A Note on Phase Transitions and Computational Pitfalls of Learning from Sequences

Antoine Cornuéjols[1] & Michèle Sebag[2]

[1]AgroParisTech / INRA
UMR 518, Mathématiques et Informatique Appliquées
F75005 Paris (France)

[2] Laboratoire de Recherche en Informatique, CNRS UMR 8623
Bât.490, Université de Paris-Sud, Orsay
91405 Orsay Cedex (France)
antoine,sebag@lri.fr

January 10, 2007

Abstract

An ever greater range of applications call for learning from sequences. Grammar induction is one prominent tool for sequence learning, it is therefore important to know its properties and limits.

This paper presents a new type of analysis for inductive learning. A few years ago, the discovery of a phase transition phenomenon in inductive logic programming proved that fundamental characteristics of the learning problems may affect the very possibility of learning under very general conditions.

We show that, in the case of grammatical inference, while there is no phase transition when considering the whole hypothesis space, there is a much more severe "gap" phenomenon affecting the effective search space of standard grammatical induction algorithms for deterministic finite automata (DFA). Focusing on standard search heuristics, we show that they overcome this difficulty to some extent, but that they are subject to overgeneralization. The paper last suggests some directions to alleviate this problem.

1 Introduction

A wide range of applications which includes language modeling in speech recognition, translation, bioinformatics, music modeling, circuit testing, alarm detection or time series, are dependent upon the capability to model, generate and learn distributions over sets of possibly infinite cardinality of strings, sequences, words, phrases and even trees. A set of representations for sequences has therefore been developed over the years. Among them: Finite State Automata (FSA) (deterministic or not), probabilistic

Finite State Automata, stochastic regular grammars, Markov chains, hidden Markov models, n-grams.

In bioinformatics, for instance, linguistic methods have provided some interesting results in the recognition of complex biological signals [Sakakibara *et al.*, 1994]. However, the hand development of grammars is difficult and, because it requires human expertise, expensive. Thus, given the enormous volume of data arising from genome projects, there is a need to automate the acquisition of grammars from sets of biological sequences.

Therefore, in front of the growing area of applications that deal with the analysis of sequences, it is important to be aware of the properties of learning from sequences.

In this paper, we concentrate on learning grammars from sequences, and more particularly regular grammars, which form the simplest class in the Chomsky hierarchy of formal languages. Regular languages correspond to Finite State Automata. An understanding of the issues and difficulties encountered in learning regular languages are indeed likely to provide insights into the problem of learning more general classes of languages.

In the field of grammatical inference, it is known that exact learning of the target regular language from an arbitrary presentation of labeled examples is a hard problem [Gold, 1978]. Gold showed that the problem of identifying the minimum state DFA (Deterministic Finite State Automata) consistent with a finite non-empty set of positive examples and possibly a finite non-empty set of negative examples is *NP*-hard. It is therefore necessary to provide additional information in order to get efficient learning algorithms. Several learning frameworks have been proposed in this perspective.

In the *query learning* framework, the learner may ask queries to an oracle. Angluin [Angluin, 1987] proved that regular languages are exactly learnable within polynomial time using membership and equivalence queries. The availability of an oracle is however questionable in many real-world settings. In the *Probably Approximately Correct* framework (PAC), one only seeks a language that is close to the target language in the sense that there is only a low probability, under the distribution on the example space, that an example be labeled differently by the target automata and the learned one. Theoretical analyzes mostly conclude that regular languages cannot be learned in this setting. One can then restrict the distributions of the examples so that *simple* examples are more probable. For instance, in the so-called PACS setting [Denis *et al.*, 1996], it has been proved that the class of regular languages is learnable [Parekh & Hanovar, 1997].

Another way to provide insightful information to the learning system is to supply it with a so-called *structurally complete* learning sample. Intuitively, this is a learning set that is sufficiently representative of the language to be learned. In concrete terms, this means that every component of the target automaton is exercised in the recognition of at least one learning example. Most of the standard grammatical inference systems have been devised following this assumption since it confers desirable properties to the search space (see section 3 below). For instance, the *regular positive and negative inference* (RPNI) algorithm is a framework for identifying in polynomial time a DFA consistent with a given sample S. Further, if S is a superset of a structurally complete set (technically, a *characteristic set*) for the target DFA, then the DFA output by the RPNI algorithm is guaranteed to be equivalent to the target [Oncina & Garcia, 1992].

