Domain of Competence of XCS Classifier System in Complexity Measurement Space

Ester Bernadó Mansilla[1] and Tin Kam Ho[2]

[1]Computer Engineering Department
Enginyeria i Arquitectura La Salle, Ramon Llull University
Quatre Camins, 2. 08022 Barcelona, Spain
E-mail: esterb@salleurl.edu Tel: +34 932 902 433

[2]Computing Sciences Research Center, Bell Laboratories, Lucent Technologies
700 Mountain Avenue, 2C-425
Murray Hill, NJ 07974-0636 USA
E-mail: tkh@research.bell-labs.com Tel: +1 908 582 5989

Abstract

The XCS classifier system has recently shown a high degree of competence on a variety of data mining problems. But to what kind of problems XCS is well and poorly suited is seldom understood, especially for real-world classification problems. The major inconvenience has been attributed to the difficulty of determining the intrinsic characteristics of real-world classification problems. This paper investigates the domain of competence of XCS by means of a methodology that characterizes the complexity of a classification problem by a set of geometrical descriptors. In a study of 392 classification problems along with their complexity characterization, we are able to identify difficult and easy domains for XCS. We focus on XCS with hyperrectangle codification, which has been predominantly used for real-attributed domains. The results show high correlations between XCS's performance and measures of length of class boundaries, compactness of classes and non-linearities of decision boundaries. We also compare the relative performance of XCS with other traditional classifier schemes. Besides confirming the high degree of competence of XCS in these problems, we are able to relate the behavior of the different classifier schemes to the geometrical complexity of the problem. Moreover, the results highlight certain regions of the complexity measurement space where a classifier scheme excels, establishing a first step towards determining the best classifier scheme for a given classification problem.

Index Terms

Learning classifier systems, geometrical complexity, genetic algorithms, pattern recognition, machine learning, classification.

I. Introduction

XCS [1], [2] is a classifier system that combines reinforcement learning [3] and genetic algorithms (GA) [4], [5] to evolve a set of rules representing the target concept. XCS descends from the lineage of learning classifier systems (LCS), which were first introduced by Holland [4], [6], [7]. Its success and robust performance in a variety of domains establish XCS as one of the major developments of learning classifier systems.

Recent investigations of XCS have been focused on a variety of aspects, with the common goal of improving our understanding of the system. The result has often been better performance and wider applicability in several domains. Some of these studies investigate XCS from an algorithmic point of view. Some remarkable efforts in this direction are [8], [9]. They investigate the dynamics of the different components of XCS, and how these components interact to evolve optimal rules that represent the target concept in an accurate and compact way. Other studies focus on certain components of the algorithm, such as the deletion algorithms [10], the definition of fitness [11], or the knowledge representation [12]–[14]. Most of these studies are restricted to artificially designed problems, because they are easy to analyze and allow control of their degree of complexity to some extent.

Other types of investigations focus on the applicability of XCS to real-world domains. In [15], XCS is applied to a data mining problem with a high degree of performance, both in terms of accuracy rate and explanatory capabilities. This study is extended in [16] with a varied set of data mining problems, where XCS appears to perform competitively with respect to other classifier techniques, such as nearest neighbors and decision trees. These studies provide a comparison of accuracy rates between XCS and other classifier schemes[1] and show that XCS is competitive. However, this information is of limited use, as there is a lack of a deeper understanding of the general behavior of XCS in real-world problems, i.e., what types of problems XCS is well or poorly suited to, and the reasons of the success or failure of XCS compared to other classifier schemes. Moreover, it is well known that there is not any classifier scheme that globally dominates in every domain. Instead, there might be some types of problems where a particular classifier scheme excels [17]. Therefore, a major issue of current research is to identify the domains of competence of a particular classifier scheme. One of the major obstacles in this kind of

[1]In the context of XCS, a classifier is a rule and a set of associated parameters. XCS evolves a set of such classifiers. The term *classifier* is also used by the pattern recognition community to refer to the whole system that classifies (e.g., nearest neighbor classifier, linear classifier, etc). In section II we use this term following the terminology used in XCS. In the rest of the paper the term classifier is used in the sense of the whole system, unless properly indicated.

investigation is the difficulty of characterizing the differences between various real-world problems, and relating the classifier's behavior to such differences.

This paper is in line of recent efforts in the pattern recognition community to characterize the behavior of a classifier system related to the features of the problem. We identify the features of a problem that are most relevant to classification accuracy as measurements of the problem's geometrical complexity. The paper analyzes the domain of applicability of XCS in such a measurement space. In particular, the paper addresses the following issues:

1) *What is the complexity of a real-world classification problem?* We will analyze the sources of difficulty in a classification problem. We will focus on the description of the geometrical complexity of the problem, such as the degree of clustering of the points of the same class, the proximity between the classes, and other factors that are critical for classification accuracy. We emphasize that we consider only those descriptors that can be extracted directly from the dataset. Therefore the analysis can be applied to arbitrary real-world data without reliance on an assumption of the generating model.
2) *Given a real-world classification problem characterized by a set of complexity descriptors, is XCS well suited for that problem?* That is, given a certain classification problem, is XCS applicable? Will XCS be able to extract an accurate knowledge representation? This issue has special relevance when we apply XCS to a real-world classification problem, where the intrinsic difficulties of the problem are often unknown, and are difficult to separate from the inadequacy of the classification algorithm. With the characterization of a real-world classification problem by a set of complexity descriptors that are not directly dependent on the classifier, we are able to investigate the relation between XCS's performance and problem complexity. This study will identify which kind of problems XCS is particularly well suited and poorly suited to. In order to infer conclusions which show the general tendency of XCS in real-world problems, we will use an extended set of 392 problems. This gives better coverage on the variety of problems than some previous studies on XCS's performance [16] that relied on a small set of problems (about 20 typically) or only artificial ones [18].
3) *Given a classification problem, what is the best suited classifier scheme?* As observed previously, there is not an outstanding classifier scheme that dominates in all sorts of classification problems. There are certain types of classification problems for which particular kinds of classifiers are best suited. Thus, a central issue is to identify the good matches between classifier schemes and problems. We will analyze XCS's performance in comparison to other learning algorithms, and relate the results to the complexity of the problem. Previous studies could hardly explain why XCS was better or worse than a particular classifier, and in which cases this happened. Relating the respective performances to the complexity characterization of each problem will give us a handle in understanding what kind of problems are best suited for a particular classifier and, even more important, why.

This study is aimed at a better understanding of XCS's behavior in real-world classification problems.

Detecting where XCS has difficulties may lead to improvement of the method. The study will also lay the bases for the characterization of the problems in a space of complexity metrics, and the identification of what kind of classifiers are more appropriate for certain regions of the measurement space.

The rest of this paper is structured as follows. First, we present a brief overview of XCS, describing how the different components are designed to achieve the learning goals. Next, since XCS's performance depends both on the algorithmic components and the knowledge representation, we analyze the knowledge representation used in domains with real valued attributes. We investigate some cases where XCS encounters different levels of difficulty due to the geometry of the problem. Section IV analyzes the sources of difficulty of real-world classification problems, and proposes several measures that represent different aspects of the problem complexity. Next, we characterize the behavior of XCS with respect to the complexity of the problem, and identify regions in the measurement space where the easiest problems and the most difficult problems for XCS are located. Section VI compares XCS with other classifier schemes, trying to determine the domains of competence of each classifier scheme. Finally, we present our main conclusions and discuss future work.

II. Description of XCS

XCS represents the knowledge extracted from the problem in a set of rules. This ruleset is incrementally evaluated by means of interacting with the environment, through a *reinforcement learning scheme*, and is improved by a search mechanism based on a *genetic algorithm*. The following is a brief description of XCS. Although XCS is applicable to single-step and multi-step problems, we restrict our description of XCS to single-step tasks like classification problems within the scope of this paper. For more details, the reader is referred to [1] and [2] for an introduction of XCS, and [19] for an algorithmic description.

A. Representation

XCS evolves a population [P] of classifiers where each classifier has a rule and a set of associated parameters estimating the quality of the rules. Each rule consists of a condition part and an action part: $condition \rightarrow action$. The condition specifies the set of input states where the classifier can be applied. For binary inputs, the condition is usually represented in the ternary alphabet: $\{0, 1, \#\}^n$, where n is the length of the input string. In this case, a condition $(c_1, c_2, ..., c_n)$ matches an input example $(x_1, x_2, ..., x_n)$, if and only if $\forall i \; c_i = x_i \vee c_i = \#$. The symbol $\#$, called *don't care*, allows the formation of generalizations in the rule's condition. The action part of the rule specifies the action or class that the classifier proposes when its condition is satisfied. It is coded as an integer.

Three main parameters estimate the quality of each classifier: a) the payoff prediction p, an estimate of the payoff that the classifier will receive if its condition matches the input and its action is selected, b) the prediction error ϵ, which estimates the average error between the classifier's prediction and the received payoff and c) the fitness F, an estimate of the accuracy of the payoff prediction. There are other parameters qualifying each classifier, such as: the experience of the classifier (denoted as *exp*), the average size of the action sets where the classifier has participated (*as*), the time-step of the last application of the genetic algorithm (*ts*) and the number of actual micro-classifiers this *macroclassifier*[2] represents, called *numerosity* (*num*). These are described in the following.

B. Performance Component

At each time step, an input x is presented to the system. Given x, the system builds a match set [M], which is formed by all the classifiers in [P] whose conditions are satisfied by the input example. If the number of actions represented in [M] is less than a threshold θ_{mna}, then covering is triggered. Covering creates new classifiers with a condition matching the current input and an action selected randomly from those not present in [M]. From the resulting match set, an action must be selected and sent to the environment. For this purpose, a payoff prediction $P(a)$ is computed for each action a in [M]. $P(a)$ estimates the payoff that the system will receive if action a is chosen. It is computed as a fitness-weighted average of the predictions of all classifiers proposing that action. The system chooses the winning action based on these prediction values. The chosen action determines the action set [A] which consists of all the classifiers in [M] advocating this action.

In classification, the winning action is usually selected using either *pure explore mode* or *pure exploit mode*. In pure explore mode, the action is selected randomly. This makes sense during *training*, i.e., when the system is learning the consequences of all possible actions for a given input. In pure exploit mode, the action is selected deterministically according to the highest prediction. This is used in *test*, that is, when the system classifies new unseen instances based on the knowledge it has acquired.

C. Reinforcement Component

Once the action is sent to the environment, the environment returns a reward r, which is used to update the parameters of the classifiers in [A]. First, the prediction of each classifier is adjusted as follows:

$$p \leftarrow p + \beta(r - p) \tag{1}$$

where β $(0 \leq \beta \leq 1)$ is the learning rate. Next, the prediction error is updated:

$$\epsilon \leftarrow \epsilon + \beta(|r - p| - \epsilon) \tag{2}$$

Then, the classifier's accuracy is computed as an inverse function of the classifier's error:

$$k = \begin{cases} \alpha(\epsilon/\epsilon_0)^{-\nu} & \epsilon \geq \epsilon_0 \\ 1 & \text{otherwise} \end{cases} \tag{3}$$

[2]Classifiers in XCS are *macroclassifiers*, i.e., each classifier represents *num* micro-classifiers having identical conditions and actions [19].

where ϵ_0 ($\epsilon_0 > 0$) determines the threshold error under which a classifier is considered to be accurate. α ($0 < \alpha < 1$) and ν ($\nu > 0$) control the degree of decline in accuracy if the classifier is inaccurate [9]. Then, the relative accuracy over the action set is computed:

$$k' = \frac{k}{\sum_{cl \in [A]} k_{cl}} \tag{4}$$

and finally, the fitness F is updated according to the classifier's relative accuracy:

$$F \leftarrow F + \beta(k' - F) \tag{5}$$

Thus, fitness is an estimate of the accuracy of the classifier's prediction relative to the accuracies of the overlapping classifiers. This provides sharing [20], [21] among the classifiers belonging to the same action set.

The experience parameter exp counts the number of times that a classifier has been updated. It is increased by one each time the classifier is included in [A]. The action set size parameter as counts the average number of classifiers present in the action sets where the classifier participates. It is updated whenever the classifier is in an action set.

D. Search Component

The search component in XCS is responsible for improving the ruleset, by discovering new promising classifiers and deleting the ones that do not contribute to the knowledge. The search in XCS is based on a genetic algorithm. The GA is triggered when the average time since the last occurrence of the GA in the action set (computed from the classifiers' parameter ts) exceeds a threshold θ_{GA}. The GA takes place in the action set, rather than in the whole population. It selects two parents from the current [A] with probability proportional to fitness and copies them. The copies undergo crossover with probability χ and mutation with probability μ per allele.

The resulting offspring are introduced to the population. First, the offspring are checked for subsumption with their parents (as described later). If an offspring classifier can not be subsumed by its parents, it is inserted into the population. If the population is full, a classifier is selected for deletion. The deletion probability of a classifier is proportional to the size of the action sets where the classifier has participated (stored in the parameter as). Furthermore, if the classifier is sufficiently experienced ($exp > \theta_{del}$, where θ_{del} is a user-defined threshold) and its fitness is low, its probability to be deleted is increased in inverse proportion to its fitness [10]. This deletion algorithm biases the search towards highly fit classifiers, and at the same time balances the classifiers' allocation in the different action sets (or niches).

E. Subsumption

Whenever a classifier is created by the GA, it is checked for subsumption with its parents before being inserted into the population. If one of the classifier's parents is sufficiently experienced ($exp > \theta_{sub}$, where

θ_{sub} is a threshold set by the user), accurate ($\epsilon < \epsilon_0$) and more general than the classifier, then the classifier is discarded, and the parent's numerosity is incremented by one. This process is called *GA subsumption*. Subsumption can also be applied to the whole action set, which is termed *action set subsumption*. In this case, the most general classifier in [A], which must be both accurate and experienced, is selected, and all other classifiers in [A] are tested for subsumption against it.

