Learning Automatic Families of Languages ⋆

Sanjay Jain[1] and Frank Stephan[2]

[1] School of Computing, National University of Singapore, Singapore 117417.
Email: sanjay@comp.nus.edu.sg

[2] Department of Mathematics and Department of Computer Science, National University of Singapore, Singapore 119076.
Email: fstephan@comp.nus.edu.sg

Abstract. A class of languages is automatic if it is uniformly regular using some regular index set for the languages. In this survey we report on work about the learnability in the limit of automatic classes of languages, with some special emphasis to automatic learners.

1 Introduction

A language is a set of strings over some finite alphabet. Consider the following model of language learning. A learner receives all elements of the target language, one elment at a time, repetition allowed, in arbitrary order (this form of information provided to the learner is called a text for the language). Note that no non-elements of the language are provided to the learner. For technical reasons, we allow a special symbol # as input, which denotes "no datum" (this allows for a text of empty language as an infinite sequence of #'s). After receiving each new element the learner outputs its conjecture about what the target language might be (this is usually expressed in the form of a grammar for the language, in some hypothesis space). If the sequence of grammars output by the learner converges to a correct grammar for the target language then the learner is said to identify the language (from the corresponding text). For learning a language, the learner is expected to learn it from all texts for the language. Learning of one language is not useful, as a learner which outputs just a grammar for the language, whatever the input might be, is similar to a person who predicts earthquake everyday, and is right on the day earthquake actually occurs. So what is more interesting is whether the same learner can learn all languages from a class of languages. This is essentially the model of learning proposed by Gold [11] and called **TxtEx**-learning. In Gold's original model there is no restriction on the memory of the learner. Thus, the learner can remember all its past input data when it comes up with its new hypothesis. In some cases below we will consider restrictions on the memory of the learner. Thus, we define a learner taking this into account.

Let $\mathbb{N}$ denote the set of natural numbers.

⋆ Research for this work is supported in part by NUS grants C252-000-087-001 (S. Jain) and R146-000-181-112 (S. Jain and F. Stephan)

A *text* T is a mapping from $\mathbb{N}$ to $\Sigma^* \cup \{\#\}$. Content of a text T, denoted $\mathrm{content}(T) = \{T(i) : i \in \mathbb{N}\} - \{\#\}$. $T[n]$ denotes the initial sequence $T(0)T(1)\ldots T(n-1)$ of the text T, of length n.

A finite sequence is an initial segment of a text. SEQ denotes the set of all finite sequences. For a finite sequence $T[n]$, $\mathrm{content}(T[n]) = \{T(i) : i \in \mathbb{N}\} - \{\#\}$.

We use $\sigma \diamond \tau$ to denote the concatenation of two finite sequences σ and τ. Similarly, $\sigma \diamond T$ denotes the concatentation of σ and T.

Definition 1 (Based on Gold [11]; see also [5, 17]). Suppose Σ is the alphabet set for the languages and $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ is a class of languages, to be learnt, where I is an index set. Let $\mathcal{H} = \{H_\beta : \beta \in J\}$ be the hypothesis space, used by the learner, where J is the index set for the hypotheses. We always assume that $\mathcal{H}$ is uniformly r.e.; in some cases below we will put more restrictions on the hypothesis space. Let ? be a special symbol not in J which denotes "no new conjecture at this point". Suppose Γ is a finite set of alphabet used for memory by the learner.

(a) A learner is an algorithmic mapping from $\Gamma^* \times (\Sigma^* \cup \{\#\})$ to $\Gamma^* \times (J \cup \{?\})$. A learner has an initial memory $mem_0 \in \Gamma^*$ and initial conjecture $hyp_0 \in J \cup \{?\}$.
Intuitively, for a learner M, $M(mem, x) = (mem', hyp)$, means that based on old memory mem and current datum x, the new memory of the learner is mem' and hyp is its conjecture.

(b) Suppose a learner M with initial memory mem_0 and initial hypothesis hyp_0 is given. Suppose T is a text for a language L.
  - **(i)** Let $mem_0^T = mem_0$ and $hyp_0^T = hyp_0$.
  - **(ii)** For $k > 0$, let $(mem_k^T, hyp_k^T) = M(mem_{k-1}^T, T(k-1))$
  - **(iii)** Define $M(T[k]) = (mem_k^T, hyp_k^T)$.
  - **(iv)** M on T *converges* on text T to the hypothesis hyp iff, for all but finitely many k, $hyp_k^T = hyp$.

(c) M **TxtEx**-learns a language L (using hypothesis space $\mathcal{H}$) if, for all texts T for L, M on T converges to a hypothesis β such that $H_\beta = L$.

