Mining Imbalanced Data with Learning Classifier Systems

Albert Orriols-Puig and Ester Bernadó-Mansilla

Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, Quatre Camins, 2. 08022, Barcelona, Spain, aorriols@salleURL.edu, esterb@salleURL.edu

Summary. This chapter investigates the capabilities of XCS for mining imbalanced datasets. Initial experiments show that, for moderate and high class imbalances, XCS tends to evolve a large proportion of overgeneral classifiers. Theoretical analyses are developed, deriving an imbalance bound up to which XCS should be able to differentiate between accurate and overgeneral classifiers. Some relevant parameters that have to be properly configured to satisfy the bound for high class imbalances are detected. Configuration guidelines are provided, and an algorithm that automatically tunes these XCS's parameters is presented. Finally, XCS is tested on a large set of real-world problems, appearing to be highly competitive to some of the most well-known machine learning techniques.

1 Introduction

Learning Classifier Systems (LCSs) [13] are complex machine learning methods that combine reinforcement learning and evolutionary computation techniques to evolve novel knowledge. Initially designed regarding the animal learning and cognitive psychology, some LCSs, and specially XCS [32, 33], have been adapted to solve ambitious machine learning problems, facing new challenges. Among them, learning from datasets that contain *rare objects* has been identified as one of the biggest challenges to many well-known learners in the machine learning realm. Many examples of real-world domains that contain rare objects can be found, such as fraudulent credit card transactions [11], learning word pronunciation [12], and detection of oil spills from satellite images [21].

Research on the detection of rare objects has been conducted from two different perspectives. The first perspective, associated mainly to supervised tasks, focuses on the drawbacks caused by training the learner with datasets that contain different proportion of instances per class. As many learners tend to minimize a global measure of error, they might be biased toward the most numerous classes in the training dataset [19,20]. The second perspective, associated to both non-supervised and supervised tasks, is concerned about the

A. Orriols-Puig and E. Bernadó-Mansilla: *Mining Imbalanced Data with Learning Classifier Systems*, Studies in Computational Intelligence (SCI) **125**, 123–145 (2008)
www.springerlink.com

distribution of examples around the feature space. It analyzes the difficulties of learning from datasets that contain rare cases around the feature space, which cause *small disjuncts* [17]. Both perspectives are really close, since imbalanced datasets usually present rare cases. In fact, rare cases or small disjuncts have been addressed as *within-class imbalances* [18]. In this paper we analyze the effect of class imbalances as a whole, but also the difficulties induced in learning classifiers systems due to *small disjuncts*.

Some studies have analyzed the behavior of LCSs on imbalanced datasets, and some approaches have been proposed to boost their learning capabilities on imbalanced data. Holmes addressed this topic in the context of epidemiological data, enhanced EpiCS [14] with a strategy based on disproportionate reinforcement per class [15], and lately built EpiXCS [16], an XCS-like system derived from EpiCS. Other studies [22,23] empirically analyzed the effects of imbalanced data on UCS [3], a learning classifier system derived from XCS specialized for classification tasks. Results evidenced that UCS was biased towards the majority class in highly imbalanced domains, and resampling techniques were presented as effective methods to alleviate the problem. Not until recently the first analyses of XCS's performance on imbalanced data have been done [24], showing that XCS is quite robust to class imbalances if it is properly configured.

The understanding of a complex system as XCS is crucial before applying it to solve real-world tasks. Aiming at this objective, this paper extends the study of XCS on imbalanced domains made in [24]. We provide a bound on the maximum amount of imbalance with which XCS can correctly deal, and we identify relevant XCS's parameters that need to be set properly to satisfy this bound in highly imbalanced datasets. Guidelines on how to tune these parameters are provided, and new experiments reveal that an appropriate setting of XCS's parameters drastically improves XCS's performance for high class imbalances. However, configuration guidelines depend on characteristics that may be unknown for real-world problems. Therefore, we present a method that automatically configures XCS's parameters, based on information collected during learning. The conclusions obtained from the analysis of XCS in artificial imbalanced domains are then applied to XCS so that it can efficiently mine imbalanced data from real-world datasets. Specifically, the algorithm of XCS's self-adaptation gives competitive results compared to learners such as C4.5, which is particularly known for yielding good performance in imbalanced datasets, as well as SMO and IBk.

The remainder of this chapter is organized as follows. Section 2 briefly introduces XCS for data mining. Next, we test XCS on an artificially imbalanced domain (Sect. 4). We model classifier's error, identify XCS's relevant parameters and provide guidelines on how to configure them in Sect. 4. Then, XCS is compared to other highly-competent learners on a large set of real-world problems. Finally, Sect. 7 summarizes, concludes and discusses further work.

2 Description of XCS

XCS is an online accuracy-based LCS that solves a problem by evolving a set of sub-solutions distributed in *niches* around the problem space. This section provides a brief description of XCS restricted to classification tasks. For further details, the reader is referred to [32, 33]; besides, an algorithmic description can be found in [10].

2.1 Knowledge Representation

XCS evolves a population of classifiers [P], where each classifier consists of a production rule of the form *condition* → *action* and a set of parameters. The most important parameters are: (a) the prediction p, which estimates the payoff that the classifier will receive when the rule is fired and its action is chosen as the output, (b) the prediction error ϵ, which estimates the error between the prediction and the received payoff, (c) the fitness F, which evaluates the accuracy of the classifier with respect to other classifiers in the same *action set*, and (d) the numerosity num, which counts the number of copies of the classifier in the population.

The condition representation was originally represented in the ternary alphabet: $\{0, 1, \#\}^{\ell}$, where ℓ is the length of the input instance. The symbol #, called *don't care*, allows to express generalizations in the classifier's condition. For real attributes, the input is codified as a set of intervals $[l_i, u_i]^{\ell}$, where l_i and u_i represent the lower and upper values that the attribute can take to apply the rule.

2.2 Performance Component

At each time step t, an input instance s_t is sampled, and XCS builds the *match set* [M], which contains all the classifiers in [P] that match the input instance. If the classifiers in [M] predict less that θ_{mna} different actions, the *covering* operator is activated, which creates new classifiers with a condition generalized from s_t and an action chosen from the ones not represented in [M] until θ_{mna} different actions are covered. For each action a_i in [M], XCS computes the payoff prediction $P(s_t, a_i)$ as a fitness weighted average of the prediction of all classifiers in [M] advocating a_i. Then, XCS selects an action to perform. Different selection regimes can be applied: from a *pure-explore regime*, in which the action is randomly selected, to a *pure-exploit regime*, in which the action with the highest prediction is chosen. Under classification problems, typically, the pure-explore regime is used during training, while the pure-exploit regime is used when the system predicts new unknown instances. Finally, XCS creates the action set [A], consisting of all classifiers in [M] that predict the chosen action.

