Ten Open Problems in Grammatical Inference[⋆]

Colin de la Higuera

Laboratoire Hubert Curien, UMR CNRS 5516
Université Jean Monnet Saint-Etienne, France
`cdlh@univ-st-etienne.fr`,
WWW home page: `http://eurise.univ-st-etienne.fr/~cdlh`

Abstract. We propose 10 different open problems in the field of grammatical inference. In all cases, problems are theoretically oriented but correspond to practical questions. They cover the areas of polynomial learning models, learning from ordered alphabets, learning deterministic POMDPs, learning negotiation processes, learning from context-free background knowledge.

1 Introduction

Results in grammatical inference can usually be of use in several different domains. For instance progress in learning stochastic finite state machines and grammars has occurred because of efforts for computational biology [1, 2], or speech recognition [3], or even document representation [4]. Another example is that of learning transducers where research has taken place in very different fields like wrapper induction [5, 6] or automatic translation [7]. In order for these fields to cross fertilise it can be useful to use theory as a common language. In the theoretical world it is possible to pose problems, and then to try to solve them.

This paper addresses only the first issue. After discussions with practitionneers and synthesis of the questions that correspond to bottlenecks for the use of grammatical inference, we visit here several problems. Each one has its motivations, is given with the main definitions, and the corresponding references. Even if the goal was to make the problems as unambiguous as possible, some require additional definitions (as part of the problem!) before attempting to solve them.

The paper is organised as follows: after giving in section 2 some basic notations, we study the question of polynomial learning (section 3). A number of key definitions have been proposed over the years and hardly no work has been done in order to compare these. We then consider the case where the alphabet is an ordered set. To deal in section 4 with these we introduce ordered automata, and suggest that their learnability could be an interesting question. A small problem of learning regular languages from a context-free background knowledge is

[⋆] This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

proposed in section 5. Solving the question of testing equivalence of regular distributions would enable to better learn stochastic finite state automata. The problem is presented in section 6.

Another line of research is that of active learning. There are in this context 3 problems. The first question is related with the fact that over the past few years the Oracle learning model defined by Angluin has ceased to be a theoretical model (section 7). Today it is of interest to consider Oracles that may make errors, or contradict themselves. How do we learn languages with these unreliable Oracles? The second (section 8) is concerned with learning Partially Observable Markov Decision Processes (POMDPs). These intervene in reinforcement learning and the question of their learnability has hardly been addressed (and certainly not from an inductive inference point of view). A last question is that of negotiation (section 9): two agents have to identify the common language by using their own language to interrogate the other agent, and thus giving away information about their language. How do we learn?

2 Notations and definitions

Strings A *string* w over Σ is a finite sequence $w = a_1 a_2 \dots a_n$ of letters. Let $|w|$ denote the length of w. Letters of Σ will be indicated by $a, b, c, \dots$, strings over Σ by $u, v, \dots, z$, and the empty string by λ.

Let $\Sigma^\star$ be the set of all finite strings over alphabet Σ.

Languages A language is any set of strings, so therefore a subset of $\Sigma^\star$. Operations over languages include: set operations (union, intersection, complement); product $L_1 \cdot L_2 = \{uv : u \in L_1, v \in L_2\}$; powerset $L^0 = \{\lambda\}$ $L^{n+1} = L^n \cdot L$; and star $L^* = \cup_{i \in \mathbb{N}} L^i$. We denote by $\mathcal{L}$ or $\mathcal{A}$ a class of languages.

Automata and regular languages 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, $\delta : Q \times \Sigma \rightarrow Q$ 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 $\mathbb{L}(A)$ denote the language recognized by automaton A: $\mathbb{L}(A) = \{w \in \Sigma^* \mid \delta(q_0, w) \in F\}$.

Other mechanism to define, generate or recognise languages are non deterministic finite state automata, context-free grammars, regular expression,… and are described in textbooks, for instance [8].

Identification in the limit. Let $\mathcal{A}$ be a class of languages. A presentation is an element of $Pres(\mathcal{A})$ which is a set of functions $\mathbb{N} \rightarrow X$ with X a set. In some way these presentations *denote* languages from $\mathcal{A}$, *i.e.* there exists a function $yield : Pres(\mathcal{A}) \rightarrow \mathcal{A}$. If $L = yield(f)$ then we will say that f is a presentation of L.

With this definition one should not think of presentations as text or informant, but in a broader sense as a sequence of informations of any type that hopefully inform us on the language we are to learn.

