Class Imbalance Problem in UCS Classifier System: Fitness Adaptation

Albert Orriols
Computer Engineering Department
Enginyeria i Arquitectura la Salle
Ramon Llull University
08022 Barcelona, Spain
aorriols@salleurl.edu

Ester Bernadó-Mansilla
Computer Engineering Department
Enginyeria i Arquitectura la Salle
Ramon Llull University
08022 Barcelona, Spain
esterb@salleurl.edu

Abstract- The class imbalance problem has been said to challenge the performance of concept learning systems. Learning systems tend to be biased towards the majority class, and thus have poor generalization for the minority class instances. We analyze the class imbalance problem in learning classifier systems based on genetic algorithms. In particular we study UCS, a rule-based classifier system which learns under a supervised learning scheme. We analyze UCS on an artificial domain with varying imbalance levels. We find UCS fairly sensitive to high levels of class imbalance, to the degree that UCS tends to evolve a simple model of the feature space classified according to the majority class. We analyze strategies for dealing with class imbalances, and find fitness adaptation based on class-sensitive accuracy a useful tool for alleviating the effects of class imbalances.

1 Introduction

In the last decades, research in genetic algorithms (GAs) and evolutionary computation (EC) has paid increasing attention to machine learning and data mining applications. Particularly, *classification* has been one of the primary interests of researchers working in data mining applications of genetic algorithms. Classification can be defined as the process of assigning a class label to a given example, given a set of examples previously classified. Many approaches exist, such as decision trees, instance-based learners, neural networks, and others.

Evolutionary learning classifier systems (LCSs) have demonstrated to be highly competitive with respect to other classifier schemes in a varied range of domains. Since the first proposal, developed by Holland [Hol75, Hol76], the field has benefited from numerous research and development, being XCS [Wil95, Wil98] one of the best representatives. At the current stage of maturity, researchers have started to analyze the domain of competence of LCSs [BH05], and tested LCSs on challenging real-world classification problems [Ber02, BLG02, BB04, But04].

Research on real-world domains has identified several sources of complexity for classifier schemes, such as the geometry of class boundaries, sparsity of the available training dataset, presence of noise, and class imbalances, among others [KK01, BH05]. Class imbalances correspond to the case where one class is represented by a larger number of instances than other classes. The issue is of great importance since it appears in many real-world domains, such as fraud detection [FP97], text classification [LR94] and medical diagnosis such as thyroid diseases [BM98]. Many classifier schemes work under the assumption of balanced classes and may suffer from biases towards the majority class when the assumption does not hold.

The aim of the present work is to analyze the effects of the class imbalance problem on LCSs. Our analysis is centered on UCS classifier system [BG03], a learning classifier system based on XCS specifically designed for supervised classification problems. Due to the similarities between both systems, we expect to extend the learning behavior and results to XCS.

To isolate the class imbalance issue from other factors of complexity of LCSs, we design an artificial domain and study UCS on different levels of class imbalance. We identify a bias towards the majority class for high class imbalance levels. In the literature, some methods, working at the sampling level or at the classifier level [JS02], have been proposed to alleviate this effect. Since we attribute the bias towards the majority class examples to the generalization pressure of the genetic algorithm, which is guided by the accuracy-based fitness, we analyze strategies working at the classifier level. Particularly, we adapt fitness computation so that we include class-sensitive accuracy. The aim of this proposal is to avoid UCS's bias towards the majority class on unbalanced datasets, while keeping the original UCS's behavior in well-balanced datasets.

The remainder of this paper is organized as follows. Section 2 describes the UCS classifier system. Section 3 gives the details on the domain generation. Next, we train UCS with varying levels of class imbalance and identify the sources of difficulty for UCS. Section 5 revises strategies for dealing with class imbalances, and centers the framework for fitness adaptation in UCS. Section 6 shows the results when training UCS with fitness adaptation, finally tuning the approach in section 7 to achieve better coverage of the feature space. Finally, section 8 summarizes the main conclusions and provides directions for further work.

2 Description of UCS

UCS (sUpervised Classifier System) [Ber02, BG03] is a learning classifier system derived from XCS [Wil95, Wil98]. UCS is specifically designed for supervised learning problems, while XCS follows a reinforcement learning scheme. In the following we give a brief description of UCS. For a more detailed description, the reader is referred

to [Ber02, BG03].

2.1 Representation

UCS evolves a population [P] of individuals. Each individual is called classifier, and consists of a rule and a set of parameters estimating the quality of the rule. A rule has the structure: $condition \rightarrow class$. The condition specifies the set of examples that will be classified with the class codified in the rule.

The representation of the condition depends on the types of attributes. Within the scope of this paper, all attributes are continuous, for which we use the hyperrectangle representation. Thus, the condition part is a set of intervals $[l_i, u_i]^n$, where n is the length of the input. An input instance $x = (x_1, ..., x_n)$ satisfies the condition of a rule if $\forall i\ l_i \leq x_i \leq u_i$. The class is usually codified as an integer.

The main parameters associated to a rule are: a) the accuracy acc, b) the fitness F, c) the experience exp, d) the niche size ns, e) the last GA application time ts and f) numerosity num. Their use is described in the following.

