10

A FAST-Based Q-Learning Algorithm

Kao-Shing Hwang[2], Yuan-Pao Hsu[1] and Hsin-Yi Lin[2]
[1]*Department of Computer Science and Information Engineering National Formosa University Yunlin,*
[2]*Department of Electrical Engineering National Chung-Cheng University Chiayi, Taiwan*

1. Introduction

Q-learning is a stochastic dynamic programming algorithm that doesn't need to have the interactive model of machine-environment [1] [2]. Taking action '*a*' according to state '*s*', the algorithm measures the feedback rewards and updates its Q values from rewards step by step. In a reasonable period of learning process, the Q-learning obtains outstanding performance and has been successfully applied on collision-avoidance, homing, and robots cooperation [3] [4].

Traditionally, in Q-learning algorithm, a decoder pre-partitions the input space into grid called boxes that map input vectors into activated boxes. However, the way of pre-partitioning the input space might not be suitable for all systems, especially for unknown systems. Therefore, theorems such as CMAC, ART, FAST, etc. have been suggested to replace the box decoder for clustering input vectors [5]-[7]. Input vectors then can be mapped into more suitable clusters for the purpose of getting better state-action values to meet the desirable control effect.

The ART [6] is one kind of unsupervised learning artificial neural networks possessing the advantage of solving the stability-plasticity dilemma. The FAST [7] [8] algorithm merges the advantage of variable vigilance value of the ART and the pruning mechanism of the GAR (Grow and Represent Model) [9] theorem, amending the disadvantage of the boxes method which maps only one activated box for each input vector. It dynamically adjusts the size and location of the sensitivity region for activated neurons, and makes boundaries between categories to be changeable, resulting in producing more suitable input for increasing the system learning speed. Furthermore, when an input pattern activates more than one neuron, the pruning mechanism of the FAST appropriately prunes one of the neurons with overlapped sensitivity region for preserving neurons to accommodate more categories in the cause of resource reservation.

The article aims at combining a FAST-based algorithm and Q-learning algorithm into a reinforcement learning algorithm called ARM Q-learning algorithm that improves Q-learning algorithm on learning speed and learning stability. The article is organized in four sections; the first Section being this introduction. In Section 2 we discuss the theories of clustering and reinforcement learning. Section 3 presents the simulation results and results analysis. Finally, a discussion is drawn in Section 4.

Source: Theory and Novel Applications of Machine Learning, Book edited by: Meng Joo Er and Yi Zhou,
ISBN 978-3-902613-55-4, pp. 376, February 2009, I-Tech, Vienna, Austria

2. Theory

2.1 Adaptive resonance theory

ART is a dynamic neural network structure. With a feedback mechanism the ART produces a top-down expectation. Not only must a neuron win the competition but also should it match this expectation before it is qualified to learn. As the interaction between the feed-forward path and the feedback path is established, the winning neuron will output the same pattern iteratively. This state is so called the resonant state and that is why the theorem is named to be Adaptive Resonance Theory.

2.2 Flexible adaptable-size topology

FAST is developed from the concept of ART algorithm with the dynamic categorization function. Merging with the dynamic vigilance parameter and online pruning mechanism, the FAST is able to realize the clustering efficiently.

Each FAST neuron j maintains an n-dimensional reference vector C_j, and a threshold, T_j, determining its sensitivity region. At the beginning, input patterns are presented and the network adapts through application of the FAST algorithm (Figure 1). If a new input vector P_j doesn't lie within any sensitivity region of any neuron, a new neuron is generated by setting its reference vector C_j centered on the input vector pattern P_j. On the other hand, the C_j and T_j of a neuron are dynamically updated if P_j lies within its sensitivity region. There is another scenario that P_j may activate more than one neuron. This implies the FAST network has redundant neurons with overlapped sensitivity regions. The pruning function is then executed to delete one of these neurons to decrease the network size. The operations can be described as the following steps:

1. Initialize the system.
2. Present an input pattern P and compute $D(P, C_j)$ for every operational neuron j. Where $D(P, C_j)$ is the Manhattan distance, a distance between two points measured along axes at right angles, e.g., on a plane with two points, p_1 at (x_1, y_1) and p_2 at (x_2, y_2), their Manhattan distance is $|x_1 - x_2| + |y_1 - y_2|$.
3. If $D(P, C_j) > T_j$ for all j, activate a new neuron j^* by initializing its reference vector to $C_j^* = P$, its threshold to $T_j^* = T_{ini}$, and its deactivation parameter to $Pr_j^* = 1$.
4. If $D(P, C_j) < T_j$, where j is an active neuron, and T_j is its threshold, update C_j and T_j as follows:

$$C_{ji}(t+1) = C_{ji}(t) + \alpha T_j\left(P_i - C_{ji}(t)\right) \tag{1}$$

$$T_j(t+1) = T_j(t) - \gamma\left(T_j(t) - T_{\min}\right) \tag{2}$$

   where α, γ, and T_{min} are learning constants. Update the global T_{ini} parameter to $T_{ini} = T_{ini}(t+1) - \gamma(T_{ini}(t) - T_{min})$.
5. If several neurons are activated in step 3, deactivate one of the neurons if $rnd(\,) > Pr_j$, where Pr_j is the deactivation parameter, and $rnd(\,)$ is a uniformly distributed random number in the range (0, 1). Increase the activated neurons' probability of deactivation as follows:

$$Pr_j(t+1) = Pr_j(t) - \eta(Pr_j(t) - Pr_{min}), \tag{3}$$

Where η and Pr_{min} are pruning constants.

6. Goto step 2.

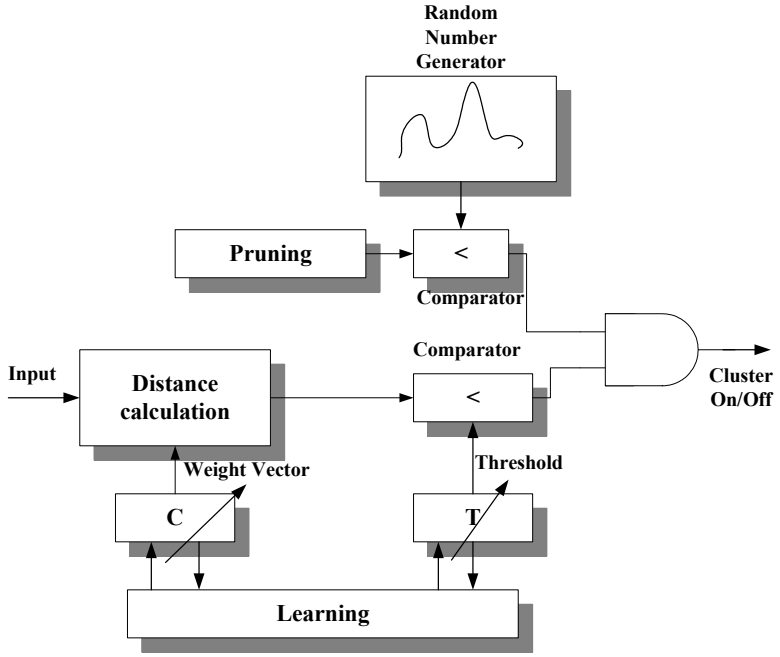


Fig. 1. Block diagram of an FAST neuron.

2.3 Q-learning algorithm

Q-learning is one of reinforcement learning algorithms (figure 2). The algorithm selects an action with max Q value (or by ε-greedy) from the Q table basing on the clustered result of current state. A feedback reinforcement signal (reward) is then read for the system to update its Q value and the system enters the next state at the same time. At the next state, the system goes again the selection of the action, and receives the reward, and enters the following state. The operations are periodically performed for reaching the ultimate purpose that the best action of each state can be trained to get the optimal Q value (Q*).

The procedure of the algorithm is list as follows.

