The Compact Classifier System: Motivation, Analysis, and First Results

Xavier Llorà
Illinois Genetic Algorithms Lab
University of Illinois at Urbana-Champaign
104 S. Mathews Ave., Urbana, IL 61801, USA
xllora@illigal.ge.uiuc.edu

Kumara Sastry
Illinois Genetic Algorithms Lab
University of Illinois at Urbana-Champaign
104 S. Mathews Ave., Urbana, IL 61801, USA
kumara@illigal.ge.uiuc.edu

David E. Goldberg
Illinois Genetic Algorithms Lab
University of Illinois at Urbana-Champaign
104 S. Mathews Ave., Urbana, IL 61801, USA
deg@illigal.ge.uiuc.edu

ABSTRACT

This paper presents an initial analysis of how maximally general and accurate rules can be evolved in a Pittsburgh-style classifier system. In order to be able to perform such analysis we introduce a simple bare-bones Pittsburgh classifier systems—the compact classifier system (CCS)—based on estimation of distribution algorithms. Using a common rule encoding scheme of Pittsburgh classifier systems, CCS maintains a dynamic set of probability vectors that compactly describe a rule set. The compact genetic algorithm is used to evolve each of the initially perturbed probability vectors which represents the rules. Results show how CCS is able to evolve in a compact, simple, and elegant manner rule sets composed by maximally general and accurate rules.

Categories & Subject Descriptors

I.2.6 [Artificial Intelligence]: Learning–Concept Learning.

General Terms

Algorithms, Design, Theory.

Keywords

Learning Classifier Systems, Maximally General Classifiers, Compact Classifier System, Spawning and Merging Populations.

1. INTRODUCTION

The work of Wilson in 1995 [7] was the starter of a major shift on the way that fitness was computed on classifier systems of the so call Michigan approach. Accuracy became a central element in the process of computing the fitness of rules (or classifiers). With the inception of XCS, the evolved rules targeted became the ones that were maximally general (cover a large number of examples) and maximally accurate (lower error). After a decade since the paper published by Wilson, such road has been shown to be a successful one. However, few attempts have been made to do the same revision exercise on the Pittsburgh-style classifier systems. This paper revisits Wilson's work and applies some of the his original ideas to Pittsburgh-style classifier systems. We present how such rules may be evolved using a simple bare-bones Pittsburgh-style classifier systems—the compact classifier system (CCS)—based on estimation of distribution algorithms. The initial results achieved show that the encoding scheme of the rules and the bias introduced in the process play a central role on the overall performance and scalability of CCS—and by extension to other Pittsburgh-style classifier systems.

2. LEARNING MAXIMALLY GENERAL AND ACCURATE RULES

In order to promote maximally general and accurate rules *a la* XCS [7], we need to compute the *accuracy* of a rule (α) and its *error* (ε). In a Pittsburgh-style classifier, the *accuracy* may be computed as the proportion of overall examples correctly classified, whereas the *error* is the proportion of incorrect classifications issued by the activation of the rule. Such a measure is independent of the rule encoding used [6]. For computation simplicity we assume $\varepsilon(r) = 1$ when all the predictions are accurate, and $\varepsilon(r) = 0$ when all were incorrectly issued. Let n_{t+} be the number of positive examples correctly classified, n_{t-} the number of negative examples correctly classified, n_m the number of times that a rule has been matched, and n_t the number of examples available. Using this values the *accuracy* and *error* of a rule r can be computed as:

$$\alpha(r) = \frac{n_{t+}(r) + n_{t-}(r)}{n_t} \quad (1)$$

$$\varepsilon(r) = \frac{n_{t+}}{n_m} \quad (2)$$

It is worth to note that the error (equation 2) only take into account the number of correct positive examples classified[1]. This is a byproduct of the close world assumption of this knowledge representation. Once the *accuracy* and *error* of a

[1]We also assume that if a rule is never matched, no error is made and, hence, $\varepsilon(r) = 1$.

rule are known, the fitness can be computed as follows.

$$f(r) = \alpha(r) \cdot \varepsilon(r)^{\gamma} \tag{3}$$

Such fitness favors rules with a good classification accuracy and a low error, or maximally general and accurate rules. Throughout the rest of this paper we assume $\gamma = 1$. Traditional Pittsburgh-style classifier systems mainly relied on some sort of fitness based only on equation 1 [1, 3, 5], introducing no bias toward maximally general and accurate rules.

3. THE COMPACT CLASSIFIER SYSTEM

Evolving a rule set may be regarded as a multimodal optimization of the particular fitness function at hand—equation 3—and some approaches have been proposed [4]. Unfortunately, such approaches have two significant drawbacks after the Pittsburgh-style literature. The first one is that the maximum number of possible rules is fixed *a priori*. The second drawback is the parallel evolution of the different rules of the chromosome. Each rule evolves on a particular direction, thus is possible that the same rule appears more than once in the rule set.