Typical presentations could be Text, Informant, Prefixes,... Here are some examples of possible presentations:

– Text=$\{f : \mathbb{N} \to \Sigma^* \cup \{\#\} : f(\mathbb{N}) = L$ or $\{\#\}$ where $L \subset \Sigma^*\}$.
– Informant=$\{f : \mathbb{N} \to \Sigma^* \times \{0, 1\} : f(\mathbb{N}) = L \times \{1\} \cup \overline{L} \times \{0\}\}$.

Definition 1. *Let $\mathcal{A}$ be a class of languages and $Pres(\mathcal{A})$ be a type of presentations for $\mathcal{A}$, with associated function yield. The setting is said to be* valid *when given 2 presentations f and g, if their range is equal (i.e. if $f(\mathbb{N})= g(\mathbb{N})$) then $yield(f) = yield(g)$.*

If a setting is not valid, $\mathcal{A}$ is not going to be learnable from $Pres(\mathcal{A})$. Practically, to a language class $\mathcal{A}$ is associated a representation class $R(\mathcal{A})$. The association is done through a *naming* function $\mathbb{L}_{\mathcal{A}}$ which associates to a representation (also called a *grammar*) G a language $L = \mathbb{L}_{\mathcal{A}}(G)$. Two grammars G_1 and G_2 are equivalent when $\mathbb{L}_{\mathcal{A}}(G_1) = \mathbb{L}_{\mathcal{A}}(G_2)$.

Given a presentation f we denote by f_n the set $\{f(j) : j \leq n\}$. Given a presentation f we denote by $f(n) \models G$ (conversely $f(n) \not\models G$) when $f(n)$ is consistent with $\mathbb{L}_{\mathcal{A}}(G)$.

A *learning algorithm* **a** is a program that takes the first n elements of a presentation and returns a representation as output. $\mathbf{a} : \bigcup_{i\in\mathbb{N}}\{f_i\} \to R(\mathcal{A})$. The following definition is directly adapted from [9]:

Definition 2. *We say that $\mathcal{A}$ is learnable from $Pres(\mathcal{A})$ in terms of $R(\mathcal{A})$ if there exists a learning algorithm **a** such that for all $L \in \mathcal{A}$ and for any presentation f of L (belonging to $Pres(\mathcal{A})$), there exists a rank n such that for all $m \geq n$, $\mathbb{L}_{\mathcal{A}}(\boldsymbol{a}(f_m)) = L$.*

In order to be able to study complexity we define sizes as follows:

– $|G|$ is the size of a grammar (number of bits);
– $|f_n| = n + 1$ is the number of items in the first elements of a presentation;
– $\|f_n\|$ is the number of symbols in in the first n+1 elements of a presentation (number of bits);
– $|L| = \min\{|G| : \mathbb{L}(G) = L\}$. The "size" of a language is the size of the smallest grammar of the considered class that can generate it;

3 About polynomial identification in the limit

The question of polynomial learning has been of interest for some time. A first discussion has taken place in [10], with further ideas in [11] and [12]. Original ideas concerning these questions (with a notion of *stochastically polynomial* learning) can be found in [13].

Nevertheless there is no general agreement between the authors in the field about which definitions should be used, neither (with the exception of [14]) have definitions been compared.

Definitions The following are some of the definitions of (some type of) polynomial identification in the limit:

The first definition just states that to produce its next hypothesis the algorithm only requires polynomial time. This definition alone is insufficient as shown in [10].

Definition 3 (Polynomial update time). *An algorithm* $\boldsymbol{a}$ *is said to have* polynomial update time *if there is a polynomial* $p()$ *such that, for every presentation* f *and every integer* n*, constructing* $H_n = \boldsymbol{a}(f_n)$ *requires* $p(\|f_n\|)$ *time.*

Another definition that has allowed more results is given by [11]:

Definition 4 (Polynomial characteristic sets). *An algorithm* $\boldsymbol{a}$ *admits polynomial characteristic sets if there exists a polynomial* $p()$ *such that* $\forall G \in R(\mathcal{A})$ $\exists W \subset X : \|W\| \leq p(|G|) \wedge W \subset f_n \implies \mathbb{L}_{\mathcal{A}}(\boldsymbol{a}(f_n)) = \mathbb{L}_{\mathcal{A}}(G)$.

A notion that has been used in various papers was introduced in [10]:

Definition 5 (Implicit prediction errors). *Given a learning algorithm* $\boldsymbol{a}$ *and a presentation* f*, we say that* $\boldsymbol{a}$ makes an implicit prediction error at time n *if* $f(n) \not\models \boldsymbol{a}(f_{n-1})$.

