8

Data Complexity Issues in Grammatical Inference

Colin de la Higuera

Summary. Grammatical inference (also known as grammar induction) is a field transversal to a number of research areas including machine learning, formal language theory, syntactic and structural pattern recognition, computational linguistics, computational biology, and speech recognition. Specificities of the problems that are studied include those related to data complexity. We argue that there are three levels at which data complexity for grammatical inference can be studied: at the first (inner) level the data can be strings, trees, or graphs; these are nontrivial objects on which topologies may not always be easy to manage. A second level is concerned with the classes and the representations of the classes used for classification; formal language theory provides us with an elegant setting based on rewriting systems and recursivity, but which is not easy to work with for classification or learning tasks. The combinatoric problems usually attached to these tasks prove to be indeed difficult. The third level relates the objects to the classes. Membership may be problematic, and this is even more the case when approximations (of the strings or the languages) are used, for instance in a noisy setting. We argue that the main difficulties arise from the fact that the structural definitions of the languages and the topological measures do not match.

8.1 Introduction

8.1.1 The Field

Grammatical inference is transversal to various fields including machine learning, formal language theory, structural and syntactic pattern recognition, computational linguistics, computational biology, and speech recognition.

In a very broad sense, a learner has access to some data that are sequential or structured (strings, words, trees, terms, or even limited forms of graphs) and is asked to return a grammar that should in some way explain these data. The grammar is supposed to be able to generate the data, or recognize it. The learner is at least partially automatic, and is therefore sometimes called an inference machine, or a learning algorithm. The induced (or inferred) grammar can then be used to classify unseen data, compress this data, or provide some suitable model for this data. Typical features of the problem are:

- The data, composed from a finite alphabet; is thus usually discrete, as opposed to numerical; but on the other hand the unbounded length of strings makes the classification tasks harder than with the usual symbolic data.

- The sort of result: a grammar or an automaton, traditional objects from formal language theory studied in computer science. The advantage of these objects is that they are understandable. The classes that are learned are described in an intelligible and sometimes even graphical way, which may not always be the case in pattern recognition or classification. In fields where human experts need to be able to derive new knowledge from what the computer provides, this is undoubtedly a key feature.
- The hardness of even the easiest of problems: Grammatical inference is a field where there are only a few positive theoretical results. Most learning problems are considered intractable and are even used as examples of hard tasks in machine learning.
- The variety of potential applications: Data in many fields is today more and more complex and structured, and therefore, techniques that concentrate on structural data are of increasing importance.

There are a number of ways of addressing the different problems in grammatical inference: searching for new algorithms, broadening the class of data for which existing techniques work, better understanding the boundaries between those problems that one can solve and those that correspond to classes that are not learnable, and the application of known techniques to new problems.

8.1.2 The Literature and the Community

Grammatical inference scientists belong to a number of larger communities: machine learning (with special emphasis on inductive inference), computational linguistics, and pattern recognition (within the structural and syntactic subgroup). There is a specific conference the *International Colloquium on Grammatical Inference* (ICGI) devoted to the subject, within whose proceedings it is possible to find a number of technical papers. These conferences have been held at Alicante, Spain [13], Montpellier, France [59], Ames, Iowa [37], Lisbon [24], and Amsterdam [1]. The web page of the grammatical inference community [85], and those of related communities can be used to find most of the papers in the field. The *Computational Learning Theory* (COLT) web page [46] or the *Algorithmic Learning Theory* (ALT) web page [92] can provide good lists of papers with the machine learning perspective. Important key papers setting the first definitions and providing important heuristics are those by Fu [27] and Fu and Booth [28]. The structural and syntactic pattern recognition approaches can be found, for instance, in Miclet's book [57] and in the survey by Bunke and Sanfeliu [10], with special interest in Miclet's chapter [58].

Surveys or introductions to the subject have been published by Lee [51], Sakakibara [76], Honavar and de la Higuera [36], and de la Higuera [20].

Books on formal languages by Harrison [35] and by Hopcroft and Ullman [38] give most of the elementary definitions and results needed for the language theoretical background to grammatical inference. Parsing issues are discussed in Aho and Ullman's textbook [2]. On machine learning the books by Mitchell [60], Natarajan [63], and Kearns and Vazirani [42] all give elements that are of use to derive grammar induction results. Another place where structural pattern recognition issues are discussed is Gonzalez and Thomason's book [33]. An early book with many important mathematical results on the subject of automata inference is that by Trakhtenbrot and Barzdin [84].

8.1.3 Organization of This Chapter

The question under study is, What are the *data complexity* issues involved in grammatical inference? The first problem is to discuss the different meanings that can be linked with these

terms. The first idea is that the actual data (strings and trees) have a specific complexity, independently of the concept classes that are involved. A second point of view may be that it concerns the classes themselves (languages) which are non trivial mathematical objects when dealing with grammatical inference: languages defined by grammars and automata. And a third point of view appears when we consider simultaneously the strings and the languages.

Before discussing these issues, we will survey some of the main definitions and results concerning grammar induction.

In section 8.2 we define the objects we intend to learn from. These are especially strings and trees, but other more complex objects have sometimes been studied, such as infinite strings or graphs. We also visit (section 8.3) the concept classes that are used in typical classification tasks. In grammatical inference these are languages that are defined by automata or grammars in the case of languages as sets of strings (section 8.3.1) or distributions (section 8.3.2). In section 8.4 we survey the topological issues regarding both strings (8.4.1) and languages (8.4.2). We deal in section 8.5 with the problems that arise from these definitions and that concern what can be grouped under the heading of "data complexity problems" for grammatical inference: problems concerning the data from which one wants to learn (section 8.5.1), problems relating to language theory and the specificity of grammatical definitions (section 8.5.2) and questions arising from the specific relationship between the data and the classes, i.e., between the strings and the languages (section 8.5.3). We draw conclusions in section 8.6.

8.2 Strings and Trees: from Which We Learn

Let Σ be a finite alphabet and $\Sigma^\star$ the (infinite) set of all strings that can be built from Σ, including the empty string denoted by λ.