2.3 Parameter's Update

The chosen action is sent to the environment, which returns a reward R that is used by XCS to update the parameters of the classifiers in [A]. First, the prediction p is adjusted: $p = p + \beta(R - p)$, where β is the learning rate ($0 < \beta \leq 1$). Next, the error ϵ is updated: $\epsilon = \epsilon + \beta(|R - p| - \epsilon)$. To update the classifier's fitness, XCS first computes the *accuracy* κ as follows:

$$\kappa = \begin{cases} 1 & \text{if } \varepsilon < \varepsilon_0 \\ \alpha(\varepsilon/\varepsilon_0)^{-\nu} & \text{otherwise} \end{cases} \tag{1}$$

where ϵ_0 is the maximum error that a classifier can take to be considered accurate, and α and ν are constants that control the rate of decline in accuracy ($0 < \alpha \leq 1$ and $\nu > 0$). κ is used to compute the *relative accuracy* κ' of the classifier in [A]: $\kappa' = \kappa / \sum_{[A]} \kappa_i$. Finally, the fitness is updated from the relative accuracy: $F = F + \beta(\kappa' - F)$. Note that the fitness is shared among the classifiers in the same action set since it is calculated from the relative accuracies.

2.4 Discovery Component

In XCS, the *genetic algorithm* (GA) is applied to the action set with a frequency fixed by the parameter θ_{GA}. It selects two parents from [A] (following either a proportionate selection scheme [32] or a tournament selection scheme [9]) and copies them creating two new classifiers, which are crossed and mutated with probabilities χ and μ respectively. The resulting offspring are introduced in the population, applying subsumption if required [33], and two classifiers are deleted to keep the population size constant.

3 XCS and Class Imbalances

In this section, we investigate how different amounts of class imbalance affect XCS. For this purpose, we designed a problem that permits to vary the complexity along the imbalance dimension: the *imbalanced multiplexer* [24].

3.1 The Imbalanced Multiplexer

The *multiplexer* [32] is a completely-balanced problem defined for binary strings of size ℓ, where the first $\log_2 \ell$ bits are the *address bits* and the remaining bits are the *position bits*. The output is the value of the position bit indicated by the decimal value of the address bits. The *imbalanced multiplexer* [24] permits to control the imbalance complexity of the multiplexer by undersampling the class labeled as '1'. In the remainder of this paper, we use the *imbalance ratio* – i.e., the ratio between the number of instances sampled of the majority and the minority class – to refer to the imbalance complexity.

3.2 XCS on the Imbalanced Multiplexer

We ran XCS on the 11-bit multiplexer with the following configuration[1]: N = 800, $\beta = 0.2$, $\theta_{GA} = 25$, $\epsilon_0 = 1$, $\alpha = 0.1$, $\nu = 5$, $\chi = 0.8$, $\mu = 0.04$, $\theta_{del} = 20$, $\delta = 0.1$, $\theta_{sub} = 200$, $P_{\#} = 0.6$. We used roulette wheel selection, two-point crossover, and niched mutation [10] for the genetic algorithm. Subsumption was applied in the GA, but not in the action set. Figure 1 shows the percentage of correct classifications of the majority class (TN rate) and the percentage of correct classifications of the minority class (TP rate) obtained by XCS with imbalance ratios from $ir = 1$ to $ir = 512$. Curves are averaged over ten runs. Note that, for $ir = 1$, the same proportion of instances per class is sampled; on the other hand, for $ir = 512$, there are 512 instances of the majority class sampled for each instance of the minority class.

Figure 1a, shows that the TN rate quickly raises to 100% for any imbalance ratio tested. On the other hand, Fig. 1b illustrates that XCS only achieves 100% TP rate for $ir \leq 16$. For $ir = 32$, the TP rate is about 95% after 10^6 iterations, reaching 100% after $2 \cdot 10^6$ explore trials (not shown in the graph for a better visibility). However, for higher imbalance ratios, XCS cannot achieve 100% of TP rate. For $ir = 64$, the TP rate remains below 20%, and increasing the number of learning iterations does not provide any improvement. For $ir > 64$, the system classifies all the inputs as they belonged to the majority class.

To explain the degradation of the TP rate with ir, we checked the final populations for the different imbalance ratios. For the lowest imbalance ratios ($ir \leq 16$), the final populations contained few *overgeneral classifiers* – classifiers that match training instances of different classes –, all of them with high error. For $ir > 16$, the numerosity of overgeneral classifiers increased exponentially with the imbalance ratio. For $ir = 64$, overgeneral rules represented about 15% of the final population; for $ir = 128$ (see Table 1), they

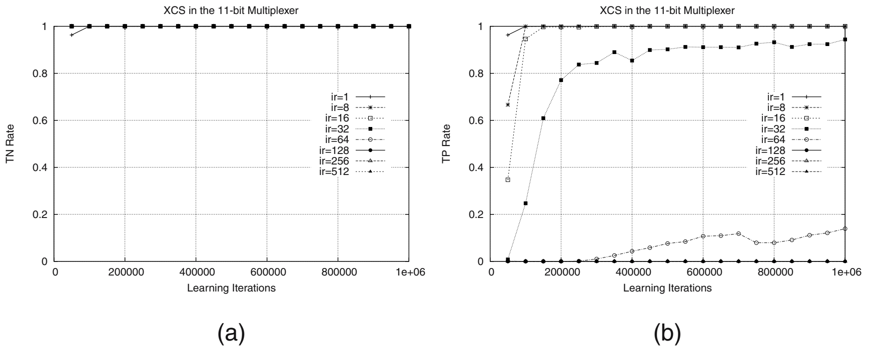


Fig. 1. TN rate (**a**) and TP rate (**b**) of XCS in the 11-bit multiplexer with imbalance ratios from $ir = 1$ to $ir = 512$

[1] For notation details, the reader is referred to [10, 32, 33].

Table 1. Most numerous rules evolved in a run of XCS with the 11-bit multiplexer for $ir = 128$. Cond. in the classifier's condition, A. the action it predicts, and p, ϵ., F and num are the prediction, error, fitness and numerosity of the classifier

Cond.	A.	p	ϵ	F	num
###########	0	1,000	$1.2 \cdot 10^{-4}$	0.98	385
###########	1	$1.2 \cdot 10^{-4}$	$7.4 \cdot 10^{-5}$	0.98	366

represented about 90% of the population, and for $ir = 256$, all the classifiers in the population were overgeneral. Moreover, for the highest ir, the error of overgeneral rules was lower than ϵ_0, and so, XCS considered these rules as accurate. For example, Table 1 shows that the error of the most overgeneral rules for $ir = 128$ is practically zero. In the next section, we theoretically analyze the effect of ir on the classifiers' error.

4 Modeling Parameter's Bounds

In this section, we theoretically relate the expected error of overgeneral classifiers with the imbalance ratio, and derive a bound on ir beyond which XCS will consider overgeneral classifiers as accurate. To derive the model, we assume that the imbalance ratio of the training dataset equals the imbalance ratio of the niches in the solution space (this assumption will be latter removed in Sect. 5). That is, we assume that instances of the minority class activate *starved niches*, and instances of the majority class trigger *nourished niches*. This assumption permits us to consider that there is a direct mapping between the imbalance ratio of the training set and the *small disjuncts* in the feature space.

4.1 Imbalance Bound

In balanced datasets, overgeneral classifiers will have a high error since they will cover, approximately, the same proportion of instances per class. Thus, the evolutionary pressures will discard them as long as more accurate classifiers exist in the population. However, as ir increases, these overgeneral rules receive less examples of the minority class, and so, they tend to have a lower error. At a given imbalance ratio, the error of these overgeneral rules will be less than ϵ_0; thus, they will be considered as accurate. We seek to derive the bound on the imbalance ratio to guarantee that overgeneral classifiers will be identified as inaccurate.

