TR-NIS- 0003

Is Gradient Descent Method Effective for XCS? Analysis of Reinforcement Process in XCSG

Atsushi WADA, Keiki TAKADAMA, Katsunori SHIMOHARA, Osamu KATAI

2004.8.23

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: 0774-95-1011　Fax: 0774-95-2653

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: +81-774-95-1108

Is Gradient Descent Method Effective for XCS? Analysis of Reinforcement Process in XCSG

Atsushi Wada(*1,*2) Keiki Takadama(*1,*3)
Katsunori Shimohara(*1,*2) Osamu Katai(*2)
*(*1) ATR Network Informatics Laboratories, Japan*
*(*2) Graduate School of Informatics, Kyoto University, Japan*
*(*3) Tokyo Institute of Technology, Interdisciplinary Graduate School of Science and Engineering, Japan*
{wada,katsu}@atr.co.jp keiki@dis.titech.ac.jp katai@i.kyoto-u.ac.jp

Abstract. In this paper, XCS and its variant, XCSG are analyzed from aspect of function approximation (FA) method for Q-learning. From the analysis, we clarified the relation between XCS, XCSG, Q-learning with FA by focusing on the three elements in the update formula: (1) payoff definition; (2) residual term; and (3) gradient term, which revealed the inconsistency of the update process between the XCSG and Q-learning with FA. Our preliminary experiment also showed that the performance improvement of XCSG is not only due to the effect of applying gradient descent method but strongly dependent on the combination of the elements in the update formula.

1 Introduction

XCS [22] is a *Learning Classifier System* which adopts *accuracy* for its classifier fitness criteria. Although there have been proposed several modifications and extensions regarding XCS, the update formula of the classifier *prediction*, which is one of the core parts of XCS has been kept unchanged except for XCS with *gradient descent* (XCSG) [3].

XCSG is a distinct variant of XCS of which classifier prediction update method is modified by applying the idea of gradient descent, which is a method used in *Q-learning* enhanced by a *generalization* technique called *function approximation* (FA). An experimental result of XCSG applied to the multi-step maze problems was reported, which showed remarkable improvement on both the performance and the prediction accuracy compared to XCS.

However, how the update formula of XCSG is derived from the gradient descent method and the what is the difference between the update formula of XCS, XCSG and Q-learning with FA is not sufficiently clarified in the detail. Furthermore, the performance improvement of XCSG is not explained in this context.

Therefore, our objective is to: (1) clarify the difference of update methods between XCS, XCSG and Q-learning with FA focused on their reinforcement processes; and (2) identify the main factor of the performance advantage of XCSG compared with XCS.

2 Gradient Descent Method and XCSG

In this section, we briefly describe the gradient descent method in Q-learning with FA and XCSG, which is required for our further analysis. See [3] and [17] for the detail.

2.1 *Gradient descent method in Q-learning*

Gradient descent method is a general function optimization method which uses the derivative of the function. The gradient descent method is used to derive the update formula of Q-values in Q-learning enhanced by FA method, where the Q-table is indirectly represented as an approximated function controlled by a set of parameters $\vec{\theta}_t = (\theta_t(1), \theta_t(2), \ldots, \theta_t(n))$, where t denotes the time step.

In Q-learning with FA, the update formula must be defined for the parameters $\vec{\theta}_t$ to deal with the Q-value indirectly represented as a function of $\vec{\theta}_t$, which is described as:

$$\vec{\theta}_{t+1} = \vec{\theta}_t + \beta\left[v_t - Q_t(s_t, a_t)\right] \nabla_{\vec{\theta}_t} Q_t(s_t, a_t), \tag{1}$$

where s_t, a_t and $Q_t(s, a)$ each denotes the state, action and Q-value at time step t, and the parameter *learning rate* is named β. Here, v_t denotes the target value defined as $r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a)$, to which the current Q-value is modified. This update formula can be derived by applying the gradient descent method to minimize the mean square error between the target value v_t and the corresponding Q-value[1].