By convention, symbols in Σ are denoted by letters from the beginning of the alphabet $(a, b, c, ...)$, and strings in $\Sigma^\star$ are denoted by the end of the alphabet letters $(..., x, y, z)$. The length of a string $x \in \Sigma^\star$ is written $|x|$. The set of all strings of length n (less than, at most n) is denoted by Σ^n ($\Sigma^{<n}$, $\Sigma^{\leq n}$). A substring of x from position i to position j is denoted as $x_i \dots x_j$. A substring $x_i \dots x_j$ with $j < i$ is the empty string λ.

Strings are totally ordered according to the hierarchical order, i.e., if x and y belong to $\Sigma^\star$, $x \leq y \Leftrightarrow |x| < |y|$ or $|x| = |y| \wedge x \leq_{\text{lex}} y$. The first strings, according to the hierarchical order (also sometimes called the length-lexicographic or length-lex order), with $\Sigma = \{a, b\}$ are $\{\lambda, a, b, aa, ab, ba, bb, aaa, \dots\}$.

Extensions of strings include:

- Trees or terms that are recursively defined from a given ranked alphabet $F = F_0 \cup F_1 ... F_k$ as $\forall c \in F_0, c \in T$ and $t_1, ..t_k \in T, f \in F_k \implies f(t_1, .., t_k) \in T$. Alternative definitions exist where the trees are unranked (the same symbol may belong to various alphabets), or unordered (the order on the subtrees is not important). Learning tree automata and grammars are of increasing importance due to the interest in tasks involving structured information (for instance XML files).
- Graphs and hypergraphs are even more complex to define by generative means, and grammars that produce them have been studied in specific fields only. No positive learning result is known, at least for nontrivial classes.
- Infinite strings are used to model situations in reactive systems. Grammatical inference has been used on such data [18, 55, 79].

8.3 Languages: What We Want to Learn

When considering languages, two points of view leading to two different settings have been studied:

- A *language* is a subset of $\Sigma^\star$. In this case problems are those of membership (Does a string belong or not to a language?) and the related issue of parsing strings, or that of the equivalence of grammars (Do they define or generate the same language?).
- A (stochastic) language can also be a distribution of probabilities over $\Sigma^\star$. In this setting the question is to use automata or grammars that assign a higher probability to the more probable strings and a lower (or null) one to the others.

8.3.1 Languages

Formal language theory has been studied consistently over the past 50 years. The usual definitions and results can be found in references [35, 38]. Languages are sets of strings defined through generative (grammars) or recognition (automata) processes. We recall here two of the simpler but also more important definitions.

Regular Languages

Definition 1. *A deterministic finite automaton (DFA) is a quintuple $A = \langle Q, \Sigma, \delta, F, q_0\rangle$ where Σ is an alphabet, Q is a finite set of states, $q_0 \in Q$ is the initial state, $\delta : Q \times \Sigma \to Q$ is a transition function, and $F \subseteq Q$ is a set of marked states, called the final states.*

It is usual to recursively extend δ to Σ^*: $\delta(q, \lambda) = q$ and $\delta(q, a.w) = \delta(\delta(q, a), w)$ for all $q \in Q, a \in \Sigma, w \in \Sigma^*$. Let $\mathcal{L}(A)$ denote the language recognized by automaton A:

$$\mathcal{L}(A) = \{w \in \Sigma^* \mid \delta(q_0, w) \in F\}$$

An alternative definition considers that the automaton may be nondeterministic [$\delta(q, a)$ may have various values]. It is well known that the languages recognized by DFA form the family of regular languages. This class is considered as a borderline case for grammatical inference [19] in the sense that DFA are learnable (in different senses), whereas slightly more complex objects are not. Also, because the class is known to be the first level of the Chomsky hierarchy, considerable attention has been given to the problem of learning it [23, 65, 66, 68].

There are alternative ways of defining regular languages; rational expressions, regular grammars, or nondeterministic finite automata are some of these, but they all lead to learning problems that are more complex than when considering DFA.

Context-Free Languages

The second level of the Chomsky hierarchy is concerned with context-free languages and grammars:

Definition 2 (Context-Free Grammar). *A* Context-Free Grammar (CFG) $G = (\Sigma, V, P, S)$ *is a quadruple where Σ is a finite alphabet (of terminal symbols), V is a finite alphabet (of variables or nonterminals), $P \subset V \times (\Sigma \cup V)^*$ is a finite set of production rules, and $S \in V$ is the axiom (start symbol).*

We denote $uTv \rightarrow uwv$ when $(T, w) \in P$. $\stackrel{*}{\rightarrow}$ is the reflexive and transitive closure of $\rightarrow$. If there exists $u_0, \ldots, u_k$ such that $u_0 \rightarrow \cdots \rightarrow u_k$, we write $u_0 \stackrel{k}{\rightarrow} u_k$. We denote by $L_G(T)$ the language $\{w \in \Sigma^* : T \stackrel{*}{\rightarrow} w\}$. Two grammars are equivalent if they generate the same language. A language is context-free if it can be generated by a context-free grammar.

Learning context-free grammars has proved to be a much harder task than learning DFA. Serious efforts in the past few years have included the following approaches:

- Subclasses of linear languages: linearity is considered by several authors to be a necessary condition for learning to be possible. There has been active and successful research of even linear grammars [44, 54, 80, 81, 82].
- Learning from structured data: learning tree automata [26, 34, 43], or context-free grammars from bracketed data [74, 75], allows to obtain better results, either with queries, regular distributions [15, 45, 72], or negative information [29]. This has also led to different studies concerning the probability estimation of such grammars [11, 50].
- Heuristics based on genetic algorithms [40, 77, 78], lattice exploring [31, 86], or simplicity bias [49].

Membership Issues

The problem of deciding if a given string belongs or not to a language is the first problem one has to deal with. In both cases (for regular languages defined by DFA and context-free languages defined by CFG) the problem is tractable: parsing can take place in linear time with a DFA and in cubic time with a CFG.