Let f *be a presentation for* L*, algorithm* $\boldsymbol{a}$ *is said to make a* polynomial number of implicit prediction errors *if there is a polynomial* $p()$ *such that, for each language* L *and each presentation* f *for* L*,* $|\{k \in \mathbb{N} : f(k+1) \not\models \boldsymbol{a}(f_k)\}| \leq p(|L|)$.

A nice alternative to counting the number of errors is that of counting the number of changes of hypothesis one makes. On its own, this is meaningless (why change?), but if combined with identification in the limit the definition makes sense:

Definition 6 (Polynomial mind changes). *Given a learning algorithm* $\boldsymbol{a}$ *and a presentation* f*, we say that* $\boldsymbol{a}$ changes its mind at time n *if* $\boldsymbol{a}(f_n) \neq \boldsymbol{a}(f_{n-1})$.

Let f *be a presentation for* L*, algorithm* $\boldsymbol{a}$ *is said to make a* polynomial number of mind changes *if there is a polynomial* $p()$ *such that, for each language* L *and each presentation* f *for* L*,* $|\{k \in \mathbb{N} : \boldsymbol{a}(f_k) \neq \boldsymbol{a}(f_{k+1})\}| \leq p(|L|)$.

Combining ideas, one gets:

Definition 7 (Yokomori [12]). *An algorithm* $\boldsymbol{a}$ *identifies a class* $\mathcal{A}$ *in the limit in* IPE *polynomial time if:*

- $\boldsymbol{a}$ *identifies* $\mathcal{A}$ *in the limit;*
- $\boldsymbol{a}$ *has polynomial update time;*
- $\boldsymbol{a}$ *makes a polynomial number of implicit prediction errors.*

Note that the first condition is not implied by the two other; in a similar way Pitt [10] introduced a definition where instead of requiring a polynomial number of implicit prediction errors, what is counted is the number of mind changes. Here also, the first condition is not implied by the two other. Generally speaking there are a number of problems related with deciding which definition applies best to which learning setting (text, informant,...). A comparison between these definitions would also be of use: is one definition more general than another? Further, can a polynomial algorithm for one setting be transformed into a polynomial algorithm in another?

If all these questions are interesting, we extract just one that has been puzzling researchers for some time:

Problem 1 *Definition 4 of characteristic sets uses as size of the characteristic sets a measure related to the number of bits needed to encode. Other authors (for instance [6]) propose to use the number of strings. Is this fair? Are there classes of grammars that are not learnable in this context?*

Discussion A clear picture of what polynomial learning means in the identification in the limit setting would help enormously the field. There are today several "schools" each with their definitions. That makes theoretical results difficult to compare.

4 Ordered alphabets

We propose to consider the case where the alphabet is ranked, *i.e.* there is a partial order over the symbols in the alphabet. The situation arises in a number of settings:

- either when the alphabet is naturally ordered as in the case of music [15];
- if the original data is numeric, the normal discretisation loses the proximity/topological characteristics that should help and that are contained in the data [16];
- sometimes the alphabet can consist in subsets of strings in which case we can also have a relation which may be a generalisation or subsumption [17].

Definitions We introduce k-edge deterministic finite state automata.

A ranked alphabet $\langle \Sigma, \leq \rangle$ is an alphabet Σ with a relation $\leq_\Sigma$ which is a partial order (reflexive, antisymmetric and transitive) over Σ.

Example 1. Here are two possible relations:

- $\langle \Sigma, \leq \rangle$ where $\Sigma = \{0, 1, 2, 3\}$ and $0 \leq 1 \leq 2 \leq 3$. This is the case in music or working with numerical data
- $\langle \Sigma, \leq \rangle$ where $\Sigma = \{00, 01, 10, 11\}$ and $00 \leq 01 \leq 11$ and $00 \leq 10 \leq 11$. This means that the automaton may not need to be based on a total order.

Definition 8 (k-edge deterministic finite state automaton). *A* k-edge deterministic finite state automaton (k-edge DFA) *A is a tuple $\langle \Sigma, Q, q_0, F, \delta \rangle$ where Σ is a finite alphabet, Q is a finite set of* states, *$q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, $\delta : Q \times \Sigma \times \Sigma \rightarrow Q$ is the transition function verifying: $\forall q \in Q$, $|\{x, y : \delta(q, x, y) \text{ is defined}\}| \leq k$, and if $\delta(q, a_1, b_1) \neq \delta(q, a_2, b_2)$ then $\{z : a_1 \leq z \leq b_1\} \cap \{z : a_2 \leq z \leq b_2\} = \emptyset$.*

A string x belongs to L if there is a sequence of states $q_0, q_1, ...q_{|x|}$ with $\delta(q_i, a_i, b_i) = q_{i+1}$ and $a_i \leq x_i \leq b_i$. And of course $q_{|x|}$ has to be final.

