知 能 と 複 雑 系 114－9
（1 9 9 8. 1 0. 2 ）

力学系と学習の融合による動的環境の学習
－ カオス予測と強化学習による実現 －

三上　貞芳
北海道大学工学部

要旨：動的（非定常な）環境でコストミニマムな動作を学習するシステムについて提案する．予測フェーズとして決定論的力学系の再構築による短期時系列予測（カオス予測）を用い，それをコスト探索のための試行錯誤学習（強化学習）に組み込んで実現する方法を提案する．この枠組みは複数の学習エージェントが互いに相手を環境の一部として学習するような場合に一般的な解決法を提示する可能性がある．その例として資源配分問題を例にとり，学習と全体のダイナミクス形成が同時に進行すること，またそれを以上の枠組みで検出・コスト最適行動へ収束させることが出来ることを示す．

Learning in Dynamic Domain by the Method of Dynamics Reconstruction and Machine Learning
(A Method by Reinforcement Learning and Chaotic Time Series Prediction)

Sadayoshi Mikami*
Faculty of Engineering
Hokkaido University

Abstract: We discuss a system that learns in dynamic domain. We separate the problem into two phases: first is the short-term state transition prediction by a method of dynamics reconstruction, second is the cost-minimum action search by using trial-and-error based learning such as Reinforcement Learning. Such system provides a general method for interacting learning agents, where each agent observes another as a part of its environment. We put a hypothesis that such interacting agents exhibit deterministic dynamics. Although theoretical proof has not be given, we could get the results that, under simple conflict resolution task by interacting multi agents, the agents with proposed method has effectively worked to converge.

1 INTRODUCTION

We discuss a system that learns in dynamic domain. We separate the problem into two phases: first is the short-term state transition prediction by a method of dynamics reconstruction, second is the cost-minimum action search by using trial-and-error based learning such as Reinforcement Learning. Such system provides a general method for interacting learning agents, where each agent observes another as a part of its environment. We put a hypothesis that such interacting agents exhibit deterministic dynamics. Although theoretical proof has not be given, we could get the results that, under simple conflict resolution task by interacting multi agents, the agents with proposed method has effectively worked to converge.

The situation where Reinforcement Learning agents are interacting with each other is a hard environment to learn. The environment continuously changes by the interaction and exhibits context dependency. Such situation will commonly be found when we plan to apply Reinforcement Learning to large-scale distributed problems without communication (Lin, 1993, Connell, 1993, Whitehead, 1995). Traffic signal control, air-conditioning systems, and network routing problems are amongst them. However, if the behavior of the other agents is perfectly predictable, these situations are reduced to normal Reinforcement Leaning problems. This is because context dependency is resolved by knowing the predicted next state. A single strategy should be applied for a given input state.

In this paper, we put a hypothesis that such interacting agents exhibit deterministic dynamics. It is known that short-term prediction of the state transition is available for a system having deterministic dynamics (Crutchfield, 1994, Pollack, 1992). For example, *embedding* is one of techniques to extract dynamics from a time series of sensory input, and to predict one time step evolution (Takens, 1981, Sugihara, 1990).

This means that, if the hypothesis hold true, then an agent can understand the other agents' policies in terms of the dynamics that the entire system obeys. The short-term prediction implemented in each agent will resolve the context dependency, and then successive Reinforcement Learning will converge. Thus, co-operation will be learned by observation.

To this end, we implement dynamics-detection-based prediction method into TD Learning algorithm (Sutton, 1998). Although theoretical proof has not been given, we get the results that, under simple conflict resolution task by interacting multi agents, the agents with proposed method has effectively worked to converge.

2 REINFORCEMENT LEARNING WITH STATE PREDICTION FUNCTION

2.1 STANDARD RL IN STATIC ENVIRONMENT

Consider a set of standard Reinforcement Learning agents. Each agent observes the other agent's behavior (such as position) as one of its observing state.

For the simplicity of the discussion, a state vector x_t is assumed to be one-dimensional discrete value. We also assume that an evaluation *(payoff)* function depends on the state transition x_t, x_{t+1} and the action a_t, as $r_t = r(x_t, x_{t+1}, a_t)$.