Other language formalisms may exist where parsing is not an easy problem. For instance, parsing with pattern languages [5, 25] is an intractable problem.

Equivalence Issues

Two grammars or automata are equivalent if they generate or recognize the same language. If equivalence of DFA is easy to check (thanks to the existence of a nice canonical minimal form), this is not the case if the regular languages are defined by other formalisms such as regular expressions of nondeterministic finite automata. In the case of context-free grammars the same holds: equivalence is undecidable.

This has direct consequences for the learning tasks, as is proved in [19].

8.3.2 Stochastic Languages

Stochastic languages and mechanisms allowing their generation are described in a number of articles and books [39, 50, 51, 56, 64, 67, 71, 87, 88]. We describe only the key ideas of stochastic languages and automata here.

A *stochastic language* $\mathcal{D}$ is a probability distribution over $\Sigma^\star$.

The probability of a string $x \in \Sigma^\star$ under the distribution $\mathcal{D}$ is denoted as $Pr_{\mathcal{D}}(x)$ and must verify $\sum_{x \in \Sigma^\star} Pr_{\mathcal{D}}(x) = 1$. If the distribution is modeled by some syntactic machine $\mathcal{A}$, the probability of x according to the probability distribution defined by $\mathcal{A}$ is denoted $Pr_{\mathcal{A}}(x)$. The distribution modeled by a machine $\mathcal{A}$ will be denoted $\mathcal{D}_{\mathcal{A}}$ and simplified to $\mathcal{D}$ in a non-ambiguous context.

If L is a language (included in $\Sigma^\star$) and $\mathcal{D}$ a distribution over $\Sigma^\star$, $Pr_{\mathcal{D}}(L) = \sum_{x \in L} Pr_{\mathcal{D}}(x)$.

Two distributions $\mathcal{D}$ and $\mathcal{D}'$ are equal (denoted by $\mathcal{D} = \mathcal{D}'$) if $\forall w \in \Sigma^\star : Pr_{\mathcal{D}}(w) = Pr_{\mathcal{D}'}(w)$.

A *sample* S is a multiset of strings; as a sample is usually built through sampling, one string may appear more than once. When considering the sample size, we note the difference between notation $|S|$, which indicates the number of (not necessarily different) strings in S, and $\|S\|$, which is the total sum of lengths of the strings in S. We also write $x \in S$ to indicate that string x is represented in the sample.

Probabilistic Automata and Languages

Probabilistic languages are generated by probabilistic automata.

Definition 3. *A PFA is a tuple $\mathcal{A}$=$\langle Q_{\mathcal{A}}, \Sigma, \delta_{\mathcal{A}}, I_{\mathcal{A}}, F_{\mathcal{A}}, P_{\mathcal{A}}\rangle$, where:*

- *$Q_{\mathcal{A}}$ is a finite set of* states*;*
- *Σ is the alphabet;*
- *$\delta_{\mathcal{A}} \subseteq Q_{\mathcal{A}} \times \Sigma \times Q_{\mathcal{A}}$ is a* set of transitions*;*
- *$I_{\mathcal{A}} : Q_{\mathcal{A}} \to \mathbb{Q}^+$ (initial-state probabilities);*
- *$P_{\mathcal{A}} : \delta_{\mathcal{A}} \to \mathbb{Q}^+$ (transition probabilities);*
- *$F_{\mathcal{A}} : Q_{\mathcal{A}} \to \mathbb{Q}^+$ (final-state probabilities).*

$I_{\mathcal{A}}$, $P_{\mathcal{A}}$ and $F_{\mathcal{A}}$ are functions such that:

$$\sum_{q \in Q_{\mathcal{A}}} I_{\mathcal{A}}(q) = 1,$$

and

$$\forall q \in Q_{\mathcal{A}},\ F_{\mathcal{A}}(q) + \sum_{a \in \Sigma,\ q' \in Q_{\mathcal{A}}} P_{\mathcal{A}}(q, a, q') = 1.$$

$P_{\mathcal{A}}$ is extended with $P_{\mathcal{A}}(q, a, q') = 0$ for all $(q, a, q') \notin \delta_{\mathcal{A}}$. Also, the subscript $_{\mathcal{A}}$ is dropped when there is no ambiguity.

The above automata definition corresponds to models that are *generative* in nature. This is in contrast with the standard definition of automata in the conventional (nonprobabilistic) formal language theory, where strings are generated by *grammars* while the automata are the *accepting* devices. From a probabilistic point of view, the process of (randomly) accepting a *given* string is essentially different from the process of generating a (random) string. Probabilistic acceptors are defined in Fu [27], but they have not been as popular in both syntactic pattern recognition and formal language theory.

Figure 8.1 shows a *graphical representation* of a PFA with four states, $Q = \{q_0, q_1, q_2, q_3\}$, only one initial state, q_0, and a four-symbol alphabet, $\Sigma = \{a, b, c, d\}$. The rational numbers in the states and in the arrows are the final state and the transition probabilities, respectively.

Deterministic Probabilistic Finite-State Automata (DPFA)

Definition 4. *A PFA $\mathcal{A} = \langle Q, \Sigma, \delta, I, F, P\rangle$ is a $DPFA$, if:*

- *$\exists q_0 \in Q$ (*initial state*), such that $I(q_0) = 1$;*
- *$\forall q \in Q,\ \forall a \in \Sigma,\ \ |\{q' : (q, a, q') \in \delta\}| \leq 1$.*

In a $DPFA$, a transition (q, a, q') is completely defined by q and a, and a $DPFA$ can be more simply denoted by $\langle Q, \Sigma, \delta, q_0, F, P\rangle$.

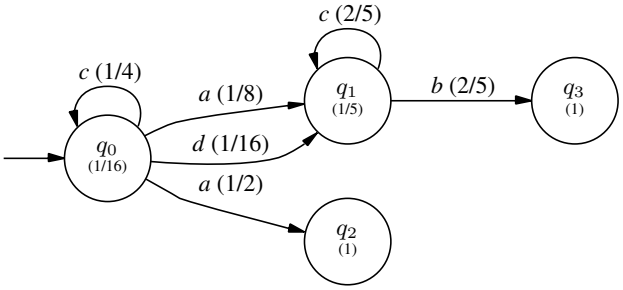

Fig. 8.1. Graphical representation of a PFA.

