the learner M . Choose U, U' so that:

$$U \subset A \subset U' \text{ and } D \cap U = D \cap A = D \cap U'.$$

The learner will receive the pause symbol \sharp .

(b) If the pair U, U' has been chosen, take the least x that has not yet appeared in the data sequence given to M , and which satisfies $x \in U$ in the case when the learner M requests a positive datum, and $x \notin U'$ in the case when M requests a negative datum. If such x does not exist (when $U = \emptyset$ or $U' = \omega$), then M is given the pause symbol \sharp .

For the proof, we first assume that M infinitely often conjectures a hypothesis incorrect for A . Then case (ii) applies infinitely often and M is given either all elements of A , or all nonelements of A . Hence M fails to BC -learn A from switching.

Otherwise, we assume that the learner M eventually ends up in case (iii), and at a certain step, U and U' are chosen so that $U \subset A \subset U'$. If infinitely often M requests positive data, then M is given all elements of U and some nonelements of U' (and hence nonelements of U). If infinitely often M requests negative data, then M is given all nonelements of U' and some elements of U (and hence elements of U'). In the first case, M is expected to learn U , and in the second case, M is expected to learn U' . However, in both cases, M almost always conjectures an index for the set A . Hence M does not learn \mathcal{L} from switching. ■

2.3 Learning Classes of Algebraic Structures

Recently, Stephan and Ventsov [24] extended learning theory from sets to algebraic structures. They investigated learnability from text for classes of c.e. substructures of certain computable algebraic structures. In particular, they considered c.e. submonoids and c.e. subgroups of a computable group, and c.e. ideals of a computable commutative ring. Several of such learnable classes have nice algebraic characterizations. Stephan and Ventsov also studied ordinal

bounds on learner's mind changes in learning classes of algebraic substructures from text. In addition, Stephan and Ventsov showed that learnability of algebraic structures can greatly depend on the semantic knowledge given at the synthesis of the learner. Since we deal with structures, semantic knowledge may consist of programs that compute structural operations. Harizanov and Stephan [11] further investigated learnability of classes of c.e. vector space. Again, some learnable classes coincide with natural algebraic classes. For these classes, in addition to learning from text, Harizanov and Stephan also considered learning with negative data.

2.3.1 Algebraic Structures

An *algebraic structure* is a nonempty set, called the domain, together with some operations and relations satisfying certain axioms. We often use the same symbol for a structure and its domain. A *semigroup* $(G, *)$ is a structure with the domain G and an associative binary operation $*$:

$$(x * y) * z = x * (y * z) \text{ for all } x, y, z \in G.$$

A semigroup is *abelian* or *commutative* if its binary operation is commutative. A *monoid* is a semigroup $(G, *)$ that contains an *identity* element e :

$$x * e = e * x = x \text{ for all } x \in G.$$

It can be shown that the identity element is unique. An example of a monoid is $(\{0, 1, 2, \dots\}, +)$. A *group* is a monoid $(G, *)$ such that for every $x \in G$, there exists an *inverse* element $x^{-1} \in G$:

$$x * x^{-1} = x^{-1} * x = e.$$

An example of a group is the additive structure of integers $(Z, +)$. Here, $e = 0$ and $x^{-1} = -x$. The structure $(\{\dots, -2, 0, 2, 4, \dots\}, +)$ is a subgroup of $(Z, +)$. Another important example of a group is the multiplicative structure of nonzero rationals $(Q \setminus \{0\}, \cdot)$. For this group, $e = 1$ and $x^{-1} = \frac{1}{x}$.

A *ring* is a structure $(R, +, \cdot)$ with two binary operations, usually called addition and multiplication, such that $(R, +)$ is a commutative group, the multiplication is associative, and

$$x \cdot (y + z) = x \cdot y + x \cdot z \text{ and } (y + z) \cdot x = y \cdot x + z \cdot x.$$

It can be shown that

$$x \cdot 0 = 0 \cdot x = 0.$$

If for a ring $(R, +, \cdot)$ the multiplication \cdot is commutative, then the ring is said to be commutative. If there is an element $1 \in R$ such that for all $x \in R$,

$$x \cdot 1 = 1 \cdot x = x,$$

then $(R, +, \cdot)$ is a ring with identity 1. If $(R, +, \cdot)$ is a nontrivial (containing at least two elements) commutative ring with identity and, in addition, every nonzero element has an inverse with respect to multiplication, then $(R, +, \cdot)$ is a *field*. For example, the ring of integers $(Z, +, \cdot)$ is not a field, while the ring of rationals $(Q, +, \cdot)$ is a field. There are also finite fields (also called Galois fields). For a prime number p , the ring $(Z_p, +_p, \cdot_p)$ of residue classes modulo p is a field.