The extension of δ is as usual: the transition function δ is classically extended to sequences by: $\forall q \in Q$, $\forall w \in \Sigma^*$, $\forall a \in \Sigma$, $\delta(q, aw) = \delta(\delta(q, a), w)$. The language recognised by A, $\mathbb{L}(A)$ is $\{w \in \Sigma^* : \delta(q_0, w) \in F\}$.

Example 2. We represent in Figure 1 a 2-edge automaton. Notice that the same language can also be represented by a 3-edge automaton, but not by a 1-edge automaton. Here, $102 \in L$,

Clearly any k-edge DFA is also a DFA but the converse is not true. Moreover some regular languages cannot be represented by k-edge DFA, for any k. Also, the case where k is the size of the alphabet is of no new interest at all, as it corresponds to normal DFA.

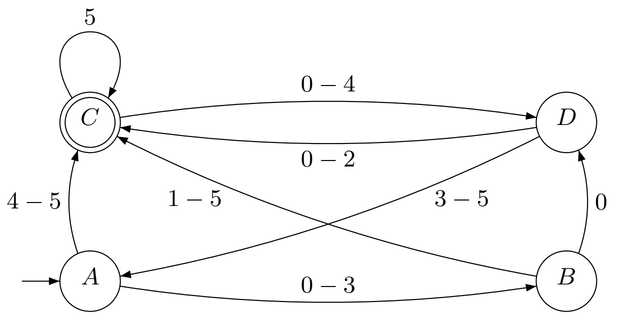


Fig. 1. A 2-edge DFA

Definition 9. *A sample $\langle X+, X-\rangle$ is k-acceptable if there is a k-edge* DFA *consistent with the data.*

Problem 2 *Given a sample $\langle X+, X-\rangle$, can we decide in polynomial time if a sample is k-acceptable?*

Problem 3 *Learn k-edge* DFA *from an informant, as in definition 4.*

Discussion We have proposed a way to use ranked alphabets in automata. Other ideas may be possible but we believe our formalism to be interesting because it is simple, it can take into account the fact the partial order may not be total; it can adapt very easily to continuous variables (even if in that case the language theory flavour will be lost); by having the intervals overlap it is possible to have various degrees of non-determinism.

There are some accessory problems more in the line of general formal language theory: do we have a pumping lemma? what are the closure properties?

5 Learning regular parts of context-free languages

In [18] was introduced the notion of learning with background knowledge. It is also well known that learning context-free grammars is much harder than learning DFA. The question would be to know if we could learn the intersection of a regular language L_1 and a context-free language L_2, given examples and counter-examples (of L_1) only from L_2.

Definitions Let $\mathcal{C}F(\Sigma)$ be the class of all context-free languages over some alphabet Σ and let $\mathcal{R}EG(\Sigma)$ be the class of all regular languages over Σ.

Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two language classes. Let $L = L_1 \cap L_2$, with $L_1 \in \mathcal{C}_1$ and $L_2 \in \mathcal{C}_2$. A presentation of L with respect to L_2 is a function $f : \mathbb{N} \to \Sigma^* \times \{0,1\} : f(\mathbb{N}) = L \times \{1\} \cup L_2 \setminus L_1 \times \{0\}$.

Definition 10 (Polynomial learning with help). *An algorithm $\boldsymbol{a}$ learns $\mathcal{C}_1$ from $\mathcal{C}_2$-knowledge if there exists a polynomial $p()$ and given any language $L = L_1 \cap L_2$, with $L_1 \in \mathcal{C}_1$ and $L_2 \in \mathcal{C}_2$, if G is the smallest grammar such that $\mathbb{L}(G) \cap L_2 = L$, and $\mathbb{L}(G) \in \mathcal{C}_1$ then $\exists W \subset X : \|W\| \leq p(|G|) \wedge W \subset f_n \implies \mathbb{L}(\boldsymbol{a}(f_n)) \cap L_2 = L$.*

Problem 4 *Find an algorithm that learns $\mathcal{R}EG(\Sigma)$ from $\mathcal{C}F(\Sigma)$-knowledge in the sense of definition 10.*

Discussion The problem may seem easy but the problem is that the characteristic sets may be inaccessible because outside language L_1. It should be noticed that attempting to learn directly language L is going to be impossible (take $L_2 = L_1$).

