Engineering

Industrial & Management Engineering fields

Okayama University *Year* 2004

A double layered state space construction method for reinforcement learning agents

Hisashi Handa
Okayama University

SICE Annual Conference in Sapporo, August 4-6, 2004
Hokkaido Institute of Tecnology, Japan

A Double Layered State Space Construction Method for Reinfrocement Learning Agents

H. Handa

Okayama University, Tsushima-Naka 3-1-1, Okayama 700-8530, JAPAN
handa@sdc.it.okayama-u.ac.jp

Abstract: In this paper, we propose a new double-layered state space construction method which consists of Fritzke's Growing Neural Gas algorithm and a class management mechanism of GNG units. The classification algorithm yields a new class by referring to anticipation error, anticipation vectors of an originated class, and anticipation vectors GNG units belonging in the originated class.

Keywords: Reinforcement Learning, Growing Neural Gas, Incremental State Space Construction

1. Introduction

Reinforcement learning is one of the most active research areas in intelligent systems[1]. Reinforcement learning agents try to maximize the total amount of reward in the future through their interaction with a complex and uncertain environment. The object of reinforcement learning agents is to discover effective policy how agents decide actions against any perceptual inputs in order to receive the most reward via their trying.

In order to constitute reinforcement learning agents with continuous inputs, it is very important to design adequate state spaces for the agents. However, the design of the state space is terribly difficult since we have to take account info the characteristics of input-output map of the agents. Moreover, such design process force us to tune the parameters for acquiring the adequate state space with a lot of "trials and errors". That is, if the designer of the agents prepare fine-grained state space, the agents have to learn huge search space due to the large number of states. On the other hand, if coarse-grained state space is given to the agents, the perceptual alias problem is occurred: sensory inputs which should be recognized as the different states are recognized as the same state.

Recently, state construction methods which autonomously generate state spaces have been broadly studied. Most of them depends on the reward signal or state values [2, 3, 4]. The researches of such studies exhibited the effectiveness of their own approach. However, we think the most important property of the state space for the reinforcement learning agents is the situatedness[5]: Agents should be able to describe their surroundings well through their experience, i.e., their perception-action series. That is, agents should know the effects of their actions – the changes of sensory inputs through their actions. Hence, we propose a double layered state space construction method, which consists of Fritzke's Growing Neural Gas Algorithms (GNG) [6] and a class management mechanism (CMM), based upon agent's anticipated sensory inputs. The GNG condenses sensory inputs and learns which area is frequently sensed. On the other hand, the CMM assign class labels onto the references of GNG nodes by referring to anticipation vectors of sensory inputs at the next time step and anticipation errors. The proposed method can incrementally constitute state space of agents while reinforcement learning algorithms learn the policy for the agents.

2. Growing Neural Gas Algorithm

The Growing Neural Gas (GNG) algorithm is a type of competitive learning neural network which consists of nodes and edges [6]. The edges represent adjacency relationships between the nodes. A weight vector of which the dimension is the same as the input vector for neural networks is associated to each node. The network topologies of neural networks in the GNG is changed adaptively: a node is added for reducing accumulated errors and is deleted when no adjacent node exists. Furthermore, the adjacency relationship between the nodes, which is represented by the edge, is dynamically changed in accordance with activation frequencies of the node and its neighboring nodes. In this paper, we adopt the GNG to learn the input distribution and to condense the input vector, which is an n-dimensional input vector, to discrete sets, i.e., nodes (neurons).

The learning procedure of the GNG algorithms is described as follows:

1. An initial network, which consists of two nodes a and b having weight vectors w_a and w_b, respectively, which are randomly generated, and an edge between the two nodes is constituted. The age of the edge is set to be 0.
2. Find the nearest node s_1 and the second nearest node s_2 for an input vector ξ:
$$s_1 = \underset{s_i}{\operatorname{argmin}} |\xi - w_{s_i}|$$
$$s_2 = \underset{s_i, s_i \neq s_1}{\operatorname{argmin}} |\xi - w_{s_i}|,$$
where s indicates a set of nodes.
3. The age of all edges connected to the node s_1 is incremented by 1.
4. The accumulated error $E(s_1)$ of the node s_1 is accumulated according to the following equation:
$$\Delta E(s_1) = \|w_{s_1} - \xi\|^2$$