Let

$$RJ =_{def} \{r \cdot j : r \in R \wedge j \in J\}.$$

A subring $(J, +, \cdot)$ of $(R, +, \cdot)$ is an *ideal* if and only if

$$RJ \subseteq J \wedge JR \subseteq J.$$

For example, $\{\dots, -2, 0, 2, 4, \dots\}$ is an ideal of $(Z, +, \cdot)$. Two obvious ideals of $(R, +, \cdot)$ are the unit ideal—the ring itself, and the trivial ideal with the domain $\{0\}$. A set $D \subseteq J$ *generates* the ideal J , in symbols $J = I(D)$, if J is the smallest ideal (with respect to the set-theoretic inclusion) that contains D . For more on rings and ideals, see [15].

A *vector space* over a field F is an additive commutative group $(V, +)$, together with scalar multiplications \cdot_γ for every element $\gamma \in F$, which satisfies additional axioms for vector addition $+$ and scalar multiplication. The scalar product of $\gamma \in F$ and a vector $x \in V$ is usually denoted by γx . The axioms are:

$$\begin{aligned}\gamma(x + y) &= \gamma x + \gamma y, \\ (\gamma + \delta)x &= \gamma x + \delta x, \\ \gamma(\delta x) &= (\gamma\delta)x, \\ 1x &= x,\end{aligned}$$

where $x, y \in V$ and $\gamma, \delta, 1 \in F$.

A *basis* for a vector space consists of a maximal set of linearly independent vectors. All bases have the same size, called the *dimension* of the space. An example of an important infinite dimensional vector space is the vector space Q^∞ over the field Q of rationals. We can think of the elements of Q^∞ , the vectors, as infinite sequences of rational numbers with only finitely many nonzero components. We have pointwise vector addition and pointwise scalar multiplication, e.g.,

$$(6, 3, -4, 0, 0, \dots) + (-1, 4, 0, 0, 0, \dots) = (5, 7, -4, 0, 0, \dots),$$

and

$$6(1, \frac{1}{2}, -\frac{2}{3}, 0, 0, \dots) = (6, 3, -4, 0, 0, \dots).$$

For example, the three vectors

$$(1, \frac{1}{2}, -\frac{2}{3}, 0, 0, \dots), (-\frac{1}{2}, 2, 0, 0, 0, \dots), (5, 7, -4, 0, 0, \dots)$$

are linearly dependent since

$$6(1, \frac{1}{2}, -\frac{2}{3}, 0, 0, \dots) + 2(-\frac{1}{2}, 2, 0, 0, 0, \dots) = (5, 7, -4, 0, 0, \dots).$$

The following vectors

$$(1, 0, 0, 0, \dots), (0, 1, 0, 0, \dots), (0, 0, 1, 0, \dots), \dots$$

are linearly independent. They form a *standard basis* for Q^∞ .

2.3.2 Computable Structures and Their Computably Enumerable Substructures

A countable structure is *computable* if its domain is a computable subset of natural numbers, and its relations and operations are uniformly computable. For example, a group $(G, *)$ is computable if its domain G is computable and the operation $*$ is computable. It then follows that the unary function $^{-1}$ assigning to each element g its inverse g^{-1} is also computable. Clearly, every finite group is computable. Examples of computable infinite groups include the additive group of integers $(\mathbb{Z}, +)$, the additive group of rationals $(\mathbb{Q}, +)$, and the multiplicative group of rationals $(\mathbb{Q} \setminus \{0\}, \cdot)$, under the standard representations, that is, coding of elements and operations by the natural numbers. A ring $(R, +, \cdot)$ is computable if the domain R is a computable set and the binary operations $+, \cdot$ are computable. An example of a computable ring is the ring of integers $(\mathbb{Z}, +, \cdot)$ under the standard representation.

Metakides and Nerode [16] showed that the study of algorithmic vector spaces can be reduced to the study of V_∞ , an infinite dimensional vector space over a computable field F , consisting of all finitely nonzero infinite sequences of elements of F , under pointwise operations. Clearly, these operations can be performed algorithmically. Every element in F can be identified with its Gödel code, which is a natural number. Unless we explicitly state otherwise, we will assume that F is infinite. In that case, we can even assume, without loss of generality, that F is the field of rationals $(\mathbb{Q}, +, \cdot)$ and identify V_∞ with \mathbb{Q}^∞ , a ‘big’ computable infinite dimensional vector space whose scalars are the rationals. A *dependence algorithm* for a vector space decides whether any finite set of its vectors is linearly dependent. Since the standard basis for V_∞ is computable, V_∞ has a dependence algorithm.

