3A2-2

POMDP 環境中での TD-Network の自動獲得: 単純再帰構造による拡張

Automatic Acquisition of TD-Network in POMDP Environments: Extension with SRN structure

牧野 貴樹[*1]
Takaki Makino

[*1]東京大学 総括プロジェクト機構
Division of Project Coordination, Tokyo University

We propose a new neural network architecture, Simple recurrent TD Networks (SR-TDNs), that learns to predict future observations in partially observable environments, using proto-predictive representation of states. SR-TDNs incorporate the structure of simple recurrent neural networks (SRNs) into temporal-difference (TD) networks to use proto-predictive representation of states. Our simulation experiments revealed that these networks have better on-line learning capacity than TD networks in various environments.

1. Introduction

Predictive representations [Littman 02, Jaeger 00] are a relatively new group of approaches to expressing and learning grounded knowledge about partially-observable dynamical systems. These approaches represent the state of a dynamical system as a vector of predictions, based on the basic principle that important knowledge about the world can be represented strictly in terms of the relationships between predictions of observable quantities. In the *predictive state representations* (PSRs) introduced by Littman et al. [Littman 02], each prediction is an estimate of the probability of *tests*, defined as some sequence of observations given a sequence of actions.

Temporal-Difference (TD) networks [Sutton 05] were one of the first proposals for an on-line learning algorithm of predictive representations. The basic idea was to apply TD-learning techniques in reinforcement learning to general predictions, i.e., train each prediction targeting at the value of another prediction or observation at a later time. The targeting relations between predictions are given as a *question network*, and another *answer network* is trained like an artificial neural network. TD networks demonstrated their efficiency for learning predictions; later, the idea was also incorporated into PSRs and proven to work [Wolfe 05].

These approaches, however, share the same restrictions derived from the basic principle of predictive representation, i.e., they have required that a specified set of tests, or question networks, to be sufficient for representing the state of a dynamical system. A sufficient set of tests, or *core tests*, can only be computed when the underlying dynamics in the environments is known in advance; if an insufficient set of tests is given, an agent can only have an insufficient representation of a state, and consequently, an accurate prediction cannot be learned. Several methods for automatically discovering the test set have been proposed [James 04, McCracken 06], but they are based on greedy searches in the space of test sets, and habe thus been theoretically unable to deal with various simple dynamics.

Our approach deviates slightly from the basic principle, and incorporates ideas from studies on connectionism (artificial neural networks). In the long history of connectionism, various network architectures have been proposed for predicting temporal sequences. One of the most famous architectures is the simple recurrent networks (SRNs; [Elman 95]). Although SRN's *context layers* (state representations in SRNs) are not grounded on observations, it is known that SRNs are capable of predicting more complex temporal sequences than the networks with only grounded state representations (e.g. Jordan's networks [Jordan 86]).

In this paper, we propose a new network architecture called Simple recurrent TD networks (SR-TDNs), which is a fusion of SRNs and TD Networks. SR-TDNs uses the state representation from SRNs and training strategy from TD networks, i.e., context layers and TD learning. In spite of deviation from the principle of predictive representations to ground state representations on observations, SR-TDNs follow the same learning strategy as TD networks, i.e., applying TD-learning to general predictions. Since the context layers have the prototypic information prior to the output prediction, we call this a *proto-predictive representation* of states. We show that SR-TDNs performs good learning in simulations.

2. Partially-Observed MDP

A partially-observable Markov decision process (POMDP) is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, P \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A} = \{a_1, \ldots, a_{|\mathcal{A}|}\}$ is the action space, $\mathcal{O} = \{o_1, \ldots, o_{|\mathcal{O}|}\}$ is the observation space, and $P(s,a,o,s') : S \times A \times O \times S \to [0,1]$ is the probability that action a in state s at time t will give observation o and lead to state s' at time $t+1$. An agent is required to learn the prediction, $T(o|a,t) = Pr(o[t+1] = o \,|\, a[t+1] = a, o[t], a[t], o[t-1], a[t-1], \cdots)$, i.e., a conditional probability of a future observation given a past sequence of observations and actions, without knowing $P(s,a,o,s')$.

