Modeling XCS in Class Imbalances: Population Size and Parameter Settings

Albert Orriols-Puig[1,2], David E. Goldberg[2], Kumara Sastry[2], Ester Bernadó-Mansilla[1]

[1]Group of Research in Intelligent Systems, Enginyeria i Arquitectura La Salle
Universitat Ramon Llull, Barcelona, Spain 08022
[2]Illinois Genetic Algorithms Laboratory (IlliGAL), Dept. of Industrial and Enterprise Systems Eng.
University of Illinois at Urbana-Champaign, Urbana, IL 61801

aorriols@salle.url.edu, deg@uiuc.edu, ksastry@uiuc.edu, esterb@salle.url.edu

ABSTRACT

This paper analyzes the scalability of the population size required in XCS to maintain niches that are infrequently activated. Facetwise models have been developed to predict the effect of the imbalance ratio—ratio between the number of instances of the majority class and the minority class that are sampled to XCS—on population initialization, and on the creation and deletion of classifiers of the minority class. While theoretical models show that, ideally, XCS scales linearly with the imbalance ratio, XCS with standard configuration scales exponentially. The causes that are potentially responsible for this deviation from the ideal scalability are also investigated. Specifically, the inheritance procedure of classifiers' parameters, mutation, and subsumption are analyzed, and improvements in XCS's mechanisms are proposed to effectively and efficiently handle imbalanced problems. Once the recommendations are incorporated to XCS, empirical results show that the population size in XCS indeed scales linearly with the imbalance ratio.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Concept learning*

General Terms

Experimentation

Keywords

Evolutionary Computation, Genetic Algorithms, Machine Learning, Learning Classifier Systems, Class Imbalance

1. INTRODUCTION

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

XCS [11, 12], one of best representatives of Michigan-style Learning Classifier Systems (LCSs) [5], is a robust method that has been applied to solve different types of *machine learning* problems. LCS literature has usually considered problems with similar proportions of instances per class. Recently, learning from imbalanced domains has been identified as one of the main challenges for machine learning techniques, and only few studies on the effect of class imbalances on LCSs have been conducted [6, 9, 10]. While population sizing and scalability in XCS has broadly been studied on balanced domains [1], similar studies are lacking for imbalanced domains.

The aim of this paper is to model the effect of learning from imbalanced data in XCS and analyze the scalability of the population size required as the amount of class imbalance increases. We develop facetwise models that indicate the impact of the imbalance ratio (ir) on the population initialization and on the creation and deletion of classifiers of the minority class. Moreover, we derive population size bounds that guarantee that XCS will create and maintain these classifiers of the minority class. We design two test problems of bounded difficulty to validate the model: the *one-bit* problem and the *parity* problem. Results obtained with the *one-bit* problem show that the population size scales exponentially with the imbalance ratio, violating the model bounds. We investigate the causes of this deviation from the ideal scalability and propose some approaches to alleviate the early deletion of classifiers of the minority class. Once these recommendations are incorporated to XCS, the empirical results on both the *one-bit* and the *parity* problems agree with the theoretical bounds, showing that the population size in XCS scales linearly with the imbalance ratio.

The remainder of the paper is organized as follows. XCS is briefly described in section 2. Next, we develop theoretical models for learning from imbalanced datasets, deriving a population size bound to ensure the growth of minority class niches. Section 4 introduces the test problems used in the experimentation. In section 5, we run XCS with one of the test problems, and empirical results show a deviation from the ideal scalability of the population size. Section 6 analyzes the causes of the deviation, resulting in a set of recommendations on how to set the system when learning from imbalanced data. These recommendations are grouped in XCS+PMC. The theoretical model is empirically validated

using XCS+PMC in sections 7 and 8. Finally, we provide further directions, summarize, and conclude.

2. XCS IN A NUTSHELL

This section provides a brief description of the XCS classifier system. For a detailed description, the reader is referred to [11, 12, 3].

XCS is an accuracy-based learning classifier system introduced in [11] 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] [7]) 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 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.

3. COMPLEXITY WHEN LEARNING FROM CLASS IMBALANCES

In this section we investigate how class imbalances influence XCS's learning mechanisms. We benefit from previous studies that analyze the computational complexity of XCS considering a uniform sampling of instances [1]. The aim of this section is to extend the theory to class-imbalanced problems.

XCS evolves a set of rules distributed in *niches* around the problem space. A niche is a region in the solution space consisting of classifiers that represent a specific *schema* [5] and predict the same class. In imbalanced domains, the supply of instances in each niche is not uniformly distributed. Since XCS is an online learner with an occurrence-based reproduction mechanism, niches more frequently activated (i.e, nourished niches) will be given more resources than infrequently activated niches (i.e., starved niches). Our aim is to study XCS's capabilities to provide solutions in these starved niches. For this purpose, we first analyze the effect of class imbalances on the following facets:

- *Population initialization.* We analyze if the covering operator can supply enough classifiers representing schemas of the minority class.
- *Generation of correct classifiers of the minority class.* We analyze the probability that the GA generates correct classifiers of the minority class when there are not representatives of the minority class in the population.
- *Time of extinction of correct classifiers of the minority class.* We derive the average life-time of correct classifiers of the minority class.