Benchmarks and competitions have spurred the development of algorithms and al-

low the community of researchers to test their methods (for instance, in the Abbadingo competition, DFA's of 512 states had to be inferred [Lang *et al.*, 1998]).

However, at present, a lot remains unknown about the feasibility of grammar induction. On the theoretical side, new frameworks have been defined, but the results are overly of a negative nature. Further and surprisingly, the statistical theory of induction linking the empirical risk minimization principle and the "capacity" of the hypothesis space [Vapnik, 1995] has not influenced the field of grammatical inference. On the practical side, many problems are still unsolved, even in old challenges like the Abbadingo one. Researchers have turned their attention toward more powerful languages, like Probabilistic Finite State Automata or Markov models, but there is still a need to better understand the induction of regular languages.

This paper presents a new type of analysis of grammar induction. Inspired by the works on Constraint Satisfaction problems (CSP) that show that a phase transition phenomenon affects many problems and conditions in great part the complexity of solving them, we undertook a closely related systematic investigation in the grammar induction framework. The results clearly demonstrates the existence of a phase transition like phenomenon in grammatical inference as well. This gives reasons for concern as the induction of FSA's, and most noticeably of DFA's, is shown to incur serious potential obstacles in the learning of a wide range of possible target languages.

Our contribution in this paper is threefold. First, we introduce a new type of analysis for grammar induction that focuses on a property linking the hypothesis representation language and the characteristics of the learning sequences. Second, thanks to this framework, we demonstrate that induction of regular languages is prone to a phase transition phenomenon that questions the feasibility of learning in other than toy environments. Third, we provide an analysis and an explanation for this phenomenon and sketch possible directions to remedy it.

The paper is organized as follows. Section 2 introduces the phase transition phenomenon in the context of inductive learning and its meaning. Then, section 3 briefly introduces the domain of Grammatical Inference, the principles of the inference algorithms and defines the control parameters used in the rest of the paper. Sections 4 reports the sets of experiments that were carried out in order to explore the behavior of inductive algorithms in grammatical inference. The performances of two representative learning algorithms are specially examined in section 5. Section 6 discusses the scope of the presented study and lays out some perspectives for future research.

2 Phase transition and learning

Inductive learning is concerned with the discovery of hypotheses, taken from an hypothesis space, that well account for the learning instances that are supposed to be representative of some general law. The usual approach is to explore the hypothesis space in order to find some hypothesis that explains the positive instances in the training set while it rejects negative instances. Induction is therefore tightly linked with a *covering test procedure* whereby an expression of the candidate hypothesis is tested against the expressions of the training instances. Furthermore, the search in the hypothesis space usually follows a *gradient like strategy* by which candidate hypotheses

of ever greater performance over the training set are explored until a best one is found.

It is therefore obvious that the covering test and its properties play an essential role in learning. First, its computational complexity determines the overall complexity of induction. In the worst scenario, induction can become intractable. Second, and even more importantly, if the gradient of the coverage of the hypotheses with regard to the learning instances is not well-behaved, then the exploration of the hypothesis space can simply become unmanageable.

As it is, it just happened that a new paradigm has been studied in the Constraint Satisfaction community since the early 90s, motivated by computational complexity concerns: where are the really hard problems? [Cheeseman *et al.*, 1991] Indeed, the worst case complexity analysis poorly accounts for the fact that, despite an exponential worst-case complexity, empirically, the complexity is low for most CSP instances. These remarks led to developing the so-called *phase transition framework* [Hogg *et al.*, 1996], which considers the satisfiability and the resolution complexity of CSP instances as random variables depending on order parameters of the problem instance (e.g. constraint density and tightness). This framework unveiled an interesting structure of the CSP landscape. Specifically, the landscape is divided into three regions: the `YES` region, corresponding to underconstrained problems, where the satisfiability probability is close to 1 and the average complexity is low; the `NO` region, corresponding to over-constrained problems, where the satisfiability probability is close to 0 and the average complexity is low too; last, a narrow region separating the `YES` and `NO` regions, referred to as *phase transition region*, where the satisfiability probability abruptly drops from 1 to 0 and which concentrates on average the computationally heaviest CSP instances.