Generally, even with a full knowledge of $P(s,a,o,s')$, one cannot determine the current state, $s[t]$, from past observations and actions. The best reasoning with knowledge about $P(s,a,o,s')$ is to maintain a belief state, $\mathbf{b}[t]$, whose element $b_s[t]$ specifies the conditional probability of the agent at time t being in state s.

$$b_{s'}[t+1] = Pr(s'|o[t+1], a[t+1], b[t]) = \frac{\sum_{s \in \mathcal{S}} b_s[t] P(s, a[t], o[t], s')}{T^*(o[t+1]|a[t+1], t)} \quad (1)$$

連絡先: 牧野 貴樹: 277-8568 千葉県柏市柏の葉 5-1-5 Tel: 04-7136-3973 mak@scint.dpc.u-tokyo.ac.jp. This work was supported by JSPS (KAKENHI 20700126).

where $T^*(o|a,t)$ is an oracle, or the theoretically best prediction obtained by learning. In the following, the learning algorithms are measured in terms of the mean squared error between the agent's prediction and the oracle, for a given time window, L.

$$\mathrm{MSE}[t_0] = 1/L \sum_{t=t_0}^{t_0+L-1} \sum_{o \in \mathcal{O}} (T(o|a[t+1],t) - T^*(o|a[t+1],t))^2 \quad (2)$$

3. Network Architectures

In this section, we give the definitions of the network architectures. First, we explain two existing network architectures, i.e., SRN with actions and TD Networks. After that, we describe our proposal, simple recurrent TD networks (SR-TDN). Figure 1 illustrates these network architectures with an emphasis on their similarities. The main differences lie in the source of state information (the bottom left of each subfigure) and the target of learning (the top of each subfigure).

3.1 Simple Recurrent Networks with Actions

Simple recurrent networks (SRN) [Elman 95] are designed to predict observations in hidden Markov models, that is equivalent to POMDPs with only one action. We made a straightforward extension to Elman's simple recurrent network (SRN) [Elman 95], so that it can predict POMDPs. Formally,

$$\mathbf{x}^H[t] = g^H(W^{HI} \cdot \mathbf{x}^I[t] + W^{HH} \cdot \mathbf{x}^H[t-1] + \mathbf{b}^H) \quad \text{and} \quad (3)$$

$$\mathbf{x}^O[t] = g^O(W^{OH} \cdot \mathbf{x}^H[t] + \mathbf{b}^O) \quad . \quad (4)$$

The state of the network, i.e., the information carried from the previous time step, is the value of the hidden layer $\mathbf{x}^H[t]$; SRN calls this the *context layer*.

The network has $N_I = |\mathcal{A}| + |\mathcal{O}|$ input units and $N_O = |\mathcal{A}||\mathcal{O}|$ output units. The input is N-to-1 representation of $o[t]$ and $a[t]$, i.e., units corresponding to the observation or the action are set to 1, and the rest are set to 0. The target of the output is the N-to-1 representation of $o[t+1]$ the output, conditioned by each possible action at $t+1$. In training, the output units conditioned with actually taken action $a[t+1]$ are trained with the result of observation $o[t+1]$. Formally,

$$\mathbf{x}^I[t] = (a_1[t], \ldots, a_{|\mathcal{A}|}[t], o_1[t], \ldots, o_{|\mathcal{O}|}[t])^T \quad \text{and} \quad (5)$$

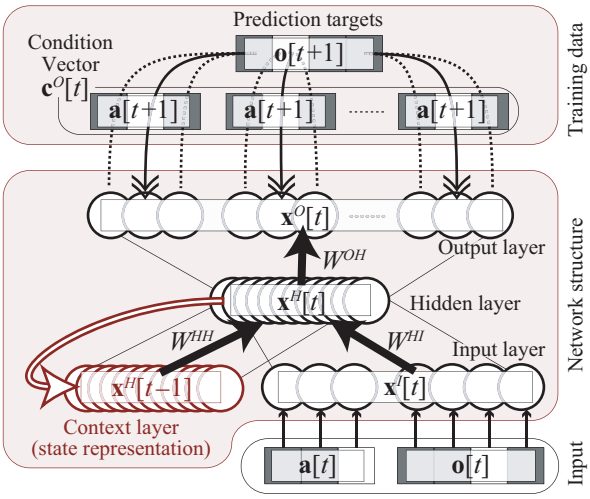
$$\langle \mathbf{t}^O[t], \mathbf{c}^O[t] \rangle = \left\langle \begin{pmatrix} o_1[t+1] \\ o_1[t+1] \\ \vdots \\ o_1[t+1] \\ o_2[t+1] \\ \vdots \\ o_{|\mathcal{O}|}[t+1] \end{pmatrix}, \begin{pmatrix} a_1[t+1] \\ a_2[t+1] \\ \vdots \\ a_{|\mathcal{A}|}[t+1] \\ a_1[t+1] \\ \vdots \\ a_{|\mathcal{A}|}[t+1] \end{pmatrix} \right\rangle , \quad (6)$$