(d) M **TxtEx**-learns the class $\mathcal{L}$ (using hypothesis space $\mathcal{H}$) iff M **TxtEx**-learns all the languages in the class $\mathcal{L}$ (using hypothesis space $\mathcal{H}$).

(e) **TxtEx** $= \{\mathcal{L}$: some learner M learns $\mathcal{L}$ using some automatic family $\mathcal{H}$ as the hypothesis space$\}$.

Intuitively, mem_k^T and hyp_k^T in part (b) above denote the memory and conjecture of the learner M after having seen the data $T[k]$.

We now consider automatic classes of languages. Intuitively, a class of languages is automatic if the class is uniformly regular. More formally, let Σ be a finite alphabet, and let @ be a special symbol not in Σ. Given two strings $x = x_0x_1\ldots x_{n-1}$ and $y = y_0y_1\ldots y_{m-1}$ over the alphabet Σ, define convolution of x and y, $conv(x, y)$ as follows. Let $r = \max(\{m, n\})$. For $i < n$, let $x'_i = x_i$; for $n \leq i < r$, let $x'_i = @$. For $i < m$, let $y'_i = y_i$; for $m \leq i < r$, let $y'_i = @$. Now, convolution of x, y is defined as $conv(x, y) = z_0z_1\ldots z_{r-1}$, where $z_i = (x'_i, y'_i)$;

note that z_i is a member of the alphabet $\Sigma \cup \{@\} \times \Sigma \cup \{@\}$. One can extend the definition of convolution to multiple strings similarly.

A class of languages $\mathcal{L}$ is said to be automatic if there is an indexing $(L_\alpha)_{\alpha \in I}$, for some regular index set I such that, $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ and $\{conv(\alpha, x) : x \in L_\alpha\}$ is regular [20]. A relation $R = \{(x_1, x_2, \ldots, x_n) : x_1, x_2, \ldots, x_n \in \Sigma^*\}$ is said to be automatic if $\{conv(x_1, x_2, \ldots, x_n) : (x_1, x_2, \ldots, x_n) \in R\}$ is regular. Similarly, a function f is said to be automatic if $\{conv(x, y) : f(x) = y\}$ is regular.

The following theorem is important and it also enables to derive that the first-order theory of any given automatic structure is decidable.

Theorem 2 (Blumensath and Grädel [4], Khoussainov and Nerode [23]). *Any relation that is first-order definable from existing automatic relations is automatic and there is an algorithm to construct the corresponding automaton from automata for the relations and functions of the automatic structure and the defining formula.*

Furthermore, one can characterise the automatic functions as functions which map convoluted tuples to convoluted tuples and which can be computed by a one-tape Turing machine with the output starting at the same position as originally the input started and computation time being linear [7]; for this characterization one can either use deterministic or non-deterministic Turing machines.

When learning automatic classes, we usually require that the hypothesis space $\mathcal{H}$ is also automatic. This paper surveys some of the results in learnability of countable automatic classes of languages. For learnability of uncountable classes we refer the reader to Jain, Luo, Semukhin and Stephan [18].

2 Characterization of Learnability of Autmatic Classes

For the characterization, we first consider the notion of tell-tale sets as introduced by Angluin. Let $x <_{ll} w$ denote that x is length-lexicographically smaller than w. That is $|x| < |w|$ or $|x| = |w|$ and x is lexicographically before w (based on some fixed ordering of the alphabet). Let $x \leq_{ll} w$ denote that $x <_{ll} w$ or $x = w$.

Definition 3 (Angluin's Tell Tale condition [2]). Suppose $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ is a class of languages.

(a) D is a *tell-tale* of L (with respect to $\mathcal{L}$) iff D is finite and for all $L' \in \mathcal{L}$, $D \subseteq L' \subseteq L$ implies $L = L'$.
(b) $\mathcal{L}$ satisfies Angluin's Tell-Tale condition iff every $L \in \mathcal{L}$ has a tell-tale with respect to $\mathcal{L}$.
(c) [17] For all $L \in \mathcal{L}$, we say that w is a *tell-tale cut-off word* for L (with respect to $\mathcal{L}$) iff $\{x \in L : x \leq_{ll} w\}$ is a tell-tale for L (with respect to $\mathcal{L}$).

Essentially, Angluin [2] showed that if a class $\mathcal{L}$ does not satisfy Angluin's tell-tale condition, then it cannot be **TxtEx**-learnable. This result applies even for general classes of r.e. languages, and even for non-recursive learners, though Angluin's stated theorem was only for indexed families and recursive learners.

Jain, Luo and Stephan showed that a class satisfying Angluin's tell-tale condition is enough for **TxtEx**-learnability of automatic classes. In particular they showed that such a learner can have several useful additional properties.

