Artificial Data Sets based on Knowledge Generators: Analysis of Learning Algorithms Efficiency

Joaquin Rios-Boutin[1], Albert Orriols-Puig[1], Josep-Maria Garrell-Guiu[1]
[1]*Grup de Recerca en Sistemes Intel·ligents*
Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, 08022, Barcelona, Spain
{jrios, aorriols, josepmg}@salle.url.edu

Abstract

This paper proposes a methodology to generate artificial data sets to evaluate the behavior of machine learning techniques. The methodology relies in the definition of a domain and the generation of data sets from this domain by means of different sampling processes. Then, learners are trained with the generated data sets and the created models are compared with the original domain to evaluate the quality of the learners. In the present work, a particular implementation of this methodology is provided, which is defined to test learning techniques that use a binary rule knowledge representation. As a case study, the behavior of XCS, the most influential learning classifier system, is analyzed following the methodology.

1. Introduction

Supervised learning systems are machine learning techniques concerned about learning classification models from sets of training data which represent real-life concepts. During the last few decades, the increasing amount of research in machine learning has resulted in the design of several competent supervised learners. The performance of these learners is usually evaluated by providing some performance indicators–such as the test accuracy–of the methods on several real-life data sets extracted from data repositories. Whilst this methodology supplies an initial idea of the excellence of the learners on the tested domains and enables comparisons among different learning techniques, it does not indicate if the learning method has been able to represent the original concept.

In the rule-based systems realm, several authors have proposed to analyze their learning systems on collections of artificial problems that permit to vary the complexity along different dimensions [1][2]. As the initial distribution of the data is known, rule sets can be scanned to analyze whether the knowledge represented in the rule set is equivalent to the initial distribution. This has resulted in several studies that detected anomalies in existing learning systems and lead to the design of new approaches to overcome this problem.

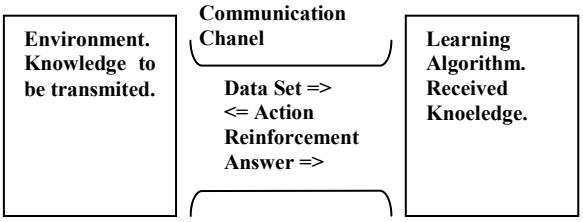

Fig. 1 Learning Process as Communication System

The purpose of this paper is to define a systematic methodology for the evaluation of rule-based systems. For this purpose, we design a procedure to create artificial multi-class data sets with binary attributes, and then use these data sets to evaluate the quality of rule-based classification systems. We consider the machine learning process as a communication system, as illustrated in Fig. 1. That is, learning acquisition involves a communication change through which a learning algorithm acquires the same knowledge that is present in the environment. Based on this idea, we propose a methodology in which an initial knowledge Ω is defined by the user to represent the environment domain. Then, several sampling techniques are developed to generate different data sets from the domain Ω. Finally, rule-based systems are trained with these data sets; the quality of the learned knowledge is evaluated according the knowledge contained in the environment. Note that this methodology serves to evaluate how a given learners acquires the knowledge provided different samplings of the same original domain. We experimentally show the results obtained with XCS [4][5], an evolutionary rule-based system that evolves binary rules. The results show that the proposed methodology enables a careful examination of the quality of the created models.

The remainder of this paper is organized as follows. Section 2 defines the entire methodology and Sect. 3 specifies the different sampling methodologies that we

propose. Section 4 briefly describes XCS, the learning technique used as a case study for our methodology. Section 5 analyzes the quality of the rule sets evolved by XCS based on the original knowledge. Finally, Section 6 summarizes, concludes, and discusses future work lines that will be followed.

2. Evaluation Methodology

In this section, we propose the general framework for generating artificial data sets and evaluate, based on the original domain, the quality of the models created by learning techniques. Figure 2 supplies a schematic of the framework. The evaluation methodology takes as starting point a knowledge representation of the domain. For this purpose, we could use several representations as decision trees or discriminative functions among others. In our particular implementation, we used rule sets with binary variables to represent the domain since in this paper we evaluate rule-based classification models. That is to say, the original domain is defined by a set of *n* binary rules of length ℓ that predict a certain class, where each variable of a rule can take the values '0', '1', or '#'. The *don't care* symbol '#' allows for generalization, and indicates that it does not care the particular value for this variable. Without loss of generalization, this rule representation could also be extended to continuous domains.

After defining the domain, data sets are generated by different sampling techniques which define a procedure to create instances from the original domain. Therefore, sampling techniques generate different data sets that explain the same concept with the aim of evaluating whether the classifier can learn the same original concept with different samples. We defined four sampling methods which are detailed in the following section.

