Theoretical XCS Parameter Settings of Learning Accurate Classifiers

Masaya Nakata
Yokohama National University
Tokiwadai 79-5, Hodogaya, Yokohama, Japan
nakata-masaya-tb@ynu.ac.jp

Will Browne
Victoria University of Wellington
Wellington 6140, Wellington, New Zealand
will.browne@ecs.vuw.ac.nz

Tomoki Hamagami
Yokohama National University
Tokiwadai 79-5, Hodogaya, Yokohama, Japan
hamagami@ynu.ac.jp

Keiki Takadama
The University of Electro-Communications
1-5-1 Chofugaoka, Chofu, Tokyo, Japan
keiki@inf.uec.ac.jp

ABSTRACT

XCS is the most popular type of Learning Classifier System, but setting optimum parameter values is more of an art than a science. Early theoretical work required the impractical assumption that classifier parameters had fully converged with infinite update times. The aim of this work is to derive a theoretical condition to mathematically guarantee that XCS identifies maximally accurate classifiers, such that subsequent deletion methods can be used optimally, in as few updates as possible. Consequently, our theory provides a universally usable setup guide for three important parameter settings; the learning rate, the accuracy update and the threshold for subsumption deletion. XCS with our best parameter settings solves the 70-bit multiplexer problem with only 21% of instances that the standard XCS setup needs. On a highly class-imbalanced multiplexer problem with inaccurate classifiers having more than 99.99% classification accuracy, our theory enables XCS to identify only 100% accurate classifiers as accurate and thus obtain the optimal performance.

CCS Concepts

•**Computing methodologies** → **Rule learning;** *Classification and regression trees;*

Keywords

learning classifier system, theory, parameter analysis

1. INTRODUCTION

Learning Classifier Systems (LCSs) [7] are a rule-based evolutionary learning system that aims at producing general classifiers as a solution to a problem. In 1995, Wilson proposed the XCS classifier system [14], which seeks to produce maximally general and maximally accurate classifiers. Since then, XCS has been actively researched on a wide range of machine learning problems as it is an effective evolutionary approach to solve complex and large scale problems. For instance, in 2013, Iqbal showed the first result that an extension of XCS solves the 135-bit multiplexer problem [9]. In 2014, XCS was applied to salient object detection [10] and sequence labeling as complex classification problems.

XCS is a framework controlled by system parameters, which can produce maximally general and maximally accurate classifiers provided that these system parameters are set adequately (here, internally learnt classifier statistics are simply termed 'parameters'). That is, XCS evolves general classifiers, learns associated parameters to identify accurate ones, and then subsumes specific classifiers to similar, general, accurate ones (i.e. subsumption deletion). Hence, evolution and learning aim to generate good candidates of classifier and to *correctly* identify them as accurate or inaccurate[1] ideally in as few generations or classifier updates as possible respectively.

The correctness of identification of accurate classifiers greatly affects the system performance. If an inaccurate classifier is wrongly identified as accurate then it is allowed to subsume its more accurate neighbors (i.e., the problematic *over-generalization* occurs). System performance is reduced as it requires additional iterations to correct this mistake. Hence, the system parameters must be adequately set up to avoid this cause of over-generalization.

However, there is a lack of a theory that guarantees the correct identification of accurate classifiers through reasonable system parameter settings. Early theoretical works, e.g. [3, 4, 6] have provided insight into the workings of LCS in relation to system parameter setting. As related work, Butz mathematically derived equations that calculated the fully converged values of classifier parameters assuming infinite update times [3]. Based on the Butz's equations, Orriols presented a theoretical setting of some system parameters tuned for the class imbalance problem [12]. However, this approach, i.e. to use the fully converged values, may be less reliable in practice due to the impractical assumption of the infinite update times (see Section 3.2 for more detailed discussion). For instance, here is an unsolvable question with

[1]XCS aims to identify the maximum accurate classifiers as accurate, otherwise inaccurate.

GECCO '17, July 15-19, 2017, Berlin, Germany

DOI: http://dx.doi.org/10.1145/3071178.3071200

those works (but will be solved by this paper); can we decide the system parameter settings in order that, in as few classifier updates as possible, XCS identifies 100% accurate classifiers as accurate and 99% accurate ones as inaccurate? This is a lack of a universally accepted set-up guide for LCSs in terms of identifying accurate classifiers. Instead, the parameter settings are normally decided by experience, especially by reference to the standard settings [5]. But XCS may need more iterations in order to handle this process than the minimum iterations it ideally needs. This fundamental inefficiency issue still remains more than 20 years after XCS was proposed.

This paper presents a theory that mathematically guarantees that XCS identifies only actually accurate classifiers as accurate and thus the subsumption deletion is applied only with such classifiers. In other words, our theory derives a limit that prevents problematic over-generalization. Here, a classifier is systematically defined as accurate if its prediction error is less than a threshold called the *standard accuracy* ϵ_0; a prediction error is a classifier parameter and it is updated by the Windrow-Hoff learning rule [13] with the learning rate β. In addition, the subsumption operator functions with reliably accurate classifiers that have been updated more than a threshold θ_{sub}. Based on this XCS mechanism, we will reasonably explain that, if those three system parameters (i.e., ϵ_0, β and θ_{sub}) are set adequately, XCS can prevent over-generalization.

Our theory does not require the assumption that the classifier parameters have been fully converged to their ideal values and it can be a universally usable set-up guide for ϵ_0, β and θ_{sub} not only for XCS but also XCS-based systems that employ the same learning mechanism as XCS. We further present that our theory can derive the best values of those parameters, which enable XCS to identify the classifiers either as accurate or as inaccurate in as few updates as possible. In the experiments, we test our theory on XCS, but also on a version of XCS called XCSSMA, which is an advanced LCS model solving complex overlapping classification problems [8]. This paper does not explain the algorithm of XCSSMA, which can be found in [8]. Our experimental results of XCS are repeatable with our code uploaded in supplementary material[2]. Note that this paper does not discuss the optimality of the standard learning mechanism of XCS, e.g. the Widrow-Hoff learning rule may not be the best approach to update classifier parameters. Also we do not focus on improving the performance in terms of evolution, i.e. we do not discuss how fast XCS can generate accurate classifiers only how fast classifiers can be labelled correctly once evolved. It is assumed that during the evolutionary process accurate classifiers will be created at various times. This has been observed in our experiments (see Section 4).

2. XCS

We describe the XCS mechanism for classification tasks (see [5] and our code in supplementary material for more detail). A classifier is a *condition-action* rule; a condition C is coded by $C \in \{0, 1, \#\}^L$, where L is the length of condition, and the symbol '#' is the *don't care symbol* which matches all input values (*i.e.*, 0 or 1). Classifiers consist of a condition, an action, and four main parameters [5, 14]: (i) the prediction p, which estimates the relative payoff that the system expects when the classifier is used; (ii) the prediction error ε, which estimates the error of the prediction p; (iii) the fitness F, which estimates the accuracy of the payoff prediction given by p; and (iv) the experience n, which is the number of times it has been updated[3].