Second, we use the facetwise analysis to derive a bound on the population size to ensure that XCS will be able to maintain correct classifiers of the minority class and that these classifiers will receive, at least, one genetic event before being removed. In the analysis, we consider problems that consist of n classes in which one of the classes, addressed as the minority class, is sampled in a lower frequency than the others. Specifically, the minority class is sampled with probability $1/(1+ir)$, where ir is the ratio between the number of instances of any class other than the minority class and the number of instances of the minority class.

3.1 Population Initialization

In this section we analyze the effect of class imbalances on the covering operator, which is responsible for initializing the population. At each learning iteration, the environment samples a new input instance, and XCS creates the match set, which consists of all classifiers in the population that match the current input instance. If some action is not represented in the match set, the covering operator is activated to create a new classifier advocating that action with a condition generalized from the current input. The amount of generalization over the inputs is controlled with the parameter $P_\#$, and its value has to be high enough—close to 1—to satisfy the covering challenge [2].

In our analysis, we assume that $P_\#$ is appropriately set to a high value to guarantee the covering challenge; consequently, covering will be only activated in the first iterations of the learning.

As ir increases, less instances of the minority class will be sampled during the first iterations. Thus, for high class imbalances, covering will be activated on examples of any class other than the minority class, and so, all classifiers' conditions, regardless the action they advocate, will be mainly a generalization of instances of any class other than the minority class. However, the covering operator should be applied on enough instances of the minority class to initially supply the population with sufficient classifiers of the minority class. In the following, we derive a lower bound of the probability of activating covering when the first instance of the minority class is sampled. According to [1], the probability that one instance is covered by, at least, one classifier is the following:

$$P(\text{cover}) = 1 - \left[1 - \left(\frac{2-\sigma[P]}{2}\right)^{\ell}\right]^{N} \quad (1)$$

where ℓ is the input length, N the population size, and $\sigma[P]$ the specificity of the population. During the first learning stage of XCS, we can approximate that: $\sigma[P] \approx 1 - P_\#$.

For a given ir, let us assume the worst case where a) XCS receives ir instances of the other classes before receiving the first instance of the minority class, and b) the covering operator is triggered for each instance supplying n classifiers per instance (where n is the number of classes). The probability that XCS covers the first instance of the minority class is:

$$P(\text{cover}) = 1 - \left[1 - \frac{1}{n}\left(\frac{2-\sigma[P]}{2}\right)^{\ell}\right]^{n \cdot ir} \quad (2)$$

The equation supposes that $N > n \cdot ir$, i.e., that XCS will

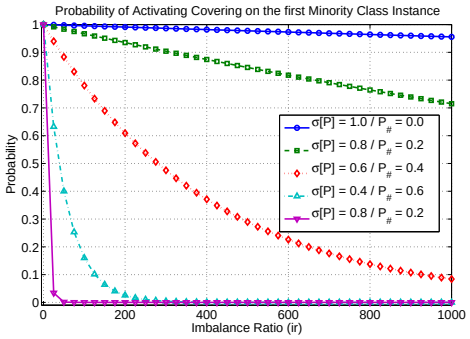

Figure 1: Probability of activating covering on a minority class instance given a certain specificity $\sigma[P]$ and the imbalance ratio ir. The curves have been drawn from formula 3 with different specificities $\sigma[P]$ and setting ℓ=20.

not delete any classifier during the first ir iterations. Given a fixed ℓ and $\sigma[P]$, the term in brackets in the right hand of the equation decreases exponentially as the imbalance ratio increases; thus, the probability of covering minority class instances tends to one exponentially with the imbalance ratio.

Provided that the probability of activating covering is $1 - P(cover)$, and recognizing that $(1 - r/n)^n \approx e^{-r}$, we obtain that the probability of activating covering having sampled a minority class instance is:

$$P(\text{activate cov. on. min.}) = 1 - P(\text{cover}) \approx e^{-ir \cdot e^{-\frac{\ell\sigma[P]}{2}}} \quad (3)$$

which decreases exponentially with the imbalance ratio and, in a higher degree, with the condition length and the initial specificity. Figure 1 depicts the equation for $\ell = 20$, $n = 2$ and different initial specificities, showing that the probability of activating covering on the first sampled instance of the minority class decreases exponentially with the imbalance ratio.

The analysis made above shows up that the covering operator fails to supply classifiers representing correct schemas of the minority class for moderate and high imbalance ratios. In classification tasks, dynamic resampling techniques could be applied to overcome this problem [10]. However, we are interested in the intrinsic capabilities of XCS to discover the minority class, and so, resampling techniques are not considered in our analysis. Consequently, XCS will start the search with a high general population that contains few schemas of the minority class. So, the genetic search will be the main responsible for obtaining the first correct classifiers of the minority class. In the next section we analyze the probabilities that the GA generates new classifiers that represent starved niches.

