Two-Step Markov Update Algorithm for Accuracy-Based Learning Classifier Systems

Mohammad Razeghi-Jahromi
Shabnam Nazmi
Abdollah Homaifar
Department of Electrical and Computer Engineering
North Carolina A&T State University
Autonomous Control and Information Technology (ACIT) Institute
1601 East Market Street
Fort IRC Building
Greensboro, NC 27411, USA
mohammad.razeghi-jahromi@us.abb.com
snazmi@aggies.ncat.edu
homaifar@ncat.edu

In this paper, we investigate the impact of a two-step Markov update scheme for the reinforcement component of XCS, a family of accuracy-based learning classifier systems. We use a mathematical framework using discrete-time dynamical system theory to analyze the stability and convergence of the proposed method. We provide frequency domain analysis for classifier parameters to investigate the achieved improvement of the XCS algorithm, employing a two-step update rule in the transient and steady-state stages of learning. An experimental analysis is performed to learn to solve a multiplexer benchmark problem to compare the results of the proposed update rules with the original XCS. The results show faster convergence, better steady-state training accuracy and less sensitivity to variations in learning rates.

Keywords: two-step Markov update rule; accuracy-based classifier system; stability and convergence analysis; linear discrete-time dynamical system

1. Introduction

Learning classifier systems (LCSs) [1] are machine learning techniques that use evolutionary computation and reinforcement learning in a supervised environment to model the system in the form of a population of "IF condition, THEN action" rules. LCSs started to become more popular after Wilson's groundbreaking proposal, a simpler LCS structure with no message list and a new fitness calculation approach based on the classifier's accuracy in predicting the environment payoff, namely XCS [2]. XCS employs a Q-learning-like [3] update strategy and can work as a reinforcement learning or supervised learning

https://doi.org/10.25088/ComplexSystems.27.1.63

framework. In XCS, a genetic algorithm favors the evolution of those classifiers that are maximally general and at the same time accurate in labeling problem instances. A genetic algorithm attempts to increase the number of classifiers with higher fitness by means of crossover and mutation.

Performance of XCS in data analysis applications has been investigated and proven to exceed some of the well-known machine learning approaches [4, 5]. A variety of studies have been done to even further simplify the structure of XCS and make it more appropriate for supervised learning problems, resulting in systems such as UCS [6]. ExSTraCS is another class of LCSs that employs a specific rule representation scheme to store specified features in the classifier condition, which helps to deal with problems with a larger number of features [7]. XCS is also proposed to address clustering problems using learning classifiers [8]. Moreover, Stolzmann introduced ACS [9], which is an anticipatory learning classifier system that is able to learn and predict the current state and also the next state of the problem and has been shown to perform well in problems such as learning a maze. In function approximation applications, XCSF [10, 11] is proposed, which is able to learn an n-dimensional function using different approximations such as hyper-rectangles, hyper-ellipsoids and convex halls. Originally, LCS was structured to model a Boolean environment, although numerous studies have been proposed to represent real-valued problems such as XCSR [5, 12] and problems with mixed attributes as in AKLR [13]. The robust bidding strategy of strength-based classifiers is studied in [14], and later in MLCS [15] is extended to handle multi-label data, while the labels are allowed to have confidence levels.

It has been shown that XCS is able to optimally solve Markov problems, whereas it suffers when handling non-Markov problems. Adding memory can help to overcome this problem through an internal register, which can store limited information about previous states. Systems that adopt memory include XCSM [16] and XCSMH [17]. In XCSM, the internal bit register is implemented directly instead of a list of messages; however, in XCSMH, a compound exploration is used in which the exploration strategy chooses internal actions (register settings) deterministically, while selection of external actions remains probabilistic.

In XCS, each classifier has a set of parameters that are updated in every cycle in which the classifier participates in learning. The XCS reinforcement module employs a Markov update scheme that uses only the current value of the parameter to calculate its future value. However, a classifier's interaction with its environment has non-Markov behavior, in the sense that given the present attributes of the classifier, future values are not independent of the past. It is useful to

explicitly use an agent's past parameter values in the update rule to improve estimates of the parameter's future values. Therefore, we initially focus on extending current update rules of XCS parameters to an update rule with an additional history value.

Moreover, we aim to address the current LCS literature's shortfall for analyzing interactions among classifier parameters in the algorithm. The lack of analysis tools affects our understanding of different aspects of the problem, such as algorithm stability, convergence and frequency domain analysis for performance investigation. Employing various mathematical tools provides better insight to the problem and future extensions of the algorithm.

This paper provides the following major contributions. First, the two-step Markov update scheme for the XCS algorithm is formulated to investigate its stability and convergence. Second, a frequency domain analysis of the discussed update scheme has been provided to investigate faster convergence and better steady-state training accuracy. Since evolutionary-based machine learning algorithms are computationally more expensive than non-evolutionary approaches, reaching a reasonable accuracy threshold with fewer training iterations is crucial. Moreover, a mathematical framework using discrete-time dynamical system theory has been used to analyze the stability and convergence of two-step Markov update rules. Additionally, the sensitivity of the two-step update scheme to variations of learning rates is studied numerically.

2. XCS Classifier Overview

In this section, a brief description of the XCS algorithm is introduced. More details are provided in [2, 18, 19].

XCS acts as a reinforcement learning agent that receives inputs regarding the current state of the environment, reacts with actions and eventually receives a payoff as an indication of the effectiveness of its action. The goal of XCS is to maximize the amount of payoff gathered in the long run.

The interaction of XCS with the environment is as follows. The core of the XCS is a population of rules, also known as classifiers, that each consists of a condition, an action and a number of parameters that indicates their accuracy. Once the algorithm receives an input from the environment, it forms a match set $[M]$ of all classifiers that have a matching condition with the current input. The algorithm starts with an empty population and for each input that has no matching classifier in the population creates a new rule with a matching condition and the correct label of the input. This process is called covering. For each unique action a in $[M]$, the prediction $P(a)$ is

https://doi.org/10.25088/ComplexSystems.27.1.63

computed, which is an estimate of the payoff that the learner expects from the system when action a is performed. The prediction array is calculated by the fitness-weighted average of all matching classifiers that support action a as

$$P_a = \frac{\sum_{cl \in [M] \wedge cl.a=a} p_{cl} F_{cl}}{\sum_{cl \in [M] \wedge cl.a=a} F_{cl}}, \tag{1}$$

where p is the prediction of each individual classifier cl and F is the classifier fitness. P_a forms a prediction array for different values of action a, and XCS selects the action with respect to the values in the prediction array, either by selecting maximum prediction or randomly selecting one from those suggested by classifiers in $[M]$ [18].

