TR-NIS-0002

Convergence and Generalization in Learning Classifier Systems: ZCS with Residual Gradient Algorithms

Atsushi WADA, Keiki TAKADAMA, Katsunori SHIMOHARA, Osamu KATAI

2004.4.7

国際電気通信基礎技術研究所 ネットワーク情報学研究所
〒619-0288 「けいはんな学研都市」光台二丁目2番地2
Tel: 0774-95-2641 Fax: 0774-95-2647

Advanced Telecommunications Research Institute International (ATR)
Network Informatics Laboratories
2-2-2, Hikaridai, "Keihanna Science City", Kyoto 619-0288, Japan
Tel: +81-774-95-1111 Fax: +81-774-95-2647

Convergence and Generalization in Learning Classifier Systems: ZCS with Residual Gradient Algorithms

Atsushi Wada WADA@ATR.JP
ATR Human Information Science Laboratories
2-2-2 Hikaridai, "Keihanna Science City" Kyoto 619-0288, Japan

Keiki Takadama KEIKI@DIS.TITECH.AC.JP
Tokyo Institute of Technology,
Interdisciplinary Graduate School of Science and Engineering
4259 Nagatsuta-cho, Midori-ku, Yokohama, 226-8503, Japan

Katsunori Shimohara KATSU@ATR.JP
ATR Human Information Science Laboratories
2-2-2 Hikaridai, "Keihanna Science City" Kyoto 619-0288, Japan

Osamu Katai KATAI@I.KYOTO-U.AC.JP
Graduate School of Informatics, Kyoto University
Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501, Japan

Abstract

Learning Classifier Systems (LCSs) are rule-based systems possessing essential functions of (a) reinforcement learning, (b) state generalization and (c) rule discovery. As the first step toward developing a theoretical basis of LCSs, here we focus on a strong relation between LCSs' learning process with generalization and reinforcement learning methods with function approximations, and aim at introducing a proof of convergence for LCSs. Based on our previous work, which showed an equivalence of learning processes between a zeroth-level classifier system (ZCS) and Q-learning with linear function approximation, in this paper, we apply a residual gradient algorithm for Q-learning to ZCS. As for the result, we obtained an LCS with generalization ability that guarantees convergence due to the proof of the residual gradient algorithm under the condition of its rule discovery process being suppressed.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

1. Introduction

Learning Classifier Systems (LCSs) are rule-based systems whose rules are named *classifiers*. The original LCS was firstly introduced by Holland (Holland, 1975), intended as a framework to study learning in condition-action rules. It included distinctive features of a *generalization* mechanism in rule conditions and *rule discovery* mechanism using genetic algorithms (GAs) (Goldberg, 1989). Later, this original LCS was revised to the "standard form" in (Holland, 1980) and brought about many variants (Booker, 1998; Riolo, 1991; Smith et al., 2000; Wilson, 1995).

Although LCSs were mainly developed in the field of evolutionary computation, they include the concept of *credit assignment*, which actually has an essential connection with reinforcement learning methods, especially temporal difference (TD) methods (Sutton, 1988). The *bucket brigade* algorithm (Holland, 1986), a well-known credit assignment mechanism for LCS, is quite similar to the Sarsa (Sutton, 1996) algorithm. Wilson proposed two types of classifier systems, ZCS and XCS with a Q-learning (Watkins, 1989)-like algorithm named the *Q-bucket brigade* (QBB) algorithm, in their learning processes.

Despite of these essential similarities, when focusing on theoretical aspects, LCSs do not have any convergence proof for learning, whereas Sarsa and Q-learing hold

the convergence theorem with the required condition clarified (Singh et al., 2000; Watkins & Dayan, 1992).

This derives from the most distinct feature of LCSs: possessing a rule discovery mechanism that uses GA, which makes it difficult to directly applying convergence proof performed in reinforcement learning researches.

Consequently, is it possible to partially introduce a convergence theorem to an LCS by suppressing the rule discovery process but possibly keeping the other distinct property of generalization? Answering this question by proposing such an LCS is the objective of this paper, which we regard as a first step to strengthen the theoretical basis of LCS, finally aiming at an LCS with a complete convergence theorem.