3.2 Generation of Correct Classifiers of the Minority Class

In this section, we analyze the probability of generating correct classifiers of the minority class assuming that covering has not provided any classifier representing any schema of the minority class. Since mutation is the primary operator for exploring new regions of the input space, we propose a simplified model where only the effect of mutation is considered. We also assume low values of the probability of mutation μ ($\mu < 0.5$) as it is usual in practice.

Accurate classifiers of the minority class can be obtained while exploring any class in the feature space. In the following, we derive the probabilities of generating a correct classifier of the minority class when sampling instances of any class, and gather them together deriving the probability of generating new correct classifiers of the minority class.

3.2.1 Sampling Minority Class Instances

A minority class instance is sampled with probability $P(inst_{min}) = 1/(1 + ir)$. As XCS chooses the class to be explored randomly, two possible scenarios are possible: 1) every 1/n times, XCS activates high rewarded niches of the minority class, and 2) the other $(n - 1)/n$ times, XCS explores low rewarded niches of another class. Next, we derive the probabilities of obtaining a correct classifier of the minority class in both cases.

First, if the GA is triggered on a niche of the minority class (assuming that there are not correct classifiers in the niche), it will generate a correct classifier of the minority class if all the bits of the schema are set to their correct value, and the class of the classifier is not flipped. We consider the worst case, that is, that all the bits of the schema need to be changed. That gives the following lower bound on the probability of generating a new correct classifier of the minority class:

$$P(cl_{min} \mid inst_{min} \wedge niche_{min}) = \left(\frac{\mu}{2}\right)^{k_m} \cdot (1 - \mu) \quad (4)$$

where μ is the probability of mutation and k_m is the order of the schema. That is, the formula defines a parabola on the values of μ; for k_m=1, the probability is maximized at $\mu = 0.5$. However, this high value of μ may introduce too much disruption in the genetic search. Increasing the order of the schema k_m, the probability of obtaining a correct classifier of the minority class decreases.

Second, if the GA is activated on a niche of any class other than the minority class, not only all bits of the schema have to be correctly set, but also the class has to be mutated to the minority class. Thus, the lower bound on the probability of generating a minority class classifier is:

$$P(cl_{min} \mid inst_{min} \wedge \neg niche_{min}) = \left(\frac{\mu}{2}\right)^{k_m} \cdot \frac{\mu}{n-1} \quad (5)$$

As above, the probability of generating a new correct classifier of the minority class depends on the mutation probability μ and the order of the schema k_m. Moreover, it also depends inversely on the number of classes of the problem.

3.2.2 Sampling Instances of other Classes

XCS can also create correct classifiers of the minority class when sampling instances that do not belong to the minority class. The probability of sampling these instances is $P(\neg inst_{min}) = ir/(1 + ir)$. Two possible scenarios are possible when an instance of any class other than the minority class is sampled: 1) a niche of the minority class is triggered, and 2) a niche of another class is triggered. The first case implies that all the bits of the schema must be specified. The second case also requires that the class is flipped to the minority class. Thus, the derived probabilities are:

$$P(cl_{min} \mid \neg\, inst_{min} \;\wedge\; niche_{min}) = \left(\frac{\mu}{2}\right)^{k_m} \cdot (1 - \mu) \quad (6)$$

$$P(cl_{min} \mid \neg\, inst_{min} \;\wedge\; \neg niche_{min}) = \left(\frac{\mu}{2}\right)^{k_m} \cdot \frac{\mu}{n-1} \quad (7)$$

These probabilities are equivalent to the ones obtained in formulas 4 and 5 respectively. They are guided by the probability of mutation μ and the order of the schema k_m.

3.2.3 Time to Create Correct Classifiers of the Minority Class

Given the derived sampling probabilities and lower bounds of formulas 4-7, we analyze the minimum time required to generate the first correct representative of the minority class. We assume that the genetic event is always applied ($\theta_{GA} = 0$) and only one classifier is created by the effect of mutation. Under these circumstances, the probability of generating a correct classifier of the minority class is the sum of the following probabilities:

- The probability p_1 of generating a minority class classifier when sampling a minority class instance. That is, $p_1 = \frac{1}{1+ir} \cdot \frac{1}{n} \left(\frac{\mu}{2}\right)^{k_m}$
- The probability p_2 of generating a minority class classifier when sampling an instance of any class other than the minority class. That is, $p_2 = \frac{ir}{1+ir} \cdot \frac{1}{n} \left(\frac{\mu}{2}\right)^{k_m}$

The time required to discover the first representative is derived from the addition of both probabilities:

$$t(cl_{min}) = \frac{1}{P(cl_{min})} = \frac{1}{p_1 + p_2} = n \left(\frac{2}{\mu}\right)^{k_m} \quad (8)$$

which depends linearly on the number of classes and exponentially on the order of the schema, but does not depend on the imbalance ratio.