After an action is selected, the action set $[A]$ is created using the classifiers that advocate the selected action. After the action is performed, the prediction p and prediction error ε of all classifiers in $[A]$ are updated through the following equations:

$$p \leftarrow p + \beta(R - p), \tag{2}$$

$$\varepsilon \leftarrow \varepsilon + \beta(\mid R - p \mid - \varepsilon), \tag{3}$$

where $\beta \in (0, 1]$ is the learning rate and R is the payoff received from the environment. Finally, fitness F is updated toward the classifier's current relative accuracy κ', which is a function of classifier prediction error as follows:

$$F \leftarrow F + \beta(\kappa' - F). \tag{4}$$

3. Introductory Review of Linear Discrete-Time Dynamical Systems

As stated in the Introduction, the stability and convergence of the proposed two-step Markov update rules are investigated in the framework of discrete-time dynamical systems. For this purpose, we give a short description of some of the fundamental mathematical tools in this section, which are necessary for analyzing linear discrete-time dynamical systems. More details are provided in [20–22].

3.1 *z*-Transform

A mathematical tool commonly used for the analysis of discrete-time dynamical systems is the z-transform. The role of the z-transform in discrete-time systems is similar to that of the Laplace transform in continuous-time systems.

The one-sided (unilateral) z-transform of a sequence $x(t)$ where t takes zero or positive integers (i.e., $x(t) = 0$ for all $t < 0$) is defined as

$$x(z) = \mathcal{Z}\{x(t)\} = \sum_{t=0}^{\infty} x(t) z^{-t}, \tag{5}$$

where the symbol $\mathcal{Z}$ denotes the z-transform of the term inside brackets. Equation (5) is, in general, an infinite sum or infinite power series, with z being a complex variable. For any given sequence, the set of values of z for which the z-transform converges is called the region of convergence (ROC). If $x(z)$ is given as a rational function inside the ROC, that is,

$$x(z) = \frac{p(z)}{q(z)}, \tag{6}$$

where $p(z)$ and $q(z)$ are polynomials in z, then the zeros are the roots of the numerator polynomial and the poles (for finite values of z) are the roots of the denominator polynomial.

3.2 Discrete-Time Fourier Transform

Another mathematical tool commonly used for the analysis of discrete-time dynamical systems is the discrete-time Fourier transform. The discrete-time Fourier transform of a sequence $x(t)$ where t takes zero or positive integers (i.e., $x(t) = 0$ for all $t < 0$) is defined as

$$x\left(e^{j\omega}\right) = \sum_{t=0}^{\infty} x(t) e^{-j\omega t}, \tag{7}$$

with ω being a real number ranging over an interval of length 2π and $j \triangleq \sqrt{-1}$. In particular, if we replace the complex variable z in equation (5) with the complex variable $e^{j\omega}$, then the z-transform reduces to the Fourier transform. This is one motivation for the notation $x\left(e^{j\omega}\right)$ for the Fourier transform; when it exists, the Fourier transform is simply $x(z)$ with $z = e^{j\omega}$. Mathematically, it means that the Fourier transform of $x(t)$ converges absolutely if and only if the ROC of the z-transform of $x(t)$ includes the unit circle.

3.3 Initial Value Theorem

If $x(t)$ has the z-transform $x(z)$ and if $\lim_{z\to\infty} x(z)$ exists, then the initial value $x\left(0\right)$ of $x(t)$ is given by

$$x\left(0\right) = \lim_{z\to\infty} x(z). \tag{8}$$

Basically, this theorem relates frequency domain analysis to the time domain behavior as time approaches zero.

https://doi.org/10.25088/ComplexSystems.27.1.63

3.4 Final Value Theorem

If $x(t)$ has the z-transform $x(z)$ and all the poles of $x(z)$ lie inside the unit circle, with the possible exception of a pole of order one at $z = 1$, then the final value of $x(t)$, that is, the value of $x(t)$ as t approaches infinity, is given by

$$\lim_{t\to\infty} x(t) = \lim_{z\to 1}\big[(z-1)x(z)\big]. \tag{9}$$

In contrast to the initial value theorem, the final value theorem relates frequency domain analysis to the time domain behavior as time approaches infinity.

3.5 Linear Discrete-Time Dynamical System

The z-transform is particularly useful in the analysis of linear time-invariant (LTI) systems described by the difference equations. Consider the n^{th}-order LTI discrete-time system characterized by the following linear difference equation:

$$\begin{aligned} &y(t) + a_1 y(t-1) + \cdots + a_n y(t-n) = \\ &\quad b_0 u(t) + b_1 u(t-1) + \cdots + b_n u(t-n), \end{aligned} \tag{10}$$

where $u(t)$ and $y(t)$ are the system's input and output, respectively, at the t^{th} time step. Note that some of the coefficients a_i and b_j may be zero. By taking the z-transform of equation (10), the system transfer function is given by

$$H(z) \triangleq \frac{y(z)}{u(z)} = \frac{b_0 z^n + b_1 z^{n-1} + \cdots + b_n}{z^n + a_1 z^{n-1} + \cdots + a_n}. \tag{11}$$

3.6 Lyapunov Stability

Consider the discrete-time system with the state equation

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \tag{12}$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{f} \in \mathbb{R}^n$ with the property that $\mathbf{f}(0, 0) = 0$. An equilibrium point $\mathbf{x} = 0$ of the unforced (zero input) dynamical system

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t), 0), \tag{13}$$

is called globally uniformly asymptotically stable (GUAS), if it is stable (in the Lyapunov sense) and every solution converges to zero as t goes to infinity, that is, $\lim_{t\to\infty} \| \mathbf{x}(t) \| = 0$.

If the unforced system in equation (13) has a GUAS equilibrium point at the origin $\mathbf{x} = 0$, then we are interested in whether a bounded input $\mathbf{u}(t)$ implies that the state $\mathbf{x}(t)$ is bounded too. This is the notion of input-to-state stability. It is shown in [20] that if

equation (13) has a GUAS equilibrium point at the origin $\mathbf{x} = \mathbf{0}$, then equation (12) is input-to-state stable.