where

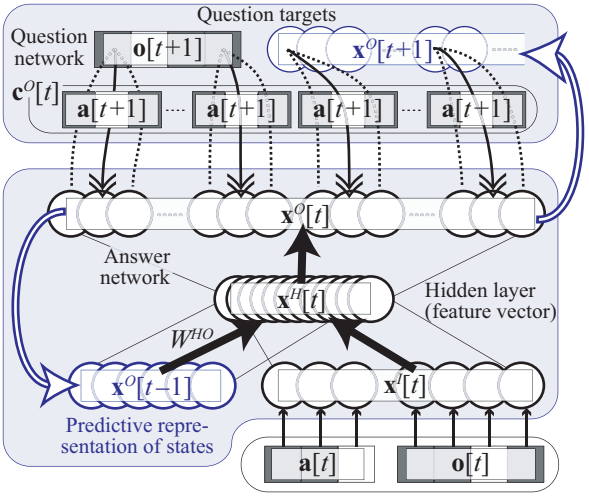
$$a_i[t] = \begin{cases} 1 & a[t] = a_i \\ 0 & \text{otherwise} \end{cases} \quad (i = 1, \ldots, |\mathcal{A}|) \quad \text{and} \quad (7)$$

$$o_j[t] = \begin{cases} 1 & o[t] = o_j \\ 0 & \text{otherwise} \end{cases} \quad (j = 1, \ldots, |\mathcal{O}|) \quad . \quad (8)$$

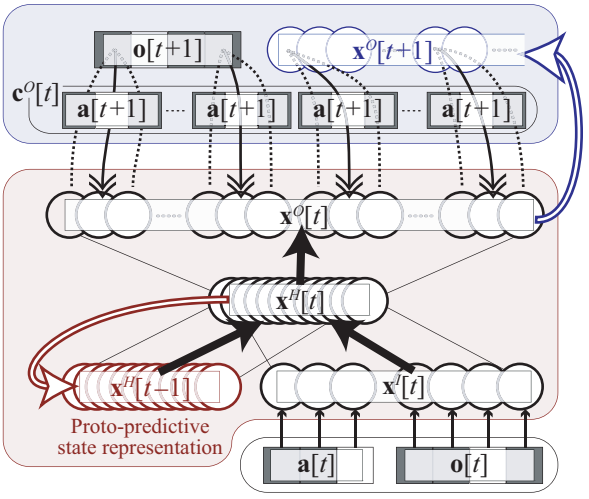
When $|\mathcal{A}| = 1$, this network is equivalent to the original SRN. Predictions about future observations can be obtained from the output units of the network, i.e., $T(o_j|a_i,t) = x^O_{j|\mathcal{A}|+i}[t]$. Other architectures explained below are also designed so that the same equation gives the prediction of the network.



(a) SRN with Actions



(b) TD Network



(c) SR-TDN

Figure 1: Network architectures. Circles denote neural units, thick black arrows denote network connections, white arrows denote copying over adjacent time steps, double-headed arrows denote feedback from training data, and dotted lines denote feedback relations whose condition is not satisfied.

Table 1. Summary of test environments.

	$\|\mathcal{A}\|$	$\|\mathcal{O}\|$	$\|S\|$
(a) Network	4	2	7
(b) Shuttle	3	5	8
(c) 4x3 Maze	4	6	11
(d) 8-state Ring	2	2	8

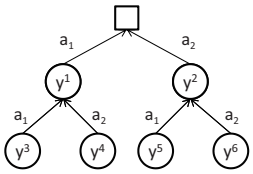


Figure 2: Example of TD network we focused on in this paper. Square denotes observation node.