Thus, even though covering fails to provide classifiers representing schemas of the minority class, XCS will be able to generate the first correct classifiers of the minority class independently of the imbalance ratio. In the following, we derive the time until the deletion of these classifiers. With both the generation and deletion time, we calculate the minimum population size to maintain these classifiers and ensure the growth of the best representatives of the minority class.

3.3 Deletion Time of Minority Class Classifiers

XCS deletes classifiers depending on their action set size as and their fitness. Having a good estimation of as, deletion would remove classifiers that belong to numerous niches and have low fitness; consequently, it would maintain accurate classifiers in starved niches. Nevertheless, overgeneral classifiers, whose presence is numerous in imbalance domains [9], tend to bias the action set size estimate of accurate classifiers that belong to the same action sets. Under these circumstances, deletion may be better approached as a random deletion. As we delete two classifiers every GA application, we obtain that:

$$P(\text{delete } cl_{min}) = \frac{2}{N} \quad (9)$$

From this formula, we derive the time until deletion:

$$t(\text{delete } cl_{min}) = \frac{N}{2} \quad (10)$$

In the following, we use formulas 8 and 10 to derive the minimum population size that guarantees the discovery, maintenance and growth of starved niches.

3.4 Bounding the Population Size

Herein, we use the formulas of generation and deletion of minority class classifiers to derive two population size bounds. First, we derive the minimum population size to ensure that XCS will be able to create and maintain correct classifiers of the minority class. Then, we derive the population size bound to ensure the growth of the niches that contain these correct classifiers of the minority class.

3.4.1 Minimum Population Size to Guarantee Representatives

Our first concern is to ensure that there would be correct classifiers representing the niches of the minority class. Our sake is to guarantee that, before deleting any classifier of the minority class, another correct classifier of the minority class will be created. Thus, we require that the time until deletion be greater than the time until a correct classifier of the minority class is generated.

$$t(\text{delete } cl_{min}) > t(cl_{min}) \quad (11)$$

That is:

$$N > 2n \left(\frac{\mu}{2}\right)^{k_m} \quad (12)$$

which indicates that, to guarantee that all the minority class niches of the system have at least one correct classifier, the population size have to increase linearly with the number of classes and exponentially with the order of the schema; however, it does not depend on the imbalance ratio.

3.4.2 Population Size Bound to Guarantee Reproductive Opportunities

Above, we discussed the minimum time required to generate the first correct classifiers of the minority class. Now, we are concerned about the requirements to ensure that classifiers of minority class will evolve to better ones.

To ensure the growth of niches of the minority class, we should guarantee that the best classifiers in the niche receive, at least, one genetic opportunity. Otherwise, XCS could be continuously creating and removing classifiers from a niche, but not searching toward better classifiers. As before, we consider $\theta_{GA} = 0$; moreover, we assume that the selection procedure chooses one of the strongest classifiers in the niche. Then, the time required for a minority class classifier to receive a genetic event is inverserly proportional to the probability of activation of the niche it belongs to:

$$t(\text{GA } niche_{min}) = n \cdot (1 + ir) \quad (13)$$

which depends on the imbalance ratio and the number of classes.

To guarantee that these strong classifiers of the minority class will receive a genetic opportunity before being deleted, we require that:

$$t(\text{delete } niche_{min}) > t(\text{GA } niche_{min}) \quad (14)$$

from which we derive the population size bound:

$$N > 2n \cdot (1 + ir) \quad (15)$$

That is, the population size has to increase linearly with the number of classes and the imbalance ratio to warrant that correct classifiers of the minority class will receive, at least, one genetic event before being deleted.

In this section we derived a theoretical model for learning under class imbalances. The analysis revealed a failure of covering to provide schemas of the minority class for high imbalance ratios, giving the responsibility for generating correct classifiers of the minority class to the GA. Under this scenario, we developed a population size bound that indicated that XCS should be able to create and maintain correct classifiers of the minority class regardless of the imbalance ratio; furthermore, we derived a second bound denoting that the population size should increase linearly with the imbalance ratio to ensure the growth of starved niches. In the next section we describe the test problems used to validate the theoretical models developed.

4. FACETWISE DESIGN OF TEST PROBLEMS

Goldberg illustrates the big importance of designing types of problems characterized by various *dimensions of problem difficulty* to permit a successful understanding of complex systems [4]. We follow this approach closely to design two test problems of *bounded difficulty*, in which we can easily control the complexity introduced by the imbalance ratio. First, we design the *one-bit* problem which completely isolates the complexity introduced by the imbalance ratio from other complexity factors, such as linkages between variables or misleading fitness pressure. Second, we design a more complex problem that requires a stronger pressure toward optimal classifiers and also permits to control the imbalance ratio: the *imbalanced parity* problem.

4.1 The Imbalanced One-Bit Problem

The *one-bit* problem is defined as follows. Given a binary input of length ℓ, the output is the value of the left-most bit. The problem contains four niches and only one *building block* or *schema* per niche; moreover, the order of the schema is one. Its complete action map consists of two maximally general and accurate classifiers predicting class '0' and class '1' respectively (i.e., the classifiers $\texttt{0\#}^{\ell-1}\texttt{:0}$ and $\texttt{1\#}^{\ell-1}\texttt{:1}$), and two maximally general but incorrect classifiers predicting class '0' and class '1' (i.e., $\texttt{1\#}^{\ell-1}\texttt{:0}$ and $\texttt{0\#}^{\ell-1}\texttt{:1}$).

