Acquiring Classifiers for Bipolarized Reward by XCS
in a Continuous Reward Environment

Takato TATSUMI * and Keiki TAKADAMA *

Abstract: In data mining, it is important to clarify how effective the acquired rules are and which elements are affected by rule evaluation. Extended learning classifier system (XCS) reveals factors that affect the classifier (rule) evaluation by generalizing the multiple classifiers that acquire the same reward (evaluation value) into a generalized classifier. In a real-world problem, because the reward of the classifier varies, XCS cannot acquire the generalized classifier. As useful classifiers with a narrow range of the acquired rewards are required, this paper proposes a new XCS (XCS based on reward bipolarization: XCS-RB) that acquires the classifiers that acquire only high rewards and classifiers that acquire only low rewards. XCS-RB was applied to the problems such as predicting the ratio of the deep sleep time of the night of the day from the care plan implemented in the care house and predicting the calculation time of a matrix-matrix product using SGEMM GPU kernel. XCS-RB acquired rules indicating care plan that leads to deep sleep and parameter settings with short calculation time. XCS-RB was able to acquire the generalized classifiers so as not to conflict with the input data; in this paper, the potential advantages of XCS-RB have been demonstrated.

Key Words: learning classifier system, accuracy criteria, reward.

1. Introduction

The rules acquired by data mining are required to be versatile and useful. The versatile rule is a rule that requires fewer conditions for its effectiveness. Even if the rule is very useful, if the rule is not versatile, the situation applied is limited and the value of the rule is reduced. Because rules are required for each situation, the number of rules acquired becomes huge. As the rule set is not much different from the data itself, it is difficult for humans to understand the rule set at once. To make use of the acquired rule, the versatility of the rule is necessary. A useful rule is easy-to-use for formulating policies. The rules with high evaluation are adopted positively, and the rules with low evaluation are avoided. The rules that receive neither high nor low evaluation are harder to reflect in formulating policies than the previous rules. To make use of the acquired rule, the usefulness of the rule is necessary.

In this paper, learning classifier systems (LCSs) [1] which acquire rules that satisfy both versatility and usefulness constitute the area of focus. The rules acquired by an LCS are called classifiers and are represented by the if-then rule. Notably, rule generalization is a way to clarify important elements. Rule generalization can be used to ignore the data elements that do not affect the evaluation of the if-then rule and retain the data elements that affect the evaluation. LCSs find the optimal combination to generalize rules using a genetic algorithm (GA) [2] and evaluate the generalized rules by reinforcement learning (RL) [3]. The generalized classifier is highly interpretable for humans, compared with other machine learning methods such as support vector machine and neural networks [4]. Among LCSs, XCS (accuracy-based LCS) [5] is the current mainstream classifier system. XCS generalizes classifiers from the accurate classifiers by a generalization mechanism called subsumption. In XCS, the classifier that always obtains the consistent reward is defined as an accurate classifier.

In the real-world problems, however, it is difficult to acquire the generalized classifiers using XCS because even when the input situation is the same, the acquired reward is not necessarily the same. As an example of the problem dealt with in this paper, the probability of the ratio of the deep sleeping time being the same is very low, just by implementing the same care plan (there is no data to show that the ratio of the deep sleeping time becomes the same actually). In such a situation, it is very hard to generalize classifiers based on XCS by subsuming the classifiers because many classifiers are not accurate due to the unstable rewards. This indicates that it is also hard to reduce the number of low generality classifiers by using the subsumption mechanism in XCS. It is required to acquire some useful knowledge (classifiers) from such bad data. To tackle this type of environment, Tatsumi et al. proposed XCS based on variance of reward (XCS-VR) [6], XCS based on mean of reward (XCS-MR) [7], and XCS based on range of reward (XCS-RR) [8] that dynamically determine the accuracy criterion from the sample standard deviation, minimum value, and maximum value of the obtained rewards. The methods of Tatsumi et al. are not suitable for analyzing real-world problems because XCS-VR assumes the range of the acquired reward of the classifier or XCS-MR and XCS-RR assume binary classification problems. As the learning strategy of XCS-VR is the complete action map, XCS-VR acquires the classifiers that acquire a reward near the average value of the whole acquired reward. As mentioned above, because the variability of the reward cannot be avoided, the classifiers acquire both good and bad rewards. In situations where failure is not permitted, the classifiers have no value. Conversely, the classifiers that acquire only high rewards or only low rewards have value.
To obtain the classifiers that have value, in this paper a new XCS, XCS-RB (XCS based on reward bipolarization) that calculates the accuracy of the classifiers based on the range of the acquired reward is proposed. In this paper, XCS-RB sets the range by using the quartile of the acquired reward. Each classifier records the maximum and minimum values of the acquired reward and judges whether it falls within the range defined by XCS-RB using these values. The classifiers that satisfy the acquired reward within the range are judged as accurate.

The rest of this paper is organized as follows. In Section 2, other XCSs that can handle the environments with uncertainty in input, output, or reward are introduced. In Section 3, the mechanism of XCS (without action) is explained. The mechanism of XCS-RB is described in Section 4. In Sections 5 and 6, the problems are presented. Experiments are conducted in Section 7. In Section 8, the experimental results are discussed. Finally, our conclusion is given in Section 9.

2. Related Work

In various preceding work, attempts have been made to improve the performance of XCS in several uncertain environments.