2.2 *Reinforcement process in XCSG*

In XCSG, the payoff $P(a_i)$ for the state s in the *prediction array*, which is defined by the next formula is regarded as a Q-value $Q(s, a_i)$ in Q-learning with FA.

$$P(a_i) = \frac{\sum_{cl_k \in [M]|_{a_i}} p_k \times F_k}{\sum_{cl_k \in [M]|_{a_i}} F_k}, \tag{2}$$

where p_k and F_k each denotes the attributes *prediction* and *fitness* of the classifier cl_k, and $[M]|_{a_i}$ denotes the set of classifier in *match set* $[M]$ having the action a_i. Here, the new update formula in XCSG is derived by applying the Formula 1 to Formula 2 and regarding the accuracy p_k as a parameter for approximated Q-value function.

$$p_k \leftarrow p_k + \beta(P - p_k)\frac{F_k}{F_{[A]_{-1}}}, \tag{3}$$

where $F_{[A]_{-1}}$ denotes $(\sum_{cl_j \in [A]_{-1}} F_j)$ using the *previous action set* $[A]_{-1}$, and P is defined as a target value $(r + \gamma \max_a P(a))$ using the reward r and the *discount factor* denoted as γ.

3 Analyzing Update Methods

In this section, we first compare the update formulae in XCS, XCSG and Q-learning with FA to clarify the differences. Next, we propose a table including all the possible update formulae which can be derived by combining the different elements clarified through the comparison.

3.1 *Difference in Update Methods*

For the comparison, we focus on the two main terms $(P - p_k)$ and $(\partial P / \partial p_k)$ in Formula 3 and name it *residual term* and *gradient term* for the convenience.

[1]The derivative of Q-value function $\nabla_{\vec{\theta}_t} Q_t(s_t, a_t)$ must be calculated, which depends on the approximation of Q-value function.

3.1.1 Residual term

The residual term calculates the difference between the target value and the current value regarding the prediction value. In both XCS and XCSG, the residual term is defined as $(P - p_k)$, which can be described as $(r + \gamma \max_a P(a) - p_k)$ by expanding the target value P. The corresponds term in Q-learning with FA is $(r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t))$, which is based on the fundamental equation in reinforcement learning called *Bellman equation*. By comparing these two formulae, an asymmetry is found that $Q_t(s_{t+1}, a)$ corresponds to $P(a)$ but $Q_t(s_t, a_t)$ corresponds to p_k.

By precisely applying the gradient descent method to Formula 2, the residual term is derived as $(P - P_{[A]_{-1}})$ where $P_{[A]_{-1}}$ denotes the payoff value of the selected action in the previous time step. In this paper, we call this form *XCS residual*, which is defined as $(P - P_{[A]_{-1}})$ and call the original form *Bellman residual*, which is defined as $(P - p_k)$.

3.1.2 Gradient term

The gradient term is a newly introduced term in XCSG compared with XCS, which is derived from the update formula in Q-learning with FA. Compared with XCSG, the gradient term of XCS can be regarded as a constant 1.

In XCSG, the gradient term $(\partial P/\partial p_k)$ is calculated as $(F_k/F_{[A]_{-1}})$ by regarding the prediction p_k as the parameter of the Q-value function.

However, the fitness value F_k is actually a function of p_k, so the derivative cannot be calculated as a simple form. This is because the update formula of F_k is based on the error attributes ϵ_k, which uses the prediction p_k in its update formula [2]. One way to precisely apply the gradient descent method is to modify the definition of the payoff $P(a)$ not to include the fitness F_k, which we discuss in the following.

3.1.3 Payoff definition

Here, we introduce an alternative payoff definition defined as the following formula, which enables to apply the gradient descent method precisely by excluding the influence of the fitness value.

$$P(a_i) = \frac{\sum_{cl_k \in [M]|_{a_i}} p_k \times num_k}{\sum_{cl_k \in [M]|_{a_i}} num_k}, \tag{4}$$

where num_k denotes the *numerosity* of the classifier cl_k, and $num_{[A]_{-1}}$ denotes $(\sum_{cl_j \in [A]_{-1}} num_j)$. For the convenient we name this definition *numerosity-weighted payoff* (Formula 4) and call the original definition in XCS *fitness-weighted payoff* (Formula 2).

By adopting the numerosity-weighted payoff, the gradient term $(\partial P/\partial p_k)$ can be precisely calculated as $(num_k/num_{[A]_{-1}})$.

[2]Furthermore, as the fitness F_k is also influenced by the value of itself in the previous time step, F_k must also be regarded as a parameter to approximate the Q-value function, which causes a contradiction between original fitness update formula in XCS.