Finally, the chosen learning algorithm is trained with the sampled data sets, and the resulting rule sets for each sample are compared to the original domain. Then, we evaluate the performance of the learner by analyzing the proportion of the original concept the learner has been able to learn for each data set sample. Therefore, this methodology allows for two types of analysis. On one hand, it permits to analyze whether the classifier is able to learn the original domain and whether the sampling methodologies cause any effect in either during the learning process or in rules contained in the final population. On the other hand, it also enables the comparison of the knowledge evolve by several learners. Notice that this methodology permits to analyze both the accuracy of the population and the adequacy of the different rules.

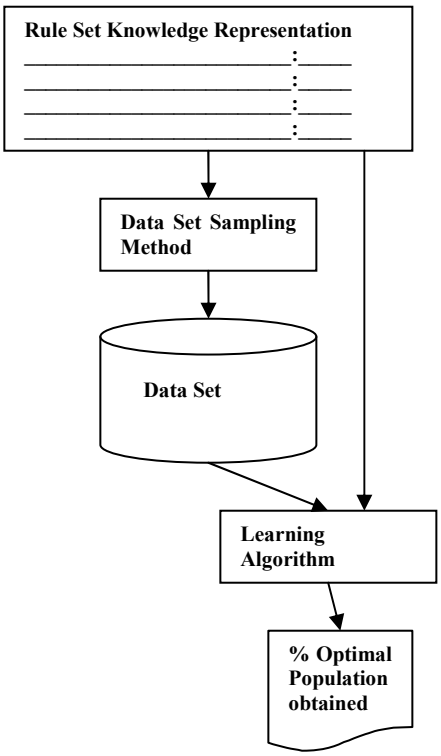

Fig. 2 DS Generation Method Analysis Framework

3. Sampling Methods

Sampling methods depart from the rules that define the original knowledge and create a set of instances that matches these rules. Note that an instance would have all bits specified. Therefore, the main differences between the proposed sampling techniques are related to how they select the original rules and generate training examples. We defined four different methods to sample the original training domain. These methods are referred to as SRS, SIS, RRS and RIS. As follows, we explain each one of the methods:

- The **SEQUENTIAL RULE SELECTION (SRS)** method selects each rule sequentially and generates a random example that matches the rule. That is, for a given rule, the example contains the specific bits with the same value as in the rule and contains random values for the position of the rule whose value is a "don't care". This generation method obtains random samples for each rule of the domain concept. This sampling method may alter the proportion of instances in the sample with respect to the original distribution of data. This happens when the rules of the target concept have different levels of generalization. Its study is interesting

to see how the possible alteration of data distribution may alter the learnt concept.

- The **SEQUENTIAL INSTANCE SELECTION (SIS)** method generates all samples that satisfy each rule. The process is similar to SRS, but instead of obtaining a random sample for each rule, all possible samples are obtained. This sampling method represents the case where all possible instances of the target concept are contained in the data set. Although this is not usual in real-world problems, its study is useful to see the classifier's response to the complete concept sampling.
- The **RANDOM RULE SELECTION (RRS)** method randomly selects a rule and generates a single instance by randomly specifying the "don't care" bits of the rule. This is somehow similar to SRS but rules are selected randomly, so there is not any guarantee that the training sample would contain a representative of each disjoint. As in SRS, the original data and the sampled data may have different distributions.
- The **RANDOM INSTANCE SELECTION (RIS)** method selects a rule with a probability proportional to the number of examples contained in each rule. Then, a random sample which satisfies the selected rule is obtained. This could be similar to the sampling processes of real-world problems, because one expects that the samples are obtained with the same distribution as the original concept. Most of current data sets generators use this method.

Note that the above sampling methods sample one or several examples at each iteration. The process is repeated until the dataset contains the required number of instances.

Each of these methods is defined to sample the target concept expressed by a disjunction of rules, although this could easily be extended to other types of knowledge representations.

After defining the four generation methods, we have specified all the elements of the methodology. The next section briefly introduces XCS, since it is used in the subsequent sections to provide a case study with the proposed methodology. We selected XCS for the analysis since it represents the knowledge as production rules, which can be easily visualized and examined. Similarly, other machine learning techniques that use a knowledge representation alike could be used for this study.

4. XCS in a Nutshell.

This section provides a brief description of the XCS classifier system. For a detailed description, the reader is referred to [5, 6].