Subsumption was introduced in [2] in order to eliminate some specialized classifiers from the population which were already covered by other accurate and more general classifiers. Thus, subsumption tends to condense the population towards the maximally general classifiers.

F. Summary and Interpretation

When XCS learns a classification problem, it receives inputs from the instances available in the training set and receives feedback of its classifications in the form of rewards. Usually, the environment is designed to give a maximum reward if the system predicts the correct classification and a minimum reward (usually zero) otherwise. There are other possible regimes where layered rewards are given, e.g., to specify the relative importance of different types of classification errors. In any case, XCS's goal is to maximize the rewards received from the environment. Internally, this goal is translated to the compound goal of evolving a *complete, consistent and minimal representation* of the target concept. This is achieved by the interaction of the different components described previously. In the following, we give a simplified interpretation of the role of each component of XCS. The reader is referred to [8], [9] for more detailed analyses.

The role of covering is to enforce a complete coverage of the input space. Whenever an input is not covered by any classifier, covering creates an initial classifier from where XCS launches its learning and search mechanisms in that part of the space.

The role of the reinforcement component is to evaluate the current classifiers, so that highly fit classifiers correspond to consistent (accurate) descriptions of the target concept. This is done by translating the reward received from the environment into a measure of fitness, which is used by the search component (i.e., genetic algorithm) to guide the search. Fitness is based on the accuracy of the reward prediction. Highly fit classifiers are those that accurately predict the environmental reward in all the situations where they match the input. Thus, XCS will tend to evolve classifiers that accurately predict the maximum reward and also those classifiers that accurately predict the minimum reward. A classifier of the first type is one that always classifies the correct class and is termed *consistently correct classifier*. A classifier of the second type is one that always predicts the incorrect class, and thus is termed *consistently incorrect classifier* [11], [22].

The search component tries to lead the ruleset towards the learning goals. The search is based on a genetic algorithm, which faces a multi-modal task: to co-evolve a set of accurate and maximally general classifiers, which together form a complete description of the concept. Accurate classifiers are evolved thanks to the definition of fitness. The fact that the GA takes place in the action sets rather than in the whole population produces a generalization pressure, which leads to the evolution of maximally general classifiers. This is explained by Wilson's generalization hypothesis [1]: given two accurate classifiers with one of them matching a subset of the input states matched by the other, then the more general one will win because it participates in more action sets and thus, has more reproductive opportunities. As a consequence, the more general classifier will tend to displace the specific classifier, resulting in compact representations. Subsumption is included to stress generalization and compactness as well.

The maintenance of a diverse set of classifiers in the population is necessary to have a complete description of the target concept[3]. In XCS, this is achieved via a niche GA, by means of: a) the selection operator, which applies local fitness pressure within niches, b) crossover, which is restricted to related classifiers and c) deletion, which tries to balance the size of the niches. In XCS, the niches are defined by the action sets, which are formed by a set of classifiers matching a common set of input states and a common action.

III. Knowledge Representation and Decision Boundaries

So far we have described the XCS classifier system from an algorithmic point of view. We have focused on the interaction of the different components of XCS, and how they lead the search towards accurate descriptions. Nevertheless, it should be noted that XCS's classification accuracy depends on two aspects: the algorithmic components, that is, the ability of the algorithm to guide the search, and the knowledge representation used in the rules, which determines the ability to represent the inherent knowledge in the dataset. In fact, the knowledge representation may impose a maximum bound on the accuracy rate that a classifier system can achieve. Saxon and Barry [23] illustrate an example for XCS. The system was tested on a complex problem described by categorical attributes, where the binary encoding did not allow XCS to achieve optimal performance. The same experiment run on an enumeration encoding enabled XCS to reach 100% classification accuracy. Lanzi [13], [14] also tested different representations for XCS and showed the correlation between the system's performance and the rule representation.

The codification based on the ternary alphabet (as described in section II-A) has proved to be well suited for a varied range of domains with binary attributes. For problems with real attributes, several representations are available. One of them implies the discretization of the real values into a set of nominal ranges, and then proceeds to code these nominal ranges into the binary encoding or enumeration encoding. Two limitations are associated with this approach. The first one is the difficulty in determining the number of necessary ranges, which can vary depending on each problem and each attribute. The second limitation concerns the fixed ranges imposed by the discretization: the minimum and maximum

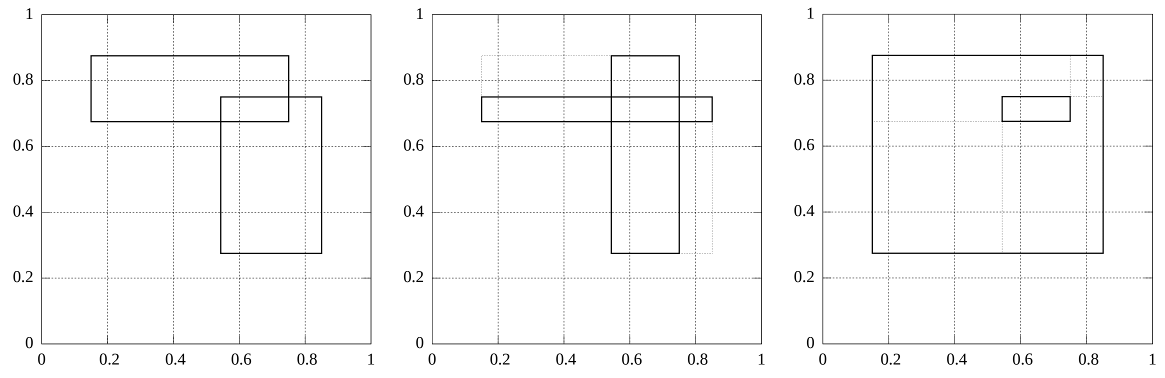
[3]If the target concept is very simple, a single classifier may suffice. But the general case is that with a complex concept that must be represented by a set of diverse classifiers.

values of each range are defined a priori, without any kind of adaptation. Although some classifier systems have used this representation [16], their performance is actually bounded by these two limitations. The hyperrectangle representation has been introduced in XCS to overcome these difficulties [12], [15]. Its main advantage is that these ranges are found adaptively by the classifier system. This representation has been widely used in a varied set of real-world classification problems and has led to high performance [15], [16]. For this reason, we adopt this representation in our experiments with XCS. This section describes the hyperrectangle codification for XCS, and studies how it approximates the boundaries between classes on some predefined artificial problems.

A. Hyperrectangle Codification

In the hyperrectangle representation, the condition of the rule is represented as a concatenation of n interval predicates of type $[l_i, u_i]$, where $l_i \leq u_i$, $1 \leq i \leq n$, and n is the number of attributes. This set of intervals describes a hyperrectangle in the feature space, where l_i and u_i are the lower and upper extremes of the hyperrectangle in each dimension i. A condition $([l_1, u_1], [l_2, u_2], \ldots, [l_n, u_n])$ matches an example $(x_1, x_2, \ldots, x_n)$ if and only if $\forall i = 1 \ldots n \; l_i \leq x_i \leq u_i$. XCSI [15] introduced this type of representation for integer attributes, where l_i and u_i were integers. Note that this representation can be easily adapted to deal with real attributes, by coding l_i and u_i as reals. For convenience, the attributes of the input examples are normalized, and thus all the intervals are restricted to the range [0,1].

The operators that directly manipulate the rules must be adapted to the new representation. This includes covering, crossover and mutation. Covering is adapted in the following way. For each attribute x_i of the input example, an interval $[l_i, u_i]$ including x_i is created, where $l_i = x_i - rand(r_0)$ and $u_i = x_i + rand(r_0)$. $rand(r_0)$ returns a random value between 0 and r_0, where r_0 is a parameter set by the user. Crossover is applied analogously to the crossover operator used in the ternary representation. A crossover point can occur anywhere in the rule, either between intervals or within an interval. Two offspring are obtained by copying from their parents the subsequences defined by the crossover points. Figure 1 shows an example of the crossover operator applied to two-attribute individuals. Figure 1(a) plots two parents and figures 1(b) and 1(c) show two possible pair of offspring that could be obtained from the recombination of the parents. Figure 1(b) shows the resulting offspring for two cut points between intervals whereas figure 1(c) shows the offspring for two cut points occurring within each interval. The mutation operator is applied to a certain allele, which can be the lower or the upper limit of the interval. Then, its value is changed by adding $\pm rand(m_0)$, where the sign is selected randomly, and m_0 is a real value set by the user. Some of these operations may produce invalid intervals, either exceeding the range [0,1] or violating the condition $l_i \leq u_i$. Thus, a repair post-process is applied in each case, and the lower and upper values are bounded so that these two conditions are satisfied.



(a) (b) (c)

Fig. 1. Example of crossover in hyperrectangle representation. Plot 1(a) draws two parent individuals in a two-attribute space. Plots 1(b) and 1(c) show two possible pairs of offspring resulting from the crossover operator. Plot 1(b) shows the offspring that results from a recombination based on crossover points occurring between the first and the second interval. Plot 1(c) shows the offspring resulting from the recombination based on two crossover points located within each of the intervals in each dimension.

B. Decision Boundaries: Case Studies

This section analyzes the properties of the decision boundaries evolved by XCS using the hyperrectangle representation. We will illustrate the limitations of the hyperrectangle representation and also their interaction with the geometry of the problem. For this purpose, we will use a graphical analysis that shows the evolved rules and the decision boundaries. To facilitate the graphical display, we restrict the analysis to two-dimensional data.

We select four classification problems that represent different geometrical situations. Unless stated differently, we train XCS with the following settings (for notation details, see [19]): $reward = 1000/0$, $N = 6400$, explore trials $= 200\,000$, $\theta_{mna} =$ number of actions, $\beta = 0.2$, $\epsilon_0 = 1.0$, $\alpha = 0.1$, $\nu = 5$, $\theta_{GA} = 25$, $\chi = 0.8$, $\mu = 0.04$, $\theta_{del} = 20$, $\delta = 0.1$, $doGASubsumption = yes$, $doActionSetSubsumption = no$, $\theta_{sub} = 30$, $r_0 = 0.6$, $m_0 = 0.1$. Figures 2 to 5 show the four problems along with the results. For each problem, we plot: (a) the training set, (b) the ruleset evolved by XCS and (c) the classification boundaries determined by XCS. Plot (a) shows the training points in the two-dimensional space depicted differently depending on the class to which they belong. The ruleset evolved by XCS shown in plot (b) is not shown entirely. Instead, we extract those rules with a high prediction value ($p > 999$), low error ($\epsilon < 10$) and high experience ($exp > 15$). This selection is performed in order to show the consistently correct rules that are mostly responsible for the classification accuracy. The remaining rules are either consistently incorrect rules or rules that the system is exploring recently and are not reliable yet. The rules are depicted using different line types, depending on the class they predict, i.e., a solid line is used for rules predicting class 0, a dashed line for rules predicting class 1, a dotted line for rules with class 2, and so forth. Plot (c) of each

TABLE I

SUMMARY OF XCS'S RESULTS IN FOUR TWO-DIMENSIONAL ARTIFICIAL PROBLEMS. COLUMNS SHOW: THE ACCURACY RATE IN TRAINING, THE TOTAL NUMBER OF RULES AND THE NUMBER OF EXPERIENCED AND CONSISTENTLY CORRECT RULES ($p > 999$, $\epsilon < 10$, $exp > 15$) EVOLVED BY XCS.

Problem	Accuracy	#Rls	#C.Rls
Checkerboard	100.00%	138	45
Tao	98.25%	620	212
Four-class	99.76%	351	125
Nested squares	95.76%	348	43

problem represents the classification boundaries induced by XCS. This classification is obtained by testing XCS against a test set which samples the search space with 10 000 uniformly distributed points. This test is done by using XCS in pure exploit mode, and therefore all the rules evolved by XCS participate in the process. Table I summarizes XCS's results in the four problems. It shows the accuracy rate achieved by XCS, the number of evolved rules and the number of experienced and correct rules, which are selected from the final ruleset as explained before.

The first problem consists of a binary class problem, where the classes are defined by 16 alternating squares distributed like a checkerboard, as depicted in figure 2. To obtain a complete and accurate description of the problem, XCS was run for 600 000 iterations. This high number of cycles implies that the problem presents some difficulties for XCS, since in other problems XCS is able to find accurate descriptions in less iterations. This difficulty can be related to the distribution of the classes: rather than being separated and clustered in few regions, they are highly interleaved. Nevertheless, the hyperrectangle representation is well suited for this problem because the boundaries between classes can be accurately approximated by rectangles. This is the reason why the final accuracy achieved by XCS on the training set is 100% (see table I). The second plot in figure 2 shows the experienced and consistently correct rules evolved by XCS. Observe how these rules approximate fairly well the geometry of the checkerboard problem, which means that XCS has been able to generalize the boundaries between classes from the available training points. The third plot shows the decision boundaries evolved by XCS. Note that the decision boundaries almost coincide with the extracted rules shown in the previous graph. This means that XCS is effectively using the consistently correct rules, which we have selected heuristically, in its decisions.

Figures 3 and 4 present two challenging problems for the hyperrectangle representation. Figure 3 plots the tao problem, which is a binary classification problem where the boundary between classes are curved. Figure 4 plots a four-class problem which also has curved boundaries between classes. Note that the generalization ability of XCS is shown in both examples. The rules tend to expand as much as possible even if there are not any examples in some regions of the search space. This generalization pressure is mostly induced by the subsumption mechanisms. Subsumption tends to reduce the population by keeping only the most general versions of the rules. Subsumption calculates the generality of a rule from its representation, not from the number of examples that it covers. For example, in the hyperrectangle codification, the generality of the rule is the proportion of the volume covered by the rule to the total volume of the search space. Thus, a rule that covers a high portion of the search space will be considered more general than another rule with less coverage, even if the proportion of covered examples is the same in both rules. That is why rules tend to expand as much as possible, even though there are no points in those regions. Also observe that the curved boundaries between classes make XCS evolve a high number of rules. This is especially evident in the tao problem, where some rules overlap in some regions, but they can not be compressed or eliminated because almost all of them are necessary to approximate the curved boundaries. XCS evolved 620 rules for the tao problem, 212 of which are experienced and consistently correct. In the four-class problem, XCS evolved 351 rules, with 125 of them being experienced and correct. The classification accuracy in the training set is 98.25% in the tao problem and 99.76% in the four-class problem.