There are many ways to realize state-space representations for the LTI discrete-time system described by equation (10) or (11) as

$$\begin{aligned} \mathbf{x}(t+1) &= \mathbf{G}\mathbf{x}(t) + \mathbf{H}\mathbf{u}(t), \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t), \end{aligned} \tag{14}$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$, $\mathbf{y} \in \mathbb{R}^r$ and $\mathbf{G}$, $\mathbf{H}$, $\mathbf{C}$ and $\mathbf{D}$ are known matrices with compatible dimensions. The origin $\mathbf{x} = \mathbf{0}$ of the unforced system

$$\mathbf{x}(t+1) = \mathbf{G}\mathbf{x}(t) \tag{15}$$

is GUAS if and only if all eigenvalues of $\mathbf{G}$ have magnitude less than 1; that is, $|\lambda_i(\mathbf{G})| < 1$. Since the characteristic equation that is the determinant of $|z\mathbf{I} - \mathbf{G}|$ can be written as

$$\Delta(z) \triangleq |z\mathbf{I} - \mathbf{G}| = z^n + a_1 z^{n-1} + \cdots + a_n, \tag{16}$$

therefore, the origin $\mathbf{x} = \mathbf{0}$ of the unforced system in equation (15) is GUAS if and only if the poles of the system transfer function in equation (11) lie inside the unit circle in the z plane.

3.7 Jury Stability Test

The Jury stability test can be applied directly to the characteristic equation $\Delta(z) = 0$ without solving for the roots. The test reveals the existence of any unstable roots, that is, the roots that lie outside the unit circle in the z plane.

4. XCS with Two-Step Markov Update Algorithm

4.1 Reinforcement Component

In this section, the two-step Markov update scheme for the reinforcement component of the XCS is formulated, based on the fact that when the present value of the classifier is known, then its future value is not independent of the past (non-Markov), and past information (observations) can be explicitly used in the classifier update process to achieve better predictions of future values. One can increase a two-step Markov update to an n-step Markov update in general. Note that only the classifier parameters in the action set $[A]$ are updated; that is, whenever a classifier participates in an action set, its prediction, prediction error and fitness are updated based on the weighted sum of its present and its one-step previous values. The weights are controlled by different learning rates β and η.

https://doi.org/10.25088/ComplexSystems.27.1.63

In XCS, the Q-learning reinforcement module [3] is restated as a difference equation in which present values are used to calculate future values of the parameters. Therefore, classifier prediction $p(t)$ and prediction error $\varepsilon(t)$ are updated through the following modified difference equations in which the update arrow is replaced with an equal sign to better serve the purpose of discrete-time analysis. The convergence of the two-step update rule is guaranteed by choosing the proper learning rates as derived in Section 4:

$$p(t+1) = p(t) + \beta\left(\frac{\beta+\eta-1}{\beta}R - p(t)\right) + (1-\eta)p(t-1), \tag{17}$$

$$\varepsilon(t+1) = \varepsilon(t) + \beta(|R - p(t)| - \varepsilon(t)) + (1-\eta)\varepsilon(t-1), \tag{18}$$

for all iterations $t \geq 0$, initial conditions $p(0) = p(-1) = 0$ and $\varepsilon(0) = \varepsilon(-1) = 0$, where $\beta \in (1-\eta, 1]$ is the one-step learning rate, $\eta \in (0, 1]$ denotes the two-step learning rate and R is the payoff received from the environment. Note that in equation (17), the coefficient of R is adapted to guarantee the asymptotic convergence of the prediction $p(t)$ to R when the model is completely trained. This is shown in the proof of Lemma 1.

To update the classifier fitness $F(t)$, the classifier accuracy $\kappa(t)$ and the classifier relative accuracy $\kappa'(t)$ are computed as in the XCS algorithm [23], as shown here:

$$\begin{aligned} \kappa(t) &= \begin{cases} 1, & \text{if} \varepsilon(t) < \varepsilon_0 \\ \alpha\left(\dfrac{\varepsilon(t)}{\varepsilon_0}\right)^{-\nu}, & \text{otherwise} \end{cases} \\ \kappa'(t) &= \frac{\kappa(t)\cdot \text{num}(t)}{\sum\limits_{cl\in[A]} \kappa_{cl}(t)\cdot \text{num}_{cl}(t)}. \end{aligned} \tag{19}$$

The parameter $\varepsilon_0 > 0$ controls the prediction error tolerance, the parameters $0 < \alpha < 1$, $\nu > 0$ are constants, and "num" is the number of copies of the classifier being updated. Finally, classifier fitness $F(t)$ is updated toward the classifier's current relative accuracy $\kappa'(t)$ through the following modified difference equation:

$$F(t+1) = F(t) + \beta\left(\frac{\beta+\eta-1}{\beta\kappa'(t) - F(t)}\right) + (1-\eta)F(t-1), \tag{20}$$

for all iterations $t \geq 0$ and initial conditions $F(0) = F(-1) = 0$. Note that in equation (20), the coefficient of $\kappa'(t)$ is adapted to guarantee the asymptotic convergence of the fitness $F(t)$ to 1 when the model is completely trained.

Corollary 1. $1-\beta$ and $1-\eta$ are the weights of present and previous values in the two-step Markov update equations (17), (18) and (20). By changing η continuously from 0^+ to 1, one can observe the entire spectrum of the algorithm behavior from pure prSevious value to pure present value update. As η approaches 1, the algorithm acts like the original XCS and vice versa. When $\beta < \eta$, more weight is given to the present than previous values and vice versa. If $\eta \in \left(1/2, 1\right]$, then there always exists $\beta \in \left(1-\eta, 1\right]$ such that $\beta = \eta$; that is, we have equal weights for the present and previous values.

4.2 Stability and Convergence Analysis

In this section, the set of classifier parameters is analyzed for stability and convergence of the learning algorithm, that is, difference equations (17), (18) and (20). By finding a proper range for learning rates β and η, we have shown that the proposed two-step Markov update algorithm is GUAS; hence the stability and convergence of the algorithm is guaranteed.

Defining a new variable $q(t) \triangleq R - p(t)$ with initial conditions $q(0) = q(-1) = R$, equations (17) and (18) can be rewritten as

$$q(t+1) = (1-\beta)q(t) + (1-\eta)q(t-1), \tag{21}$$

$$\varepsilon(t+1) = (1-\beta)\varepsilon(t) + (1-\eta)\varepsilon(t-1) + \beta|q(t)|, \tag{22}$$

which is a discrete-time cascade dynamical system and has an equilibrium point at $(q, \varepsilon) = (0, 0)$. The set of dynamical systems (21) and (22) is called cascade since (21) does not depend on $\varepsilon(t)$ and the output of (21), that is, $q(t)$, is fed as an input to (22). To show that the equilibrium point of the set of equations (21) and (22) is GUAS, the following lemma is introduced.