XCS is an accuracy-based learning classifier system introduced in [5] that computes fitness from the accuracy of the reward prediction instead on the reward itself. The accuracy-based approach makes XCS evolve a complete action map (denoted as [O] [4]) of the environment, evolving not only high-rewarded rules (i.e., consistently correct rules), but also consistently incorrect rules (i.e., rules with zero prediction and low error). XCS works as a model-free online learner. For each input example, XCS forms the match set [M] consisting of all classifiers with matching condition. If not enough actions are covered in [M], XCS triggers the covering operator, which creates new classifiers with uncovered actions. Under pure exploration, an action is selected randomly, and all classifiers predicting that action form the action set [A]. The action is sent to the environment and the received reward is used to update the parameters of the classifiers in [A]. Eventually, the genetic algorithm is triggered in the action set [A], and subsumption [6] may be applied to avoid the presence of accurate but unnecessarily specialized classifiers in favor of accurate and general classifiers. Under exploit mode, given an input instance, a vote for each action is determined by a fitness-weighted average of all the matching classifiers that advocate the action, and the most voted action is chosen as the output.

5. Experiments

To build experiments based on our own framework, we should first define the target concept that is under study and codify it as a disjunction of rules. We chose two known problems: the *multiplexer* problem [5], and the *position* problem [7]. The multiplexer problem is one of the most used benchmark problems in the field of LCSs. We used the multiplexer problem of length 6. Its output corresponds to the value of the bit which is addressed by the address bits. Table 1 shows the rules coding the target concept. The position problem has as an output the position of the left-most bit. Thus, it is a multi-class problem. It presents imbalances, in the sense that the target concept is defined by rules ranging from maximal generalization to maximal specificity. See table 1 for an example of the position problem at length 5.

We applied sampling methods SRS, SIS, RRS and RIS to each of the problems. All the sampling methods

that use random processes (i.e., SRS, RRS and RIS) are repeated *nexp* times to test robustness of the problem-method generation process in front of random selection. In our experiments, we took *nexp = 3*.

Multiplexer6	*Position5*
000###:0	00000:0
001###:1	00001:1
01#0##:0	0001#:2
01#1##:1	001##:3
10##0#:0	01###:4
10##1#:1	1####:5
11###0:0	
11###1:1	

Table 1 Multiplexer6 and Position5

For each experiment, we have to set the number of instances contained in the training dataset. Prior to this, we computed the minimum number of instances that each dataset should contain to have a full representative dataset, which we name as LLIL (Low Learning Instances Limit). That is, how many instances should be generated if we used method SIS, which systematically obtains all the possible instances for each rule. To avoid the effect of sparsity, we guaranteed that our datasets in this set of experiments contained at least this number of instances. Taking into account this issue, we generated datasets containing 100, 1000, 10000, 100000 and 1000000 samples using each method. When XCS learns from these training datasets, it samples the examples sequentially. If the number of instances of the dataset is larger than the number of XCS's cycles, then some instances are not used for learning. Otherwise, if the number of XCS's iterations is higher than the number of instances, the dataset is repeatedly sampled until XCS's learning is finished.

XCS is run with two different random seeds and different population sizes. We show the averaged results for population sizes of 500 and 2000.

Note that the structure of selected problem is also very important. The multiplexer problem and the position problem represent two different concept structures. This may have a different influence on XCS's behavior. Also the different sampling strategies may have a different impact on each problem.

6. Results and Analysis

We ran XCS classifier system with the multiplexer and position problem, which were sampled according to strategies SRS, SIS, RRS and RIS, as explained in the last section. The results obtained are displayed in figures 3 to 10. Each figure plots the percentage of the optimal population achieved by XCS along the training iterations. Each figure is associated to a given problem and a data set size. Each curve corresponds to a different sampling strategy. The experiments run with same strategy and same number of instances is averaged.

Figures 3 to 6 show the results of XCS with the multiplexer problem with different sample sizes. See that no differences are found with respect to the different sample sizes. This is due to the fact that in all cases the number of sampled instances is higher than the minimum number of different samples needed to represent the full concept if SIS was used. Even in the strategies that do not guarantee that all the available instances are sampled behave similarly to SIS. There are no differences among the different sampling strategies because there is not any significant alteration between the original distribution of samples in the original concept domain and the sampled domain.

The results with the position problem are significantly different from those with the multiplexer. Firstly, figure 7 shows the results with a sample containing 100 instances. Since the minimum number of instances to represent the full concept is 32, SIS dataset would have at least one representative of each example. However, XCS is not able to discover the target concept with SIS sampling. This is due to the difficulty of the target concept itself.