6 Testing equivalence of regular deterministic distributions

Learning stochastic languages is an important topic in grammatical inference. A number of algorithms have been proposed, with partial results: distributions defined by stochastic deterministic finite automata can be identified in the limit whith probability one. The algorithms are polynomial but obviously do not work from only polynomial amount of data. The question of learning such automata from only a reasonable quantity of data thus still remains open. One step towards an answer consist in being able (or not) to decide the equivalence between distributions given finite samples.

Definitions 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 modelled by some syntactic machine A, the probability of x according to the probability distribution defined by A is denoted $\Pr_A(x)$. The distribution modelled by a machine A will be denoted $\mathcal{D}_A$ and simplified to $\mathcal{D}$ if the context is non ambiguous.

Two distributions $\mathcal{D}$ and $\mathcal{D}'$ are equal (denoted by $\mathcal{D} = \mathcal{D}'$) if $\forall w \in \Sigma^\star : \Pr_{\mathcal{D}}(w) = \Pr_{\mathcal{D}'}(w)$. Alternatively one can define distances between distributions; a survey can be found in [19].

Problem 5 *Find an algorithm (or prove that it does not exist) that, given 2 samples extracted from* Sdfa *A and B, will tell us with high probability if A and B are equivalent (or close). Formally, if d is some distance between distributions:*

given $\epsilon, \delta > 0$, find an algorithm that given 2 samples extracted from Sdfa *A and B, will tell us if $d(\mathcal{D}_A, \mathcal{D}_B) < \epsilon$ with probability at least $1 - \delta$.*

Discussion To tighten the definition it would be necessary to use a *sampling query* allowing to sample (in $\mathcal{O}(1)$) $\mathcal{D}_A$ and $\mathcal{D}_B$. The number of examples should, like the algorithm, be polynomial in the sizes of the automata, $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$. If the hardness of the equivalence problem is closely linked with that of identifying in the limit [11], the related problem in the context of stochastic learning is the one above. The above problem is what most Sdfa learning algorithms try to solve, through comparing prefixes in the case of [20], or looking for some important string [21].

Solving this problem in a better way would allow to derive a test for the compatibility between 2 states in typical state merging algorithms. On the other hand, a negative answer to the above question would give us the clue towards proving the non learnability of Sdfa. Alternatively, partial results over special classes of Sdfa would also be helpful.

7 Queries

Learning with queries was introduced by Angluin [22] in order to study the non learnability of languages. The first results were therefore essentially negative [23], until a first algorithm ($L*$) was proposed [24] which could learn deterministic finite automata from membership queries (does this string belong to the target?) and equivalence queries (is this a representation of the language? If not, please provide me with a counter-example). This combination of queries (called a *minimum adequate teacher*) has been studied in [25] where the possibility of trading off some equivalence queries for some membership queries is explained. A survey paper is [26].

But these studies are based on the general idea that the Oracle is some perfect abstract machine. Indeed, Angluin [24] actually proposed techniques to implement the Oracle. *A contrario*, we believe that one should think of an Oracle as something quite different from that. Here are some arguments in favour of this point:

- Learning neural networks [16] that simulate automata. In that case, an issue is that of extracting an automaton from a neural network. A way to do this is to use the black box/neural network as an Oracle.
- Testing hardware [27]: the physical system or chip to be tested is the Oracle.
- System Squirrel [6] is used for wrapper induction. The system will interrogate the (human) user who will mark web pages.
- Today the World wide web can be seen as an Oracle. The knowledge is there, you cannot expect it to be sampled for you, nor to be able to use it all.

Definitions In a standard query learning algorithm, the learner interacts with an *oracle* (also called *minimally adequate teacher*), who knows the *target language* (a regular language L over a known alphabet) and is assumed to answer correctly. The teacher has to answer two types of queries: *membership queries* - the teacher's answer is *yes* or *no*, depending if the submitted string belongs or not to the language and *equivalence queries* - the learner produces a DFA A and asks whether $\mathcal{L}(A) = L$; the teacher answers *yes* if they accept the same language or *no* otherwise. If the answer is no, a string s in the symmetric difference of $\mathbb{L}(A)$ and L is returned. This returned string is called *counterexample.*

Definition 11. *Let $\epsilon > 0$. An ϵ-correct membership query is made by giving the Oracle a string. Then the Oracle answers correctly with probability 1-ϵ, unless it has already been submitted that query, in which case it replies consistently.*

Problem 6 *Are* DFA *learnable from ϵ-correct membership queries and equivalence queries?*
Discussion Equivalence queries should probably not be exact either. Being able to learn with such queries would allow to consider active learning in settings where it is unrealistic to believe that answers are going to be correct, *e.g.* from the Web.