For the current sensory input s, XCS builds a *match set* $[M]$ containing the classifiers in the population $[P]$ whose condition matches s. If $[M]$ does not contain all the feasible actions, *covering* takes place to create a set of classifiers that matches s and cover all the missing actions. For each possible action a in $[M]$, XCS computes the *system prediction* $P(s, a)$ which estimates the payoff that XCS expects if action a is performed in s. The system prediction $P(s, a)$ is computed by Equation 1 as the fitness weighted average of the predictions of classifiers in $[M]$ that advocate action a, where, $[M](a)$ represents the subset of classifiers of $[M]$ with action a, $cl.p$ and $cl.F$ identify the prediction and fitness of classifier $cl \in [M](a)$, respectively. Then XCS selects an action to perform; the classifiers in $[M]$ that advocate the selected action form the current *action set* $[A]$. The selected action a is performed, and a scalar reward r is returned to XCS. Then XCS receives a new input s.

$$P(s, a) = \sum_{cl \in [M](a)} cl.p \times \frac{cl.F}{\sum_{c \in [M](a)} c.F} \tag{1}$$

Classifiers' parameters are updated in the following order: experience, prediction error, prediction, and finally fitness. Note, these parameters, except for the experience, are normally updated using the *moyenne adaptive modifee* (MAM) procedure that replaces the weighted update with the learning rate β ($0 \leq \beta \leq 1$) with a simple average value at the start of training (i.e. $n < 1/\beta$) [5]. The MAM procedure helps a classifier's parameters to converge faster. Firstly, the experience n is increased by one. Then, the prediction error $cl.\epsilon_n$ and prediction $cl.p_n$ of classifier cl, both for experience n, are updated by the Widrow-Hoff learning rule:

$$cl.\epsilon_n = cl.\epsilon_{n-1} + \beta(|r - cl.p_{n-1}| - \epsilon_{n-1}) \tag{2}$$

$$cl.p_n = cl.p_{n-1} + \beta(r - cl.p_{n-1}) \tag{3}$$

Then, the absolute accuracy $cl.\kappa$ is calculated by Equation 4 dependent on the *standard accuracy* ϵ_0. Then the fitness is updated as usual [5]. Classifiers are defined as accurate classifiers ($cl.\kappa = 1$) if $\epsilon_n < \epsilon_0$. Finally, a Genetic Algorithm (GA) is applied to classifiers in $[A]$. The *subsumption* operator is applied to the classifiers in $[A]$ after updating the classifier parameters and also to offspring after the GA. A classifier can be subsumed by a more general classifier than it, provided that the more general classifier is reliably accurate and sufficiently updated (i.e., $\epsilon_n < \epsilon_0$, $n > \theta_{sub}$).

$$cl.\kappa = \begin{cases} 1 & \text{if } \ cl.\epsilon_n < \epsilon_0, \\ \alpha(cl.\epsilon_n/\epsilon_0)^{-\nu} & \text{otherwise.} \end{cases} \tag{4}$$

3. THEORY

3.1 Preparation

Goal. While XCS defines the accurate classifiers with the condition $\epsilon_n < \epsilon_0$ (see Equation 4), this condition is still unclear to determine the reliably accurate classifiers as it does not specify how many times a prediction error should

[2]Our code is based on Butz's `XCSJava` code [1].

[3]The experience is originally denoted by *exp* in [5].

be updated with the learning rate β to be trustworthy. Instead, the subsumption operator employs the specific condition $\epsilon_n < \epsilon_0, n > \theta_{sub}$ to identify the reliably accurate classifiers. Accordingly, our theory aims to derive a theoretical condition which mathematically guarantees that XCS identifies the reliably accurate classifiers satisfying the condition $\epsilon_n < \epsilon_0, n > \theta_{sub}$, that is, we derive theoretical values of ϵ_0, θ_{sub} and β.

Assumptions. We here use the following two assumptions. Firstly, we consider a classification task with binary reward scheme represented as r_{max}/r_{min}. XCS will receive the maximum reward r_{max} when executing the correct action to an input; otherwise the minimum reward r_{max}. This paper sets r_{max} and r_{min} to 1000 and 0 respectively. Second, we consider that two classifier parameters, prediction p and prediction error ϵ, are updated without the MAM procedure in order to simplify our theory derivation. Hence, in our theory, those parameters are always updated as in Equations 2 and 3 [4]. However, to determine the impact of this assumption, we conduct experiments to compare the performances of XCS with/without MAM update.

Procedure. We firstly derive specific estimation equations for the classifier parameters of prediction and prediction error for any experience n. Second, we determine the system parameter values of ϵ_0, θ_{sub} and β, which satisfy the condition $\epsilon_n < \epsilon_0, exp > \theta_{sub}$. Finally, we find the best settings for those parameters, which enable XCS to identify the accurate classifiers in as few updates as possible.

3.2 Estimation Equation Derivation

Prediction Estimation. Let p_n be the prediction p of classifier cl for experience n, that is, p_n has been updated n times; and $r_n \in r_{max}, r_{min}$ be a reward which is used to update p_n. An initial prediction p_I ($n = 0$) can be any value as it is initialized by the covering operator or in the GA when it is generated[5]. Equation 3 can be rewritten as Equation 5[6]. We can see each reward r_k is weighted by $\beta(1-\beta)^{n-k}$: newer rewards r_k (i.e., k is close to n) are weighed more than older ones.

$$p_n = (1-\beta)^n p_I + \sum_{k=1}^{n} \beta(1-\beta)^{n-k} r_k. \quad (5)$$

Accordingly, the value of p_n depends on *how many* and *when* it received r_{max} or r_{min}. The number of r_{max} (or r_{min}) earned by a classifier depends on the classification accuracy. Let P_C be the *true* classification accuracy of classifier cl. Then we can suppose the number of r_{max} can be nP_C, and thus the number of r_{min} will be $n(1-P_C)$. That is, when a classifier is updated by n times, ideally, it receives nP_C r_{max} rewards, and $n(1-P_C)$ r_{min} rewards. For instance, a classifier with $P_C = 0.8$ would receive 8 r_{max} rewards for $n = 10$, and so 2 r_{min} rewards.

Let $\boldsymbol{R}(P_C) = [r_1, r_2, \cdots, r_n]$ be the matrix of reward that a classifier cl with P_C received, for experience n; again, the number of $r_{max} \in \boldsymbol{R}(P_C)$ is nP_C and the number of $r_{min} \in \boldsymbol{R}(P_C)$ is $n(1-P_C)$. Here, $\boldsymbol{R}(P_C)$ can take different combinations of r_{max} and r_{min}. For instance, with $P_C = 0.8$, n=10 and $r_k \in 1000, 0$, $\boldsymbol{R}(0.8)$ can be $[0,0,1,1,1,1,1,1,1,1]$, $[0,1,0,1,1,1,1,1,1,1]$, and so on, where "1" and "0" denote r_{max} and r_{min} respectively. The different combinations represent that the classifier received the same numbers of r_{min} and r_{min}, but at different time points. Correspondingly, p_n also takes different values although its P_C is the same value, since each reward r_k is weighted by each value of $\beta(1-\beta)^{n-k}$ (see Equation 5). Figure 1 plots possible predictions of all combinations for $\boldsymbol{R}(0.8)$, which are calculated from Equation 5 where $P_C = 0.8, n = 10$, $p_I = 0$, $r_{max} = 1000$, $r_{min} = 0$ and $\beta = 0.2$. As shown by this figure, the predictions depend on when it received r_{max}/r_{min} rewards.