3.2 *Relation between the update methods*

From the previous analysis which clarified the difference of the update methods in the payoff definition, the residual term, and the gradient term, here we present a systematic view to categorize the update methods based on these differences. By listing all the possible combinations of different elements, we can consider 8 different categories, which are listed in Table 1.

In this table, XCS is in the category F-I because XCS adopts the fitness-based payoff, the XCS residual and the gradient term in the form of 1. XCSG is in the category F-II, as it differs from XCS in the gradient term. By modifying the update method of XCS by adopting the update form of each category, eight different XCS variants can be generated[3]. Within such systems, N-IV based XCS is the only system of which update method is consistent with the Q-learning with FA of which update method derived from the gradient descent method.

Table 1: The update methods categorized by its payoff definition, gradient term, and residual term in the update formula.

Update method	**Payoff definition**	**Residual**	**Gradient**
F-I (XCS)	$(\sum_{cl_k} p_k \times F_k)/\sum_{cl_k} F_k$	$(P - p_k)$	1
F-II (XCSG)	$(\sum_{cl_k} p_k \times F_k)/\sum_{cl_k} F_k$	$(P - p_k)$	$(F_k/F_{[A]_{-1}})$
F-III	$(\sum_{cl_k} p_k \times F_k)/\sum_{cl_k} F_k$	$(P - P_{[A]_{-1}})$	1
F-IV	$(\sum_{cl_k} p_k \times F_k)/\sum_{cl_k} F_k$	$(P - P_{[A]_{-1}})$	$(F_k/F_{[A]_{-1}})$
N-I	$(\sum_{cl_k} p_k \times num_k)/\sum_{cl_k} num_k$	$(P - p_k)$	1
N-II	$(\sum_{cl_k} p_k \times num_k)/\sum_{cl_k} num_k$	$(P - p_k)$	$(num_k/num_{[A]_{-1}})$
N-III	$(\sum_{cl_k} p_k \times num_k)/\sum_{cl_k} num_k$	$(P - P_{[A]_{-1}})$	1
N-IV (Q-learning + FA)	$(\sum_{cl_k} p_k \times num_k)/\sum_{cl_k} num_k$	$(P - P_{[A]_{-1}})$	$(num_k/num_{[A]_{-1}})$

4 Preliminary Experiment

In this section, we present a preliminary experiment to compare the performance of the eight XCS variants representing the categories in Table 1. We use `Maze6` problem, which is a same problem used in [3] to double-check the simulation results and to make the comparison easier[4]. For the performance measure, the number of steps to the goal is used. All the statistics are averaged over 20 experiments.

Figure 1 reports the performance of methods F-I,F-II,F-III and F-IV. Figure 2 reports the performance of methods N-I,N-II,N-III and N-IV. In both the graphs, the number of learning problems is on the the horizontal axis and the average number of steps to goal is on the vertical axis. From both the graphs, we can find that the update method which showed better performance than F-I (XCS) was only F-II (XCSG), which converged to the optimal performance of `Maze6`. On the other hand, F-IV and N-IV showed the worst performance.

[3]Note that each of these categories specifies only the form of the update method. Several XCS variants can be included in a category which adopt different mechanisms in the other components such as classifier fitness criteria.

[4]For this reason we use the same parameter settings with [3] as follows: the population size N is 3000 classifiers, $P_{\#} = 0.3$, $\beta = 0.2$, $\gamma = 0.7$, $\chi = 0.8$, $\mu = 0.01$, $\theta_{mna} = 8$, $p_{explr} = 1.0$, $\theta_{GA} = 100$, $\epsilon_0 = 1$, $\theta_{del} = 20$, $doGASubsumption = 0$, $doASSubsumption = 0$.