Distribution Modeled by a PFA

PFA generate strings of finite length. Given a PFA $\mathcal{A}$, a string is generating by randomly choosing (with respect to a distribution I over the initial states) one state q_0 in Q as the initial state. Call q this state. Then iteratively decide whether to *halt*, with probability $F(q)$, or to produce a *move* (q, a, q') with probability $P(q, a, q')$ where $a \in \Sigma$ and $q' \in Q$ in which case a symbol a is emitted and the current state is set to q'.

It should be noticed that this generative process facilitates only generating a string but not computing the probability of a given string, which can be generated in different ways or *paths*. Dynamic programming techniques are used for this computation.

If $\sum_x Pr_{\mathcal{A}}(x) = 1$, then $\mathcal{A}$ defines a distribution $\mathcal{D}$ on $\Sigma^\star$.

Definition 5. *A distribution is* regular *if it can be generated by some PFA.*

An alternative definition could be used: a regular distribution is a probabilistic distribution on a regular language. However, we do not assume this definition because it would present the following problem: there would exist regular distributions that could not be generated by any PFA. This result can be easily derived from Wetherell [91].

Definition 6. *A distribution is* regular deterministic *if it can be generated by some $DPFA$.*

Definition 7. *Two PFAs are* equivalent *if they generate the same distribution.*

From the definition of PFA and $DPFA$ the following hierarchy follows:

Proposition 1. *A regular deterministic distribution is also a regular distribution.*

We do not provide here the definition of context-free stochastic (or probabilistic) grammars, which results from a combination of the definitions of context-free grammars and that of stochastic automata. These have been studied in a variety of works, including [16, 21, 39, 50].

Questions Relating to Stochastic Languages

Following the Chomsky hierarchy, it is possible to extend the previous definitions to cope with stochastic tree languages defined by stochastic tree automata or stochastic context-free grammars.

Parsing issues have also been well studied. If the deterministic case is easy to solve, a number of dynamic programming techniques have been used for the nondeterministic setting.

Equivalence of PFA is decidable, but the exact complexity is not known. More than the question of equivalence, the one of the *closeness* (How close is one distribution to another?) is a hard question, to which only partial answers have been given [62].

8.4 Topological Issues

The way the set of instances or of possible examples is organized depends on the topology that is used on this space. Whereas metrics over vectors and numerical representations are fairly well known and obey mathematically well-studied laws, things get harder when dealing with structured objects such as strings or trees.

Distances can be used to organize the set of examples, but also, sometimes, the set of possible classes. We discuss both types of distance measures in this section.

8.4.1 Distances Between Strings

Given two strings x and y, a distance can be computed in several different ways. In a broad sense the possibilities are:

- To compute the minimal number of operations needed to transform x into y. This is the principle of the *edit* or Levenshtein distances [3, 89].
- To compute some similarity measure $S(x, y)$ between the two strings, and then, with an adequate function ($2^{-S(x,y)}$) convert it into a distance.
- To select a set of measurable features over the strings and to use these to associate with each string a vector in $\mathbb{R}^n$. The vectors can then be compared by standard metrics.

In practice all three ideas have their advantages and their disadvantages.

- The edit distance is certainly the most natural form of distances on strings. But the computation of this distance in quadratic in the length of the strings and many reasonable operations (such as searching for the center of a set of strings [22]) can be intractable.
- Similarity measures abound: an easy distance based on this principle consists in computing the length of the longest common prefix and using this as $S(x, y)$. Yet convincing similarity measures have not been produced.
- Finding a finite set of features over strings and using numerical representations is a standard technique: n-grams are a typical way of doing this. But the intrinsic structure of the string is usually lost.

8.4.2 Distances Between Languages

In the nonstochastic setting (when languages are just sets of strings), comparison is done by usual set-theoretic tools. It should be noticed that without a distribution of probabilities over the strings, the number of strings in the symmetric difference between two sets is not a valid indicator, as it can easily be infinite.

The usual setting to study distances between languages is therefore the stochastic one. Defining similarity measures between distributions is the most natural way of comparing them. In tasks involving the learning of PFA or $DPFA$, one wants to measure the quality of the result or of the learning process. When learning takes place from a sample, measuring how far the learned automaton is from the sample can also be done by comparing distributions since a sample can easily be encoded as a $DPFA$.

There are two families of distance measures: those that are true distances, and those that measure a cross-entropy.

Distances Between Distributions

All the definitions hereafter are seen as definitions of distances between distributions over $\Sigma^\star$. In doing so they implicitly define distances between automata, but also between automata and samples, or even between samples.

- The most general family of distances are referred to as the Minkowski distances or distances for the norm L_n. The general definition is as follows:

$$d_n(\mathcal{D}, \mathcal{D}') = \left(\sum_{x \in \Sigma^\star} |Pr_{\mathcal{D}}(x) - Pr_{\mathcal{D}'}(x)|^n \right)^{\frac{1}{n}}.$$

- The following distance is used in [7] (under the name d_* or distance for the L_∞ norm):

$$d_{\max}(\mathcal{D}, \mathcal{D}') = \max_{x \in \Sigma^\star} |Pr_{\mathcal{D}}(x) - Pr_{\mathcal{D}'}(x)|.$$

Entropy-Based Measures

Based on entropy definitions we can use the well-known Kullback-Leibler divergence:

$$d_{KL}(\mathcal{D}, \mathcal{D}') = \sum_{x \in \Sigma^\star} Pr_{\mathcal{D}}(x) \cdot \log \frac{Pr_{\mathcal{D}}(x)}{Pr_{\mathcal{D}'}(x)}.$$

We set in a standard way $0 \log 0 = 0$ and $\frac{0}{0} = 1$.

In the case where some string has a null probability in $\mathcal{D}'$, but not in $\mathcal{D}$, then the Kullback-Leibler divergence is infinite.