Lemma 1. If equation (22), with $q(t)$ as its input, is input-to-state stable and equation (21) has a GUAS equilibrium point at the origin $q = 0$, then the cascade dynamical system (21) and (22) has a GUAS equilibrium point at the origin $(q, \varepsilon) = (0, 0)$.

Proof. Taking the z-transform of equation (21) with the initial condition $q(0) = R$ leads to

$$q(z) = \frac{Rz^2}{z^2 - (1-\beta)z - (1-\eta)}. \tag{23}$$

Equation (21) has a GUAS equilibrium point at its origin $q = 0$ if and only if the characteristic equation

$$\Delta(z) = z^2 - (1-\beta)z - (1-\eta) \tag{24}$$

https://doi.org/10.25088/ComplexSystems.27.1.63

is stable, that is, all the poles of $q(z)$ lie within the unit circle in the z plane. The Jury stability test is applied to determine the stability of the characteristic equation (24) without solving for the roots. Therefore, the following conditions must be satisfied:

$$\begin{aligned} |1-\eta| < 1 &\Rightarrow 0 < \eta < 2 \\ \beta + \eta - 1 > 0 &\Rightarrow \beta > 1 - \eta \\ -\beta + \eta + 1 > 0 &\Rightarrow \beta < 1 + \eta. \end{aligned} \tag{25}$$

In addition to these three conditions, two additional sufficient conditions are imposed by the non-negativity of present and previous updating weights on the states of the system to make sure the prediction, prediction error and fitness values are all non-negative for all t, as in the following:

$$\eta \le 1 \text{ and } \beta \le 1. \tag{26}$$

Considering the inequalities (25) and (26) together implies that

$$0 < \eta \le 1 \text{ and } 1 - \eta < \beta \le 1. \tag{27}$$

From the global asymptotic stability of the origin of equation (21), it is concluded that

$$\lim_{t\to\infty} q(t) = 0 \Rightarrow \lim_{t\to\infty} p(t) = R, \tag{28}$$

which shows that the classifier prediction $p(t)$ is bounded for all t and globally asymptotically converges to R.

In order to show that equation (22) is input-to-state stable, it is required that the unforced system

$$\varepsilon(t+1) = (1-\beta)\varepsilon(t) + (1-\eta)\varepsilon(t-1) \tag{29}$$

have a GUAS equilibrium point at its origin $\varepsilon = 0$, which is equivalent to having the same stable characteristic equation as (24). Hence, choosing the learning rates as in (27) ensures that (22) is input-to-state stable. Therefore, using the fact that the origin of system (21) is GUAS, we conclude that the origin of system (22), which is $\varepsilon = 0$, is also GUAS; that is, $\lim_{t\to\infty} \varepsilon(t) = 0$. This proves that the classifier prediction error $\varepsilon(t)$ is bounded for all t and globally asymptotically converges to zero. This concludes the proof of Lemma 1. □

Corollary 2. Satisfying the learning rates β and η in equation (27) forces both poles of the characteristic equation (24) to be always inside the unit circle. Furthermore, the poles always appear as one having a positive real root $(r_+ \ge 0)$ and one a negative real root $(r_- \le 0)$, as $(1-\beta)^2 + 4(1-\eta) \ge (1-\beta)^2$ for every β and η in equation (27). From characteristic equation (24), we have

$$r_+ = \frac{1-\beta+\sqrt{(1-\beta)^2+4(1-\eta)}}{2},$$
$$r_- = \frac{1-\beta-\sqrt{(1-\beta)^2+4(1-\eta)}}{2}. \tag{30}$$

Also, equation (27) implies that

$$0 \le 1-\eta < \beta \le 1 \Rightarrow 1-\beta \le \sqrt{(1-\beta)^2+4(1-\eta)} < 1+\beta. \tag{31}$$

Combining equations (30) and (31), we have

$$-1 \le -\beta < r_- \le 0 \le 1-\beta \le r_+ < 1. \tag{32}$$

In the case of $\eta = 1$, that is, $r_- = 0$, $r_+ = 1-\beta$, the characteristic equation (24) would be a first-order characteristic equation with a single pole at $z = 1-\beta$, as there is a pole zero cancellation in equation (23) at $z = 0$, which is the original one-step XCS.

Corollary 3. As explained in Corollary 2, the classifier prediction update equation (17) is a second-order system with two real distinct poles r_+ and r_- inside the unit circle. This system is called overdamped; that is, the response $p(t)$ is smooth and nonoscillatory and asymptotically follows the payoff R. Therefore, for every t there are

$$p(t) \le R \Rightarrow R - p(t) \ge 0. \tag{33}$$

As a result, $|R-p(t)|$ that originally is introduced in the XCS update rule can be replaced with $R-p(t)$ in the prediction error update equation (18). The same reasoning applies to one-step prediction as in equation (3), which is a stable first-order system that never oscillates.

An obvious challenge for such an evolving, distributed knowledge representation is the continuous support of all problem subspaces, which is identified as niche support. In XCS, niche support is guaranteed by a niche-based deletion method plus an occurrence-based reproduction method. In [24], assuming a Markov chain model for niche support, population is distributed according to a binomial distribution over all niches of the problem in steady state. It concludes that at steady state, equation (19) results in

$$\lim_{t\to\infty} \kappa'(t) = 1. \tag{34}$$

https://doi.org/10.25088/ComplexSystems.27.1.63

The relative accuracy $\kappa'(t)$ given in equation (19) is also bounded by $0 \le \kappa'(t) \le 1$ with $\kappa'(0) \neq 0$ as $\kappa(0) = 1$, due to initialization of $\varepsilon(0) = 0$. Using the final value theorem, equation (34) implies that $\kappa'(z)$ can be decomposed as

$$\mathcal{Z}\{\kappa'(t)\} = \kappa'(z) = \frac{1}{z-1} \cdot \frac{b(z)}{d(z)}, \tag{35}$$

where $b(z)$ and $d(z)$ are polynomials in z with no factor of $(z-1)$. Using the initial value theorem, we also have

$$\kappa'(0) \neq 0 \Rightarrow \lim_{z\to\infty} \kappa'(z) = \lim_{z\to\infty} \frac{1}{z-1} \cdot \frac{b(z)}{d(z)} \neq 0, \tag{36}$$

where $b(z) \,/\, d(z)$ is a stable rational transfer function that satisfies the following degree condition:

$$\deg(b(z)) = \deg(d(z)) + 1. \tag{37}$$

Since $\kappa'(z)$ satisfies the conditions of the final value theorem, as all of its poles lie inside the unit circle with a pole of order one at $z = 1$, we conclude that

$$1 = \lim_{t\to\infty} \kappa'(t) = \lim_{z\to 1} (z-1)\kappa'(z) = \frac{b(1)}{d(1)}. \tag{38}$$