Based on our previous work that showed equivalence between the learning process of the ZCS classifier system and Q-learning from the aspect of the function approximation method, we apply it to ZCS *residual gradient algorithms* (Baird, 1999), alternative algorithms for updating values, which will introduce proof of convergence to several classes of function approximations for a variety of reinforcement learning methods. This should result in an LCS having convergence proof due to a residual gradient algorithm while retaining its generalization ability.

The rest of the paper is organized as follows. Section 2 introduces related researches. Section 3 describes the learning process of ZCS from viewpoint of Q-learning with function approximation, which is required for Section 4 where residual gradient algorithms are applied to ZCS. Finally, Section 5 gives our conclusions.

2. Related Research

Because LCSs are essentialy connected with TD methods such as Q-learning, some works comparing both of them have been carried out.

Dorigo et al. compared the originally designed VSCS (Very Simple Classifier System) with Q-learning and showed that the learning process would be equivalent under the limitation of VSCS having neither generalization ability nor creation and deletion of classifiers (Dorigo & Bersini, 1994).

In a general study to compare LCS and Q-learning, Lanzi implemented LCS by starting from simple Q-learning, extending it to a rule-based model, adding a reinforcement learning mechanism, and finally adding generalization ability (Lanzi, 2002). However, the objective of this work is not to show equivalence with Q-learning nor to prove convergence of learning in particular. Consequently, the final description of LCS having generalization is in a form incompatible with Q-learning.

We regard these works as achieving our objective only to a minor extent, as the convergence theorem for Q-learning can be directly applied when the models have neither the ability to generalize nor invoke the rule discovery mechanism. However, these results are quite limited, because most existing LCSs do allow generalization of classifier conditions.

Here, we focus on our previous work that built a framework capable of dealing with the generalization ability of LCSs by showing equivalence between the learning process of the ZCS classifier system and Q-learning with the *function approximation* method, a method that enables generalization of reinforcement learning methods. In this work, we also mentioned that the class of generalization in ZCS corresponds to linear approximation.

Based on this work, we introduce proof of convergence to ZCS under the condition that its rule discovery process is suppressed, but its generalization ability is retained. As the leaning process of ZCS is shown to be equivalent to Q-learning with function approximation, we can use the contributions in a reinforcement learning field, which introduces proof of convergence to Q-learning with linear function approximation. As we already proposed in the introduction, in this work we focus on residual gradient algorithms, an alternative algorithm of updating values, which will introduce proof of convergence to several classes of function approximations for many reinforcement learning methods. We will apply a residual gradient algorithm for Q-learning in order to achieve our objective.

3. Learning Processes of ZCS from viewpoint of Q-learning with Function Approximation

In this section, we describe the learning processes of ZCS from the viewpoint of Q-learning with function approximation based on our previous work, which is necessary for the next step to introduce residual gradient algorithms to ZCS. First, we introduce the ZCS learning process in detail. Next, we explain Q-learning with function approximation. Finally, we presents the ZCS learning process in the form of Q-learning with function approximation.

3.1. Learning process in ZCS

ZCS (Zeroth-level Classifier System) is an LCS model introduced by Wilson with a simple architecture that implements distinctive features of LCS, functions as a rule-based model, adopts a learning process, uses rule representation with the ability to state generalizations, and features a rule discovery process, providing dynamic rule creation and deletion through learning. To focus on the learning process of ZCS, here we only describe a learning process of classifier strength to keep our discussion general. See (Wilson, 1994) for a more detailed description of the model, including the rule discovery process.

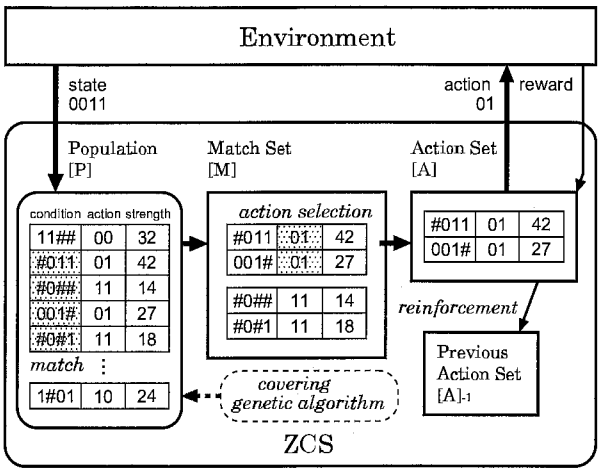

Figure 1. System architecture of ZCS, which shows the process flow from ZCS receiving an input state from the environment to ZCS obtaining a reward from the environment by performing an action.