PR0002/04/0000-2698 ¥400

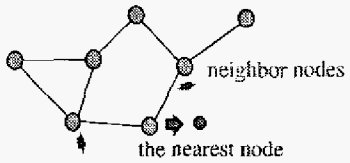

Figure 1: A depiction of the weight modification of GNG nodes

5. Move the weight vector w_{s_1} of the node s_1 to the interior division of the input vector ξ for which the interior fraction is ϵ_b. In addition, all nodes connected n to the node s_1 are moved to the interior division of the input vector ξ for which the interior fraction is ϵ_n:

$$\Delta w_{s_1} = \epsilon_b(\xi - w_{s_1})$$
$$\Delta w_n = \epsilon_n(\xi - w_n),$$

wh'ere $0 < \epsilon_n < \epsilon_b < 1$. Hence, the nearest node s_1 approaches more than the neighboring nodes, as depicted in Fig. 1.

6. If an edge exists between nodes s_1 and s_2, the age of the edge is reset to 0. Otherwise, a new edge of age 0 is added between the nodes.

7. Edges of age greater than a_{max} are removed. As a consequence, nodes which are not connected to other nodes are also removed.

8. At every λ step, a new node is added to the network.

9. The accumulated errors for all nodes are decreased by multiplication by a constant parameter d defined in advance.

10. Go back to 2) if terminal conditions have not held.

The node addition in the step 8) in the above procedure is carried out as follows:

8a) First, find a node q with the maximum accumulated error.

8b) Next, find a node f which has the maximum accumulated error among all nodes connected by the node q. A new node r is then generated at the midpoint between the nodes q and f, that is, the weight vector of the node r is defined as follows:

$$w_r = 0.5 \times (w_q + w_f)$$

8c) The edge between nodes q, f is removed. Instead, two new edges, i.e. the edge between nodes q, r and the edge between nodes r, f, are added to the network.

8d) The accumulated error of the nodes q, f is reduced by multiplication by a predefined constant value η. In addition, the accumulated error of the node r is initialized as the accumulated error $E(q)$ of the node q:

$$\Delta E(q) = -(1-\eta)E(q)$$
$$\Delta E(f) = -(1-\eta)E(f)$$
$$E(r) = E(q)$$

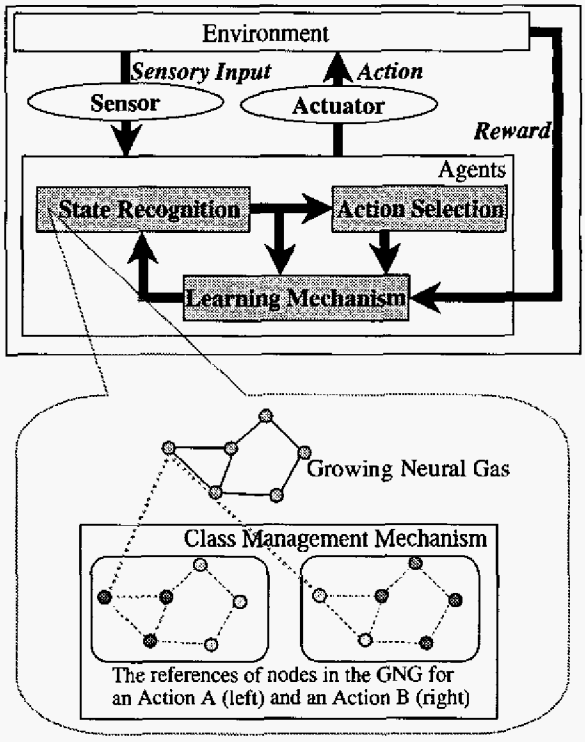

Figure 2: An overview of the proposed method

3. Double Layered State Space Construction Method for Reinforcement Learning Agents

3.1 An Overview of the Proposed Method

As depicted in Fig. 2, the proposed method serves as "the state recognition unit" which maps continuous n-dimensional inputs to discrete states. The balloon area in this figure shows the constitution of the proposed method. In the proposed method, the Growing Neural Gas (GNG) module, which is identical to the conventional GNG algorithms mentioned in the previous section, learns the topology of the input space of the reinforcement learning agents. The Class Management Mechanism (CMM) module administrates the lookup table from nodes in the GNG module to states presented to the action selection module in the reinforcement learning agent. Because the GNG module is identical to the conventional GNG algorithms, we will introduce the CMM module in the next subsection.