In order to evolve maximally general and accurate rules, we use the compact genetic algorithm (cGA) [2]. It is important to mention here, that such algorithm does not provide any nitching capability and, hence, will only produce one accurate and maximally general rule after optimizing equation 3. A first step toward rule set learning relies on perturbating the initial probability vector with an uniform noise. Several runs of cGA using different perturbed initial probability vectors may lead to different accurate and maximally general rules. Instead of the initial cGA probability vector $p^0_{x_i} = 0.5$, we used

$$p^0_{x_i} = 0.5 + \mathcal{U}(-0.4, 0.4). \tag{4}$$

The probability vector evolved by the cGA represents a population of candidate rules treated as a single optima problem. If we want to evolve a rule set that contains n rules, then n probability vectors are required. This leads to two important questions: (1) when to spawn a new probability vector, and (2) when to fuse two probability vectors because they describe the same rule.

The simplest spawning criteria we use in the CCS is the rule set fitness. Given a set $\mathcal{D}$ of probability distribution vectors, we can form a rule set $\mathcal{R}$ by sampling each of them. Then, if $f(\mathcal{R}) < 1.0$ then the problem is not accurately solved and, thus, a new probability distribution vector needs to be spawned and added to $\mathcal{D}$. Our current research assumes noise-free problems with perfect information, such as the multiplexer. Further research will be needed in the presence of noise, since $f(\mathcal{R}) < 1.0$ may not be achievable due to noise interference. On the other hand, the criteria to decide when to fuse two probability vectors in CCS uses rule information instead. Based on such mechanism, we can assume that we can compute—using a simple Euclidean distance how different—two probability vectors are—$d(p_i, p_j)$. Moreover, since we deal with a binary encoding, there is a threshold θ below which they represent the same rule population. Two probability vectors may be different if the is a chance that at least one bit is different. Hence, the strict value for θ is $1/\ell$. Relaxing θ value may lead to an implicit bounding of the maximum number of different probability vectors in $\mathcal{D}$. Table 1 presents the algorithmic description of CCS, integrating the required elements to evolve a rule set composed of maximally general and accurate rules.

Table 1: Algorithmic description of the CCS.

```
1.  $\mathcal{D} \leftarrow \{pert(p_0), \ldots, pert(p_k)\}$.
2.  Foreach $p_i \in \mathcal{D}$ run cGA.
3.  $\mathcal{R} \leftarrow \{r_i$ sampled from $p_i\}$.
4.  Compute $f(\mathcal{R})$ using equation 3.
5.  If given $p_i, p_j \in \mathcal{D}$ and $d(p_i, p_j) < \theta$
    then $\mathcal{D} \leftarrow \mathcal{D} \setminus \{p_i\}$.
6.  If $f(\mathcal{R}) = 1.0$ return $\mathcal{R}$
    else $\mathcal{D} \leftarrow \mathcal{D} \cup \{pert(p)\}$ and goto 2.
```

4. CONCLUSIONS

Preliminary results showed that the scalability of CCS—and by extension, of other Pittsburgh-style systems—using the multiplexer problem tend to scale exponentially with the number of address bits [6]. For each multiplexer problem there are, at least, 2^k maximally accurate rules, where k is the number of address bits. Another scalability factor unveiled by CCS is the rule encoding scheme. The encoding scheme proposed by De Jong & Spears [1] adds an extra element of difficulty as the problem size increases, due to the exponential growth of unmatchable rules [6].

5. ACKNOWLEDGMENTS

This work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF (F49620-03-1-0129), and by the Technology Research, Education, and Commercialization Center (TRECC), at University of Illinois at Urbana-Champaign (N00014-01-1-0175).

6. REFERENCES

[1] K. A. De Jong and W. M. Spears. Learning Concept Classification Rules using Genetic Algorithms. In *Proceedings of the Twelfth International Conference on Artificial Intelligence IJCAI-91*, volume 2, pages 651–656. Morgan Kaufmann, 1991.

[2] G. Harik, F. Lobo, and D. E. Goldberg. The compact genetic algorithm. *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 523–528, 1998. (Also IlliGAL Report No. 97006).

[3] C. Janikow. A Knowledge Intensive Genetic Algorithm for Supervised Learning. *Machine Learning*, 13:198–228, 1993.

[4] P. Larrañaga and J. A. Lozano, editors. *Estimation of Distribution Algorithms.* Kluwer Academic Publishers, Boston, MA, 2002.

[5] X. Llorà and J. Garrell. Knowledge-Independent Data Mining with Fine-Grained Parallel Evolutionary Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 461–468. Morgan Kaufmann Publishers, 2001.

[6] X. Llorà, K. Sastry, and D. E. Goldberg. Binary Rule Encoding Scheme: A Study Using The Compact Classifier System. In *International Workshop on Learning Classifier Systems (IWLCS 2005), accepted*, 2005.

[7] S. W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.