The first uncertainty is related to an input from an environment. An example includes a partially observable Markov decision process (POMDP) environment. In POMDP, an environmental input is uncertain and missing parts of the information are required to distinguish states instantly. To tackle this type of environment, Lanzi et al. [9] and Webb et al. [10] proposed an XCS with memory to solve the POMDP maze problems. However, it is necessary to know in advance how many previous internal states are needed.

The second uncertainty is related to an output to an environment. An example includes the stochastic environment where an output (i.e., action) probabilistically replaces another output. Such a stochastic environment is difficult for XCS because a state sometimes transits to a different state predicted by XCS. To tackle this type of environment, Lanzi et al. proposed XCSμ for the stochastic environment [11]. XCSμ subtracts the prediction error of all classifiers by the minimum value of prediction error among the classifier set. This way, XCSμ eliminates uncertainty and makes it possible to identify accurate classifiers. Also, XCSμ does not assume that in an environment the degree of uncertainty is different for each state.

The final uncertainty is related to a reward from an environment. An example includes the environment where a reward varies such as evaluations by a human in Interactive Evolutionary Computation [12]. In such an environment, the consistent reward is not guaranteed to be obtained, even if the same action is performed as the output for the same input. Owing to this feature, all classifiers are determined as being inaccurate, which prevents XCS from generalizing the classifiers. To tackle this type of environment, Tatsumi et al. proposed XCS-MR [13] that dynamically determines the accuracy criterion from the sample standard deviation of the obtained rewards.

3. Accuracy Based Learning Classifier System (XCS) without Action

3.1 Overview

XCS is composed of 1) a performance component, 2) a reinforcement component, and 3) a rule-discovery component, as shown in Fig. 1. These components evolve a set of classifiers in the population [P].

3.2 Classifier and Its Generalization

The classifier in XCS is composed of the condition part C, the action part A, the prediction p, the prediction error ϵ which is the difference between the prediction and the reward P, the fitness F, and the numerosity n. An LCS acquires knowledge by evolving classifiers to fit multiple environmental conditions. When the condition part is represented by a bit string with a fixed length of 0 or 1, XCS generalizes the classifiers by using the symbol # representing “don’t-care.” For example, “10###” represents eight types of bit string.

3.3 Mechanism

3.3.1 Performance component

XCS selects the classifiers that match the current environmental condition. The classifiers in which each element of the condition part is of the same value as the corresponding current environmental condition or is a # symbol are judged as matching the current input. The classifiers in which their condition part matches the input in [P] are stored in the Match Set [M]. XCS acquires a reward as an output for the input in the problems without action. After receiving the reward, the reinforcement component is executed, and the evolution component is executed after a certain time.

3.3.2 Reinforcement component

The classifiers in [M] have their parameters updated in this component as follows: (i) the prediction p is updated from the obtained reward P as follows:

$$ cl.p \leftarrow cl.p + \beta(P - cl.p) $$

(1)

The variable β is the learning rate and influences the learning speed. (ii) The error ϵ that is the difference between P and cl.p is updated as follows:

$$ cl.\epsilon \leftarrow cl.\epsilon + \beta(|P - cl.p| - cl.\epsilon) $$

(2)

However, when the number of evaluations of the classifier (cl.exp) is less than 1/β, Eqs. (1) and (2) are replaced with Eqs. (3) and (4), respectively:

$$ cl.p \leftarrow cl.p + (P - cl.p)/cl.exp $$

(3)

$$ cl.\epsilon \leftarrow cl.\epsilon + (|P - cl.p| - cl.\epsilon)/cl.exp $$

(4)

These operations increase the learning speed at the beginning of learning. (iii) The fitness F is calculated on the basis of accuracy $\kappa$ as follows:
\[
k(cl) = \begin{cases} 
1 & \text{if } \epsilon < \epsilon_0, \\
\alpha \left( \frac{\epsilon}{\epsilon_0} \right)^{-\nu} & \text{otherwise}.
\end{cases}
\] (5)

In this equation, \(\epsilon_0 (\epsilon_0 > 0)\) is a constant variable that indicates the accuracy criterion. When \(\epsilon\) is smaller than \(\epsilon_0\), the classifier is accurate. The variable \(\alpha (0 \leq \alpha \leq 1)\) and \(\nu (\nu > 0)\) control the reduction rate of the accuracy. The relative accuracy of the classifier \(\kappa'\) is then calculated:

\[
\kappa' = \frac{k(cl) \times cn}{\sum_{x \in M} k(x) \times x.n}.
\] (6)

In this equation, the parameter \(n\) indicates the numerosity of the classifier. (iv) The fitness \(F\) is updated as follows:

\[
cl.F \leftarrow cl.F + \beta (\kappa' - cl.F).
\] (7)

3.3.3 Rule-discovery component

A genetic algorithm evolves the classifiers in \([P]\). The algorithm in this component is summarized as follows: (i) Two parent individuals are selected according to the ratio of their fitness as a selection probability. (ii) Two child individuals are generated by crossing these parent individuals. (iii) The elements of the condition part of these child individuals are mutated with probability \(\mu\). (iv) When the number of the classifiers in \([P]\) exceeds the parameter \(N\), the low fitness classifiers are deleted preferentially.