3.2 Class Management Module

The CMM module has several groups of the references of the GNG nodes. Each group corresponds to each action of the agents. Hence, the number of groups is the same as the number of actions. In addition, the CMM module has several classes (states) for each group. The classes in a group (action) a consist of the state-action value of each class $V(s, a)$,

an anticipation vector $A_{s,a}$ of the class s, and an anticipated error $E_A(s,a)$. Each reference u in a group (action) a has an anticipation vector $A_{u,a}$, and an index of the member class (state) s. By referring to the anticipation error of the class, the CMM module decides whether a new class is generated. The following paragraphs describe the calculations in the references, the calculations in the classes, and the class management of the CMM module.

Reference-level behavior is described as follows: The anticipation $A_{u,a}$ of a reference u belonging in a group (action) a indicates an n-dimensional vector which estimates the sensory changes by taking the action a in which the reference u is activated. Suppose that j denotes the number of occurrences of situations that take an action a in which the reference u is activated. Moreover, let δ_i be a sensory difference between the time step $\xi(t^j)$, in which the i^{th} occurrence is observed, and its next time step is $\xi(t^i+1)$, i.e. $\delta_i = \xi(t^i+1) - \xi(t^i)$. The anticipation vector $A_{u,a}$ is incrementally updated as follows:

$$\Delta A_{u,a} = \epsilon_a(\delta - A_{u,a}), \tag{1}$$

where ϵ_a denotes a predefined constant value.

The calculation in the class level is carried out as follows: The anticipation vector $A_{c,a}$ is updated by the same means in equation (1):

$$\Delta A_{c,a} = \epsilon_a(\delta - A_{c,a}),$$

Moreover, the anticipation error is accumulated for each occurrence:

$$\Delta E_A(c,a) = \beta(\|A_{c,a} - \delta_j\|^2 - E_A(c,a)), \tag{2}$$

where β is a predefined constant parameter. The method by which to update the state-action values of class (state) depends on the reinforcement learning algorithms. However, we can adopt various kinds of conventional reinforcement learning Algorithms, such as SALSA, TD(λ), and Profit Sharing, to combine with the proposed method, in this paper, we employ the Q-Learning Method to update the state action values [7]:

$$\Delta V(s_t,a_t) = \alpha(R_{t+1} + \gamma \max_a V(s_{t+1},a_{t+1}) - V(s_t,a_t)),$$

where the subscript t indicates the time step. In the calculation of $\max_a V(s_{t+1},a_{t+1})$, s_{t+1} corresponds to classes (states) in all groups (actions), for which the reference refers the same GNG node activated in the time step $t+1$.

The CMM module, at every predefined interval, divides a class, such that its anticipation error is the highest among all classes and is greater than the threshold, into two new classes. The division procedure is described as follows:

1. Find a class c_1 which has the maximum anticipation error.
2. If the anticipation error $E_A(c_1,a)$ of the class c_1 is greater than ε_c, go to step 3). Otherwise, exit this division procedure.