Roughly speaking, a c.e. vector space over a computable field is one where the set of vectors is c.e., operations of vector addition and scalar multiplication are partial computable, and the vector equality is a c.e. relation. Metakides and Nerode [16] showed that any c.e. vector space is isomorphic to the *quotient space* $\frac{V_\infty}{V}$, where V is a c.e. subspace of V_∞ . In $\frac{V_\infty}{V}$, the equality of vectors is *modulo* V . That is, two vectors u and w are equal if their difference $u - w$ is 0 *modulo* V , that is, belongs to V . Thus, the class of all c.e. subspaces (substructures) of $\frac{V_\infty}{V}$ can be viewed as the class $\mathcal{L}(V)$ of all c.e. subspaces of V_∞ that contain V , the superspaces of V . Let V_0, V_1, V_2, \dots be an *algorithmic list* of all c.e. subspaces of V_∞ indexed by their Gödel codes. For example, we can assume that V_e is generated by the c.e. set W_e of independent vectors in V_∞ , where W_0, W_1, W_2, \dots is a fixed *algorithmic list* of all c.e. subsets of independent vectors. This is possible since V_∞ has a computable basis, which can be identified with the set of natural numbers.

2.3.3 Learning Classes of Ring Ideals from Text

A commutative ring with identity $(R, +, \cdot)$ is *Noetherian* if it does not have an infinite strictly ascending chain of ideals. Equivalently, $(R, +, \cdot)$ is Noetherian

if and only if every ideal of $(R, +, \cdot)$ is *finitely generated* (see [15]). Hence every ideal of a computable commutative Noetherian ring is c.e. It can be shown that it is even computable [25]. An example of a computable commutative Noetherian ring with identity is the ring $Q[x_1, \dots, x_n]$ of polynomials with rational coefficients and variables x_1, \dots, x_n . On the other hand, the ring $Q[x_1, x_2, x_3, \dots]$ of rational polynomials in infinitely many variables is not Noetherian.

The following result from [24] algebraically characterizes *BC*-learnability from text of ideals of a computable commutative ring.

Theorem 2.6 (*Stephan and Ventsov*) *Let $(R, +, \cdot)$ be a computable commutative ring with identity. Let \mathcal{L} be the set of all ideals of R . Then the following two statements are equivalent.*

- (i) *The ring $(R, +, \cdot)$ is Noetherian.*
- (ii) *The family \mathcal{L} (of c.e. ideals) is algorithmically *BC*-learnable from text.*

Proof. (i) \Rightarrow (ii) For a given finite set D of positive data, the learner hypothesizes a code for the ideal generated by D . There is a stage by which the learner has seen all elements of a finite set that generates the ideal to be learned. Hence, from that stage on the learner correctly guesses a code for the ideal to be learned.

(ii) \Rightarrow (i) Assume that M is a *BC*-learner for \mathcal{L} . We will show that every ideal of $(R, +, \cdot)$ is finitely generated. Let I be an ideal in \mathcal{L} . Fix a locking sequence σ for I . Consider the ideal J generated by $\text{ran}(\sigma)$. For any sequence τ of elements in J , after seeing the sequence $\sigma \hat{\ } \tau$ of positive data, the learner M guesses an index for I . Since $J \in \mathcal{L}$ and M learns \mathcal{L} , it follows that $I = J$. Hence I is generated by the finite set $\text{ran}(\sigma)$. ■

Stephan and Ventsov [24] constructed a ring, which allowed them to establish the following negative result about *EX*-learnability.

Theorem 2.7 (*Stephan and Ventsov*) *There is a computable commutative Noetherian ring with identity whose class of all ideals is not *EX*-learnable from text.*

On the other hand, for certain computable commutative Noetherian rings with identity, the classes of their ideals are *EX*-learnable from text. An example of such a ring is the polynomial ring $Q[x_1, \dots, x_n]$ under the standard representation. This example generalizes to the following result (see [24]).

Theorem 2.8 (*Stephan and Ventsov*) *Let $(R, +, \cdot)$ be a computable commutative Noetherian ring with identity. Let \mathcal{L} be the set of all ideals of R . Then the following two statements are equivalent.*

- (i) *There is a uniformly computable sequence of all ideals in \mathcal{L} .*
- (ii) *The set \mathcal{L} is *EX*-learnable from text by a self-comprising, consistent, and conservative algorithmic learner.*

The following result from [24] gives a computability theoretic characterization of those computable Noetherian rings whose ideals are *EX*-learnable. This characterization is in terms of dominating time for enumeration of elements in the (finitely generated) ideals.