8 Partially Observable Markov Decision Processes

Partialy Observable Markov Decision Processes are an extension of Markov Decision Processes, these being related to Hidden Markov Models. We consider here the case where the outputs at the states are some sort of reward function. They are important models in the setting of reinforcement learning, and inferring POMDPs is still a relatively untouched problem. The relationship with multiplicity automata has been shown in [28], and algorithm to learn these have also been studied [29].

Definitions A POMDP is defined by a set of states Q, an input alphabet Σ, an initial state q_0 and 2 functions:

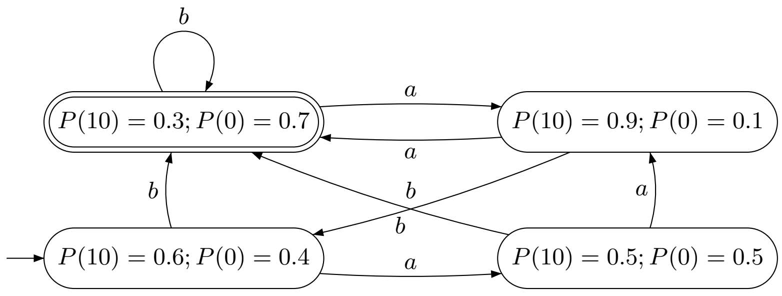
- A probabilistic transition function $\delta : Q \times \Sigma \times Q \rightarrow \mathbb{R}^+$ with $\forall q \in Q$, $\sum_{q \in Q, a \in \Sigma} \delta(q, a, q') = 1$.
- A probabilistic reward function $r : Q \rightarrow \mathbb{R}$

A POMDP is deterministic if function δ is deterministic, *i.e.* if $\forall q \in Q, \forall a \in \Sigma, \exists q' \in Q$ such that $\delta(q, a, q') = 1$.

Example 3. Figure 2 represents a deterministic POMDP. Notice that the output function may be much more complex. A possible outcome of taking the decisions $b \cdot b \cdot a$ could be 30, which would have occurred with probability $0.6 \cdot 0.3 \cdot 0.3 \cdot 0.1 + 0.6 \cdot 0.3 \cdot 0.7 \cdot 0.9 + 0.6 \cdot 0.7 \cdot 0.3 \cdot 0.9 + 0.4 \cdot 0.3 \cdot 0.3 \cdot 0.9 = 0.3132$

Problem 7 *Study the learnability of deterministic* POMDP*s.*

Problem 8 *Study the learnability of ordinary* POMDP*s.*



$P(10) = 0.3; P(0) = 0.7$
$P(10) = 0.9; P(0) = 0.1$
$P(10) = 0.6; P(0) = 0.4$
$P(10) = 0.5; P(0) = 0.5$
a
b

Fig. 2. A deterministic POMDP

Discussion There are several issues in the study of the learnability of POMPDs. Identification in the limit with probability 1 is an issue. But using convergence criteria from reinforcement learning (such as *regret*) is probably a better idea. The problem should also be related with strategy learning, as in [30].

9 Negotiation

Consider the situation where two adversaries have to negotiate something. The goal of each is to learn the model of the opponent while giving away as little information as possible. The situation can be modelled as follow:

Let L_1 be the language of adversary 1 and L_2 be the language of adversary 2. We suppose here that the languages are regular and can be represented by deterministic finite automata with respectively n_1 and n_2 states. The goal for each is to learn the common language, *i.e.* language $L_1 \cap L_2$.

It is well known that language $L_1 \cap L_2$ is also a regular language which can be represented by the product automaton. So there is an automaton of at most $n_1 * n_2$ states recognising $L_1 \cap L_2$.

The rule is that each adversary can only query the opponent by asking questions from his own language. This means that when player 1 names string w, then $w \in L_1$. In turn, the adversary will answer if, or not, string w belongs to the language.

Definitions The goal of each adversary is to identify language $L_1 \cap L_2$. This means that the protocol goes as follows:

- Player 1 announces some string w from L_1. Player 2 answers YES if this string belongs to L_2, NO otherwise.
- From this answer player 1 may update his hypothesis H_1 of language $L_1 \cap L_2$.
- From the information $w \in L_1$, player 2 may update his hypothesis H_2 of language $L_1 \cap L_2$
- Player 2 announces some string w from L_2.
- ...