XCS has a subsumption mechanism to integrate the classifiers with a low generality into more generalized (more \# in the condition part) classifiers. The classifiers whose experience (the number of times obtained rewards) exceeds \(\theta_{sub}\) and which are judged to be accurate \((k(cl) = 1)\) can subsume classifiers having their condition part included in the condition part of the classifier. The numerosity of the subsumed classifier is added to the classifier that subsumes.

4. XCS Based on Reward Bipolarization (XCS-RB)

4.1 Architecture

4.1.1 Classifier

The classifier of XCS-RB has the condition part \(C\), the action part \(A\), the prediction \(p\), the prediction error \(\epsilon\), the fitness \(F\), and the numerosity \(n\) that are the same as the classifier of XCS. Unlike XCS, \(\epsilon\) is calculated from the maximum and minimum values of the acquired reward. Each classifier of XCS-RB has a parameter accuracy criterion \(\epsilon_0\). In addition, each classifier records the maximum acquired reward \(cl.Max\) and the minimum acquired reward \(cl.Min\). These parameters are obtained from the reward acquired by each classifier and are not overall one.

4.1.2 Recording of all rewards acquired by XCS-RB

The bipolarization threshold is determined by using the reward acquired by XCS-RB as a whole. XCS-RB records all acquired rewards to obtain the threshold. In this paper, the aim is to use XCS-RB to acquire the classifiers whose rewards to acquire are from the maximum value \(All.Max\), the minimum value \(All.Min\), the upper threshold \(U\), and the lower threshold \(L\).

4.2 Mechanism

Most aspects of the learning mechanism of XCS-RB are the same as the mechanism of XCS. The main differences from XCS are (a) the parameter updating in the reinforcement component and (b) the subsumption condition in the rule-discovery component. In Fig. 2, the learning mechanism of XCS-RB is shown with the different parts that are blackened. In this section, the different parts are described in detail.

4.2.1 Reinforcement component

The difference from XCS in the reinforcement component is the setting of the parameters \(\epsilon\) and \(\epsilon_0\). These parameters are calculated from \(All.Max\), \(All.Min\), \(U\), and \(L\) calculated from all the rewards acquired by XCS-RB as follows.

The parameters \(\epsilon\) and \(\epsilon_0\) are determined by the magnitude relation between the distance between the maximum value \((cl.Max)\) of the reward acquired by the classifier \(cl\) and the maximum value of all acquired reward \((All.Max)\) and the distance between the minimum value \((cl.Min)\) of the reward acquired by the classifier \(cl\) and the minimum value of all acquired reward \((All.Min)\). It is expressed by Eqs. (8) and (9):

\[
cl.\epsilon \leftarrow \begin{cases} 
cl.Max - All.Min & \text{if } cl.Max - All.Min < All.Max - cl.Min, \\
All.Max - cl.Min & \text{otherwise},
\end{cases}
\] (8)

\[
cl.\epsilon_0 \leftarrow \begin{cases} 
L - All.Min & \text{if } cl.Max - All.Min < All.Max - cl.Min, \\
All.Max - U & \text{otherwise}.
\end{cases}
\] (9)

It is possible to judge whether all the rewards acquired by the classifier \(cl\) are within one of the polarized ranges from these equations.

In Fig. 3, the setting of the parameters \(\epsilon\) and \(\epsilon_0\) (Eqs. (8) and (9)) are shown. Classifier A acquires the rewards in a range which is between a value near the minimum value of all acquired reward \((All.Min)\) and a value near the lower threshold \((L)\). That is, the upper side of Eqs. (8) and (9) are applied. The
prediction error $cl_A, \epsilon$ is $cl_A, \text{Max} - \text{All.Min}$. The accuracy criterion $cl_A, \epsilon$ is $L = \text{All.Min}$. The maximum value of the reward acquired by classifier A is less than the lower threshold ($L$). All rewards acquired by classifier A are within one of the polarized ranges. As $cl_A, \epsilon < cl_A, \epsilon_0$, the accuracy of the accurate classifier A can be appropriately evaluated. Classifier B acquires the rewards in the range which is between a value near the upper threshold ($U$) and a value near the maximum value of all acquired reward ($\text{All.Max}$). That is, the lower side of Eqs. (8) and (9) are applied. The prediction error $cl_B, \epsilon$ is $\text{All.Max} - cl_B, \text{Min}$. The accuracy criterion $cl_B, \epsilon$ is $U = \text{All.Min}$. The minimum value of the reward acquired by classifier B is less than the upper threshold ($U$). A part of the rewards acquired by classifier B are without the polarized ranges. As $cl_B, \epsilon > cl_B, \epsilon_0$, the accuracy of the inaccurate classifier B can be appropriately evaluated.

Last, XCS-RB updates the fitness $F$ of the classifier executing Eqs. (5), (6), and (7) the same as in the case of XCS.

4.2.2 Subsumption condition

Because the classifiers with high generality are matched against many inputs, the variable $\theta_{\text{sub}}$ is used to determine the accuracy of the classifiers. When $\theta_{\text{sub}}$ is not large enough, the over generalized classifiers are erroneously determined to be accurate and erroneous subsumption occurs.

For this issue, XCS-RB provides the subsumption condition, as in XCS based on range of reward (XCS-RR) proposed by Tatsumi et al. [8] In this additional condition, the experience of the classifier that subsumes others should be greater than the experience of any candidate of the subsumed classifiers. As the number of evaluations of the classifier that subsumes others is generally large, the possibility that the classifier is accurate increases.