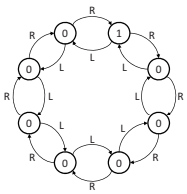
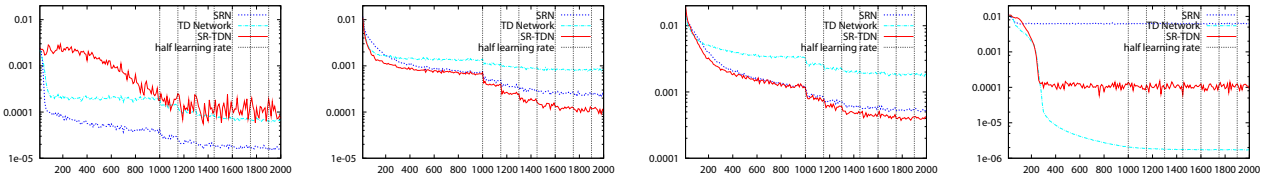


Figure 3: 8-state ring world.



(a) Network (b) Shuttle (c) 4x3 Maze (d) 8-state ring

Figure 4: Test results, plotted with simulation time in X-axis and the mean squared error in Y-axis.

3.2 TD Networks

The purpose of TD networks [Sutton 05] is to learn the prediction of future observations obtained from the environment. A TD network consists of a set of nodes and links, and each node represents a single scalar prediction. A node has one or more links directed to other nodes or observations from the environment, which denotes the targets of the node's prediction. A link may have a condition, which indicates that the node is a conditional prediction of the target. This set of nodes and links is called a *question network* since each node represents some question about the environment.

As in the previous studies [Tanner 05b], we have focused on a subset of TD networks, in which every node has a single target (hereafter called, the parent node), every link is conditioned with an action, and there are no loops in the question network. Figure 2 is an example of such a TD network for a scalar observation. Node y_1 predicts observation at the next step if action a_1 is taken. Node y_4 predicts the value of node y_1 at the next step if action a_2 is taken; consequently, y_4 gives the prediction for observation after two steps of actions, a_2 and a_1. In the following, we have denoted $p(y_i) \in \{o_1, \cdots, o_{|\mathcal{O}|}, y_1, \cdots, y_{i-1}\}$ as the parent node of y_i, and $a(y_i)$ as the condition for the link between y_i and $p(y_i)$.

To provide answers to the questions asked by the question network, each node in a TD network also works as a function approximator. The inputs to the function approximator of a node are defined by *answer network*, taking values from other nodes, available observations, and actions to be taken. These function approximators are trained so that the output of the nodes becomes the answers to the question asked by the question network.

TD networks can be represented as the following (Fig. 1b):

$$\mathbf{x}^H[t] = g^H(W^{HI} \cdot \mathbf{x}^I[t] + W^{HO} \cdot \mathbf{x}^O[t-1] + \mathbf{b}^H) \quad \text{and} \tag{9}$$

$$\mathbf{x}^O[t] = g^O(W^{OH} \cdot \mathbf{x}^H[t] + \mathbf{b}^O) \quad . \tag{10}$$

Since the state of the network is the result of prediction at the previous time step, we say that the TD network uses a predictive state representation. This network can make an accurate prediction only if possible belief states in the environments can be distinguished by the representation; in other words, the set of questions must be sufficient to represent the state.

The training data and condition vector are as follows:

$$\langle \mathbf{t}^O[t], \mathbf{c}^O[t] \rangle = \left\langle \begin{pmatrix} p(y_1)[t+1] \\ \vdots \\ p(y_n)[t+1] \end{pmatrix}, \begin{pmatrix} a(y_1)[t+1] \\ \vdots \\ a(y_n)[t+1] \end{pmatrix} \right\rangle \tag{11}$$

where

$$p(y_i)[t] = \begin{cases} o_j[t] & \text{if } p(y_i) = o_j \ (j \in \{1, \ldots, |\mathcal{O}|\}) \\ x^O_k[t] & \text{if } p(y_i) = y_k \ (k \in \{1, \ldots, i-1\}) \end{cases} . \tag{12}$$