Theorem 2.9 (*Stephan and Ventsov*) *Let $(R, +, \cdot)$ be a computable commutative Noetherian ring with identity, and let \mathcal{L} be the set of all ideals of $(R, +, \cdot)$. Then \mathcal{L} is EX-learnable from text if and only if there is a computable function f with the property that for every $I(D) \in \mathcal{L}$, where D is finite, every $x \in I(D)$ can be algorithmically enumerated in $I(D)$ within $f(x)$ computation steps.*

A commutative ring is *Artinian* if it does not have an infinite strictly descending chain of ideals. Every commutative Artinian ring with identity is Noetherian. Hence every commutative Artinian ring with identity has a finite length n , where n is the maximum number such that there is a chain of $(n + 1)$ ideals.

Theorem 2.10 (*Stephan and Ventsov*) *Let $(R, +, \cdot)$ be a computable commutative ring with identity, and let \mathcal{L} be the set of all ideals of $(R, +, \cdot)$. Then \mathcal{L} is learnable from text with a constant bound on the number of mind changes if and only if the ring $(R, +, \cdot)$ is Artinian. The constant bound is exactly the length of the ring.*

Since every ideal of a computable commutative Noetherian ring is computable, Stephan and Ventsov also investigated learning Gödel codes for characteristic functions of these ideals (see [24]).

Theorem 2.11 (*Stephan and Ventsov*) *Let $(R, +, \cdot)$ be a computable commutative Noetherian ring with identity. Let \mathcal{L} be the set of all ideals of R .*

(i) *If the class \mathcal{L} is EX-learnable from text, then the characteristic function indices of ideals in \mathcal{L} can be EX-learned by a confident learner.*

(ii) *If the characteristic function indices of ideals in \mathcal{L} are BC-learnable from text, then these characteristic function indices can be also EX-learned by a confident learner.*

2.3.4 Characterizing Text-Learnable Classes of Vector Spaces

We will show that the classes of c.e. vector subspaces of the form $\mathcal{L}(V)$ syntactically inferable from text have a nice algebraic characterization. Moreover, this characterization does not change if we allow either semantic convergence or inference by switching.

Theorem 2.12 (*Harizanov and Stephan*) *Assume that the dimension of $\bigvee_{\mathbb{N}} V_{\infty}$ is finite. Then the class $\mathcal{L}(V)$ is EX-learnable from text by an algorithmic learner.*

Proof. The learner guesses that the space to be learned is generated by $V \cup \{x_0, \dots, x_{n-1}\}$, where x_0, \dots, x_{n-1} is the sequence of positive data given to the learner so far. Hence this guess will change whenever the learner receives a new element. However, since the dimension of $\frac{V_\infty}{V}$ is finite, every space A in $\mathcal{L}(V)$ is generated by $V \cup D$ for some finite set D of elements independent over V . If that A is the space whose text is given to the learner, then for some m , the set $\{x_0, \dots, x_{m-1}\}$ will include all of D (and possibly some elements from V). Thus, the learner will correctly learn A in the limit. Moreover, it has to learn A syntactically.

Now, we have to show that this can be done in an algorithmic manner. Notice that, since the dimension of $\frac{V_\infty}{V}$ is finite, the space V is computable. In addition, there is an algorithm which checks for every finite set D of vectors in V_∞ and every vector v , whether v is in the linear closure of $V \cup D$ (i.e., in the space generated by $V \cup D$). Such an algorithm exists, since V_∞ has a computable basis consisting of vectors from a (computable) basis for V plus finitely many additional vectors.

Let i be a fixed index for a c.e. set of independent vectors that generates V . Assume that at some stage the learner is given x_0, \dots, x_{n-1} . The learner first algorithmically checks whether x_0 belongs to V . If it does, the learner omits it from the sequence, and next checks whether x_1 belongs to V . If x_0 does not belong to V , then the learner keeps x_0 in the sequence, and algorithmically checks whether x_1 belongs to the linear closure of $V \cup \{x_0\}$. If it does, the learner omits it from the sequence, and next checks whether x_2 belongs to the linear closure of $V \cup \{x_0\}$. If x_1 does not belong to the linear closure of $V \cup \{x_0\}$, then the learner keeps x_0, x_1 in the sequence, and algorithmically checks whether x_2 belongs to the linear closure of $V \cup \{x_0, x_1\}$. After n rounds, the learner ends up with a subsequence x'_0, \dots, x'_{k-1} of elements ($k \leq n$), each linearly independent from the previous ones together with V . Now the learner executes an algorithm that on input $(i, x'_0, \dots, x'_{k-1})$ outputs a code e for the space V_e generated by $V \cup \{x_0, \dots, x_{n-1}\}$. Moreover, the learner changes its hypothesis only when the space generated by $V \cup \{x_0, \dots, x_n\}$ properly contains the previous space $V \cup \{x_0, \dots, x_{n-1}\}$, where for $n = 0$ the previous space is V . Since the sequence x'_0, \dots, x'_{k-1} eventually stabilizes, the learner will keep outputting the same code e and will *EX*-learn A . Thus, the number of mind changes that the learner makes is bounded by the dimension of $\frac{V_\infty}{V}$. ■