Figure 5 presents a ten-class problem where the examples of the same class are distributed on nested squares. In this case, the first plot of figure 5 represents how the training points are distributed in nested squares. The training set is designed so that it presents unbalanced classes: the sampling of training points in the feature space is uniform, which implies that the inner classes contain less representative points than the outer classes. The exact location of each training point is not shown in the plot for clarity. The results of XCS after 800 000 iterations reveal the difficulty for XCS to evolve the inner squares. Note that the rules that should classify the inner classes are not evolved yet (see the second plot in figure 5). The classification boundaries evolved by XCS using all rules (third plot) reveal that the examples belonging to the inner squares are classified as belonging to the same class. This means that there are overgeneral rules covering that region of the search space. Since these overgeneral rules are not sufficiently accurate, they are not shown in the previous plot because of our filtering. The fact that the inner classes contain the fewest number of training examples points out that the sparsity of certain regions of the search space may be a factor of difficulty for XCS. Despite this difficulty, the classification accuracy is fairly high (95.76%). This is because the points that are improperly classified are those in the inner squares that represent a small portion of the dataset.

The difficulty of XCS to evolve the less numerous classes was also observed in the tao problem to a certain degree. There, XCS evolved first those rules belonging to the regions of the search space with higher number of examples, and later the small regions, which had few representatives in the training set. This can be explained by the evolutionary algorithms in XCS, which depend on the frequency of examples. Rules covering large regions of the search space tend to activate more often, participating in more action sets and thus receiving more reproductive opportunities than those covering small regions of the search space. The triggering mechanism of the GA tries to balance this, but when there are strong differences

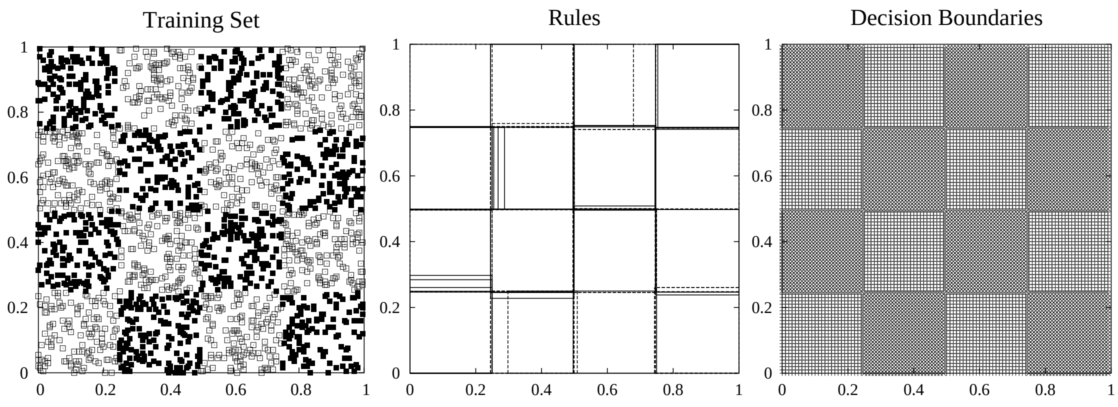


Fig. 2. Checkerboard problem. The graphs show respectively: the training points, the experienced and consistently correct rules evolved by XCS, and the decision boundaries obtained by XCS.

in the number of examples of different classification regions, this can not be avoided completely. The niche GA effectively prevents frequent rules from overtaking all the population; the GA applies locally to the action set using fitness, which is based on the accuracy of rules relative to that of the action set. So the selective pressure is applied locally to each niche or equivalently, to each region of the search space. Also the deletion mechanism enforces that all the different niches have the same size, despite the number of training examples that each action set covers. So, once the less numerous rules are discovered, the evolutionary mechanisms tend to preserve them into the population, maintaining the different niches simultaneously. Subsumption does not prevent infrequent rules from being discovered either. It might appear that subsumption tends to enforce general rules, displacing the most specific rules. But this is not true. In fact, general and accurate rules that do not overlap with specific rules can not subsume the specific ones. In case that there are overgeneral rules and specific rules covering the same regions of classification, subsumption won't be applied, since overgeneral rules, which are not accurate, can not act as subsumers. Thus, subsumption can only have the positive effect of increasing the generality of rules until these rules reach the boundaries of the classification region. So, the difficulty of covering the small regions of the space can be attributed to the triggering of the evolutionary algorithms. This behavior is also supported by other studies performed on binary artificial problems [24], [25].

XCS shows a common behavior on this set of artificial problems. It tends to evolve a complete description of the search space by means of a ruleset, where each rule tries to code the largest hyperrectangle containing only points of the same class. Although the performance obtained in these case studies is high, some sources of difficulty have become apparent. The curved boundaries between classes require that XCS evolves a large number of rules. Most of these rules overlap partially in some regions, but are necessary to approximate the boundaries. This makes a very complex knowledge representation for XCS and the performance can consequently be worsened. The distribution of the classes in the search space

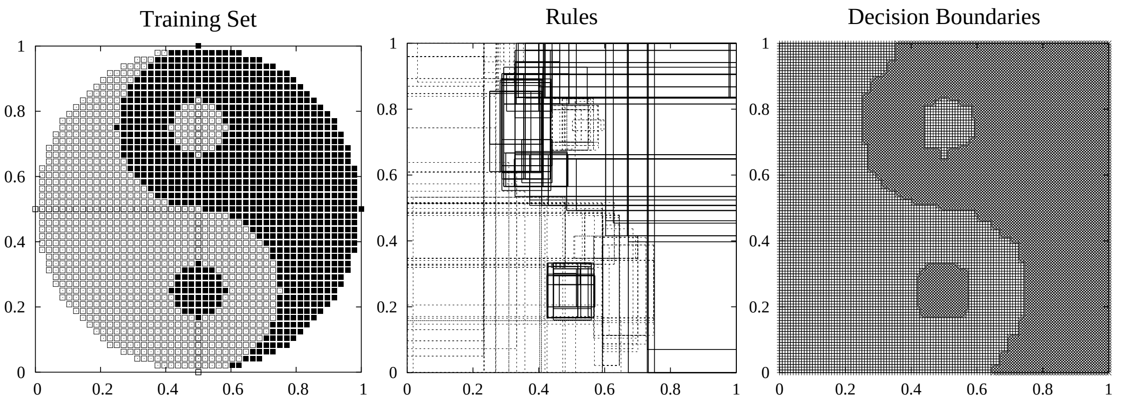


Fig. 3. Tao problem. The graphs show respectively: the training points, the experienced and consistently correct rules evolved by XCS, and the decision boundaries obtained by XCS.

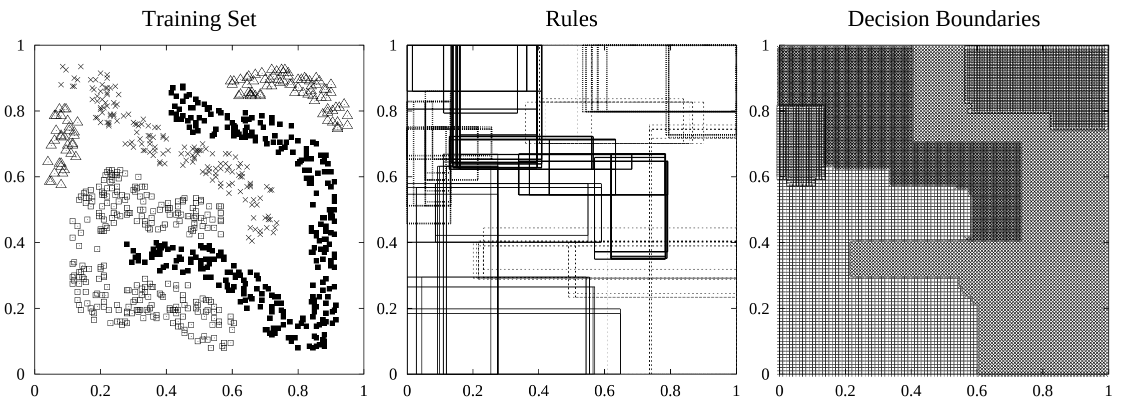


Fig. 4. Four-class problem. The graphs show respectively: the training points, the experienced and consistently correct rules evolved by XCS, and the decision boundaries obtained by XCS.

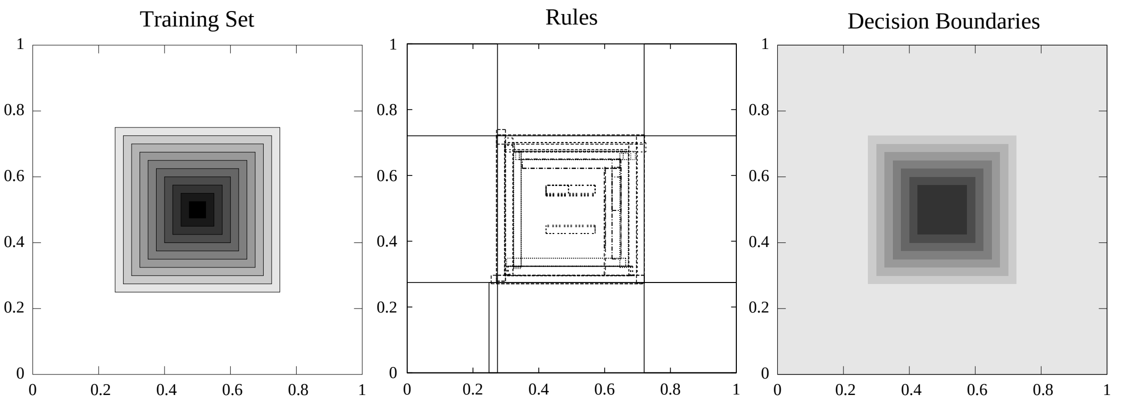


Fig. 5. Nested squares. The graphs show respectively: the training points, the experienced and consistently correct rules evolved by XCS, and the decision boundaries obtained by XCS.

may also contribute to the complexity. If the classes are wide spread and interleaved, XCS will need more complex representations. This is the case of the checkerboard problem. On the contrary, if the points of the same class are clustered, XCS will find easier descriptions. Also the sparsity of representatives of some regions of the search space is a factor in the complexity, as was observed in the problem with nested squares. Given these observations, we emphasize the need of relating the geometrical characteristics of the problem to XCS's performance and see if we can find patterns of general behavior where we can predict the degree of a problem's difficulty for XCS given its data geometry. In order to extract general observations applicable to real-world problems, we need to perform an extended analysis on more datasets, including n-dimensional problems. We also find it necessary to use more systematic ways to compute the geometrical characteristics of a problem, since a graphical analysis like the one used in these case studies is not feasible for n-dimensional data. The next section covers this issue and proposes computable measures of a problem's geometrical complexity.

IV. Characterization of Problem Complexity

A. *Sources of Difficulty of a Classification Problem*

The complexity of a classification problem can be attributed to three main different sources [26]:

- *Class ambiguity.* Some classification problems may have classes which are not distinguishable. For example, there may be instances belonging to different classes but their feature values are the same. This may be due to the intrinsic ambiguity of the problem, or to the fact that the features are not sufficient to discriminate between the classes. In the latter case, the effect may be alleviated by redefining or augmenting the set of features. Class ambiguity sets a lower bound on the achievable error rate, which is called Bayes error.
- *Training set sparsity.* The number and representativeness of the available instances in the training set influences the generalization ability of the classifier: although the classification performance in training is high, the classification accuracy when tested with unseen instances may be very poor.
- *Boundary complexity.* Some problems may have a complex boundary, which requires a long description or a complex algorithm to represent it. The boundary complexity can be characterized by its Kolmogorov complexity [27], or the minimum length of a computer program needed to reproduce it. This is independent of the class ambiguity or training set sparsity: even though enough points are provided without ambiguity in the class labels, the description of the boundary may still be very complex; in the worst case it may require an enumeration of all the points and their class labels.

Classification problems can have difficulties coming from more than one source. Class ambiguity and training set sparsity can be minimized in the sampling process to some extent. But once the dataset is fixed, there is little to do to avoid these effects. They set a minimum achievable error rate that cannot be overcome by any classifier. The boundary complexity can be characterized by the Kolmogorov complexity.

However, the Kolmogorov complexity is known to be algorithmically uncomputable, so we will rely on a set of measures emphasizing different aspects of this complexity. Our study mainly focuses on the boundary complexity of a problem, where the design of a classifier may play an important role. Additionally, we also take into account a measure of sparsity, since little representativeness in the dataset may explain the poor performance of classifiers.