- Initialize the system. All Q(*s*, *a*) are set to be zero.
- Repeat (for each episode):
  - Repeat (for each step of episode):
    - Observe current state *s*.
    - Select an action *a* with max Q value from Q table (or select an action by ε-greedy policy).
    - Take action *a*, read reward *r* caused by action *a*, and observe next state *s′*.
    - Update Q value corresponding to state *s* and action *a* as equation (4).

$$Q(s,a) = Q(s,a) + \beta(r + \gamma Max(Q(s',a)) - Q(s,a)) \quad (4)$$

Where *r* is reinforcement signal (reward), *s′* is all possible states t hat state *s* can visit, and β and γ are coefficients lie between 0 and 1.

- Replace *s* with *s'*.
- Until *s* is terminal or stop condition has been met.

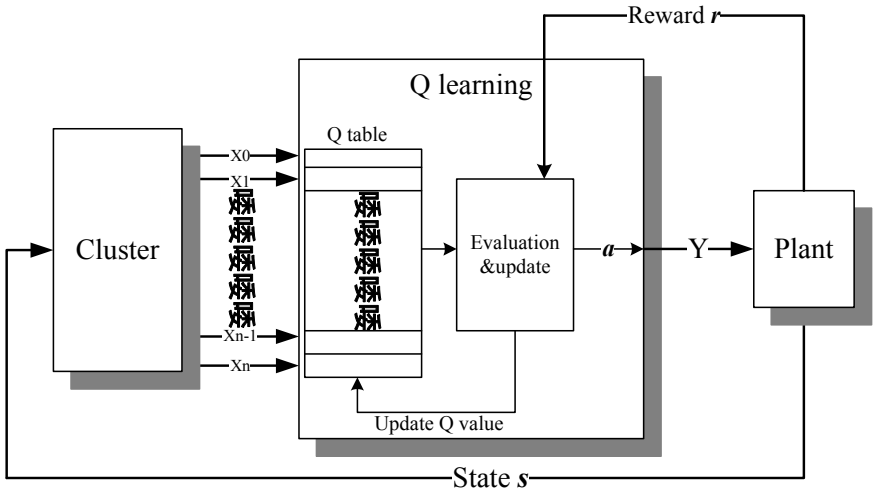


Fig. 2. Q-learning block diagram.

2.4 Adaptive resonance method

The ARM architecture modified from FAST is developed to provide more suitable clustered results to the Q-learning architecture for achieving the purpose of improving the learning speed and learning stability. The modification involves three parts: normalization, weight update, and pruning mechanism.

2.4.1 Normalization

The FAST compares an input vector with each neuron's center point to calculate the Manhattan distances *D*. However, the variation of the input vector in its distinct dimension may not have the same scale. Consequently, in the system input space, a slightly changing of a parameter in some dimension of an input vector might make the calculated *D* larger than *T* value causing a new neuron being generated. For another parameter, on the other hand, even having a great deal of change, the *D* is still unable to exceed the *T* value to generate a new neuron. Therefore, a normalization mechanism should be added in front of the FAST to normalize each dimension's value of the input vector into the same scale, so as to appropriately categorize the input vectors and to generate new neurons.

2.4.2 Weight update

When a neuron has been activated repeatedly, its center point moves following the successive new input vectors. The more number of times of a neuron has been activated, the larger moving distance of the neuron will have. This causes the FAST encoding the same input vector into different cluster and the Q-learning will receive erroneous clusters.

As shown in figure 3, (a) represents that the system generates a neuron centered by C_0 for the input vector P_1; (b) shows that the center point C_0 has been moved to C_0' for the input