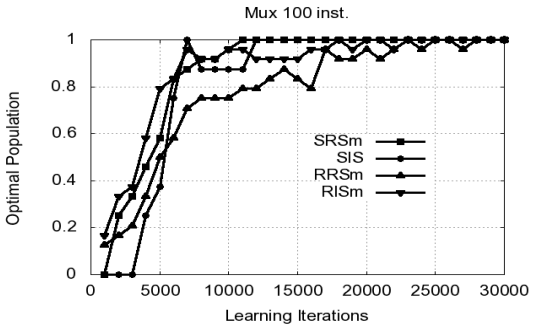

Fig. 3 Evolution of %[O] in the Multiplexer problem with 100 instances

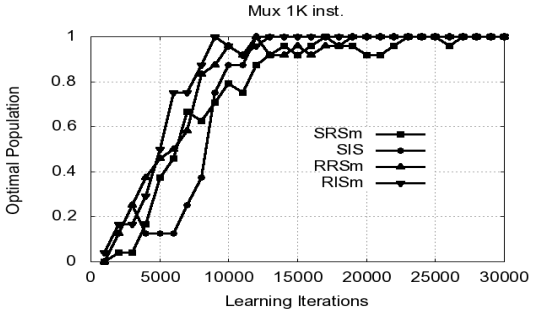

Fig. 4 Evolution of %[O] in the Multiplexer problem with 1,000 instances

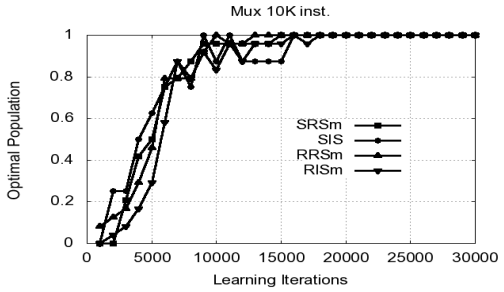

Fig. 5 Evolution of %[O] in the Multiplexer problem with 10,000 instances

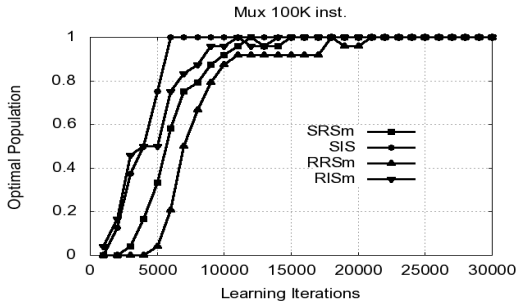

Fig. 6 Evolution of %[O] in the Multiplexer problem with 1,000,000 instances

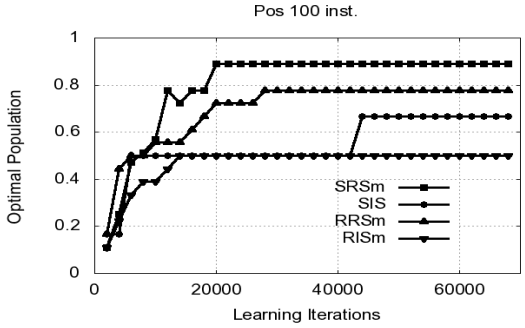

Fig. 7 Evolution of %[O] in the Position problem with 100 instances

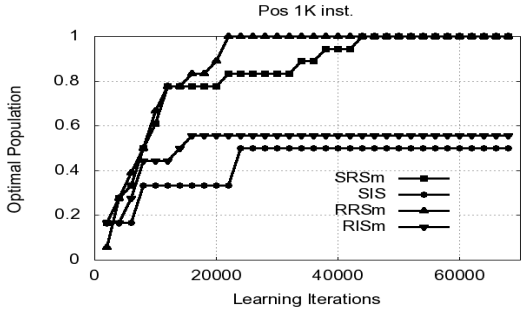

Fig. 8 Evolution of %[O] in the Position problem with 1,000 instances

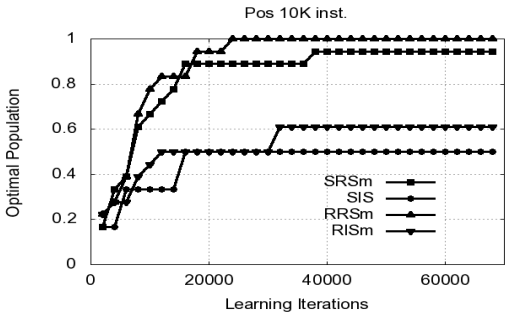

Fig. 9 Evolution of %[O] in the Position problem with 10,000 instances

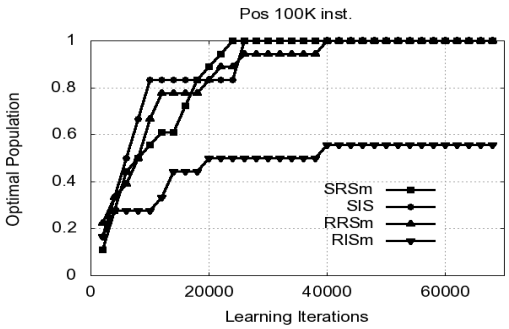

Fig. 10 Evolution of %[O] in the Position problem with 100,000 instances

With only 100 random samples no all problem instances are included as show systems behavior of figure 3. With higher samples (see figures 8 to 10), XCS is able to learn the target concept with strategies SRS and RRS. Precisely, these strategies alter the distribution of the samples with respect to the original distribution. Surprisingly, strategy SIS does not allow XCS to discover the target concept in most of cases. RIS behaves similarly to SIS. This behavior can be justified as follows. With SIS, there is a single example for the most specific rules (i.e., rules 00000:0 and 00001:1 only have a single example each one), while there are several examples for the most general rules (i.e., rule 1####:5). Since the learning mechanisms of XCS are frequency-based, the system would assign better resources to the discovery of the most represented rules. Thus, XCS will discover the general rules very fast and have many difficulties with the most specific rules. For more details, see [7]. Strategies RRS and SRS change this distribution. Each rule will have the same proportion of representative examples, independently of the generalization of the rules. This is why XCS discovers the target concept much earlier with these strategies.

Thus, there are two different kinds of complexities associated with XCS's learning. The first one is the complexity of the target concept itself. The second one is due to the sampling of this target concept. The

sampling process can add more difficulties or on the contrary, can counterbalance the complexity of the target concept.

7. Summary, Conclusions, and Further Work