Theorem 2.13 (*Harizanov and Stephan*) *Assume that the class $\mathcal{L}(V)$ is BC-learnable from text by an algorithmic learner. Then the dimension of $\frac{V_\infty}{V}$ is finite.*

Proof. Let v_0, v_1, \dots be a computable enumeration of V_∞ . We define U_n for $n \geq 0$ to be the vector space generated by $V \cup \{v_0, v_1, \dots, v_n\}$. Since $U_{n+1} = U_n \cup \{v_{n+1}\}$, the dimension of U_{n+1} is either the same as for U_n or increases by 1. Clearly, V_∞ is the ascending union of all spaces U_n :

$$\begin{aligned} V_\infty &= U_0 \cup U_1 \cup U_2 \cup \dots, \text{ and} \\ U_0 &\subseteq U_1 \subseteq U_2 \subseteq \dots \end{aligned}$$

If follows from a learning theoretic result that such a class $\mathcal{L}(V)$ can be *BC*-learned from text if there are only finitely many distinct sets in this ascending chain. Hence, there is m such that

$$U_m = U_{m+1} = U_{m+2} = \dots$$

It follows that

$$V_\infty = U_0 \cup \dots \cup U_m = U_m.$$

Hence the dimension of $\frac{V_\infty}{V}$ is at most $m + 1$, and thus finite. ■

Theorem 2.14 (*Harizanov and Stephan*) *Assume that the class $\mathcal{L}(V)$ is *EX*-learnable from switching by an algorithmic learner. Then the dimension of $\frac{V_\infty}{V}$ is finite.*

Proof. We will assume that $\mathcal{L}(V)$ is *EX*-learnable from switching and that $V \neq V_\infty$. We will first prove the following claim.

Claim. *If $\mathcal{L}(V)$ is *EX*-learnable from switching, then V is computable.*

Assuming the Claim, we will now prove the theorem. To obtain a contradiction, we suppose that $\frac{V_\infty}{V}$ is infinite dimensional. Then, since V is computable, we can find a computable basis $\{u_0, u_1, \dots\}$ of a vector space U such that $U \cap V = \{0\}$. Recall that K denotes the halting set. Let W be the linear closure of

$$V \cup \{u_n : n \in K\}.$$

The space W is not computable, so it follows by Claim that $\mathcal{L}(W)$ is not *EX*-learnable from switching. Since W contains V , the class $\mathcal{L}(V)$ is contained in the class $\mathcal{L}(W)$. Hence, $\mathcal{L}(V)$ is also not *EX*-learnable from switching, which is a contradiction.

To prove Claim, assume that M is an algorithmic *SwEX*-learner for $\mathcal{L}(V)$. Let \prec be a fixed computable ordering of elements of V_∞ .

(i) If the current hypothesis of M is old (i.e., the index that M guesses is the same as the previous one) and there is a finite sequence $\vec{x} = x_1, x_2, \dots, x_k$ of data consistent with V , corresponding to requests y_1, y_2, \dots, y_k of M , such that M changes its hypothesis after seeing \vec{x} , then M is given the least (with respect to \prec) datum x_1 from the first shortest such sequence.

(ii) If the current hypothesis of M is new (i.e., the index that M guesses is different from the previous one) then, after M requests a datum of type y ($y \in \{0, 1\}$), M is given the least x that has not yet been given to the learner and $V(x) = y$.

Clearly, the learning process either goes finitely or infinitely many times both through case (i) and case (ii). First assume that the sequence (i)–(ii) is executed infinitely many times. Hence, it follows that the learner M has made infinitely many different hypotheses. However, the learner has either given all

elements of V (if M requested all type 1 data after some step), or all elements of \overline{V} (if M requested all type 0 data after some step). Thus, the learner is given information about V according to the *Sw*-protocol without converging syntactically. This contradicts the assumption of the Claim. Thus, both (i) and (ii) are executed finitely many times.

Since (i) is executed only finitely many times, there is a stage in the learning process without M 's further mind change, provided M is given data consistent with V (that is, elements of V upon requests of type 1, and elements of \overline{V} upon requests of type 0). We can even assume that M 's hypothesis from some point on is a fixed index for V , since otherwise, M would not learn V when given information about V according to the *Sw*-protocol. Now, with this assumption, we further consider two cases.