In the original TD networks [Sutton 05], the answer network was represented as a matrix calculation from a *feature vector*, which was given by a fixed function calculated from $\mathbf{x}^O[t-1]$, $\mathbf{a}[t]$, and $\mathbf{o}[t]$. This feature vector corresponds to the hidden layer, $\mathbf{x}^H$, in our formalization. Moreover, we can represent the fixed feature-vector function as fixing W_{HI} and W_{HO}, the incoming connections to the hidden layer and giving specialized output function g_H. For example, the feature vector used in some previous studies [Tanner 05a, Tanner 05b] can be written in the following form (we have omitted the details on W_{HI}, W_{HO}, and g_H):

$$\begin{aligned} \mathbf{x}^H[t] = (&a_1[t]o_1[t], a_2[t]o_1[t], \ldots, a_{|\mathcal{A}|}[t]\,o_1[t], \\ &a_1[t]o_2[t], \ldots\ldots, a_{|\mathcal{A}|}[t]o_{|\mathcal{O}|}[t], \\ &a_1[t]x^I_1[t-1], \ldots, a_{|\mathcal{A}|}[t]x^I_1[t-1], \\ &a_1[t]x^I_{N_I}[t-1], \ldots\ldots, a_{|\mathcal{A}|}[t]\,x^I_{N_I}[t-1])^T . \end{aligned} \tag{13}$$

In the experiments, we use question networks that are mechanically generated as follows. The first $|\mathcal{O}||\mathcal{A}|$ nodes were targeted to predict all observation bit for all possible action: Then, for each prediction node sequentially from y_1, $|\mathcal{A}|$ child nodes are assigned, and conditioned by each possible action. This assignment is repeated until the number of nodes reaches the given limit N_C. Formally, the targets and conditions for all nodes are as follows:

$$\begin{aligned} \langle p(y_1 &), a(y_1 &)\rangle &= \langle o_1 &, a_1 &\rangle \\ \langle p(y_2 &), a(y_2 &)\rangle &= \langle o_1 &, a_2 &\rangle \\ &\vdots \\ \langle p(y_{|\mathcal{O}||\mathcal{A}|} &), a(y_{|\mathcal{O}||\mathcal{A}|} &)\rangle &= \langle o_{|\mathcal{O}|}, &a_{|\mathcal{A}|} &\rangle \\ \langle p(y_{|\mathcal{O}||\mathcal{A}|+1} &), a(y_{|\mathcal{O}||\mathcal{A}|+1} &)\rangle &= \langle y_1 &, a_1 &\rangle \\ &\vdots \\ \langle p(y_{|\mathcal{O}||\mathcal{A}|+|\mathcal{A}|} &), a(y_{|\mathcal{O}||\mathcal{A}|+|\mathcal{A}|} &)\rangle &= \langle y_1 &, a_{|\mathcal{A}|} &\rangle \\ \langle p(y_{|\mathcal{O}||\mathcal{A}|+|\mathcal{A}|+1}), &a(y_{|\mathcal{O}||\mathcal{A}|+|\mathcal{A}|+1}) &\rangle &= \langle y_2 &, a_1 &\rangle \\ &&&\vdots \ . \end{aligned} \tag{14}$$

3.3 Simple Recurrent TD Networks

We propose a modified version of the TD network, a *simple recurrent TD network* (SR-TDN), which has the structure of an SRN with the learning strategy of TD networks. The core idea underlying TD networks is to train a network to predict its own future output, in addition to the observation, through temporal differences. Although TD networks use $\mathbf{x}^O[t-1]$, the output prediction at time $t-1$, as the state representation at time t, it is reasonable to replace the state representation with $\mathbf{x}^H[t-1]$, i.e., the values of the hidden units at time $t-1$, because $\mathbf{x}^H[t-1]$ represents a prototypic data prior to $\mathbf{x}^O[t-1]$. However, note that this replacement causes a slight deviation from the principle of predictive representation that uses state representation grounded on the observable quantities. To distinguish from predictive representations, we say that SR-TDNs use *proto-predictive representations* of states.

This idea can be implemented by incorporating the idea of TD networks into an SRN (Fig. 1c). More precisely, an SR-TDN has the same connectivity as an SRN (eqs.3 and 4). Input and training data for the SR-TDN is the same as a TD network, i.e., Eq. (5) as input, and Eq. (11) as the training data.

4. Experiments

We conducted a series of simulation experiments in various POMDP environments that are used by existing studies..

Figure 3 shows one of these, an 8-state ring world. There are two possible observations, 0 or 1, and two actions are available, L and R. Some of the other environments are taken from the POMDP problem repository [Cassandra 99]. Table 1 summarized the environments.