Reviewing the standard (TD) learning, the objective of the learner is to maximize a cumulative discounted payoff such that,

$$u(x_t) = \sum_{t'=t}^{\infty} \gamma^{t'-t} r(x_{t'}, x_{t'+1}, a_{t'}) , \qquad (1)$$

which represents a sum of future payoff under the condition where the best actions are selected from t to ∞. In the form of the time evolving system, it is rewritten as,

$$u(x_t) = r(x_t, x_{t+1}, a_t) + \gamma u(x_{t+1}) . \qquad (2)$$

Define that an action a_t is optimal if it satisfies

$$u(x_t) = p(x_t, a_t, t) . \qquad (3)$$

$p(x_t, a_t, t)$ describes a function that gives a policy when a_t is applied to the state x_t at time t. From this definition, the optimal action is selected as the a that satisfies

$$\arg\max_a p(x_t, a, t) . \qquad (4)$$

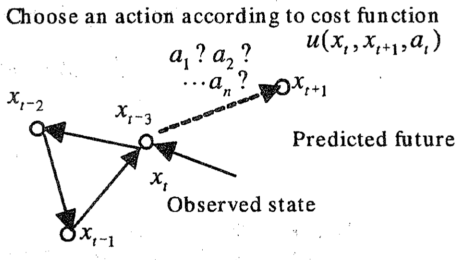


Figure 1: Prediction based learning. A cost function is associated to the current state, future state and the performed action.

2.2 LEARNING IN STATIC DOMAIN

In standard RL, the state is regarded as static. It means that there exists a function f such that $x_{t+1} = f(x_t, a_t)$. By using f, the function p is rewritten as,

$$p(x_t, a_t, t) = r(x_t, f(x_t, a_t), a_t) + u(f(x_t, a_t)) . \qquad (5)$$

Note that the equation does not contain x_{t+1}, and the function $p(\bullet)$ does not depend on time variable t. An optimal action is then selected by looking up only $p(x, a)$ table.

This lets the learning phase as simple as follows: Recall that the learning is a process of refining the approximation for $u(x_t)$ by using actual payoff values given through trials. Assume that a trial was performed and the system has then moved to a new state y. Let r be a payoff given by the trial. From the eq.2, the following must hold,

$$u(x) = r + \gamma u(y) , \qquad (6)$$

Let $\hat{u}(x)$ be the current approximation of $u(x)$. The error value for $\hat{u}(x)$ is ,

$$d = (r + \gamma u(y)) - \hat{u}(x) . \qquad (7)$$

It is used to gradually refine the $\hat{u}(x)$ as,

$$\hat{u}(x) \leftarrow \hat{u}(x) + \alpha d , \qquad (8)$$

where $\alpha \in [0,1]$ is a learning coefficient. From the definition, the policy p has the same error value. Thus, by using the symbol $\hat{p}$ for the current approximation for p, the following updating rule is applied:

$$\hat{p}(x, a) \leftarrow \hat{p}(x, a) + \beta d , \qquad (9)$$

where $\beta \in [0,1]$ is the learning coefficient for the policy.

2.3 EXTENSION TO INCLUDE STATE PREDICTION FUNCTION

Let us consider the case where the time variable t is not removed from the policy table $p(x,a,t)$. If x_{t+1} is known from x_t, a_t and t, we can remove the time t dependency of p.

Let g be a function that returns a (predicted) next state, such that $x_{t+1} = g(x_t,a_t,t)$. The table $p(x_t,a_t,t)$ is then,

$$p(x_t,a_t,t) = r(x_t,a_t,g(x_t,a_t,t)) + u(g(x_t,a_t,t)), \quad (10)$$

Note that, in the equation, $g(x_t,a_t,t)$ is already known at time t. By rewriting $g(x_t,a_t,t)$ as y, we get an equation for p that do not use t term as follows:

$$p(x_t,a_t,y) \equiv p(x_t,a_t,t) = r(x_t,a_t,t) + u(y). \quad (11)$$

This means that the table p must be addressed by three independent variables. Thus, the selection of an appropriate action should be done by a slightly different way: For every possible action a, a (predicted) next state $y(a)$ is calculated by using the function g as $y(a) = g(x_t,a,t)$. This is then used to look up a utility of applying the action a in the state x_t at time t, $p(x_t,a,y(a))$. Clearly, like equation (4), an appropriate action a is selected by the one that gives the maximum payoff,