Rewriting the Kullback-Leibler divergence as

$$d_{KL}(\mathcal{D}, \mathcal{D}') = \sum_{x \in \Sigma^\star} (Pr_{\mathcal{D}}(x) \cdot \log Pr_{\mathcal{D}}(x) - Pr_{\mathcal{D}}(x) \cdot \log Pr_{\mathcal{D}'}(x)),$$

one can note the first term is the entropy of $\mathcal{D}$, and does not depend on $\mathcal{D}'$ and the second term is the cross-entropy of $\mathcal{D}$ given $\mathcal{D}'$. From the information theory interpretation [17], the first term measures the optimal number of bits needed to encode $\mathcal{D}$ and the second one measures the cost (in number of bits of encoding) of estimating $\mathcal{D}$ using $\mathcal{D}'$.

Computing Distances

The computation of distance d_n and d_{max} has been studied for hidden Markov models in [53], where it is shown that only for even values of n is the computation possible. Similar results hold for $DPFA$ [62]. The computation of d_{KL} is possible in polynomial time for $DPFA$ [12].

8.5 How Data Complexity Can Affect Grammatical Inference

We now turn to the question of data complexity. As discussed earlier, we discuss three aspects of the problem: the one specific to the data, the one concerning the classes, and the third one which interrelates strings and classes.

8.5.1 Problems Concerning the Data

Strings and trees are nontrivial structures. Their typical definitions are recursive, and they allow for complex extensions (infinite strings, unordered trees). For a number of reasons in grammatical inference, some basic operations over strings and trees are needed, and are not always simple:

- Distances have to be computed. In section 8.4.1 we reviewed some of the typical distances that could be used over strings, and noticed that the computation issues were not trivial.
- Finding patterns in strings [3] is a tricky issue also, for which elaborate algorithms have been built.
- Comparing strings (or trees): is a string a substring of another? A number of algorithms have been proposed for these tasks, but time complexity can be high.

All this has consequences when wanting to use traditional pattern analysis techniques (such as a nearest neighbours approach) in an application where the data is described by strings and requires specific heuristics [61, 73].

8.5.2 Problems Concerning the Languages

Formal language theory has been built on the Chomsky hierarchy which defines classes of languages through the restrictions that one can make to the type of grammar that is admitted [2]. Moreover, recursivity is central to this theory, which gives it elegance but also may not help our understanding of it. A typical example of this is the intuitive distance that there can be between a language and a grammar that generates it.

These definitions have another drawback: small changes made on the grammar (or the automaton) are likely to affect, in an important way, the language that is generated. The converse also holds, as modifying in a very small way a language can easily lead to the construction of a new grammar quite unrelated to the first one. The consequences of this is that the concept classes are very sensitive to noise. Even a very small amount of noisy data in the learning process affects what is being induced. Grammatical inference, because of this data complexity issue, is quite incapable of dealing with noisy data. As in practice, noisy data is abundant; this is a real problem for the field.

The question has probably not yet been explored in a systematic way, and there is still much to argue about this. The point we raise here is simply that (and in contrast to other fields in pattern recognition or machine learning) the topological theories over strings and the formal language theories do not match.

Knowing if two representations of classes are equivalent is a central question both in formal language theory and in learning. Indeed if the problem is intractable, then one should not expect learning to be feasible:

- This may mean that the characteristic sets needed to learn are of a size that one cannot expect to reach in reasonable time [19].
- This also means that learning from membership queries only is not tractable. If not, one could use both grammars as black boxes and learn independently from each, and at the end compare the results.

8.5.3 Problems Concerning the Relationship Between Strings and Languages

When considering strings on their own, we have a problem with the complexity of the data. When considering the classes and their representations, we have another. What happens when we consider both issues together is that new hard problems arise. We mention some of them now.

Negative Results Concerning the Combinatorics

A typical way of considering a classification task from both examples and counter-examples is to find some representation of a class consistent with the observed data. In grammatical inference, in general, this is insufficient, as any finite set of strings is a regular or a context-free language. In that case, theory teaches us that it is a good policy to find a simplest consistent class. This obeys the Occam's razor principle.

Gold [32] proves that the problem of finding the smallest DFA consistent with a given set of strings is NP-hard. Angluin [4] proves that this is the case even when the target automaton only has two states (this peculiar result is quoted by Pitt and Warmuth [70]), or even when only a very small fraction of all the strings up to length n, where n is the size of the target, is absent. Trakhtenbrot and Barzdin [84] had shown previously that when all such strings were present the problem was tractable.

There are several negative consequences of these combinatorical results. This is not enough to obtain a direct proof that learning in polynomial time is impossible, but Angluin proves the hardness of the task even using *membership queries* [6] (a string can be proposed to the oracle who must answer if the string belongs or not to the target language) or *equivalence queries* [8] (a grammar is proposed to the oracle, who answers yes if the hypothesis is equivalent to the target, and provides a counterexample if not). In the second case Angluin introduced the combinatorial notion of *approximate fingerprints* of independent interest, which correspond to a subset of hypotheses out of which only a small fraction can be excluded, given any counterexample, resulting in the necessity of using an exponential number of equivalence queries to isolate a single hypothesis. Gavaldà [30] studies this notion with care. Pitt [68] uses this result to prove the intractability of the task of identifying DFA with a polynomial number of mind changes only (i.e., the number of times the online learning algorithm changes its hypothesis should be polynomial in the size of the target DFA). The problem is proved to be hard [69], and by typical reduction techniques [90] is proved complete (hardest) in its class. But Pitt's model may be itself too demanding, as it is closely linked with Littlestone's learning model [52].

Problems with Approximation

The combinatorical results above imply that finding the exact solution is too hard, but there could be hope for approximately learning. The probably approximately correct (PAC) paradigm has been widely used in machine learning to provide a theoretical setting for this. But even for the case of DFA, most results are negative, Kearns and Valiant [41] linked the difficulty of learning DFA with that of solving cryptographic problems believed to be intractable (a nice proof is published in Kearns and Vazirani's book [42]).