The imbalance complexity is controlled by sampling an instance of the minority class with probability P_{smin}. At each learning iteration, the environment chooses an instance randomly. If it is an instance of class '0', it is automatically given to XCS. Otherwise, the instance is passed to XCS with probability P_{smin}. If it is not accepted, a new instance is randomly sampled, which undergoes the same decision process. In the experimentation, we control the imbalance complexity by increasing the proportion of majority class instances w.r.t. the minority class instances (imbalance ratio ir). So, the probability of sampling a majority class instance is $P_{maj} = ir/(1+ir)$, and the probability of sampling a minority class instance is $P_{min} = 1/(1+ir)$.

4.2 The Imbalanced Parity Problem

The *parity* problem is a two-class binary problem originally introduced in [8]. The problem is defined by the length of the input ℓ and the number of relevant bits k, where $\ell \geq k$. Given a binary string of length ℓ, the output is the number of one-valued bits in the first k bits modulo two. The difficulty of the problem is that all the k first bits form a single building block, and so, they have to be processed together. Besides, there are $p = \ell - k$ bits that are not relevant for determining the class, and so, they should be generalized.

The imbalance complexity is introduced by starving the class labeled as '1'. Besides ℓ and k, the problem is also defined by the imbalance ratio ir, where $ir \geq 1$. For $ir = 1$, the problem is equal to the parity problem. For $ir > 1$, $\frac{ir-1}{ir}2^{\ell-1}$ instances of the class labeled as '1' are taken out uniformly from the minority class niches. Regardless of the imbalance ratio, XCS is expected to evolve the same complete action map as in the parity problem.

5. XCS ON THE ONE-BIT PROBLEM

To validate the model derived in section 3, we first ran XCS with the *one-bit* problem with condition length $\ell = 20$ and imbalance ratios from ir=1 to ir=1024. The system was configured as follows: α=0.1, $\epsilon_0 = 1$, ν=5, θ_{GA}=25, χ=0.8, μ=0.04, θ_{del}=20, δ=0.1, θ_{sub}=200, $P_{\#}$=0.8. We used tournament selection, two point crossover and niched mutation [3] for the genetic algorithm. Subsumption was applied on the GA but not in the action set, and β was set as suggested in [9] to get good parameter estimates. In each experimentation, we ran XCS during $10,000 \cdot ir$ iterations to ensure that the system received the same number of instances of the minority class for all ir.

The experimentation was made as follows. For each imbalance level, we ran XCS with different population sizes and took the minimum population size required to solve the problem. After training, we tested XCS with all the training instances, and measured the percentage of correct classifications of instances of the majority class (TN rate) and the percentage of correct classifications of instances of the minority class (TP rate). All the results were averaged over 10 different random seeds. We considered that XCS succeeded if the product of TP rate and TN rate was greater than a certain threshold θ (we set $\theta = 95\%$).

Figure 2 shows the minimum population size required by XCS at different imbalance ratios. The points indicate the empirical values, and the dashed line draws an exponential curve that fits the points for high class imbalances. Two different regimes can be observed in the figure. In the lowest imbalance ratios, i.e., from ir=1 to ir=8, the population size required to solve the problem is constant. On the other hand, for the highest imbalance levels, i.e., from ir=256 to ir=1024, the population size increases exponentially (fitted by the curve with dashed line). Between ir=8 and ir=256 there is a transition region.

For $ir > 1024$, XCS failed regardless of the population size, even though ϵ_0 was decreased to ensure that overgeneral classifiers were not considered as accurate. Specifically, for ir=2048 we tried population sizes up to N=6,000, which is the population size predicted by the exponential curve. XCS was not able to solve the *one-bit* problem in any of the runs.

While facetwise models derived in the previous section revealed that the population size should increase linearly to ensure the growth of starved niches, empirical results show that the population size scales exponentially, and that XCS only can solve the *one-bit* problem up to ir=1024. In the next section, we study the causes that are potentially responsible for the deviation of the empirical results from the theoretical models.

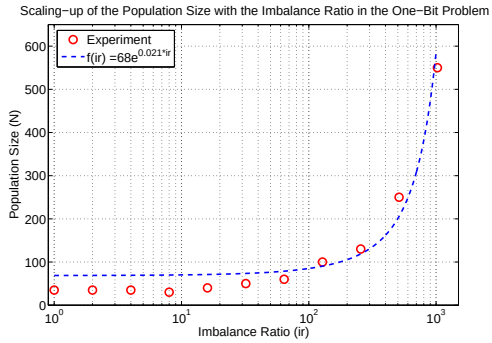

Figure 2: Scalability of the population size (N) with the imbalance ratio (ir) in the imbalanced *one-bit* problem with $\ell = 20$. The points indicate the empirical values of the minimum population size required by XCS. The dashed line shows the curve defined by the function $f(ir) = 68 \cdot e^{0.021 \cdot ir}$, which grows exponentially and fits the scalability of the population size for the highest imbalance ratios.