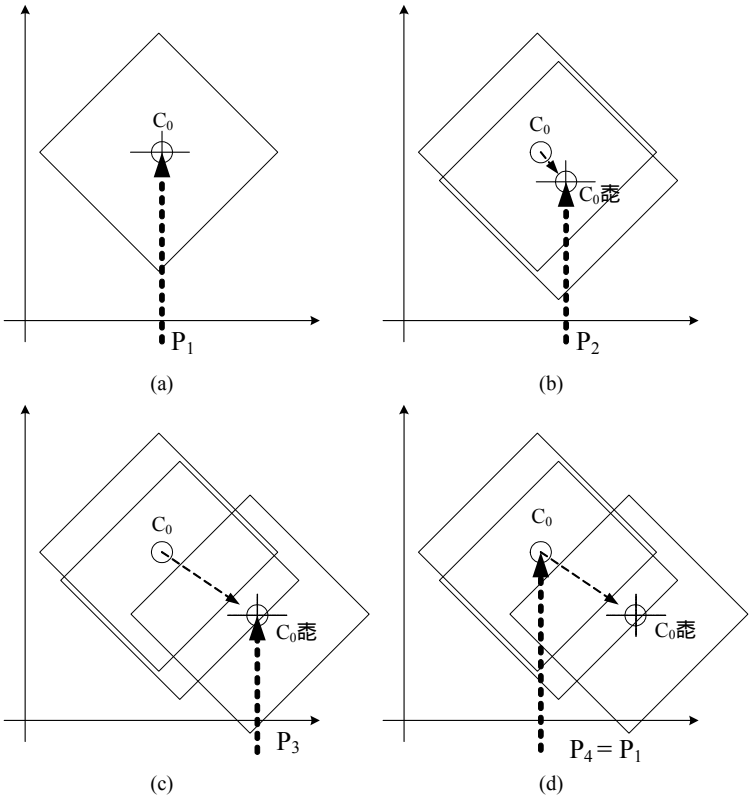


Fig. 3. Center adjustment of a FAST neuron.

vector P_2 which lies within the sensitive region of C_0, but P_2 is different from P_1; (c) represents that the center point C_0 has been moved farther away from its original center in (a) because of the input P_3; (d) points out that the input vector P_4 lies outside of the sensitive region of C_0 and will be categorized as another cluster which is different from the cluster of the P_1, even P_4 is the same as P_1. The system would not be able to refer to the learned experiences for this reason and jeopardize its performance significantly. To resolve this difficulty, the learning rate α should be limited under some small value in order that the FAST can cluster input vectors stably.

2.4.3 Pruning mechanism

The original concept of pruning mechanism of the FAST is depended upon the pruning probability, whereas the pruning probability is determined by the number of times of a neuron being activated. From this concept, a neuron is prone to be pruned when its times of being activated is relatively larger than other neurons. However, it is also possible that similar input vectors are easily to repeatedly present on the input making the FAST regenerates new neurons for the input vectors that they might have just caused some neurons to be pruned. The system will be stuck on the unstable situation of switching between pruning and generating. Moreover, neurons that were generated by input vectors

appeared only once will stay in the system and unreasonably never have the chance of being pruned. This leads us to modify the pruning mechanism that keeps neurons that are having high probability of being activated, deletes neurons which have low probability of being re-activated.

The pruning mechanism is modified to be: when a neuron is activated, its *Pr* value is set to be 1, while the *Pr* values of other un-activated neurons are updated basing on (3). The neuron number is limited to be *K*. When the number of generated neurons of the system reaches *K*, the pruning mechanism is triggered to check the *Pr* values of each neuron. Those neurons with *Pr* values smaller than a random number *rnd()* will be pruned.

The algorithm of the modified FAST is listed as the following steps.

1. Normalize input vector *P*.
2. Calculate Manhattan distance $D(P,C_j)$ between input vector *P* and neurons *j*.
3. If $D(P,C_j)>T_j$, input vector *P* doesn't belong to any existing clusters, generate the (*j* + *1*)th new neuron and initialize its weight value being $C_{j+1} = P$, and $T_{j+1} = T_{ini}$, and $Pr_{j+1} = 1$.
4. If $D(P,C_j)<T_j$托categorize input vector *P* being the cluster of the *jth* neuron, set $Pr_j = 1$, update *Cj* and T_j according to (1) and (2). Update *Pr* value of each un-activated along (3).
5. If generated neurons reaching *K*, compare each neuron's Pr_j with a random number *rnd()* (between 0 and 1), delete one of the neurons when their *Pr* are larger than *rnd()*.
6. Get next input vector *P*, go to step 1婇

These modifications of the FAST give birth to the ARM. The ARM replaces the cluster module in figure 2 to encode input vectors for the Q-learning. Section 3 will demonstrate its performance.