(a) In every possible situation when the learner M requests a datum of type 1, it is given an element of V such that at some later stage M requests a datum of type 0. This allows us to take some proper c.e. superspace W of V (since $V \neq V_\infty$), and at every request of M for a datum of type 0, give M the least element x in \overline{W} , which M had not seen so far. Then M does not infer the space W , although M is given data for W .

(b) In the remaining possible case, M is fed finitely many data consistent with V such that M never later requests any datum of type 0. Consider the stage after which M requests only data of type 1. Let D be the set of all data of type 0 given to M up to that stage. We can now algorithmically enumerate \overline{V} as follows: $x \in \overline{V}$ iff either M changes its mind while being fed (positive) data from the linear closure of $V \cup \{x\}$, or M requests a datum of type 0, or some element of D is enumerated into the linear closure of $V \cup \{x\}$. Hence \overline{V} is c.e., and thus V is computable. ■

We can now summarize the previous results as follows.

Corollary 2.15 *The following statements are equivalent for a c.e. subspace V of V_∞ .*

- (i) *The dimension of $\frac{V_\infty}{V}$ is finite.*
- (ii) *The class $\mathcal{L}(V)$ is EX-learnable from text by an algorithmic learner.*
- (iii) *The class $\mathcal{L}(V)$ is BC-learnable from text by an algorithmic learner.*
- (iv) *The class $\mathcal{L}(V)$ is EX-learnable from switching by an algorithmic learner.*

Proof. (i) \Rightarrow (ii), (iii) \Rightarrow (i) and (iv) \Rightarrow (i) These follow from the above results.

(ii) \Rightarrow ((iii) & (iv)) These follow directly from the definitions, since EX-learning from text is stronger than BC-learning from text, as well as than EX-learning from switching. ■

Next, we will investigate the learnability of the class $\mathcal{L}(V)$ when the dimension of $\frac{V_\infty}{V}$ is infinite.

2.3.5 Characterizing *SwBC*-Learnable Classes of Vector Spaces

Assume that V be a c.e. subspace of V_∞ such that the dimension of $\frac{V_\infty}{V}$ is infinite. Let k be a natural number, possibly 0. In [14], Kalantari and Retzlaff introduced the following notion of a k -thin space.

Definition 2.16 *A space $V \in \mathcal{L}(V_\infty)$ with infinite dimension of $\frac{V_\infty}{V}$ is called k -thin, $k \geq 0$, if for every c.e. subspace W of V_∞ such that $W \supseteq V$:*

(i) *either the dimension of $\frac{V_\infty}{W}$ is at most k , or*

(ii) *the dimension of $\frac{W}{V}$ is finite,*

and there exists $U \in \mathcal{L}(V_\infty)$ such that the dimension of $\frac{V_\infty}{U}$ is exactly k .

Kalantari and Retzlaff proved that for every $k \geq 0$, there exists a k -thin subspace of V_∞ .

Theorem 2.17 (*Harizanov and Stephan*) *The following statements are equivalent for a c.e. subspace V of V_∞ .*

(i) *The class $\mathcal{L}(V)$ is *BC*-learnable from switching by an algorithmic learner.*

(ii) *The dimension of $\frac{V_\infty}{V}$ is finite, or V is 0-thin, or V is 1-thin.*

Proof. To prove (i) \Rightarrow (ii), we will apply Theorem 2.5. First assume that $\frac{V_\infty}{V}$ has infinite dimension, and that V is neither 0-thin nor 1-thin. Then there is a c.e. space W such that $V \subset W \subset V_\infty$, the quotient space $\frac{W}{V}$ has infinite dimension, and $\frac{V_\infty}{W}$ has dimension at least 2. In particular, there are vectors x_1, x_2 such that $x_1 \notin W$, $x_2 \notin W$, and x_1, x_2 are linearly independent over W . Now, for every finite set D of vectors, one can choose a positive integer n such that none of the vectors in $D - W$ is in the linear closure of $W \cup \{x_1 + nx_2\}$. Furthermore, the linear closure of $V \cup (W \cap D)$ has finite dimension over V , and thus is different from W . So the condition of Theorem 2.5 is satisfied, and hence $\mathcal{L}(V)$ is not *BC*-learnable from switching.

To prove the converse, (ii) \Rightarrow (i), we have to consider only the cases of 0-thin and 1-thin spaces, since Theorem 2.12 deals with the case when the dimension of $\frac{V_\infty}{V}$ is finite. In these two cases, there is a minimal space W such that $V \subseteq W$ and $\frac{W}{V}$ is infinite dimensional. Furthermore, if V is 0-thin, we have that $W = V_\infty$. If V is 1-thin, we have that $W \subset V_\infty$, and there is no other such c.e. vector space U with the quotient space $\frac{U}{V}$ of infinite dimension. This property allows us to give the following learning algorithm for $\mathcal{L}(V)$.