B. Measures of Geometrical Complexity

Ho and Basu [28] investigate a set of 12 metrics describing different aspects of boundary complexity of a two-class problem. A diverse set of metrics is found to be necessary to represent varieties in the class distributions. Also some complexity measures may correlate well with some classifiers' performance but not with others'. From these 12 metrics, we select six of them which jointly reflect the most important aspects of problem complexity. There is evidence that these six metrics provide a reasonable measurement space in which problems as difficult as random noise are well separated from simple problems [26].

a) Length of class boundary: This measure refers to the percentage of points in the dataset that lie near the class boundary. To calculate it, we compute the minimum spanning tree (MST) [29] connecting all training samples, using the Euclidean distances between each pair of points. Then, we count the number of edges in the MST that connect two opposite classes, and we normalize this count by the total number of edges in the MST, which is always $n - 1$, where n is the number of samples. This measure is sensitive to the separability of classes and the clustering tendency of points belonging to the same class [30]. If the points of opposite classes are very interleaved, this metric will be high. Otherwise, if the points are compact within each class, the measure will be low. However, this metric cannot fully explain the complexity of a problem. For example, a linearly separable problem could have a thin and long boundary, i.e., a length of class boundary very high with many crossing edges connecting opposite classes. But another linearly separable problem could have wide margins between classes relative to the intra-class distances, and have only one edge going across the classes. On the other hand, a problem could have a very complicated and non-linear boundary with only one crossing edge between opposite classes.

b) Percentage of retained adherence subsets: This measure computes the percentage of points with their adherence subset retained. A (highest order) adherence subset associated with a point p is the largest hyperspherical neighborhood centered around p that contains only points with the same class as that of p [31]. The measure is computed as follows. First, for each point, we calculate its adherence subset. If an adherence subset of a certain point is totally enclosed inside another one, then this adherence subset is not retained. The number of retained adherence subsets is then counted and normalized by the number of points. The metric conveys information on to what extent the classes are distributed as hyperspheres in the feature space, not only on the number of hyperspheres but also on their sizes. If all points of one class are distributed within a hypersphere centered at a certain point, only one adherence subset will be retained for that class. On the other hand, if the points are spread over a long and thin structure along the class boundary, many adherence subsets will be retained.

c) Ratio of average intra/inter class nearest neighbor distances: This metric computes the dispersion of points within classes relative to the separability between classes. For each point, we calculate its nearest neighbor belonging to the same class and the nearest neighbor in the opposite class. Then the averaged distances connecting intra-class nearest neighbor points is divided by the averaged distances between inter-class nearest neighbors. The resulting ratio is used to compare the dispersion with respect to the separability of classes. A high ratio may mean that the classes are either very interleaved or dispersed over a long boundary. On the contrary, a low ratio indicates that the classes are compact.

d) Nonlinearity of nearest neighbor and linear classifiers: A measure of non-linearity proposed in [32] is considered here. First, we compute the convex hull for each class. A convex hull is estimated by a linear interpolation (with random coefficients) between randomly drawn pairs of points from the same class. Then, a classifier is trained with the original training set and tested with the extended set of points approximating the convex hull. The error of the classifier is taken as a measure of the non-linearity of the dataset. We use a linear classifier [33] and a nearest neighbor classifier [34]. The measure is sensitive to the overlapping of the convex hulls of the two classes. If the class boundary is curved and irregular, then the overlapping area will be high. The metric is also dependent on the classifier's decisions. A high error rate may mean either a high overlapping area or a poorly performing classifier unable to match the shape of the class boundary, or both reasons.

e) Maximum Fisher's discriminant ratio: We consider the discriminative power of the attributes. For each attribute, the Fisher's discriminant ratio can be computed as:

$$f = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2} \tag{6}$$

where μ_1, μ_2 and σ_1^2, σ_2^2 are the means and variances of the attribute for each of the two classes respectively. So, we have an estimate of the Fisher's discriminant ratio for each attribute. The problem will be easy if there exists at least one discriminating feature. So we only consider the maximum f over all features. However, a maximum f with zero value does not mean that the classes are inseparable; it could just be that the separating boundary is not parallel to an axis in the given feature space.

f) Training set size relative to feature space dimensionality: We also consider a measure of the sparsity of the points in the dataset, computed as the ratio of the number of points divided by the number of dimensions. Although this measure is not directly related to the geometry of the boundary, it may explain the behavior of a classifier. If the training set is sparse, the generalization ability of the system might be poor.

V. XCS's Performance and Problem Complexity

In this section, we study the relation between XCS's performance and problem complexity. This study is to determine if the complexity metrics defined previously are well suited to characterize the difficulty of a problem for XCS. We also seek to characterize the regions of the complexity space where XCS has more difficulty. Such a characterization could be useful for focusing the efforts of XCS's improvements on those difficult areas. We restrict our investigation to two-class problems, since most of the complexity metrics are well defined for these types of problems.

A. Methodology

The collection of problems used in this investigation consists of 392 problems, which were constructed as follows. First, 14 datasets were selected from the UCI-Irvine machine learning repository [35], on the basis that they contain at least 500 points with no missing values. These problems are: *abalone, car, german, kr-vs-kp, letter, lrs, nursery, pima, segmentation, splice, tic-tac-toe, vehicle, wdbc and yeast*. All the categorical values were translated into numerical values by replacing the categorical variable with a set of binary variables, one for each category. That is, a categorical variable is expanded into K binary variables, each representing membership to a distinct category. Then, the discrimination of each pair of classes in each dataset was considered as an individual problem. This resulted in 844 binary classification problems. 452 of them were found to be linearly separable by a linear programming procedure [33] and were omitted from our study. The reason of this omission is that these problems were characterized elsewhere [26], [28] as easy problems, and therefore their investigation is less interesting. The remaining 392 problems were used as our testbed.

For each problem, we estimate the complexity and the error rate obtained by XCS. The complexity is estimated according to the six metrics described previously. To estimate the error rate of XCS, we run a ten-pass two-fold cross-validation test. The procedure is as follows. The dataset is randomly permuted and then divided into two disjoint parts of approximately equal size. Then, the classifier system is trained on each of the two parts separately and tested on the other part. The error rate for this particular permutation and split is estimated as the sum of the errors on each half. Then, the dataset is randomly permuted using a different seed and the process is repeated. This was performed 10 times. The final error rate of the classifier system is the average of the 10 error rates obtained in each permutation.

XCS is run with the parameter settings given in section III-B, except for the number of explore trials which was fixed to 100 000. To choose this value, several experiments with higher training times were performed on a reduced set of problems and no differences were found with respect to the classification accuracy. The values of the parameters correspond to fairly standard settings. Even though some adjustment of parameters in a particular problem or set of problems could lead to significant improvements, this was not considered. A common set of parameters establishes a lower bound of XCS's performance on a varied set of problems. It also estimates the robustness of the method across several problems belonging to different domains and different characteristics.

B. Results

Figure 6 shows a set of scatter plots relating XCS's error rates with respect to the complexity measures. The x-axis of each plot represents a complexity metric and the y-axis represents XCS's error rate. Each point in the plot refers to one of the 392 classification problems in our testbed.

First, we examine the effects of the boundary length on the error rate, plotted in figure 6(a). Observe that there is almost a linear correlation between XCS's error rate and the percentage of points in the boundary. This means that the percentage of boundary points is a good measure of problem complexity for XCS. An examination of the error rates of other classifiers, such as the decision tree or the nearest neighbor (not shown here), also revealed the dependence between the classifier's error and this metric. Thus, the fraction of points in the boundary appears to impose a limit on the minimum achievable error rate. Nevertheless, there are some exceptions to this rule, where XCS performs reasonable well compared to other problems with the same boundary values. In particular, there are four cases with a percentage of points in boundary between 20% and 30% where XCS's error rate is below the general trend. These cases are: *car (acc - good), kr-vs-kp (no win - won), nursery (priority - spec_prior)* and *tic-tac-toe (neg. - pos.)*. Examining them, we observed that they have very low non-linearities with both nearest neighbor and linear classifiers. This points out that the interaction of several measures can define better the complexity of a problem.

There are also other observed correlations between XCS's performance and other metrics. Figure 6(c) shows the error's correlation with the dispersion within classes relative to the separation between classes. The more dispersed the dataset, the higher the error rate. Figure 6(d) plots the error rate against the non-linearity of the linear classifier (the non-linearity of the nearest neighbor is not plotted due to its similarity with the non-linearity of the linear classifier). This metric is related to the boundary's smoothness and the overlap between the convex hulls of the classes. If the metric is high, the classes are probably very interleaved, which results in a high error rate. On the contrary, a low non-linearity indicates a smooth boundary and/or low overlap and consequently, XCS's error is low.

Figure 6(b) depicts XCS's error rate against the percentage of retained adherence subsets. Although the correlation is not as notable as with the boundary metric, we can observe the following relations. There are a number of problems with small percentage of retained adherence subsets (less than 80%) that correspond to very low error rates. But a high value of retained adherence subsets does not necessarily lead to high error rates: when this metric is high, XCS's error rate is spread over a range from 0% to 50%. Nevertheless, the highest error rates of XCS (e.g., $error > 40\%$) always occur in problems with high percentage of retained adherence subsets.

The sparsity of the training set is a metric that does not influence XCS's error rate by itself, as indicated by figure 6(f). However, we observe that there are some problems where XCS's error rate is high (greater than 40%) that correspond to a ratio of number of points to number of dimensions below 50. Nor can

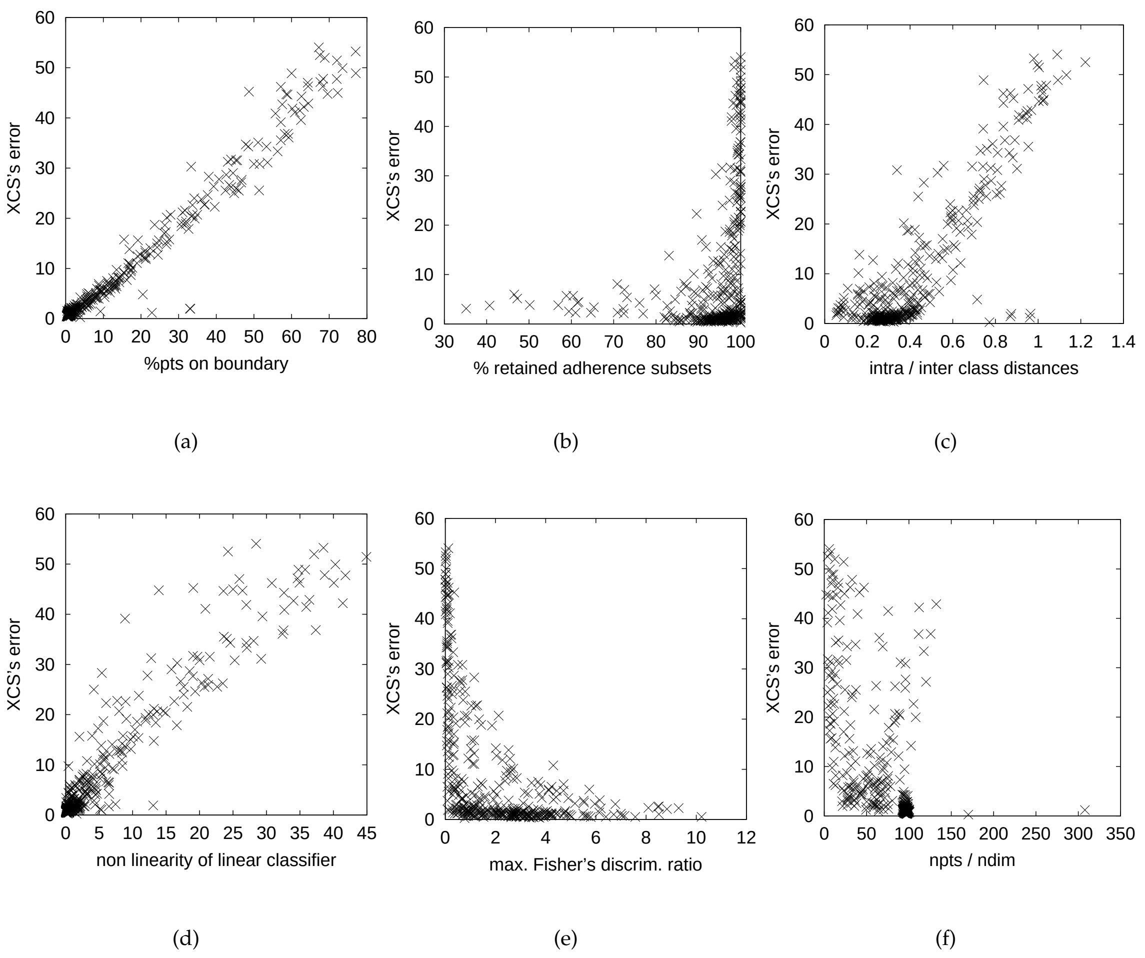


Fig. 6. XCS's error versus problem complexity. Each graph depicts the relation between XCS's error rate (y-axis) and a particular complexity metric (x-axis): (a) the percentage of points in boundary, (b) the percentage of retained adherence subsets, (c) the ratio of intra-inter class nearest neighbor distances, (d) the non-linearity of the linear classifier, (e) the maximum Fisher's discriminant ratio and (f) the ratio of the number of points to the number of dimensions. Each point plotted in a particular graph belongs to a particular classification problem.

the maximum Fisher's discriminant ratio fully explain the error rate of XCS. Still, we can extract some associations. Figure 6(e) shows that high values of maximum Fisher's discriminant ratio (greater than 3) are always associated with low error rates. These correspond to cases where one attribute discriminates fairly well and thus, the classification is easier. The converse is not necessarily true. A low value of Fisher's metric does not necessarily result in high error rates. However, note that the highest error rates all belong to low Fisher's values.

From these plots, we can draw the regions that determine the most difficult problems and the easiest problems for XCS. Examining these regions, we observe that the most difficult problems are placed in:

- high percentage of points in the boundary between classes

- high percentage of retained adherence subsets
- high dispersion within classes compared to the separation between classes
- high non-linearities
- low values of Fisher's discriminant ratio
- high training set sparsity.

On the contrary, the easiest problems are localized in:

- small percentage of points in boundary
- small dispersion compared to class separation
- low non-linearities
- and a varied range over different percentages of retained adherence subsets, maximum Fisher's discriminant ratio and sparsity.