Definition 4. Suppose M is a learner. The notation is as in Definition 1 for memory and hypothesis of M on a text T.

(a) [3] M is said to be *consistent* on L if, for all texts T for L, for all n, $H_{hyp_k^T} \supseteq \text{content}(T[k])$. M is said to be consistent on $\mathcal{L}$ if it is consistent on each $L \in \mathcal{L}$.
(b) [2] M is said to be *conservative* on L if, for all texts T for L, for all k, if $\text{content}(T[k+1]) \subseteq H_{hyp_k^T}$, then $hyp_{k+1}^T = hyp_k^T$. M is said to be conservative on $\mathcal{L}$ if it is conservative on each $L \in \mathcal{L}$.
(c) [28, 31] M is said to be *set-driven* if, for all σ and τ in SEQ, if $\text{content}(\sigma) = \text{content}(\tau)$, then $M(\sigma) = M(\tau)$.

When we say that M consistently (conservatively, set-drivenly, etc.) learns $\mathcal{L}$, we mean that M **TxtEx**-learns $\mathcal{L}$, and is consistent (resepctively conservative, set-driven) on $\mathcal{L}$.

Theorem 5 (Jain, Luo and Stephan [17]). *Suppose $\mathcal{L}$ is automatic and satisfies Angluin's tell-tale condition. Then there exists a learner M which is set-driven, consistent and conservative on $\mathcal{L}$ and which* **TxtEx***-learns $\mathcal{L}$.*

Note that if an automatic class $\mathcal{L}$ does not satisfy Angluin's tell-tale condition, then there is no learner (even non-recursive learner) which **TxtEx**-learns $\mathcal{L}$. As the tell-tale condition is first-order definable, we get the following corollary:

Corollary 6 (Jain, Luo and Stephan [17]). *It is decidable whether a given family $\{L_\alpha : \alpha \in I\}$ is* **TxtEx***-learnable (where the decision algorithm gets as input the alphabet Σ and DFAs for regular set I and the regular set $\{conv(\alpha, x) : x \in L_\alpha\}$).*

A similar characterization for some other learning criteria can also be obtained. Let us consider Finite learning.

Definition 7 (Gold [11]).

(a) M **TxtFin**-learns L (using hypothesis space $\mathcal{H} = (H_\beta)_{\beta \in J}$) iff for all texts T for L, there exists an n and a β such that:
  (i) For $m < n$, $M(T[n]) \in \Gamma \times \{?\}$;
  (ii) For $m \geq n$, $M(T[n]) \in \Gamma \times \{\beta\}$;
  (iii) $H_\beta = L$.
(b) M **TxtFin**-learns $\mathcal{L}$ (using hypothesis space $\mathcal{H}$) iff it **TxtFin**-learns each $L \in \mathcal{L}$ (using hypothesis space $\mathcal{H}$).
(c) $\textbf{TxtFin} = \{\mathcal{L} : (\exists M)[M$ **TxtFin**-learns $\mathcal{L}$ using some automatic hypothesis space $\mathcal{H}]\}$.

A useful concept for **TxtFin**-learnability is the concept of characteristic sample.

Definition 8 (Lange and Zeugmann [26], Mukouchi [27]).

(a) A finite set S is a *characteristic sample* for L with respect to the class $\mathcal{L}$ iff
  (i) $S \subseteq L$.
  (ii) For all $L' \in \mathcal{L}$, $S \subseteq L'$ implies $L = L'$.
(b) $\mathcal{L}$ satisfies the characteristic sample property iff every $L \in \mathcal{L}$ has a characteristic sample with respect to $\mathcal{L}$.

Theorem 9 (Jain, Luo and Stephan [17]). *Suppose $\mathcal{L}$ is an automatic class. Then $\mathcal{L} \in$ **TxtFin** iff it satisfies the characteristic sample property.*

3 Automatic Learners

A learner M is automatic if the function it computes is automatic. That is, the mapping (old mem, datum) $\mapsto$ (new mem, hyp) is automatic. When requiring the learners to be automatic, we add the term **Auto** in the learning criteria (for example **AutoTxtEx** means **TxtEx**-learnability using automatic learners). Besides, we often require some restrictions on the memory. These are mentioned in the following.

Definition 10. Fix a learner M. For a text T, let mem_k^T and hyp_k^T be as in Definition 1.

(a) [33] The learner M is *iterative* if, for all texts T and k, $mem_k^T = hyp_k^T$.
(b) [17] The learner M is *word-size memory limited* if there exists a constant c such that for all T and k, $|mem_k^T| \leq \max(\{|T(m)| : m < k\}) + c$.
(c) [17] The learner M is *hypothesis-size memory limited* if there exists a constant c such that for all T and k, $|mem_k^T| \leq |hyp_k^T| + c$.