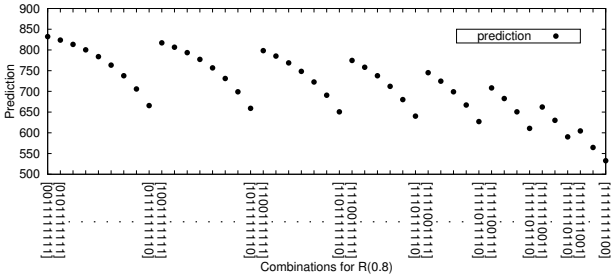


Figure 1: Classifier prediction p_{10} with $\beta = 0.2$, where its true classification accuracy $P_C = 0.8$, its experience $n = 10$, its initial prediction $p_I = 0$ and the $r_{max} = 1000$, $r_{min} = 0$ rewards.

Consequently, it is difficult to derive an estimation equation that estimates all possible predictions since we do not exactly know when a classifier receives the r_{max}/r_{min} rewards. So, we derive the estimation equation based on expected reward that a classifier receives. Let $E_r(P_C)$ be the expected reward that the classifier with P_C received:

$$E_r(P_C) = r_{max}P_C + r_{min}(1-P_C) = 1000P_C \quad (6)$$

Note, this expected value does not consider the learning rate β. In [3], Butz suggested the fully-converged prediction (i.e. p_∞) can be approximated by this expected reward. We complement his approximation with two aspects. Firstly, we claim that the fully-converged prediction p_∞ may not actually converge to $E_r(P_C)$ with an inadequate setting of β. It needs a further condition to take into account this approximation. This is a well known fact, but we will explain this point to aid understanding on our theoretical parameter settings. Secondly, we extend this approximation of the prediction p_∞ to estimate p_n for any experience n.

First, we explain why the system cannot take this approximation of p_∞ with an inadequate setting of β. Figure 2 shows the variances of possible predictions that are calculated from all combinations for $\boldsymbol{R}(0.5)$, where $P_C = 0.5, n = 10$, $r_k \in 1000, 0$, $p_I = E_r(0.5) = 500$ and $\beta = 0.1, 0.2, 0.3$ and 0.4. Note, to investigate how much the prediction is over-estimated from the expected value with different β, we set p_I=$E_r(0.5)$=500. This figure shows, with larger β values, i.e., $\beta = 0.3, 0.4$, the possible predictions are not values near the expected value $E_r(0.5) = 500$. However, with smaller β values, i.e., $\beta = 0.1, 0.2$, the possible predictions have a tendency to be values near $E_r(0.5) = 500$. Accordingly, to take into account the approximation that estimates the predic-

[4]Note that, the action set size which is not relevant to our theory is calculated with the MAM procedure as in [5].

[5]For $n = 0$, the prediction and the prediction error are suitable to be represented as p_0, ϵ_0. However, we denote both as p_I ϵ_I respectively, because ϵ_0 is already defined as the standard accuracy.

[6]We do not show the mathematical proof due to lack of space but it can be proved by mathematical induction.

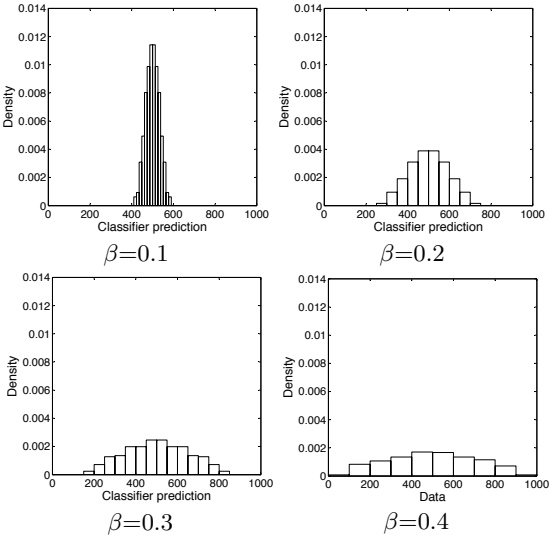


Figure 2: The variances of classifier prediction p_{10}, where its true classification accuracy $P_C = 0.5$ and its experience $n = 10$, its initial prediction $p_I = E_r(0.5) = 500$ and the $1000/0$ rewards.

tion using the expected reward, the learning rate β should be set as small as possible.

We derive an equation that estimates p_n for any experience n. In Equation 5, now we suppose the reward r_k can be replaced with $E_r(P_C)$ for each experience k. Let $\hat{p}_n(P_C)$ be the *approximated* prediction of classifier with P_C for total experience n. From Equations 5 and 6, $\hat{p}_n(P_C)$ can be:

$$\begin{aligned}\hat{p}_n(P_C) &= (1-\beta)^n p_I + \sum_{k=1}^{n} \beta(1-\beta)^{n-k} E_r(P_C), \\ &= 1000P_C - (1-\beta)^n(1000P_C - p_I) \qquad (7)\end{aligned}$$

Here, $\hat{p}_\infty(P_C)$ converges to $1000P_C$, which is the same value as the fully-converged value introduced by Butz [5].

Prediction Error Estimation. Next, we estimate the prediction error of a classifier. Let ϵ_n be the prediction error of classifier cl after experience n. Again, an initial prediction error ϵ_I $(n = 0)$ can take any value as its initial value is set in the covering operator or in the GA when it is generated. In the same manner as we derived Equation 5 from Equation 3, Equation 2 can be rewritten as

$$\epsilon_n = (1-\beta)^n \epsilon_I + \sum_{k=1}^{n} \beta(1-\beta)^{n-k} |r_k - p_{k-1}|. \qquad (8)$$

Note, $p_0(k = 1)$ is the initial prediction p_I; and as shown by Equation 2, ϵ_k is updated with the previous prediction p_{k-1}. The absolute error $|r_k - p_{k-1}|$ also depends on when the classifier receives r_{max}/r_{min} rewards. Accordingly we calculate the expected value of absolute error. Let $E_{\epsilon,n}(P_C)$ be the expected absolute error, for experience n, that is $|r_n - p_{n-1}| \simeq E_{\epsilon,n}(P_C)$. $E_{\epsilon,n}(P_C)$ can be written as

$$\begin{aligned}E_{\epsilon,n}(P_C) &= |r_{max} - p_{n-1}|P_C + |r_{min} - p_{n-1}|(1-P_C) \\ &\simeq |1000 - \hat{p}_{n-1}(P_C)|P_C + |0 - \hat{p}_{n-1}(P_C)|(1-P_C) \\ &= 2000(P_C - {P_C}^2) \\ &\quad - (1-\beta)^{n-1}(1-2P_C)\,(1000P_C - p_I)\,. \qquad (9)\end{aligned}$$

Note that, $0 \le \hat{p}_{n-1}(P_C) \le 1000$. We estimate the *approximated* prediction error from Equations 8 and 9. Let $\hat{\epsilon}_n(P_C)$ be the approximated prediction error of classifier with P_C for experience n. $\hat{\epsilon}_n(P_C)$ can be written as Equation 10; $\hat{\epsilon}_\infty(P_C)$ converges to $2000(P_C - {P_C}^2)$, which is the same fully-converged prediction error introduced by Butz [5].

$$\begin{aligned}\hat{\epsilon}_n(P_C) &= (1-\beta)^n \epsilon_I + \sum_{k=1}^{n} \beta(1-\beta)^{n-k} E_{\epsilon,k}(P_C) \\ &= (1-\beta)^n \epsilon_I + 2000(P_C - {P_C}^2)\,[1-(1-\beta)^n] \\ &\quad - n\beta(1-\beta)^{n-1}(1-2P_C)\,(1000P_C - p_I)\,. \qquad (10)\end{aligned}$$

In summary, we derived two estimation equations that calculate the prediction and the prediction error, both are valid for any experience n. Again, β should be set as small as possible. Next we discuss how we can determine the setting of β and the standard accuracy ϵ_0 based on these equations.