According to [8], the prediction p of a classifier can be approximated by:

$$p = P_c(cl) \cdot R_{max} + (1 - P_c(cl)) \cdot R_{min} \tag{2}$$

where $P_c(cl)$ is the probability that a classifier predicts the matching input correctly, R_{max} is the maximum reward, and R_{min} the minimum reward given by the environment. Then, the error of a classifier can be approximated by:

$$\epsilon = |p - R_{max}| \cdot P_c(cl) + |p - R_{min}| \cdot (1 - Pc(cl)) \tag{3}$$

For classification problems, R_{min} is usually 0, so that the prediction of a classifier can be estimated by: $p = P_c(cl) \cdot R_{max}$. Substituting p into (3), we get the following prediction error estimate:

$$\epsilon = 2R_{max} \cdot (P_c(cl) - P_c(cl)^2) \tag{4}$$

Now, let's relate $P_c(cl)$ with ir. In average, overgeneral classifiers will match ir examples of the majority class for each example of the minority class. Assuming that p is correctly estimated, a classifier would correctly predict the output for the ir instances of the majority class, and would give an erroneous prediction for the example of the minority class. Thus, $P_c(cl)$ can be approximated as:

$$P_c(cl) = \frac{ir}{1 + ir} \tag{5}$$

and its error estimate as:

$$\epsilon = 2 \cdot R_{max} \frac{ir}{(1 + ir)^2} \tag{6}$$

An overgeneral classifier will be considered inaccurate as long as:

$$\epsilon \geq \epsilon_0 \tag{7}$$

Using (6), we obtain that:

$$2 \cdot R_{max} \frac{ir}{(1 + ir)^2} \geq \epsilon_0 \tag{8}$$

which can be written as:

$$-ir^2 \epsilon_0 + 2ir(R_{max} - \epsilon_0) - \epsilon_0 \geq 0 \tag{9}$$

This represents a parabola where ϵ takes values higher than ϵ_0 for ir ranging between ir_l and ir_u, where $ir_l < ir_u$. We are concerned about the maximum imbalance ratio up to which XCS would consider overgeneral classifiers as inaccurate; that is, ir_u. Solving (9), and assuming that $\epsilon_0 << R_{max}$, we obtain the following expression:

$$ir_u \approx \frac{2R_{max}}{\epsilon_0} \tag{10}$$

That is, the maximum imbalance ratio up to which XCS will be able to detect overgeneral classifiers depends proportionally on R_{max} and inversely proportionally on ϵ_0. Substituting $\epsilon_0 = 1$ and $R_{max} = 1{,}000$, the maximum imbalance ratio is: $ir_u \approx 2{,}000$. Nonetheless, our experiments at $ir = 128$ showed that the final populations evolved by XCS consisted mostly of overgeneral classifiers, indicating that XCS was not able to maintain accurate classifiers. In the following we analyze the potential causes of these deviations.

4.2 Theoretical and Experimental Bounds: Analysis of the Deviation

We investigate the potential causes of the deviation between the experiments and the theory. The theoretical bound shows that XCS should be able to learn up to an imbalance ratio of 2,000. However, the results obtained were far from this bound.

The theoretical bound is derived from (7), where we specify that the error of the classifier should be higher than ϵ_0. Thus, we assumed that the error of the classifiers is well estimated. Since XCS learns incrementally and updates parameters based on a windowed weighted average, the effect of some rewards may be forgotten if some examples come very infrequently. In the next section, we revise the method of error estimation and whether it can suffer from high imbalance ratios.

XCS's genetic algorithm is activated on a frequency basis, which may also be prone to high imbalance ratios. As ir increases, instances of the minority class are sampled more infrequently; so, starved niches (which are matched by these instances) are activated more infrequently with respect to nourished niches. This means that nourished niches tend to receive a higher number of genetic events, having more offspring which may overtake the population. This effect is uncorrelated with the parameter estimation. Even though parameters are well estimated, XCS's genetic algorithm may be responsible for the deviation between the experiments and the theoretical bound. Section 4.4 analyses the mechanism of occurrence-based reproduction and suggests ways to counterbalance the effects.

4.3 Learning Rate and Error Estimates

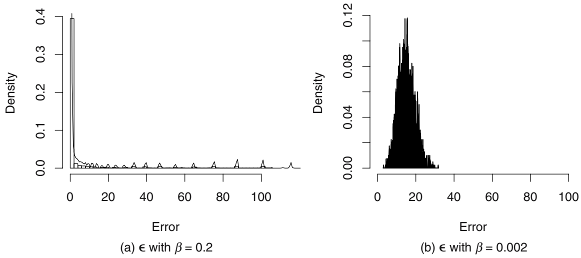
In XCS, classifier's parameters are adjusted using the standard *Widrow–Hoff delta rule* [31] with learning rate parameter β, where $0 < \beta \leq 1$ (see Sect. 2 for details). Thus, classifier's parameters are updated incrementally at learning rate β; they are expected to be stabilized to their theoretical average values after the classifier receives a certain number of updates. Nonetheless, in the experiments with high imbalance ratios, the parameters of overgeneral classifiers did not converge to their theoretical values. Note that, for $ir = 128$ (see Table 1), the population basically consisted of two overgeneral classifiers: (a) cl_1:`###########:0` with $P = 1{,}000$ and ϵ practically 0, and (b)

cl_2:###########:1 with both the error and the prediction really close to 0. The values of p and ϵ do not correspond to their theoretical ones, which can be approximated as follows (see (2) and (3)):

$$p_{cl_1} = 992.24 \qquad p_{cl_2} = 7.75 \tag{11}$$
$$\epsilon_{cl_1} = 15.38 \qquad \epsilon_{cl_2} = 15.38 \tag{12}$$

The difference between the theoretical estimates and the real values taken in the experiments may be due to the value of the learning rate β. In fact, β determines the amount of update in the parameter estimates. High values of β produce big corrections of the classifier's parameters. Usually, this allows for a faster convergence of the classifier's parameters to their real values. Low values of β would cause small corrections and so a slower convergence; for very small values, accurate offspring classifiers may loose against overgeneral parents at the beginning of the run, since their fitness increases slowly. In the reported experiments, we set $\beta = 0.2$, which is a typical value used for XCS [8, 32, 33]. In all these cases, XCS received a uniform distribution of samples, and so, classifiers of any class were updated with a similar frequency. In case of high imbalance ratios, overgeneral classifiers match one instance of the minority class every ir instances of the majority class. Consequently, high values of β may produce oscillations on the parameter estimates when the minority class instance is sampled. Small values of β may reduce the oscillations, but then the convergence time will be increased.

To check if the parameters of overgeneral classifiers oscillated, we monitored the error of the classifier ###########:0 along a single run of XCS for $\beta = 0.2$ and $\beta = 0.002$. The population was initialized with the two maximum overgeneral classifiers, and both covering and GA were switched off. The error of the classifier was sampled every 1,000 iterations. Figure 2 shows the distribution of the error for $\beta = 0.2$ (Fig. 2a and $\beta = 0.002$ (Fig. 2b)). For $\beta = 0.2$, the error of the classifier was almost zero most of the time, while the theoretical value is 7.75; moreover, there are some peaks in the distribution density, which indicate that the error momentarily changed when it received an example of the minority class, but quickly recovered the value of 0. Note that as $\epsilon \approx 0$, it satisfies that $\epsilon < \epsilon_0$, and so, the classifier is considered accurate most of the



(a) ϵ with $\beta = 0.2$

(b) ϵ with $\beta = 0.002$

Fig. 2. Evolution of the parameter ϵ of the classifier ###########:0 in the 11-bit multiplexer at imbalance ratio $ir = 128$: (**a**) using $\beta = 0.2$; (**b**) using $\beta = 0.002$

time. Decreasing β to 0.002 smoothes the density curves and the distribution becomes closer to the theoretical value of the error. Although not shown for brevity, the same conclusions can be drawn for the classifier's prediction.