Figure 1 shows how ZCS operates by following the flow of processes. The basic component of ZCS is a set of rules, named classifiers. Each classifier comprises three parts: condition, action and strength. ZCS maintains a set of classifiers named population [P], and when the input state arrives, the condition part of each classifier in [P] is matched with the state. All classifiers whose conditions match the specified state are collected to organize a *match set* [M]. For example, in Fig. 1, four classifiers whose conditions match the input state "0011" are collected to organize a match set. Here, we do not describe the representation of classifier condition and its matching definition in detail, though each condition of all the classifiers in the match set matches at least one state, including "0011." Next, an action is selected from among those advocated by members of [M]. Many action selection schemes are possible, for example roulette-wheel selection, which selects the action stochastically by the proportional probability based on the total strength of the classifiers whose action part is the same. After action selection, an *action set* [A] is formed from classifiers in the match set [M] that have the same action as the selected action. For example, in Fig. 1, two classifiers whose action is "01" are collected to organize an action set. The action set [A] is preserved to the next time step but renamed as $[A]_{-1}$ while the new action set [A] is formed. The learning process occurs when ZCS obtains an immediate reward, which has the following result:

$$S_{[A]_{-1}} \longleftarrow (1-\beta)S_{[A]_{-1}} + \beta\left[r + \gamma S_{[A]}\right]. \qquad (1)$$

The expression $S_{[A]}$ denotes the total value of the strength of all classifiers in action set [A], and the left arrow in the formula denotes the operation to set the value of the right-hand side to the left-hand side. Parameter β denotes the *learning rate*, which controls the flexibility of learning. In this case, each classifier in the previous action set $[A]_{-1}$ is added to the value of the right-hand side, equally divided by the number of classifiers in previous action set $[A]_{-1}$. Wilson also proposed an alternative update formula quite similar to the Q-learning learning process:

$$S_{[A]_{-1}} \longleftarrow (1-\beta)S_{[A]_{-1}} + \beta\left[r + \gamma \max_a S_{[M]|_a}\right], \qquad (2)$$

where $[M]|_a$ denotes the set of classifiers included in match set [M] having action a. In (Wilson, 1994), Wilson discussed the relevance between ZCS and Q-learning based on the formula above, which is named Q-bucket brigade (QBB) algorithm. However, this macroscopic viewpoint only deals with the aggregated value of the classifier strengths. Such a viewpoint lacks the ability of a microscopic view to relate ZCS to Q-learning at the level of each classifier strength reinforcement, which is the topic discussed in the next subsection.

3.2. Q-learning with function approximation method

Q-learning is a popular learning model to solve online reinforcement learning problems. The name Q denotes *action value function* $Q(s,a)$, which estimates the *action value* for taking action a in state s defined as an expectation of total future rewards. In Q-learning, Q values are updated at each time step as defined below, where s_t, a_t, r_t and Q_t denote agent's state, action, received reward, and Q values at time step t, respectively:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha\left[v_t - Q_t(s_t, a_t)\right], \quad (3)$$

where v_t is a target value for the update of $Q_t(s_t, a_t)$. In Q-learning, v_t is defined as

$$v_t = r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a). \quad (4)$$

Parameter α denotes the *learning rate*, and this controls the flexibility of learning. Parameter γ denotes a *discount factor* for determining present value of future rewards, which is also important for avoiding the divergence of action values. The action value function $Q(s, a)$ is called a Q-table, since it holds action values for all combinations of states and actions represented as $\mathcal{S} \times \mathcal{A}$, where $\mathcal{S}$ and $\mathcal{A}$ are sets of all possible states and actions. This causes the serious problem of state-space explosion when the number of dimensions of the states becomes too large.