- $cl, \text{exp} > \theta_{\text{sub}}$
- $cl, \epsilon < cl, \epsilon_0$
- $cl, \text{exp} > \sup(c, \text{exp}, c \text{ could be subsumed by } cl)$

where, $c \text{ could be subsumed by } cl$ is the classifier $c$ belonging to $[M]$ in that iteration and to be subsumed by classifier $cl$.

5. Care Plan-Sleep Problem Description

In this paper, a problem was designed to predict the ratio of the deep sleep time that is the night of the day from behavior record using these data in the nursing home. It is important to reveal behaviors leading to deep sleep because taking a sleep with long deep sleep time helps to improve the quality of life. If the behavior subset leading to deep sleep is revealed, that subset becomes a reference to generate a better care plan. Deep sleep comprises Stages 3 and 4 of non-rapid eye movement sleep. Because the greater the proportion of time of the deep sleep, the higher the quality of sleep, the ratio of the time of the deep sleep is taken as the evaluation value. XCSs receive behavior record as input and the rate of time of the deep sleep as the reward. The subject was 82 years old in 2010. The period of observation was from September 2010 to March 2012. However, because the interruption period was included during that period, the data was acquired for 323 days. The sleep stage and the ratio of the time of the deep sleep are estimated using a bed sensor mat. For details, please refer to our paper [14] using the same data.

### Table 1 Behavior record.

| Behavior       | Value                        |
|----------------|------------------------------|
| Wake up time   | early(0), average(1), late(2), none(3) |
| Time in bed    | early(0), average(1), late(2), none(3) |
| Tea time       | none(0), A.M.(1), P.M.(2), A.M. + P.M.(3) |
| Gardening      | none(0), A.M.(1), P.M.(2), A.M. + P.M.(3) |
| Bath time      | none(0), A.M.(1), P.M.(2), A.M. + P.M.(3) |
| Snack time     | none(0), A.M.(1), P.M.(2), A.M. + P.M.(3) |
| Newspaper reading | none(0), A.M.(1), P.M.(2), A.M. + P.M.(3) |
| Rehabilitation | none(0), A.M.(1), P.M.(2), A.M. + P.M.(3) |

### Table 2 GPU record.

| Parameters | Value |
|------------|-------|
| MWG, NWG   | 16, 32, 64, 128 |
| KWG        | 16, 32 |
| MDIMC, NDMI | 8, 16, 32 |
| MDIMA, NDMB | 8, 16, 32 |
| KWI        | 2, 8 |
| VWM, VWN   | 1, 2, 4, 8 |
| STRM, STRN | 0, 1 |
| SA, SB     | 0, 1 |

In Table 1, the behavior records are shown. A.M. + P.M. indicates that the behavior was taken both in the morning and in the afternoon. Each item is given to XCS with the numbers in parentheses.

6. SGEMM GPU Kernel Performance Problem

This is a problem to estimate the calculation time from the parameter values using SGEMM GPU kernel performance Data Set [15] in UCI Machine Learning Repository. The parameters of this problem and their values are shown in Table 2. Each data records four calculation time, but the averaged value is adopted as the reward of the data. As the averaged values were sorted in ascending order, since they increased exponentially, the base 10 logarithm was taken for the averaged values.

7. Experiments

There is no absolutely correct answer like the testbed problem in the problem described in Section 5. Because this study aims to acquire the classifiers that acquire only one of the extremely large or extremely small rewards, this paper defines the first quartile ($Q_{1/4}$) and third quartile ($Q_{3/4}$) as the lower threshold $L$ and the upper threshold $U$, respectively.

XCS has been expanded to adopt to high dimensional input and regression problems, among other things. There are no extended XCS families aimed at acquiring classifiers similar to XCS-RB. This paper adopts XCS as the conventional method used for comparison.

7.1 Evaluation Criterion

Since the purpose of this paper is to extract the useful knowledge, it does not make sense to verify the learning and verification data separately. In this paper, the aim is to acquire the classifiers that acquire only high and low rewards.

The evaluation criterion of the care plan-sleep problem is whether the classifier in which $\epsilon = 1$ and $\text{exp} > 20$ covers 24 types of inputs shown in Tables 3 and 4. The number of instance columns of Tables 3 and 4 show how many instances of its inputs are in 323 inputs. In the Max and Min columns of Tables 3 and 4, the maximum or minimum values of the reward of the corresponding input are shown. Each input is shown in Tables 3 and 4, taking only the top 25% (0.168 or more) or lower
0.3. As over-generalization occurs when the threshold value of

25% (0.125 or less) of the ratio of the deep sleep time. The performance problem, these parameters are set to 100,000 and

800 or 100

in the SGEMM GPU kernel performance problem is the accurate classifier covers 120,784 inputs whose

1 are recognized as classifiers to be re-

acquired classifiers in the care plan-sleep problem match only

stable.

Next, the results of the SGEMM GPU kernel performance problem are described. In Table 7, the classifiers acquired by

0.2 or

1a n d

parameters (prediction p, e, e0, fitness F, numerosity n, experience exp), respectively. The classifiers in each table are sorted in ascending order by the value of p. The classifiers listed in Tables 5 and 6 are not all of the classifiers acquired by XCS or XCS-RB; only classifiers with k = 1 and exp > 20 are extracted.