The experiments herein point out that the deviation caused by the online update of classifier parameters is significant for high imbalance ratios and high values of β. Thus, we could rewrite (7) but considering the deviation with respect to the theoretical bound as follows:

$$\epsilon \pm \sigma > \epsilon_0 \tag{13}$$

where σ is the maximum deviation in the parameters of overgeneral classifiers caused by the online update mechanism.

We decreased β to 0.002 to minimize the effect of the deviations and reran XCS with the 11-bit multiplexer. We found that the classifier parameters were better estimated, but the global TP rate was not improved as overgeneral classifiers persisted in the population for higher imbalance ratios. This indicates that there are more complexities affecting XCS for high class imbalances.

4.4 Occurrence-Based Reproduction

The imbalance ratio affects the proportion of reproductive opportunities that the different classifiers receive. As *ir* increases, *starved niches* are activated less frequently, and so, accurate classifiers that belong to these niches receive a minor number of genetic events. On the other hand, accurate classifiers that form *nourished niches* and overgeneral classifiers covering several input states get a higher number of genetic events. Thus, there is a disproportion, which increases with *ir*, on the number of genetic events that classifiers belonging to starved niches receive with respect to those of overgeneral classifiers and classifiers that belong to nourished niches. Since reproduction is niche-based, but deletion is population-based, an excessive disproportion may hinder starved niches from being evolved, and eventually, accurate classifiers contained in starved niches may be removed from the population. In this section, we theoretically model the genetic opportunities of these classifiers, and suggest a method to counterbalance this disproportion.

For this purpose, we focus on the reproduction opportunities that receive (a) accurate classifiers belonging to nourished niches, (b) accurate classifiers belonging to starved niches, and (c) the most overgeneral classifiers. As the selection in XCS is niche-based, we first compute the classifier's probability of belonging to an action set, which we denote as P_{occ}.

Instances of the minority class are sampled with probability $1/(1+ir)$. As XCS chooses the class to explore randomly, the niches activated by these instances (referred as *starved niches*) are activated with the following probability:

$$P_{occ_{starved}} = \frac{1}{n \cdot m_s} \cdot \frac{1}{1+ir} \tag{14}$$

where n is the number of classes, and m_s the number of starved niches. Similarly, recognizing that instances of the majority class are sampled with probability $ir/(1+ir)$, the niches activated by these instances (addressed as *nourished niches*) have the following probability of occurrence:

$$P_{occ_{nourished}} = \frac{1}{n \cdot m_n} \cdot \frac{ir}{1+ir} \tag{15}$$

where m_n is the number of nourished niches. Finally, the most overgeneral classifiers always participate in the match set, and the action set they advocate is randomly selected with probability 1/n:

$$P_{occ_{ovg}} = \frac{1}{n} \tag{16}$$

Once an action set is activated, the parameter update procedure is triggered, and the parameter values are adjusted according to the reward received. Thus, overgeneral classifiers and classifiers that belong to nourished niches would be updated more frequently, and so, they would have more reliable estimates. In the remainder of this analysis we consider that all classifier's parameters are accurate.

An action set receives a genetic event if the average time since the last application of the GA on this action set is greater than θ_{GA}. If the period of activation T_{occ} of a niche is higher than θ_{GA}, the classifiers that belong to that niche will receive a genetic event every time the action set is formed; thus, the period of application of the GA (T_{GA}) will be $T_{GA} = T_{occ}$. Otherwise, if $T_{occ} < \theta_{GA}$, the period of application of the GA will be: $T_{GA} \approx \theta_{GA}$.

The period of occurrence T_{occ} of the three types of classifiers is:

$$T_{occ_{starved}} = n \cdot m_s \cdot (1+ir) \tag{17}$$

$$T_{occ_{nourished}} = n \cdot m_n \cdot \frac{1+ir}{ir} \tag{18}$$

$$T_{occ_{ovg}} = n \tag{19}$$

Assuming that (a) ir is high (i.e., $ir/(ir+1) \rightarrow 1$), and (b) $m_n \cdot n \leq \theta_{GA}$, and not considering overlapping classifiers, we derive the period of GA application T_{GA} for the three types of rules:

$$T_{GA_{starved}} \approx n \cdot m_s \cdot (1+ir) \tag{20}$$

$$T_{GA_{nourished}} \approx \theta_{GA} \tag{21}$$

$$T_{GA_{ovg}} \approx \theta_{GA} \tag{22}$$

Note that the time between genetic events of starved niches increases linearly with the imbalance ratio and the number of classes. For the other classifiers, T_{GA} depends only on θ_{GA}. The relation between the number of genetic opportunities received by classifiers that belong to starved niches with respect to the other classifiers is:

$$\frac{T_{GA_{starved}}}{T_{GA_{nourished}}} \approx \frac{T_{GA_{starved}}}{T_{GA_{ovg}}} \approx n \cdot m_s \cdot \frac{(1+ir)}{\theta_{GA}} \tag{23}$$

If $\theta_{GA} << ir$, overgeneral classifiers and classifiers that belong to nourished niches will receive an increasing amount of genetic opportunities with respect to classifiers belonging to starved niches, and so, they will have more offspring.

To counterbalance the number of genetic opportunities that the different niches receive, we require the following condition:

$$T_{GA_{nourished}} = T_{GA_{starved}} \tag{24}$$

Using (20) and (21), we obtain that:

$$\theta_{GA} \approx n \cdot m_s \cdot (1+ir) \tag{25}$$

In $O-$notation, and disregarding the effect of n and m_s:

$$\theta_{GA} = O(ir) \tag{26}$$

This indicates that θ_{GA} should be set according to the imbalance ratio ir to guarantee that nourished niches and starved niches will receive a similar proportion of genetic events. Also note that, under this condition, the classifiers belonging to nourished niches and overgeneral classifiers will receive more parameter's updates than the classifiers that belong to starved niches, and so, will have more accurate estimates.

4.5 Guidelines for Parameters Configuration

The above analysis provided insight into the role of some XCS's parameters and their influence on learning from imbalanced data. In the following, we derive some guidelines on how to set these parameters depending on the imbalance ratio.

First, ϵ_0 and R_{max} determine the maximum imbalance ratio up to which XCS will consider overgeneral classifiers as inaccurate classifiers (see (9)). Thus, these parameters set the threshold between negligible noise and imbalance. Regarding (7), if the error of an overgeneral classifier ($\epsilon > 0$) is smaller than ϵ_0, XCS will consider the classifier as accurate. Therefore, the few examples responsible for the error are considered as noise. Otherwise, if the error of an overgeneral classifier is higher than ϵ_0, XCS takes the classifier as inaccurate. So, the examples that make the classifier erroneous are considered as relevant examples and the classifier should not overgeneralize them.

