<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>A GAuGE Approach to Learning DFA from Noisy Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Authors(s)</strong></td>
<td>Nicolau, Miguel; Ryan, Conor; Ryan, Eoin</td>
</tr>
<tr>
<td><strong>Publication date</strong></td>
<td>2004-06-30</td>
</tr>
<tr>
<td><strong>Conference details</strong></td>
<td>Genetic and Evolutionary Computation - GECCO 2004: Genetic and Evolutionary Computation Conference, Seattle, Washington, USA, 26-30 June 2004</td>
</tr>
<tr>
<td><strong>Item record/more information</strong></td>
<td><a href="http://hdl.handle.net/10197/8339">http://hdl.handle.net/10197/8339</a></td>
</tr>
</tbody>
</table>
A GAuGE Approach to Learning DFA from Noisy Samples

Miguel Nicolau, Conor Ryan and Eoin Ryan

Biocomputing and Developmental Systems Group
Computer Science and Information Systems Department
University of Limerick, Ireland

Abstract. This paper describes the adaptation of the GAuGE system to classify binary sequences generated by random DFA. Experiments were conducted, which, although not highly successful, illustrate the potential of applying GAuGE like systems to this problem domain.

1 The Problem

The problem was stated as follows. Given a training set of binary sequences, each with a binary class label, the system should generate a predictor that classifies unlabelled sequences in a given test set. The training and test sets consisted of random binary sequences, labelled by a randomly constructed Deterministic Finite Automata (DFA). Each system was allowed to run for 10 minutes.

Although the training and test sets were generated by fairly small DFA (10 to 50 states), the training set had a high level of noise (10%): this noise was introduced by mutating each label in the training set with a probability of 0.1. The size of the training sample was also moderate: 1000 instances for data generated by a DFA with 10 states, 2000 for a DFA with 20 states, etc.

2 Implementation

As all implementations were accepted, we decided to adapt the GAuGE system (Genetic Algorithms using Grammatical Evolution) [2, 1], to generate DFA classifiers. Using a standard genetic algorithm to evolve binary strings, a GAuGE-like mapping was then applied. For example, to generate DFA with 10 states, strings of 90 bits were used. These were then divided onto 10 sections, of 9 bits each, which represented the 10 states (0 to 9) of the DFA to be generated.

From each group of 9 bits, the first four were used to encode state0, which specified the state to go to if the input is a 0; the next four encoded state1, specifying the state to go if the input is a 1; and the last remaining bit encoded fieldA, which specified if this was an accepting state.

To ensure each of state0 and state1 were within range, the mod operator was used: each field was modded by 10, to make sure they represented a valid state.
Figure 1 illustrates the latter part of this process. Once the binary string has been segmented and all fields have been normalised, a DFA can then be constructed. Note that this process is quite similar for DFA with 20, 30, 40 and 50 states; only the initial size of the binary strings and the number of bits to encode each field need to change.

2.1 Experimental setup

The experimental setup was the standard used with GAuGE: r.-wheel selection, steady-state replacement, crossover probability of 0.9, and (point) mutation probability of 0.01. The population size was 1600 individuals, the same for all DFA sizes. No maximum number of generations was set: instead, if a run would converge, a new run would then start, with a different random seed, until the 10 minutes of run-time were over. A run was said to converge when the average fitness of the population did not improve for two consecutive generations.

3 Results

Using the provided data sets, the performance of the system could be measured. The results were encouraging for size 10, but with sizes 20 through 50, the system could not find solutions within 10 minutes. Probable explanations could be early convergence, or the need of genetic operators adapted to the problem domain.

References