We denote the above memory restrictions on a learner by using the terms **It**, **Word** and **Index** in the criteria names. For example, **AutoWordTxtEx** denotes **TxtEx**-learning by an automatic learner with word-size memory limitation. The next theorem shows that requiring learners to be automatic can be very restrictive. This is not only because automatic learners have limited memory: automatic learners cannot even learn classes which can be iteratively learnt.

Theorem 11 (Jain, Luo and Stephan [17]). *The automatic class*

$$\{\{0,1\}^* - \{x\} : x \in \{0,1\}^*\}$$

is **ItTxtEx**-*learnable but not* **AutoTxtEx**-*learnable.*

Theorem 12 (Jain, Luo and Stephan [17]).

(a) **AutoItTxtEx** $\subseteq$ **AutoWordTxtEx** $\subseteq$ **AutoTxtEx**.
(b) **AutoItTxtEx** $\subseteq$ **AutoIndexTxtEx** $\subseteq$ **AutoTxtEx**.
(c) **AutoWordTxtEx** $\not\subseteq$ **AutoIndexTxtEx**.

At the time of writing, unsolved problems are whether **AutoWordTxtEx** = **AutoTxtEx** and whether **AutoIndexTxtEx** = **AutoItTxtEx**. Furthermore, it is open whether **AutoIndexTxtEx** $\subseteq$ **AutoWordTxtEx**.

However, if the alphabet for languages is unary, then **AutoIndexTxtEx** $\subset$ **AutoWordTxtEx** = **AutoTxtEx**.

An interesting class which is automatically learnable is unions of regular patterns languages which have variables only among the last n symbols.

Angluin [1] introduced the pattern languages and Shinohara [32] investigated learnability of the class of pattern languages generated by regular patterns. Automatic classes of pattern languages are a special case of classes of pattern languages generated by regular patterns and the n-th automatic class $\mathcal{P}_n$ is defined as follows: For a finite alphabet Σ, $\mathcal{P}_n$ contains all pattern languages of the form $\alpha b_1 b_2 \ldots b_n$, where $\alpha \in \Sigma^*$ and each $b_1, b_2, \ldots, b_n$ is either a member of Σ or the set Σ^*; for example, if $\Sigma = \{0, 1, 2\}$, then $0112101012\Sigma^*21\Sigma^*$ is an automatic pattern language in $\mathcal{P}_4$.

Theorem 13 (Case et al. [9]). *Suppose $|\Sigma| \geq 3$ and $n > 0$. Then, $\mathcal{L} = \{L_1 \cup L_2 : L_1, L_2 \in \mathcal{P}_n\}$ is* **AutoWordTxtEx***-learnable via a learner that, furthermore, for all texts T for a language L outside $\mathcal{L}$, converges to an index for a language L' such that $L - L'$ is finite.*

4 Automatic Learning From Fat Text

A text T is called *fat* (see [29]) if every member of content(T) appears infinitely often in the text. That is, for all $x \in$ content(T), there exist infinitely many n such that $T(n) = x$.

As the automatic learners are very much memory limited, one may expect to overcome some of these limitations using a fat text. In fact that is indeed the case as shown by the following result. For learnability of a class from fat texts, we just require the learnability when the input text is fat, and do not care what happens when the input text is not fat.

Theorem 14 (Jain, Luo and Stephan [17]). *Suppose $\mathcal{L}$ is an automatic class which satisfies Angluin's tell-tale condition. Then there exists a learner M which* **AutoWordTxtEx***-learns $\mathcal{L}$ from fat texts.*

Osherson, Stob and Weinstein [29] considered partial learning in which the learner need not converge to a correct hypothesis but instead satisfy the following for any text T for the language L being learnt:

(a) Exactly one hypothesis is output infinitely often — that is, there exists exactly one p such that $hyp_k^T = p$ for infinitely many k;
(b) The unique p which is output infinitely often is a grammar for the input language L.

Theorem 15 (Jain, Luo and Stephan [17]). *Every automatic class $\mathcal{L}$ can be partially learnt by an automatic learner with word-size limited memory from fat texts.*

5 Negative Counterexamples

The model of learning in which the learners get only positive data, as considered in most of the literature on inductive inference, is based on the studies by linguists that children mainly get only positive data. However, this is not entirely true as the children are often told about the errors they make. Thus, there is some negative data, in the form of counterexamples that is given to the children. In this section we consider giving the learner negative counterexamples, if any, to their conjectures. This is given in the form of a separate text, where the new datum is either an appropriate negative counterexample (if it exists) to the previous conjecture or a # (indicating no negative counterexample).