XCS updates classifiers' parameters as a time-weighted average of their values. The learning rate parameter β adjusts the importance of the recent rewards in the parameter update mechanism. Low values of β mean that classifiers' parameters suffer small corrections every time they are updated. High values of β imply big corrections in classifier's parameters; so, parameter's estimates reflect the weighted average of few instances. In this case, we showed that the parameters of overgeneral classifiers can fluctuate, and so, overgeneral classifiers can be considered accurate during most of the learning time. Our suggestion to avoid this is to set β according to the activation frequency of the most starved niche (f_{min}) and the most nourished niche (f_{maj}), ensuring that the rewards provided when sampling instances of the minority class will be reflected in the parameters' values:

$$\beta = k_1 \cdot \frac{f_{min}}{f_{maj}} \tag{27}$$

where k_1 is an arbitrary constant. Under the initial assumptions of only having two types of niches, the starved and the nourished niches, the ratio of frequencies equals the inverse of the imbalance ratio: $f_{min}/f_{max} = 1/ir$. Thus:

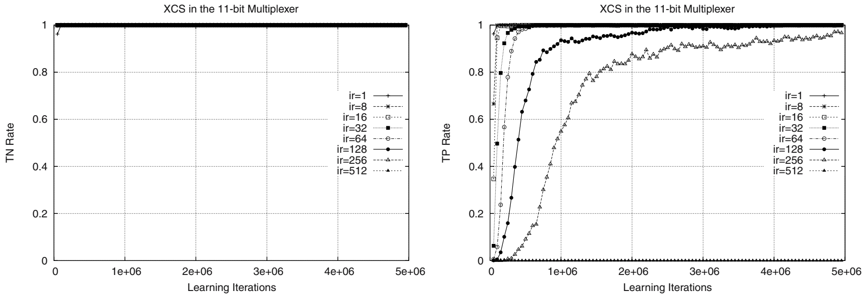
$$\beta = \frac{k_1}{ir} \tag{28}$$

Finally, Sect. 4.4 argued that θ_{GA} should increase linearly with the imbalance ratio to ensure that nourished and starved niches received a similar number of genetic events. Generalizing, we write the following equation:

$$\theta_{GA} = k_2 \cdot ir \tag{29}$$

where k_2 is an arbitrary constant. For $k_2 = 1$, all niches will receive, approximately, the same number genetic events.

We ran the same experiments with the 11-bit multiplexer but setting XCS as indicated by the configuration guidelines. We only changed the parameters of the runs that failed: $ir = \{64, 128, 256, 512\}$. Specifically, we set $\theta_{GA} = \{200, 400, 800, 1{,}600\}$ and $\beta = \{0.04, 0.02, 0.01, 0.005\}$ for each imbalance ratio respectively. Figure 3 shows the results obtained. It can be observed that XCS solves the 11-bit multiplexer up to an imbalance ratio of $ir = 256$, which supposes a big improvement with respect to the initial experiments.



(a) (b)

Fig. 3. (**a**) TN rate and (**b**) TP rate of XCS in the 11-bit multiplexer with imbalance ratios from $ir = 1$ to $ir = 512$. Parameters are configured according to the guidelines

The theoretical bound derived in Sect. 4.1 indicates that XCS might be able to solve the problem up to $ir = 2{,}000$. We got closer to this bound with appropriate parameter settings. However, we are still far from $ir = 2{,}000$. We hypothesize that the gap between the theoretical and the empirical maximum ir may be due to an insufficient number of classifiers of the minority class in the initial population. This would prevent XCS to settle the minority class niches and let them grow. As future work, we will analyze population sizing to guarantee the initial supply of classifiers belonging to starved niches at extremely high class imbalances.

5 Online Configuration of XCS to Handle Imbalanced Problems

The analysis in the last section was done assuming that there were only two types of niches in the solution space, the nourished niches and the starved niches, and that they were activated with a frequency directly proportional to the imbalance ratio. This was the case of the multiplexer problem. Nonetheless, in real-world problems, niches are unknown before running the system; consequently, niche frequencies cannot be estimated and may not be related to the imbalance ratio. In fact, the imbalance ratio reports about the proportion of examples per class, but does not provide any information about the distribution of the niches in the solution space. For example, even with a balanced dataset (i.e., $ir = 1$), there might be starved niches in the feature space.

Thus, we are concerned about the ratio between the frequencies of nourished niches and starved niches that lay closely in the feature space, rather than about the imbalance ratio of the training dataset. In this context, the guidelines proposed in Sect. 4.5 still hold, but now replacing the imbalance ratio of the training dataset ir by the *niche imbalance ratio* ir_n, defined as the ratio between the frequencies of the most nourished and the most starved niche. However, obtaining an accurate estimate of ir_n poses a big challenge to XCS, since the niches that XCS has to evolve and their frequencies are completely unknown in a real-world problem. Next, we present an approach, addressed as the *online adaptation algorithm*, that estimates ir_n from information gathered during XCS's learning; then, it substitutes this estimate in the formulas presented in the previous section to adapt XCS's parameters online.

5.1 Online Adaptation Algorithm

The online adaptation algorithm benefits from the potential information contained in the overgeneral classifiers to estimate the niche imbalance ratio ir_n of an unknown problem, and then use this estimate to tune β and θ_{GA}. Overgeneral classifiers are activated in different niches, which tend to be close in the solution space. From these overgeneral classifiers, we estimate ir_n with

Algorithm 5.1: Pseudocode for the *online adaptation algorithm*

```
Algorithm: OnlineAdaptation ( cl is classifier )
if $cl$ is $overgeneral$ then
    $ir_n := \frac{exp_{maj}(cl)}{exp_{maj}+exp_{min}(cl)}$
    if ( $ir_n < \frac{2R_{max}}{\epsilon_0} \wedge exp_{cl} > \theta_{ir} \wedge num_{cl} > \overline{num}_{[P]}$ ) then
        Adapt $\beta$ (cl)
        Adapt $\theta_{GA}$ (cl)
    end
end
```

the relative imbalance ratio in the region of the feature space that they cover, by computing the ratio between the number of instances of each class that the overgeneral classifier matches.

Algorithm 5.1 shows the pseudocode for the online adaptation algorithm. After every parameter update, the algorithm is triggered for each classifier *cl* in the match set. The first condition of the algorithm checks if *cl* is overgeneral. In this case, ir_n is calculated as the number of instances of the majority class with respect to the number of instances covered by the classifier. Then, the algorithm checks if ir_n is smaller than the maximum imbalance ratio up to which XCS should be able to distinguish an overgeneral classifier (see (10)). If the condition is satisfied, it indicates that the overgeneral rule has a higher error than what is considered as negligible noise. In this case, if the classifier has sufficient experience and high numerosity, β and θ_{GA} are adapted following the guidelines derived in Sect. 4.5.

β is adjusted so that the real prediction value of the overgeneral classifier is close to the theoretical one. To do that, we consider the worst case: we suppose that the classifier receives one example of the minority class, and then, ir_n examples of the majority class. We compute the error value that the classifier will have after receiving these ir_n+1 instances with the correspondent β value using the following series:

$$p_{ir_{\beta_0}} = R_{max} \cdot (1-\beta) \tag{30}$$

$$\forall 1 < i \leq ir_{cl} : p_{ir_{\beta_i}} = \beta(R_{max} - p_{ir_{\beta_{i-1}}}) \tag{31}$$

If $p_{ir_{\beta_i}}$ is far from the theoretical value, we decrease β by a proportion $\zeta < 1$ and repeat the same process. Consequently, the algorithm guarantees that, even in the worst case, the estimate of the classifiers parameters will be close to their real values.