The phase transition paradigm has been transported to relational machine learning and inductive logic programming (ILP) by [Giordana & Saitta, 2000], motivated by the fact that the covering test most used in ILP [Muggleton & Raedt, 1994] is equivalent to a CSP. As anticipated, a phase transition phenomenon appears in the framework of ILP: a wide `YES` (respectively `NO`) region includes all hypotheses which cover (resp. reject) all examples, and the hypotheses that can discriminate the examples lie in the narrow phase transition region, where the average computational complexity of the covering test reaches its maximum.

Besides computational complexity, the phase transition phenomenon has far-reaching effects on the success of relational learning [Botta *et al.*, 2003]. For instance, a wide *Failure Region* is observed: for all target concepts/training sets in this region, no learning algorithms among the prominent ILP ones could find hypotheses better than random guessing [Botta *et al.*, 2003].

The phase transition paradigm thus provides another perspective on the pitfalls facing machine learning, focusing on the combinatoric search aspects while statistical learning focuses on the statistical aspects.

3 Grammatical inference

After introducing general notations and definitions, this section briefly discusses the state of the art and introduces the order parameters used in the rest of the paper.

3.1 Notations and definitions

Grammatical inference is concerned with inferring grammars from positive (and possibly negative) examples. It is known that any regular language can be produced by a finite-state automaton (FSA), and that any FSA generates a regular language. In the remaining of the paper, we will mostly use the terminology of finite-state automata. A FSA is a 5-tuple $A = \langle \Sigma, \mathcal{Q}, \mathcal{Q}_0, F, \delta \rangle$ where Σ is a finite alphabet, $\mathcal{Q}$ is a finite set of states, $\mathcal{Q}_0 \subseteq \mathcal{Q}$ is the set of initial states, $F \subseteq \mathcal{Q}$ is the set of final states, δ is the transition function defined from $\mathcal{Q} \times \Sigma$ to $2^{\mathcal{Q}}$.

A positive example of a FSA is a string on Σ, produced by following any path in the graph linking one initial state q_0 to any accepting state.

A finite state-automaton (FSA) is deterministic (DFA) if $\mathcal{Q}_0$ contains exactly one element q_0 and if $\forall q \in \mathcal{Q}, \forall x \in \Sigma, Card(\delta(q, x)) \leq 1$. Otherwise it is non-deterministic (NFA). Every NFA can be translated into an equivalent DFA, possibly at the price of being exponentially more complex in terms of number of states. Given any FSA A', there exists a minimum state DFA (also called *canonical DFA*) A such that $L(A) = L(A')$ (where $L(A)$ denotes the set of strings accepted by A). Without loss of generality, it can be assumed that the target automaton being learned is a canonical DFA.

A set S^+ is said to be *structurally complete* with respect to a DFA A if S^+ covers each transition of A and uses every element of the set of final states of A as an accepting state. Clearly, $L(PTA(S^+)) = S^+$.

Given a FSA A and a partition π on the set of states $\mathcal{Q}$ of A, the *quotient automaton* is obtained by merging the states of A that belong to the same block in partition π (see [Dupont *et al.*, 1994] for more details). Note that a quotient automaton of a DFA might be a NFA and vice versa. The set of all quotient automata obtained by systematically merging the states of a DFA A represents a lattice of FSAs. This lattice is ordered by the *grammar cover* relation $\preceq$. The transitive closure of $\preceq$ is denoted by $\ll$. We say that $A_{\pi_i} \ll A_{\pi_j}$ *iff* $L(A_{\pi_i}) \subseteq L(A_{\pi_j})$. Given a canonical DFA A and a set S^+ that is structurally complete with respect to A, the lattice derived from $PTA(S^+)$ is guaranteed to contain A.

From these assumptions follows the paradigmatic approach of most grammatical inference algorithms (see, e.g., [Dupont *et al.*, 1994, Pitt, 1989, Sakakibara, 1997]), which equates generalization with state merging operations starting from the PTA.

3.2 Learning biases in grammatical inference

The core task of GI algorithms is thus to select iteratively a pair of states to be merged. The differences among algorithms is related to the choice of: (i) the search criterion (which merge is the best one); (ii) the search strategy (how is the search space explored); and (iii) the stopping criterion.