- A learner M requests data of type 0 until such a datum is enumerated into W . The learner's hypothesis is V_∞ as long as no datum of type 0 (except the pause symbol) is given, and W otherwise.
- If some datum of type 0 appears in W , then M requests data of type 1, and guesses the linear closure of $V \cup E$, where E is the set of all data of type 1 seen so far.

In the case when M guesses V_∞ or W , the learner M requests only data of type 0. If none are supplied, the hypothesis V_∞ is correct, and if some negative data are given, but they are in the complement of W , then the hypothesis W is correct. Otherwise, the vector space to be learned is the linear closure of $V \cup D$ for some finite set D . Since this space cannot contain all of W , a datum of type 0 and in W shows up and causes that from that time on M requests only data of type 1. So, the teacher has to eventually reveal to M all elements of the linear closure of $V \cup D$. Since the space given to M is finite dimensional over V , beginning at some stage, D is contained in the linear closure of $V \cup E$, where E is the set of positive data given to M by that stage. ■

If the space V_∞ is over a *finite* field, and V is a c.e. subspace of V_∞ , then the quotient space $\frac{V_\infty}{V}$ has a dependence algorithm. Hence we obtain a different result on *BC*-learnability from switching. On the other hand, Corollary 2.15 remains the same since its proof does not depend on the fact that the field is infinite.

Proposition 2.18 (*Harizanov and Stephan*) *Assume that V_∞ is over a finite field of scalars. Let V be a c.e. subspace of V_∞ , and let k be any natural number. If V is k -thin, then $\mathcal{L}(V)$ is *BC*-learnable from switching by an algorithmic learner.*

Learning from an informant does not have a nice algebraic characterization, at least not one in terms of thin vector spaces, as the following result from [11] shows.

Theorem 2.19 (*Harizanov and Stephan*)

(i) *There is a 0-thin vector space V such that the class $\mathcal{L}(V)$ is *EX*-learnable from an informant by an algorithmic learner.*

(ii) *There is a 0-thin vector space V such that the class $\mathcal{L}(V)$ is not *EX*-learnable from an informant by an algorithmic learner.*

2.4 Conclusion

We presented inductive inference of classes of c.e. sets, c.e. ring ideals, and c.e. vector spaces. With respect to the convergence criterion of learner's hypotheses, we considered both syntactic inference (*EX*-identification) and semantic inference (*BC*-identification). We also considered different ways of giving positive and negative data to the learner. For classes of ring ideals, we considered only inference from text, while for classes of c.e. sets and c.e. vector spaces, we also considered inference from switching, and inference from an informant. It will be interesting to also study inference from switching and from an informant for classes of ring ideals.

Furthermore, it would be worthwhile to systematically investigate inductive inference of c.e. substructures for other important classes of computable

algebraic structures. Some specific results in this direction have already been obtained. Stephan and Ventsov [24] studied inference of classes of *monoids*. For example, they showed that for the standard representation of the integers $(\mathbb{Z}, +)$, the class of all monoids of $(\mathbb{Z}, +)$ can be *EX*-learned from text with the learner's mind change complexity ω^2 . Moreover, this bound is optimal. Stephan and Ventsov also studied inference of classes of closed sets of computably enumerable *partially ordered structures* (P, \sqsubseteq) . Here, the binary relation \sqsubseteq is reflexive ($x \sqsubseteq x$) and transitive ($(x \sqsubseteq y \wedge y \sqsubseteq z) \Rightarrow x \sqsubseteq z$). A subset A of P is *closed* if for every $x \in A$ and $y \sqsubseteq x$, we have $y \in A$. For example, Stephan and Ventsov showed that there is a linear ordering on the natural numbers such that the family of closed sets is *BC*-inferable, but not *EX*-inferable.

Sets and vector spaces can be considered as examples of *closure systems*, which are abstract mathematical structures with dependence relations satisfying certain axioms (see [7]). Such dependence relations generalize linear dependence of vectors in vector spaces, and the identity of elements in sets. Harizanov and Stephan [11] studied inference of classes of c.e. substructures of computable closure systems. For example, Harizanov and Stephan showed that there is a computable closure system whose class of all c.e. closure subsystems can be *EX*-learned from switching, but not with any bound on the number of switches.

Learning with access to an *oracle* (external set with given information about its elements and nonelements) is also important. Some results have already been obtained. For example, Harizanov and Stephan [11] showed that there is a computable closure system whose class of all c.e. subsystems can be *BC*-learned from switching with oracle for the halting set K , but not with any oracle to which K is not Turing reducible.