2.2 Performance Component

UCS learns incrementally according to a supervised learning scheme. During *training*, examples coming from the training set are provided to UCS. Each example is codified by a set of attributes x = (x_1, ..., x_n) and the given class c. Then, UCS forms a match set [M] consisting of those classifiers whose conditions match the attributes of the input example. From [M], the classifiers that correctly predict class c form the correct set [C]. If [C] is empty, covering is triggered, creating new classifiers with a condition matching the example attributes and the same class as the example. Then, the parameters of classifiers are updated and eventually, the GA is applied (as described later).

In *test* mode, an input x is given, and UCS predicts its associated class. For each class, the system sums the fitness of all classifiers predicting this class. The class with the highest value is chosen as the predicted class for input x. Under *test* mode, parameter updates and the GA are disabled.

2.3 Parameter Updates

The parameters of classifiers belonging to [M] are updated. First, the experience (exp) is increased. It corresponds to the number of examples that the classifier has covered. Next, the classifier's accuracy is updated:

$$acc = \frac{number\ of\ correct\ classifications}{experience} \tag{1}$$

Finally, fitness is computed as a function of accuracy:

$$F = (acc)^{\nu} \tag{2}$$

where ν is a parameter set by the user. A typical value is 10.

The niche set size ns stores the average number of classifiers in [C]. This is updated whenever the classifier belongs to a correct set.

2.4 Genetic Algorithm

The genetic algorithm is used as the search mechanism. It has a multimodal task: to co-evolve simultaneously a set of rules which jointly represent the target concept.

The GA is applied locally to the current [C]. This favors niching [Wil98]. The GA is triggered if the time elapsed since the last application of the GA in the current set (computed from the average ts of classifiers in [C]) exceeds a threshold θ_{GA}. If GA triggers, then it selects two parents from [C] with probability proportional to fitness and copies them. Then, the copies undergo crossover and mutation with probabilities χ and μ respectively.

The resulting offspring are introduced into the population. First, each offspring is checked for subsumption with each parent. If one of the parents is accurate and more general than the offspring, then the offspring is not introduced and its parent's numerosity is increased. Otherwise, the offspring is introduced into the population.

Inserting new classifiers into the population makes other classifiers to be deleted, if the population is full. The deletion probability of a classifier is proportional to the parameter ns. Additionally, if the classifier is sufficiently experienced and its fitness is low, its probability to be deleted is inversely proportional to its fitness. Thus, the deletion vote of each classifier is:

$$dv = \begin{cases} ns \cdot \frac{\bar{F}}{F} & \text{if } exp > \theta_{del} \text{ and } F < \delta\bar{F} \\ ns & \text{otherwise} \end{cases} \tag{3}$$

where $\bar{F}$ is the average fitness of all the population, and δ and θ_{del} are parameters set by the user. The probability of being deleted is:

$$p_{del} = \frac{dv}{\sum_{j=1}^{N} dv_j} \tag{4}$$

where N is the population size.

This leads the search towards highly fit classifiers, and at the same time balances the allocation of classifiers in the different niches.

3 Dataset Design

To analyze the class imbalance problem on UCS, we designed an artificial domain which allowed us to isolate the class imbalance factor from other complexity factors identified in LCS [BG03]. The domain has two real attributes ranging in the interval [0,1], and two classes distributed in alternating squares drawing a checkerboard in the feature space. The problem is denoted as chk.

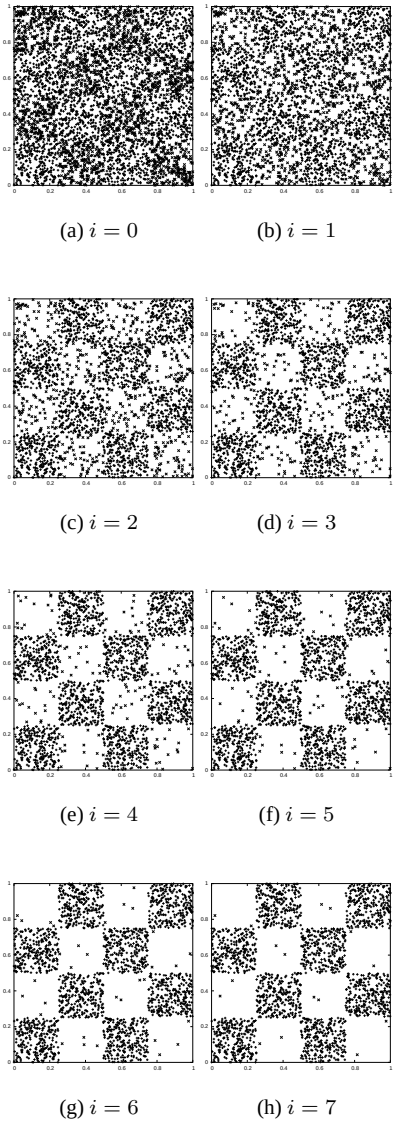
Similarly to [JS02], the complexity of the problem can be tuned along three different dimensions: the *dataset size* (s), the *concept complexity* (c), and the *imbalance level* between two classes (i). *Dataset size* is the size of the completely-balanced training set. The *concept complexity* defines the number of boundaries between the two classes, which is identified as one of the most critical complexity factors in LCSs [BH05]. The *imbalance level* determines the ratio of the number of examples between the minority class and the majority class.