We shall consider here the setting of learning FSAs from positive and negative examples, and describe the algorithms studied in section 4. In this setting, the stopping criterion is determined from the negative examples: generalization proceeds as long as the candidate solutions remain correct, not covering any negative example[1]

[1] In this paper, following the standard Machine Learning terminology, a string is said to be covered by a FSA *iff* it belongs to the language thereof.

The RPNI algorithm [Oncina & Garcia, 1992] uses a depth first search strategy with some backtracking ability, favoring the pair of states which is closest to the start state, such that their generalization (FSA obtained by merging the two states and subsequently applying the determinisation operator) does not cover any negative example.

The RED-BLUE algorithm (also known as BLUE-FRINGE) [Lang *et al.*, 1998] uses a beam search from a candidate list, selecting the pair of states after the *Evidence-Driven* State Merging (EDSM) criterion, i.e. such that their generalization involves a minimal number of final states. RED-BLUE thus also performs a search with limited backtracking, based on a more complex criterion and a wider search width than RPNI.

4 Phase transition in grammatical inference

Following the methodology introduced in [Giordana & Saitta, 2000], the phase transition phenomenon is investigated along so-called *order parameters* chosen in accordance with the parameters used in the *Abbaddingo* challenge [Lang *et al.*, 1998]:

- The number Q of states in the DFA.
- The number B of output edges on each state.
- The number L of letters on each edge.
- The fraction a of accepting states, taken in [0,1].
- The size $|\Sigma|$ of the alphabet considered.
- The length ℓ of the test examples. Also the maximal length ℓ of the learning examples in S^+ (as explained below).

The study first focuses on the intrinsic properties of the search space (section 4) with no regard to the learning algorithms. Using a set of order parameters, the average coverage of automata is studied analytically and empirically.

In a second step (section 4.2), we test the coverage of deterministic and non-deterministic Finite-State Automata in the subspace actually investigated by grammatical inference algorithms. Indeed, the vast majority of these algorithms first construct a least general generalization of the positive examples, or Prefix Tree Acceptor (PTA), and restrict the search to the generalizations of the PTA, or *generalization cone*[2].

In section 5, we examine the capacity of the studied learning algorithms to approximate a target automaton, based on *positive* and *negative* sampling.

4.1 Phase Transition in the whole FSA space

The sampling mechanism on the whole deterministic FSA space (DFA) is defined as follows. Given the order parameter values $(Q, B, L, a, |\Sigma|)$:

[2] More precisely, the Prefix Tree Acceptor is obtained by merging the states that share the same prefix in the Maximal Canonical Automaton (MCA), which represents the whole positive learning set as an automaton. A PTA is therefore a DFA with a tree-like structure.

- for every state q, (i) B output edges (q, q') are created, where q' is uniformly selected with no replacement among the Q states; (ii) $L \times B$ distinct letters are uniformly selected in Σ; and (iii) these letters are evenly distributed among the B edges above.

- every state q is turned into an accepting state with probability a.

The sampling mechanism for NFA differs from the above in a single respect: two edges with same origin state are not required to carry distinct letters.

For each setting of the order parameters, 100 independent problem instances are constructed. For each considered FSA (the sampling mechanisms are detailed below), the coverage rate is measured as the percentage of covered examples among 1,000 examples (strings of length ℓ) uniformly sampled.

Fig. 1 shows the average coverage in the (a, B) plane, for $|\Sigma| = 2$, $L = 1$ and $\ell = 10$, where the accepting rate a varies in $[0, 1]$ and the branching factor B varies in $\{1,2\}$. Each point reports the average coverage of a sample string s by a FSA (averaged over 100 FSA drawn with accepting rate a and branching factor B, tested on $1,000$ strings s of length ℓ).

These empirical results are analytically explained from the simple equations below, giving the probability that a string of length ℓ be accepted by a FSA defined on an alphabet of size $|\Sigma|$, with a branching factor B and L letters on each edge, in the DFA and NFA cases (the number of states Q is irrelevant here).

$$P(\text{accept}) = \begin{cases} a \cdot (\frac{B \cdot L}{|\Sigma|})^{\ell} & \text{for a DFA} \\ a \cdot [1 - (1 - \frac{L}{|\Sigma|})^{B}]^{\ell} & \text{for a NFA} \end{cases} \quad (1)$$

The coverage of the FSA decreases as a and B decrease. The slope is more abrupt in the DFA case than in the NFA case; still, there is clearly no phase transition here.