To further confirm these observations, we plot the distribution of two extreme types of problems for XCS: very difficult problems, where XCS's error is equal or greater than 45% (an error rate almost comparable to a random classification), and very easy problems, where XCS's error is equal or less than 5%. Figure 7 shows the distribution of these two types of problems for each of the complexity metrics. To place these distributions in the appropriate context, we also plot the distribution of the rest of the problems where XCS's error rates are between 5% and 45%. Each distribution for a particular type of problem is plotted with a boxplot. The difficult problems are plotted in solid lines, easy problems in dashed lines and the remainder problems in dotted lines. The interpretation of the boxplot is the following: the extreme marks (called whiskers) show the position of the minimum and maximum values of the distribution, the lower and upper edges of the box show the 25% quartile and the 75% quartile respectively, and the line in the center of the box shows the median. Thus, the boxplot shows the spread of values, with half of the distribution placed in the box and the other half represented by the whiskers. The mean is represented by a cross. For simplicity, each metric is normalized to the range [0,1].

Looking at figure 7, we observe that the metrics that best discriminate between difficult and easy problems are the length of class boundary, the ratio of intra-inter class distances, and the non-linearities. High values of these metrics tend to correspond to difficult problems, while low values usually correspond to very easy problems. For the remaining metrics, there is a larger overlap between difficult, easy and intermediately difficult problems.

Therefore, difficult problems for XCS contain highly interleaved classes and high number of small clusters dispersed in the feature space. Since XCS is evolving partitions of the search space in the form of hyperrectangles, as long as there are more partitions needed to represent the classification regions, XCS's performance is worsened. This is in accordance with the study by Wilson in the Wisconsin Breast Cancer dataset [15]. The study observed the presence of high number of classifiers covering the regions closer to the discrimination surface, although the study by Wilson did not quantify the complexity of the problem numerically. XCS may find it difficult to evolve those small clusters dispersed and isolated in the feature space. As explained in section III, the discovery mechanisms of XCS depend on the frequency of representative examples, so sparse regions get less reproductive opportunities. Nevertheless, these results are depending on both the XCS algorithms and the chosen representation (hyperrectangle), so it is difficult to find which one is most responsible of the observed effects. We would suggest the study of other possible representations for real-world problems, such as centroid-based [12], unordered bound [36] or messy [13], [14] to analyze if this tendency remains or it less pronounced.

Using figures 6 and 7, we could extrapolate the present distribution of error rates over the complexity space to predict an expectation of accuracy of XCS for new problems. Given a particular classification problem, characterized by its complexity metrics, we could predict what level of accuracy can be expected for XCS; at least, whether the problem will be difficult, easy, or intermediately difficult for XCS. This initial estimate of the difficulty of the problem for XCS can be useful in setting the configuration parameters of XCS appropriately. A topic for future investigation is whether there are different settings of parameters appropriate for different levels of complexity and if so, how to adapt the parameter settings to the problem. For example, if the problem is predicted to be difficult for XCS, we could adjust the most critical parameters, e.g., the population size or the number of maximum explore trials could be increased. Also, identifying difficult problems for XCS and their characteristics is the first step towards the analysis of improvement of XCS in these types of problems. Once we know what kind of features present in real-world classification problems can make learning difficult, we could design artificial datasets with such features which may facilitate a deeper analysis and understanding of the classifier system's performance. This type of study would lead to an improvement of the performance and applicability of XCS in a wider range of problems.

VI. Comparative Performance of XCS

This section compares the performance of XCS to those of other classifiers. The purpose of this comparison is not only to analyze how well XCS performs with respect to other classifiers, but also to interpret these results in the context of the problem complexity. The study performed in the last section showed the suitability of the geometrical metrics to describe the complexity of the problem. We will use the same metrics to explain the performance of XCS related to that of other classifiers. Section V also highlighted some regions of the complexity space where XCS obtained high error rates and low error rates. The study is complemented here by the analysis of to which regions of the complexity space XCS is more appropriate or less appropriate than other classifier schemes. This investigation could lead us to a better understanding of to what types of problems XCS is best/worst suited. And also the converse, given a certain problem described by a set of complexity metrics, is XCS well suited? and furthermore, which classifier is best suited?

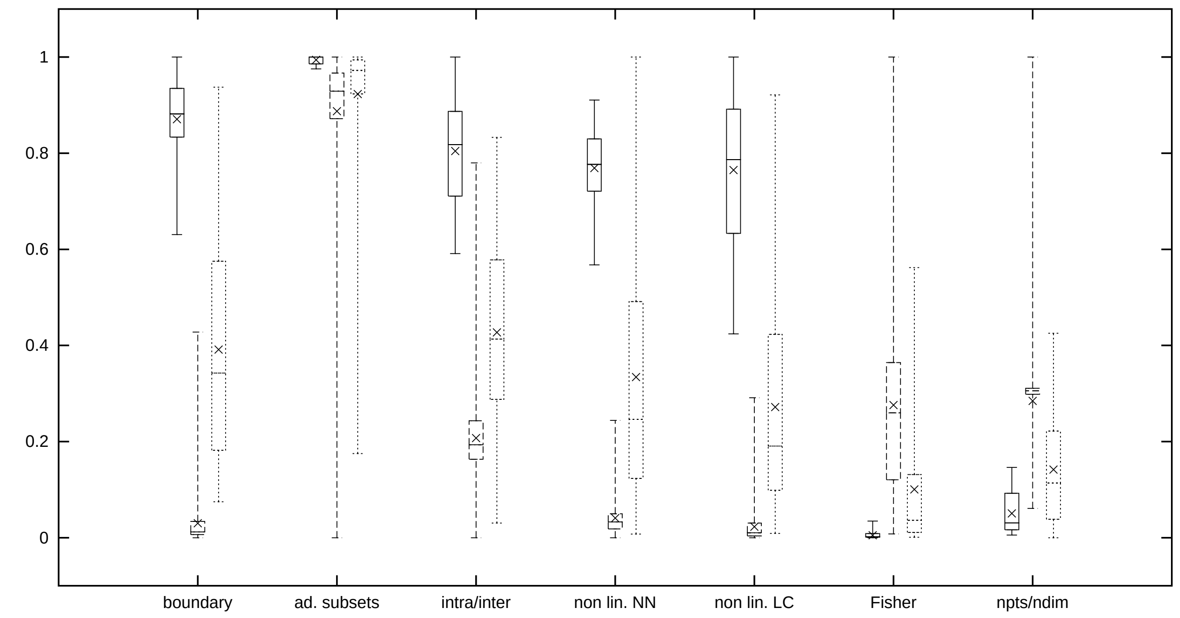


Fig. 7. Distribution of easy and difficult problems for XCS in the complexity measurement space. Difficult problems (XCS's error $\geq$ 45%) are plotted in solid lines, easy problems (XCS's error $\leq$ 5%) are plotted in dashed lines, and the rest of the problems (5% $<$ XCS's error $<$ 45%) are plotted in dotted lines. Each distribution is represented by a box plot, where the extreme points correspond to the minimum and maximum value of the distribution, and the three marks in the box correspond to the 25% quartile, the median and the 75% quartile respectively. The mean is given by a cross. The x-axis represents individually each of the metrics of complexity, and the y-axis gives the value of the metric for a particular distribution. For readability, the y-axis is normalized with respect to each of the metrics, so 0 corresponds to the minimum value of each metric in all the problems and 1 to the maximum value.

A. Methodology

XCS is compared to five types of classifier schemes, which represent different solutions to classification problems. These are: a nearest neighbor classifier, a linear classifier, a decision tree and two types of decision forests. The *nearest neighbor classifier* (NN) [34] assigns each point to the class of the nearest neighbor in the training set. The *linear classifier* (LC) separates the classes by linear boundaries, i.e., hyperplanes. We used a linear classifier computed by linear programming using the AMPL software [28], [37], [38]. The *decision tree* included in this study uses oblique hyperplanes to split the data at each internal node [39]. The hyperplanes are derived using a simplified Fisher's method, as described in [40]. Decision forests are known to outperform decision trees in a varied range of domains [41]. The decision forests used here represent two cases of *classifier combination*. The *subsample decision forest*, also known as *bagged decision trees*, is based on the bootstrapping method [42]. It constructs training subsets by randomly selecting with replacement points from the original training set. Then, it trains a decision tree with each subset and the decisions of the resulting trees on a test point are combined by averaging the posterior probabilities at the leaf nodes. The *subspace decision forest* [40] is based on the random subspace method, where the training sets for the individual decision trees are constructed by randomly selecting features instead of training points. Then, the decision trees are combined in the same way as in the subsample forest. Both decision forests use oblique decision trees. Classifier combination can have several advantages over single classifiers. In the case of decision forests, they may overcome the overfitting tendency of single trees and present more robustness on different problems.

To compare XCS with these five classifiers, we use the same datasets and methodology as explained in section V. We train each classifier in the set of 392 binary-class problems using a ten-pass two-fold cross-validation test. We obtain ten error rates for each classifier and each problem. Then, we compute the differences between the ten XCS's error rates and those of each classifier. These ten differences are used to test the significance of XCS's difference from each classifier, using a paired t-test [43] with a 95% confidence level.

B. *Results*

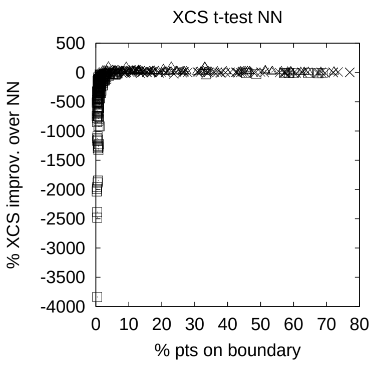
We analyze the performance of XCS relative to each of the classifier schemes included in this study. For each comparison, we present several scatter plots relating XCS's performance to each of the complexity metrics (see figures 8 - 13), so that we can analyze when XCS is better, equivalent and worse than each of the other classifiers at different levels of problem complexity. Each point in the scatter plots represents one classification problem. A triangle is used when XCS significantly improves over the classifier to which it is compared for the particular problem, a square is used when XCS is significantly worse, and a cross when no significant differences are found.

We first examine the performance of XCS compared to that of the nearest neighbor classifier (NN). Figure 8 shows different views of this comparison. Plots 8(a) - 8(g) show the percentage of improvement of XCS over NN versus each of the six geometrical descriptors. The percentage of XCS's improvement over NN is computed as:

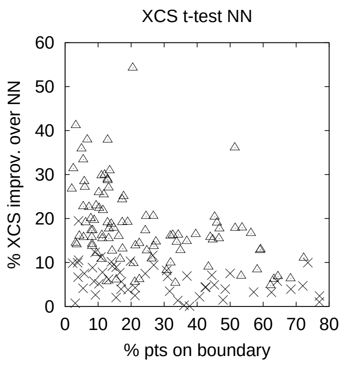
$$\frac{error(NN) - error(XCS)}{error(NN)} * 100 \tag{7}$$

where the error of a classifier is the average error of the ten permutations performed in a particular problem.

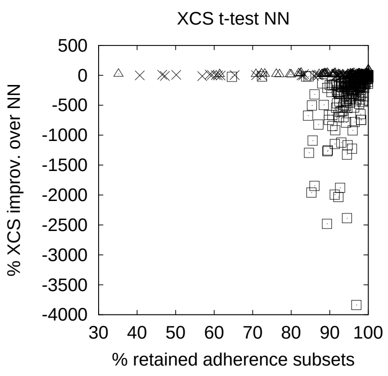
Looking at figure 8(a) which relates XCS's improvement to the percentage of points on boundary, we observe that the problems are distributed in two groups: problems in one group have very few points on boundary, and the remaining problems are spread over higher fractions of points on boundary. For very low boundary lengths, XCS performs worse than NN. The high negative values in the percentage of improvement are due to the way we compute this measure. In these problems, the average error rate of NN is very low (0.5035% in average), which results in high negative values of XCS's improvement with respect to NN. The rest of the problems mainly are cases where XCS is equivalent or better than NN. These are clearly seen in figure 8(b), where we only show the problems where XCS's percentage of improvement is positive, plotted against the percentage of points on boundary.