The domain generation process creates a well-balanced dataset, and then proceeds to unbalance it by removing some of the minority class instances. The original balanced dataset is defined by s instances and concept complexity c, which corresponds to c^2 alternating squares. We randomly draw s points in the feature space so that each checkerboard square receives s/c^2 instances. Since the original dataset does not present any imbalance, i is set to zero.

The unbalanced datasets are generated by removing progressively half of the examples of the minority class. For $i = 1$, half of the minority class instances are removed from the well-balanced dataset. In general, the dataset at imbalance level $i > 0$ is generated by randomly removing half of the examples of the minority class from the dataset obtained at imbalance level $(i - 1)$. This means that given an imbalance level $i > 0$, the dataset generated contains s/c^2 instances in each majority class square, and $s/(2^i \cdot c^2)$ instances in each square of the minority class. Thus, for $i > 0$, the ratio between the minority class instances and majority class instances is $1/2^i$.

In our experiments, we set $s = 4096$ and $c = 4$, and only varied the imbalance level from $i = 0$ to $i = 7$. Figure 1 shows the resulting training datasets. Note that minority class instances are progressively removed from the previous level. The highest unbalanced dataset, shown in figure 1(h), has only two instances into each minority class square.

Class boundaries are lineal, which fits perfectly with the hyperrectangle codification used in UCS. This means that the problem topology would not cause any main difficulty to UCS's learning process, as long as each square of the checkerboard can be codified with only one rule. In fact, a dataset can be correctly classified with c^2 rules.



(a) $i = 0$ (b) $i = 1$

(c) $i = 2$ (d) $i = 3$

(e) $i = 4$ (f) $i = 5$

(g) $i = 6$ (h) $i = 7$

Figure 1: Training datasets for the checkerboard problem with dataset size $s = 4096$, concept complexity $c = 4$, and imbalance levels from 0 to 7. Each figure shows the position of each training point and the class to which it belongs, plotted with a different type of point.

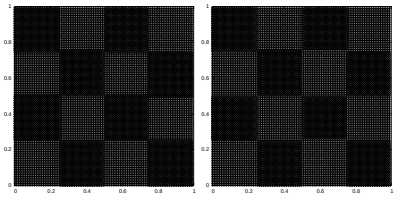
4 Training UCS with Unbalanced Class Datasets

We ran UCS with the following parameter settings (see [BW01] and [BG03] for the notation): N=400, ν=10, β=0.2, θ_{GA}=25, χ=0.8, μ=0.04, δ=0.1, θ_{del}=20, GASub = true, [A]Sub=false, $\theta_{sub} = 20$, acc_0=0.99, Specify=true, Ns =20, Ps =0.5. We trained UCS with the datasets shown in figure 1 for 200,000 learning iterations. To analyze the boundaries evolved by UCS, we tested UCS with a dense dataset which sampled the feature space with 10,000 uniformly distributed points. Then, we drew the class prediction of UCS for each point.

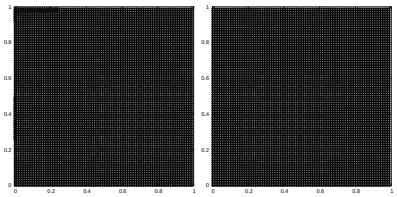
Figure 2 shows the boundaries evolved by UCS on the checkerboard problem. Boundaries of the minority class squares are plotted in black, while boundaries of the majority class squares are plotted in gray. For brevity, we only show the results for imbalance levels from $i = 2$ to $i = 5$, which reflect the most significant behavior of UCS. Figures 2(a) and 2(b) show the results for imbalance ratios of 1:4 and 1:8 respectively. In both cases, UCS was able to evolve the correct boundaries. For lower imbalance levels, UCS's boundaries were also correct. For imbalance levels $i = 4$ and higher, UCS begins to find difficulties classifying the minority class examples. Note that UCS predicted almost all the feature space as belonging to the majority class.

For a deeper analysis of UCS's behavior, let's look at the population evolved by UCS for $i = 4$, where UCS has abruptly changed its behavior with respect to the previous imbalance levels. Table 1 shows some of the classifiers of the final population, sorted by class and numerosity. For each classifier, we show its condition and class and the most important parameters. Each condition has two interval predicates, one for each dimension. For each predicate we show the lower and upper extremes of the interval. The majority class is 0 and the minority class is 1.

The table shows that, in fact, UCS evolved accurate and maximally general classifiers covering each of the squares belonging to the minority class. Actually, the eight most numerous rules are those that cover the eight minority class



(a) $i = 2$ (b) $i = 3$



(c) $i = 4$ (d) $i = 5$

Figure 2: Space model evolved by UCS with imbalance levels from 2 to 5. Black regions are those classified as the minority class and gray regions are those classified as the majority class.

squares. Additionally, the table contains other less numerous rules predicting the majority class. The problem is that these rules are too general; instead of covering only the squares of the majority class, they are generalized to the whole feature space. Note that all rules predicting class 0 have interval predicates of type $at_1 \in [0, 1]$ and $at_2 \in [0, 1]$. The accuracy of these rules is 0.94, which corresponds to the case of covering accurately all the instances of the majority class and covering wrongly the instances of the minority class. As the instances of the minority class are less frequent, their incidence in the accuracy of overgeneral rules is not very significant.