6. ANALYSIS OF THE DEVIATIONS BETWEEN THEORY AND EXPERIMENTS

Last section evidenced a deviation of the empirical results from the models derived in section 3. This section studies the causes that are potentially responsible for that deviation. We analyze the effect of the mutation scheme in the discovery of new minority class classifiers, the initialization of the parameters of minority class classifiers, and the effect of subsumption on the whole population; we also introduce the need of stabilizing the population before testing. The analysis results in a series of recommendations that aim at protecting minority class classifiers from an early deletion.

6.1 Niched Mutation versus Free Mutation

In the theoretical model derived in section 3, we assumed an unrestricted or free mutation, which permitted to create new classifiers of the minority class from any niche of the solution space. However, the experiments made in section 5 used a niched mutation, as defined in [3], which tries to preserve the niche by forcing new classifiers to match the input instance from which the action set was created. Under niched mutation, the first correct classifiers of the minority class can only be created if a minority class instance is sampled. Thus, the time until generation of the first representant of the minority class is:

$$t(cl_{min} \mid \text{niched mutation}) \quad = \quad n \cdot (1 + ir) \cdot \left(\frac{2}{\mu}\right)^{k_m} \quad (16)$$

which depends linearly on the imbalance ratio and the number of classes, and exponentially on the order of the schema. Thus, with niched mutation, the minimum population size to guarantee that the niches of the minority class will have correct representatives is the following (see equation 11):

$$N > 2n \cdot (1 + ir) \left(\frac{2}{\mu}\right)^{k_m} \quad (17)$$

which indicates that N should increase linearly with the imbalance ratio to maintain correct classifiers in the minority class niches.

This equation indicates that free mutation incentives the discovery of minority class classifiers by permitting to expore beyond the niche. This is a crucial factor when covering cannot supply correct schemas of the different classes. Nonetheless, this is not the only factor that may hinder XCS from learning minority class niches in highly imbalance datasets. Even with niched mutation, the model indicates that the population size should increase linearly with the imbalance ratio; however, the experimentation showed that the scalability of the population size was exponential at the highest imbalance levels.

6.2 Inheritance Error of Classifiers' Parameters

The parameters of new classifiers created by the genetic algorithm are initialized as a copy or a discounted value of their parents [3]. If the classifier participates frequently in action sets, the values of its parameters would be quickly adjusted to their theoretical values. Nonetheless, when learning from imbalance data, classifiers that belong to starved niches are infrequently updated, and so, have inaccurate parameters values during a longer time.

The parameter that controls the initial deletion probability of a classifier is the action set size as. Having an initial overestimated value of this parameter may cause an early deletion of accurate but infrequently activated classifiers. To avoid that, we suggest to initialize the action set size of a new classifier to 1, minimizing the deletion probability of a minority class classifier before being updated for the first time. This approach may be applied cautelously in problems where classifiers that do not match any input could be created. In this case, deletion policies of non-activated classifiers may be applied.

6.3 The Effect of Subsumption

Subsumption deletion is a mechanism introduced in [12] with the aim of increasing the pressure toward generalization to obtain highly accurate and compact populations. The main idea behind subsumption is to delete accurate but specific classifiers if other accurate but more general classifiers exist in the population (a classifier is considered to be a subsumer if its experience is higher than θ_{sub} and its error is lower than ϵ_0).

Although subsumption is a powerful tool to obtain a highly accurate and maximally general population, it may hinder XCS's performance in highly imbalanced datasets if it is not adjusted properly. That is, in imbalanced datasets, an overgeneral classifier of the majority class may receive ir positive rewards before receiving the first negative reward. Consequently, XCS may consider these classifiers as accurate during ir iterations. If $ir > \theta_{sub}$ some correct but more specific classifiers of the majority class may be subsumed by overgeneral classifiers. Thus, our approach is to set $\theta_{sub} \geq ir$ to avoid this situation.

6.4 Stabilizing the Population before Testing

Finally, we are concerned about the composition of the final population. As the GA is continuously applied, new classifiers are generated until the end of the learning. Thus, there may be some classifiers in the final population that have been poorly evaluated, and so, their parameters may not be reliable. In highly imbalanced datasets, as instances of the minority class are rarely sampled and classifiers of the majority class receive genetic events frequently, the presence

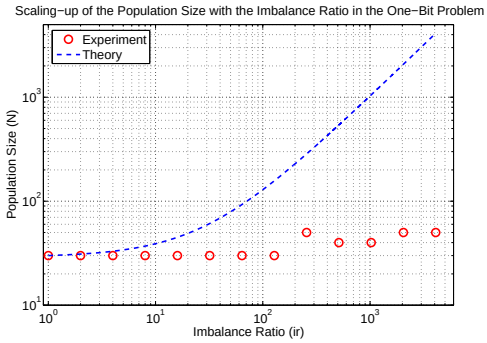

Figure 3: Scalability of the population size (N) with the imbalance ratio (ir) in the imbalanced *one-bit* problem with $\ell = 20$. The points show the empirical values of the minimum population size requirements. To compare the empirical results with the theory, we plot a curve (in dashed line) that departs from the minimum population size required at ir=1 and increases linearly with slope equal to 1 with the imbalance ratio.

of overgeneral classifiers with inaccurate estimations of their parameters in the final population will be stronger.