For this, the learner is considered as a mapping from (old mem, new datum, new counter example) to (new mem, new hypothesis). We can then define $M(T[n], T'[n])$ as the pair of memory and conjecture of the learner after having seen the text $T[n]$ and corresponding counterexamples given by $T'[n]$. As this definition is a straightforward generalization of Definition 1, we omit the details of the learner but concentrate on how the counterexample text is defined.

Definition 16 (Jain and Kinber [13]). Suppose M is a learner and $\mathcal{H} = (H_\beta)_{\beta \in j}$ is the hypothesis space used by M.

- **(a)** T' is a counterexample text for M on input text T for a language L iff for all n, where $M(T[n], T'[n]) = (mem, hyp)$,
  if $H_{hyp} \subseteq L$, then $T'(n) = \#$
  if $H_{hyp} \not\subseteq L$, then $T'(n) \in H_{hyp} - L$.
- **(b)** T' is a least-counterexample text for M on input text T for a language L iff for all n, where $M(T[n], T'[n]) = (mem, hyp)$,
  if $H_{hyp} \subseteq L$, then $T'(n) = \#$
  if $H_{hyp} \not\subseteq L$, then $T'(n) = \min(H_{hyp} - L)$.
- **(c)** M **NCEx**-learns a language L (using hypothesis space $\mathcal{H}$) iff for all texts T for L, and all corresponding counterexample texts T' for M on the input text T, $M(T, T')$ converges to a hypothesis hyp such that $H_{hyp} = L$.
- **(d)** M **NCEx**-learns a class $\mathcal{L}$ of languages (using hypothesis space $\mathcal{H}$) iff it **NCEx**-learns each language from $\mathcal{L}$ (using hypothesis space $\mathcal{H}$).
- **(e)** **NCEx** $= \{\mathcal{L} : (\exists M)[M$ **NCEx**-learns $\mathcal{L}$ using some automatic family $\mathcal{H}$ as the hypothesis space$\}$.

One can similarly define **ItNCEx**, **ItLNCEx**, **LNCEx** and other learning criteria (here **LNC** stands for least counterexample).

Theorem 17 (Jain and Kinber [13]). *Suppose $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ is an automatic family.*

- **(a)** $\mathcal{L} \in$ **AutoItNCEx** *via a learner that uses a class preserving hypothesis space, $\mathcal{H} = \{H_\beta : \beta \in J\}$, where the languages in $\mathcal{H}$ are same as that in $\mathcal{L}$, though indexing may be different (with potentially several copies of the same language).*

(b) *$\mathcal{L} \in$ **AutoWordNCEx** via a learner that uses the hypothesis space $(H_\alpha)_{\alpha \in I}$, where $H_\alpha = L_\alpha$.*

The learners witnessing the above result are however inconsistent. For consistency, we need least negative counterexamples.

Theorem 18 (Jain, Kinber and Stephan [16]). *Every automatic family is* **AutoWordLNCEx**-*learnable via a consistent learner.*

The learner in the above proof however uses a general automatic hypothesis space, which may contain lanugages outside the class $\mathcal{L}$. It can be shown that some automatic class $\mathcal{L}$ cannot be **AutoLNCEx**-learnt using a class preserving automatic hypothesis space.

6 Parallel Learning of Automatic Classes

In parallel learning, the learner simultaneously gets texts for n distinct languages from the target class $\mathcal{L}$, and outputs its conjectures for the corresponding texts. This study was originated for general **TxtEx**-learning by [24] and then studied by [14, 15] for the case of learning automatic families. These learning criteria are denoted by (m, n)-**TxtEx** and (m, n)-**TxtFin**-learning. Here, note that for the above learning criteria, when the input texts are for distinct languages in the class, we require the sequence of conjectures to converge on all the texts, whether the corresponding texts are actually learnt or not.

Furthermore, one can also distinguish the cases of the learner being required to specify the m texts which it learns, and the learner only being required to learn m of the n texts, without any constraint on specifying which texts it learnt. The requirement of specifying the texts which are learnt is denoted by using **Super** in the name of the learning criteria.

Theorem 19 (Jain and Kinber [14, 15]).

(a) *Suppose $0 < m \leq n$, $\mathcal{L}$ is an automatic family and all except at most $n - m$ langauges $L \in \mathcal{L}$ have a characteristic sample with respect to $\mathcal{L}$. Then $\mathcal{L}$ is (m, n)-**SuperTxtFin**-learnable.*

(b) *Suppose $0 < m \leq n$ and $\mathcal{L}$ is an automatic class having at least $2n + 1 - m$ languages. Then (m, n)-**SuperTxtFin**-learnability of $\mathcal{L}$ implies that there are at most $n - m$ languages in $\mathcal{L}$ which do not have a characteristic sample with respect to $\mathcal{L}$.*

For the case when the learner is not required to specify the languages it learns, the characterisation is slightly different.