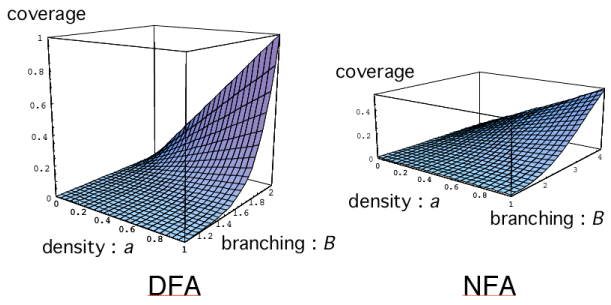

Figure 1: Coverage landscapes for Deterministic and Non-Deterministic FSA, for $|\Sigma|$=2, L=1 and ℓ=10. The density of accepting states a and the branching factor B respectively vary in $[0, 1]$ and $\{1, 2\}$.

4.2 Phase Transition in the Generalization Cone

The coverage landscape displayed in Fig. 1 might suggest that grammatical inference takes place in a well-behaved search space. However, grammatical inference algorithms do not explore the whole FSA space. Rather, as stated in section 4, the search is restricted to the generalization cone, the set of generalizations of the PTA formed from the set S^+ of the positive examples. The next step is thus to consider the search space actually explored by GI algorithms.

A new sampling mechanism is defined to explore the DFA generalization cone:

1. $|S^+|$ ($= 200$ in the experiments) examples of length ℓ are uniformly and independently sampled within the space of all strings of length $< \ell$, and the corresponding PTA is constructed;

2. N ($= 50$ in the experiments) PTAs are constructed in that way.

3. K ($= 20$ in the experiments) generalization paths, leading from each PTA to the most general FSA or Universal Acceptor (UA), are constructed;
   In each generalization path ($A_0 = PTA, A_1, \ldots, A_t = UA$), the i-th FSA A_i is constructed from A_{i-1} by merging two uniformly selected states in A_{i-1}, and subsequently applying the determinisation operator.

4. The generalization cone sample is made of all the FSAs in all generalization paths (circa 270,000 FSAs in the experiments).

The sampling mechanism on the non-deterministic generalization cone differs from the above in a single respect: the determinisation operator is never applied.

Fig. 2 (left) shows the behaviour of the coverage in the DFA generalization cone for $|\Sigma| = 4$ and $\ell = 8$. Each DFA A is depicted as a point with coordinates (Q, c), where Q is the number of states of A and c is its coverage (measured as in section 4). The coverage rate for each FSA in the sample is evaluated from the coverage rate on 1000 test strings of length ℓ. Typical of all experimental results in the range of observation ($|\Sigma| = 2, 4, 8, 16$, and $\ell = 2, 4, 6, 8, 16, 17$), the figure shows a clear-cut phase transition. Specifically, here, the coverage abruptly jumps from circa 13% to 54%; and this jump coincides with a gap in the number of states of the DFAs in the generalization cone: no DFA with a number of states in $[180, 420]$ was found. The gap is even more dramatic as the length of the training and test sequences ℓ is increased.

Fig. 2 (right) similarly shows the behaviour of the coverage in the NFA generalisation cone, with $|\Sigma| = 4$ and $\ell = 16$. Interestingly, a much smoother picture appears. Although the coverage rapidly increases when the number of states decreases from 300 to 200, no gap can be seen, neither in the number of states nor in the coverage rate itself[3].

In the following, we focus on the induction of DFAs.

[3] The difference with the DFA case is due to the determinisation process that forces further states merging when needed. A diffusion like analytical model was devised, that predicts the observed start of the gap with 15% precision.