3.3 System Parameter Derivation

We theoretically determine the system parameter values of ϵ_0, θ_{sub} and β, which satisfy the condition $\epsilon_n < \epsilon_0, n > \theta_{sub}$. We firstly give two specific conditions derived from the condition $\epsilon_n < \epsilon_0, n > \theta_{sub}$;

- From the condition $\epsilon_n < \epsilon_0$, for both an accurate classifier cl^* and an inaccurate classifier cl' which the system wants to identify them as accurate and inaccurate respectively, the accurate classifier should have a prediction error $cl^*.\epsilon_{n^*} < \epsilon_0$; while the inaccurate classifier cl' should have $cl'.\epsilon_{n'} \ge \epsilon_0$, i.e., $cl^*.\epsilon_{n^*} < cl'.\epsilon_{n'}$ should be satisfied;
- Additionally, from the condition $n > \theta_{sub}$, the above condition $cl^*.\epsilon_{n^*} < cl'.\epsilon_{n'}$ should be satisfied for any experience n^*, $n' > \theta_{sub}$; n^*, n' are experience of accurate/inaccurate classifiers respectively and can be different values.

Let ${P_C}^*$ and ${P_C}'$ be the true classification accuracy of cl^* that XCS wants to identify as *accurate*, and that of cl' that XCS wants to identify as *inaccurate*, respectively. XCS normally attempts to find maximally accurate classifiers with the 100% classification accuracy, and so we set ${P_C}^* = 1.0$ and $0.5 \le {P_C}' < {P_C}^*$ (see Appendix for detailed configuration of ${P_C}'$). From the above discussions, the condition $\epsilon_n < \epsilon_0, n > \theta_{sub}$ can be rewritten as;

$$\hat{\epsilon}_{n^*}({P_C}^*) < \hat{\epsilon}_{n'}({P_C}'), \qquad (11)$$

where, $\hat{\epsilon}_{n^*}({P_C}^*)$ and $\hat{\epsilon}_{n'}({P_C}')$ are the approximated prediction errors of cl^* and cl', and n^*, n' are the experience of cl^* and cl', respectively. Note that n^*, $n' > \theta_{sub}$.

Equation 11 includes classifier parameters, i.e., ${p_I}^*$, ${\epsilon_I}^*$, n^* for cl^* and ${p_I}'$, ${\epsilon_I}'$, n' for cl', as well as system parameters β (see Equation 10) and θ_{sub}, which are now treated as variables. The approximated prediction errors $\hat{\epsilon}_{n^*}({P_C}^*)$ and $\hat{\epsilon}_{n'}({P_C}')$ can take different values depending on these variables. Accordingly Equation 11 needs to be satisfied for any values of these variables. Thus, we consider the *worst* variable values given that classifiers are updated the maximum times to satisfy Equation 11; that is, the worst values maximize $\hat{\epsilon}_{n^*}({P_C}^*)$ whilst they minimize $\hat{\epsilon}_{n'}({P_C}')$. Then, the worst values of variables in Equation 11 except for β, θ_{sub} can be set to constant values by the following conditions;

$$\arg \max_{{p_I}^*, {\epsilon_I}^*, n^*} \hat{\epsilon}_{n^*}({P_C}^*), \quad \arg \min_{{p_I}', {\epsilon_I}', n'} \hat{\epsilon}_{n'}({P_C}'). \qquad (12)$$

When any constant values of $P_C{}'$ and θ_{sub} are given, we can firstly decide ${p_I}^* = 0$, ${\epsilon_I}^* = 1000$, ${p_I}' = 1000$ and ${\epsilon_I}' = 0$[7]. With these values, $\hat{\epsilon}_{n^*}({P_C}^*)$ and $\hat{\epsilon}_{n'}(P_C{}')$ can be a monotonic decreasing function for n^* and a monotonic increasing function for n' respectively, i.e., at experience $n^*{=}n'{=}\theta_{sub}+1$, $\hat{\epsilon}_{n^*}({P_C}^*)$ is the maximum value while $\hat{\epsilon}_{n'}(P_C{}')$ is the minimum value. Thus we can determine $n^* = n' = \theta_{sub}+1$ as their worst values.

Again, to satisfy Equation 11 for any variables' values, except for β, we only need to prove Equation 11 under the above worst values. Consequently, the learning rate β can be a variable that decides whether Equation 11 is satisfied under the worst values of the other variables. There may be different ways to decide β, but we here take a simple approach. First, we calculate a value of β^* that satisfies a boundary equation at the experience $n^* = n' = \theta_{sub}$: $\hat{\epsilon}_{\theta_{sub}}({P_C}^*) = \hat{\epsilon}_{\theta_{sub}}(P_C{}')$ with the worst values. This boundary equation cannot be solved for β, but we can calculate an approximate β^* by numerical computation methods, such as Newton's method. Again, since $\hat{\epsilon}_{n^*}({P_C}^*)$ is the monotonic decreasing function for experience n^*, for $n^* = \theta_{sub}+1$, $\hat{\epsilon}_{\theta_{sub}+1}({P_C}^*)$ is smaller than $\hat{\epsilon}_{\theta_{sub}}({P_C}^*)$. In contrast, $\hat{\epsilon}_{\theta_{sub}+1}(P_C{}')$ is larger than $\hat{\epsilon}_{\theta_{sub}}(P_C{}')$. Therefore, Equation 11 is always satisfied for any experience n^*, $n' > \theta_{sub}$; and we can decide $\beta = \beta^*$ and $\epsilon_0 = \hat{\epsilon}_{\theta_{sub}}({P_C}^*) = \hat{\epsilon}_{\theta_{sub}}(P_C{}')$ with β^*. Note that, $\hat{\epsilon}_{n^*}({P_C}^*)$ and $\hat{\epsilon}_{n'}(P_C{}')$ both with the worst values can also be a monotonic decreasing or increasing function for β respectively. Hence, Equation 11 with the worst values can also be satisfied for any $\beta \geq \beta^*$. However, as discussed in Section 3.2, β should be set as small as possible to take into account our approximation of expected values. Thus, β should be set to β^* as the possible minimum value.