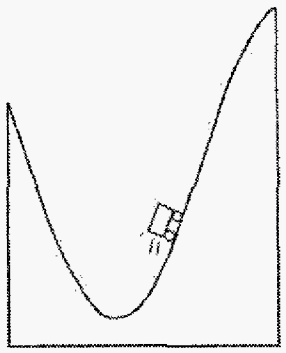

Figure 3: A depiction of the mountain-car task

3. Find a reference u_f for which the anticipation vector is the farthest from the anticipation vector $A_{c_1,a}$ of c_1 among all references which have the index to the class c_1.
4. Generate a new class c_n and move the reference u_f from class c_1 to c_n.
5. For each reference u' in the class c_1, which is connected to references in the class c_n, if the distance between the anticipation vectors $A_{u',a}$ and $A_{c_1,a}$ is longer than the distance between the anticipation vectors $A_{u',a}$ and $A_{u_f,a}$, then the node u' is moved to class c_n.
6. The previous step is recursively applied while the newly moved references exist.
7. The anticipation error $E_A(c_n,a)$ of the class c_n is set to be 0. The anticipation error $E_A(c_1,a)$ of the class c_1 is multiplied by a constant value g. Moreover, the anticipation vectors $A_{c_n,a}$ and $A_{c_1,a}$ are set to be the averaged vectors of the anticipation vectors in the classes c_n and c_1, respectively.

4. Computational Simulation

4.1 Problem Settings

In this paper, we compare the proposed method to tile-coding to examine the mountain-car task [1]. As depicted in Fig. 3, the agent is surrounded by hills on both the left and right sides. The agent's task is to reach the hill on the right side. However, the agent does not potentially have enough power to reach the right hill, e.g., the agent cannot reach the right hill using only the action "right" from the bottom of the valley with the initial velocity = 0. In the mountain-car task problems written in the textbook by Sutton [1], the agent can perform three types of action: "go left", "do nothing", and "go right". In this paper, for simplicity, the agent can perform only two types of action (cf. Fig. 6): "go left" and "go right". The agent can sense its current position and velocity, as shown on the abscissa in Fig. 3. The agent is rewarded -1 at each time step. The state value of the goal state, i.e., the right end in Fig. 3 is set to 0. One episode finishes either

Table 1: Parameters for the proposed method

ϵ_b	0.05
ϵ_n	0.01
λ	100
a_{max}	20
η	0.5
ϵ_a	0.7
β	0.7
ε_c	6.0

when the agent achieves the goal state or the number of steps reaches 500. For each run, 200 episodes are examined.

The parameter for the proposed method is summarized in Table 1. We adopt a Q-Learning as the reinforcement learning algorithm for the proposed method. We compared the proposed method to the conventional Q-Learning method with tile coding, in which the tile size was set to be 3, 5, 7, 9, 11, 13, or 15. The learning parameters α and γ for the proposed method and the conventional Q-Learning are set to be 0.9.

4.2 Experimental Results

Figure 4 and Fig. 5 show the temporal changes in the moving average of the success ratio and the number of steps required in order to achieve to the goal state, respectively. In both graphs, the x axis denotes the number of episodes. In Fig. 4, the y axis denotes the success ratio, which is defined as the fraction of runs required in order to achieve the goal state. In Fig. 5, the y axis denotes the number of steps. If the agent cannot attain the goal state during a certain run, the number of steps in the run is set to 500, which is the same as the maximum episode length. The conventional Q-learning with coarse-grained tile-coding, e.g., tile size = 3 or 5, shows worse performance than the proposed method and the conventional Q-Learning with fine-grained tile-coding. In the proposed method, the number of steps decreases faster than for other algorithms.

Next, we investigate acquired state segmentations in a typical run as delineated in Fig. 6. In this figure, a neural network learned by the GNG module (upper graph), anticipation vectors for each references (middle graphs), and assigned classes (lower graphs) are plotted. The left and right sides in the middle and lower graphs indicates the results for the action "left" and "right", respectively. The horizontal and vertical axes for all graphs represent the position and the velocity of agents, respectively. The location depicted on these axes is indicated as 0 on the other axes. In addition, we define the zero point of the position of the agent as the bottom point of the valley in Fig. 3. The dots in the upper graph indicate nodes in the GNG module. The references in the CMM module in the middle and lower graphs are located at the same coordinates as corresponding nodes in the GNG module. The numbers in the lower graphs indicate the assigned states. Circled numbers indicate that the state-action value of that reference is greater than the others. The distribution of nodes in acquired neural networks by the GNG module is not uniform. The GNG module does not yield nodes for unseen inputs so that the shape of the resultant node distribution depends greatly on the characteristics of an agent in a given environment. For example, the agent in this problem cannot maintain high speed at the top of mountains. Hence, the proposed method generates no state for such inputs (i.e. the corners of the graphs). Referring the lower graphs, the number of states for the actions "left" and "right" in this run are 11 and 17. The distribution of circled numbers indicates that the agent learns a proper policy.

