Search Diversification Techniques for Grammatical Inference

Sandro Spina

Department of Computer Science and Artificial Intelligence
University of Malta

Abstract. Grammatical Inference (GI) addresses the problem of learning a grammar G, from a finite set of strings generated by G. By using GI techniques we want to be able to learn relations between syntactically structured sequences. This process of inferring the target grammar G can easily be posed as a search problem through a lattice of possible solutions. The vast majority of research being carried out in this area focuses on non-monotonic searches, i.e. use the same heuristic function to perform a depth first search into the lattice until a hypothesis is chosen. EDSM [**?**] and S-EDSM [6] are prime examples of this technique. In this paper we discuss the introduction of *diversification* into our search space [5]. By introducing diversification through pairwise incompatible merges, we traverse multiple disjoint paths in the search lattice and obtain better results for the inference process.

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 data. 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.

Machine learning of grammars finds a variety of applications in syntactic pattern recognition, diagnosis, computational biology, systems modeling, prediction, natural language acquisition, data mining and knowledge discovery.

One of the main challenges in DFA learning algorithms is that of learning grammars from as sparse as possible training set of that particular grammar. In real-life scenarios the amount of data available for the learning task might be very limited. Both the Evidence Driven State Merging (EDSM) [**?**] and the Shared Evidence Driven State Merging (S-EDSM) [6, 1] heuristic aim at improving target convergence when using sparse data. Sharing the evidence gathered from different potential merges has in effect improved the convergence rate of grammatical inference. The interested reader is referred to [6] for a detailed description of how the S-EDSM heuristic works.

A number of DFA learning tasks still remain very difficult to solve. This difficulty measure depends both on the size of the target automata and the size of the training set. Figure 1 refers to the current Gowachin map for DFA problems [1]. The graph plots the size of the target automata against the number of training strings. The sparser the training set the more difficult it is for a learning algorithm to identify the target grammar.

[1] http://www.irisa.fr/Gowachin/

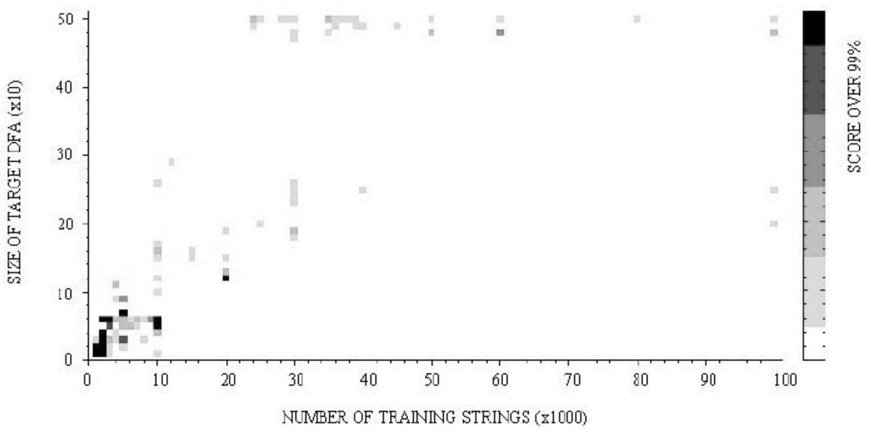

Fig. 1. Gowachin Map

2 Search Diversification

This section discusses the motivations behind our introduction of search diversification techniques. 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 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. We shall start by defining what state compatibility means followed by pairwise compatibility of merges. We then explore how search diversification can be obtained by taking into account *pairwise incompatibility* of merges.

2.1 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 [6] as follows:

Definition 1 (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.*

Choosing which states to merge next depends on the heuristic adopted. Both EDSM and S-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) [**?**]. 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.

With search diversification we explore more than one sequence of merges. Our goal is to order these sequences in such a way such that the optimal one (i.e. the one producing the target grammar) is produced in the first few sequences.

2.2 Diversification through Pairwise Incompatible Merges

The most basic interaction between two merges is *pairwise 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 2 (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 $s \in H_2$.*

Suppose we have a set $\mathcal{S}$ of high scoring merges generated by the EDSM or S-EDSM algorithms. Each merge M in $\mathcal{S}$ might not necessarily be pairwise compatible with all the other merges in the same set $\mathcal{S}$. Let merges $M1$ and $M2$ be two pairwise incompatible merges from the set of merges $\mathcal{S}$. As a result of their incompatibility we know that the merge path followed after $M1$ is carried out in the DFA lattice can never cross the merge path which follows $M2$. Thus we can state that pairwise incompatible merges guarantee diversification of the search paths.