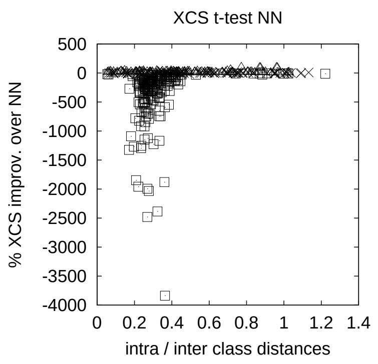
(a)



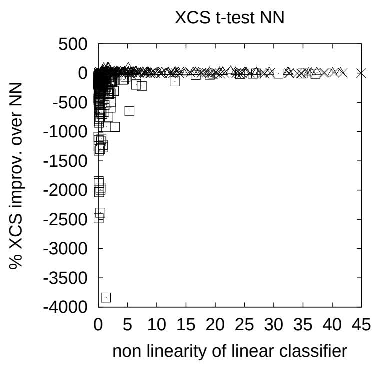
(b)



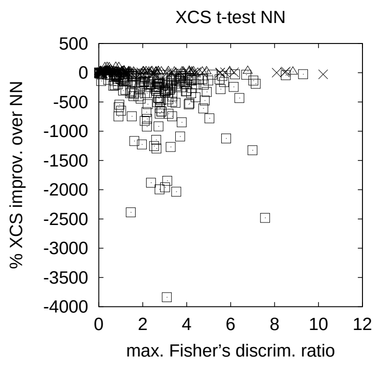
(c)



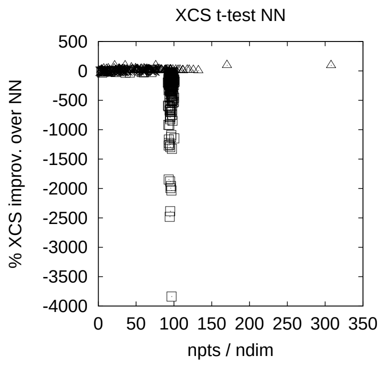
(d)



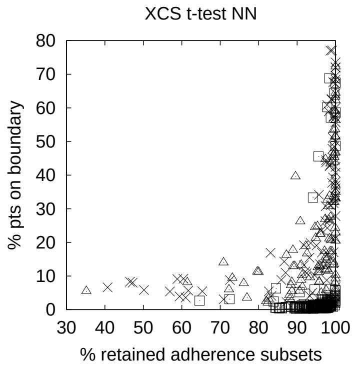
(e)



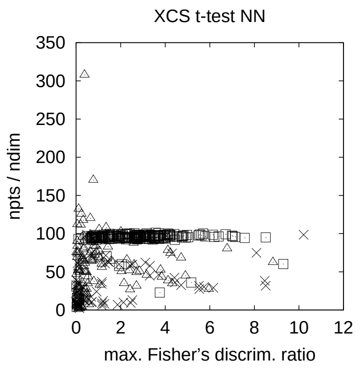
(f)



(g)

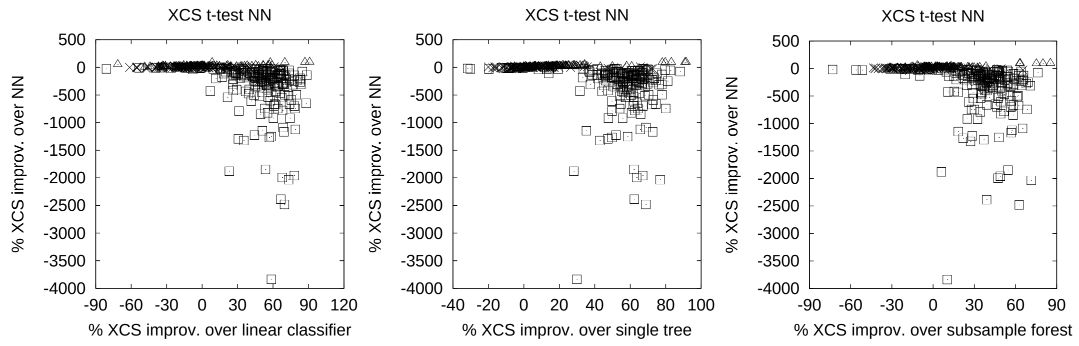


(h)



(i)

Fig. 8. XCS versus nearest neighbor (NN). The notation used in the plots is: △ is used when XCS is significantly better than NN for a given problem, □ when XCS is worse than NN, and × when there is no significant difference between XCS and NN. In the first seven plots (a)-(g) the y-axis represents the percentage of improvement of XCS with respect to NN. The x-axis represents each of the six complexity metrics: (a) and (b) the percentage of points in boundary, (c) the percentage of retained adherence subsets, (d) the ratio of intra-inter class nearest neighbor distances, (e) the non-linearity of the linear classifier, (f) the maximum Fisher's discriminant ratio and (g) the ratio of the number of points to the number of dimensions. Plot (b) is a zoom of plot (a), with more detail into the region where XCS improves NN. Figures (h) and (i) show combination of metrics.



(a) (b) (c)

Fig. 9. XCS versus nearest neighbor (NN), compared to other classifiers. △ denotes that XCS is significantly better than NN, □ denotes that XCS is significantly worse than NN and × denotes no significant difference between XCS and NN. The y-axis represents the percentage of XCS's improvement over NN. The y-axis represents: (a) XCS's improvement over the linear classifier, (b) XCS's improvement over the single tree, and (c) XCS's improvement over the subsample forest.

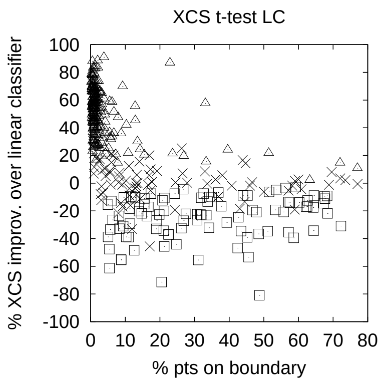
Correlating these observations with the rest of the complexity metrics (figures 8(a) - 8(g)), we find that XCS is significantly worse than NN when the problems have very low percentage of points on boundary, relatively high percentage of retained adherence subsets (figure 8(c)), a fairly low ratio of intra-inter class nearest neighbor distances (figure 8(d)), low non-linearity (figure 8(e)), a varied range of maximum Fisher's discriminant ratios (figure 8(f)), and a fixed and moderately high ratio of the number of points over the number of dimensions (figure 8(g)). What is the interpretation of these metrics? What types of problems are characterized by these observations? The fact that the ratio of intra-inter class distances is relatively small suggests that the points are closer to other points of the same class than to the points belonging to the other class. That is, classes are well separated relative to the class groupings. This observation is confirmed by the small percentage of points lying between the class boundary. The fact that the percentage of adherence subsets is mostly high may mean that the points are spread over a long and thin structure along the class boundary. This might be the case where there are many clusters of points within the same class, but the clusters are close to each other so that the nearest neighbor distances are closer among points of the same class than among the points of opposite classes. That is why the nearest neighbor algorithm approximates the class boundaries so well. Examining which problems present this set of characteristics, we find that most of them belong to the letter recognition dataset. That is why all these problems have the same ratio of the number of points to the number of dimensions. This might indicate that this particular metric does not influence the relation between XCS and NN, and the fact that all problems where XCS is worse than NN have a ratio of 100 is just a coincidence in our particular choice of problems. We also find that the maximum Fisher's discriminant ratio is not a significant metric to distinguish between XCS's performance and NN's performance.

But although XCS is worse than NN in this kind of problems, XCS's approximation is moderately good. In fact, XCS outperforms the rest of the classifiers in almost all these types of problems. Figure 9 demonstrates this for the linear classifier, the single tree and the subsample forest. For example, figure 9(a) plots the percentage of XCS's improvement over NN against XCS's improvement over the linear classifier. Observe that those problems where XCS is worse than NN (plotted by a square) correspond to positive improvements of XCS with respect to the linear classifier. This means that XCS is worse than NN but better than LC. The same is observed in figures 9(b) and 9(c) which compares XCS to the single tree and the subsample forest respectively. And the same happens for the subspace forest, which is not plotted for brevity. We can conclude that NN dominates the rest of classifiers in those problems that present small boundary lengths, low ratios of intra-inter class nearest neighbor distances and small non-linearities. In these problems, XCS places second.

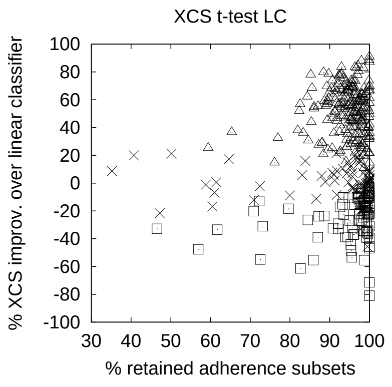
The problems where XCS significantly outperforms the nearest neighbor classifier have a large variety in boundary length and percentages of retained adherence subsets. This means that when the classes tend to be spread, XCS offers better approximations than the nearest neighbor.

We analyze the performance of XCS compared to that of the linear classifier (LC). Figure 10 shows several scatter plots relating the performance of these two classifiers to the geometrical complexity metrics. From these plots, we observe some problems where XCS outperforms LC and others where XCS is worse. Although they are not always clearly separated, we can extract the following observations. XCS is better than the linear classifier in the easiest problems; that is, when there are few points in the class boundary (figure 10(a)), low ratio of intra-inter nearest neighbor distances (figure 10(c)) and low non-linearities (figure 10(d)). On the other hand, XCS is superseded by LC in a wide range of problems with different complexity. Comparing these with the problems where XCS is better, we observe that for problems where XCS is worse than LC the number of points in the boundary tends to be higher, the maximum Fisher's discriminant ratio is generally smaller, while the non-linearities are spread over a wide range of values. The sparsity of the training set also tends to be more severe (less points relative to the number of dimensions) for the problems where XCS is worse. The pretopological measure of the percentage of adherence subsets does not influence the relative performances.

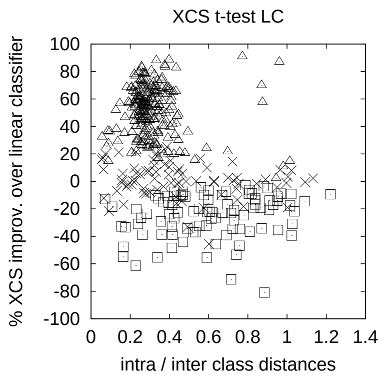
This behavior is surprising because there are a number of problems where a linear classifier approach outperforms XCS. A deeper examination (not reported here) also showed that in most of these problems, LC is the best approach among all the classifiers we have tested. One possible explanation is that for the easiest problems, XCS can find more accurate approximations than a linear approach, which makes a linear and rough approximation of the class boundaries. On the contrary, when the problem presents high complexities, like high numbers of points in the boundary, high dispersion, etc., with a sparse training set, an approximation like that provided by XCS can overfit the complexity of the class geometry, despite the generalization mechanisms that it includes. It may induce class boundaries that are too specific for the training points and may not be general enough to have good generalization accuracy. Then, when XCS



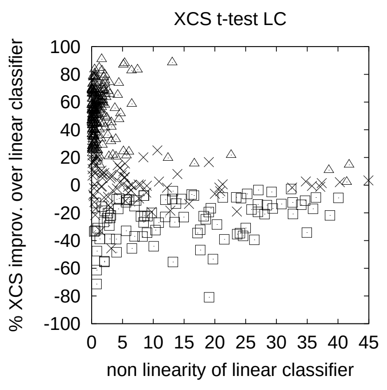
(a)



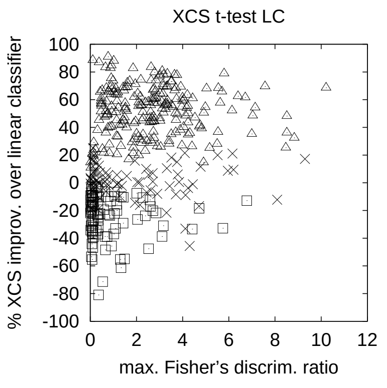
(b)



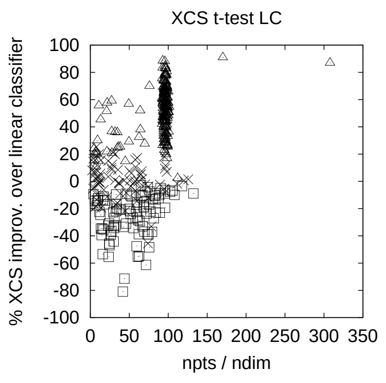
(c)



(d)



(e)



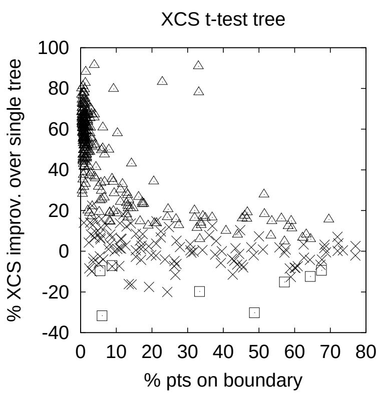
(f)

Fig. 10. XCS versus linear classifier (LC). △ denotes that XCS is significantly better than LC, □ denotes that XCS is significantly worse than LC and × denotes no significant difference between XCS and LC. The y-axis represents the percentage of improvement of XCS with respect to LC. The x-axis represents each of the six complexity metrics: (a) the percentage of points in boundary, (b) the percentage of retained adherence subsets, (c) the ratio of intra-inter class nearest neighbor distances, (d) the non-linearity of the linear classifier, (e) the maximum Fisher's discriminant ratio and (f) the ratio of the number of points to the number of dimensions.

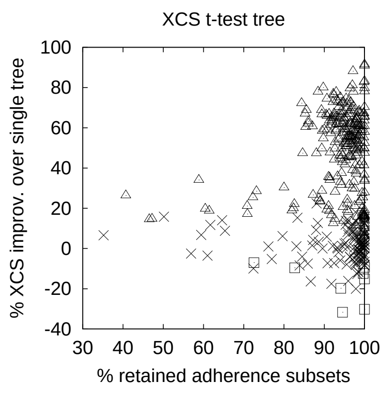
is applied to new unseen data, its generalization ability is worsened. In these cases, a simpler approach like a linear boundary seems to work better. The linear classifier tends to be less sensitive to overfitting and thus generalizes better. We cannot validate this hypothesis because we cannot compute the true sparsity of the current training sets, since we work with real-world classification problems and we do not know the true distribution of the point population. The ratio of the number of points to the number of dimensions is a rough estimate of the sparsity which may fail to reflect the true sparsity of the training set. We could only estimate the real sparsity of a training set if we had control over the real distribution of examples. This could be achieved by using artificial datasets, so we consider the design of artificial datasets highlighting this issue as an important future work.

Figure 11 compares the performance of XCS to that of the single tree classifier, and relates these to the complexity metrics. Note that XCS is worse than the single tree in very few cases (eight problems out of 392). Figure 11(e) shows that these eight problems tend to have very low maximum Fisher's discriminant ratios. Nevertheless, a low Fisher's ratio does not necessarily imply that XCS performs worse than the single tree. There are other problems with low Fisher's ratios where XCS is equivalent or better than the single tree. In most of the problems, XCS is better than the single tree; these problems constitute 69% of the classification problems. The remaining 28% problems are cases where XCS is equivalent to the single tree. We observe that the problems where XCS outperforms the single tree cover a wide range, so it is difficult to relate this behavior to the complexity metrics. The significant improvements of XCS are distributed over different error rates, different percentages of points on boundary, several values of the percentage of retained adherence subsets, etc. In general, we can conclude that XCS is preferable to the single tree, because XCS almost always performs equivalently or better than the single tree, at least the single tree with oblique hyperplanes, over the test cases included in these experiments.

