On-line Discovery of Temporal-Difference Networks

Takaki Makino MAK@SCINT.DPC.U-TOKYO.AC.JP
Division of Project Coordinate, Tokyo University, 5-1-5 Kashiwa-no-ha, Kashiwa-shi, Chiba 277-8568 Japan

Toshihisa Takagi TT@K.U-TOKYO.AC.JP
Database Center for Life Science, Research Organization of Information and Systems, Tokyo 113-0022 Japan

Abstract

We present an algorithm for on-line, incremental discovery of temporal-difference (TD) networks. The key contribution is the establishment of three criteria to expand a node in TD network: a node is expanded when the node is well-known, independent, and has a prediction error that requires further explanation. Since none of these criteria requires centralized calculation operations, they are easily computed in a parallel and distributed manner, and scalable for bigger problems compared to other discovery methods of predictive state representations. Through computer experiments, we demonstrate the empirical effectiveness of our algorithm.

1. Introduction

Predictive representations (Littman et al., 2002; Jaeger, 2000) are a relatively new group of approaches for expressing and learning grounded knowledge about dynamical systems. These approaches represent the state of a dynamical system as a vector of predictions, based on the hypothesis that important knowledge about the world can be represented strictly in terms of relationships between predictions of observable quantities. In the *predictive state representations* (PSRs) introduced by Littman et al. (2002), each prediction is an estimate of the probability of *tests*, defined as some sequence of observations given a sequence of actions. Sutton and Tanner (2005) proposed another approach for predictive representations, namely Temporal-Difference (TD) networks. TD networks are developed as a generalization of PSRs: in TD networks, each prediction is an estimate of the probability or expected value of some function of future predictions, actions and observations. The predictions can be considered as "answers" to a set of "questions" represented in the TD network.

One important problem in the research of predictive representations is the *discovery problem*, that is, the problem of determining the set of questions (or *core tests*) so that the state of a dynamical system is correctly represented by the vector of predictions for these questions. Many of the existing studies on this problem (Rosencrantz et al., 2004; James & Singh, 2004; Wolfe et al., 2005) utilize off-line discovery and learning on PSRs, and therefore they are hardly applicable to both implementing live-interaction agents and finding corresponding activity in the human brain. There is an on-line discovery algorithm (McCracken & Bowling, 2006) for PSR core tests, but it requires complex operations on a large matrix, such as calculation of the maximum linear independent set, and is therefore not suitable for parallel and distributed computing. As far as we are aware, no algorithm has been proposed for discovery of questions in TD networks. Study of a TD-network discovery algorithm that is suitable for parallel distributed computing would contribute not only to research on predictive representations but also to research on cognitive science by providing a hypothesis for the algorithm actually used in the human brain.

In this study, we propose an algorithm for discovering the correct set of tests by incremental node expansion in a TD network. Our key contribution is in the criteria that we have developed for node expansion: a node is expanded when the node is well-known, independent, and has a prediction error that requires further explanation. To check these criteria, our algorithm maintains the average squared error and average variance for each node, and it introduces a dependency detection network. Since none of these criteria requires centralized operations such as calculation of linear independence in the whole representation matrix, they

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008. Copyright 2008 by the author(s)/owner(s).

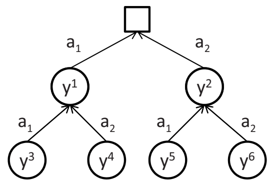


Figure 1. Example TD network we focus in this paper.

are easily computed in a parallel and distributed manner, which is an important property for seeking the algorithm used in the human brain. Although the algorithm has no theoretical guarantee to find the question network (indeed, it is known to be failed in some cases), our simulation experiments demonstrate the empirical effectiveness of our algorithm.

Section 2 reviews PSR and TD network that is necessary to understand our algorithm. Section 3 describes our incremental discovery algorithm. Experiments and results are shown in Section 4. After that, we discuss related work and future directions in Sections 5 and 6.

2. TD Networks

In this section, we make a brief review on TD network, based on the work by Tanner and Sutton (2005a).

The purpose of TD networks is to learn prediction of future observation obtained from the environment. Consider a partially observable environment, which changes its state according to an agent's action $a_t \in \mathcal{A}$ at every time step t, but the agent can only have a partial (and possibly noisy) observation $o_{t+1} \in \mathcal{O}$ of the state. Generally o_t can be a vector consisting of l bits, but in this paper, we consider the case that the observation is a single bit ($l = 1$).