To avoid classifiers with inaccurate estimations in the final population, we introduced some extra runs at the end of the learning process with the GA switched off. In that way, we could stabilize classifiers parameters and get a consistent population.

Summarizing, we attribute the deviation from the model to a combined effect of the mutation scheme, the parameter inheritance procedure, subsumption and parameter estimates of inexperienced classifiers. Based on the analysis, we propose a set of recommendations which are gathered under the name XCS+PMC (XCS with mechanisms Protecting the Minority Class). In the following section, we repeat the experiments with the *one-bit* problem, and compare the empirical results with the theoretical model.

7. RESULTS WITH XCS PROTECTING THE MINORITY CLASS: XCS+PMC

We repeated the experimentation with the *one-bit* problem presented in section 5, but following the recommendations given in section 6. So, niched mutation was replaced by free mutation; as of new classifiers was initialized to 1, as proposed in section 6.2; subsumption was configured in relation to the imbalance ratio, that is, $\theta_{sub} = ir$; and, after learning, we ran N_{conds} runs without applying the GA, where $N_{conds} = 1000 \cdot ir$.

Figure 3 shows the minimum population size required to solve the *one-bit* problem with imbalance ratios from ir=1 to ir=4096. Note that in the experiments made in section 5 XCS could only solve the problem up to *ir=1024*. The points show the empirical values of the minimum population size required at different imbalance ratios, while the theoretical bound is shown by the dashed line.

The scalability of the population size shows two different facets. From $ir = 1$ to $ir = 128$, XCS could solve the *one-bit* problem with a constant population size (N=35). For higher imbalance levels, i.e., from ir=128 to ir=4096, the population size had to be slightly increased to solve the problem; for ir=4096, XCS succeeded with N=50. That is, in this second facet the population size needed to increase linearly but with a slope close to zero.

Results obtained with the *one-bit* problem indicate that the population size bound to ensure the growth of niches of the minority class is valid but a little over-estimated (see formula 15). We hypothesize that, as the *one-bit* problem is really simple, XCS can solve it only maintaining some correct classifiers in niches of the minority class without the need of a strong genetic pressure to evolve these classifiers to better ones. Under this assumption, the results obtained fit formula 12, which indicate that population size does not need to increase with the imbalance ratio to maintain correct classifiers of the minority class. Thus, the empirical results support that XCS can maintain representatives in niches of the minority class regardless of the imbalance ratio.

8. INCREASING THE DIFFICULTY: THE PARITY PROBLEM

In the last section we used a simple problem to demonstrate that XCS could maintain minority class classifiers at high class imbalances with very little increase of the population size, validating formula 12. Now, our aim is to empirically show how the population size increases in a more complex problem, the *parity* problem, where it is necessary a stronger guide toward optimal classifiers.

We ran XCS with the parity problem setting $\ell = 20$ and $k = 3$. The system was configured as described in section 7. Figure 4 shows the minimum population size required to solve the *parity* problem for imbalance ratios up to $ir = 4096$. The points show the empirical values at different imbalance ratios, while the theoretical bound is plot in dashed line.

As in the *one-bit* problem, the scalability of the population size in the *parity* problem shows two different facets. First, for low imbalance ratios—from ir=1 to ir=128—the problem can be solved without increasing the population size. This behavior was already observed in the *one-bit* problem. Our hypothesis is that, for these low imbalance ratios, the estimation of as is accurate, and so, the population size bound is over-sized. Thus, XCS appears to be really efficient for moderate imbalance ratios.

On the other hand, for the highest imbalance ratios, i.e., from ir=128 to ir=4096, the population size requirements increase linearly with the imbalance ratio. In these cases, the empirical results fit the population size bound derived in formula 15, validating that the theory predicts correctly the population size scalability in imbalanced datasets.