In this paper, we have taken the first steps toward the definition of a methodology for evaluation and analysis of supervised learning methods. The methodology, which has been implemented for binary rule-based systems, aims at providing a systematic framework for the generation of data sets and analysis of learning algorithms. Although the particular analysis of XCS was not in the scope of this paper, a case study using the methodology has made evident several aspects of the online learner that are worth studying.

Several future work lines will be followed in light of the present paper. Firstly, the methodology will be extended to deal with rule-based rules that contain continuous variables. The inclusion of continuous variables will imply the redefinition of some of the sampling techniques, especially those that exhaustively sample all the search space. Moreover, we also aim at including several metrics that have been recently used to evaluate the complexity of real-life classification domains [3]. Thence, the complexity of the generated data sets will be evaluated according to these indicators. Then, this information would be considered to analyze which types of sampling derive in certain complexities and how this affects the learning process.

A second future work line will point toward the design of sampling methods that include either noise or class imbalances in the learning domains, with the aim of evaluating how the learners perform when learning from problems with these characteristics.

Acknowledgements

The authors would like to thank *Ministerio de Educación y Ciencia* for its support under projects TIN2005-08386-C05-01 and TIN2005-08386-C05-04, and Generalitat de Catalunya for its support under grants 2005FI-00252 and 2005SGR-00302.

References

[1] M. V. Butz and D. E. Goldberg and T.K. Tharankunnel. Analysis and improvement of fitness exploration in XCS: Bounding models, tournament selection, and bilateral accuracy. *Evolutionary Computation*, 11(3):239-277, 2003.

[2] T. Kovacs and M. Kerber. What Makes a Problem Hard for XCS?. Advances in Learning Classifier Systems. LNAI, pages 80-99. Springer-Verlag , London, 2001.

[3] T. K. Ho and M. Basu. Complexity Measures of Supervised Classification Problems. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(3):289–300, 2002.

[4] T. Kovacs. XCS Classifier System Reliably Evolves Accurate, Complete, and Minimal Representations for Boolean Functions. In *Soft Computing in Engineering Design and Manufacturing*, pages 59–68. Springer-Verlag, London, 1997.

[5] S. W. Wilson. Classifier Fitness Based on Accuracy. Evolutionary Computation, 3(2):149–175, 1995.

[6] S. W. Wilson. Generalization in the XCS Classifier System. In GP'98, pages 665–674. Morgan Kaufmann, 1998.

[7] E. Bernardó and J. M. Garrell. Accuracy-Based Learning Classifier Systems: Models, Analysis and Applications to Classification Tasks. In *Special Issue on Learning Classifier Systems, Evolutionary Computation*, 11(3):209-238,2003