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 the observation from the environment, which denotes the targets for prediction of the node. A link may have a condition, which indicates that the node is a conditional prediction of the target. This set of nodes and links are called the *question network* since each node is some question about the environment.

As in the previous studies (Tanner & Sutton, 2005b; Tanner & Sutton, 2005a), we focus on a subset of TD networks, in which every node has a single target (hereafter, the parent node) and every link is conditioned with an action. Figure 1 is an example of such a TD network. The node y^1 predicts the observation at the next step if action a_1 is taken. The node y^4 predicts the value of the node y^1 at the next step if action a_2 is taken, and so on.

To provide an answer for the questions asked by the question network, each node in a TD network works also 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. However, to provide an accurate answer, the set of nodes have to be a sufficient representation for the environmental state; in other words, a correct set of questions have to be posed by the question network. The focus of this paper is the discovery of the question network, i.e., to find the structure of the question network that is sufficient for prediction.

Formally, we denote the prediction for node i at time step t as $y_t^i \in [0,1]$, $i = 1, \ldots, n$. The prediction vector $\mathbf{y}_t = (y_t^1, \ldots, y_t^n)^T$ is given by the answer network:

$$\mathbf{y}_t = \sigma(\mathbf{W}_t \mathbf{x}_t) \quad , \tag{1}$$

where $\mathbf{x}_t \in \Re^m$ is a feature vector, $\mathbf{W}_t$ is a $n \times m$ matrix of modifiable weights, and σ is the S-shaped logistic function $\sigma(s) = (1 + e^{-x})^{-1}$.

The feature vector is a function of the preceding action, observation, and node values:

$$\mathbf{x}_t = \mathbf{x}(a_{t-1}, o_t, \mathbf{y}_{t-1}) \in \mathbb{R}^m \quad . \tag{2}$$

We used the similar form of feature vector as appeared in the work of Tanner and Sutton (2005a); in our experiments, where two actions (L and R) are possible,

$$\mathbf{x}(a,o,\mathbf{y}) = \begin{cases} (o, 1-o, y^1, \ldots, y^n, 0, \ldots, 0)^T & a = \mathrm{L} \\ (0, \ldots, 0, o, 1-o, y^1, \ldots, y^n)^T & a = \mathrm{R} \end{cases} \tag{3}$$

This is equivalent to separate W for each action.

The question network, which gives the target of predictions in terms of the node values at the next time steps, is represented by a $n \times (n + l)$ matrix Z^a and a vector c. Without eligibility traces, the vector of the target values is

$$\mathbf{z}_{t-1} = \mathbf{c}_t \odot \mathbf{Z} \begin{pmatrix} \mathbf{y}_t \\ o_t \end{pmatrix} + \bar{\mathbf{c}}_t \odot \mathbf{y}_{t-1} \quad , \tag{4}$$

where $\odot$ is element-by-element multiplication, and each element of $\mathbf{Z}$, $\mathbf{c}$ and $\bar{\mathbf{c}}$ is:

$$z^{ij} = \begin{cases} 1 & y^i \text{ is the parent node of } y^j \\ 0 & \text{otherwise} \end{cases} \quad , \tag{5}$$

$$c_t^i = \begin{cases} 1 & a_t \text{ satisfies the node } y^i\text{'s condition} \\ 0 & \text{otherwise} \end{cases} \quad , \tag{6}$$

$$\bar{c}_t^i = 1 - c_t^i \quad . \tag{7}$$

The elements of Z are assigned so that $z_{t-1}^i = y_t^{p(i)}$ for any i with $c_t^i = 1$, where $p(i)$ is the parent node of i.

On the other hand, if $c_t^i = 0$, $z_{t-1}^i = y_{t-1}^i$, and the TD error of the node becomes zero so that only the weights for the nodes that satisfy the condition are updated.

In this study we employ eligibility traces (Tanner & Sutton, 2005a), which is a technique to accelerate learning in TD-error learning by incorporating further prediction into the learning target. In a forward view,

$$\mathbf{z}_{t-1} = \mathbf{c}_t \odot \mathbf{Z}(\lambda \mathbf{z}_t + (1-\lambda)\mathbf{y}_t) + \bar{\mathbf{c}}_t \odot \mathbf{y}_{t-1} \quad , \quad (8)$$

where $\lambda \in [0, 1]$ is a parameter that controls the balance of temporally distant results in the learning target. When $\lambda = 0$, eq. (8) is equivalent to eq. (4), and no eligibility traces are used.