We argue that for the problem to be well posed, one should be careful to avoid passive strategies, and count appropriately.

1. If L_1 is empty then the game is over straight away: player 1 has, as unique hypothesis possible for $L_1 \cap L_2$, the empty set which is correct.
2. The strategy "I shall wait and say nothing and let the opponent uncover his cards" is not always a good strategy: it is well known that DFA cannot be identified from text [9]: and take the situation where $L_1 = \Sigma^\star$, then the problem reduces to learning L_2, and using positive information only is not sufficient. But this holds only if the goal is to identify the common language!
3. It is therefore essential to count errors made, but only if the goal is reached. A draw should be counted as a loss for both players. In a sense the issue is similar to that of definition 7.

We propose the following definitions of successful learning: to make things simpler, we shall only consider here deterministic (non randomised) learners, in which case a game is defined by a pair $\langle L_1, L_2\rangle$, and the outcome of each game is unique for each pair of players A and B. We call $errors(X)$ the set of errors made by player X: an error is made when the string proposed by player X receives a negative answer from the other player.

Definition 12. *Given a game $\langle L_1, L_2\rangle$ and 2 players A and B, A wins if A identifies $L_1 \cap L_2$ in the limit and if either B identifies $L_1 \cap L_2$ in the limit and $|errors(A)| < |errors(B)|$ or if B does not identify $L_1 \cap L_2$ in the limit.*

Problem 9 *Study the general problem. What is a good strategy? Can identification be avoided? Are there any "no win" situations? Are there strategies that are so close one to the other (corresponding to what Angluin called "lock automata") that through only membership queries, learning is going to take too long?*

Problem 10 *Using definition 12, find a winning algorithm, in the case where $n_1 = n_2$.*

Discussion Being a good learning algorithm can be defined in alternative ways. One can want to be uniformally better than an adversary, than all the adversaries...

An intriguing question is: what happens if both opponents "agree" on a stalemate position, *i.e.* are satisfied with an identical language L which in fact is a subset of the target?

And, in a similar line to [30] one may considerlearning the strategy of the adversary; if we notice the adversary is daring/cautious, do we have anything better to do?

Acknowledgements

Thanks to Jose Oncina for different discussions that led to several definitions and problems from sections 4 and 6. Work with Henning Fernau on polynomial learning is where the ideas in section 3 come from. Philippe Jaillon gave me the initial ideas for the negotiation problem in section 9. Discussions with

Cristina Bibire and Leonor Becerra led to seeing the importance of the questions from section 7. I have also worked with Franck Thollard on learning stochastic deterministic finite state automata (section 6). It is also clear that had other specialists in grammatical inference given their problems, an alltogether different list would have been compiled. I therefore apologize to those who will not find their favourite problem here, or who believe that the really important are elsewhere.

References

1. Sakakibara, Y., Brown, M., Hughley, R., Mian, I., Sjolander, K., Underwood, R., Haussler, D.: Stochastic context-free grammars for tRNA modeling. Nuclear Acids Res. **22** (1994) 5112–5120
2. Abe, N., Mamitsuka, H.: Predicting protein secondary structure using stochastic tree grammars. Machine Learning Journal **29** (1997) 275–301
3. Kashyap, R.L.: Syntactic decision rules for recognition of spoken words and phrases using stochastic automaton. IEEE Trans. on Pattern Analysis and machine Intelligence **1** (1979) 154–163
4. Young-Lai, M., Tompa, F.W.: Stochastic grammatical inference of text database structure. Machine Learning Journal **40** (2000) 111–137
5. Chidlovskii, B., Ragetli, J., de Rijke, M.: Wrapper generation via grammar induction. In: Machine Learning: ECML 2000, 11th European Conference on Machine Learning. Volume 1810., Springer-Verlag (2000) 96–108
6. Carme, J., Gilleron, R., Lemay, A., Niehren, J.: Interactive learning of node selecting tree transducer. In: IJCAI Workshop on Grammatical Inference. (2005) Submitted to a Journal.
7. Amengual, J.C., Benedí, J.M., Casacuberta, F., Castaño, A., Castellanos, A., Jiménez, V.M., Llorens, D., Marzal, A., Pastor, M., Prat, F., Vidal, E., Vilar, J.M.: The EuTrans-I speech translation system. Machine Translation **15** (2001) 75–103
8. Harrison, M.H.: Introduction to Formal Language Theory. Addison-Wesley Publishing Company, Inc., Reading, MA (1978)
9. Gold, E.M.: Language identification in the limit. Information and Control **10** (1967) 447–474
10. Pitt, L.: Inductive inference, DFA's, and computational complexity. In: Analogical and Inductive Inference. Number 397 in LNAI. Springer-Verlag, Berlin, Heidelberg (1989) 18–44
11. de la Higuera, C.: Characteristic sets for polynomial grammatical inference. Machine Learning Journal **27** (1997) 125–138
12. Yokomori, T.: Polynomial-time identification of very simple grammars from positive data. Theoretical Computer Science **1** (2003) 179–206
13. Zeugmann, T.: Can learning in the limit be done efficiently? In Gavaldà, R., Jantke, K., Takimoto, E., eds.: ALT. Number 2842 in LNCS, Berlin, Heidelberg, Springer-Verlag (2003) 17–38
14. Parekh, R.J., Honavar, V.: On the relationship between models for learning in helpful environments. In de Oliveira, A., ed.: Grammatical Inference: Algorithms and Applications, Proceedings of ICGI '00. Volume 1891 of LNAI., Berlin, Heidelberg, Springer-Verlag (2000) 207–220