For inference from text or from switching, many classes of inferable algebraic structures have natural algebraic characterizations. It would be interesting to also find algebraic properties of structures, which exactly correspond to the inferability from an informant, both for *EX*-learning and *BC*-learning.

Acknowledgment. I wish to thank John Case for helpful comments on this paper and many valuable discussions about algorithmic learning theory.

Bibliography

- [1] Angluin, D. (1980). “Inductive Inference of Formal Languages from Positive Data”, *Information and Control* 45, 117–135.
- [2] Baliga, G., Case, J. and Jain, S. (1995). “Language Learning with Some Negative Information”, *Journal of Computer and System Sciences* 51, 273–285.
- [3] Blum, L. and Blum, M. (1975). “Toward a Mathematical Theory of Inductive Inference”, *Information and Control* 28, 125–155.
- [4] Case, J. and Lynes, C. (1982). “Machine Inductive Inference and Language Identification”, in Nielsen and Schmidt [18], 107–115.
- [5] Case, J. and Smith, C. (1983). “Comparison of Identification Criteria for Machine Inductive Inference”, *Theoretical Computer Science* 25, 193–220.
- [6] Cesa-Bianchi, N., Numao, M. and Reischuk, R. (eds.) (2002). *Algorithmic Learning Theory: 13th International Conference*, Lecture Notes in Artificial Intelligence 2533, Berlin: Springer-Verlag.
- [7] Downey, R.G. and Remmel, J.B. (1998). “Computable Algebras and Closure Systems: Coding Properties”, in Ershov, Goncharov, Nerode and Remmel [8], 977–1039.
- [8] Ershov, Yu.L., Goncharov, S.S., Nerode, A. and Remmel, J.B. (eds.) (1998). *Handbook of Recursive Mathematics*, Vol. 2, Amsterdam: Elsevier.
- [9] Gold, E.M. (1967). “Language Identification in the Limit”, *Information and Control* 10-5, 447–474.
- [10] Griffor, E.R. (ed.) (1999). *Handbook of Computability Theory*, Amsterdam: Elsevier.
- [11] Harizanov, V.S. and Stephan, F. (2002). “On the Learnability of Vector Spaces”, in Cesa-Bianchi, Numao and Reischuk [6], 233–247.
- [12] Jain, S. and Stephan, F. (2003). “Learning by Switching Type of Information”, *Information and Computation* 185, 89–104.

- [13] Jain, S., Osherson, D., Royer, J.S. and Sharma, A. (1999). *Systems That Learn: An Introduction to Learning Theory*, 2nd ed., Cambridge (Mass.): MIT Press.
- [14] Kalantari, I. and Retzlaff, A. (1977). “Maximal Vector Spaces Under Automorphisms of the Lattice of Recursively Enumerable Vector Spaces”, *Journal of Symbolic Logic* 42-4, 481–491.
- [15] Kaplansky, I. (1974). *Commutative Rings*, Chicago: The University of Chicago Press.
- [16] Metakides, G. and Nerode, A. (1977). “Recursively Enumerable Vector Spaces”, *Annals of Mathematical Logic* 11, 147–171.
- [17] Motoki, T. (1991). “Inductive Inference from All Positive and Some Negative Data”, *Information Processing Letters* 39-4, 177–182.
- [18] Nielsen, M. and Schmidt, E.M. (eds.) (1982). *Automata, Languages and Programming: Proceedings of the 9th International Colloquium*, Lecture Notes in Computer Science 140, Berlin: Springer-Verlag.
- [19] Odifreddi, P. (1989). *Classical Recursion Theory*, Amsterdam: North-Holland.
- [20] Osherson, D.N. and Weinstein, S. (1982). “Criteria of Language Learning”, *Information and Control* 52-2, 123–138.
- [21] Osherson, D.N., Stob, M. and Weinstein, S. (1986). *Systems That Learn: An Introduction to Learning Theory for Cognitive and Computer Scientists*, Cambridge (Mass.): MIT Press.
- [22] Sharma, A. (1998). “A Note on Batch and Incremental Learnability”, *Journal of Computer and System Sciences* 56-3, 272–276.
- [23] Soare, R.I. (1987). *Recursively Enumerable Sets and Degrees. A Study of Computable Functions and Computably Generated Sets*, Berlin: Springer-Verlag.
- [24] Stephan, F. and Ventsov, Yu. (2001). “Learning Algebraic Structures from Text”, *Theoretical Computer Science* 268, 221–273.
- [25] Stoltenberg-Hansen, V. and Tucker, J.V. (1999). “Computable Rings and Fields”, in Griffor [10], 363–447.

DEPARTMENT OF MATHEMATICS, GEORGE WASHINGTON UNIVERSITY,
 WASHINGTON, D.C. 20052, U.S.A., HARIZANV@GWU.EDU