From Tables 5 and 7, the classifiers acquired by XCS were aggregated into an over-generalized classifier where all items of the condition part are #. The over-generalized classifier was judged to be accurate, but some classifiers were not sub-

sumed. Some classifiers were judged as accurate by XCS. Because XCS did not determine the classifier accuracy correctly, XCS failed to learn correctly. From Table 6, XCS-RB acquired 22 classifiers in the care plan-sleep problem and 80,846 clas-

sifiers in the SGEMM GPU kernel performance problem. The acquired classifiers in the care plan-sleep problem match only with the inputs to be covered in Tables 3 and 4. As

value of

ρ.

8. Discussion

8.1 Classifiers Acquired by XCS

XCS acquired a classifier whose condition part is "#####". Because this classifier matched all inputs, it was obviously not a classifier to be acquired in this study because a

Table 3 Input state to be covered (lower 25%).

| Wake up time | Time in bed | Tea | Gardening | Bath | Snack | Newspaper | Rehabilitation | Number of instances | Max  | Min |
|--------------|-------------|-----|-----------|------|-------|------------|------------------|---------------------|------|-----|
| 0            | 1           | 0   | 0         | 0    | 0     | 0          | 0                | 2                   | 0.0038 | 0.0465 |
| 1            | 0           | 0   | 0         | 0    | 0     | 0          | 0                | 1                   | 0.124 | 0.124 |
| 1            | 0           | 0   | 2         | 0    | 0     | 0          | 1                | 0                   | 0.084 | 0.084 |
| 1            | 0           | 2   | 2         | 2    | 2     | 1          | 1                | 0.120 | 0.120 |
| 1            | 0           | 3   | 2         | 0    | 0     | 1          | 1                | 0.124 | 0.124 |
| 1            | 1           | 0   | 0         | 1    | 0     | 0          | 0                | 1                   | 0.007 | 0.007 |
| 1            | 1           | 0   | 0         | 2    | 2     | 0          | 0                | 1                   | 0.118 | 0.118 |
| 1            | 1           | 1   | 0         | 0    | 0     | 0          | 1                | 1                   | 0.197 | 0.197 |
| 1            | 1           | 1   | 0         | 0    | 0     | 0          | 1                | 0.058 | 0.058 |
| 1            | 2           | 0   | 0         | 0    | 0     | 0          | 1                | 0.0032 | 0.0032 |
| 1            | 2           | 0   | 0         | 1    | 0     | 0          | 1                | 0.084 | 0.084 |

Table 4 Input state to be covered (top 25%).

| Wake up time | Time in bed | Tea | Gardening | Bath | Snack | Newspaper | Rehabilitation | Number of instances | Max  | Min |
|--------------|-------------|-----|-----------|------|-------|------------|------------------|---------------------|------|-----|
| 0            | 1           | 0   | 0         | 0    | 0     | 0          | 0                | 2                   | 0.0038 | 0.0465 |
| 1            | 0           | 0   | 0         | 0    | 0     | 0          | 0                | 1                   | 0.124 | 0.124 |
| 1            | 0           | 0   | 2         | 0    | 0     | 0          | 1                | 0                   | 0.084 | 0.084 |
| 1            | 0           | 2   | 2         | 2    | 2     | 1          | 1                | 0.120 | 0.120 |
| 1            | 0           | 3   | 2         | 0    | 0     | 1          | 1                | 0.124 | 0.124 |
| 1            | 1           | 0   | 0         | 1    | 0     | 0          | 0                | 1                   | 0.007 | 0.007 |
| 1            | 1           | 0   | 0         | 2    | 2     | 0          | 0                | 1                   | 0.118 | 0.118 |
| 1            | 1           | 1   | 0         | 0    | 0     | 0          | 1                | 1                   | 0.197 | 0.197 |
| 1            | 1           | 1   | 0         | 0    | 0     | 0          | 1                | 0                   | 0.058 | 0.058 |
| 1            | 2           | 0   | 0         | 0    | 0     | 0          | 1                | 0.0032 | 0.0032 |
| 1            | 2           | 0   | 0         | 2    | 0     | 0          | 0                | 1                   | 0.084 | 0.084 |

For the parameter settings, the following values were em-

ployed in the experiments, which are the most standard ones in

the conventional XCS [16]: N = 800 or 100,000, ε0 = 0.02 or

0.3, μ = 0.04, ρk = 0.35, χ = 0.8, ν = 50, θd,k = 25, θdel = 20, and

θsub = 20. In the care plan-sleep problem, N and ε0 are set to 800 and 0.02, respectively. In the SGEMM GPU kernel performance problem, these parameters are set to 100,000 and

0.3. As over-generalization occurs when ε0 is larger than nec-

essary, the ε0 value is set to close to half the average value of the reward range of the input to be covered.

The learning data consists of 323 instances and 241,600 in-

stances that are repeated in chronological order such that there are a total of 1,000,000 items. Every time a combination of one input and reward is given, XCS learns.

7.3 Results

First, the results of the care plan-sleep problem are described. In Table 5, the classifiers acquired by XCS are shown. In Table 6, the classifiers acquired by XCS-RB are given. Each column of the tables represents a condition part (Wake up time, Time in bed, Tea time, Gardening, Bath time, Snack time, Newspaper reading, Rehabilitation) and parameters (prediction p, e, e0, fitness F, numerosity n, experience exp), respectively. The classifiers in each table are sorted in ascending order by the

experience classifiers, the learning performance of XCS-RB is

stable.