To explain UCS's tendency to evolve these overgeneral rules, we checked whether overgeneral rules were also evolved in lower imbalance levels. Surprisingly, we found that all populations evolved with imbalance levels greater than 1 contained the most general rule ($at_1 \in [0, 1]$ and $at_2 \in [0, 1]$). The only difference was that the accuracy value was different in each imbalance level; i.e., the accuracy of the overgeneral rule was greater for higher imbalance levels. Table 2 shows the population evolved for $i = 3$. Note that the population contains the eight rules corresponding to the eight minority class squares and several other overgeneral rules covering all the feature space. Accuracy of overgeneral rules is 0.89 for $i = 3$ and 0.94 for $i = 4$.

We hypothesize that these overgeneral rules are created due to the generalization pressure induced by the application of the GA to the correct sets (see [BP01] for a study on evolutionary pressures on XCS). General rules tend to participate in more correct sets, and consequently they have more reproductive opportunities. For highly unbalanced datasets, overgeneral rules have moderate fitness values, because accuracy is biased towards the majority class. In addi-

Table 1: Most numerous rules evolved by UCS in the chk problem with imbalance level i=4, sorted by class and numerosity. Columns show respectively: the rule number, the condition and class (C), where 0 is the majority class and 1 the minority class, the accuracy (acc), fitness (F), and numerosity (N).

id	condition		C	acc	F	N
1	[0.509, 0.750]	[0.259, 0.492]	1	1.00	1.00	39
2	[0.000, 0.231]	[0.252, 0.492]	1	1.00	1.00	38
3	[0.000, 0.248]	[0.755, 1.000]	1	1.00	1.00	35
4	[0.761, 1.000]	[0.000, 0.249]	1	1.00	1.00	34
5	[0.255, 0.498]	[0.520, 0.730]	1	1.00	1.00	33
6	[0.751, 1.000]	[0.514, 0.737]	1	1.00	1.00	31
7	[0.259, 0.498]	[0.000, 0.244]	1	1.00	1.00	27
8	[0.501, 0.743]	[0.751, 1.000]	1	1.00	1.00	18
9	[0.500, 0.743]	[0.751, 1.000]	1	1.00	1.00	9
10	[0.751, 1.000]	[0.531, 0.737]	1	1.00	1.00	8
		...				
18	[0.509, 0.750]	[0.246, 0.492]	1	0.64	0.01	1
19	[0.000, 1.000]	[0.000, 1.000]	0	0.94	0.54	20
20	[0.000, 1.000]	[0.000, 0.990]	0	0.94	0.54	13
21	[0.012, 1.000]	[0.000, 0.990]	0	0.94	0.54	10
		...				
64	[0.012, 1.000]	[0.038, 0.973]	0	0.94	0.54	1

Table 2: Most numerous rules evolved by UCS in the chk problem with imbalance level i=3, sorted by class and numerosity. Columns show respectively: the rule number, the condition and class (C), where 0 is the majority class and 1 the minority class, the accuracy (acc), fitness (F), and numerosity (N).

id	condition		C	acc	F	N
1	[0.251, 0.498]	[0.000, 0.244]	1	1.00	1.00	39
2	[0.501, 0.751]	[0.760, 1.000]	1	1.00	1.00	37
3	[0.000, 0.246]	[0.259, 0.500]	1	1.00	1.00	36
4	[0.259, 0.499]	[0.504, 0.751]	1	1.00	1.00	33
5	[0.506, 0.746]	[0.263, 0.498]	1	1.00	1.00	30
6	[0.751, 1.000]	[0.502, 0.749]	1	1.00	1.00	29
7	[0.752, 1.000]	[0.000, 0.240]	1	1.00	1.00	27
8	[0.000, 0.246]	[0.759, 1.000]	1	1.00	1.00	20
		...				
25	[0.000, 0.233]	[0.584, 1.000]	1	0.13	0.00	1
26	[0.000, 1.000]	[0.000, 1.000]	0	0.89	0.31	13
27	[0.010, 1.000]	[0.000, 1.000]	0	0.89	0.31	12
		...				
60	[0.051, 1.000]	[0.017, 0.926]	0	0.89	0.31	1

tion, the specific classifiers trying to cover minority examples have a lower GA exposure. Thus, overgeneral rules tend to be created and maintained in the population. In balanced or low unbalanced datasets, overgeneral rules will have low fitness. For a well-balanced dataset, an overgeneral rule covering all the feature space has an accuracy of 0.50. Thus, in these cases an overgeneral rule will not be very significant; in the case it is created, it will tend to be removed from the population due to its low fitness. At least, their numerosity will be smaller than that of overgeneral rules in high imbalance levels.