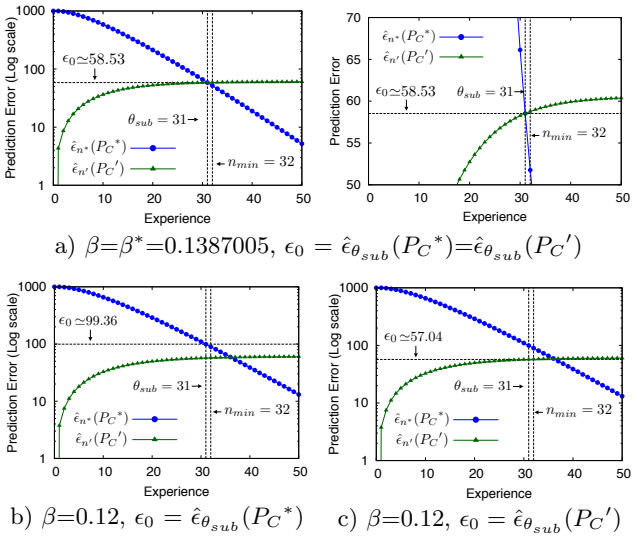
Let us organize the above procedure to derive the parameter settings of ϵ_0, θ_{sub} and β that guarantees XCS correctly identifies only the classifiers having the maximum classification accuracy as accurate. First, we decide a value of θ_{sub}. Next, given $P_C{}'$ (and ${P_C}^* = 1.0$), calculating β^* from the boundary equation $\hat{\epsilon}_{\theta_{sub}}({P_C}^*) = \hat{\epsilon}_{\theta_{sub}}(P_C{}')$ with the worst values of other variables. Then, we obtain $\beta = \beta^*$ and $\epsilon_0 = \hat{\epsilon}_{\theta_{sub}}({P_C}^*) = \hat{\epsilon}_{\theta_{sub}}(P_C{}')$ from Equation 10 with β^*.

3.4 Best Parameter Setting Derivation

Equation 11, decided by the presented procedure, is ideally satisfied for any θ_{sub}. But θ_{sub} should be set to be equal to or more than the minimum value that XCS ideally needs to identify the accurate classifiers. In the other words, when θ_{sub} is set to the minimum value, we can derive the best parameter settings of ϵ_0, θ_{sub} and β which enable XCS to correctly identify the accurate classifiers in as few updates as possible. Here we decide the minimum value of θ_{sub}.

To identify a classifier either as accurate or as inaccurate, inaccurate classifiers should be updated via new instances until they receive at least one incorrect reward r_{min}. The inaccurate classifiers having $P_C{}'$ ideally receive $n(1-P_C{}')$ incorrect rewards (see Section 3.2). Thus the minimum experience n_{min} where inaccurate classifiers receive at least one incorrect reward can be; $n_{min} = 1/(1-P_C{}')$. This means all classifiers that have been updated n_{min} times can be identified as accurate or inaccurate. Thus, the minimum value of θ_{sub} can be set to $n_{min}-1 = 1/(1-P_C{}')-1$. Then, our theory suggests the best settings of β and ϵ_0, using the procedure explained in Section 3.3, with the minimum θ_{sub}. Our minimum value is an ideal number and it is not considered probabilistically. This issue is out of scope in this paper, but θ_{sub} can be set to the probable minimum value and then our theory can derive the best parameter settings.

[7]Note, $0 \leq p_I, \epsilon_I \leq 1000$; $n^*, n' \in \mathbb{N}$.



a) $\beta{=}\beta^*{=}0.1387005$, $\epsilon_0 = \hat{\epsilon}_{\theta_{sub}}({P_C}^*){=}\hat{\epsilon}_{\theta_{sub}}(P_C{}')$

b) $\beta{=}0.12$, $\epsilon_0 = \hat{\epsilon}_{\theta_{sub}}({P_C}^*)$ c) $\beta{=}0.12$, $\epsilon_0 = \hat{\epsilon}_{\theta_{sub}}(P_C{}')$

Figure 3: The curves of $\hat{\epsilon}_{n^*}({P_C}^*)$ and $\hat{\epsilon}_{n'}(P_C{}')$ with different values of β for experience n, where ${P_C}^* = 1$, $P_C{}' = 0.96875$, $\theta_{sub} = 31$, ${p_I}^* = {\epsilon_I}'{=}0$ and ${\epsilon_I}^*{=}{p_I}'{=}1000$.

3.5 Example

Here we show an example of how to set β, ϵ_0 and θ_{sub} using our theory on the multiplexer problem [14]. The multiplexer (MUX) [14] is defined over a binary string of $k+2^k$ bits; the first k bits represent an address pointing to the remaining 2^k bits. For instance, with $k=2$, the input string 110001 will return 1 as an *answer* (i.e., class), while when applied to 110110 it will return 0. When the system performs the correct action, it receives a 1000 reward, otherwise it receives 0. In the 20-MUX ($k=4$), there are optimal classifiers (${P_C}^* = 1$); for instance $1111\#\cdots\#1{:}1$. The inaccurate classifiers that have the highest true classification accuracy is[8], for instance, $\#\#\#\#\#1\cdots1{:}1$; 31 of all 32 instances that this classifier matches, return correct rewards, and so $P_C{}'$ is 31/32=0.96875, and thus $\theta_{sub} = n_{min}-1 = 31$ where $n_{min}{=}32$. We can get $\beta^* \simeq 0.1387005$ by Newton's method.

Accordingly, we can determine $\beta = \beta^* \simeq 0.1387005$ and then we can get $\epsilon_0 = \hat{\epsilon}_{n^*}({P_C}^*) = \hat{\epsilon}_{n'}(P_C{}') \simeq 58.53$. Figure 3 a) shows the values of $\hat{\epsilon}_{n^*}({P_C}^*)$ and $\hat{\epsilon}_{n'}(P_C{}')$ for experience n with $\beta = 0.1387005$ (the left side) and its expanded figure at the right side. As we noted, both $\hat{\epsilon}_{n^*}({P_C}^*)$ and $\hat{\epsilon}_{n'}(P_C{}')$ monotonically decrease or increase respectively; and that $\hat{\epsilon}_{\theta_{sub}}({P_C}^*)$ is equal to $\hat{\epsilon}_{\theta_{sub}}(P_C{}')$ at experience $\theta_{sub} = 31$.

[8]A problem includes inaccurate classifiers with different true classification accuracies $0.5 \leq P_C{}' < {P_C}^*$, but we can consider only the inaccurate classifiers with the highest true classification accuracy. Since not-highest true classification accuracies result in larger prediction errors than that of the highest true classification accuracy, Equation 11 is always satisfied for any $0.5 \leq P_C{}' < {P_C}^*$.

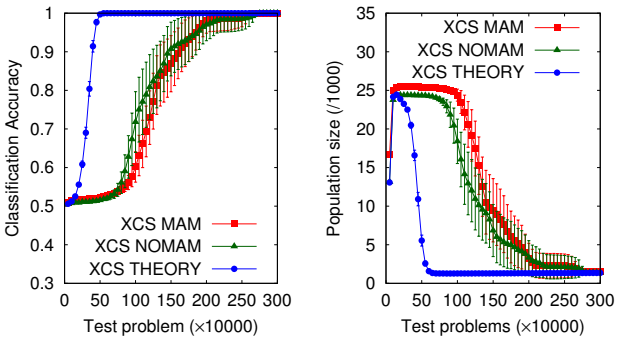


Figure 4: Average Performances (left) and population size (right) on the 70-bit multiplexer problem.