8. Discussion

8.1 Classifiers Acquired by XCS

XCS acquired a classifier whose condition part is "#####". Because this classifier matched all inputs, it was obviously not a classifier to be acquired in this study because a
classifier matching all inputs acquires all rewards. As the value of $\epsilon$ was larger than $\epsilon_0$, the classifier should have been judged to be inaccurate and therefore deleted. In fact, it was not so; the over-generalized classifier was acquired as an accurate classifier.

The reason for this is the distribution of the reward. In Fig. 4, the reward distribution of the care plan-sleep problem is shown. In this figure, the horizontal axis indicates the value of the reward acquired by XCSs and the vertical axis indicates the number of rewards acquired. As the value of the reward is a continuous value, the acquired rewards were divided into 20, and the number of acquiring times was calculated. Many rewards were distributed around the average value of the reward. As shown in Eq. (2), $\epsilon$ of XCS is strongly affected by the recently acquired reward. If the values of the recently acquired reward are close, $\epsilon$ becomes small. A classifier whose value of $\epsilon$ is less than $\epsilon_0$ and should be inaccurate is determined to be accurate. As shown in...
Fig. 4, because many rewards were distributed around the average value of the reward, the classifier whose condition part was “########” was judged as an accurate classifier. At the time, the classifiers belonging to \( [M] \) were subsumed by the classifier whose condition part was “########.” On the other hand, the all # classifier acquired rewards of various values at the least and was judged as an inaccurate classifier in some cases (iteration). At the time, the classifiers belonging to \( [M] \) were not subsumed by the classifier whose condition part was “########.” The non-subsumed classifiers are the classifiers in Tables 5 and 7. Because the subject of the subsumed classifier is the classifier belonging to \( [M] \) and in this experiment, the learning data is the care plan is repeated in the order of the date, the specific classifier remained in \( [P] \).

The reason why the over-generalized classifier remained is the instability of the value of \( \epsilon \) mentioned above and the numerosity \( n \) being large; \( n \) increases with each subsumption of the other classifiers. The over-generalized classifier could subsume all the other classifiers. Thus, \( n \) of the over-generalized classifier became very large. Even if it is determined that the over-generalized classifier is inaccurate and it is subject to deletion in the rule-discovery component, because the value of \( n \) only decreases, it remained in \( [P] \). The speed at which the value of \( n \) of the over-generalized classifier increases is greater than the speed at which it decreases. In problems such as the one addressed in this paper, the calculation of \( \epsilon \) according to Eq. (2) is inappropriate.

### 8.2 Classifiers Acquired by XCS-RB in the Care Plan-Sleep Problem

The 22 classifiers acquired by XCS-RB that are shown in Table 6 cover the 24 type inputs to be covered (shown in Tables 3 and 4); \( \epsilon_0 \) of the classifiers that acquire only the lower 25% of the reward is 0.0801, and \( \epsilon_0 \) of the classifiers that acquire only the top 25% of the reward is 0.0996; \( p \), representing the prediction (expectation) value of the reward acquired by the classifier on the upper side and the lower side in Table 6, is largely different. XCS-RB properly recognizes which classifier the classifier is close to. The acquired classifiers seem to be generalized at first glance because the condition part of the classifiers has the # symbol. However, the number of inputs to be matched by each acquired classifier is less than the maximum number of types of input to be matched. Four of those classifiers matched two inputs, respectively. The other classifiers each match only one type of input. The reason for the small number of types of input to be matched for the acquired classifier is the small number of input types of learning data. As shown in Table 1, the eight behaviors each take four types of values. The size of the input space is \( 4^8 = 65536 \). There are 323 learning data, and the input type of the learning data is 61. This means that only about 0.1% of the state space is given to XCS and XCS-RB. No information on the remaining 99.9% of input is given to XCSs at all. As with other machine learning methods, XCSs cannot handle inputs not included in the learning data. In this case, XCSs not only cannot predict the correct reward, but the generality of the classifiers is higher than that when all the data are given to XCSs. XCS-RB determines whether the classifiers whose acquired reward is within a certain range are accurate. The inputs not included in learning data are not taken into account in the generalization of the classifier. XCS-RB judges whether the reward acquired by the classifier falls within a certain range of the learning data. It is not possible to judge whether the range of the reward acquired by the classifier in unknown data not included in the learning data is within the certain range or not. The inputs not included in learning data do not affect classifier parameters. Because \( p \) and \( \epsilon \) of the accurate classifier do not increase, their inputs are treated as inputs that can be matched by the classifier generalization. The classifiers are generalized such that the generality of it is maximized to the extent not inconsistent with learning data, but not generalized to blindly. The classifiers are over-generalized as more than one classifier matches unknown data in learning data.

As described above, the excessive classifier generalization is regarded as an over-learning of learning data. On the other hand, it can also be seen that the behavior items to be considered have been clarified more clearly. It is a rare case that all state spaces are covered in real-world problems such as the care plan-sleep problem in this paper. Thus, very limited combinations (inputs) are repeated. When trying to cover all state spaces, the input becomes unnatural or costly, and it becomes unrealistic. In such a situation, the overlearning described above becomes a difficult problem to manage. XCS-RB will cut off items that need not be taken into consideration; hence, the readability of the classifiers will be further improved. For example, in Table 4, the value of the column “Wake up time” is only 1. Because this column always takes the same value, it is not necessary to consider it in estimating the ratio of the time of deep sleep. In the lower classifier of Table 6, actually acquired by XCS-RB, the entire “Wake up time” column contains only the # symbol.