9. FURTHER WORK

Apart from modeling the scalability of the population size with class imbalances, the theoretical analysis provided in this paper also served to detect different aspects that may be improved to learn from imbalanced data more efficiently. One of the most important ones is the covering operator. As observed in section 6, the probability that covering generates correct classifiers of the minority class decreases exponentially with the imbalance ratio. In this way, mutation is the main responsible for creating correct classifiers of the minority class. We identified that the time to create these classifiers depended exponentially on the order of the schema k_m. In some problems, k_m can also depend on the imbalance ratio (e.g., the position problem [10]). In this case,

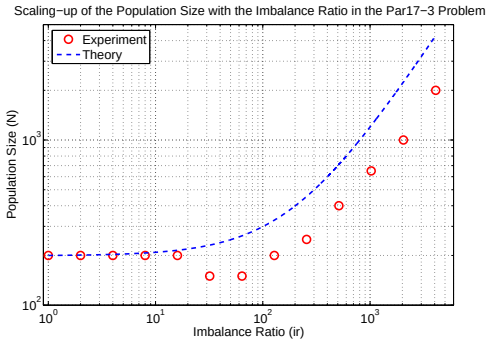

Figure 4: Scalability of the population size (N) with the imbalance ratio (ir) in the imbalanced *parity* problem with $\ell = 20$ and $k = 3$. The points show the minimum population size required at each imbalance level to solve the problem. To compare the empirical results with the theory, we plot a curve (in dashed line) that departs from the minimum population size required at ir=1 and increases linearly with slope equal to 1 with the imbalance ratio.

an initial supply of correct classifiers of the minority class should be crucial to speed up the learning and decrease the population size requirements in XCS. In further work, we will investigate other covering-based mechanisms to provide these initial classifiers of the minority class.

10. SUMMARY AND CONCLUSIONS

This paper investigated the capabilities of XCS to discover and maintain niches that are nourished infrequently. We derived theoretical models that analyzed the effects of class imbalances on the covering operator, the creation and deletion of correct classifiers of the minority class, and the number of genetic opportunities that these classifiers receive. From this, we obtained a population size bound indicating that XCS could maintain niches of the minority class regardless of the imbalance ratio, but the population size needed to increase linearly with *ir* to ensure the growth of better classifiers in the minority class niches.

We used a facetwise approach to design two test problems that permitted us to control the difficulty introduced by the imbalance ratio: the *one-bit* problem and the *parity* problem. We empirically showed that the population size in standard XCS scales exponentially with the imbalance ratio in the *one-bit* problem. The causes of the deviation from the model were attributed to the mutation scheme, the parameter inheritance procedure, subsumption, and the effect of unstable parameter estimates of unexperienced classifiers. The study resulted in a set of recommendations which were incorporated to XCS+PMC.

XCS+PMC could solve the *one-bit* problem up to *ir*=4092 with a population size nearly constant. Although solving a problem as the *one-bit* could seem a trivial task for a learner, the results obtained are really important since they demonstrate that, if genetic pressures lead to the creation of classifiers that represent a niche of the minority class, XCS will be able to maintain this niche. Furthermore, we introduced the *imbalanced parity* problem to validate the bound in a problem that requires a stronger pressure toward optimal classifiers. The results showed again two different facets. For low imbalance ratios, the population size remained constant. For high class imbalances, the population size increased linearly, showing that the theory predicts correctly the scalability of the population size with the imbalance ratio.

Acknowledgements

The authors thank Pier Luca Lanzi for the feedback he provided throughout this research. We also thank the support of *Enginyeria i Arquitectura La Salle*, Ramon Llull University, as well as the support of *Ministerio de Ciencia y Tecnología* under project TIN2005-08386-C05-04, and *Generalitat de Catalunya* under Grants 2005FI-00252 and 2005SGR-00302.

This work was also sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant FA9550-06-1-0096, the National Science Foundation under ITR grant DMR-03-25939 at Materials Computation Center, UIUC. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the National Science Foundation, or the U.S. Government.

11. REFERENCES

[1] M. V. Butz. *Rule-Based Evolutionary Online Learning Systems: A Principled Approach to LCS Analysis and Design*, volume 109 of *Studies in Fuzziness and Soft Computing*. Springer, 2006.

[2] M. V. Butz, T. Kovacs, P. L. Lanzi, and S. W. Wilson. How XCS Evolves Accurate Classifiers. In *GECCO'01*, pages 927–934. San Francisco, CA: Morgan Kaufmann, 2001.

[3] M. V. Butz and S. W. Wilson. An Algorithmic Description of XCS. In P. e. a. Lanzi, editor, *Advances in Learning Classifier Systems*, volume 1996 of *LNAI*, pages 253–272. Springer, 2001.

[4] D. E. Goldberg. *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers, 1 edition, 2002.

[5] J. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

[6] J. Holmes. Differential Negative Reinforcement Improves Classifier System Learning Rate in two-class Problems with Unequal Base Rates. In *GP'98*, pages 635–642. Morgan Kaufmann, 1998.

[7] 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.

[8] T. Kovacs and M. Kerber. What Makes a Problem Hard for XCS. In *Advances in Learning Classifier Systems*, pages 80–99. Springer-Verlag, 2001.

[9] A. Orriols-Puig and E. Bernadó-Mansilla. Bounding XCS Parameters for Unbalanced Datasets. In *GECCO '06*, pages 1561–1568, New York, NY, USA, 2006. ACM Press.

[10] A. Orriols-Puig and E. Bernadó-Mansilla. The Class Imbalance Problem in UCS Classifier System: A

Preliminary Study. In *Advances at the frontier of LCS*. Springer, InPress.

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

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