To avoid this problem, a function approximation method can be applied to compress a Q-table with a large number of states by approximating it with a small number of parameters. Instead of updating a single cell in a Q-table, these parameters are updated using the gradient-descent method described as follows. Let $\boldsymbol{\theta_t} = (\theta_t(1), \theta_t(2), ..., \theta_t(n))^T$ ("T" here denotes transpose) approximate the action value function, where $Q_t(s, a)$ is a smooth differentiable function of $\boldsymbol{\theta_t}$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$. The gradient-descent methods update $Q_t(s, a)$ by adjusting the parameter vector as the following formulas,

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \Delta\boldsymbol{\theta}_t, \quad (5)$$

$$\Delta\boldsymbol{\theta}_t = \alpha [v_t - Q_t(s_t, a_t)] \nabla_{\boldsymbol{\theta}_t} Q_t(s_t, a_t). \quad (6)$$

Here, gradient ∇_{θ_t} for a function f is defined as follows.

$$\nabla_{\theta_t} f(\boldsymbol{\theta_t}) = \left(\frac{\partial f(\boldsymbol{\theta_t})}{\partial \theta_t(1)}, \frac{\partial f(\boldsymbol{\theta_t})}{\partial \theta_t(2)}, ..., \frac{\partial f(\boldsymbol{\theta_t})}{\partial \theta_t(n)} \right)^T. \quad (7)$$

Especially if Q_t is linear to each parameter in the parameter vector $\boldsymbol{\theta_t}$, Q_t can be expressed as a product of parameter vector $\boldsymbol{\theta_t}$ and feature vector $\boldsymbol{\phi}_{sa}$ independent of $\boldsymbol{\theta_t}$. This type of function approximation is called a linear function approximation.

$$Q_t(s, a) = \sum_i \theta_t(i) \phi_{sa}(i) = \boldsymbol{\theta}_t^T \boldsymbol{\phi}_{sa}. \quad (8)$$

Simple Q-learning using a Q-table can be described as a special case of linear function approximation in which the parameter vector is composed by listing all of the Q-values in the Q-table in a row.

3.3. Relate ZCS and Q-learning with function approximation

In this section, we show that the ZCS learning process can also be derived by applying the updating formula of the function approximation method in Q-learning, which can be accomplished by focusing on the generalization mechanisms. This approach allows us to prove the equivalence of the learning processes between ZCS and Q-learning with function approximation.

3.3.1. Notation

Some notations are introduced here for the subsequent analysis. The sets P_t, M_t and A_t denote classifier population, match set, and action set at time step t, respectively, while regarding classifiers as their elements. Let cl_i be a classifier, and then each of the three parts composing the classifier are labeled $cl_i.condition \in \mathcal{C}$, $cl_i.action \in \mathcal{A}$, and $cl_i.strength \in \mathcal{R}$, where the set $\mathcal{C}$ denotes a set of all possible condition expressions allowed under the classifier representation. To keep our discussion general, we define $\mathcal{C}$ as a set of all possible combination of states $\mathcal{S}$ and regard a condition $c \in \mathcal{C}$ as a set of states that can be matched with the condition. Function $cl_i.match(s, a)$ for each classifier cl_i is defined for $s \in \mathcal{S}$, $a \in \mathcal{A}$, which returns 1 when $s \in cl_i.condition$ and $a = cl_i.action$, otherwise returns 0[1].

3.3.2. Strength Update described using Function Approximation rule

In ZCS, action selection is based on the total strength of the classifiers for each action $a \in \mathcal{A}$ in the match set M_t, which matches the state s_t. Here, we assume that the total strength of the classifiers for action a in the match set for state s_t represents the approximation of the corresponding Q-value $Q_t(s_t, a)$, which gives the formula

$$Q_t(s, a) = \sum_{cl \in P_t} cl.strength \times cl.match(s, a) \quad (9)$$

$$= \begin{bmatrix} cl_1.strength \\ \vdots \\ cl_n.strength \end{bmatrix}^T \cdot \begin{bmatrix} cl_1.match(s, a) \\ \vdots \\ cl_n.match(s, a) \end{bmatrix} \quad (10)$$

$$= \boldsymbol{\theta}_t{}^T \cdot \boldsymbol{\phi}_{sa}. \quad (11)$$

[1] To keep the discussion general, here we do not give the detailed relation between classifier condition representation and $\mathcal{C}$. By defining this relation appropriately, the following discussions can be applied to several classifier representations, such as the real-valued representations used in the XCSR classifier system (Wilson, 2000).