However, this formula recursively contains future values of z, and it is not easy to be calculated on-line. Tanner and Sutton (2005a) proposes an algorithm that performs on-line update of the weight vector to make the equivalent update as (8). Then each component w_t^{ij} of $\mathbf{W}_t$ is updated by the learning rule:

$$\begin{aligned} w_{t+1}^{ij} &= w_t^{ij} + \alpha(z_t^i - y_t^i)\frac{\partial y_t^i}{\partial w_t^{ij}} \\ &= w_t^{ij} + \alpha(z_t^i - y_t^i)y_t^i(1-y_t^i)x_t^j \quad , \quad (9) \end{aligned}$$

in which the second line is derived from eq. (1).

Roughly, the operation of a TD network proceeds by repeating the following steps: (1) Choose an action a_{t-1} and receive an observation o_t from the environment. (2) Operate the answer network, i.e., calculate feature vector $\mathbf{x}_t = \mathbf{x}(a_{t-1}, o_t, \mathbf{y}_{t-1})$ and obtain the new predictions $\mathbf{y}_t = \sigma(\mathbf{W}_t \mathbf{x}_t)$. (3) Use the question network to obtain the target value for the previous predictions $\mathbf{z}_{t-1} = \mathbf{z}(\mathbf{y}_t, o_t)$, and update the weights W according to the TD error $\mathbf{z}_{t-1} - \mathbf{y}_{t-1}$. For details, readers should consult the original paper of the TD network (Sutton & Tanner, 2005) to see subtle points, such as the precise order of calculation.

3. On-line Discovery Algorithm

We propose an algorithm that performs on-line discovery of the question network. Our algorithm starts with the minimal network, which consists of the observation node and a set of prediction nodes for the observation nodes, one for each action. During learning, the algorithm grows the network by *expanding* leaf nodes by adding a set of prediction nodes for the node. Intuitively, a node is expanded when the following three criteria holds:

1. **The node is well-known**: The agent has sufficient experience to learn the node. This criterion prevents relatively unexplored nodes to be expanded.
2. **The node is independent**: The prediction of the node cannot be calculated in terms of other, formerly known node values. In terms of PSRs, the node represents a core test. This criterion avoids redundant expansion of the nodes.
3. **The node's error requires further explanation**: The node's prediction error is not smaller than expected from the prediction error of the node's parent node. This criterion chooses the node that has the best descriptive power for the error in the parent node, and stops expansion when unpredictability is solved.

In the following, we first describe the variables that our algorithm maintains to check these criteria, and we present more detailed conditions for the criteria.

3.1. Variables

3.1.1. Dependency Detection Network

Our algorithm uses a dependency detection network, which tries to represent a prediction of the node y^i in terms of observation o and values of the nodes with younger index y^j $(j < i)$. If the network succeeds to represent y^i with small error, we can see that y^i is dependent to the predictions with younger index (namely, not a core test of PSRs), and exclude it from the candidate of node expansion. Otherwise, we can assume that y^i is an independent node. Note that it corresponds to the core test in non-linear PSRs (Rudary & Singh, 2004) because the nodes in TD networks correspond to e-tests in non-linear PSRs (Sutton & Tanner, 2005) and we use sigmoidal function in the answer network.

Formally, the dependency detection network is represented by $\mathbf{D}_t$, a $n \times (n+l)$ matrix of modifiable weights. In the matrix d^{ij} is restricted to zero if $i \geq j - l$. The output of the network is $\mathbf{d}_t = \sigma(\mathbf{D}_t \binom{o_t}{\mathbf{y}_t})$. The network is trained so that $\mathbf{d}_t$ is close to $\mathbf{y}_t$; in other words, the network tries to represent a prediction of the node y^i in terms of observation o and predictions with nodes with smaller index y^j $(j < i)$. Since the indices of the nodes are numbered in order, newly expanded nodes are always given higher indices, and are not used by the dependency detection network for describing older nodes with lower indices.

Each component d_t^{ij} of $\mathbf{D}_t$ is updated by the learning rule similar to eq. 9:

$$d_{t+1}^{ij} = d_t^{ij} + \alpha_D(y_t^i - d_t^i)\frac{\partial d_t^i}{\partial d_t^{ij}} \quad (10)$$

$$= d_t^{ij} + \alpha_D(y_t^i - d_t^i)d_t^i(1-d_t^i)y_t^j \quad (11)$$

But the update is limited in the area $i < j - l$. We assign the learning rate of the dependency detection network α_D to be larger than that of the answer network α to allow the dependency detection network to track changes in the calculated node values during the learning of the answer network (as long as the values are dependent).