The analysis of relative accuracy will shed light on the convergence of classifier fitness. To show that classifier fitness $F(t)$ is updated toward the classifier's current relative accuracy $\kappa'(t)$, equation (20) is rewritten as

$$F(t+1) = (1-\beta)F(t) + (1-\eta)F(t-1) + (\beta+\eta-1)\kappa'(t), \tag{39}$$

which is input-to-state stable with $\kappa'(t)$ as its input because the unforced equation (39) has the same characteristic equation as (24). Taking the z-transform of (39) with the initial condition $F(0) = 0$ yields

$$F(z) = \frac{(\beta+\eta-1)z}{z^2 - (1-\beta)z - (1-\eta)} \cdot \kappa'(z). \tag{40}$$

From equations (27) and (35) it can be concluded that $F(z)$ satisfies the conditions of the final value theorem. Consequently, we have

$$\lim_{t\to\infty} F(t) = \lim_{z\to 1} (z-1)F(z) = \lim_{z\to 1} (z-1)\kappa'(z) = \lim_{t\to\infty} \kappa'(t) = 1 \tag{41}$$

Therefore, the classifier fitness $F(t)$ is also bounded for all t and globally asymptotically converges to 1.

4.3 Frequency Domain Analysis

In order to gain a better understanding of the impact of the modified reinforcement component of the XCS on the dynamic of the overall algorithm, in this section, the frequency responses of the transfer functions between the input payoff R and the output prediction error $\varepsilon(t)$ are studied. In particular, this analysis will provide a clear understanding of the transient behavior and the steady-state accuracy of the proposed algorithm. For the proposed two-step and the original XCS update algorithms, parameters are denoted here by the subscripts "2-step" and "1-step," respectively.

Lemma 2. The two-step update scheme has a faster prediction error convergence rate in the transient and better steady-state accuracy compared to the original XCS.

Proof. The prediction and prediction error update equations for the one-step update scheme are given by

$$\begin{aligned} p_{\text{1-step}}(t+1) &= (1-\beta)p_{\text{1-step}}(t) + \beta R, \\ \varepsilon_{\text{1-step}}(t+1) &= (1-\beta)\varepsilon_{\text{1-step}}(t) - \beta p_{\text{1-step}}(t) + \beta R, \end{aligned} \tag{42}$$

which is a linear discrete-time cascade dynamical system. The transfer function from the input payoff R to the output prediction error $\varepsilon_{\text{1-step}}(t)$ is given by

$$H_{\text{1-step}}(z) \triangleq \frac{\varepsilon_{\text{1-step}}(z)}{R(z)} = \frac{\beta(z-1)}{\left(z-(1-\beta)\right)^2}. \tag{43}$$

Similarly, equations (17) and (18) together are also a linear discrete-time cascade dynamical system of the proposed method, and the transfer function from the input payoff R to the output prediction error $\varepsilon_{\text{2-step}}(t)$ is given by

$$H_{\text{2-step}}(z) \triangleq \frac{\varepsilon_{\text{2-step}}(z)}{R(z)} = \frac{\beta z(z-1)(z+1-\eta)}{\left(z^2-(1-\beta)z-(1-\eta)\right)^2}. \tag{44}$$

To compare the two transfer functions (43) and (44), we first need to define their ratio as

$$H(z) \triangleq \frac{H_{\text{2-step}}(z)}{H_{\text{1-step}}(z)} = \frac{\varepsilon_{\text{2-step}}(z)}{\varepsilon_{\text{1-step}}(z)} = \frac{z(z+1-\eta)(z-1+\beta)^2}{\left(z^2-(1-\beta)z-(1-\eta)\right)^2}. \tag{45}$$

Clearly for $\eta = 1$, the ratio in equation (45) is always 1, regardless of the value of β.

Since equation (45) is a stable rational system transfer function, its ROC includes the unit circle. Therefore, if we replace the complex

https://doi.org/10.25088/ComplexSystems.27.1.63

variable z in equation (45) with the complex quantity $e^{j\omega}$, then the z-transform reduces to the Fourier transform.

Next, we show that the two-step update scheme has a faster convergence rate in the transient and better steady-state accuracy compared to the original XCS. First, we need to show that the amplitude of the Fourier transform of equation (44) is greater than equation (43) or equivalently, the amplitude of the Fourier transform of equation (45) is greater than 1 over the low-frequency range, that is, ω around 0, and the high-frequency range, that is, ω around $\pm\pi$. For the low-frequency range, we have

$$\lim_{\omega\to 0}\left|H\left(e^{j\omega}\right)\right| = \lim_{z\to 1}|H(z)| = \frac{\beta^2(2-\eta)}{(\beta+\eta-1)^2} \geq 1, \tag{46}$$

where the inequality is satisfied, with the equality holding only for $\eta = 1$.

The inequality in equation (46) is satisfied because according to equation (27), $2-\eta \geq 1$ and

$$\begin{aligned} &0 \leq 1-\eta < \beta \leq 1 \Rightarrow 0 \leq \frac{1-\eta}{\beta} < 1 \Rightarrow 0 < 1 - \frac{1-\eta}{\beta} \leq 1 \\ &\Rightarrow \frac{\beta}{\beta+\eta-1} \geq 1 \Rightarrow \frac{\beta^2(2-\eta)}{(\beta+\eta-1)^2} \geq 1. \end{aligned} \tag{47}$$

Similarly, for the high-frequency range we have

$$\lim_{\omega\to\pm\pi}\left|H\left(e^{j\omega}\right)\right| = \lim_{z\to -1}|H(z)| = \frac{\eta(2-\beta)^2}{(1-\beta+\eta)^2} \geq 1, \tag{48}$$

where the inequality is satisfied, with the equality holding only for $\eta = 1$.