Finally, the algorithm uses ir_n to tune θ_{GA} by applying (29):

$$\theta_{GA} = k_2 \cdot ir_n \approx k_2 \cdot \frac{exp_{maj}}{exp_{min} + exp_{maj}} \tag{32}$$

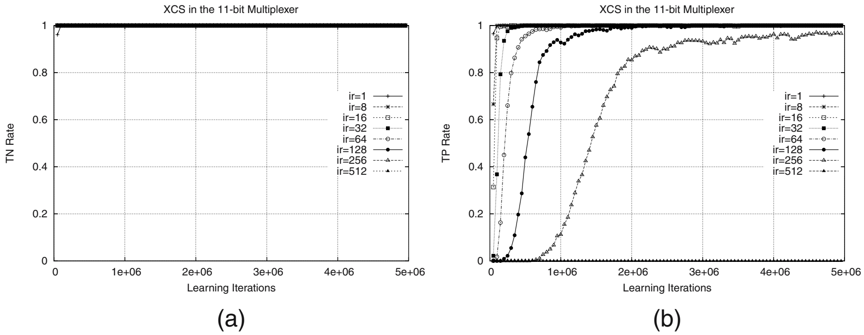


Fig. 4. (**a**) TN rate and (**b**) TP rate of XCS with online adaptation of parameters in the 11-bit multiplexer with imbalance ratios from $ir = 1$ to $ir = 512$

where k_2 fixes the minimum number of minority class examples that the starved niche has to match before going through a genetic event. This allows to counterbalance the genetic opportunities between starved and nourished niches. $k_2 = 1$ means that the classifiers belonging to starved niches are updated only once before receiving a genetic event. Higher values of k_2 allow better parameter estimates since these classifiers receive more updates between GA applications. In the experiments made in this section, we set $k_2 = 5$.

5.2 Results

Figure 4 shows the results obtained by XCS with online adaptation of parameters. The initial configuration reported in Sect. 3 was used. As β and θ_{GA} are adapted online, their initial value was set to $\theta_{GA} = 25$ and $\beta = 0.2$. The results are similar to those shown for XCS configured following the guidelines (see Fig. 3), where XCS could solve the 11-bit multiplexer up to $ir = 256$. With the online adaptation algorithm, the convergence is a little delayed since XCS needs some time to realize the existence of overgeneral classifiers, estimate ir_n, tune β and θ_{GA}, and let the evolutionary search remove overgeneral classifiers and discover accurate ones.

Let's note that this approach is essential in real-world problems, since there is not previous information about niche frequencies. In such a situation, the online adaptation algorithm introduces a significant improvement in imbalanced datasets.

6 LCS for Mining Imbalanced Datasets

XCS with online adaptation of parameters has demonstrated to be able to handle high amounts of class imbalance in artificially imbalanced problems. In this section, we investigate the capabilities of XCS for mining imbalanced data. Thus, we test XCS on a set of real-world problems with different imbalance ratios, and compare the system with other well-known machine learning techniques.

6.1 Methodology

We created a testbed consisting of 25 real-world two-class problems with different characteristics and imbalance ratios as follows. First, the following twelve problems were chosen: *balance-scale*, *bupa*, *glass*, *heart disease*, *pima indian diabetes*, *tao*, *thyroid disease*, *waveform*, *Wisconsin breast cancer database*, *Wisconsin diagnostic breast cancer*, *wine recognition data*, and *Wisconsin pronostic breast cancer*. All these problems where obtained from the UCI repository [7], except from *tao*, which was selected from a local repository [6]. For datasets with more than two classes, the discrimination of each pair of classes was considered as an individual problem. Thus, $\binom{n}{2}$ two-class problems were created from any problem with n classes (where $n > 2$), resulting in a testbed that consisted of 25 two-class real-world problems. Table 2 gathers the most relevant features of the problems.

The performance of XCS was compared to three of the most competent machine learning techniques: C4.5 [26], SMO [25] and IBk [1]. C4.5, derived from ID3, is one of the best representative decision trees which has been widely applied to tackle highly imbalanced problems. SMO is a fast method to train *support vector machines* [30] which has been able to handle very large training datasets [25]; in our experiments we used a linear kernel. IBk [1] is a nearest neighbor algorithm which decides that the output of a new input instance is the majority class of its k nearest neighbors; in the experiments, we set $k = 5$. All these machine learning methods were run using WEKA [34].

The metric of performance used in the comparison was the product of the TN rate and the TP rate, since this metric is not influenced by the imbalance ratio of the training dataset. To have good estimates of the TN rate and the TP rate, we ran the experiments on a ten-fold cross-validation [28]. After verifying that the results satisfied the condition of normality with the *Kolmogorov-Smirnov test* [29], the parametric statistical test of *repeated measures ANOVA* [27] was used to check if all the learning methods performed the same in average. Moreover, the performance of each pair of algorithms was compared using a *paired Student t-test* [28]. Next section shows the comparison among the four systems.

6.2 Results

Table 3 compares the performance of the four learners on the 25 datasets. The *repeated-measures ANOVA* did not permit to reject the null hypothesis that all the learners performed the same in average. This result is not surprising; in fact, the no-free-lunch theorem [35,36] justifies that, if no knowledge about the domain is used, no learning algorithm can systematically outperform the

Table 2. Description of the datasets properties. The columns describe the dataset identifier (Id.), the original name of the dataset (Dataset), the number of problem instances (#Ins.), the number of attributes (#At.), the proportion of minority class instances (%Min.), the proportion of majority class instances (%Maj.), and the imbalance ratio (ir)

Id.	Dataset	#Ins.	#At.	%Min.	%Maj.	ir
bald1	*balance-scale disc. 1*	625	4	7.84	92.16	11.76
bald2	*balance-scale disc. 2*	625	4	46.08	53.92	1.17
bald3	*balance-scale disc. 3*	625	4	46.08	53.92	1.17
bpa	*bupa*	345	6	42.03	57.97	1.38
glsd1	*glass disc. 1*	214	9	4.21	95.79	22.75
glsd2	*glass disc. 2*	214	9	6.07	93.93	15.47
glsd3	*glass disc. 3*	214	9	7.94	92.06	11.59
glsd4	*glass disc. 4*	214	9	13.55	86.45	6.38
glsd5	*glass disc. 5*	214	9	32.71	67.29	2.06
glsd6	*glass disc. 6*	214	9	35.51	64.49	1.82
h-s	*heart-disease*	270	13	44.44	55.56	1.25
pim	*pima-inidan*	768	8	34.90	65.10	1.87
tao	*tao-grid*	1,888	2	50.00	50.00	1.00
thyd1	*thyroid disc. 1*	215	5	13.95	86.05	6.17
thyd2	*thyroid disc. 2*	215	5	16.28	83.72	5.14
thyd3	*thyroid disc. 3*	215	5	30.23	69.77	2.31
wavd1	*waveform disc. 1*	5,000	40	33.06	66.94	2.02
wavd2	*waveform disc. 2*	5,000	40	33.84	66.16	1.96
wavd3	*waveform disc. 3*	5,000	40	33.10	66.90	2.02
wbcd	*Wis. breast cancer*	699	9	34.48	65.52	1.90
wdbc	*Wis. diag. breast cancer*	569	30	37.26	62.74	1.68
wined1	*wine disc. 1*	178	13	26.97	73.03	2.71
wined2	*wine disc. 2*	178	13	33.15	66.85	2.02
wined3	*wine disc. 3*	178	13	39.89	60.11	1.51
wpbc	*wine disc. 4*	198	33	23.74	76.26	3.21

others. However, we are interested in learners that are robust in average. For this purpose, we applied statistical pairwise comparisons (on a significance level of 0.99), which are shown as follows. The • and ◦ symbols indicate a significant degradation/improvement of the method with respect to another learner in the specific dataset. The last row of the table counts the number of times that a method has significantly degraded/improved the performance of another method.