3.1.2. Average Errors

Our algorithm gathers statistical information about the prediction and error of the TD network and the dependency detection network. To allow on-line learning, the statistical variables are calculated in forms of exponential moving averages:

$$\mathbf{y}_{t+1}^{\mathrm{LERR}} = \rho_2\, \mathbf{y}_t^{\mathrm{LERR}} + (1-\rho_2)((\mathbf{z}_t - \mathbf{y}_t) \odot (\mathbf{z}_t - \mathbf{y}_t)) \quad (12)$$
$$\mathbf{d}_{t+1}^{\mathrm{SERR}} = \rho_1\, \mathbf{d}_t^{\mathrm{SERR}} + (1-\rho_1)((\mathbf{y}_t - \mathbf{d}_t) \odot (\mathbf{y}_t - \mathbf{d}_t)) \quad (13)$$
$$\mathbf{d}_{t+1}^{\mathrm{LERR}} = \rho_2\, \mathbf{d}_t^{\mathrm{LERR}} + (1-\rho_2)((\mathbf{y}_t - \mathbf{d}_t) \odot (\mathbf{y}_t - \mathbf{d}_t)) \quad (14)$$

where $\mathbf{d}_t^{\mathrm{SERR}}$ is a short-term average of squared prediction errors, $\mathbf{d}_t^{\mathrm{LERR}}$ is a long-term average of squared prediction errors, and $0 < \rho_1 < \rho_2 < 1$ is a temporal factor. When a node y^i is added to the network, statistical variables are initialized as $y^{\mathrm{LERR}\ i} = d_t^{\mathrm{SERR}\ i} = d_t^{\mathrm{LERR}\ i} = 1.0$so that the variables show larger errors during the initial period of the node.

Without eligibility traces, the target variable $\mathbf{z}_t$ is available at time $t+1$, so these parameters are easily calculated on-line. However, since eq. (12) contains quadratic term for $\mathbf{z}_t$, on-line calculation technique with eligibility traces such as used in Tanner and Sutton's work (2005a) cannot be used directly. Our implementation keeps the record of last k steps of node values, where k is the maximum depth of the current question network, and calculates errors of $\mathbf{y}_t$ and $\mathbf{d}_t$ at time $t + k$.

3.2. Expansion Criteria

Using these variables, we check the criteria described in the beginning of Section 3 as follows. The criteria are checked for every time step. Since all criteria are described on exponential moving averages, the precise timing of criteria check and node expansion is not important; it should be inserted somewhere in the TD(λ) network learning algorithm (Tanner & Sutton, 2005a). The expansion criterion are designed to avoid redundant expansion as much as possible because no shrinking criterion is given.

3.2.1. The node is well-known

To avoid expanding relatively unexplored nodes, we use the following criteria to determine whether the node y^i is well-known.

- Learning error is not in a decreasing trend (we assume the error is always decreasing during initial learning phase).
- Learning error of the node gets smaller compared with that of its parent node.

In formal representation,

$$d_t^{\mathrm{SERR}\ i} \geq d_t^{\mathrm{LERR}\ i} \quad \mathit{and} \quad (15)$$
$$d_t^{\mathrm{LERR}\ i} \leq d_t^{\mathrm{LERR}\ p(i)} \quad . \quad (16)$$

If $p(i)$ is the observation bit, then $d_t^{\mathrm{LERR}\ p(i)}$ is considered as 1 (the largest possible value).

Eq. 15 works because these variables, representing moving averages of errors, are initialized with the highest possible value (see Section 3.1.2). Thus it is expected that the short-term average error variables stay lower than the long-term ones until the end of the initial learning period, in which the learning error is constantly decreasing.

Eq. 16 is usually satisfied with a plenty amount of experience because the dependency network has no connection from a child node to a parent node. The parent node always has fewer inputs in the dependency network than the child node; if the child node has an unexplained dependency (large $d_t^{\mathrm{LERR}\ i}$), it is likely that the parent node also has an unexplained dependency.

3.2.2. The node is independent

To prevent dependent (non-core test) nodes to be expanded, we require that the learning error in the dependency detection network is not small.

$$d_t^{\mathrm{LERR}\ i} \geq \theta_1 \quad (17)$$

θ_1 is a threshold parameter that controls the requirement for independence.