Since the state-action pair (s_t, a_t) may take any combination within the set $\mathcal{S}\times\mathcal{A}$, we can say that the Q-table $Q_t(s_t, a_t)$ is represented by the set of classifiers existing in the classifier population P_t, with each classifier cl having the strength value $cl.strengh$ and featured with $cl.condition$. This discussion follows from the above assumption and makes it seem natural that the set of classifiers in the classifier population approximates the Q-table by means of the function approximation method for Q-learning.

To prove this understanding, we should apply the updating Formula 6 of the function approximation method by using the Q-value function defined as Formula 11 and compare the result with the original ZCS update process. Let $\boldsymbol{\theta_t}$ be a parameter vector composed of the set of strength values $cl_i.strength$ for all classifiers in classifier population $P_t = \{cl_1, cl_2, ..., cl_n\}$, where n denotes the total number of classifiers in P_t. Then let $\boldsymbol{\phi}_{s_t a_t}$ be a feature vector defined as $(cl_1.match(s_t, a_t), ..., cl_n.match(s_t, a_t))^T$. These two definitions transform Formula 11 into the form of a linear function approximation like Formula 8. By calculating $\nabla_{\boldsymbol{\theta}_t} Q_t(s_t, a_t)$, we obtain

$$\begin{aligned}\nabla_{\theta_t} Q_t(s_t, a_t) \\ &= \left(\frac{\partial Q_t(s_t, a_t)}{\partial \theta_t(1)}, ..., \frac{\partial Q_t(s_t, a_t)}{\partial \theta_t(n)}\right)^T \quad (12)\\ &= \boldsymbol{\phi}_{s_t a_t}. \quad (13)\end{aligned}$$

Using this result, we finally obtain the updating formula

$$\begin{aligned}\boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \alpha\Big[\Big(r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a)\Big) \\ &\quad - Q_t(s_t, a_t)\Big]\boldsymbol{\phi}_{s_t a_t}. \quad (14)\end{aligned}$$

This formula corresponds to the process of updating the strength of the classifiers whose conditions match the state s_t and have the same action a maximizing $Q_t(s_{t+1}, a)$, while the other classifiers' strengths are not modified. Here, if we set the value of α to $\beta/|A|$, where $|A|$ denotes the number of classifiers to be updated, this update process would be equivalent to that of ZCS.

3.3.3. Conditions for the equivalence

Next, we clarify conditions for equivalence between the learning processes of Q-learning with function approximation and ZCS. As Q-learning does not feature any dynamic mechanism to change the structure of the function approximator, any operation to change the composition of the classifier population [P] will break the equivalence of learning processes between Q-learning and ZCS. This limitation is quite strong for ZCS, since it cuts off the most distinctive feature of dynamic classifier creation and deletion. However, it still assures the equivalence in the time periods between the invocations of such operations.

4. Applying residual gradient algorithms to ZCS

From the analytical result in the previous section, the convergence of the learning process in ZCS can be discussed under the condition described in Section 3.3.3. In general, simple Q-learning using a Q-table receives the benefit of the convergence theorem proving convergence to a probability of 1 under the condition that learning rate α decreases appropriately (Watkins, 1989). However, in the case of Q-learning with the function approximation method, the applicability of the convergence theorem depends on the class of the approximation function. For some special cases in the class of linear function approximation functions, such as state aggregation, the convergence theorem has already been proved (Gordon, 1995; Singh et al., 1995; Tsitsiklis & Roy, 1996). However, in the case of linear function approximation, Baird presented some counterexamples showing that the value function diverged as the learning proceeded.

The learning process in ZCS can be viewed as one of Q-learning with the function approximation method, which uses Formula 11 as its approximation function and is included in a class of linear function approximations. Accordingly, the stability of ZCS cannot be proved, even under the rule discovery process being suppressed.

This limitation can be avoided by applying *residual gradient algorithms* (Baird, 1999), an alternative algorithm of updating values proposed by Baird, which will introduce proof of convergence to several classes of function approximations for many reinforcement learning methods, including Q-learning.

4.1. Residual gradient algorithms for Q-learning

Residual gradient algorithms are modifications of gradient descent algorithms for updating parameters used with function approximation methods for reinforcement learning. They were originally proposed by Baird (Baird, 1995) to avoid the limit of gradient descent method by proposing several counter examples showing the unstability in function approximation methods applied to off-policy TD control methods such as Q-learning with linear and non-linear approximation

classes.