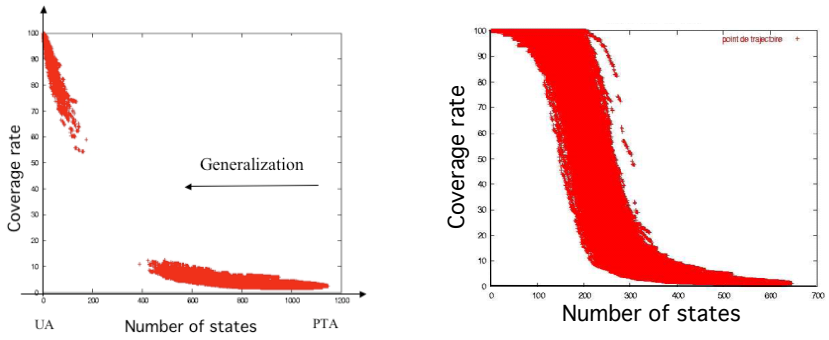

Figure 2: (Left) Coverage landscape in the DFA generalization cone ($|\Sigma| = 4$, $\ell = 8$). At the far right stand the 50 PTA sampled, with circa 1150 states each. The generalization cone of each PTA includes 1,000 generalization paths, leading from the PTA to the Universal Acceptor. Each point reports the coverage of a DFA, evaluated over a sample of 1,000 strings. This graph shows the existence of a large gap regarding both the number of states and the coverage of the DFAs that can be reached by generalization. (Right) Coverage landscape in the NFA generalization cone, with same order parameters as in fig. 2 (left).

5 Phase transition and search trajectories

The coverage landscape, for the DFAs, shows a hole in the generalization cone, with a density of hypotheses of coverage in between a large interval (typically between less than 20% to approximately 60%) falling abruptly. Therefore, a random exploration of the generalization cone would face severe difficulties in finding a hypothesis in this region and would likely return hypotheses of poor performance if the target concept had a coverage rate in this "no man's land" interval.

It is consequently of utmost importance to examine the search heuristics that are used in the classical grammatical inference systems. First, are they able to thwart the *a priori* very low density of hypotheses in the gap? Second, are they able to guide the search toward hypotheses of appropriate coverage rate, specially if this coverage falls in the gap?

The study next focuses on two standard algorithms in grammatical inference, namely [Oncina & Garcia, 1992, Lang *et al.*, 1998] the RPNI and the RED-BLUE (aka. EDSM) algorithms.

5.1 Experimental setting

Previous experiments considered training sets made of positive randomly drawn strings sequences only. However, in order to assess the performance of learning algorithms, the hypothesis learned must now be compared to the target automaton. Therefore, another experimental setting is used in this section, with the sampling of target automata, and

the construction of training and test sets. These data sets include positive and negative examples as most GI algorithms (and specifically RPNI and RED-BLUE) use negative examples in order to stop the generalization process.

In our first experiments, we tested whether heuristically guided inference algorithms can find good approximations of the target automata considering target automata with approximately (i) 50% coverage rate (as considered in the influential *Abbadingo* challenge, and in the middle of the "gap"), and (ii) 5% coverage rate.

For each target coverage rate, we used the same experimental setting as described in [Lang *et al.*, 1998] in order to retain a certain number of target automata with a mean size of Q states ($Q = 50$, in our experiments). For each automaton then, we generated N (=20) training sets of size $|S|$ (= 100) labeled according to the target automaton, with an equal number of positive and negative instances ($|S^+| = |S^-| = 50$) of length $\ell = 14$. The coverage rate was computed as before on 1000 uniformly drawn strings (with no intersection with the training set).

In a second set of experiments, we analyzed the learning performances of the algorithms with respect to test errors, both false positive and false negative.

In these experiments, we chose the type of target automata by setting the number of states Q and some predetermined structural properties[4].

5.2 The heuristically guided search space

Due to space limitation, only the graph obtained for the RPNI algorithm is reported (see figure 3), with three typical learning trajectories. Similar results were obtained with the RED-BLUE algorithm.

One immediate result is that both the RPNI and the EDSM heuristics manage to densely probe the "gap". This can explain why the gap phenomenon was not discovered before, and why the RED-BLUE algorithm for instance could solve some cases of the *Abbadingo* challenge where the target concepts have a coverage rate of approximately 50%. However, where RPNI tends to overspecialize the target automaton, RED-BLUE tends to overgeneralize it by 5% to 10%.

In order to test the capacity of the algorithms to return automata with a coverage rate close to the target coverage, we repeated these experiments with target automata of coverage rate of approximately 3%. The results (figure 3) shows that, in this case, RPNI ends up with automata of coverage 4 to 6 times greater than the target coverage. The effect is even more pronounced with RED-BLUE which returns automata of average coverage rate around 30%!