Fitness \( F \) of the last and the second from the bottom classifiers in Table 6 are smaller than \( F \) of the other classifiers acquired by XCS-RB. The condition part of these classifiers are “###21###0” and “####1#20.” These classifiers match only an input “10221020.” As these classifiers are matched to the same input, the value of \( \kappa \) decreases according to Eq. (6). Because the value of \( \kappa \) is small, the value of \( F \) decreases. The multiplexer problem often used as an LCS testbed problem has only one optimally generalized classifier subset. The classifiers in that optimal subset have no duplication and omission in the matching input. The classifier with no duplication in the input to be compared with other classifiers has higher \( F \), and the classifier with the duplication has low \( F \). In the multiplexer problem, XCS learning is stable because there is only one subset of the classifiers without duplication and leakage. Such a subset does not necessarily exist in a real-world problem. The example is the classifier “###21###0” and “####1#20.” As both classifiers match to the same single input, their evaluation (parameters) is almost the same. XCS-RB cannot decide which classifier to delete.

The above problems are caused by the expressive ability of the # symbol and the small number of data compared to the size of the input space. The # symbol means all values of that column. Given the care plan-sleep problem in this paper, there is a column that takes three or more values. In that case, the generalization by # symbol is too rough. If only one value is a factor to acquire a different reward, the classifiers are not generalized but remain divided by the number of values that the column takes. It is necessary to introduce a mechanism that realizes a method of collectively representing only a part of values.
in the column, as in Wilson’s paper [17]. From the viewpoint of preventing overlearning, the introduction mechanism for adjusting the number of # symbols according to the number of inputs to be matched by the classifier is a measure. Since these measures may increase the number of acquired classifiers or decrease adaptability to unknown data, they need to be verified. On the other hand, since it may take much time and cost to acquire data like sleep data used in this paper, a method to acquire generalized classifier efficiently with a small number of data is required.

Since classifiers are generated by genetic algorithm, XCS-RB also generate the “#” classifier. However, since XCS-RB determines the accuracy of the classifier based on the maximum and minimum values of the acquired reward by the classifier, the “#” classifier is immediately judged as inaccurate. Even if the acquired reward of the classifier is biased to a specific range, since the maximum and minimum values of the acquired reward are held, the judgment of the accuracy of the classifiers in XCS-RB is stable. The “#” classifier cannot subsume the other classifiers.

### 8.3 Validity of the Classifiers Acquired by XCS-RB in the Care Plan-Sleep Problem

To verify the validity of the acquired classifiers, this section compares the acquired classifiers with the rule derived by Apriori [18] which is a representative method of association rule mining. Since association rule learning can handle only discrete data, it is necessary to convert the continuous reward to discrete values. The rewards less than the lower threshold $L$ (0.125), the rewards greater than the upper threshold $U$ (0.168), and the other rewards are labeled -1, 1, and 0, respectively. In order to extract the rule taking only the label -1, or 1, Apriori’s parameter confidence was set to 1. Apriori generated 776 rules taking the label 1 and 766 rules taking the label -1. Table 9 shows the association rules with length 2 or less acquired by Apriori as part of the acquired rules. Each column of the tables represents the left-hand side of the rules, the right-hand side of the rules, the support value, the confidence value, the lift value, and the count value, respectively. Since there is only one learning data with Rehabilitation equal to 3, these rules match only the input “11000003”. The “#” classifier in Table 6 acquired by XCS-RB corresponds to these rules. Since Apriori lists a large number of combinations, it is difficult to make care plans from those combinations. The classifiers acquired by XCS-RB are the same as the rules with the minimum length in the rule matching to the same input among the rules acquired by Apriori. Although omitted from the space of this paper, it was confirmed that XCS-RB could acquire the useful classifiers even if the value of the upper threshold $U$ and lower threshold $L$ were changed. XCS-RB is more suitable for this paper than Apriori.

### 8.4 Classifiers Acquired by XCS-RB in the SGEMM GPU Kernel Performance Problem

Table 8 shows 10 classifiers whose number is large among the acquired accurate classifiers. From Table 8, since KWG, MDIMA, NDIMB, KWI, VWM, STRM, and STRN columns are # symbol in all classifiers, these columns do not affect the calculation speed. The other columns containing many non-# symbol values have great influences on the calculation speed. For example, the larger NDIMC value, the faster the calculation speed. By extracting the accurate classifiers with large $n$, it is possible to acquire the general classifiers that match many inputs and recognize important GPU kernel parameters.

### 9. Conclusions

This paper proposes new XCS, XCS-RB (XCS based Reward Bipolarization) that acquires only classifiers that acquire only high rewards and classifiers that acquire only low rewards as one kind of useful knowledge in continuous reward environment. The conventional method XCS cannot set accuracy criterion $\epsilon_0$ to an appropriate value since prediction error $\epsilon$ is calculated from the difference between prediction $p$ and the actually acquired reward. Examining learning data beforehand, even if the value of $\epsilon_0$ is set appropriately, since the value of $\epsilon$ is unstable, the over-generalization of the classifier occurs. In the experiment of this paper, a classifier in which all items of the condition part are # symbol is acquired. XCS-RB determines the values of $\epsilon$ and $\epsilon_0$ using the maximum and minimum rewards acquired by the classifier. The classifiers acquired by XCS-RB are stably evaluated. XCS-RB can acquire only classifiers that acquire only high rewards and classifiers that acquire only low rewards. The acquired classifiers matched all the input to be covered.