To show that the inequality in equation (48) is satisfied, we know from equation (27) that

$$0 \leq 1-\beta < \eta \text{ and } 1-\eta \geq 0 \text{ and } \eta^2 \leq \eta. \tag{49}$$

Then,

$$\begin{aligned} &(1-\beta)^2 < \eta^2 \leq \eta \\ &\Rightarrow (1-\eta)\left((1-\beta)^2 - \eta\right) \leq 0 \\ &\Rightarrow (1-\eta)(\beta^2 - 2\beta) + (1-\eta)^2 \leq 0 \\ &\Rightarrow 1 + \beta^2 + \eta^2 - 2\beta + 2\eta - 2\beta\eta \leq 4\eta - 4\eta\beta + \eta\beta^2 \\ &\Rightarrow (1-\beta+\eta)^2 \leq \eta(2-\beta)^2 \\ &\Rightarrow \frac{\eta(2-\beta)^2}{(1-\beta+\eta)^2} \geq 1. \end{aligned} \tag{50}$$

This concludes the proof of Lemma 2. □

The overall accuracy of the XCS model is a function of its individual rule's prediction error; therefore, the two-step update XCS algorithm has a faster convergence and better steady-state accuracy. Figure 1: (a, b) is the amplitude plot of the Fourier transform of equation (45); that is,

$$\left|H\left(e^{j\omega}\right)\right| = \frac{\left|H_{2\text{-step}}\left(e^{j\omega}\right)\right|}{\left|H_{1\text{-step}}\left(e^{j\omega}\right)\right|} = \left|\frac{z\left(z+1-\eta\right)\left(z-1+\beta\right)^2}{\left(z^2-\left(1-\beta\right)z-\left(1-\eta\right)\right)^2}\right|_{z=e^{j\omega}}, \tag{51}$$

for the range of ω between $-\pi$ and π for some values of η and β, which shows how the amplitudes vary over this frequency range. Figure 1(a) is magnified around unity amplitude and is plotted in Figure 1(b) for better illustration. As we have shown in Lemma 2, the two-step update scheme has a faster convergence rate over the high-frequency range, that is, ω around $\pm\pi$, and better steady-state accuracy over the low-frequency range, that is, ω around 0, as implied by $\left|H\left(e^{j\omega}\right)\right|$, which is greater than 1.

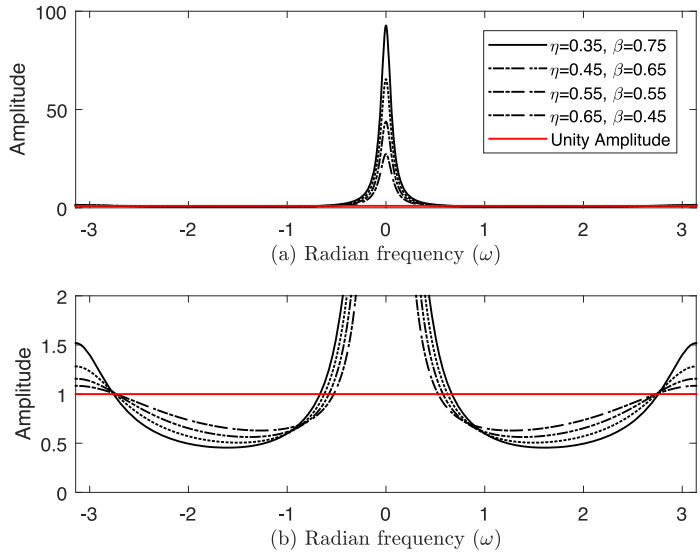

Figure 1. (a) $\left|H\left(e^{j\omega}\right)\right|$. (b) Magnified plot (a) around unity amplitude.

5. Numerical Results and Discussion

In this section, to investigate the effect of the proposed two-step update scheme compared to the original one-step update, the XCS algorithm is trained to solve the multiplexer problem. For this

purpose, three multiplexer problems, that is, 6-MUX, 11-MUX and 20-MUX, are solved. The common parameters used for all the problems are $\alpha = 0.1$, $\varepsilon_0 = 0.0001$, $\nu = 25$, $\mu = 0.04$, $R = 1000$, and the probability of specifying an attribute is equal to 0.5 [23]. Finally, the frequency of applying GA is 10, and tournament selection is used for reproduction. Population size (N) and crossover probability (χ) are specified for each experiment later in the problem. β and η are selected according to equation (27) to observe variations in the results.

Due to the stochastic nature of the algorithm, to have a reliable result, the model needs to be trained multiple times and accuracy averaged over trained models. We define the empirical mean $\overline{X}$ of m runs that forms a collection of m random variables $X_1, \ldots, X_m$, which are independent and identically distributed (i.i.d.) and bounded by the interval $[0, 1]$. The empirical mean is given as

$$\overline{X} = \frac{X_1 + \cdots + X_m}{m}. \tag{52}$$

If we choose

$$m \geq \frac{\ln \frac{2}{\lambda}}{2\sigma^2}, \tag{53}$$

through Hoeffding's inequality [25], then with probability at least $1 - \lambda$, the difference between the empirical mean $\overline{X}$ and the true mean $E[\overline{X}]$ is at most σ, where $E[\cdot]$ denotes the mathematical expected value of its argument. The σ says how far we are willing to allow the empirical mean to be from the true mean, and the λ says with what probability we are willing to allow a deviation larger than σ. For a given $\lambda = 0.10$ and $\sigma = 0.15$ in equation (53), we have $m \geq 66$; therefore, simulation results are averaged over 66 runs.

In the literature, the value of β is selected to be 0.2 [24, 26], yet it is worthwhile to investigate how the algorithm will behave with different learning rates. The accuracy is computed as the percentage of correct classifications:

$$\text{accuracy} = \frac{\text{number of correct classifications}}{\text{number of covered instances}}. \tag{54}$$

In the first experiment, three different combinations of β and η are selected and the training accuracy of the algorithm is plotted in Figure 2 for both update rules. As expected from the weighted sum of the current and previous values in the two-step update, the result shows that the two-step update rule is less sensitive to variations in learning rates β and η and has a faster convergence rate in the transient and steady state, while in the one-step approach, the training accuracy

deteriorates as β increases. Nonetheless, the final training accuracies are almost equal for both updates.