When all the nodes that match this criterion are expanded, then all the leaf nodes in the question network becomes dependent nodes, and no further expansion occurs. Thus, this criterion is equivalent to the assumption that independent (core test) nodes does not exist as a child of dependent (non-core test) nodes.

3.2.3. The node's error requires further explanation

If the prediction error of a node is larger than expected from its parent node, it is reasonable to require further prediction on the node to reduce the error. Otherwise, we can infer that the error in the parent node has other causes (e.g. another prediction node with different action conditions has large prediction error), and further

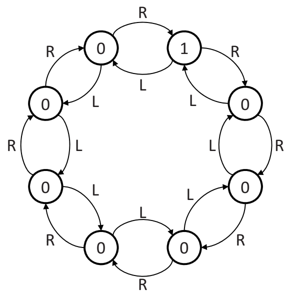


Figure 2. 8-state ring world. The digit in a node represents observation from the state, and edge labels denote actions.

Table 1. Tests for various θ_1 and θ_2 values in the 8-state ring world. Values are [final number of nodes] / [steps ($\times 10^4$) until learned (MSE reduces less than 10^{-4})].

$\theta_1 \backslash \theta_2$	0.005	0.0075	0.01	0.0125	0.015
0.001	18/47	20/46	16/49	6/–	6/–
0.002	18/49	18/48	18/51	6/–	6/–
0.003	18/53	18/55	16/55	6/–	6/–
0.004	18/54	18/57	16/58	6/–	6/–
0.005	18/59	18/59	16/60	6/–	6/–

prediction for the node is less important to reduce the error of the final prediction for the observation.

In case that the parent node $p(i)$ is purely probabilistic, error distribution of the $p(i)$'s child nodes is proportional to the probability that the conditions of the nodes are matched; the following condition checks that the error of the node is greater than that:

$$y_t^{\mathrm{LERR}\ i} \geq \gamma \frac{\#y^i}{\#y^{p(i)}} y_t^{\mathrm{LERR}\ p(i)} + \theta_2 \quad , \qquad (18)$$

where $\#y^i$ is the frequency that the conditions on the chain from the observation bit to node y^i is matched (thus, $\frac{\#y^i}{\#y^{p(i)}}$ is the relative probability that the condition of the node is matched). If $p(i)$ is the observation bit, then $y_t^{\mathrm{LERR}\ p(i)}$ is assumed to be zero. θ_2 is a threshold parameter that controls tolerance for noise.

4. Experiments

To test the efficiency of our algorithm, we conducted a series of computer experiments on n-state ring world ($n = 5, 8$) (Tanner & Sutton, 2005b). Figure 2 illustrates 8-state ring world. There are two observations, 0 or 1, and two actions, L and R.

Since the ring worlds contains only deterministic state transitions and observations, we also tested our algorithm on some standard probabilistic POMDP environments taken from repository (Cassandra, 1999). Among them, environments with one-bit observation are used, and adapted to non-reward situation (we followed a previous work that describe details; McCracken, 2005).

We generated a sequence of experience by a uniform random policy and applied our algorithm online. Through all experiments we used $\rho_1 = 0.99$, $\rho_2 = 0.999$, $\alpha_D = 0.2$, $\lambda = 0.9$, $\theta_1 = 0.0045$, and $\theta_2 = 0.0005$. α is initialized with 0.1, and after 800,000 steps, α is halved with every 100,000 step. We measured the error of the prediction for the selected action, compared to the oracle (observation probability calculated from the structure of the environment), and the mean squared errors for every 10,000 steps are plotted.