The dashed line in Fig. 5 shows the calculation time of XCS-RB. The amount of increase in the calculation time increases as
the number of data to be read increases. Since the all acquired rewards are recorded in ascending order, the calculation order for calculating \( L \) and \( U \) is \( O(i) \), where \( i \) is the current iteration number (the number of receiving input times). Since \( n \) does not exceed the parameter \( \bar{N} \), \( i \gg n \). It takes much time to calculate the threshold \( L \) and \( U \). The solid line in Fig. 5 shows the calculation time of XCS-RB without calculation of \( L \) and \( U \). The calculation time is linearly increasing regardless of the number of learning data. The calculation times were measured using a computer that has Intel Core i7-6700 @3.4 GHz, 16 GB RAM. When handling a large amount of data, it is necessary to reduce the computational complexity of XCS-RB.

Such important directions must be pursued in the near future in addition to the following future research: (1) improvement of the \# symbol representation method according to the type of value that input item takes and the generality of acquired classifiers; (2) improvement of learning efficiency with a small number of data; and (3) adaptation to environments with vast input space.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number JP17J03593.

References

[1] J.H. Holland: Escaping Brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems, Machine Learning, pp. 593–623, Morgan Kaufmann, 1986.
[2] D.E. Goldberg: Genetic Algorithms in Search, Optimization and Machine Learning, 1st edition, Addison-Wesley Longman Publishing, 1989.
[3] R.S. Sutton: Learning to predict by the methods of temporal differences, Machine Learning, Vol. 3, No. 1, pp. 9–44, 1988.
[4] S.W. Wilson: Get Real! XCS with Continuous-Valued Inputs, pp. 209–219, Springer Berlin Heidelberg, 2000.
[5] S.W. Wilson: Classifier fitness based on accuracy, Evol. Comput., Vol. 3, No. 2, pp. 149–175, June 1995.
[6] T. Tatsumi, T. Komine, M. Nakata, H. Sato, and K. Takadama: A learning classifier system that adapts accuracy criterion, Transaction of the Japanese Society for Evolutionary Computation, Vol. 6, No. 2, pp. 90–103, 2015.
[7] T. Tatsumi, H. Sato, T. Kovacs, and K. Takadama: Applying variance-based learning classifier system without convergence of reward estimation into various reward distribution, 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 2630–2637, June 2017.
[8] T. Tatsumi, H. Sato, and K. Takadama: Automatic adjustment of selection pressure based on range of reward in learning classifier system, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO ’17), pp. 505–512, New York, NY, USA, 2017.
[9] P.L. Lanzi and S.W. Wilson: Toward optimal classifier system performance in non-Markov environments, Evol. Comput., Vol. 8, No. 4, pp. 393–418, December 2000.
[10] A. Webb, E. Hart, P. Ross, and A. Lawson: Controlling a Simulated Khepera with an XCS Classifier System with Memory, pp. 885–892, Springer Berlin Heidelberg, 2003.
[11] P.L. Lanzi and M. Colombetti: An extension to the XCS classifier system for stochastic environments, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99), pp. 353–360, 1999.
[12] H. Takagi: Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation, Proceedings of the IEEE, Vol. 89, No. 9, pp. 1275–1296, 2001.
[13] T. Tatsumi, T. Komine, M. Nakata, H. Sato, T. Kovacs, and K. Takadama: Variance-based learning classifier system without convergence of reward estimation, Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion (GECCO ’16 Companion), pp. 67–68, 2016.
[14] K. Takadama and M. Nakata: Extracting both generalized and specialized knowledge by XCS using attribute tracking and feedback, 2015 IEEE Congress on Evolutionary Computation (CEC), pp. 3034–3041, May 2015.
[15] C. Nugteren and V. Codreanu: CLTune: A generic auto-tuner for OpenCL kernels, 2015 IEEE 9th International Symposium on Embedded Multicore/Many-core Systems-on-Chip, pp. 195–202, Sept. 2015.
[16] M.V. Butz, T. Kovacs, P.L. Lanzi, and S.W. Wilson: Toward a theory of generalization and learning in XCS, IEEE Transactions on Evolutionary Computation, Vol. 8, No. 1, pp. 28–46, 2004.
[17] S.W. Wilson: Mining oblique data with XCS, Revised Papers from the Third International Workshop on Advances in Learning Classifier Systems, IWLCS ’00, pp. 158–176, Springer-Verlag, 2001.
[18] R. Agrawal and R. Srikant: Fast algorithms for mining association rules, Proc. 20th Intl. Conf. VLDB, pp. 487–499, 1994.

Takato Tatsumi (Student Member)

He received his B.S. and M.S. degrees from the University of Electro-Communications, Japan, in 2015 and 2017, respectively. He is currently a Ph.D. student at the University of Electro-Communications. His research interests include evolutionary computation and reinforcement learning.

Keiki Takadama (Member)

He received his M.S. and Ph.D. degrees from Kyoto University, Japan, in 1995 and 1998, respectively. In 2006, he joined the University of Electro-Communications, where he is currently a Professor of the Department of Informatics. His research interests include evolutionary computation, reinforcement learning, multi-agent system, and healthcare system. He is a member of IEEE, ACM.