By using the target value v_t defined as Formula 4, the residual gradient algorithm for Q-learning is described as an update formula,

$$\Delta\boldsymbol{\theta}_t = \alpha\left[v_t - Q_t(s_t, a_t)\right]\left[\boldsymbol{\phi}_{s_t a_t} - \gamma \max_a \boldsymbol{\phi}_{s'a}\right] \quad (15)$$

$$= \Delta_1\boldsymbol{\theta}_t + \Delta_2\boldsymbol{\theta}_t, \quad (16)$$

which is divided into two delta values $\Delta_1\boldsymbol{\theta}_t$ and $\Delta_2\boldsymbol{\theta}_t$ defined as,

$$\Delta_1\boldsymbol{\theta}_t = \alpha\left[v_t - Q_t(s_t, a_t)\right]\boldsymbol{\phi}_{s_t a_t} \quad (17)$$

$$\Delta_2\boldsymbol{\theta}_t = -\alpha\gamma\left[v_t - Q_t(s_t, a_t)\right]\max_a \boldsymbol{\phi}_{s'a}. \quad (18)$$

The difference between this update formula and the original update described in Formula 6 is the additional delta value $\Delta_2\boldsymbol{\theta}_t$, which modifies the values of parameters concerned with $\max_a Q_t(s', a)$.

4.2. ZCS with residual gradient algorithms

To apply residual gradient algorithm for Q-learning, the additional delta value for modifying the parameters concerned with $\max_a Q_t(s', a)$ must be introduced to ZCS. We modify the update process of ZCS by adding this process to result in an algorithmic description, as shown in Figure 2. It is mostly the same as the original ZCS; however, it differs where the set of classifiers $[A']$ representing the maximum aggregated value for an action a' are updated from procedure 23 to 25 in Fig. 2.

By suppressing the rule discovery process of procedures 5, 15 and 26, which is same with the condition we mentioned for equivalence between ZCS and Q-learning in Section 3.3.3, this algorithm will become equivalent with with Q-learning with residual gradient algorithm.

Residual gradient algorithm for Q-learning requires the learning rate at time step t, which is denoted as α_t to satisfy the formula,

$$\sum_{t=0}^{\infty}\alpha_t = \infty \quad \text{and} \quad \sum_{t=0}^{\infty}\alpha_t^2 < \infty, \quad (19)$$

to assure convergence with probability 1. In the case of ZCS, the learning rate α corresponds to $\beta/|A|$. As the number of classifiers in the action set is greater or equal than 1, and less or equal than $|P|$, the total number of classifiers in the population, the learning rate is in the range of $1/|P| \leq \beta/|A| \leq \beta$. Accordingly, when β is set appropriately, for example $\beta_t = 1/t$, the learning rate $\alpha_t = \beta_t/|A|$ satisfies the requirement of Formula 19 for convergence with probability 1.

1 Initialize $[P]$
2 Repeat (for each episode):
3 $s \leftarrow$ initial state of episode
4 $[M] \leftarrow \{i \in [P] \mid s \in cl_i.condition\}$
5 Invoke COVERING until $[M]$ satisfies covering condition
6 For all $a \in \mathcal{A}(s)$:
7 $[M]|_a \leftarrow \{i \in [M] \mid cl_i.action = a\}$
8 $payoff_a \leftarrow \sum_{i\in[M]|_a} cl_i.strength$
9 Repeat (for each step of episode):
10 Choose action $a \in \mathcal{A}(s)$ using policy derived from $payoff_a$ (e.g., ϵ-greedy)
11 $[A] \leftarrow \{i \in [M] \mid cl_i.action = a\}$
12 Take action a, observer reward r, and next state s'
13 $\delta \leftarrow r - payoff_a$
14 $[M] \leftarrow \{i \in [P] \mid s' \in cl_i.condition\}$
15 Invoke COVERING until $[M]$ satisfies covering condition
16 For all $a \in \mathcal{A}(s')$:
17 $[M]|_a \leftarrow \{i \in [M] \mid cl_i.action = a\}$
18 $payoff_a \leftarrow \sum_{i\in[M]|_a} cl_i.strength$
19 $a' \leftarrow \arg\max_a payoff_a$
20 $\delta \leftarrow \delta + \gamma payoff_{a'}$
21 For all i in $[A]$
22 $cl_i.strength \leftarrow cl_i.strength + \beta\delta/|A|$
23 $[A'] \leftarrow \{i \in [M] \mid cl_i.action = a'\}$
24 For all i in $[A']$
25 $cl_i.strength \leftarrow cl_i.strength - \beta\delta\gamma/|A|$
26 Invoke GENETIC ALGORITHM in probability $\rho/2$
27 until s' is terminal

Figure 2. Residual gradient algorithm for Q-learning applied to ZCS. The process of updating classifier strength is concerned with the target value is added.