For all environments, 30 sequences of 2×10^6 observations and actions were generated with a uniform random policy. All network architectures were trained on the same set of sequences; a network was initialized with random connection weights before the beginning of each sequence. The mean squared error of prediction was measured as in Eq. (2) with $L = 10,000$. The parameter T_{back}, the number of backpropagating steps in BPTT, was set to 3.

Figure 4 plots the results of experiments. We can see that all the network architectures made good predictions in simple environments such as (a). However, in complex environments (b–c), the predictions by TD networks became less accurate. The reason may be that the given question network was insufficient for correctly representing the states of these environments. This is likely because a question network is mechanically generated for every possible pair of observations and actions up to a specified number of units ($N_P = 40$), and these environments have a relatively large number of observations and actions. It is noteworthy that the SR-TDNs successfully learned accurate predictions in these environments, using the same, insufficient question networks.

In the graph for the 8-state ring (d), the SR-TDN seems much worse than the TD networks. This is because the SR-TDN failed to learn prediction in 1 out of 30 trials*1. In the other 29 trials, SR-TDN learns equally accurate prediction to the TD network with feature vectors. These results imply that SR-TDNs are unstable due to deviation from the principle of predictive representation. Also note that, in additional experiments with $T_{\text{back}} = 1$, TD networks with feature vectors produced much worse results, while SR-TDNs experienced little change.

*1 Since error was averaged arithmetically and the Y-axis of the graph is log-scaled, 1 failure out of 30 trials made a large difference in the graph.

5. Conclusion

We investigated a new neural network architecture for POMDP learning. A Simple Recurrent TD Networks (SR-TDNs) are a combination of the learning strategy of TD networks and the state representation of SRNs. Through computer simulation experiments, we found that the learning capacity of these architectures are beyond the limitation from the given question network but they showed some unstable behavior in learning, possibly caused by the deviation from the principle of predictive representation.

References

[Cassandra 99] Cassandra, A.: Tony's POMDP file repository page, URL http://www.cs.brown.edu/research/ai/pomdp/examples/index.html (1999)

[Elman 95] Elman, J. L.: Language as a dynamical system, in Port, R. F. and Gelder, van T. eds., *Mind as Motion: Explorations in the Dynamics of Cognition*, pp. 195–223, MIT Press, Cambridge, MA (1995)

[Jaeger 00] Jaeger, H.: Observable Operator Models for Discrete Stochastic Time Series, *Neural Computation*, Vol. 12, No. 6, pp. 1371–1398 (2000)

[James 04] James, M. R. and Singh, S.: Learning and discovery of predictive state representations in dynamical systems with reset, in *Proc. of ICML'04*, p. 53, New York (2004), ACM Press

[Jordan 86] Jordan, M.: Serial order: A parallel distributed processing approach, Technical Report ICS Report 8604, Institute for Cognitive Science, University of California at San Diego, La Jolla, CA (1986)

[Littman 02] Littman, M. L., Sutton, R. S., and Singh, S.: Predictive Representations of State, in *Advances in Neural Information Processing Systems 14*, pp. 1555–1561, MIT Press, Cambridge, MA (2002)

[McCracken 06] McCracken, P. and Bowling, M.: Online Discovery and Learning of Predictive State Representations, in *Advances in Neural Information Processing Systems 18*, pp. 875–882, MIT Press, Cambridge, MA (2006)

[Sutton 05] Sutton, R. S. and Tanner, B.: Temporal-Difference Networks, in *Advances in Neural Information Processing Systems 17*, pp. 1377–1384, MIT Press, Cambridge, MA (2005)

[Tanner 05a] Tanner, B. and Sutton, R. S.: TD(λ) networks: temporal-difference networks with eligibility traces, in *Proc. of ICML'05*, pp. 888–895, New York (2005), ACM Press

[Tanner 05b] Tanner, B. and Sutton, R. S.: Temporal-Difference Networks with History, in *Proc. of IJCAI'05*, pp. 865–870 (2005)

[Wolfe 05] Wolfe, B., James, M. R., and Singh, S.: Learning predictive state representations in dynamical systems without reset, in *Proc. of ICML'05*, pp. 980–987, New York (2005), ACM Press