At experience 32 where the accurate classifiers can be identified with the minimum experience $n_{min} = 32$ according to our theory, $\hat{\epsilon}_{n^*}(P_C{}^*)$ is smaller than ϵ_0 while $\hat{\epsilon}_{n'}(P_C{}')$ is larger than ϵ_0. When n is further increased, $\hat{\epsilon}_n(P_C{}^*)$ and $\hat{\epsilon}_{n'}(P_C{}')$ are also smaller or larger than ϵ_0 respectively. Thus inaccurate classifiers having $P_C{}'$, which have been updated more than 31 times, are never identified as accurate and so subsumption does not use them.

Next, we explain why $\beta < \beta^*$ is inadequate against our best parameter settings. We here set $\beta = 0.12 < \beta^*$. Since $\hat{\epsilon}_{\theta_{sub}}(P_C{}^*)$ is not equal to $\hat{\epsilon}_{\theta_{sub}}(P_C{}')$ with the inadequate β, we consider two cases; ϵ_0 is set to $\hat{\epsilon}_{\theta_{sub}}(P_C{}^*)$ and to $\hat{\epsilon}_{\theta_{sub}}(P_C{}')$. Then, we can get $\epsilon_0 = \hat{\epsilon}_{\theta_{sub}}(P_C{}^*) = 99.36$ and $\epsilon_0 = \hat{\epsilon}_{\theta_{sub}}(P_C{}') = 57.04$. For $\epsilon_0 = \hat{\epsilon}_{\theta_{sub}}(P_C{}^*) = 99.36$, as shown in Figure 3 b), at experience $n_{min} = 32$, both $\hat{\epsilon}_{n^*}(P_C{}^*)$ and $\hat{\epsilon}_{n'}(P_C{}')$ are smaller than ϵ_0. Thus, both the accurate classifiers and the inaccurate classifiers will be identified as accurate. Thus, subsmption wrongly performs as a classifier can be subsumed to the inaccurate classifiers, and so overgeneralization can occur. For $\epsilon_0 = \hat{\epsilon}_{\theta_{sub}}(P_C{}') = 57.04$, as shown in Figure 3 c), at experience $n_{min} = 32$, both accurate and inaccurate classifiers would not be identified as accurate since both $\hat{\epsilon}_{n^*}(P_C{}^*)$ and $\hat{\epsilon}_{n'}(P_C{}')$ are larger than ϵ_0; at experience 36, $\hat{\epsilon}_{n^*}(P_C{}^*)$ is smaller than ϵ_0 and $\hat{\epsilon}_{n'}(P_C{}')$ is larger than ϵ_0. Accordingly XCS can identify the accurate classifiers, but it requires 36 update times larger than the minimum experience $n_{min} = 32$; XCS cannot identify them in as few updates as possible.

4. EXPERIMENT

We compare the performances of XCS with our theory and with the standard parameter settings on two classification problems. The first problem is the multiplexer problem [14], which is a binary-class classification problem. The second problem is a class-imbalanced multiplexer problem [12], which has a more complex classification boundary than the standard multiplexer problem due to class-imbalance. We also test our theory on XCSSMA, to determine if it extends to an XCS-based system, with a complex overlapping problem (i.e. the majority-on problem).

4.1 Experimental Setting

This paper uses the following experimental paradigm: during learning problems, the system selects actions randomly from those represented in $[M]$, and during test problems, the system selects the action with highest expected return. Learning problems and test problems alternate. When the system performs the correct action, it receives $r_{max} = 1000$, otherwise it receives $r_{min} = 0$. The genetic algorithm and the update of classifier parameters are enabled only during learning problems, and those are turned off during test problems. We compare the three versions of XCS: 1) *XCS MAM*, which uses the MAM update and the standard parameter settings; 2) *XCS NOMAM*, which does not use the MAM update, but does use the standard parameter settings; 3) *XCS THEORY*, which does not use the MAM update, but uses our theoretical parameter settings. We use the classification accuracy as an evaluation criterion, which is the rate of correct actions the system executed during the test problems. All the plots are averages over 30 experiments.

4.2 Results

70-bit multiplexer problem. This paper uses *70-MUX* as it has a large search space of 2^{70} inputs. Although a recent work showed an extension of XCS [9] solves the *135-MUX* with 2^{135} inputs, *70-MUX* is still a useful large space problem with a feasible running time.

We use the following standard parameters [9]; N=30,000; β=0.2, $\epsilon_0 = 10$, $\theta_{sub} = 20$, $\alpha = 0.1$, $\delta = 0.1$, $\nu = 5$, $\theta_{GA} = 25$, $\theta_{del} = 20$, $\chi = 0.8$, $\mu = 0.01$, $P_\# = 1.0$, $fitness\ reduction = 0.1$, The action set subsumption and the GA subsumption are turned on, the uniform crossover and the tournament selection with the tournament size $\tau = 0.4$ are used. For the theoretical parameter settings, β, ϵ_0 and θ_{sub} are determined as follows; in the *70-MUX*, $P_C{}^*$ is 1 and $P_C{}'$ is 0.9921875; the inaccurate classifiers with the highest true classification accuracy is, for instance, ####$\cdots$###1$\cdots$1:1, which matches 128 (=2^7) inputs including one incorrect input 00000001$\cdots$1 with the correct class 0 and other inputs have the correct class 1, and thus, $P_C{}' = 127/128 = 0.9921875$. Thus, θ_{sub} is set to 127 as the minimum experience and we can calculate $\beta = \beta^* \simeq 0.04739$ and $\epsilon_0 = \hat{\epsilon}_{\theta_{sub}}(P_C{}^*) = \hat{\epsilon}_{\theta_{sub}}(P_C{}') = 15.36777$.

Figure 4 shows the classification accuracy of the three versions of XCS on the *70-MUX*. In addition, we here compare the population size which is a number of classifiers in the population. The plots are reported as moving average of 50,000 test problems with 95% confidence interval. *XCS MAM* and *XCS NOMAM* averagely reach 100% classification accuracy after 2,700,000 test problems, but both XCSs unstably performs with a wide confidence interval. *XCS THEORY* significantly outperforms the other versions of XCS. *XCS THEORY* reaches optimal and stable performance after 550,000 with a very narrow confidence interval. Consequently, *XCS THEORY* evolves a small population size faster than other versions, which indicates that XCS accurately generalizes classifiers.