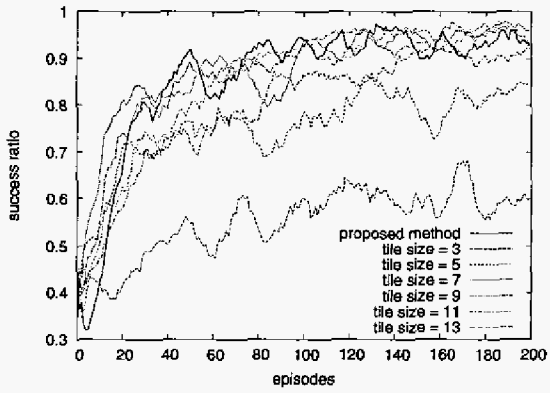

Figure 4: Temporal changes in the moving average of the success ratio

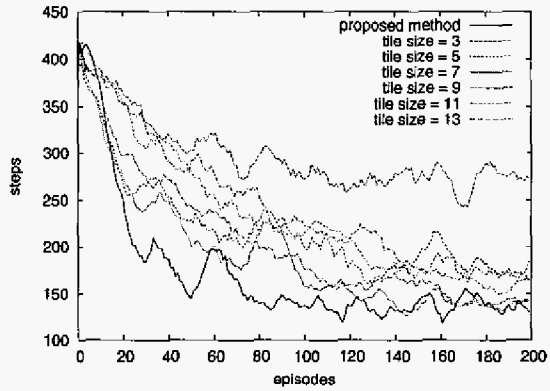

Figure 5: Temporal changes in the moving average of the number of steps required to achieve the goal state

5. Conclusions

In this paper, we proposed an double layered state space construction method for reinforcement learning agents, which consists of Fritzke's Growing Neural Gas algorithms and the Class Management Mechanism. The proposed method con-