After analyzing why the overgeneral rules are created and maintained in the population as the imbalance level increases, let's return to the population shown in table 1. For imbalance level i=4, UCS generalizes all the feature space by the majority class. However, for i=3, the minority class squares are correctly classified despite the presence of overgeneral rules in the population. The reason may be found in the way that UCS selects the predicted class. Recall that UCS computes a vote weighted by fitness for each of the classes represented in [M], and chooses the class with the highest vote as the prediction for the given example. Thus, if there are many overgeneral classifiers and their fitness is moderately high, overgeneral classifiers can get a higher vote than accurate classifiers. This happens for $i = 4$ and higher imbalance levels.

5 Strategies for Dealing with Class Imbalances

There are several methods that have been proposed in the literature to deal with class imbalances. Some of the best well-known approaches are applied at the sampling level [JS02, HDW00]. They are based on sampling appropriately the training dataset so that they balance the a-priori probabilities of classes. This can be done either *oversampling* the minority class examples or *undersampling* the majority class examples. Both methods can be applied in any concept learning system, since they act as a preprocessing phase, allowing the learning system to receive the training instances as if they belonged to a well-balanced dataset. Thus, any bias of the system towards the majority class due to the different proportion of examples per class would be expected to be suppressed. However, we caution that these methods are changing somehow the available information in the training dataset, and probably their success depends on the topology and geometry of class boundaries.

Other contributions deal with the class imbalance problem at the classifier system level [Hol98]. We have focused on this approach. Having identified the class imbalance problem in UCS system, we propose a modification in the system which aims to alleviate the class imbalance effect without changing the system behavior on well-balanced datasets.

The class imbalance problem reported in section 4 has been ascribed to the fact that fitness is based on accuracy, which presents a high bias towards the majority class instances, combined with the generalization tendency of the GA. This made UCS to evolve easily overgeneral classifiers that covered all the feature space. Making accuracy class-sensitive rather than instance-sensitive could help in the identification of overgeneral classifiers predicting large regions of the search space as belonging to the majority class. The idea is to restrict classifiers to cover regions formed by examples of a single class. Thus, we modify accuracy so that each class is considered equally important regardless of the number of instances representing each class. The method will be referred as *class-sensitive accuracy*.

Next section analyzes UCS under class-sensitive accuracy on the set of chk problems at different levels of class imbalances. As a future work, we acknowledge that the study could also be extended to the analysis of sampling strategies and their comparison with the proposed strategy.

6 Class-Sensitive Accuracy

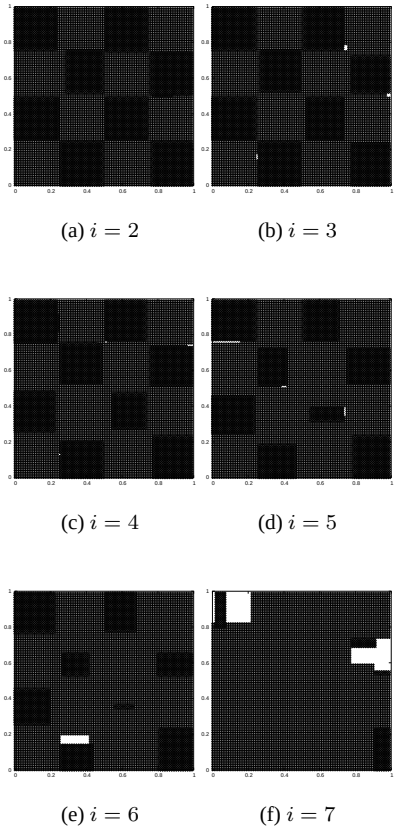
We modify the way in which accuracy is computed in UCS as described as follows. For each classifier, we compute the accuracy on each of the classes acc_i separately. We also compute individually the number of examples that the classifier covers of each class. We name it as the classifier's experience on that class, and denote as exp_i. The compound accuracy is then obtained by the average of all individual accuracies whose experience is higher than 0:

$$acc = \frac{1}{C_e} \sum_{i=1|exp_i>0}^{C} acc_i \tag{5}$$

where C is the number of classes of the problem, and C_e is the number of different classes that the rule covers (i.e., the number of classes where exp_i is greater than 0). Note that we average only the accuracies of the classes such that $exp_i > 0$. Changing accuracy also changes fitness so that we expect the GA to be guided towards classifiers predicting only instances of a single class.

We ran UCS with class-sensitive accuracy in the chk problem using the same parameter settings as in section 4. Figure 3 shows the results. Figures 3(a) and 3(b) show the models evolved with imbalance levels $i = 2$ and $i = 3$ respectively. In these cases, both UCS and UCS with class-sensitive accuracy evolved correct models of the feature space. However, UCS with class-sensitive accuracy shows a little tendency to evolve smaller minority class regions, leaving some parts of the feature space uncovered. During learning, when a classifier that covers either a minority or a majority class region is enlarged by a genetic operator and starts covering some examples of the opposite class, its accuracy is decreased to the half, and therefore its fitness is abruptly decreased. Classifiers that cover minority-class regions have more probabilities of suffering this effect, as long as there are more majority-class instances than minority-class ones. The result is that those regions closer to the boundaries are often left uncovered.