Theorem 20 (Jain and Kinber [14, 15]).

(a) *Suppose $0 < m \leq n$, $\mathcal{L}$ is an automatic family, and there exists a subset $\mathcal{S}$ of $\mathcal{L}$ of cardinality at most $n - m$ such that every language in $\mathcal{L} - \mathcal{S}$ has a characteristic sample with respect to $\mathcal{L} - \mathcal{S}$. Then $\mathcal{L}$ is (m, n)-**TxtFin**-learnable.*

(b) *Suppose* $0 < m \leq n$, $\mathcal{L}$ *is an automatic class having at least* $2n + 1 - m$ *languages. Then* (m, n)-**TxtFin**-*learnability of* $\mathcal{L}$ *implies that there exists a subclass* $\mathcal{S}$ *of* $\mathcal{L}$ *of cardinality at most* $n - m$ *such that every language in* $\mathcal{L} - \mathcal{S}$ *has a characteristic sample with respect to* $\mathcal{L} - \mathcal{S}$.

For finite automatic classes the situation becomes a bit more complicated and full characterization depends on a combinatorial argument. We refer the reader to [15] for full details. Furthermore, one can show a hierarchy for (m, n)-learnability as follows.

Theorem 21 (Jain and Kinber [14, 15]).

(a) *Suppose* $0 < m < n$. *Then, there exists an automatic class* $\mathcal{L}$ *which is* (m, n)-**SuperTxtFin**-*learnable but not* $(m + 1, n)$-**TxtFin**-*learnable.*
(b) *There exists an automatic class* $\mathcal{L}$ *such that for all* $n > 1$, $\mathcal{L}$ *is* $(n - 1, n)$-**TxtFin**-*learnable but not* $(1, n)$-**SuperTxtFin**-*learnable.*

For **TxtEx**-learnability, (m, n)-**TxtEx**-learnability and (m, n)-**SuperTxtEx**-learnability coincide [15]; thus we give the following results only for (m, n)-**TxtEx**.

Theorem 22 (Jain and Kinber [14, 15]).

(a) *Suppose* $0 < m \leq n$ *and* $\mathcal{L}$ *is an automatic class. Then* $\mathcal{L}$ *is* (m, n)-**TxtEx**-*learnable iff at most* $n - m$ *languages in* $\mathcal{L}$ *do not have a tell-tale with respect to* $\mathcal{L}$.
(b) *For* $n > 0$ *there exists an automatic* $(1, n + 1)$-**TxtEx**-*learnable class that is not* $(1, n)$-**TxtEx**-*learnable.*
(c) *For* $n > 0$, $(1, n)$-**TxtFin** $\subseteq$ **TxtEx**.

Jain and Kinber also explore the above model of parallel learning when the learners are automtic. However, here the picture becomes more complicated and full characterization is not yet known.

7 Robust Learning of Automatic Classes

Intuitively, a class of objects is robustly learnable if every computable transformation of the class is learnable. Robust learnability seems a desirable property as it indicates that learning of the class is not due to presence of some artificial coding within the data, but is due to the structure of the class itself (Bārzdiņš, in the 1970s). Bārzdiņš reasoned that if a class is learnable only due to some self-referential property then this self-referential part can be "removed" via computable transformations, and thus the class be transformed into an unlearnable class. This line of research has been explored in various papers such as [6, 8, 10, 21, 12, 25, 30]. However, most of these work were on function learning and there does not seem to be a good definition for robust learning of classes of languages. Using automatic classes and related transformations, [19] explored robust learning of classes of automatic languages. The translations which were considered valid for this are defined as follows:

Definition 23 (Jain, Martin and Stephan [19]). Let an automatic class $(L_\alpha)_{\alpha \in I}$ be given. Let Φ be a first order formula, with a distinguished variable x as a unique free variable, where Φ is allowed to use predicates "$y \in X$" and "$y \in L_\alpha$" along with the set I. Let $\Phi(L)$ be the the language consisting of all strings s such that $\Phi[s/x]$ is true, where X is taken to be L. Φ is an automatic translator (for the automatic family $\mathcal{L}$) if the following conditions hold:

(a) For all languages L, L', if $L \subseteq L'$, then $\Phi(L) \subseteq \Phi(L')$;
(b) For all languages $L, L' \in \mathcal{L}$, if $L \nsubseteq L'$, then $\Phi(L) \nsubseteq \Phi(L')$.