5.3 Generalization error

Table 1, obtained for different sizes of the target automata and for training sets of structural completeness above 40%, confirms that both RPNI and RED-BLUE return overgeneralized hypotheses. On one hand, their average coverage is vastly greater than the coverage of the target automata, on the other hand, they tend to cover only part of the positive test instances, while they cover a large proportion of the negative test

[4] The datasets and more detail are available at `http://www.lri.fr/~antoine/www/pt-gi/`

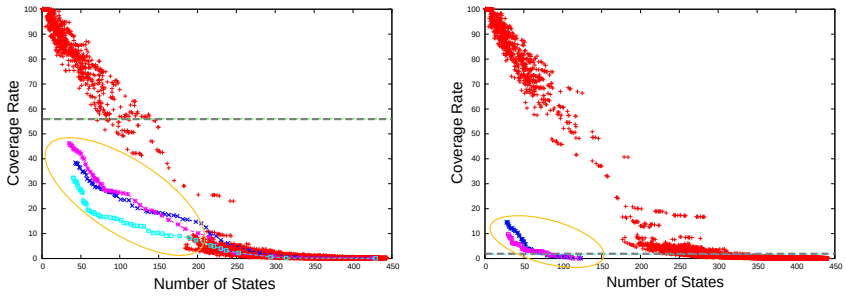

Figure 3: (Left) Three RPNI learning trajectories for a target concept of coverage=56%. Their extremity is outlined in the oval on the left. The doted horizontal line corresponds to the coverage of the target concept. The cloud of points corresponds to random trajectories. (Right) Same as in figure 3, except for the coverage of the target concept, here 3%.

instances. This shows that the heuristics used in both RPNI and RED-BLUE may be inadequate for target concepts of low coverage.

Algo.	Q_c	$ucov_c$	Q_f	$ucov_f$	$pcov_f$	$ncov_f$
RB	15	5.97	10.38	33.81	60.93	34.69
RB	25	4.88	12.77	40.35	62.68	37.87
RB	50	4.2	14.23	45.38	66.14	42.23
RB	100	3.39	13.13	30.35	42.81	28.69
RPNI	15	5.95	5.14	22.9	57.51	26.99
RPNI	25	4.7	7.56	23.07	56.38	25.98
RPNI	50	3.87	14.08	23.45	51.89	24.42
RPNI	100	3.12	26.41	23.151	50.12	24.40

Table 1: Performances of RED-BLUE (RB) and RPNI for target DFA of sizes Q = 15, 25, 50 and 100 states. Q_f, $ucov_f$, $pcov_f$ and $ncov_f$ respectively denote the average size of the learned automata, their average coverage, the true positive and the false positive rates.

6 Conclusion

This paper is a step toward a better understanding of grammar induction. Its first contribution is to bring a new kind of analysis that emphasizes the importance of the properties of the coverage landscape in determining the feasibility of learning. Specifically, it was shown that if the landscape seems well-behaved when the whole space of finite

state automata is considered, the picture is completely different when the space actually searched by existing learning algorithms is taken into account. These algorithms that are all based upon state merging operators are threatened to be unable to probe a wide range of candidate automata.

In fact, an analysis of the phase transitions found in the DFA case points to two causes. First, equation 1 for the DFA shows that when the size ℓ of the strings is higher than 10 say, a very small change in the branching factor B or the number of letters per edge L can dramatically modify $P(accept)$. Second, DFA induction implies that state merging operations take place from time to time in order to restore determinism in the induced automaton. But, our study, and a computer model that we developed, demonstrate that these operations tend to occur in chain reaction like processes when the choice of merging operation is random during induction. This phenomenon, compounded with the first observation, predicts very well the gap observed in the experiments.

This explains why sophisticated search biases are needed for grammatical inference algorithms in the problem range corresponding with the hole in the generalization cone. Regrettably, a second finding of this research concerns the limitations of the search operators in the standard algorithms RPNI and RED-BLUE, especially outside the region of intermediate coverage target concepts. Experiments with artificial learning problems built from target concepts with coverage less than 10% reveal that RPNI and RED-BLUE alike tend to learn overly general hypotheses; with respect to both the size (estimated by the number of states) and the coverage of the hypotheses, often larger by an order of magnitude than that of the target concept. What is even more worrying is that this overgeneralization *does not imply* that the found hypotheses are complete: on the contrary, the coverage of the positive examples remains below 65%, in all but one setting. Unfortunately, many real-world applications involve target concepts of low coverage (e.g. molecules that are active against a specific biological target). There are therefore reasons for concern.

