Non-Monotonic Search Strategies for Grammatical Inference

Sandro Spina and John Abela

Department of Computer Science and AI,
University of Malta

Abstract. Advances in DFA learning algorithms have been relatively slow over the past few years. After the introduction of Rodney Price's EDSM heuristic [4], pushing the limits of DFA learning appears to be a very difficult task. The S-EDSM heuristic proposed in [6, 1], manages to improve slightly on what EDSM can do. In this paper we outline our current research results, and propose the use of non-monotonic search strategies in order to improve the success rate of DFA inference.

1 Introduction

Grammatical Inference (GI) is an instance of inductive inference and can be described as the algorithmic process of discovering syntactic patterns from a corpus of training examples. GI addresses the following problem:

> **Given a finite set of strings that belong to some unknown formal language L, and possibly a finite set of strings that do not belong to L, we require a learning algorithm that infers L.**

Although GI addresses the whole of Chomsky's language hierarchy we are mainly interested in learning regular languages. Machine learning of grammars finds a variety of applications in syntactic pattern recognition, adaptive intelligent agents, diagnosis, computational biology, systems modeling, prediction, natural language acquisition, data mining and knowledge discovery.

However, while the final measure of success of GI algorithms is the performance of these algorithms on real-world data (as in the case of the examples mentioned above), it is important at times to separate ourselves from real world problems and to study the purely theoretical aspects of learning. These studies can aid research in understanding the limits of what can be learned efficiently by machines, as well as help researchers in the design of algorithms that not only perform well in theory, but also perform well in practice.

2 Preliminary Definitions

This section introduces the reader with the terms and definitions used throughout this paper. It is being assumed that the reader is already familiar with the definitions and results in set theory and formal languages, as well as the area of DFA learning in particular state merging DFA learning algorithms. If this is not the case the interested reader is referred to [3] and [6] for an exposure to formal languages and grammatical inference respectively.

2.1 Transition Trees

Transition trees represent the set of string suffixes of a language L from a particular state. Transition trees are *mapped* onto each other by taking the state partitions of two transition trees and joining them into new blocks to form a new state set partition. The mapping operation is recursively defined as follows:

Definition 1 (Map) *A transition tree t_1 is **mapped** onto transition tree t_2 by recursively joining together blocks of the set partitions of t_1 and t_2, for each common string prefix present in t_1 and t_2. The result of this mapping operation is a set partition π, consisting of a number of blocks b. Each block in π, is a set of states which have been merged together.*

2.2 State Compatibility and Merges

States in a hypothesis DFA are either unlabeled or labeled as accepting or rejecting. Two state labels A,B are **compatible** in all cases except when, A is accepting and B is rejecting, or, A is rejecting and B is accepting. Two states are **state compatible** if they have compatible labels. The set of all possible merges is divided between the set of **valid** merges, $\mathcal{M}_\mathcal{V}$, and that of **invalid** merges, $\mathcal{M}_\mathcal{I}$. A valid merge is defined in terms of the transition trees mapping operation as follows:

Definition 2 (Valid Merge) *A **valid merge** M_V in a hypothesis DFA H is defined as (q, q'), where q and q' are the states being merged, such that, the* mapping *of q' onto q results in a state partition π of H, with a number of blocks b, such that for each block $b \in \pi$, all states in b are **state compatible** with each other.*

3 Evidence Driven State Merging (EDSM)

The EDSM DFA learning algorithm developed by Price [4] emerged from the Abbadingo One DFA learning competition organised by Lang and Pearlmutter [4] in 1998. EDSM searches for a target DFA within a lattice of hypotheses (automata) enclosed between the augmented prefix tree acceptor (APTA) and the Universal Acceptor Automaton (UA) [2]. It is assumed that the target DFA lies in the search space of EDSM. It therefore follows, that at least one sequence of merges exists that will lead to the target DFA.

The algorithm starts by constructing an augmented prefix tree acceptor (APTA) and then progressively performing merges. The search is guided by an evidence heuristic which determines which pair of states are to be merged [4]. The heuristic is based upon a score, that computes the number of compatible states found in the transition trees of the two states under consideration. At each iteration, all possible pairs of states in the current hypothesis are evaluated. The pair with the highest evidence score (i.e. compatible states) is chosen for the next merge. This procedure is repeated until no more states can be merged.