Let $\Phi((L_\alpha)_{\alpha \in I}) = (\Phi(L_\alpha))_{\alpha \in I}$. It is easy to verify that any translation of an automatic family is automatic. Jain, Martin and Stephan [19] considered various properties such as consistency, conservativeness, strong monotonicity and confidence and obtained various charaterizations on when automatic classes are robustly learnable and when some translations of automatic classes are learnable under above constraints. We consider some of these characterizations below.

Theorem 24 (Jain, Martin and Stephan [19]). *Given an automatic class $\mathcal{L} = (L_\alpha)_{\alpha \in I}$, the following are equivalent:*

(a) *Every translation of $\mathcal{L}$ is* **TxtEx**-*learnable.*
(b) *For all $\alpha \in I$, there exists a $b_\alpha \in I$ such that for all $\beta \in I$, either $L_\beta \not\subset L_\alpha$ or there exists a $\gamma \leq_{ll} b_\alpha$ such that $L_\alpha \nsubseteq L_\gamma$ and $L_\beta \subseteq L_\gamma$.*

Theorem 25 (Jain, Martin and Stephan [19]). *Suppose $\mathcal{L}$ is an automatic class all of whose translations are* **TxtEx**-*learnable. Then, every translation of $\mathcal{L}$ is consistently and conservatively* **TxtEx**-*learnable iff $\mathcal{L}$ is well founded under inclusion.*

A learner M is said to be *strong monotonic* [22] if, for all texts T, for all $m < n$, if $hyp_m^T \neq ?$ and $hyp_n^T \neq ?$, then $H_{hyp_m^T} \subseteq H_{hyp_n^T}$, where hyp_s^T denotes the hypothesis of M after seeing input $T[s]$.

Theorem 26 (Jain, Martin and Stephan [19]). *Given an automatic class $\mathcal{L} = (L_\alpha)_{\alpha \in I}$, the following are equivalent:*

(a) *Every translation of $\mathcal{L}$ is strong monotonically* **TxtEx**-*learnable.*
(b) *For all $\alpha \in I$, there exists a $b_\alpha \in I$ such that for all $\beta \in I$, either $L_\alpha \subseteq L_\beta$ or there exists a $\gamma \leq_{ll} b_\alpha$ such that $L_\alpha \nsubseteq L_\gamma$ and $L_\beta \subseteq L_\gamma$.*

Furthermore, every automatic class has some translation which is strong monotonically **TxtEx**-learnable.

Acknowledgements: This survey consists of work done with several authors: John Case, Efim Kinber, Trong Dao Le, Qinglong Luo, Eric Martin, Yuh Shin Ong, Shi Pu, Samuel Seah and Pavel Semukhin.

References

1. D. Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21(1):46–62, 1980.
2. D. Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45:117–135, 1980.
3. J. Bārzdiņš. Inductive inference of automata, functions and programs. In *Proceedings of the 20th International Congress of Mathematicians, Vancouver*, pages 455–460, 1974. In Russian. English translation in American Mathematical Society Translations: Series 2, 109:107-112, 1977.
4. A. Blumensath and E. Grädel. Automatic structures. In *15th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 51–62. IEEE Computer Society, 2000.
5. J. Case, S. Jain, Y. S. Ong, P. Semukhin and F. Stephan. Automatic learners with feedback queries. *Journal of Computer and System Sciences*, 80:806–820, 2014.
6. J. Case, S. Jain, M. Ott, A. Sharma and F. Stephan. Robust learning aided by context. *Journal of Computer and System Sciences (Special Issue for COLT'98)*, 60:234–257, 2000.
7. J. Case, S. Jain, S. Seah and F. Stephan. Automatic functions, linear time and learning. In S.B. Cooper, A. Dawar and B. Löwe, editors, *How the World Computes - Turing Centenary Conference and Eighth Conference on Computability in Europe (CiE 2012), Proceedings*, volume 7318 of *Lecture Notes In Computer Science*, pages 96–106. Springer, Berlin, 2012.
8. J. Case, S. Jain, F. Stephan and R. Wiehagen. Robust learning – rich and poor. *Journal of Computer and System Sciences*, 69(2):123–165, 2004.
9. John Case, Sanjay Jain, Trong Dao Le, Yuh Shin Ong, Pavel Semukhin and Frank Stephan. Automatic learning of subclasses of pattern languages. *Information and Computation*, 218:17–35, 2012.
10. M. Fulk. Robust separations in inductive inference. In *31st Annual IEEE Symposium on Foundations of Computer Science*, pages 405–410. IEEE Computer Society Press, 1990.
11. E. M. Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.
12. S. Jain. Robust behaviorally correct learning. *Information and Computation*, 153(2):238–248, September 1999.
13. S. Jain and E. Kinber. Automatic learning from positive data and negative counterexamples. In Nicolas Vayatis Nader Bshouty, Gilles Stoltz and Thomas Zeugmann, editors, *Algorithmic Learning Theory: 23rd International Conference (ALT' 2012)*, volume 7568 of *Lecture Notes in Artificial Intelligence*, pages 66–80. Springer-Verlag, 2012.
14. S. Jain and E. Kinber. Parallel learning of automatic classes of languages. In Sandra Zilles Peter Auer, Alexander Clark and Thomas Zeugmann, editors, *Algorithmic Learning Theory: 23rd International Conference (ALT' 2014)*, volume 8776 of *Lecture Notes in Artificial Intelligence*, pages 70–84. Springer-Verlag, 2014.
15. S. Jain and E. Kinber. Parallel learning of automatic classes of languages. *Theoretical Computer Science A*, 2016. Accepted. Special Issue on Algorithmic Learning Theory 2014.
16. S. Jain, E. Kinber and F. Stephan. Automatic learning from positive data and negative counterexamples. 2014. Manuscript.

