Inducing Regular Languages Using Grammar-Based Classifier System

Olgierd Unold

Institute of Computer Engineering, Control and Robotics,
Wroclaw University of Technology, Wyb. Wyspianskiego 27, 50-370 Wroclaw, Poland
olgierd.unold@pwr.wroc.pl
http://olgierd.unold.staff.iiar.pwr.wroc.pl/

Abstract. This paper takes up the topic of a task of training Grammar-based Classifier System (GCS) to regular grammars from data. GCS is a new model of Learning Classifier Systems in which the population of classifiers has a form of a context-free grammar rule set in a Chomsky Normal Form. Near-optimal solutions or better than reported in the literature were obtained.

Keywords: Grammatical Inference, Learning Classifier Systems, FSA Induction.

1 Introduction

In this paper, we are interested in inducing a grammar that accepts a regular language (RL) given a finite number of positive and negative examples drawn from that language. The approaches to learning RL or equivalent Deterministic Finite Automata (DFA) base mainly on evolutionary algorithms, recurrent neural network or combination of these two methods. It has been proved that RL/DFA induction is a hard task by a number of criteria.

This paper addresses RL induction using Grammar-based Classifier System (GCS) [5] - a new model of Learning Classifier System (LCS). LCS is machine learning paradigm introduced by Holland [2]. It exploits evolutionary computation and reinforcement learning to develop a set of condition-action rules (the classifiers) which represent a target task that the system has learned from online experience. Although there are some approaches to handle with context-free grammar (CFG), there is no one work on inducing RLs with LCSs.

GCS [5] evolves population of classifiers in a form of a CFG rule set, each rule in a Chomsky Normal Form (CNF). CNF allows only production rules in the form of $A \rightarrow a$ or $A \rightarrow BC$, where A, B, C are the non-terminal symbols and a is a terminal symbol. The first rule is an instance of terminal rewriting rule not affected by the genetic algorithm (GA), and generated automatically as the system meets unknown terminal symbol. Left hand side of the rule plays a role of classifier's action while the right side a classifier's condition. All classifiers form a population (one CFG) of evolving individuals. Environment of classifier system is made up by an array of Cocke-Younger-Kasami parser. GCS matches

A. Clark, F. Coste, and L. Miclet (Eds.): ICGI 2008, LNAI 5278, pp. 298–300, 2008.

the rules according to the current environmental state (state of parsing) and generates an action/actions pushing the parsing process toward the complete derivation of the sentence analyzed. Apart from the GA, the *covering* procedure explores the space searching for new, better productions. We refer the reader to [5] for more details of GCS.

2 Regular Language Induction Using GCS

The datasets most commonly used in DFA learning is Tomita sets. In this paper GCS will be compared against the evolutionary methods proposed by Lucas and Reynolds [3] and Luke et al. [4]. Both methods present one of the best-known results in the area of RL/DFA induction. All of compared evolutionary methods will assume the same training and test sets. Some comparisions will be made also to EDSM method [1], the current most powerful passive approach to DFAs inference. Fifty independent experiments were performed, evolution on each training corpus ran for 5,000 generations, with the following GCS parameters: number of nonterminal symbols 19, number of terminal symbols 7, crossover probability 0.2, mutation probability 0.8, population consisted of maximal 40 classifiers where 30 of them were created randomly in the first generation, crowding factor 18, crowding size 3. The approach presented in [4] (denoted by GP) applies gene regulation to evolve DFA. In this approach genes are states in the automaton, and a gene-regulation-like mechanism determines state transitions. Each gene has Boolean value indicating whether or not it was an accepting state. The main results are summarized in Table 1. For compared methods induction of L3 language appeared to be hard task. Both in GP and in GCS only the one run over 50 successfully finished. But GP found the solution in 12450 iterations, whereas GCS in only 666 steps. For the same language GCS correctly classified all of the unseen examples, while GP achieved 66%. As to an indicator nGen, GP was not able correctly classified unseen strings for any language from the tested corpora, while GCS induced a grammar fully general to the language in 4 cases. It is interesting to compare the results of induction for L5 language. GP approach could not find the proper grammar (DFA) for any run, while GCS found the solution in all runs, on average in 201 steps. While learning L1 and L2 languages, GP found the proper grammars not in all runs, whereas for GCS this task appeared to be trivial (100% nGen, 50/50 nSuccess, and nEvals 2 steps) Table 1 shows also the cost of induction (an indicator nEvals) for the methods Plain, Smart (Sm), and nSmart (nSm) taken from [3], GP approach, and GCS. Lucas i Reynolds [3] used different method to evolving DFA. In contrary to [4], only transition matrix was evolved, supported by a simple deterministic procedure to optimally assign state labels. This approach is based on evolutionary strategy (1+1). Three versions of induction algorithm were prepared: an approach in which both the transition matrix and the state label vector evolve (Plain), so-called Smart method evolving only the transition matrix and the number of the states was fixed and equal to 10, and finally nSmart method in which the number of the DFA states is equal to the size of minimal automata. GCS obtained the best results for the