4 Shared Evidence Driven State Merging (S-EDSM)

S-EDSM was developed with the aim of improving EDSM's heuristic, thus improving on the path traveled through the lattice of automata. S-EDSM is an attempt at a heuristic that tries to minimise

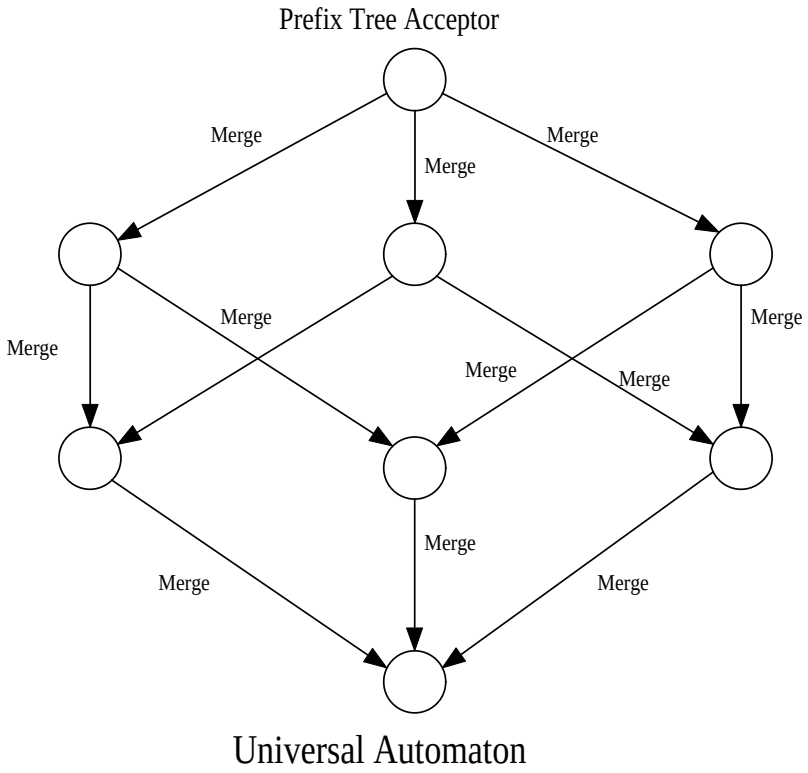

Fig. 1. GI Search Space — A Lattice of Automata

the risk of a merge. Undoubtedly there exist no risk free merges, however as opposed to EDSM, S-EDSM tries to share this risk across multiple merges. The main idea behind S-EDSM is that evidence can be enhanced (augmented) by making use of the knowledge gained when checking how valid merges interact. A heuristic is then developed based on some of the properties derived from this analysis.

The most basic interaction between two merges is compatibility. Merge M is said to be *pairwise compatible* to merge M' if after performing merge M, M' remains a valid merge in the hypothesis automaton as changed by M. More formally two merges are pairwise compatible, if the following property holds:

Definition 3 (Pairwise Compatible) *Let π_1 and π_2 be the state partitions resulting from the application of the map operator to the two merges M_1 and M_2 on hypothesis H. Let H_1 and H_2 be the hypotheses resulting from π_1, π_2 respectively. M_1 and M_2 are **pairwise compatible** if for each state $s \in H$, $s \in H_1$ is state compatible with $s \in H_2$.*

The reader is referred to [6] were merge interactions are further discussed. In practice, S-EDSM has yielded a number of interesting results. These include:

- an improvement on the average classification rate for DFA learning, and
- an improvement on the final hypothesis size, whenever the final hypothesis achieves a classification rate of approximately 50%.

Both S-EDSM and EDSM are monotonic search algorithm. This means that, the algorithms do not have any mechanisms by which they can backtrack to previously visited nodes in the lattice and try to follow different paths. Clearly, if all merging decisions carried out throughout the learning process are correctly taken, then monotonic search is ideal. However, the problem lies in the fact that no algorithm can ever tell *when* a wrong decision has been taken (thus the need for the heuristic to optimise the merge choice). We believe that in order to be able to improve DFA learning algorithms, non-monotonic search strategies (backtracking algorithms) have to be used.

5 Non-Monotonic Search Strategies

GI algorithms are essentially graph-searching algorithms. Up till now our research has focused mainly on depth-first monotonic searches within the lattice of automata. Clearly, this has its advantages in terms of algorithm complexity. However, we also know that when the training sets being used are sparse, this strategy is not always able to appropriately learn the target language[1]. We will thus be introducing S-EDSM within the framework of a best-first search strategy. Consider the following algorithm, A^*, from [5]:

1. Create a search tree, Tr, consisting solely of the start node, n_0. Put n_0 on an ordered list called $OPEN$
2. Create a list called $CLOSED$ that is initially empty.
3. If OPEN is empty, exit with failure.
4. Select the first node on $OPEN$, remove it from $OPEN$, and put it on $CLOSED$. Call this node n.
5. If n is a goal node, exit successfully with the solution obtained.
6. Expand node n, generating a set, $\mathcal{M}$, of successors. Install $\mathcal{M}$ as successors of n in Tr by creating arcs from n to each member of $\mathcal{M}$
7. reorder the list $OPEN$, according to S-EDSM's heuristic scores and states reduction gradient
8. Go to step 3.