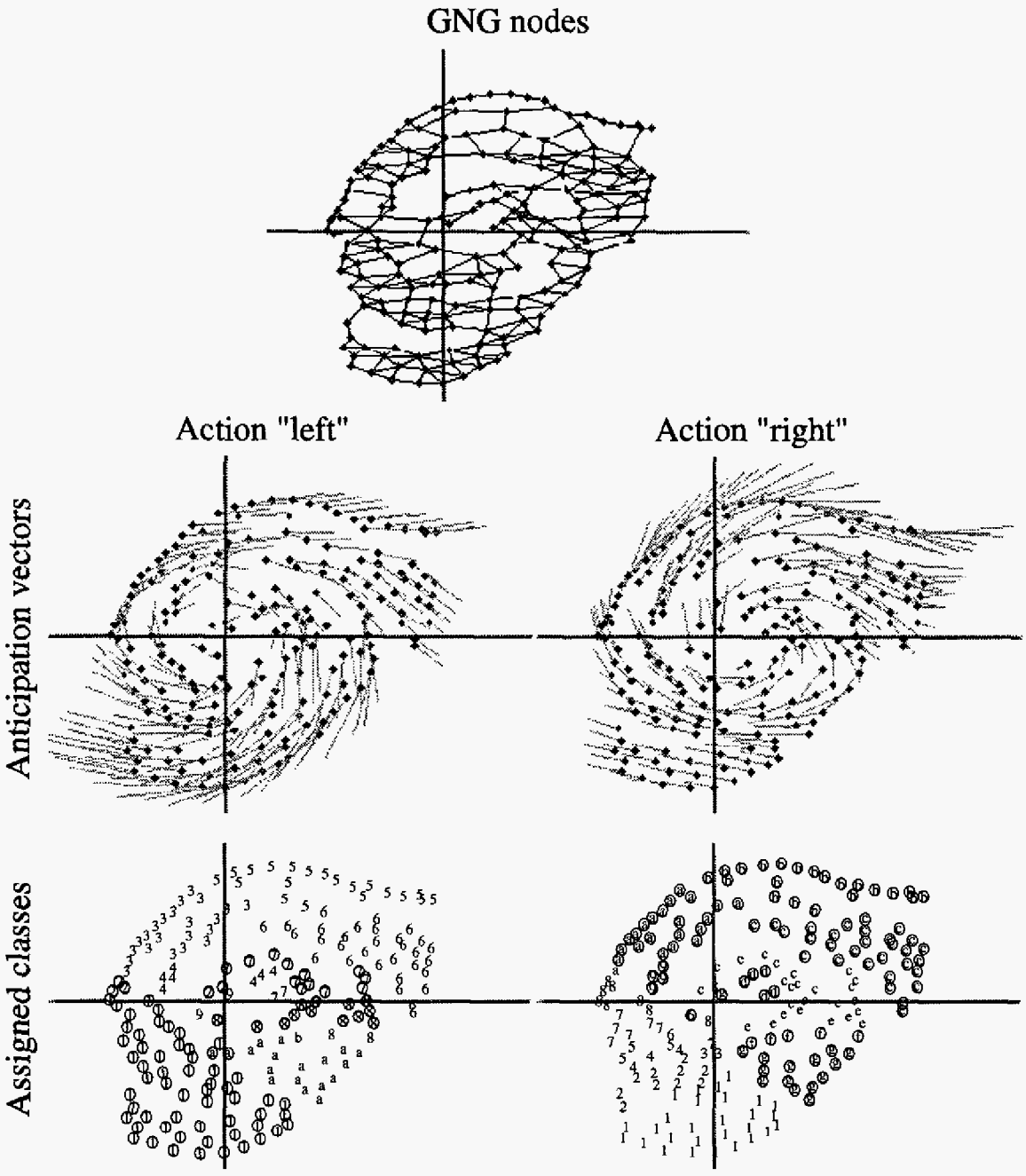

Figure 6: Depictions of state segmentation acquired by the proposed method: a neural network learned by the GNG module (upper graph), anticipation vectors for each references (middle graphs), and assigned classes (lower graphs). The left and right sides of the middle and lower graphs indicate the results for actions "left" and "right", respectively. Horizontal and vertical axes for all graphs represent the position and the velocity of agents, respectively.

stitutes a state space based on anticipated behaviors of an agent, namely, a couple of actions by the agent and the resultant change in sensory inputs. Computational simulations using the mountain-car task clarified the effectiveness of the proposed method. In future studies, we would like to apply the proposed method to other kinds of reinforcement learning problems, including real robots, and to extend the proposed method, by for example incorporating rewards or control variables into the anticipation vectors.

Acknowledgment

This research was supported in part by the Ministry of Education, Science, Sports and Culture through a Grant-in-Aid for Young Scientists (B), 15700159, 2003.

References

[1] R. S. Sutton and A. G. Barto, "Reinforcement Learning; An Introduction," The MIT Press, 1998.

[2] A.Dubrawski and P.Reignier, "Learning to Categorize Perceptual Space of a Mobile Robot Using Fuzzy-ART Neural Network", *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems IROS'94*, Vol.2, pp.1272-1277, 1994.

[3] M. Asada, S. Noda, and K. Hosoda, "Action-Based State Space Construction for Robot Learning," JRSJ Vol.15, No.6, pp.76-82, 1997 (in Japanese)

[4] A. Ueno, S. Nakasuka and K. Hori, "Simultaneous Learning of Situation Classification and Behavior Rules for Autonomous Agents," Journal of Japanese Society for Artificial Intelligence Vol.15, No.2, pp. 297–308 (2000)

[5] R. Pfeifer and C. Scheier, "Understanding Intelligence, " The MIT Press, 1999.

[6] B. Fritzke, "A Growing Neural Gas Network Learns Topologies," *Advances in Neural Information Processing Systems 7*, MIT Press, pp. 625–632, 1995.

[7] C. J. C. H. Watkins, and P. Dayan, "Technical note: Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.

[8] R.Sutton, "Learning to Predict by the Method of Temporal Differences", *Machine Learning*, vol. 3, pp. 9–44, 1988.