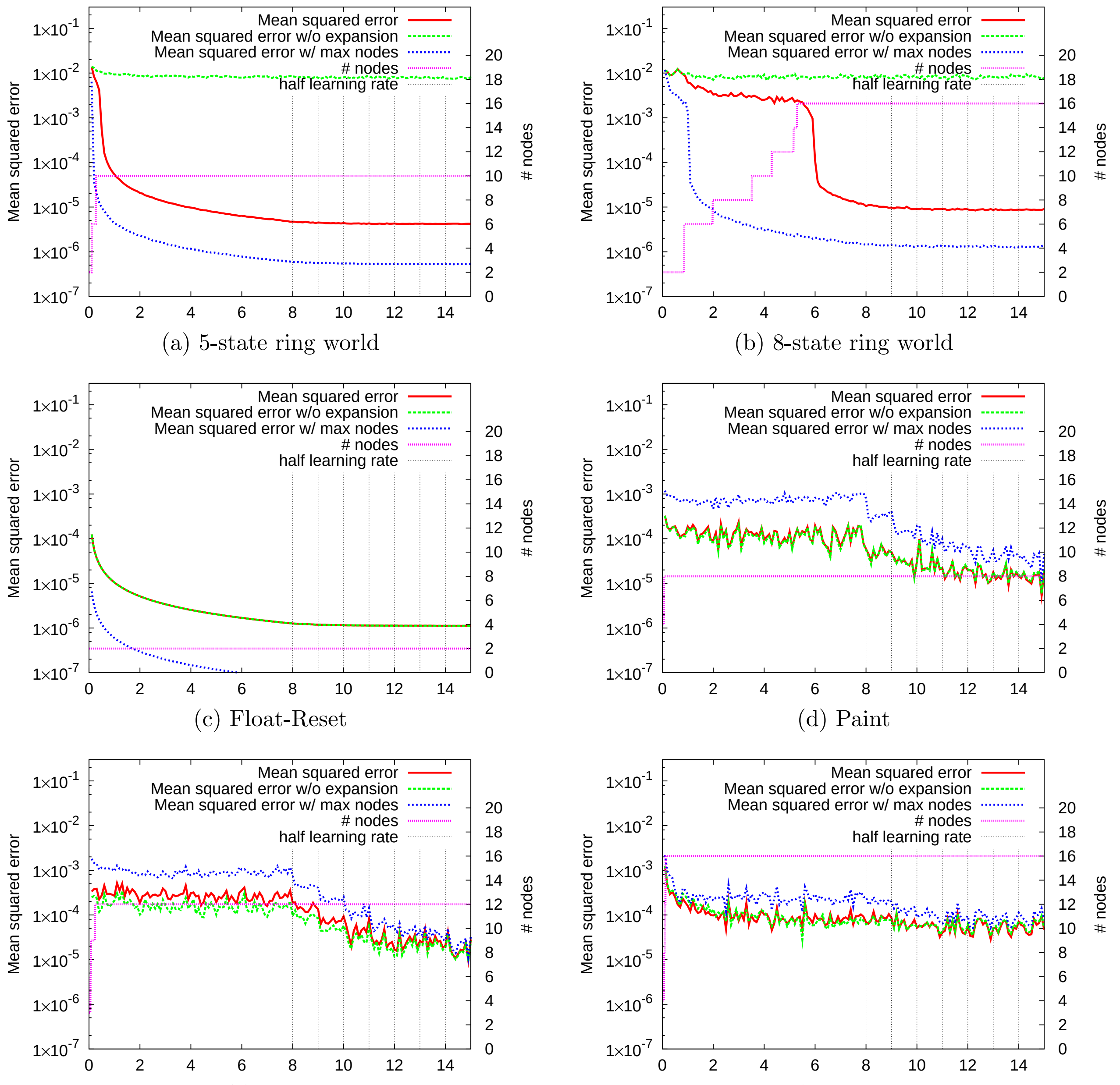
Figures 3(a) and 3(b) are the results in 5- and 8-state ring worlds. We can see that the number of nodes in the TD network increases as a result of node expansion until the prediction error decreases. This indicates that nodes in the TD-networks are expanded only when it is required.

We made additional tests with various parameters θ_1 and θ_2 on the 8-state ring world (Table 1). We found that θ_2 affects the final number of nodes. With larger θ_2, algorithm failed to make a required node expansion; with smaller θ_2, the algorithm made some spurious node expansions (though the learning was successful). On the other hand, θ_1 mainly affects the learning time, but less related to the number of nodes.

Figures 3(c) to 3(f) show the results of our algorithm in other well-known POMDP environments. Our algorithm has successfully learned predictions in all cases. However, we found that the initial form of the TD network without node expansion can learn equally well (compared to the case started with a large TD network, which is mechanically generated by expanding all nodes), due to the high generalization capacity of TD-network with sigmoid function. Thus these experiments are not helpful for evaluating our discovery algorithm. However, we see that some node expansions are occurred in these experiments. This indicates that our algorithm sometimes makes spurious node expansions, especially in these probabilistic environments.

We also evaluated the criteria we selected in terms of the discovered question network for the 8-state ring world. Figure 4 compares the mean square errors and the number of nodes in the discovered network with various settings of criteria (in these experiments α is kept constant to 0.1). Although the prediction error approaches to zero on all conditions, increase in the number of nodes is observed if any one of the criteria is removed.