Several observations can be drawn from the results. The overall degradation/improvement count shows that XCS is one of the most robust methods, specially for the most imbalanced datasets. Its performance is only degraded in eight occasions, the majority of which are concentrated in the problems: *bald2*, *bald3* and *tao*. *bald2* and *bald3* are two of the three datasets obtained from the discrimination of classes of the *balance-scale* problem. Both problems

Table 3. Comparison of the learning performance of C4.5, SMO, IBk, and XCS on the 25 real-world problems testbed. The • and ○ symbols indicate that the correspondent learning algorithm performed significantly worst/better on a significance level of 0.99 (pairwise t-test) than another learning algorithm for the concrete problem. The row *Avg.* averages the performance of each method over all the 25 datasets, and the row *Score* counts the number of times that a method has performed worst-better than another for a specific problem

	C4.5	SMO	IBk	XCS
bald1	0,00	0,00	0,00	0,00
bald2	69,28 ••	83,98 ○○	81,16 ○○	71,22 ••
bald3	71,21 ••	85,69 ○○	82,11 ○○	70,07 ••
bpa	33,50 ○	0,00 • • •	32,40 •○	47,22 ○○
glsd1	79,60 ○○	0,00 ••	69,32 ○	20,00 •
glsd2	33,95	15,00	24,13	59,40
glsd3	28,78	0,00	0,00	0,00
gls2c4	73,36	80,33	77,07	80,33
gls2c5	65,35 ○	9,58 • • •	62,26 ○	67,82 ○
gls2c6	52,03 ○	0,00 • • •	61,74 ○	61,08 ○
h-s	63,70	68,80	64,40	60,32
pim	44,96	48,36	46,91	46,06
tao	91,00 • ○ ○	70,57 • • •	94,25 ○ ○ ○	82,90 • • ○
thyd1	87,53	76,67	76,67	78,69
thyd2	93,12 ○	54,17 •	77,90	82,50
thyd3	87,31 ○	33,81 • • •	81,12 ○	89,74 ○
wavd1	67,80 • • •	78,65 ○○	72,28 • • ○	80,43 ○○
wavd2	62,54 • • •	72,35 ○○	67,49 • • ○	73,48 ○○
wavd3	68,61 • • •	79,61 ○○	74,14 • • ○	81,01 ○○
wbcd	89,10	92,72	92,72	92,29
wdbc	88,83	94,27 ○	93,47	90,30 •
wined1	85,58	98,46	94,98	99,23
wined2	91,83	97,51	97,50	99,17
wined3	87,64	97,14	87,94	93,43
wpbc	33,96 ○	9,37 ••	28,98 ○	20,99
Avg.	66,02	53,88	65,64	60,17
Score	14–10	20–11	7–16	8–12

have nearly the same proportion of instances per class ($ir = 1$). The reason why XCS is outperformed by SMO and IBk is not explainable, and cannot be caused by the imbalance ratio; thus, there may be other complexities affecting XCS's behavior. The *tao* problem is a completely balanced dataset ($ir = 1$) in which the boundary between classes is curved. In [4] it is shown that curved boundaries pose a challenge in XCS due to its hyperrectangular representation, which tends to concentrate high proportions of error.

On the other hand, XCS outperforms other methods in twelve occasions; from them, the datasets for which XCS outperforms more than one learner are: *bpa*, *wavd1*, *wavd2*, and *wavd3*. *bpa* is a quasi-balanced problem ($ir = 1.38$), and *wavd1*, *wavd2*, and *wavd3* are the three parwise discriminations of classes of the *waveform* problem, which present imbalance ratios of 2.02, 1.96, and 2.02 respectively. Besides, the *waveform* problem has the largest number of instances (5,000) and attributes (40) in the testbed. Although the difference between XCS and the other learners cannot be easily explained, these results indicate that XCS performs really competitively even when the size of the dataset increases. Let's also note that these results could be further explained by extracting the complexity of the training datasets. This approach was followed in [4, 5], evidencing a high correlation between some geometrical indicators of the training datasets and XCS's performance measured by the test error.

Let's now compare the learners in terms of imbalance robustness. To do that, we consider the problems with the highest *ir*; specifically, we analyze the performance on problems with $ir > 5$: *bald1*, *glsd1*, *glsd2*, *glsd3*, *glsd4*, *thyd1*, and *thyd2*. In all these problems, XCS performs really competitively. XCS is only outperformed in the problem *glsd1* by C4.5. In all the other problems, XCS performs equivalently to IBk and C4.5. SMO presents the worst performance. It has widely been shown in several works that C4.5 is able to deal with really imbalanced data [2, 19, 20]. Thus, the comparison indicates that XCS is robust (comparable to C4.5) for datasets with high disproportions of instances per class.

Finally, it is worth noting that there is not a direct mapping between the imbalance ratio in the learning dataset and the niche imbalance ratio, although both measures are related. Thus, even completely balanced datasets could cause *small niches* or *small disjuncts* to occur. As further work, we aim at designing metrics that, given a dataset, evaluate the presence of small disjuncts. This would allow to investigate the relationship between the classifiers' behavior and the presence of small disjuncts, and provide better understanding of the classifiers' performance on imbalanced datasets.

7 Summary and Conclusions

This work investigated the behavior of XCS on imbalanced classification problems. First, we empirically showed that XCS with a standard configuration can solve the multiplexer problem for moderate class imbalances ($ir \leq 32$). We identified that the number of overgeneral rules in the population tends to increase quickly with the imbalance ratio beyond a certain threshold of imbalance. For $ir > 64$, overgeneral rules represented near 100% of the population.

To provide further explanations on this tendency, we theoretically analyzed how the imbalance ratio affected the error of overgeneral classifiers, deriving a bound on the imbalance ratio under which XCS should be able to distinguish between accurate and overgeneral classifiers. As the theoretical bounds did not agree with the experimental observations, we analyzed the potential motifs of discordance. We detected that (a) the learning rate parameter β should be properly configured to ensure that overgeneral classifiers will have accurate estimates of their parameters, and (b) the GA (whose application rate is controlled by θ_{GA}) should be applied with a similar frequency to all niches to avoid that the offspring of classifiers that belong to nourished niches overtake the population. The analysis resulted in a set of recommendations on how to set β and θ_{GA} depending on the imbalance ratio, and results evidenced a significant improvement of XCS's behavior. We further argued the necessity of focusing on the niche imbalance ratio rather than in the imbalance ratio of the learning dataset to deal with real-world problems that may present small disjuncts. So, we proposed a method that estimates the niche imbalance ratio and automatically adjusts β and θ_{GA} from this estimate.