$$a = \arg\max_a p(x_t,y(a),a). \quad (12)$$

The learning algorithm is the same as the standard TD, by modifying an element of the table $p(x_t,a_t)$ using x_t,a_t, and r. In this dynamic environment version, the updating scheme is extended to incorporate in the transited state y. Note that the utility storage $\hat{u}(x)$ is still used in the same way as the standard TD. At first, the TD error d is given by $d = (r + \gamma u(x_{t+1})) - \hat{u}(x_t)$. Then, Eq.8 is used to update $\hat{u}$. The content of p to be modified by using d should be addressed by using current (meaning that the modification is done after state transition) state x_{t+1}, the last performed action a_t, and the previous state x_t. The updating equation is then,

$$p(x_t,x_{t+1},a_t) \leftarrow p(x_t,x_{t+1},a_t) + \beta d. \quad (13)$$

An important characteristic of this scheme is that, the action selection is done by a forecast state (using the state transition function g), but the modification of policy y is done by the actually transited state. The latter is based on reliable information since everything it uses has already happened. The error in the function g is reset in the succeeding policy modification phase.

2.4. STATE PREDICTION BY DYNAMICS RECONSTRUCTION

In our method, the learner is expected to know the deterministic dynamics that produces its sensory input sequence. Since it is not explicitly given, the learner has to reconstruct it only through observation. Then, the reconstructed dynamics is used to predict the short-term future of its sensory input. The function g, $x_{t+1} = g(x_t,a_t,t)$, is such a dynamics-reconstruction-based time series forecaster.

A method to realize g, is known as the *local reconstruction by embedding* (Alligood, 1996). The method embeds a time-series $\{x_t\}$ into higher dimensional space by using delay-coordinate. The embedded time series is defined as a series of a vector $\mathbf{z}_t = (x_t, x_{t+\tau}, \dots, x_{t+(n-1)\tau})$ for the delay time τ. From the *theory of embedding* (Takens, 1981), it is known that if a time-series is produced by an n-dimensional deterministic dynamics, and if the dynamics has an attractor, then, by embedding the series into a delay-coordinate system that is higher than n, the attractor is reconstructed into the delay-coordinate. We write the embedded dynamics as $\mathbf{z}_{t+1} = h(\mathbf{z}_t)$.

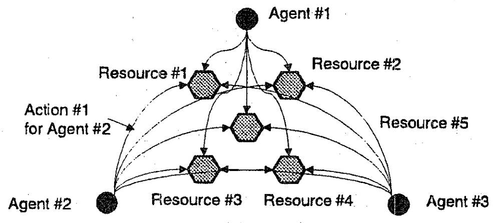


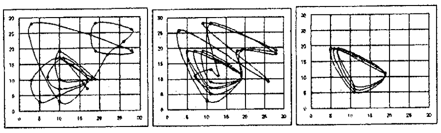
Figure 2: A conflict resolution game by three agents for five resources. Size of state space is 32, and actions are 5. Gaussian action selection at temperature 1 is used.

Consider a point $\mathbf{z}_t$ in h. From the property of an attractor, a set of points $\{\mathbf{z}_u; |\mathbf{z}_t, \mathbf{z}_u| < \varepsilon\}$ in h that are close to $\mathbf{z}_t$ will be transferred to the neighboring points $\{\mathbf{z}_{u+1}; |\mathbf{z}_{t+1}, \mathbf{z}_{u+1}| < \varepsilon'\}$ in the same attractor h. The embedding-based time series forecast uses this property. Let us find the next state $\hat{x}_{t+1}$ for x_t. At first, x_t is embedded into $\mathbf{z}_t$ with past states. From the property of h, some points - $\mathbf{z}_u, \mathbf{z}_v$, and $\mathbf{z}_w$, for example - that are close to $\mathbf{z}_t$, will be transferred to the neighbors of $\mathbf{z}_{t+1}$. Although $\mathbf{z}_{t+1}$ is not known, we can approximate its location so that it preserves geometrical relationship between z_t and z_u, z_v, z_w, such as preserving the ratio of areas surrounded by these points (Sugihara, 1990). $\hat{x}_{t+1}$ is separated from the approximated z_{t+1}. Then, the RL uses $\hat{x}_{t+1}$ to select an action, modify the policy, and store that policy.