Figure 5 further examines the difference of the discovered TD network. In the TD network discovered using all of the criteria (Figure 5(a)), all the expanded nodes



(a) 5-state ring world (b) 8-state ring world

(c) Float-Reset (d) Paint

(e) Tiger (f) Network

Figure 3. Results of experiments. X-axis shows time steps ($\times\ 10^5$)

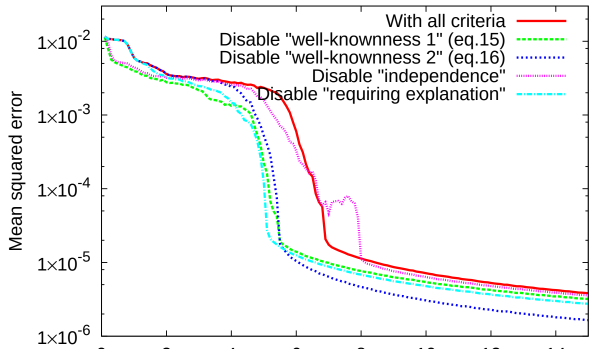
are independent of each other, thus the discovered network is the best network for the 8-state ring world that can be discovered by our algorithm. This shows clear contrast to TD network discovered solely by dependency detection network, shown in Figure 5(b). Although this network has correctly learned to predict the 8-state ring world, some spurious node expansions are observed. This indicates that the dependency detection network is not powerful enough to choose the right node to be expanded, and shows importance of other criteria in our algorithm.

5. Discussion

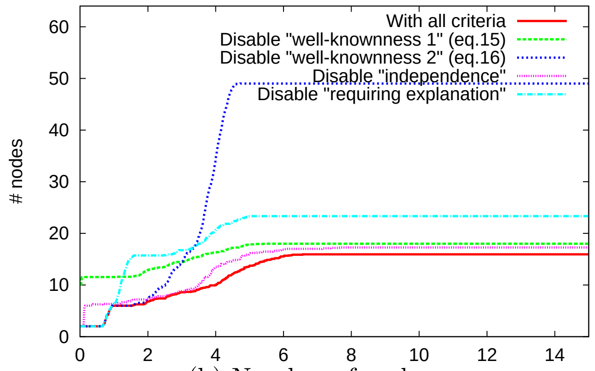
The algorithm we have presented has some limitations. It seems unavoidable because the algorithm has to make inference using an incomplete question network. In this section, we discuss some of the limitations that arose in our approach.

5.1. Deep Dependency

In our algorithm, Criterion 3 (requires more explanation) are devoted for distinguishing apparently unpredictable events, caused by an incomplete question net-

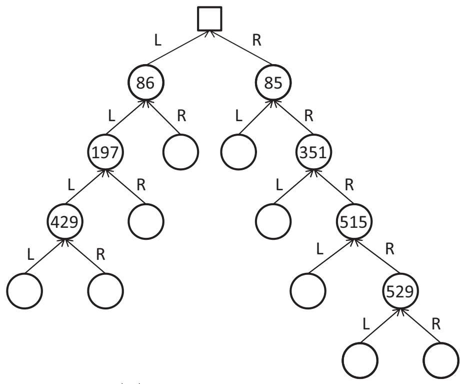


(a) Mean squared error

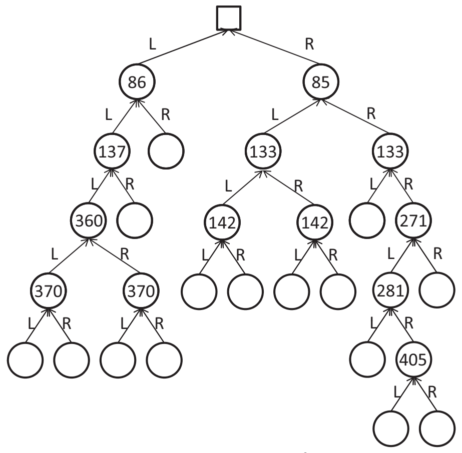


(b) Number of nodes

Figure 4. Results of experiments (average of 30 trials) on 8-state ring world with disabled criteria.



(a) With all criteria



(b) Without criterion "requires further explanation"

Figure 5. An example of discovered TD network for 8-state ring world. The number in a node denotes the time the node is expanded ($\times 10^3$).

work, from inherently random events. However, we can imagine cases in which this criterion does not work well, especially when the observation depends only on the actions several steps before.