Another point of view on the hardness of approximation can be seen through the different competitions that have been organized for grammatical inference. In 1997, the ABBADINGO competition [47] raised interest in the problem of DFA inference, and although a neural

network technique seemed to do well, at the end, an *evidence-driven* technique developed by Price, based on classical state merging, won (Lang et al., [48]). The idea was to try different merges but keep the one that had the highest score. A (cheap) alternative to evidence-driven heuristics is data-driven heuristics (de la Higuera et al., [23]), where the idea is to try merging those states through which most information is known. Some problems from that competition are still open.

Learning with Noise

As a consequence of the above observations, learning in a noisy setting is very hard. Whereas in other areas of machine learning and pattern recognition there are many methods and results allowing us to work on cases where the data may be imperfect, this is not the case in grammatical inference. One of the few hopes of doing anything of use in this case is to learn stochastic automata [14, 83].

8.6 Conclusion

Grammatical inference is faced with a number of intrinsic difficulties that differentiate the field from other fields in pattern recognition or machine learning. These correspond to the specific recursive unbounded structure of strings and trees, the special way languages are defined, which is not some extension of well-known classes over vectors, and the combined complexity of strings and languages when considering parsing issues.

When faced with these hardness questions a number of answers seem possible:

- Modifying the internal structure of the data offers only limited possibilities. Windowing the data or associating to strings a finite number of characteristics has been done, but there is necessarily a price to pay in terms of representation;
- Finding novel ways of defining languages that do not obey to the Chomsky hierarchy would certainly be a good choice; for instance, different forms of *pattern languages* [5, 9, 25] have been studied and in certain areas, such as computational biology, are indeed of use.
- Finally the inherent difficulties should not force researchers to abandon the idea of learning languages from strings of trees. The successes of grammatical inference show that there still is a lot of room for new profound and useful results.

Acknowledgments

The author is grateful to Rafael Carrasco, Francisco Casacuberta, Rémi Eyraud, Jean-Christophe Janodet, Thierry Murgue, Jose Oncina, Franck Thollard, and Enrique Vidal, for discussions and previous related work that fostered the ideas in this chapter.

References

[1] P. Adriaans, H. Fernau, M. van Zaannen, eds. *Grammatical Inference: Algorithms and Applications, Proceedings of ICGI '02*, volume 2484 of *LNAI*. Berlin, Heidelberg: Springer-Verlag, 2002.

[2] A. Aho, J. D. Ullman. *The Theory of Parsing, Translation and Compiling, Vol 1: Parsing*. Englewood Cliffs, NJ: Prentice-Hall, 1972.
[3] A .V. Aho. Algorithms for Finding Patterns in Strings. *Handbook of Theoretical Computer Science*, pages 290–300. Amsterdam: Elsevier, 1990.
[4] D. Angluin. On the complexity of minimum inference of regular sets. *Information and Control*, 39, 337–350, 1978.
[5] D. Angluin. Finding patterns common to a set of strings. In *Conference record of the eleventh annual ACM Symposium on Theory of Computing*, pages 130–141. New York: ACM Press, 1979.
[6] D. Angluin. Queries and concept learning. *Machine Learning Journal*, 2, 319–342, 1987.
[7] D. Angluin. Identifying languages from stochastic examples. Technical Report YALEU/DCS/RR-614, Yale University, March 1988.
[8] D. Angluin. Negative results for equivalence queries. *Machine Learning Journal*, 5, 121–150, 1990.
[9] A. Brazma, I. Jonassen, J. Vilo, E. Ukkonen. Pattern discovery in biosequences. In Honavar and Slutski [37], pages 257–270.
[10] H. Bunke, A. Sanfeliu, eds. *Syntactic and Structural Pattern Recognition, Theory and Applications*, volume 7 of *Series in Computer Science*. Singapore: World Scientific, 1990.
[11] J. Calera-Rubio, R.C. Carrasco. Computing the relative entropy between regular tree languages. *Information Processing Letters*, 68(6), 283–289, 1998.
[12] R.C. Carrasco. Accurate computation of the relative entropy between stochastic regular grammars. *RAIRO (Theoretical Informatics and Applications)*, 31(5), 437–444, 1997.
[13] R.C. Carrasco, J. Oncina, eds. *Grammatical Inference and Applications, Proceedings of ICGI '94*, number 862 in LNAI, Berlin: Springer, 1994.
[14] R.C. Carrasco, J. Oncina. Learning stochastic regular grammars by means of a state merging method. In ICGI'94 [13], pages 139–150.
[15] R.C. Carrasco, J. Oncina, J. Calera-Rubio. Stochastic inference of regular tree languages. *Machine Learning Journal*, 44(1), 185–197, 2001.
[16] Z. Chi, S. Geman. Estimation of probabilistic context-free grammars. *Computational Linguistics*, 24(2), 298–305, 1998.
[17] T. Cover, J. Thomas. *Elements of Information Theory.* New York: John Wiley, 1991.
[18] C. de la Higuera, J-C. Janodet. Inference of ω-languages from prefixes. *Theoretical Computer Science*, 313(2), 295–312, 2004.
[19] C. de la Higuera. Characteristic sets for polynomial grammatical inference. *Machine Learning Journal*, 27, 125–138, 1997.
[20] C. de la Higuera. Current trends in grammatical inference. In F.J. Ferri et al., eds. *Advances in Pattern Recognition, Joint IAPR International Workshops SSPR+SPR 2000*, volume 1876 of *LNCS*, pages 28–31. New York: Springer-Verlag, 2000.
[21] C. de la Higuera, P. Adriaans, M. van Zaanen, J. Oncina, eds. *Proceedings of the Workshop and Tutorial on Learning Context-free grammars*. ISBN 953-6690-39-X, 2003.
[22] C. de la Higuera, F. Casacuberta. Topology of strings: median string is NP-complete. *Theoretical Computer Science*, 230, 39–48, 2000.
[23] C. de la Higuera, J. Oncina, E. Vidal. Identification of DFA: data-dependent versus data-independent algorithm. In Miclet and de la Higuera [59], pages 313–325.
[24] A. de Oliveira, ed. *Grammatical Inference: Algorithms and Applications, Proceedings of ICGI '00*, volume 1891 of *LNAI*, Berlin: Springer, 2000.