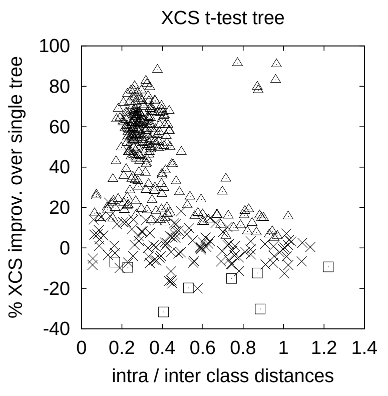
It was found that decision forests can outperform decision trees in a variety of problems [41]. Figure 12(a) shows the percentage of improvement of XCS over the subsample decision forest versus the percentage of improvement of the subsample forest over the single tree. Observe that in most of the problems there are positive improvements of the subsample forest over the single tree. There is a group of problems where XCS outperforms the forest and another group where XCS's performance is worse than the forest. In particular, where the subsample forest offers a higher improvement over the single tree, XCS is able to improve on the subsample forest, and consequently XCS's improvement over the single tree is even higher than the forest. On the contrary, the problems where the forest moderately improves the single tree are those in which XCS is worse. Tracing these two groups of problems along the complexity measures (see figures (12(b)-12(f)), we find that XCS is better than the subsample forest when the percentage of points on boundary and the non-linearities are very small, the ratio of the number of points over the number of dimensions tends to be high, and the maximum Fisher's discriminant ratio also tends to be high. In contrast, XCS is worse than the subsample forest when Fisher's ratio is very small, the training set is sparse (measured as the number of points over the number of dimensions) and



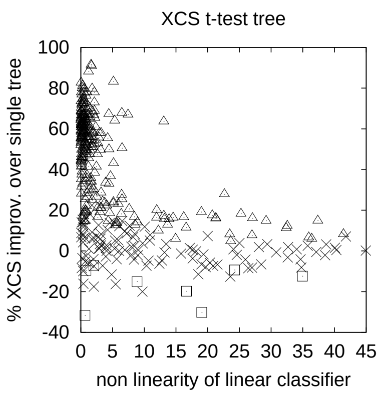
(a)



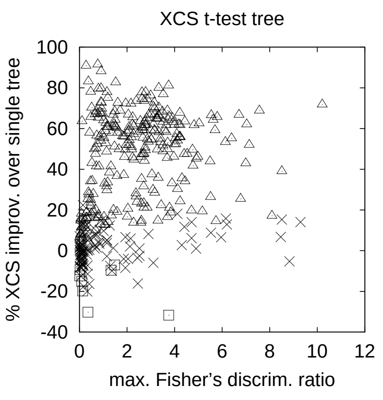
(b)



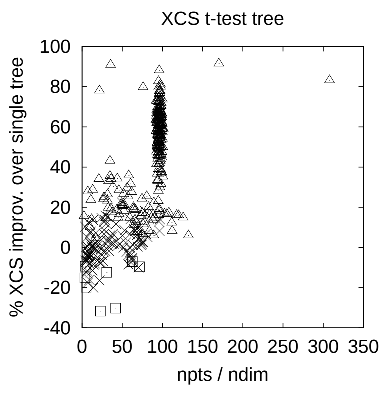
(c)



(d)



(e)



(f)

Fig. 11. XCS versus the single oblique tree. △ denotes that XCS is significantly better than the tree, □ denotes that XCS is significantly worse than the tree and × denotes no significant difference between XCS and the tree. The y-axis represents the percentage of improvement of XCS with respect to the single tree. The x-axis represents each of the six complexity metrics: (a) the percentage of points in boundary, (b) the percentage of retained adherence subsets, (c) the ratio of intra-inter class nearest neighbor distances, (d) the non-linearity of the linear classifier, (e) the maximum Fisher's discriminant ratio and (f) the ratio of the number of points to the number of dimensions.

there is a higher number of points on class boundary.

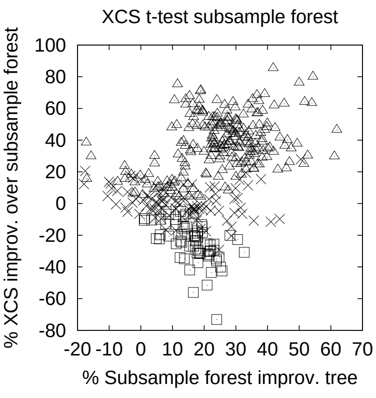
Figure 13 compares the results of XCS to those of the subspace forest. Figure 13(a) shows again that in a majority of problems there is an improvement by the decision forest over the single tree. And for almost all these cases, XCS is equivalent or superior to the subspace decision forest. There are though some problems where XCS is outperformed by the subspace forests (37 out of 392) but these can hardly be related to the complexity metrics. There are different problems with the same complexity values that are better solved with XCS in some cases, and better solved with the subspace forest in other cases, or solved with both classifiers equivalently (see figures 13(b)-13(f)). This may suggest that these metrics are not discriminating enough to distinguish when a problem can be better solved with XCS than with the subspace forest.

The results obtained with the decision forests are consistent with previous experiments in the literature [41], where the subsample forests are shown to excel in those problems with sparse training sets, small maximum Fisher's discriminant ratio and highly non-linear boundaries. On the contrary, subspace forests were found to perform better when the class boundary is smoother (low non-linearities) and for large training sets relative to dimensionality. It seems that subsampling the training set has an effect of overcoming the sparsity of the training set, small Fisher's ratios and high non-linearities, while sampling the feature space tends to work best for less complex boundaries. In this sense, XCS's behavior is more similar to the subspace forest. In fact, we can view XCS as a combination of classifiers, where each classifier contains a rule that may have generalizations in some of the attributes, having an effect similar to that of sampling over the feature space.

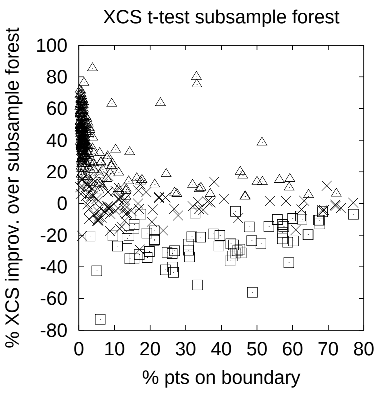
VII. Conclusions

This paper investigates the domain of competence of the XCS classifier system, namely to what kind of problems XCS is well suited and poorly suited, along with a comparison of performance with other classifier schemes. Such investigation is possible because of our characterization of the complexity of real-world classification problems by a set of geometrical descriptors.

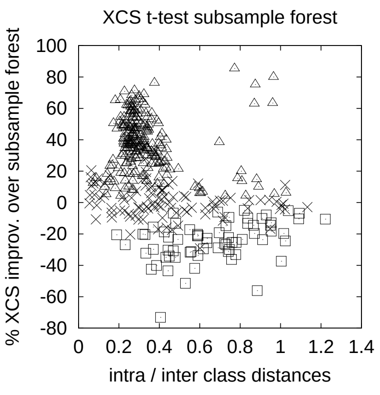
Several factors of problem complexity have an effect on the accuracy of a classifier. The most discriminating features of problems that are difficult for XCS are the percentage of points on the boundary, the ratio of intra-inter class nearest neighbor distances and the non-linearity. XCS's performance presents almost a linear correlation with these metrics. But although a high value on these metrics usually implies high levels of complexity, there are also some exceptions to this rule where associations between



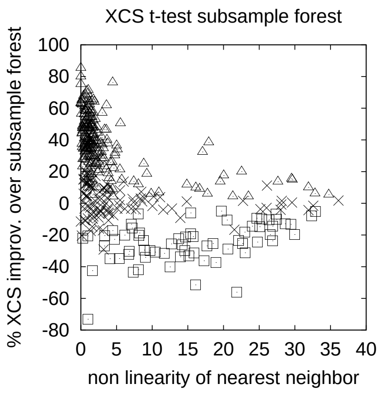
(a)



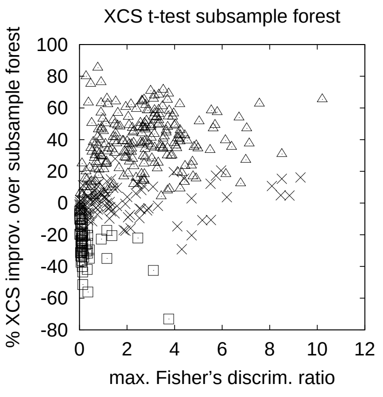
(b)



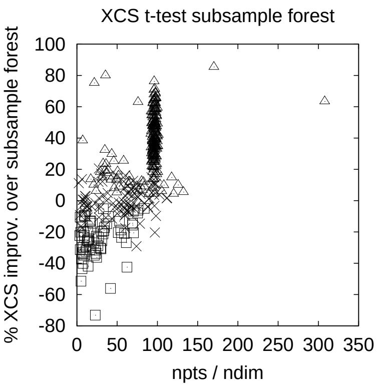
(c)



(d)

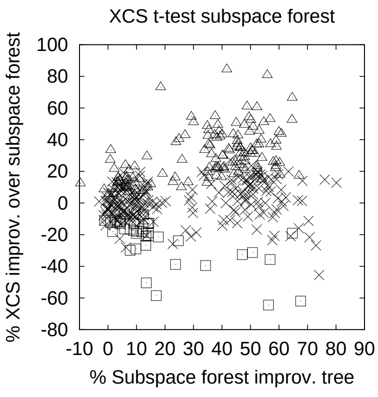


(e)

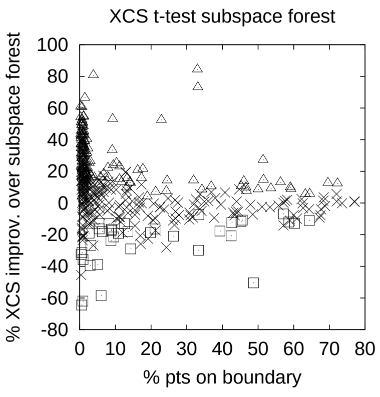


(f)

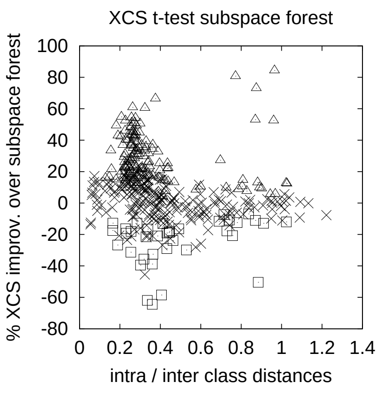
Fig. 12. XCS versus subsample forests. △ denotes that XCS is significantly better than the subsample forest, □ denotes that XCS is significantly worse than the subsample forest and × denotes no significant difference between XCS and the subsample forest. Plot (a) shows the percentage of improvement of XCS over the subsample forest against the percentage of improvement of the subsample forest over the single tree. The rest of the plots show the relation of XCS's improvement over the subsample forest against selected metrics: (b) the percentage of points in boundary, (c) the ratio of intra-inter class nearest neighbor distances, (d) the non-linearity of the nearest neighbor, (e) the maximum Fisher's discriminant ratio and (f) the ratio of the number of points to the number of dimensions.



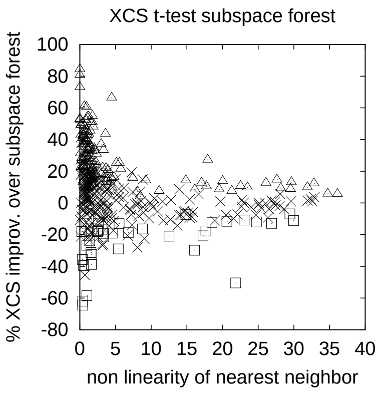
(a)



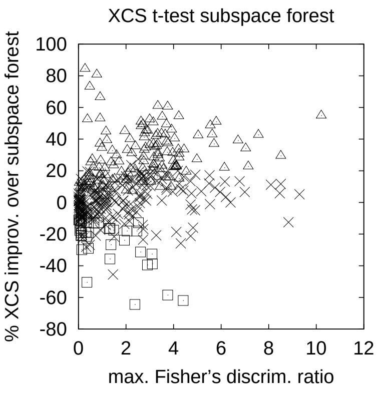
(b)



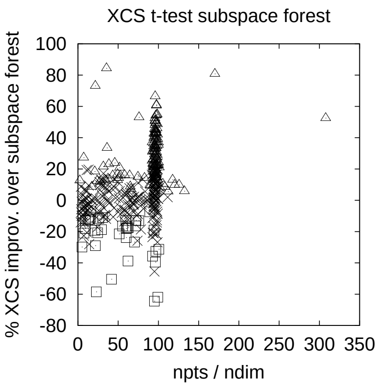
(c)



(d)



(e)



(f)

Fig. 13. XCS versus subspace forests. △ denotes that XCS is significantly better than the subspace forest, □ denotes that XCS is significantly worse than the subspace forest and × denotes no significant difference between XCS and the subspace forest. The first plot shows the percentage of improvement of XCS over the subspace forest against the percentage of improvement of the subspace forest over the single tree. The rest of the plots show the relation of XCS's improvement over the subspace forest against selected metrics: (b) the percentage of points in boundary, (c) the ratio of intra-inter class nearest neighbor distances, (d) the non-linearity of the nearest neighbor, (e) the maximum Fisher's discriminant ratio and (f) the ratio of the number of points to the number of dimensions.

several geometrical metrics are needed to explain XCS's behavior, revealing the multifaceted nature of the complexity of a problem.