As an example, suppose an n-step delay world: $\mathcal{A} = \mathcal{O} = \{0, 1\}$, and the agent observes its own action with n-step delay. If the agent chooses actions randomly, then the observation is also random string. Moreover, unless the depth of the question network reaches n, the agent can predict nothing. For $n \geq 2$, the nodes in a question network are likely to fail satisfying Criterion 3, and as a result, the algorithm fails to achieve the correct test set for the environment.

A partial solution may be to introduce active learning (planned exploration). In the example above, if the agent could adopt some biased choice of actions for a while, one would observe informative result that might satisfy Criterion 3.

5.2. Selection of Expanding Node

The algorithm always expands the node that requires more explanation, but it is possible that the node is not the best one to be expanded. The algorithm depends on an assumption that, if there is a better node to be expanded, the node also satisfies the expansion criteria sooner or later. We need to work for some theoretical support for the assumption. However, if the assumption is correct, the algorithm may perform some spurious expansion due to its distributed-processing nature. A complete solution would require either centralized processing or node shrinking.

5.3. Parameter Selection

The algorithm depends on a number of parameters. Our selection of parameters seems working for the tested problems, but not guaranteed for others. In particular, the algorithm is sensitive to θ_2, a threshold value used in Criterion 3 for distinguishing apparently unpredictable events from inherently random events. Due to the limitation of our approach, it seems impossible to provide a universal parameter set for producing the minimum network for any environment; a better solution would be to use lower thresholds to overgenerate the question network and shrink it afterwards.

5.4. Handling Wider Observation

In this paper, we considered only the simple case with one observation bit ($l = 1$). When $l \geq 2$, it would be necessary to consider sets of nodes that share a common action conditions and associated to different observations, and to expand all nodes in a set simultaneously. In addition, it is necessary to extend our criteria have to handle the node sets.

6. Related Work

McCracken and Bowling (2006) proposes the on-line discovery and learning algorithm for predictive state representation. However, their algorithm has to keep substantially long (in their example, 1000 steps) history in memory and calculate linear independency, and requires quite complex efforts to normalize the probability in their representation. On the other hand, our algorithm requires only a small amount of memory (up to the size required for eligibility traces) and, thanks to the sigmoidal function used in the TD network, no effort is required to normalize the result. We argue that these differences cause larger difference in calculation costs in a complex environment.

7. Summary

We presented an algorithm for on-line, incremental discovery of temporal-difference (TD) networks. The key contribution is the establishment of criteria to expand a leaf node in TD network: the algorithm expands a node when the node is well-known, independent, and has a prediction error that requires further explanation. Since none of these criteria requires centralized calculation operations, they can be computed in a parallel and distributed manner. Through computer experiments on n-state ring worlds, we demonstrated the empirical effectiveness of our algorithm.

Among the future work the most important is to evaluate our algorithm on various environments for comparison with other discovery algorithms. Agent planning with TD network should be also studied to combine with our algorithm for developing an agent that explores environments (Bowling et al., 2006).

Acknowledgments

I'd like to thank Prof. Steven Kraines for his kind help. This research is partially supported by the MEXT Grant-in-Aid for Young Scientists (B) (20700126).

References

Bowling, M., McCracken, P., James, M., Neufeld, J., & Wilkinson, D. (2006). Learning predictive state representations using non-blind policies. *In Proc. of ICML '06* (pp. 129–136). ACM Press.

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

Jaeger, H. (2000). Observable operator models for discrete stochastic time series. *Neural Computation*, *12*, 1371–1398.

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

Littman, M. L., Sutton, R. S., & Singh, S. (2002). Predictive representations of state. In *Advances in neural information processing systems 14*, 1555–1561. MIT Press.

McCracken, P. (2005). An online algorithm for discovery and learning of predictive state representations. Master's thesis, University of Alberta.

McCracken, P., & Bowling, M. (2006). Online discovery and learning of predictive state representations. In *Advances in neural information processing systems 18*, 875–882. MIT Press.

Rosencrantz, M., Gordon, G., & Thrun, S. (2004). Learning low dimensional predictive representations. *Proc. of ICML '04* (p. 88). ACM Press.

Rudary, M. R., & Singh, S. (2004). A nonlinear predictive state representation. In *Advances in neural information processing systems 16*. MIT Press.

Sutton, R. S., & Tanner, B. (2005). Temporal-difference networks. In *Advances in neural information processing systems 17*, 1377–1384. MIT Press.

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

Tanner, B., & Sutton, R. S. (2005b). Temporal-difference networks with history. In *Proc. of IJCAI'05*, 865–870.

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