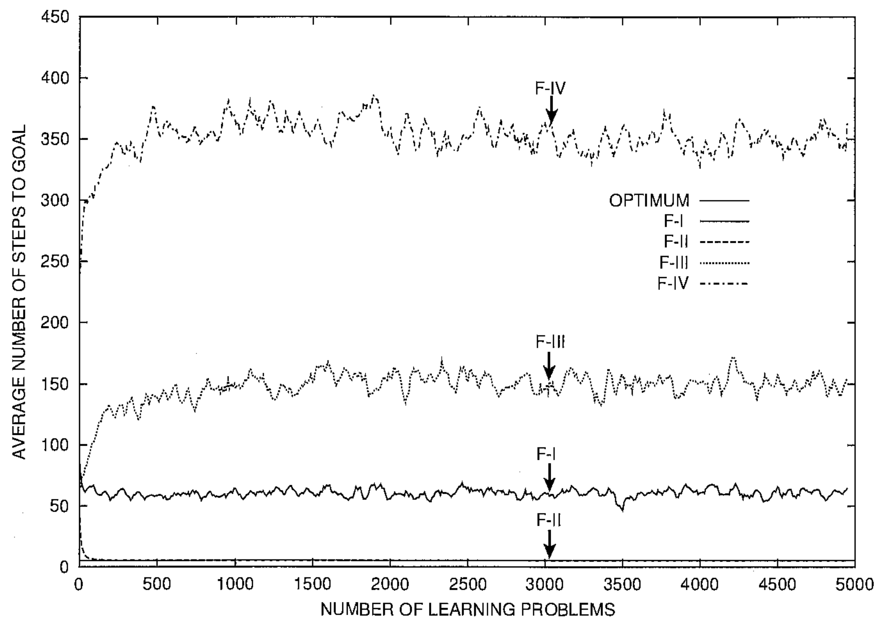

Figure 1: The performance of methods F-I to F-IV applied to `Maze6`.

5 Discussions and Conclusion

To clarify the relation between XCS, XCSG, Q-learning with FA by focusing on the three elements in the update formula: (1) payoff definition; (2) residual term; and (3) gradient term. And presented eight categories by combining these differences, which revealed that XCSG are not precisely the gradient descent based XCS.

Furthermore, we presented a preliminary experiment by applying these update methods to `Maze6`, which showed that the main factor in performance improvement of XCSG, which we named F-II is due to the combination of the XCS residual term and the gradient term newly proposed in XCSG. However, this combination is effective only in the case of fitness-weighted payoff, as the similar method N-IV adopting numerosity-weighted payoff showed the worst performance within the others.

From these results, we can conclude that the performance improvement in XCSG is not due to the gradient descent method in the strict meaning, but enhancing the framework of XCS, which might be a distinct advantage of XCS framework over Q-learning with FA.

In this paper, we did not analyze deeply about the method N-IV, however, update formula of N-IV is consistent with Q-learning with FA and we think that the analysis between the XCS with N-IV update and Q-learning with FA will help understanding XCS in the context of reinforcement learning researches.

Acknowledgements

This research was conducted as part of 'Research on Human Communication' with funding from the National Institute of Information and Communications Technology of Japan and the Okawa foundation for information and telecommunications.

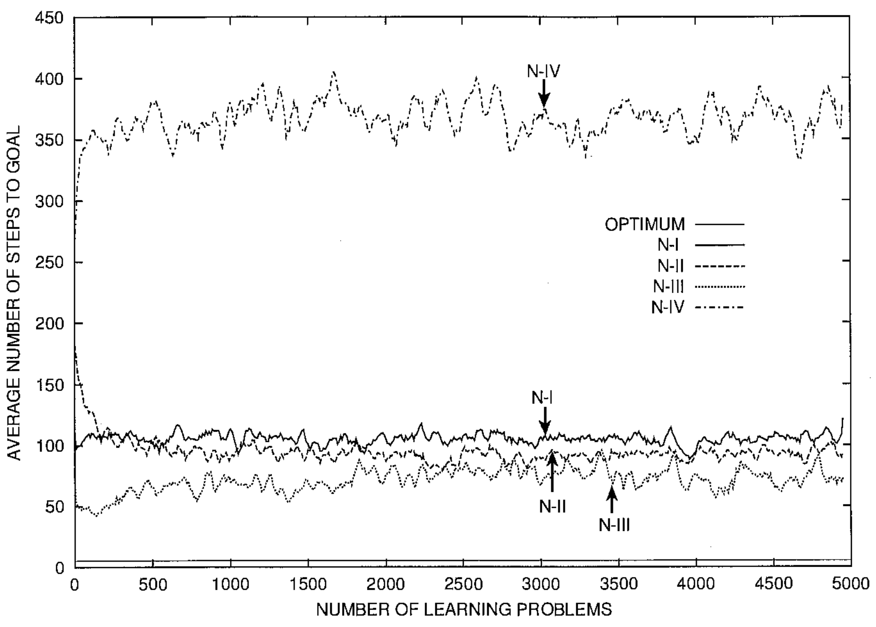

Figure 2: The performance of methods N-I to N-IV applied to `Maze6`.