But this analysis also opens several perspectives for further research. First, it suggests that, unlike in existing algorithms, the learning search, and especially the stopping criterion, could be controlled using a hyper-parameter: the coverage rate of the target concept (possibly supplied by the expert, or estimated e.g. by cross-validation). Secondly, more conservative generalisation operators can be used. Preliminary experiments done with e.g. reverted generalisation (same operator as in RPNI, applied on the reverted example strings) show that such operators can delay the determinisation cascade, and offer a finer control of the final coverage rate of the hypotheses.

To sum up, the analysis based on the study of the coverage landscape offers fundamental clues about the interaction, in induction, of the hypothesis space and the example space. The generic properties of a family of learning algorithms can be studied, leading to better understanding of their limits but also providing indications about possible ways to alleviate these.

Acknowledgments

The authors are partially supported by the PASCAL Network of Excellence IST-2002-506 778. They gratefully acknowledge the help of Nicolas Pernot for the work done at the beginning of this project.

References

[Angluin, 1987] Angluin, D. (1987). *Information and Computation,* **75** (2), 87–106.

[Botta *et al.*, 2003] Botta, M., Giordana, A., Saitta, L., & Sebag, M. (2003). *Journal of Machine Learning Research,* **4**, 431–463.

[Cheeseman *et al.*, 1991] Cheeseman, P., Kanefsky, B., & Taylor, W. M. (1991). In: *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, IJCAI-91, Sidney, Australia* pp. 331–337,.

[Denis *et al.*, 1996] Denis, F., D'Halluin, C., & Gilleron, R. (1996). In: *13th Annual Symposium on Theoretical Aspects of Computer Science* volume LNCS-1046 pp. 231–242, Grenoble, France: Springer.

[Dupont *et al.*, 1994] Dupont, P., Miclet, L., & Vidal, E. (1994). In: *Proceedings of the Second International Colloquium on Grammatical Inference and Applications, ICGI-94* pp. 25–37,.

[Giordana & Saitta, 2000] Giordana, A. & Saitta, L. (2000). *Machine Learning,* **41**, 217–251.

[Gold, 1978] Gold, E. M. (1978). *Information and Control,* **37**, 302–320.

[Hogg *et al.*, 1996] Hogg, T., Hubberman, B., & Williams, C. (1996). *Artificial Intelligence,* **81**, 1–15.

[Lang *et al.*, 1998] Lang, K., Pearlmutter, B., & Price, R. (1998). In: *Fourth International Colloquium on Grammatical Inference (ICGI-98)* volume LNCS-1433 pp. 1–12, Springer Verlag.

[Muggleton & Raedt, 1994] Muggleton, S. & Raedt, L. D. (1994). *Journal of Logic Programming,* **19/20**, 629–679.

[Oncina & Garcia, 1992] Oncina, J. & Garcia, P. (1992). *Pattern Recognition and Image Analysis,* , 49–61.

[Parekh & Hanovar, 1997] Parekh, R. & Hanovar, V. (1997). In: *Workshop on Algorithmic Learning Theory (ALT-97)*, (Li, M. & Maruoka, A., eds) volume LNAI-1316 pp. 116–131, Berlin, Germany: Springer.

[Pitt, 1989] Pitt, L. (1989). In: *Proceedings of the Workshop on Analogical and Inductive Inference (AII-89)* volume LNCS-397 pp. 18–44, Springer Verlag.

[Sakakibara, 1997] Sakakibara, Y. (1997). *Theoretical Computer Science,* **185**, 15–45.

[Sakakibara *et al.*, 1994] Sakakibara, Y., Brown, M., Hughey, R., Mian, I. S., Sjlander, K., Underwood, R. C., & Haussler, D. (1994). *Nucleic Acids Research (NAR),* **22**, 5112–5120.

[Vapnik, 1995] Vapnik, V. (1995). *The Nature of Statistical Learning.* : Springer.