[25] T. Erlebach, P. Rossmanith, H. Stadtherr, A. Steger, T. Zeugmann. Learning one-variable pattern languages very efficiently on average, in parallel, and by asking queries. In M. Li, A. Maruoka, eds. *Proceedings of ALT '97*, volume 1316 of *LNCS*, pages 260–276, Berlin: Springer, 1997.

[26] H. Fernau. Learning tree languages from text. In J. Kivinen, R.H. Sloan, eds. *Proceedings of COLT 2002*, number 2375 in LNAI, pages 153–168, Berlin: Springer, 2002.

[27] K.S. Fu. *Syntactic Methods in Pattern Recognition*. New York: Academic Press, 1974.

[28] K.S. Fu, T.L. Booth. Grammatical inference: Introduction and survey. Part I and II. *IEEE Transactions on Syst. Man. and Cybern.*, 5, 59–72, 409–423, 1975.

[29] P. García, J. Oncina. Inference of recognizable tree sets. Technical Report DSIC-II/47/93, Departamento de Lenguajes y Sistemas Informáticos, Universidad Politécnica de Valencia, Spain, 1993.

[30] R. Gavaldà. On the power of equivalence queries. In *Proceedings of the 1st European Conference on Computational Learning Theory*, volume 53 of *The Institute of Mathematics and its Applications Conference Series, new series*, pages 193–203. Oxford: Oxford University Press, 1993.

[31] J.Y. Giordano. Inference of context-free grammars by enumeration: Structural containment as an ordering bias. In Carrasco and Oncina [13], pages 212–221.

[32] E.M. Gold. Complexity of automaton identification from given data. *Information and Control*, 37, 302–320, 1978.

[33] R. Gonzalez and M. Thomason. *Syntactic Pattern Recognition: an Introduction*. Reading MA: Addison-Wesley, 1978.

[34] A. Habrard, M. Bernard, F. Jacquenet. Generalized stochastic tree automata for multi-relational data mining. In Adriaans et al. [1], pages 120–133.

[35] M. H. Harrison. *Introduction to Formal Language Theory*. Reading, MA: Addison-Wesley, 1978.

[36] V. Honavar, C. de la Higuera. Introduction. *Machine Learning Journal*, 44(1), 5–7, 2001.

[37] V. Honavar, G. Slutski, eds. *Grammatical Inference, Proceedings of ICGI '98*, number 1433 in LNAI, Berlin: Springer-Verlag, 1998.

[38] J.E. Hopcroft, J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Reading, MA: Addison-Wesley, 1979.

[39] A. Jagota, R.B. Lyngsø, C.N.S. Pedersen. Comparing a hidden Markov model and a stochastic context-free grammar. In *Proceedings of WABI '01*, number 2149 in LNCS, pages 69–74, Berlin: Springer-Verlag, 2001.

[40] T. Kammeyer, R.K. Belew. Stochastic context-free grammar induction with a genetic algorithm using local search. In R.K. Belew, M. Vose, eds. *Foundations of Genetic Algorithms IV*, San Mateo, CA: Morgan Kaufmann, 1996.

[41] M. Kearns, L. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. In *21st ACM Symposium on Theory of Computing*, pages 433–444, 1989.

[42] M. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory*. Cambridge, MA: MIT press, 1994.

[43] T. Knuutila, M. Steinby. Inference of tree languages from a finite sample: an algebraic approach. *Theoretical Computer Science*, 129, 337–367, 1994.

[44] T. Koshiba, E. Mäkinen, Y. Takada. Inferring pure context-free languages from positive data. *Acta Cybernetica*, 14(3), 469–477, 2000.

[45] S.C. Kremer. Parallel stochastic grammar induction. In *Proceedings of the 1997 International Conference on Neural Networks (ICNN '97)*, volume I, pages 612–616, 1997.

[46] S.S. Kwerk. Colt: Computational learning theory. http://www.learningtheory.org, 1999.

[47] K. Lang, B.A. Pearlmutter. The Abbadingo one DFA learning competition, 1997.
[48] K.J. Lang, B.A. Pearlmutter, R.A. Price. Results of the Abbadingo one DFA learning competition and a new evidence-driven state merging algorithm. In Honavar and Slutski [37], pages 1–12.
[49] P. Langley, S. Stromsten. Learning context-free grammars with a simplicity bias. In *Proceedings of ECML 2000, 11th European Conference on Machine Learning,*, volume 1810 of *LNCS*, pages 220–228. New York: Springer-Verlag, 2000.
[50] K. Lari, S.J. Young. The estimation of stochastic context free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4, 35–56, 1990.
[51] S. Lee. Learning of context-free languages: A survey of the literature. Technical Report TR-12-96. Cambridge, MA: Center for Research in Computing Technology, Harvard University Press, 1996.
[52] N. Littlestone. Learning quickly when irrelevant attributes abound: a new linear threshold. *Machine Learning Journal*, 2, 285–318, 1987.
[53] R.B. Lyngsø, C.N.S. Pedersen. The consensus string problem and the complexity of comparing hidden markov models. *Journal of Computing and System Science*, 65(3), 545–569, 2002.
[54] E. Mäkinen. A note on the grammatical inference problem for even linear languages. *Fundamenta Informaticae*, 25(2), 175–182, 1996.
[55] O. Maler, A. Pnueli. On the learnability of infinitary regular sets. In *Proceedings of COLT*, pages 128–136, San Mateo, CA: Morgan Kaufmann, 1991.
[56] F. Maryanski, M. G. Thomason. Properties of stochastic syntax-directed translation schemata. *International Journal of Computer and information Science*, 8(2), 89–110, 1979.
[57] L. Miclet. *Structural Methods in Pattern Recognition*. New York: Chapman and Hall, 1986.
[58] L. Miclet. *Syntactic and Structural Pattern Recognition, Theory and Applications*, In *Grammatical Inference*. pages 237–290. Singapore: World Scientific, 1990.
[59] L. Miclet, C. de la Higuera, eds. *Proceedings of ICGI '96*, number 1147 in LNAI, Berlin, Heidelberg: Springer-Verlag, 1996.
[60] T. M. Mitchell. *Machine Learning*. New York: McGraw-Hill, 1997.
[61] F. Moreno-Seco, L. Micó, J. Oncina. A modification of the LAESA algorithm for approximated k-nn classification. *Pattern Recognition Letters*, 24(1-3), 47–53, 2003.
[62] T. Murgue, C. de la Higuera. Distances between distributions: Comparing language models. In A. Fred, T. Caelli, R. Duin, A. Campilho, D. de Ridder, eds. *Structural, Syntactic and Statistical Pattern Recognition, Proceedings of SSPR and SPR 2004*, volume 3138 of *LNCS*, pages 269–277. New York: Springer-Verlag, 2004.
[63] B.L. Natarajan. *Machine Learning: a Theoretical Approach*. San Mateo, CA: Morgan Kaufmann, 1991.
[64] H. Ney. Stochastic grammars and pattern recognition. In P. Laface, R. De Mori, eds. *Proceedings of the NATO Advanced Study Institute*, pages 313–344. New York: Springer-Verlag, 1992.
[65] J. Oncina, P. García. Identifying regular languages in polynomial time. In H. Bunke, ed. *Advances in Structural and Syntactic Pattern Recognition*, volume 5 of *Series in Machine Perception and Artificial Intelligence*, pages 99–108. Singapore: World Scientific, 1992.
[66] R. J. Parekh, C. Nichitiu, V. Honavar. A polynomial time incremental algorithm for learning DFA. In Honavar and Slutski [37], pages 37–49.
[67] A. Paz. *Introduction to Probabilistic Automata*. New York: Academic Press, 1971.