3. INTERACTING LEARNERS

The forecasting function g is used in conjunction with Eq.10-13 to provide dynamics-forecasting-based RL (DFRL). The hypothesis here is that, if the agents using DFRL are interacting with each other, then the time evolution of the states of these agents will converge into fixed dynamics. In our scenario, forming dynamics of having an attractor and the forecasting based on that attractor will concurrently work together. However, the theoretical analysis is not yet given until now.

We conducted basic experiments to test this hypothesis. The test problem is a simple conflict resolution game by three agents. The action of an agent is to choose one resource (place) out of five. The decision making is synchronized over the agents. If more than one agent choose the same resource, they are penalized by -1, otherwise rewarded by 1. An agent only observes the other agents' choices of the last trial. (Fig.2)



(*a*) from 1 to 24 steps (*b*) from 25 to 50 steps
(*c*) from 51 to 100 steps

Figure 3: Plot of states in embedded space. X and y-axes correspond to x_t and x_{t+1}.

Two-dimensional embedding method was applied for the dynamics-forecast-based RL. The results are shown in Figs.3 and 4. It is observed that an attractor was explicitly formed. After 50 trials, the reconstruction of state was almost converged, so that the rate of failure was reduced to almost zero, whilst normal Q-learning doesn't achieve any improvement. The agents were synchronously choosing the resource (0,1,3), (0,1,4), and (1,0,2).

Although this is a very limited experiment, the result is encourageous because it is shown that such a system having highly-freedom could form attractor in a short period. this is a preliminary result

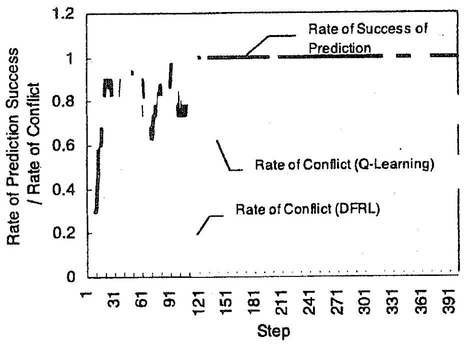


Figure 4: Rate of conflict and success for prediction.

4. CONCLUSION

For the acquisition of co-operation for a group of agents without communication, we have proposed an approach based on a combination of time-series prediction of an unknown dynamics with Reinforcement Learning. The assumption that the interaction of these learning agents exhibits attractor in low dimension is difficult to prove. But if this assumption holds true, it seems possible by our algorithm to acquire reactive rules under a large class of complicated (even in chaotic) problems with many systems interacting with each other, such as traffic control, dynamic routing, and navigation of vehicles.

References

Alligood, K.T., Sauer, T.D & Yorke, J.A. (1996) Chaos - an introduction to dynamical systems, Springer Verlag.
Connell, J.H. & Mahadevan, S. (1993) Robot Learning, 1, Kluer Academic Press.
Crutchfield, J.P. (1994) The Calculi of Emergence: Computation, Dynamics and Induction, Physica D, 75, pp.11-54.
Sugihara, G. and R. M. May (1990). Nonlinear Forecasting as a Way of Distinguishing Chaos from Measurement Error in Time Series, Nature, 344, pp.734-740.
Lin, L-J. (1993) Reinforcement Learning with hidden states, From animals to animats 2, pp.271-278, The MIT Press.
Moore, A.W. & Atkeson, C.G. (1993) Memory-Based Reinforcement Learning: Converging with Less Data and Less Real Time, Robot Learning, pp.79-103, Kluer Academic Press.
Pollack, J.B. (1992) The Induction of Dynamical Recognizers, Proc. IJCNN.
Sandholm, T. & Crites, R.H. (1995) Multiagent Reinforcement Learning in the Iterated Prisoner's Dilemma, Biosystems, 37, pp.147-166.
Sutton, R. & Barto, A. (1998) Reinforcement Learning: An Introduction: The MIT Press.
Takens, F. (1981) Detecting strange attractors in turbulence, Lecture Notes in Mathematics, 898, pp.366-381.
Whitehead, S.D. & Lin, L-J. (1995) Reinforcement learning of non-Markov decision processes, Artificial Intelligence, 73, pp.271-306.