Figures 3(c) and 3(d) show the models evolved with imbalance levels $i = 4$ and $i = 5$. The boundaries evolved by UCS under *class-sensitive accuracy* clearly improve those ones evolved by raw UCS. Besides, the population obtained shows that the tendency of evolving overgeneral rules is



(a) $i = 2$ (b) $i = 3$

(c) $i = 4$ (d) $i = 5$

(e) $i = 6$ (f) $i = 7$

Figure 3: Space model evolved by UCS with class-sensitive accuracy, for imbalance levels ranging from 2 to 7. Black regions are those classified as the minority class, while gray regions are those classified as the majority class. White regions are uncovered domain regions.

avoided. Table 3 depicts the most numerous rules evolved by UCS under class-sensitive accuracy for imbalance level $i = 4$. Now, we show the individual accuracy for each class rather than the compound accuracy. Note that the population does not contain any overgeneral rule, and the most numerous rules are those that predict correctly each of the 16 squares.

Finally, figures 3(e) and 3(f) show the models evolved with imbalance levels $i = 6$ and $i = 7$. As the imbalance level increases, the system is able to discover fewer minority class regions. For the highest imbalance level, UCS can only discover four regions of the minority class. Looking at the population evolved, not detailed for brevity, we hypothesize that the problem is attributable to the fact that the imbalance ratio is so high (1:128) that the problem almost derives to a sparsity problem. There are so few instances of the minority class squares that UCS can hardly evolve generalizations.

7 Weighted Class-Sensitive Accuracy

Class-sensitive accuracy causes a high deletion pressure towards overgeneral classifiers. So, it tends to remove the classifiers closer to class boundaries as long as their conditions slightly exceed the class boundary, decreasing their accuracy to 0.5. This leaves some uncovered regions in the feature space. Taking this fact to the extreme, the system would maintain only those rules that classify instances of the same class.

Table 3: Most numerous rules evolved by UCS with class-sensitive accurcy, at imbalance level i=4. Columns show respectively: the rule number, the condition and class (C), where 0 is the majority class and 1 the minority class, the accuracy for each class (a_0 and a_1), fitness (F), and numerosity (N).

id	condition		C	a_0	a_1	F	N
1	[0.485, 0.756]	[0.483, 0.753]	0	1	-	1.00	34
2	[0.000, 0.253]	[0.502, 0.756]	0	1	-	1.00	34
3	[0.252, 0.505]	[0.750, 1.000]	0	1	-	1.00	32
4	[0.753, 1.000]	[0.749, 1.000]	0	1	-	1.00	31
5	[0.737, 1.000]	[0.238, 0.515]	0	1	-	1.00	29
6	[0.499, 0.772]	[0.000, 0.277]	0	1	-	1.00	27
7	[0.000, 0.244]	[0.000, 0.248]	0	1	-	1.00	27
8	[0.225, 0.544]	[0.223, 0.529]	0	1	-	1.00	27
9	[0.252, 0.499]	[0.000, 0.207]	1	-	1	1.00	21
10	[0.752, 1.000]	[0.000, 0.242]	1	-	1	1.00	18
11	[0.751, 1.000]	[0.502, 0.738]	1	-	1	1.00	15
12	[0.506, 0.734]	[0.761, 1.000]	1	-	1	1.00	15
13	[0.510, 0.741]	[0.252, 0.479]	1	-	1	1.00	13
14	[0.000, 0.233]	[0.757, 1.000]	1	-	1	1.00	12
15	[0.000, 0.240]	[0.254, 0.485]	1	-	1	1.00	11
16	[0.252, 0.488]	[0.516, 0.743]	1	-	1	1.00	6
17	[0.252, 0.498]	[0.516, 0.692]	1	-	1	1.00	6
18	[0.000, 0.227]	[0.757, 1.000]	1	-	1	1.00	4
19	[0.504, 0.772]	[0.000, 0.277]	0	1	-	1.00	4
		...					

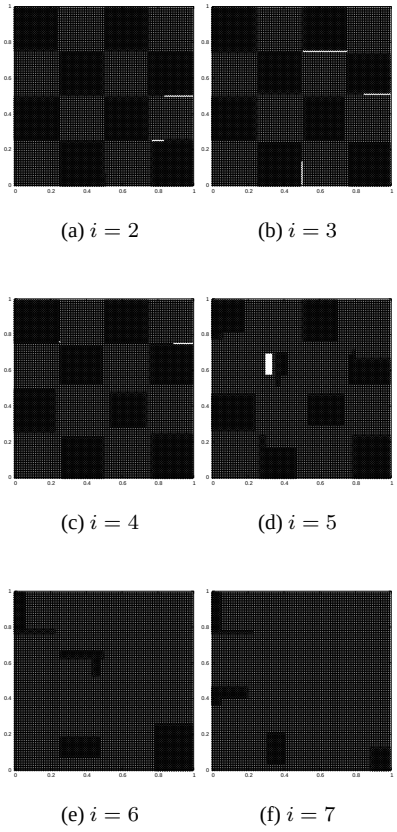
Herein, we modify the class-sensitive accuracy function to give more opportunities to classifiers approaching class boundaries while they are not much experienced. We expect that giving them more recombination opportunities, new better classifiers will be generated before the overgeneral ones are removed from the population. The modification only applies if the classifier has some class accuracies with little experience ($exp_i < \theta_{acc}$) and other class accuracies with high experience ($exp_i \geq \theta_{acc}$). In the remaining cases, we use the accuracy function of formula 5. In the former case, the compound accuracy is weighted according to each class experience:

$$acc = \frac{1}{C_e} \sum_{i=1|exp_i>0}^{C} acc_i \cdot w_i \tag{6}$$

where w_i weights the contribution of each class accuracy to the compound accuracy depending on the classifier's experience in each class. It is computed as follows:

$$w_i = \begin{cases} \frac{exp_i}{\theta_{acc}} & if\ 0 < exp_i < \theta_{acc} \\ \frac{C_e \cdot \theta_{acc} - \sum_{i=1|0<exp_i<\theta_{acc}}^{C} exp_i}{C_{ee} \cdot \theta_{acc}} & if\ exp_i \geq \theta_{acc} \end{cases} \tag{7}$$

where C_{ee} is the number of experienced classes (i.e., the number of classes whose accuracy experience is higher than θ_{acc}), and θ_{acc} is a threshold below which the class accuracy is considered inexperienced. Thus, for each rule we compute separately the accuracy for each class (acc_i) and the



(a) $i = 2$ (b) $i = 3$

(c) $i = 4$ (d) $i = 5$

(e) $i = 6$ (f) $i = 7$

Figure 4: Space model evolved by UCS with weighted class-sensitive accuracy, for imbalance levels ranging from 2 to 7. Black regions are those classified as the minority class, while gray regions are those classified as the majority class. White regions are uncovered domain regions.

experience for each class (exp_i). The contribution of the ith class to the global accuracy is weighted by the ratio of exp_i to θ_{acc}, if this class accuracy is inexperienced. If the rest of the classes have experienced accuracies, their contribution to the global accuracy is the same among them.

The new expression gives more opportunities to inaccurate classifiers, smoothing the abruptness introduced by formula 5 during the first θ_{acc} participations in [M] predicting an specific class. Parameter θ_{acc} defines the opportunities given to a classifier. If it is too low, the behavior is the same as in formula 5. If it is too high, the system gives too many opportunities to inaccurate classifiers. It should be set depending on the θ_{GA} threshold, because θ_{GA} defines the GA application frequency on [C]. In all experiments made herein, $\theta_{acc} = 2 \cdot \theta_{GA} = 50$.

Figure 4 shows the models evolved with weighted class-sensitive accuracy. As expected, uncovered regions that appeared previously in figure 3 have now been reduced. Coverage has improved. However, minority class regions are harder to discover, since the new average function introduces more pressure towards more general rules. Looking at the population evolved, we see that overgeneral rules tend to appear especially for the highest imbalance levels, but to a lower extent than with the original UCS approach. In fact, for the highest imbalance levels there are so few representatives of the minority class regions that we may debate whether these points are representative of a sparse region or whether they can be attributed to noise cases. In the latter case, we would acknowledge that UCS should not evolve any distinctive region for these cases and thus, the result obtained in these cases would be desirable.

8 Conclusions

This paper analyzed UCS's behavior on class unbalanced datasets. We found that for low unbalanced datasets, UCS's boundaries are not biased by the majority class instances. However, UCS tends to evolve overgeneral rules covering large regions of the feature space. For low unbalanced datasets, the ruleset behaves as a default hierarchy and thus, the predicted boundaries are not affected significantly by these overgeneral rules. For moderate and high unbalanced datasets, overgeneral rules interfere with specific rules covering the minority class regions to the extreme that all the feature space gets classified by the majority class. In our domain, this happened for imbalance ratios equal to or greater than 1:16. We would like to extend this study to other problems to see if this ratio is generalizable to other domains, although we suspect that this will depend on the distribution of classes in the feature space.

We studied strategies to help UCS discover the correct boundaries regardless of the number of examples each contained. We identifed UCS's bias towards the majority class attributable to the generalization pressure of the genetic algorithm, coupled with the fitness guidance provided by the current accuracy computation. Thus, we proposed a fitness adaptation based on class-sensitive accuracy, which penalizes rules covering examples belonging to different classes. Results showed that UCS was able to discover the right boundaries, while avoiding the tendency to evolve overgeneral rules. Besides, UCS maintained its performance in well-balanced datasets, although we observed some difficulties covering the examples near the class boundaries. Weighted class-sensitive accuracy made UCS's learning smoother so that rules approaching class boundaries had more opportunities. Results showed better coverage of the feature space.

We studied UCS's behavior on a particular type of classification problem. As the original balanced problem imposed no difficulties to UCS system, we could vary the imbalance level and attribute the differences to this. However, the study would be much enhanced analyzing jointly the contribution to each of the complexity factors (dataset size and concept complexity) to UCS's behavior and studying to what degree class imbalance is related to them. We could also extend the analysis towards other types of problems with different topologies, including problems with multiple unbalanced classes. Finally, the addition of unbalanced real-world datasets could serve as a testbed for validating our results.