[68] L. Pitt. Inductive inference, DFA's, and computational complexity. In *Analogical and Inductive Inference*, number 397 in LNAI, pages 18–44. Berlin, Heidelberg: Springer-Verlag, 1989.

[69] L. Pitt, M. Warmuth. Reductions among prediction problems: on the difficulty of predicting automata. In *3rd Conference on Structure in Complexity Theory*, pages 60–69, 1988.

[70] L. Pitt, M. Warmuth. The minimum consistent DFA problem cannot be approximated within any polynomial. *Journal of the Association for Computing Machinery*, 40(1), 95–142, 1993.

[71] M.O. Rabin. Probabilistic automata. *Information and Control*, 6, 230–245, 1966.

[72] J.R. Rico-Juan, J. Calera-Rubio, and R.C. Carrasco. Stochastic k-testable tree languages and applications. In Adriaans et al. [1], pages 199–212.

[73] J. R. Rico-Juan, L. Micó. Comparison of AESA and LAESA search algorithms using string and tree-edit-distances. *Pattern Recognition Letters*, 24(9-10), 1417–1426, 2003.

[74] Y. Sakakibara. Learning context-free grammars from structural data in polynomial time. *Theoretical Computer Science*, 76, 223–242, 1990.

[75] Y. Sakakibara. Efficient learning of context-free grammars from positive structural examples. *Information and Computation*, 97, 23–60, 1992.

[76] Y. Sakakibara. Recent advances of grammatical inference. *Theoretical Computer Science*, 185, 15–45, 1997.

[77] Y. Sakakibara, M. Kondo. Ga-based learning of context-free grammars using tabular representations. In *Proceedings of 16th International Conference on Machine Learning (ICML-99)*, pages 354–360, 1999.

[78] Y. Sakakibara, H. Muramatsu. Learning context-free grammars from partially structured examples. In de Oliveira [24], pages 229–240.

[79] A. Saoudi, T. Yokomori. Learning local and recognizable ω-languages and monadic logic programs. In *Proceedings of EUROCOLT*, LNCS, pages 157–169. New York: Springer-Verlag, 1993.

[80] J. M. Sempere, P. García. A characterisation of even linear languages and its application to the learning problem. In Carrasco and Oncina [13], pages 38–44.

[81] Y. Takada. Grammatical inference for even linear languages based on control sets. *Information Processing Letters*, 28(4), 193–199, 1988.

[82] Y. Takada. A hierarchy of language families learnable by regular language learners. In Carrasco and Oncina [13], pages 16–24.

[83] F. Thollard, P. Dupont, C. de la Higuera. Probabilistic DFA inference using Kullback-Leibler divergence and minimality. In *Proc. 17th International Conf. on Machine Learning*, pages 975–982. San Francisco, CA: Morgan Kaufmann, 2000.

[84] B. Trakhtenbrot, Y. Barzdin. *Finite Automata: Behavior and Synthesis*. Amesterdam: North Holland, 1973.

[85] M. van Zaanen. The grammatical inference homepage. http://eurise.univ-st-etienne.fr/gi/gi.html, 2003.

[86] K. Vanlehn, W. Ball. A version space approach to learning context-free grammars. *Machine Learning Journal*, 2, 39–74, 1987.

[87] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, R. C. Carrasco. Probabilistic finite state automata – part I. *Pattern Analysis and Machine Intelligence*, 27(7), 1013–1025, 2005.

[88] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, R. C. Carrasco. Probabilistic finite state automata – part II. *Pattern Analysis and Machine Intelligence*, 27(7), 1026–1039, 2005.

[89] R. Wagner, M. Fisher. The string-to-string correction problem. *Journal of the ACM*, 21, 168–178, 1974.

[90] M. Warmuth. Towards representation independence in pac-learning. In K. P. Jantke, ed. *Proceedings of AII'89*, volume 397 of *LNAI*, pages 78–103. New York: Springer-Verlag, 1989.

[91] C. S. Wetherell. Probabilistic languages: a review and some open questions. *Computing Surveys*, 12(4), 361–379, 1980.

[92] T. Zeugmann. Alt series home page. http://www.tcs.mu-luebeck.de/pages/thomas/WALT/waltn.jhtml, 1999.