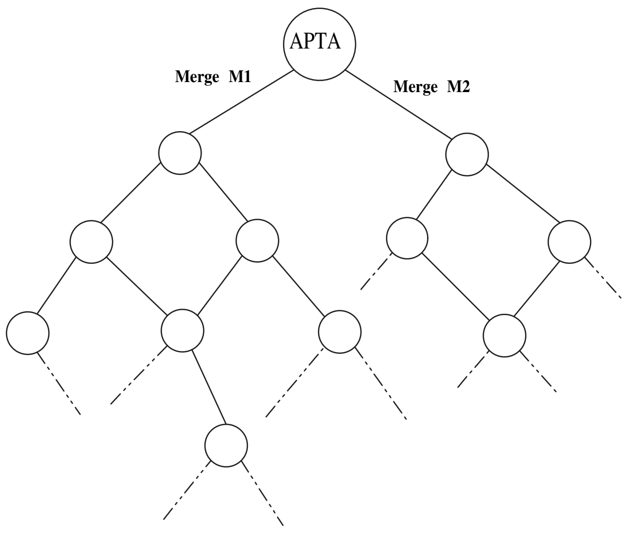

Fig. 2. Paths in DFA Lattice

Figure 2 gives a simple illustration of how performing merges $M1$ and $M2$ produces two disjoint paths in the DFA lattice. Now suppose that after performing $M2$ there are two further merges (at the same level), we can further diversify our search paths by performing these two merges. When doing so we are increasing our probabilities of inferring the target grammar.

Clearly, by increasing the degree of diversification (i.e. the number of disjoint lattice paths explored) we are also increasing the amount of computation required. For this reason a distributed application has been developed in [5]. A central authority would determine which path in the lattice each daemon connected to the system has to traverse. Finally each daemon sends back its hypothesis and the central authority of the system decides which of the received hypothesis is closest to the target DFA (by comparing the number of states of the hypotheses with the number of states of the target DFA).

The following example illustrates the notation that has been developed to describe diversification strategies.

```
M1:M1
M2:PI(M1)
M3:PI(M1 ∧ M2)
M4:F(M2)
M5:F(M2) ∧ PI(M4)
```

In this example $M1$ is carried out first. $M2$ is then carried out at the same level of $M1$. Note that $M2$ is pairwise incompatible with $M1$. $M3$ is pairwise incompatible with both $M1$ and $M2$ and is carried out at their same level. The algorithm then performs further diversification on the node following merge $M2$. The two paths followed are $M4$ and $M5$. Note that $M4$ simply follows from $M2$, while $M5$ follows after $M2$ and is also pairwise incompatible with $M4$. Various search diversification strategies can easily be described through this notation.

3 Results

This section presents some preliminary results of our search diversification strategies. The training sets of all the problems used for experimentation were downloaded from the Gowachin server. 60 state DFAs with 4000 training examples have been used with the following diversification strategy.

```
M1:M1
M2:PI(M1)
M3:PI(M1 ∧ M2)
```

The EDSM heuristic was used to order the valid merges. The following table illustrates the classification rate for five DFA learning problems. Three daemons were used to traverse the three different search paths. In these examples diversification has been carried out on the APTA node of the DFA lattice.

Problems	Daemon 1	Daemon 2	Daemon 3
1	100	99.1	98
2	97.8	99.3	**99.5**
3	59.1	90.4	**92.9**
4	100	98.4	99.6
5	100	99.2	99.2

It is interesting to note that in two occasions (out of five), the third daemon performed better than the first one. The first daemon performs the standard EDSM algorithm (i.e. always traversing the highest scoring merge), while the third daemon traverses a path which starts with a merge which is incompatible with the first two merges. This indicates that when introducing search diversification through pairwise incompatible merges we are able to considerable improve on what EDSM and S-EDSM can do.

For a complete listing of results the reader is referred to [5].

4 Conclusion and Future Perspectives

Search diversification is clearly an important technique to exploit in DFA learning. Instead of focusing solely on improving the heuristic by which merges are ordered, we are also exploiting a heuristic which orders different merge sequences in the DFA lattice. This heuristic is based on pairwise incompatibility of merges. Further research in this area is currently being carried out.

Another area which is currently being researched is that of developing algorithms which perform well when noise is present in the training set. There is currently no algorithm which is capable of learning a 50 state DFA from a training set of 5000 strings with 10% noise.

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. Stephen Pace. Search diversication and backtracking strategies for dfa learning, 2005.
6. Sandro Spina. Merge heuristics : A new heuristic for automata learning, 2004.