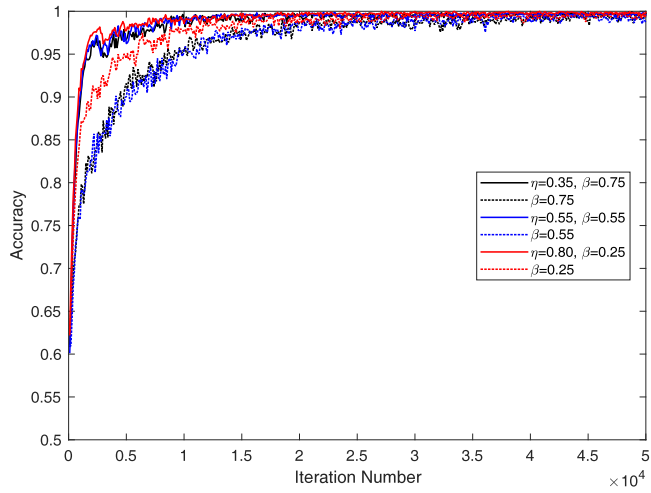

Figure 2. Comparison of learning accuracy for two-step (solid line) and one-step (dotted line) update rules for the 6-MUX problem.

To elaborate more on the effects of the proposed update rules, using information from one extra step back in time acts like a lag-lead compensator in conventional control theory, which helps to have faster convergence and better steady-state accuracy. In other words, adding memory will help to generate a better estimate of future classifier parameters specifically when the present values of parameters are experiencing undesired variations for partially correct classifiers. Partially correct classifiers refer to relatively general classifiers that can match more problem instances, which do not necessarily classify correctly.

Experiments on 11-MUX and 20-MUX problems as demonstrated in Figures 3 and 4 also confirm the improvement in the rate at which the algorithm learns the problem. The results show that the two-step update rule is less sensitive to variations in β and η, has a faster convergence rate in the transient and steady state, and achieves better training accuracy specifically for the 20-MUX problem. The only exception in these results corresponds to $\beta = 0.25$ for the one-step update in the 11-MUX problem, which demonstrates much faster convergence compared to its respective two-step update.

According to Lemma 2, the two-step update scheme is expected to improve the performance of the prediction error in the early stage of learning and the steady state. However, it seems unfeasible to find an

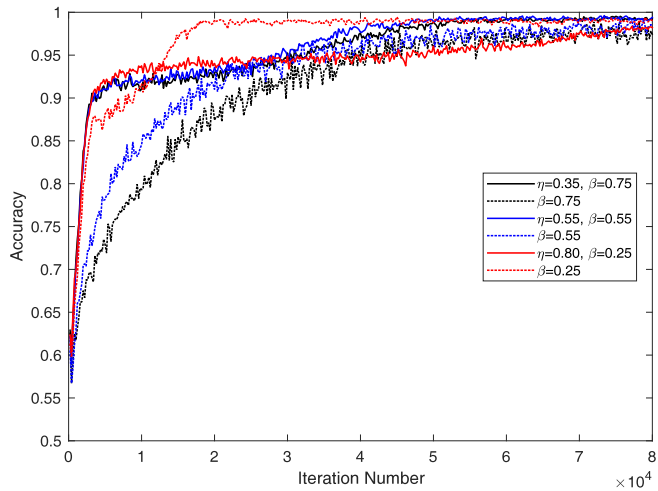

Figure 3. Comparison of learning accuracy for two-step (solid line) and one-step (dotted line) update rules for the 11-MUX problem.

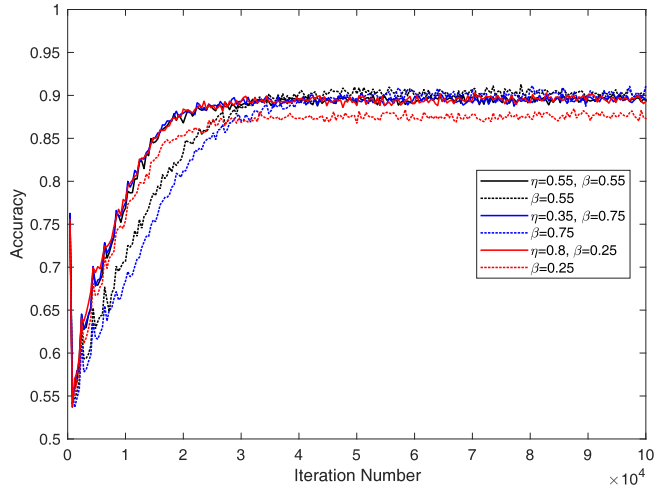

Figure 4. Comparison of learning accuracy for two-step (solid line) and one-step (dotted line) update rules for the 20-MUX problem. $N = 4000$, $\chi = 0.95$, 2000 samples out of 2^{20} environment size.

exact mapping between the frequency domain and time domain for a learning algorithm. This becomes even worse when the performance evaluation is problem independent—in other words, no complexity measure of the problem is considered in the performance analysis. The 6-MUX problem is less complex compared to 11-MUX and

20-MUX problems with more inputs; as a result its performance has improved over the entire learning time. On the other hand, in 11-MUX, the problem complexity influences the expected improvement such that the effective range (namely low-frequency and high-frequency ranges) of the proposed update is different than the 6-MUX problem. More specifically, for the case of $\eta = 0.80$, $\beta = 0.25$, the effect of the two-step update rule vanishes faster than other combinations of the learning rates. Therefore, the one-step update scheme outperforms the proposed update rule. The 20-MUX problem behaves as expected for all combinations of learning rates and the entire learning time.

According to the experiments, the fact that the algorithm tends to converge faster suggests the idea that the proposed method relatively supports those classifiers with correct actions that tend to be accurate in two consecutive appearances in the action set. This is an extra pressure toward evolving accurate solutions.

6. Conclusions and Future Work

In this work, a novel two-step Markov update scheme for XCS classifier systems and mathematical representations for rules to update the classifier parameters using discrete-time dynamical systems theory are presented. The stability and convergence of the two-step update rules are analyzed. The frequency domain analysis for classifier parameters has been provided. We also presented the robustness and performance improvement of the XCS algorithm employing the two-step update rule using numerical experiments. The implementation cost includes adding four float-type variables for each macro classifier and some arithmetic calculations, while its computational complexity and algorithm run time are almost the same. We have shown that using a non-Markov update contributes to faster learning and more accurate training at steady state.

Thus far, our analysis has been carried out independently of problem size and generality of classifiers, which may substantially impact XCS performance. In future work, the authors will address these issues and investigate their effects on algorithm dynamics. The performance of the algorithm will also be studied with real-world problems.