Finally, XCS was compared to C4.5, SMO and IBk on 25 real-world problems. The overall results showed that, although no learner performed statistically better than the others, XCS turned up to be really competitive to the other three machine learning techniques. The comparative analysis also denoted some differences in the performance of the learners that could not be easily explained by simply looking at the imbalance ratio. In fact, the imbalance ratio in the training dataset does not directly determines the presence of small disjuncts. Even completely balanced datasets can present small disjuncts depending on the distribution of instances around the feature space and the knowledge representation used by the learner. As further work, we propose to design metrics that evaluate the presence of small disjuncts, relating the performance of XCS with these indicators. Moreover, this information may be used in a corrective way, resampling the training dataset to diminish the presence of the small disjuncts.

Acknowledgements

The authors 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.

References

1. D.W. Aha, D.F. Kibler, and M.K. Albert. Instance-Based Learning Algorithms. *Machine Learning*, 6(1):37–66, 1991.
2. G. Batista, R.C. Prati, and M.C. Monrad. A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. *SIGKDD Explorations Newsletter*, 6(1):20–29, 2004.
3. E. Bernadó-Mansilla and J.M. Garrell. Accuracy-Based Learning Classifier Systems: Models, Analysis and Applications to Classification Tasks. *Evolutionary Computation*, 11(3):209–238, 2003.

4. E. Bernadó-Mansilla and T.K. Ho. Domain of Competence of XCS Classifier System in Complexity Measurement Space. *IEEE-TEC*, 9(1):1–23, 2005.
5. E. Bernadó-Mansilla, T.K. Ho, and A. Orriols-Puig. *Data Complexity and Evolutionary Learning: Classifierś Behavior and Domain of Competence*, pp. 115–134. Springer, Berlin Heidelberg New York, 2006.
6. E. Bernadó-Mansilla, X. Llorà, and J.M. Garrell. XCS and GALE: A Comparative Study of Two Learning Classifier Systems on Data Mining. In *Advances in Learning Classifier Systems*, volume 2321 of *LNAI*, pages 115–132. Springer, Berlin Heidelberg New York, 2002.
7. C.L Blake and C.J. Merz. *UCI Repository of Machine Learning Databases: http://www.ics.uc.edu/ mlearn/MLRepository.html.* University of California, 1998.
8. M.V. Butz, 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.
9. M.V. Butz, K. Sastry, and D.E. Goldberg. Tournament Selection in XCS. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2003)* , pages 1857–1869. Springer, Berlin Heidelberg New York, 2003.
10. M.V. Butz and S.W. Wilson. An Algorithmic Description of XCS. In P.L. Lanzi, W. Stolzmann, and S.W. Wilson, editors, *Advances in Learning Classifier Systems: Proceedings of the Third International Workshop*, volume 1996 of *Lecture Notes in Artificial Intelligence*, pages 253–272. Springer, Berlin Heidelberg New York, 2001.
11. P.K. Chan and S.J. Stolfo. Toward Scalable Learning with Non-Uniform Class and Cost Distributions: A Case Study in Credit Card Fraud Detection. In *Knowledge Discovery and Data Mining*, pages 164–168, 1998.
12. A. Van den Bosch, T. Weijters, and J. Van den Herik. When Small Disjuncts Abound, Try Lazy Learning: A Case Study. In *Proceedings Seventh BENELEARN Conference*, pages 109–118, 1997.
13. J.H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Michigan, 1975.
14. J.H. Holmes. Discovering Risk of Disease with a Learning Classifier System. In *Proceedings of 7th International Conference on Genetic Algorithms.* Morgan Kaufmann, San Mateo, 1997.
15. J.H. Holmes. Differential Negative Reinforcement Improves Classifier System Learning Rate in two-class Problems with Unequal Base Rates. In *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 635–642. Morgan Kaufmann, San Mateo, 1998.
16. J.H. Holmes. *Rule Discovery in Epidemiologic Surveillance Data Using EpiXCS: An Evoluationary Computation Approach*, volume 358, pages 444–452. Springer, Berlin Heidelberg New York, 2005.
17. R.C. Holte, L.E. Acker, and B.W. Porter. Concept Learning and the Problem of Small Disjuncts. In *IJCAI'89*, pages 813–818, 1989.
18. N. Japkowicz. Concept Learning in the Presence of Between-Class and Within-Class Imbalances. In *Proceedings of the 14th Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, pages 67–77. Springer, Berlin Heidelberg New York, 2001.
19. N. Japkowicz and S. Stephen. The Class Imbalance Problem: Significance and Strategies. In *IC-AI'00*, volume 1, pages 111–117, 2000.

20. N. Japkowicz and S. Stephen. The Class Imbalance Problem: A Systematic Study. *Intelligent Data Analisis*, 6(5):429–450, November 2002.
21. M. Kubat, R.C. Holte, and S. Matwin. Machine Learning for the Detection of Oil Spills in Satellite Radar Images. *Machine Learning*, 30(2–3):195–215, 1998.
22. A. Orriols-Puig and E. Bernadó-Mansilla. The Class Imbalance Problem in Learning Classifier Systems: A Preliminary Study. In *Genetic and Evolutionary Computation Conference (GECCO2005) workshop program*, pages 74–78. ACM Press, Washington D.C., USA, 25–29 June 2005.
23. A. Orriols-Puig and E. Bernadó-Mansilla. The Class Imbalance Problem in UCS Classifier System: Fitness Adaptation. In *Congress on Evolutionary Computation*, volume 1, pages 604–611. IEEE Press, Edinburgh, UK, 2–5 September 2005.
24. A. Orriols-Puig and E. Bernadó-Mansilla. Bounding XCS Parameters for Unbalanced Datasets. In *GECCO '06*, pages 1561–1568. ACM Press, Washington D.C., USA, 2006.
25. J. Platt. Fast Training of Support Vector Machines using Sequential Minimal Opt. In *Adv. in Kernel Methods - Support Vector Lear.* MIT, New York, 1998.
26. J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, 1995.
27. R.A. Fisher. *Statistical Methods and Scientific Inference*,2nd edition. Hafner Publishing Co, New York, 1959.
28. T.G. Dietterich. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10(7):1895–1924, 1998.
29. T. Thode. *Testing for Normality*. Marcel Dekker, New York, 2001.
30. V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, Berlin Heidelberg New York, 1995.
31. B. Widrow and M. Hoff. Adaptive Switching Circuits. In Insitute of Radio Engineers, editor, *IRE WESCON Convention Record*, volume 4, pages 96–104, 1960.
32. S.W. Wilson. Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.
33. S.W. Wilson. Generalization in the XCS Classifier System. In *3rd Annual Conference on Genetic Programming*, pages 665–674. Morgan Kaufmann, San Mateo, 1998.
34. I.H Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd edition. Morgan Kaufmann, San Francisco, 2005.
35. D.H. Wolpert. Stacked Generalization. *Neural Networks*, 5(2):241–259, 1992.
36. D.H. Wolpert. The Lack of A Priori Distinctions Between Learning Algorithms. *Neural Computation*, 8(7):1341–1390, 1996.