Table 1. Comparison of GCS with different evolutionary methods and non-evolutionary EDSM. For each learning corpus, the table shows the target language, and three sets of results. The first indicator nSuccess is the number of runs with success gained by GCS within 50 experiments and compared approach. The second one nGen is the percentage of all unseen strings correctly classified, and the last one nEvals indicates the average number of generations needed to reach the 100% fitness.

Lang.	nSuccess		nGen					nEvals				
	GP	GCS	GP	GCS	Sm	nSm	EDSM	GP	GCS	Plain	Sm	nSm
L1	31/50	50/50	88.4	100	81.8	100	52.4	30	2	107	25	15
L2	7/50	50/50	84.0	100	88.8	95.5	91.8	1010	2	186	37	40
L3	1/50	1/50	66.3	100	71.8	90.8	86.1	12450	666	1809	237	833
L4	3/50	24/50	65.3	100	61.1	100	100	7870	2455	1453	177	654
L5	0/50	50/50	68.7	92.4	65.9	100	100	13670	201	1059	195	734
L6	47/50	49/50	95.9	96.9	61.9	100	100	2580	1471	734	93	82
L7	1/50	11/50	67.7	92.0	62.6	82.9	71.9	11320	2902	1243	188	1377

L1 and L2 languages among comparable methods. The result 201 steps for L5 is comparable with the best result of 195 reached by nSmart. Although GCS reached similar result for language L3 as the best method (666 for GCS, and 237 for Smart), it is hard to compare for this language these methods, because of low value of nSuccess for GCS - only one run over 50 finished with success. For the languages L4, L6, and L7 fixed-size structured methods (Plain, Smart, and nSmart) achieved better results than variable-size methods (GP and GCS). Finally, Table 1 shows the percentage of all unseen strings correctly classified (an indicator nGen) for the methods Smart, nSmart, EDSM, GP, and GCS. Recall that the EDSM, as a heuristic and non-evolutionary method, was single-time executed during learning phase. Model GCS achieved the best results from all tested approaches for L1, L2, L3, and L7 languages. For the language L4 the same 100% accuracy was obtained by proposed method, nSmart, and EDSM. For the L5 and L6 languages GCS obtained the second result, higher than 90%.

References

1. Cicchello, O., Kremer, S.C.: Beyond EDSM. In: Adriaans, P.W., Fernau, H., van Zaanen, M. (eds.) ICGI 2002. LNCS (LNAI), vol. 2484, pp. 37–48. Springer, Heidelberg (2002)
2. Holland, J.: Escaping Brittleness: The possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems. In: Michalski, R.S., et al. (eds.) Machine Learning, an Artificial Intelligence Approach, vol. II, pp. 593–623. Morgan Kaufmann, San Francisco (1986)
3. Lucas, S., Reynolds, T.J.: Learning DFA: Evolution versus Evidence Driven State Merging. In: Proc. Congress Evolutionary Computation, pp. 351–358 (2003)
4. Luke, S., Hamahashi, S., Kitano, H.: 'Genetic' Programming. In: Banzhaf, W., et al. (eds.) Proc. Genetic and Evolutionary Computation Conf., pp. 1098–1105 (1999)
5. Unold, O.: Context-free grammar induction with grammar-based classifier system. Archives of Control Science 15 (LI)(4), 681–690 (2005)