Acknowledgments

This paper is based on research sponsored by the Air Force Research Laboratory and the Office of the Secretary of Defense (OSD) under agreement number FA8750-15-2-0116. The US government is

authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory and the OSD or the US government. The authors would like to thank the Air Force Research Laboratory and the OSD.

References

[1] J. H. Holland, "Adaptation," *Progress in Theoretical Biology*, Vol. 4 (F. M. Snell and R. Rosen, eds.), New York: Academic Press, Inc., 1976 pp. 263–293. doi:10.1016/B978-0-12-543104-0.50012-3.

[2] S. W. Wilson, "Classifier Fitness Based on Accuracy," *Evolutionary Computation*, **3**(2), 1995 pp. 149–175. doi:10.1162/evco.1995.3.2.149.

[3] C. J. C. H. Watkins and P. Dayan, "Q-Learning," *Machine Learning*, 8(3–4), 1992 pp. 279–292. doi:10.1007/BF00992698.

[4] P. L. Lanzi, "Mining Interesting Knowledge from Data with the XCS Classifier System," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)* (L. Spector, ed.), San Francisco: Morgan Kaufmann Publishers, 2001 pp. 958–965.

[5] S. W. Wilson, "Mining Oblique Data with XCS," in *Advances in Learning Classifier Systems (IWLCS 2000)* (P. L. Lanzi, W. Stolzmann and S. W. Wilson, eds.), Berlin, Heidelberg: Springer, 2001 pp. 158–174. doi:10.1007/3-540-44640-0_ 11.

[6] E. Bernadó-Mansilla and J. M. Garrell-Guiu, "Accuracy-Based Learning Classifier Systems: Models, Analysis and Applications to Classification Tasks," *Evolutionary Computation*, **11**(3), 2003 pp. 209–238. doi:10.1162/106365603322365289.

[7] R. Urbanowicz, N. Ramanand and J. Moore, "Continuous Endpoint Data Mining with ExSTraCS: A Supervised Learning Classifier System," in *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, Madrid, Spain, New York: ACM, 2015 pp. 1029–1036. doi:10.1145/2739482.2768453.

[8] K. Tamee, L. Bull and O. Pinngern, "Towards Clustering with XCS," in *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, London, England, New York: ACM, 2007 pp. 1854–1860. doi:10.1145/1276958.1277326.

[9] W. Stolzmann, "An Introduction to Anticipatory Classifier Systems," in *International Workshop on Learning Classifier Systems (IWLCS 1999)*, Berlin, Heidelberg: Springer-Verlag, 2000 pp. 175–194. doi:10.1007/3-540-45027-0_ 9.

[10] M. V. Butz, P. L. Lanzi and S. W. Wilson, "Function Approximation with XCS: Hyperellipsoidal Conditions, Recursive Least Squares, and Compaction," *IEEE Transactions on Evolutionary Computation*, **12**(3), 2008 pp. 355–376. doi:10.1109/TEVC.2007.903551.

[11] S. W. Wilson, "Classifiers That Approximate Functions," *Natural Computing*, **1**(2–3), 2002 pp. 211–234. doi:10.1023/A:1016535925043.

[12] C. Stone and L. Bull, "For Real! XCS with Continuous-Valued Inputs," *Evolutionary Computation*, **11**(3), 2003 pp. 299–336. doi:10.1162/106365603322365315.

[13] J. Bacardit and N. Krasnogor, "A Mixed Discrete-Continuous Attribute List Representation for Large Scale Classification Domains," in *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (GECCO '09)*, Montreal, New York: ACM, 2009 pp. 1155–1162. doi:10.1145/1569901.1570057.

[14] A. Workineh and A. Homaifar, "Robust Bidding in Learning Classifier Systems Using Loan and Bid History," *Complex Systems*, **19**(3), 2011 pp. 287–303. www.complex-systems.com/pdf/19-3-6.pdf.

[15] S. Nazmi, M. Razeghi-Jahromi and A. Homaifar, "Multi-label Classification with Weighted Labels Using Learning Classifier Systems," in *Proceedings of the 16th IEEE International Conference on Machine Learning and Applications (ICMLA 2017)*, Cancun, Mexico (X. Chen, B. Luo, F. Luo, V. Palade and M. A. Wani, eds.), Piscataway, NJ: IEEE, 2017.

[16] P. L. Lanzi, "Adding Memory to XCS," in *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence*, Anchorage, AK, Piscataway, NJ: IEEE, 1998 pp. 609–614.

[17] P. L. Lanzi and S. W. Wilson, "Toward Optimal Classifier System Performance in Non-Markov Environments," *Evolutionary Computation*, **8**(4), 2000 pp. 393–418. doi:10.1162/106365600568239.

[18] M. V. Butz and S. W. Wilson, "An Algorithmic Description of XCS," in *International Workshop on Learning Classifier Systems (IWLCS 2000)* (P. L. Lanzi, W. Stolzmann and S. W. Wilson, eds.), Berlin, Heidelberg: Springer, 2000 pp. 253–272. doi:10.1007/3-540-44640-0_ 15.

[19] S. W. Wilson, "Generalization in the XCS Classifier System," in *Genetic Programming 1998: Proceedings of the Third Annual Conference*, Madison, WI (J. Koza, et al., eds.), San Francisco: Morgan Kaufmann, 1998 pp. 665–674.

[20] H. K. Khalil, *Nonlinear Systems*, 3rd ed., Englewood Cliffs, NJ: Prentice Hall, 2002.

[21] K. Ogata, *Discrete-Time Control Systems*, 2nd ed., Upper Saddle River, NJ: Prentice Hall, 1995.

[22] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 3rd ed., Upper Saddle River, NJ: Prentice-Hall, 2010.

[23] 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*, 8(1), 2004 pp. 28–46. doi:10.1109/TEVC.2003.818194.

[24] M. V. Butz, D. E. Goldberg, P. L. Lanzi and K. Sastry, *Bounding the Population Size to Ensure Niche Support in XCS*, IlliGAL Report No. 2004033, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, 2004.

[25] H. Dehling and W. Philipp, "Empirical Process Techniques for Dependent Data," *Empirical Process Techniques for Dependent Data* (H. Dehling, T. Mikosch and M. Sørensen, eds.), Boston: Birkhäuser, 2012. doi:10.1007/978-1-4612-0099-4_ 1.

[26] A. Orriols-Puig and E. Bernadó-Mansilla, "A Further Look at UCS Classifier System," in *The Genetic and Evolutionary Computation Conference (GECCO '06)*, Seattle, New York: ACM, 2006 pp. 8–12.