Class-Imbalanced Multiplexer problem. The l-bit class-imbalanced multiplexer problem or *l-IMUX* [12] generates a binary string of $l = k + 2^k$ bits as an instance. If the answer of its instance, which is computed as in the normal *l-MUX*, belongs to the *minority class*, then its instance is sent to XCS with probability $1/2^i$, where i is the imbalance level. If the answer belongs to the *majority class*, its instance is accepted as the input at all times. Accepted inputs are sent to the XCS, and we set the minority class to 0 and the majority class to 1. Note that, during test problems, the instances for both the majority class and the minority class are sent to

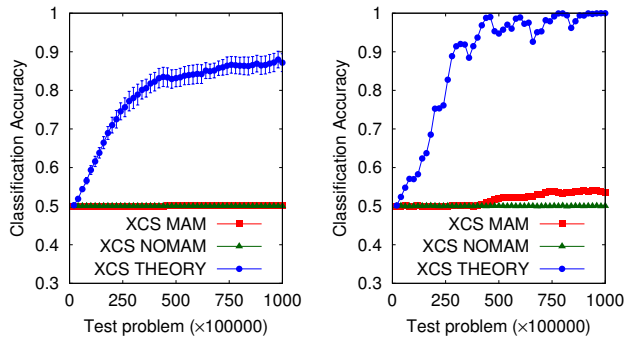


Figure 5: Average Performances (left) and Best performance (right) on the 11-bit class-imbalanced multiplexer problem with $ir = 10$.

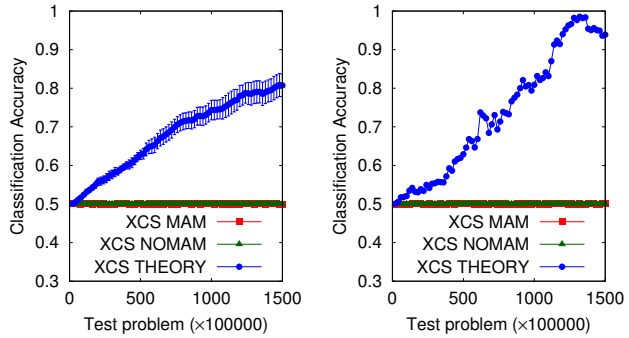


Figure 6: Average Performances (left) and Best performance (right) on the 11-bit class-imbalanced multiplexer problem with $ir = 11$.

XCS with the same probability as in *l-MUX*. A recent work showed an extended XCS derives a sub-optimal performance on the *11-IMUX* with $i = 9$ [11]. We here consider a more complex problem *11-IMUX* with $i = 10, 11$.

For the standard parameter settings, we use the specialized settings for class-imbalance [12]; $N = 800$, $\epsilon_0 = 1$, $\theta_{sub} = 200$, $P_\# = 0.6$, $\mu = 0.04$, $\beta = \{0.0025, 0.00125\}$, and $\theta_{GA} = \{3200, 6400\}$, both for $i = \{10, 11\}$ respectively, action set subsumption is turned off, but GA subsumption is turned on. Two-point crossover and roulette-wheel selection are used; for other parameters we use the same settings in the previous experiment of *70-MUX*. In the *IMUX*, an adequate evolutionary pressure is required in order to solve these problems. Hence, θ_{GA} is set to a high value so that all niches will receive approximately the same opportunities during evolution [12]. For the theoretical parameter settings, β, ϵ_0 and θ_{sub} are determined as follows; in the standard *11-MUX*, $P_C{}^*$ is 1 and $P_C{}'$ is 0.9375 [9], but due to the bias of the class-imbalance, $P_C{}'$ depends on the imbalance level i for the *11-IMUX*. The modified $P_{C_i}{}'$ for the imbalance level i can be calculated as

$$P_{C_i}{}' = \frac{P_C{}' \times (1 - 2^{-i})}{P_C{}' \times (1 - 2^{-i}) + (1 - P_C{}') \times 2^{-i}}, \qquad (13)$$

and thus, we can obtain $P_{C_{10}}{}' = 0.999934836$, $P_{C_{11}}{}' = 0.999967433$. Since XCS is required to distinguish accurate classifiers (100% accurate) from inaccurate ones (about 99.99 % accurate), *11-IMUX* with $i = 10, 11$ is a very difficult problem. We set $\theta_{sub} = \{15345, 30705\}$, $\beta = \beta^* \simeq \{0.0007472, 0.000398\}$, $\epsilon_0 = \hat{\epsilon}_{\theta_{sub}}(P_C{}^*) = \hat{\epsilon}_{\theta_{sub}}(P_C{}') = \{0.13\ 0206, 0.06\ 5016\}$, for $i = \{10, 11\}$ respectively.

Figure 5 shows the classification accuracy on the *11-IMUX* with $i = 10$. The figure on the left side shows the average performances of the three versions of XCS tested here; the figure on the right side shows the best performance the system based on one run of the 30 runs. *XCS-MAM* and *XCS-NOMAM* completely fail to solve the problem, while *XCS-THEORY* reaches 86% average classification accuracy but does not reach 100% on average. We can suppose this is because an evolutionary pressure still fails to evolve classifiers evenly for all niches; XCS evolves classifiers focusing on the majority class, and so the accurate classifiers for the minority class cannot be evolved successfully. However, the results still suggests that our theory enables XCS to identify the maximally accurate classifiers. This is highlighted in the graph of best classification accuracy. The best classification accuracy of *XCS-THEORY* eventually reaches 100% classification accuracy. Similar to the result on *11-IMUX* $ir = 10$, as shown in Figure 6 with $ir = 11$, *XCS-THEORY* robustly performs well with near 100% classification accuracy. This indicates that XCS with our theoretical parameter setting can potentially solve *11-IMUX* $ir = 10, 11$. The performance could improve with enhanced evolutionary pressure.

Majority on Problem. We test our theory on a derived version of XCS (i.e., XCSSMA). In the majority-on problem (MOP), if the number of ones is greater than the number of zeros, the problem instance is of class one, otherwise class zero. In majority-on problem domain, the complete solution consists of strongly overlapping classifiers, so is therefore difficult to learn [8]. Iqbal reported XCSSMA completely solves the 7-bit MOP, but XCS fails to reach 100% performance [8]. We test the three versions of XCSSMA on 11-bit MOP. We use the following parameter settings from [8]; $P_\# = 0.5$, $\chi = 0.8$, number of states $= 5$, input alphabet $\Sigma = \{0, 1\}$, output alphabet $\Delta = \{0, 1\}$; for other parameters we use the same settings as in the multiplexer problems. Note N should be set to a large value to support all niches in problems containing the overlapping classifiers [2], and so we set $N = 6000$. In the 11-bit MOP, the $P_C{}^*$ is 1 and $P_C{}'$ is 0.984375, and then we set $\beta = 0.081998$, $\epsilon_0 = \hat{\epsilon}_{\theta_{sub}}(P_C{}^*) = \hat{\epsilon}_{\theta_{sub}}(P_C{}') = 30.233279$ and $\theta_{sub} = 63$ for XCSSMA with our theory [10].

Figure 7 shows the performances on the 11-bit MOP. The figure on the left side shows the average performances of three versions of XCS as usual; the figure on the right side shows the best performance the system based on one run of the 30 runs. Average performance of all LCSs does not reach to 100% performance due to the more complex overlapping classifiers than the 7-bit MOP. This is because, as we discussed in the class-imbalanced problem, an evolutionary pressure needs to be designed to evolve classifiers to evenly support all niches, since overlapping classifiers cause imbalance of niches sizes [2]. However, XCSSMA with our

[9]The inaccurate classifiers with the highest true classification accuracy is, for instance, $\#\#\#\#1\cdots1{:}1$. Accordingly, $P_C{}' = 15/16 = 0.9375$.