Proposals for dealing with class imbalances at the sampling level could be analyzed under this extended testbed. Class-sensitive accuracy has alleviated significantly the bias

towards majority class rules in UCS. However, as UCS is an online learner and examples come once at a time, the systems suffers from sparsity if minority class instances come very infrequently with respect to majority class instances. When the imbalance ratio is very high, UCS with class-sensitive accuracy can hardly generalize the minority class instances. In these cases, a sampling strategy such as oversampling could aid the system to see sparse examples more frequently and allow UCS to learn them. Also an strategy combining oversampling and class-sensitive accuracy could be designed.

Another possible limitation of the current approach based on class-sensitive accuracy is the presence of noise in the dataset. If the dataset contains mislabeled instances, a class-sensitive accuracy can make the system evolve boundaries too tailored to the training dataset and present few generalization capability on unseen instances. In fact, we should analyze to what extent noise can affect UCS's performance with class-sensitive accuracy. If the percentage of noisy instances is small, UCS may perceive it as sparse instances and generalize these examples under the enveloping class. It may depend strongly on the geometry of class boundaries and the distribution of these noisy instances. We suspect that oversampling and undersampling also would suffer difficulties under noisy problems. We should analyze to what extend noise interferes with strategies dealing with class imbalances.

The study could also be enhanced with the analysis of other LCSs. Preliminary results made on two evolutionary learning classifier systems, GAssist and HIDER, showed a high bias towards the prediction of the majority classes. Moreover, it would be also interesting to extend the comparison to other classifier schemes not based on evolutionary algorithms.

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 TIC2002-04036-C05-03, and *Generalitat de Catalunya* under Grants 2002SGR-00155 and 2005FI-00252.

Bibliography

[BB04] Bacardit, J. and Butz, M. V. Data Mining in Learning Classifier Systems: Comparing XCS with GAssist. In *Seventh International Workshop on Learning Classifier Systems*, 2004.

[Ber02] Bernadó, E. *Contributions to Genetic Based Classifier Systems*. PhD thesis, Enginyeria i Arquitectura la Salle, Ramon Llull University, Barcelona, 2002.

[BG03] Bernadó, E. and Garrell, J. M. Accuracy-Based Learning Classifier Systems: Models, Analysis and Applications to Classification Tasks. *Evolutionary Computation*, 11(3):209–238, 2003.

[BH05] Bernadó, E. and Ho, T. K. Domain of Competence of XCS Classifier System in Complexity Measurement Space. *IEEE Transactions on Evolutionary Computation*, 9(1):82–104, 2005.

[BLG02] Bernadó, E., Llorà, X., and Garrell, J. M. XCS and GALE: a Comparative Study of Two Learning Classifier Systems on Data Mining. In *Advances in Learning Classifier Systems, 4th International Workshop*, volume 2321 of *Lecture Notes in Artificial Intelligence*, pages 115–132. Springer, 2002.

[BM98] Blake, C.L. and Merz, C.J. *UCI Repository of machine learning databases, [http://www.ics.uc.edu/ mlearn/MLRepository.html]*. University of California, Irvine, Department of Information and Computer Sciences, 1998.

[BP01] Butz, M.V. and Pelikan, M. Analyzing the Evolutionary Pressures in XCS. In *Genetic and Evolutionary Computation Conference*, pages 935–942. San Francisco, CA: Morgan Kaufmann, 2001.

[But04] Butz, M. V. *Rule-based Evolutionary Online Learning Systems: Learning Bounds, Classification and Prediction*. PhD thesis, Illinois Genetic Algorithms Laboratory - University of Illinois at Urbana Champaign, 2004.

[BW01] Butz, M.V. and Wilson, S.W. An algorithmic description of XCS. In *Advances in Learning Classifier Systems: Proceedings of the Third International Workshop*, volume 1996 of *Lecture Notes in Artificial Intelligence*, pages 253–272. Springer, 2001.

[FP97] R.E. Fawcett and F. Provost. Adaptive Fraud Detection. *Data Mining and Knowledge Discovery*, 3(1):291–316, 1997.

[HDW00] Holmes, J.H., Durbin, D.R., and Winston, F.K. A New Bootstrapping Method to Improve Classification Performance in Learning Classifier Systems. In *Parallel Problem Solving from Nature VI*, pages 745–754, 2000.

[Hol75] Holland, J.H. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

[Hol76] Holland, J.H. Adaptation. In R. Rosen and F. Snell, editors, *Progress in Theoretical Biology*, volume 4, pages 263–293. New York: Academic Press, 1976.

[Hol98] Holmes, J.H. 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, 1998.

[JS02] Japkowicz, N. and Stephen, S. The Class Imbalance Problem: A Systematic Study. *Intelligent Data Analisis*, 6(5):429–450, November 2002.

[KK01] Kovacs, T. and Kerber, M. What makes a problem hard for XCS. In *Advances in Learning Classifier Systems: Proceedings of the Third International Workshop*, Lecture Notes in Artificial Intelligence, pages 80–99. Springer, 2001.

[LR94] Lewis, D. and Ringuette, M. A Comparison of Two Learning Algorithms for Text Categorization. In *3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 81–93, 1994.

[Wil95] Wilson, S.W. Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

[Wil98] Wilson, S.W. Generalization in the XCS Classifier System. In *Genetic Programming: Proceedings of the Third Annual Conference*, pages 665–674. Morgan Kaufmann, 1998.