References

[1] Leemon C. Baird. Residual algorithms: Reinforcement learning with function approximation. In *International Conference on Machine Learning*, pages 30–37, 1995.

[2] Leemon C. Baird. *Reinforcement Learning Through Gradient Descent.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA 15213, 1999.

[3] Martin V. Butz, David E. Goldberg, and Pier L. Lanzi. Gradient descent methods in Learning Classifier Systems: Improving XCS performance in multistep problems. Illinois Genetic Algorithms Laboratory (IlliGAL) Technical Report No. 2003028, 2003.

[4] Martin V. Butz, Tim Kovacs, Pier L. Lanzi, and Stewart W. Wilson. How XCS evolves accurate classifiers. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 927–934, San Francisco, California, USA, 7-11 2001. Morgan Kaufmann.

[5] Martin V. Butz and Stewart W. Wilson. An algorithmic description of XCS. *Lecture Notes in Computer Science*, 1996:253+, 2001.

[6] Marco Dorigo and Hugues Bersini. A comparison of Q-learning and classifier systems. In *Proceedings of From Animals to Animats, Third International Conference on Simulation of Adaptive Behavior*, 1994.

[7] Geoffrey J. Gordon. Stable function approximation in dynamic programming. In Armand Prieditis and Stuart Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning*, pages 261–268, San Francisco, CA, 1995. Morgan Kaufmann.

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

[9] Tim Kovacs. Evolving optimal populations with XCS classifier systems. Technical Report CSRP-96-17, October 1996.

[10] Tim Kovacs. Deletion schemes for classifier systems. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 99)*, pages 329–336. Morgan Kaufmann, July 1999.

[11] Tim Kovacs. Two views of classifier systems. In *Fourth International Workshop on Learning Classifier Systems - IWLCS-2001*, pages 367–371, San Francisco, California, USA, 7 2001.

[12] Pier L. Lanzi. Learning classifier systems from a reinforcement learning perspective. *Soft Computing*, 6:162–170, 2002.

[13] Pier Luca Lanzi. Extending the representation of classifier conditions part i: From binary to messy coding. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 99)*, pages 337–344. Morgan Kaufmann, July 1999.

[14] Pier Luca Lanzi and Alessandro Perrucci. Extending the representation of classifier conditions part ii: From messy coding to s-expressions. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 99)*, pages 345–352. Morgan Kaufmann, July 1999.

[15] Satinder P. Singh, Tommi Jaakkola, and Michael I. Jordan. Reinforcement learning with soft state aggregation. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 361–368. The MIT Press, 1995.

[16] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[17] Richard S. Sutton and Andrew G. Barto. *An introduction to reinforcement learning*. MIT Press, Cambridge, MA., 1998.

[18] John N. Tsitsiklis and Benjamin Van Roy. Feature-based methods for large scale dynamic programming. *Machine Learning*, 22(1-3):59–94, 1996.

[19] Atsushi Wada, Keiki Takadama, Katsunori Shimohara, and Osamu Katai. Comparison between Q-Learning and ZCS Learning Classifier System: From aspect of function approximation. In *The 8th Conference on Intelligent Autonomous Systems*, 2004. To appear.

[20] Christopher J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, 1989.

[21] Christopher J. C. H. Watkins and Peter Dayan. Technical note: Q-learning. *Machine Learning*, 8:279–292, 1992.

[22] Stewart W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

[23] Stewart W. Wilson. Generalization in the XCS classifier system. In *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 665–674, University of Wisconsin, Madison, Wisconsin, USA, 22-25 1998. Morgan Kaufmann.

[24] Stewart W. Wilson. Get real! xcs with continuous-valued inputs. *Lecture Notes in Computer Science*, 1813:209–222, 2000.

[25] Stewart W. Wilson. Mining oblique data with XCS. *Lecture Notes in Computer Science*, 1996:158–??, 2001.