17. S. Jain, Q. Luo and F. Stephan. Learnability of automatic classes. *Journal of Computer and System Sciences*, 78(6):1910–1927, 2012.
18. S. Jain, Q. Luo, F. Stephan and P. Semukhin. Uncountable automatic classes and learning. *Theoretical Computer Science*, 412(19):1805–1820, 2011. Special issue: Algorithmic Learning Theory, 2009.
19. S. Jain, E. Martin and F. Stephan. Robust learning of automatic classes of langauges. *Journal of Computer and System Sciences*, 80:777–795, 2014.
20. S. Jain, Y. S. Ong, S. Pu and F. Stephan. On automatic families. In T. Arai, Q. Feng, B. Kim, G. Wu and Y. Yang, editors, *Proceedings of the 11th Asian Logic Conference, in Honor of Professor Chong Chitat's 60th birthday, 2009*, pages 94–113. World Scientific, 2011.
21. S. Jain, C. Smith and R. Wiehagen. Robust learning is rich. *Journal of Computer and System Sciences*, 62(1):178–212, 2001.
22. K. Jantke. Monotonic and nonmonotonic inductive inference of functions and patterns. In *Nonmonotonic and Inductive Logic, 1st International Workshop, Karlsruhe, Germany*, volume 543 of *Lecture Notes in Computer Science*, pages 161–177. Springer-Verlag, 1990.
23. B. Khoussainov and A. Nerode. Automatic presentations of structures. In *Logical and Computational Complexity, (International Workshop LCC 1994)*, volume 960 of *Lecture Notes in Computer Science*, pages 367–392. Springer, 1995.
24. E. Kinber, C. Smith, M. Velauthapillai and R. Wiehagen. On learning multiple concepts in parallel. *Journal of Computer and System Sciences*, 50:41–52, 1995.
25. S. Kurtz and C. Smith. A refutation of Bārzdiņš' conjecture. In K. P. Jantke, editor, *Analogical and Inductive Inference, Proceedings of the Second International Workshop (AII '89)*, volume 397 of *Lecture Notes in Artificial Intelligence*, pages 171–176. Springer-Verlag, 1989.
26. S. Lange and T. Zeugmann. Types of monotonic language learning and their characterization. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 377–390. ACM Press, 1992.
27. Y. Mukouchi. Characterization of finite identification. In K. Jantke, editor, *Analogical and Inductive Inference, Proceedings of the Third International Workshop*, pages 260–267, 1992.
28. D. Osherson, M. Stob and S. Weinstein. Learning strategies. *Information and Control*, 53:32–51, 1982.
29. D. Osherson, M. Stob and S. Weinstein. *Systems that Learn: An Introduction to Learning Theory for Cognitive and Computer Scientists.* MIT Press, 1986.
30. M. Ott and F. Stephan. Avoiding coding tricks by hyperrobust learning. *Theoretical Computer Science*, 284(1):161–180, 2002.
31. G. Schäfer-Richter. *Über Eingabeabhängigkeit und Komplexität von Inferenzstrategien.* PhD thesis, RWTH Aachen, 1984.
32. T. Shinohara. Polynomial time inference of extended regular pattern languages. In *RIMS Symposia on Software Science and Engineering, Kyoto, Japan*, volume 147 of *Lecture Notes in Computer Science*, pages 115–127. Springer-Verlag, 1982.
33. R. Wiehagen. Limes-Erkennung rekursiver Funktionen durch spezielle Strategien. *Journal of Information Processing and Cybernetics (EIK)*, 12(1–2):93–99, 1976.