[10]The inaccurate classifiers that have the highest true classification accuracy is, for instance, $\#\#\#\#\#\#1\cdots1{:}1$. Accordingly, $P_C{}' = 63/64 = 0.984375$.

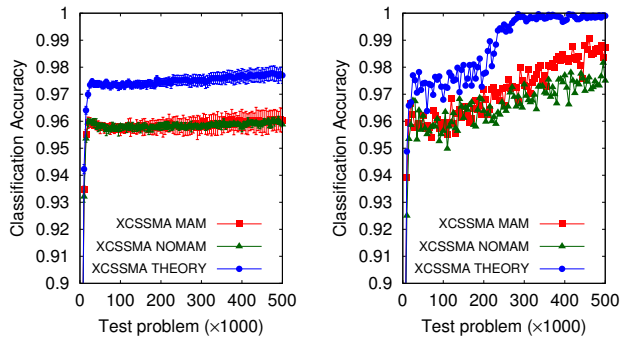


Figure 7: Average Performances (left) and Best performance (right) on the 11-bit majority-on problem.

theory outperforms other LCSs. Additionally, the best performance of XCSSMA with our theory reaches to near 100% performance. This suggests that the parameter settings suggested by our theory can potentially solve the 11-bit MOP. An adequate evolutionary pressure is required to completely solve problems containing overlapping classifiers in order to produce a stable performance.

5. CONCLUSION

We presented a theory for XCS that mathematically guarantees that the XCS classifier system identifies maximally accurate classifiers, both correctly and quickly without the need for infinity-based assumptions. The main assumption made here is knowing, or being able to estimate, the performance of the best inaccurate classifier. A further benefit is the ability to estimate the future approximate value of classifier parameters for any given experience by considering the effect of the learning rate. The best parameter settings of three system parameters (the learning rate, the standard accuracy and the threshold for subsumption) can now be determined, which enables XCS to identify the maximally accurate classifiers in as few updates as possible. These settings improve the performance of XCS and derivative versions of XCS (e.g. XCSSMA). This determines a bound by which to judge future XCS improvements. We assumed classifiers are updated without the MAM update, so investigating the effects of MAM would be useful for further analysis of classifier parameters and the learning rate.

APPENDIX

XCS evolves two types of accurate classifiers; the *positive* accurate classifier cl_p^* with its true classification accuracy $P_{C_p}^*$ (normally it is 1); the *negative* accurate classifier cl_n^* with $P_{C_n}^*$ (normally it is 0). Correspondingly, we can suppose the two type of inaccurate classifiers; the positive ones cl_p' and the negative ones cl_n'. Let P_{C_p}' and P_{C_n}' be the true classification accuracy of cl_p' and cl_n' respectively. If $P_{C_\cdot}'$ is a value near to $P_{C_\cdot}^*$, that is, $|P_{C_\cdot}^* - P_{C_\cdot}'|$ is a small value, it would be difficult for XCS to handle accurate generalization[11]. Accordingly, the difficulties of generalization for the positive and negative accurate classifiers can be different. To prevent over-generalization we need to consider the more difficult case. That is, if $|P_{C_p}^* - P_{C_p}'| \leq |P_{C_n}^* - P_{C_n}'|$, XCS needs to distinguish cl_p^* with $P_{C_p}^*$ from cl_p' with P_{C_p}'; otherwise cl_n^* with $P_{C_n}^*$ from cl_n' with P_{C_n}'. For the later case (i.e., $|P_{C_p}^* - P_{C_p}'| > |P_{C_n}^* - P_{C_n}'|$), since XCS symmetrically identifies positive and negative classifiers as accurate with the same standard accuracy, we can rephrase that cl_p^* with $1 - P_{C_n}^*$ should be distinguished from cl_p' with $1 - P_{C_n}'$. This means, to prevent over generalization for all (i.e., positive and negative) possible classifiers, we can consider only the positive case. Thus, with $P_{C_p}^* = 1.0$ and $P_{C_n}^* = 0.0$, we can also consider a boundary of $P_{C_p}' \geq 0.5$.

In the benchmark classification problems employed in this paper (i.e., the multiplexer problem, the class-imbalanced multiplexer problem and the majority-on problem), $P_{C_p}^*$=1.0 while $P_{C_n}^*$=0.0; P_{C_p}' is equal to $1 - P_{C_n}'$ for any positive inaccurate classifiers having P_{C_p}' and negative ones having P_{C_n}'. Then, we can always determine $P_{C^*} = 1.0$ and $P_{C'} = P_{C_p}'$.

[11]Here, "." can be p or n.

A. REFERENCES

[1] M. V. Butz. XCSJava 1.0: An Implementation of the XCS classifier system in Java. Tec. report, 2000.

[2] M. V. Butz. *Rule-Based Evolutionary Online Learning Systems*. Springer, 2006.

[3] M. V. Butz, D. E. Goldberg, and K. Tharakunnel. Analysis and Improvement of Fitness Exploitation in XCS: Bounding Models, Tournament Selection, and Bilateral Accuracy. *Evolutionary Computation*, 11(3):239–277, 2003.

[4] M. V. Butz, T. Kovacs, P. L. Lanzi, and S. W. Wilson. Toward a Theory of Generalization and Learning in XCS. *IEEE Trans. on Evo. Comp.*, 8(1):28–46, 2004.

[5] M. V. Butz and S. W. Wilson. An Algorithmic Description of XCS. *Soft Computing*, 6(3–4):144–153, 2002.

[6] J. Drugowitsch. Design and analysis of learning classifier systems. *Sringer, Berlin*, 2008.

[7] J. H. Holland. Escaping Brittleness: The Possibilities of General Purpose Learning Algorithms Applied to Parallel Rule-based system. *Machine Learning*, 2:593–623, 1986.

[8] M. Iqbal, W. Browne, and M. Zhang. Extending learning classifier system with cyclic graphs for scalability on complex, large-scale boolean problems. pages 1045–1052, New York, NY, USA, 2013. ACM.

[9] M. Iqbal, W. Browne, and M. Zhang. Reusing Building Blocks of Extracted Knowledge to Solve Complex, Large-Scale Boolean Problems. *Evolutionary Computation, IEEE Transactions on*, 18(4):465–480, 2014.

[10] M. Iqbal, S. S. Naqvi, W. Browne, C. Hollitt, and M. Zhang. Salient Object Detection Using Learning Classifiersystems That Compute Action Mappings. In *GECCO2014*, pages 525–532. ACM, 2014.

[11] T. Nguyen, S. Foitong, P. Srinil, and O. Pinngern. Towards Adapting XCS for Imbalance Problems. In *PRICAI 2008*, volume 5351 of *LNCS*, pages 1028–1033. Springer, 2008.

[12] A. Orriols-Puig and E. Bernadó-Mansilla. Bounding XCS's parameters for unbalanced datasets. In *GECCO2006*, pages 1561–1568. ACM, 2006.

[13] B. Widrow and M. E. Hoff. Adaptive Switching Circuits. *1960 IRE Western Electric Show and Convention Record 4*, pages 96–104, 1960.

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