15. Cruz, P., Vidal, E.: Learning regular grammars to model musical style: Comparing different coding schemes. [31] 211–222
16. Giles, C.L., Lawrence, S., Tsoi, A.: Noisy time series prediction using recurrent neural networks and grammatical inference. Machine Learning Journal **44** (2001) 161–183
17. Dupont, P., Chase, L.: Using symbol clustering to improve probabilistic automaton inference. [31] 232–243
18. Kermorvant, C., de la Higuera, C.: Learning languages with help. In Adriaans, P., Fernau, H., van Zaannen, M., eds.: Grammatical Inference: Algorithms and Applications, Proceedings of ICGI '02. Volume 2484 of LNAI., Berlin, Heidelberg, Springer-Verlag (2002) 161–173
19. Vidal, E., Thollard, F., de la Higuera, C., Casacuberta, F., Carrasco, R.C.: Probabilistic finite state automata – part I and II. Pattern Analysis and Machine Intelligence **27** (2005) 1013–1039
20. Carrasco, R.C., Oncina, J.: Learning stochastic regular grammars by means of a state merging method. In Carrasco, R.C., Oncina, J., eds.: Grammatical Inference and Applications, Proceedings of ICGI '94. Number 862 in LNAI, Berlin, Heidelberg, Springer-Verlag (1994) 139–150
21. Ron, D., Singer, Y., Tishby, N.: On the learnability and usage of acyclic probabilistic finite automata. In: Proceedings of COLT 1995. (1995) 31–40
22. Angluin, D.: Queries and concept learning. Machine Learning Journal **2** (1987) 319–342
23. Angluin, D.: A note on the number of queries needed to identify regular languages. Information and Control **51** (1981) 76–87
24. Angluin, D.: Learning regular sets from queries and counterexamples. Information and Control **39** (1987) 337–350
25. Balcazar, J.L., Diaz, J., Gavaldà, R., Watanabe, O.: The query complexity of learning DFA. New Generation Computing **12** (1994) 337–358
26. Angluin, D.: Queries revisited. In Abe, N., Khardon, R., Zeugmann, T., eds.: Proceedings of ALT 2001. Number 2225 in LNCS, Berlin, Heidelberg, Springer-Verlag (2001) 12–31
27. Hagerer, A., Hungar, H., Niese, O., Steffen, B.: Model generation by moderated regular extrapolation. In R. Kutsche, H.W., ed.: Proc. of the 5th Int. Conference on Fundamental Approaches to Software Engineering (FASE '02). Volume 2306 of LNCS., Heidelberg, Germany, Springer-Verlag (2002) 80–95
28. Eyal Even-Dar, S.K., Mansour, Y.: Planning in pomdps using multiplicity automata. In: Proceedings of 21st Conference on Uncertainty in Artificial Intelligence (UAI). (2005) 185–192
29. Beimel, A., Bergadano, F., Bshouty, N.H., Kushilevitz, E., Varricchio, S.: Learning functions represented as multiplicity automata. J. ACM **47** (2000) 506–530
30. Carmel, D., Markovitch, S.: Exploration strategies for model-based learning in multiagent systems. Autonomous Agents and Multi-agent Systems **2** (1999) 141–172
31. Honavar, V., Slutski, G., eds.: Grammatical Inference, Proceedings of ICGI '98. In Honavar, V., Slutski, G., eds.: Grammatical Inference, Proceedings of ICGI '98. Number 1433 in LNAI, Berlin, Heidelberg, Springer-Verlag (1998)