The expansion of any particular node n in the search tree, creates the set M which stores the possible merges which can be carried out from node n. These new merges (other nodes in the lattice) are ordered according to the heuristic used in the $OPEN$ list. It is therefore essential to create a heuristic which is able to model how well the lattice is being explored. One very important parameter, which will form part of the heuristic is the 'number of states' gradient[2]. From preliminary experiments, it seems that a heuristic needs to be developed which tries to optimise this gradient. The graph in figure 2, plots the number of states in the hypothesis DFA against the number of merges.

Monitoring the change in states reduction could be a key factor in order to determine when to backtrack and to which node in the search tree to backtrack to. This analysis is currently being carried out.

[1] Refer to [6] for an explanation of GI training sets

[2] Clearly with every merge, states are being absorbed, thus the number of states in the hypothesis DFA decreases

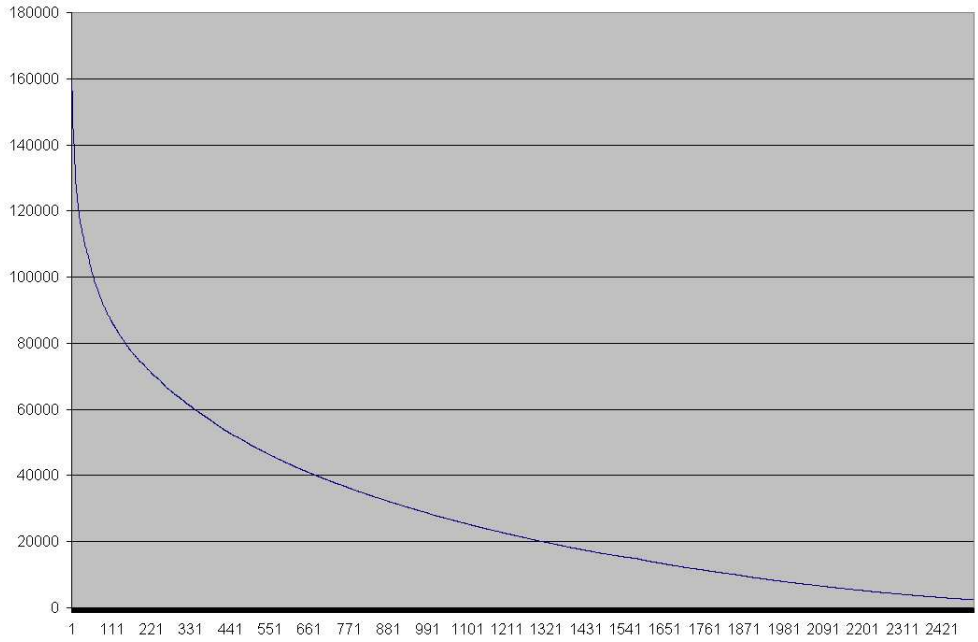

Fig. 2. States Reduction Graph

6 Conclusion and Future Perspectives

Apart from using other search strategies, the idea behind shared evidence has given rise to another algorithm - this time based on evidence of individual states. Different valid merges label states as either accepting or rejecting. Suppose a number of valid merges label a particular state s as accepting. The idea is that of labeling state s (without merging any states initially) as accepting, depending on the number of valid merges which contribute to this evidence. This state labeling continues until there are a sufficient number of states labeled in the APTA. S-EDSM (or EDSM) is then used to infer the target DFA. The study of this algorithm, with which we can gain further knowledge of where labeling errors are being done, is currently in progress.

Very often, real-world data suffers from the presence of noise. Another question which we'd like to pose is - can S-EDSM perform better by combining information between different merges, when noise is present in the training set? In theory, the extra information gathered from merge interactions with S-EDSM can be used to discover noise in the training set. This is definitely another research area, which can benefit from the analysis of merge interactions.

This paper has presented an overview of the research currently being carried out by the Grammatical Inference Research Group (GIRG) at the CSAI department. Our aim is that of designing better DFA learning algorithms which further push the boundaries of DFA learning.

References

1. John Abela, Francois Coste, and Sandro Spina. Mutually compatible and incompatible merges for the search of the smallest consistent dfa. *Grammatical Inference: Emphasizing on Innovate Applications. ICGI 2004.*, 2004.

2. P. Dupont, L. Miclet, and E. Vidal. What is the search space of the regular inference ? In *Grammatical Inference and Applications, ICGI'94*, number 862 in Lecture Notes in Artificial Intelligence, pages 25–37. Springer Verlag, 1994.
3. J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison Wesley, Reading, Massachusetts, 1979.
4. K. Lang, B. Pearlmutter, and R. Price. Results of the abbadingo one dfa learning competition and a new evidence-driven state merging algorithm. *Grammatical Inference. ICGI 1998.*, LNAI 1433:1–12, 1998.
5. Nils J. Nilsson. Morgan Kaufmann, United States of America, 1998.
6. Sandro Spina. Merge heuristics : A new heuristic for automata learning, 2004.