Two opposite regions of complexity are identified for XCS in our measurement space. A difficult problem for XCS is determined by a high number of points near the boundary, high percentage of adherence subsets, high dispersion of clusters of classes and high non-linearities. All these cases represent highly interleaved classes and high number of small clusters dispersed in the feature space. These are complex problems in the Kolmogorov sense for XCS, because XCS must evolve a high number of rules covering small regions of the feature space, with few possible generalizations. The easiest problems for XCS are placed in the opposite extreme, representing those cases where classes are more compact.

Identifying difficult problems for XCS and their associated characteristics enhances our current understanding of XCS's behavior in real-world classification problems. This is an important step towards future studies that will analyze possible improvements of XCS in these known difficulties. Also, identifying what features make real-world problems difficult is important to the design of new artificial datasets that allow a finer control of their degree of difficulty and are usually more comprehensible to study the classifier's response. The results shown here are also significant in setting expectations of accuracies for XCS in new classification problems, provided that they are characterized by their geometrical complexity descriptors.

We have also investigated the relative performance of XCS and other classifiers, with reference to the problem complexity. The results show that there is not a clear winner on all the problems. Instead, we should talk about winners on certain domains of the complexity space. Different classifier systems have different domains of competence. The nearest neighbor classifier seems to be the best solution when the problem is characterized by points clustered near other points of the same class and classes are well separated with respect to the class groupings. In these cases, XCS is outperformed by NN, but still offers a fairly accurate solution which can be considered as a supplement to the result of the NN. While NN has a knowledge representation based on the Voronoi cells, XCS represents the solution by a set of rules that can be useful for human readability. A number of problems are better solved with the linear classifier. Some of them are complex problems with a sparse training set. There a linear approach works better because of its resistance to overfitting. XCS outperforms the decision tree in almost all problems, offering similar advantages to the decision forests, which belong to the category of classifier combination methods. Classifier combination is recently considered an effective means for improving the accuracy and generalization ability of individual classifier schemes. In particular, subsample forests work better for sparse training sets, non-linearities and poorly discriminative features. On the contrary, subspace forests work better for larger training sets and smoother boundaries, which is closer to the domain of competence of XCS.

An interesting derivation of this study is to predict what types of classifiers from those tested here are the most appropriate given a particular problem. Using our results, the representation of the problem on the complexity measurement space could suggest what classifier or types of classifiers are best suited to the problem. The process could be done automatically, without much a priori knowledge about the problem domain. This could be considered a kind of meta-learning: we would perform a classification of classification problems, i.e., we would specify classes of problems according to the classifier that best solves it. Therefore, the accuracy of this prediction should also be analyzed in order to see if our conclusions can be extrapolated to new unseen problems.

Some limitations of this study point us to directions of future research. Although the experiments are performed on a set of 392 problems, this might be still small and not sufficiently representative of all possible classification problems. Although the selection is not biased towards any particular type of problem, the conclusions extracted herein are still limited to this selection, and any possible extrapolation must be done with caution. Moreover, the set of problems do not present a uniform distribution on the measurement space. There are some empty regions in the measurement space that make the extraction of conclusions difficult. We still do not know if these regions could be filled with some new problems or if they represent some constraints of geometry and topology. Further search of more problems should be done, along with the design of artificial problems which may fill those gaps in the measurement space and provide a continuum of complexity. Also the conclusions are obtained with particular classifier schemes. XCS's performance depends on the choice of representation, so other choices may lead to different results. The choice of the hyperrectangle codification has been supported by the most part of studies that have used this representation in real-world classification problems. Nevertheless, the study of other types of representations, such as messy [13], [14] or recently proposed representations, like the unordered bound representation [36], may be interesting for investigating to what degree XCS's performance depends on the representation. Similarly, the decision tree uses oblique hyperplanes, and the extrapolation of results may not apply to other types of trees. Other limitations come from the selection of the complexity metrics. There are still some results that can hardly be related to the values of these metrics. Some aspects of problem complexity may still be hidden because our selection of the geometrical metrics cannot fully describe all relevant aspects of the geometry of the problem. An enhancement or refinement of these metrics could lead to more conclusive relationships between the classifier behavior and problem complexity.

The research conducted here is a step further on the continuing effort of improving the understanding of classifier behavior, which is done simultaneously by different researchers on different families of classifier schemes. We encourage the use of this methodology to test the domain of competence of other types of classifiers based on evolutionary algorithms, such as Pittsburgh classifier systems or classifiers based on genetic programming.

ACKNOWLEDGEMENTS

Ester Bernadó thanks Tin Kam Ho and Computer Sciences Research Center, Bell Labs, for inviting her to Bell Labs where this work was mainly developed. She also acknowledges the support of *Enginyeria i*

Arquitectura La Salle during part of this study, as well as the support of *Ministerio de Ciencia y Tecnologa* under grant TIC2002-04036-C05-03 with Feder funds, and *Generalitat de Catalunya* under grant 2002SGR-00/55.

REFERENCES

[1] S. W. Wilson, "Classifier Fitness Based on Accuracy," *Evolutionary Computation*, vol. 3, no. 2, pp. 149–175, 1995.

[2] ——, "Generalization in the XCS Classifier System," in *Genetic Programming: Proceedings of the Third Annual Conference*, J. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. Fogel, M. Garzon, D. Goldberg, H. Iba, and R. Riolo, Eds. San Francisco, CA: Morgan Kaufmann, 1998.

[3] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: The MIT Press/Bradford Books, 1998.

[4] J. H. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

[5] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Inc., 1989.

[6] J. H. Holland and J. S. Reitman, "Cognitive systems based on adaptive algorithms," *Pattern directed inference systems*, pp. 313–329, 1978.

[7] J. H. Holland, "Escaping Brittleness: The Possibilities of General Purpose Learning Algorithms Applied to Parallel Rule-Based Systems," *Machine Learning: An Artificial Intelligence Approach*, vol. II, pp. 593–623, 1986.

[8] M. V. Butz, T. Kovacs, P. L. Lanzi, and S. W. Wilson, "How XCS Evolves Accurate Classifiers," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, L. Spector, E. Goodman, A. Wu, W. Langdon, H. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, Eds. San Francisco, CA: Morgan Kaufmann, 2001, pp. 927–934.

[9] M. V. Butz and M. Pelikan, "Analyzing the Evolutionary Pressures in XCS," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, L. Spector, E. Goodman, A. Wu, W. Langdon, H. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, Eds. San Francisco, CA: Morgan Kaufmann, 2001, pp. 935–942.

[10] T. Kovacs, "Deletion Schemes for Classifier Systems," in *Proceedings of the Genetic and Evolutionary Computation Conference, (GECCO-99)*, W. Banzhaf, J. Daida, A. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, Eds. Morgan Kaufmann, 1999, pp. 329–336.

[11] ——, "Strength or Accuracy? A comparison of two approaches to fitness calculation in learning classifier systems," in *Learning Classifier Systems: From Foundations to Applications*, ser. Lecture Notes in Computer Science, W. Stolzmann, P. Lanzi, and S. Wilson, Eds., vol. 1813. Springer-Verlag Berlin Heidelberg, 2000, pp. 143–160.

[12] S. W. Wilson, "Get Real! XCS with Continuous-Valued Inputs," in *Festschrift in Honor of John H. Holland*, L. Booker, S. Forrest, M. Mitchell, and R. Riolo, Eds. Center for the Study of Complex Systems, University of Michigan, 1999, pp. 111–121.

[13] P. L. Lanzi, "Extending the Representation of Classifier Conditions. Part I: From Binary to Messy Coding," in *Proceedings of the Genetic and Evolutionary Computation Conference, (GECCO-99)*, W. Banzhaf, J. Daida, A. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, Eds. Morgan Kaufmann, 1999, pp. 337–344.

[14] ——, "Extending the Representation of Classifier Conditions. Part II: From Messy Coding to S-Expressions," in *Proceedings of the Genetic and Evolutionary Computation Conference, (GECCO-99)*, W. Banzhaf, J. Daida, A. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, Eds. Morgan Kaufmann, 1999, pp. 345–352.

[15] S. W. Wilson, "Mining Oblique Data with XCS," in *Advances in Learning Classifier Systems: Proceedings of the Third International Workshop*, ser. Lecture Notes in Artificial Intelligence, P. Lanzi, W. Stolzmann, and S. Wilson, Eds., vol. 1996. Springer-Verlag Berlin Heidelberg, 2001, pp. 158–176.

[16] E. Bernadó Mansilla, X. Llorà Fàbrega, and J. M. Garrell Guiu, "XCS and GALE: a Comparative Study of Two Learning Classifier Systems on Data Mining," in *Advances in Learning Classifier Systems, 4th International Workshop*, ser. Lecture Notes in Computer Science, P. Lanzi, W. Stolzmann, and S. Wilson, Eds., vol. 2321. Springer, 2002, pp. 115–132.

[17] C. Brodley, "Recursive automatic bias selection for classifier construction," *Machine Learning, Vol. 20*, pp. 63–94, 1995.

[18] T. Kovacs and M. Kerber, "What makes a problem hard for XCS?" in *Advances in Learning Classifier Systems: Proceedings of the Third International Workshop*, ser. Lecture Notes in Artificial Intelligence, P. Lanzi, W. Stolzmann, and S. Wilson, Eds., vol. 1996. Springer-Verlag Berlin Heidelberg, 2001, pp. 80–99.

[19] M. Butz and S. Wilson, "An algorithmic description of XCS," in *Advances in Learning Classifier Systems: Proceedings of the Third International Workshop*, ser. Lecture Notes in Artificial Intelligence, P. Lanzi, W. Stolzmann, and S. Wilson, Eds., vol. 1996. Springer-Verlag Berlin Heidelberg, 2001, pp. 253–272.

[20] D. E. Goldberg and J. Richardson, "Genetic Algorithms with sharing for multimodal function optimization." in *Proceedings of the Second International Conference on Genetic Algorithms*, J. Grefenstette, Ed. Lawrence Erlbaum, 1987, pp. 41–49.

[21] J. Horn, D. E. Goldberg, and K. Deb, "Implicit Niching in a Learning Classifier System: Nature's Way," *Evolutionary Computation, 2(1)*, pp. 37–66, 1994.

[22] T. Kovacs, "Towards a Theory of Strong Overgeneral Classifiers," in *Foundations of Genetic Algorithms*, vol. 6. Morgan Kaufmann, 2001, pp. 165–184.

[23] S. Saxon and A. Barry, "XCS and the Monk's Problem," in *Learning Classifier Systems: From Foundations to Applications*, ser. Lecture Notes in Computer Science, P. Lanzi, W. Stolzmann, and S. Wilson, Eds., vol. 1813. Springer-Verlag Berlin Heidelberg, 2000, pp. 223–242.

[24] E. Bernadó Mansilla, "Contributions to Genetic Based Classifier Systems," Ph.D. dissertation, Enginyeria i Arquitectura La Salle, Ramon Lull University, 2002.

[25] E. Bernadó Mansilla and J. M. Garrell Guiu, "Accuracy-Based Learning Classifier Systems: Models, Analysis and Applications to Classification Tasks," *Evolutionary Computation*, vol. 11, no. 3, pp. 209–238, 2003.

[26] T. K. Ho and M. Basu, "Measuring the complexity of classification problems," in *15th International Conference on Pattern Recognition*, 2000, pp. 43–47.

[27] M. Li and P. Vitanyi, *An Introduction to Kolmogorov Complexity and Its Applications*. Springer Verlag, 1993.

[28] T. K. Ho and M. Basu, "Complexity Measures of Supervised Classification Problems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 3, pp. 289–300, March, 2002.

[29] R. Prim, "Shortest Connection Networks and Some Generalizations," *Bell System Technical Journal*, vol. 36, pp. 1389–1401, 1957.

[30] J. Friedman and L. Rafsdy, "Multivariate generalizations of the Wald-Wolfowitz and Smirnov two-sample tests," *The Annals of Statistics*, vol. 7, pp. 697–717, 1979.

[31] F. Lebourgeois and H. Emptoz, "Pretopological approach for supervised learning," in *Proceedings of the 13th International Conference on Pattern Recognition*, 1996, pp. 256–260.

[32] A. Hoekstra and R. Duin, "On the nonlinearity of patter classifiers," in *13th International Conference on Pattern Recognition*, 1996, pp. 271–275.

[33] M. Basu and T. K. Ho, "The learning behavior of single neuron classifiers on linearly separable or nonseparable input," in *International Joint Conference on Neural Networks*, 1999.

[34] D. Aha, D. Kibler, and M. Albert, "Instance-based Learning Algorithms," *Machine Learning, Vol. 6*, pp. 37–66, 1991.

[35] C. Blake and C. Merz, " UCI Repository of machine learning databases, [http://www.ics.uci.edu/∼mlearn/MLRepository.html]. University of California, Irvine, Department of Information and Computer Sciences," 1998.

[36] C. Stone and L. Bull, "For Real! XCS with Continuous-Valued Inputs," *Evolutionary Computation*, vol. 11, no. 3, pp. 299–336, 2003.

[37] F. Smith, "Pattern classifier design by linear programming," *IEEE Transactions on Computers*, vol. C-17, pp. 367–372, 1968.

[38] R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*. Brooks/Cole Publishing Company, 2003.

[39] S. Murthy, S. Kasif, and S. Salzberg, "A System for Induction of Oblique Decision Trees," *Journal of Artificial Intelligence Research*, vol. 2, no. 1, pp. 1–32, 1994.

[40] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Transcations on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[41] ——, "A Data Complexity Analysis of Comparative Adavantages of Decision Forest Constructors," *Pattern Analysis and Applications*, vol. 5, pp. 102–112, 2002.

[42] L. Breiman, “Bagging predictors,” *Machine Learning*, vol. 24, pp. 123–140, 1996.

[43] T. G. Dietterich, “Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms,” *Neural Computation*, vol. 10, no. 7, pp. 1895–1924, 1998.