5. Conclusion

In this paper, as a first step towards developing a theoretical basis of LCSs, we aimed at introducing a proof of convergence for LCSs. Based on our previous work, which showed an equivalence of learning processes between a zeroth-level classifier system (ZCS) and Q-learning with linear function approximation, we applied a residual gradient algorithm for Q-learning to ZCS. As for the result, we obtained an LCS with generalization ability, which guarantees convergence due to the proof of the residual gradient algorithm under the condition of its rule discovery process being suppressed. In future works, the rule discovery process should also be analyzed, eventually aiming at an LCS with a complete convergence theorem.

Acknowledgements

The research reported here was supported in part by a contract with the Telecommunications Advancement Organization of Japan entitled "Research on Human Communication," and the Okawa Foundation for Information and Telecommunications.

References

Baird, L. C. (1995). Residual algorithms: Reinforcement learning with function approximation. *International Conference on Machine Learning* (pp. 30–37).

Baird, L. C. (1999). *Reinforcement learning through gradient descent.* Doctoral dissertation, Carnegie Mellon University, Pittsburgh, PA 15213.

Booker, L. B. (1998). *Learning classifier systems*, chapter Do We Really Need to Estimate Rule Utilities in Classifier Systems?, 125–142. Springer.

Dorigo, M., & Bersini, H. (1994). A comparison of Q-learning and classifier systems. *Proceedings of From Animals to Animats, Third International Conference on Simulation of Adaptive Behavior.*

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning.* MA.: Addison-Wesley.

Gordon, G. J. (1995). Stable function approximation in dynamic programming. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 261–268). San Francisco, CA: Morgan Kaufmann.

Holland, J. H. (1975). *Adaptation in natural and artifical systems.* Michigan: The University of Michigan Press.

Holland, J. H. (1980). Adaptive algorithms for discovering and using general patterns in growing knowledge bases. *International Journal for Policy Analysis and Information Systems, 4*, 245–268.

Holland, J. H. (1986). Escaping brittleness: the possibilities of general-purpose. *Machine Learning, an artificial intelligence approach, 2.*

Lanzi, P. L. (2002). Learning classifier systems from a reinforcement learning perspective. *Soft Computing, 6*, 162–170.

Riolo, R. L. (1991). Lookahead planning and latent learning in a classifier system. *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior* (pp. 316–326).

Singh, S. P., Jaakkola, T., & Jordan, M. I. (1995). Reinforcement learning with soft state aggregation. *Advances in Neural Information Processing Systems* (pp. 361–368). The MIT Press.

Singh, S. P., Jaakkola, T., Littman, M. L., & Szepesvári, C. (2000). Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning, 38*, 287–308.

Smith, R. E., Dike, B. A., Ravichandran, B., El-Fallah, A., & Mehra, R. K. (2000). The fighter aircraft LCS: A case of different LCS goals and techniques. *Lecture Notes in Computer Science, 1813*, 283–300.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning, 3*, 9–44.

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in Neural Information Processing Systems* (pp. 1038–1044). The MIT Press.

Tsitsiklis, J. N., & Roy, B. V. (1996). Feature-based methods for large scale dynamic programming. *Machine Learning, 22*, 59–94.

Watkins, J. C. H. (1989). *Learning from delayed rewards.* Doctoral dissertation, Cambridge University.

Watkins, J. C. H., & Dayan, P. (1992). Technical note: Q-learning. *Machine Learning, 8*, 279–292.

Wilson, S. W. (1994). ZCS: A zeroth level classifier system. *Evolutionary Computation, 2*, 1–18.

Wilson, S. W. (1995). Classifier fitness based on accuracy. *Evolutionary Computation, 3*, 149–175.

Wilson, S. W. (2000). Get real! xcs with continuous-valued inputs. *Lecture Notes in Computer Science, 1813*, 209–222.