2.4.4 Modified Q-learning

The original Box Q-learning selects the action with the biggest Q value from the Q table referred to by the box that triggered by input state. But, in our architecture, we calculate the Manhattan distances between the center point of current state and triggered neurons. These distances determine the percentage of each activated neurons that contribute to the calculation of Q value. If a triggered neuron is near the current state, then their Manhattan distance would be small, so that neuron provides a large share of the Q value. The proportion scale that each activated neuron contributes to the Q value is shown in equation (5).

$$w_i = \begin{cases} \dfrac{\sum distance - distance_i}{(n-1)\times\sum distance}, when\ \ n \neq 1\ ; \\ 1.0 \qquad\qquad\qquad\qquad\quad , when\ n\ = 1\ . \end{cases} \tag{5}$$

Where w_j represents the proportion that activated neuron *j* contributes; *n* denotes the number of activated neurons; $\sum distance$ stands to be the summation distance of all activated neurons; $distance_j$ denotes the total distance of neuron *j*.

Expected value *V* is calculated as follows:

$$V_t = \max_a \sum w_j Q_j(s,a)\,, \tag{6}$$

where V_t is expected value at time *t*.
Q is updated according to equation (7).

$$Q_j(s_t, a) = Q_j(s_t, a) + \beta(r_t + \gamma V_t \times w_j - Q_j(s_t, a)) \tag{7}$$

3. Simulation

3.1 Cart-Pole simulation

The well-known cart-pole simulation having been used in verification of the performance of reinforcement learning algorithms is simulated here to check if the proposed architecture works. A typical mathematical model of a cart-pole system can be represented by forms as (8) and (9). We use fourth order Runge-Kutta to simulate the cart-pole model by a personal computer [10].

$$\ddot{\theta}_t = \frac{g\sin\theta_t + \cos\theta_t\left[\frac{-F_t - ml\dot{\theta}_t^2\sin\theta_t + \mu_c\,\mathrm{sgn}(\dot{x})}{m_c + m}\right] - \frac{\mu_p\dot{\theta}_t}{ml}}{l\left[\frac{4}{3} - \frac{m\cos^2\theta_t}{m_c + m}\right]}, \tag{8}$$

$$\ddot{x}_t = \frac{F_t + ml[\dot{\theta}_t^2\sin\theta_t - \ddot{\theta}_t\cos\theta_t] - \mu_c\,\mathrm{sgn}(\dot{x}_t)}{m_c + m}. \tag{9}$$

Where g is the gravity constant, 9.8 m/s²; m_c is the mass of cart, 1 kg; m_p is the mass of pole, 0.1kg; l is the half length of pole, 0.5 m; μ_p is the friction coefficient of the pole on the cart; μ_c is the friction coefficient of the cart on the track; F_t is the applied force generated by the controller. The dynamics of the systems are assumed to be unknown but the states are measurable. In the simulation, frictions are eliminated. Hence μ_p and μ_c are set to be zero.
The cart travels left or right along a one dimensional track. The pole is hinged to the top of the cart and is free to move in the vertical plane along with the track. The objective is to learn to drive the cart left or right so as to keep the pole balanced vertically above the cart, and to keep the cart from colliding with the ends of track. The learner accesses the state vector $(x, \dot{x}, \theta, \dot{\theta})$ at each time step and selects one of two actions, a rightward or leftward force on the cart. The cart-pole system begins with $x = 0$, $\dot{x} = 0$, $\theta = 0$, and $\dot{\theta} = 0$. If the pole falls over more than 12 degrees from vertical, or if the cart hits the track boundary (track length 2.4m), a failure occurs. All immediate rewards are zero except that when a failure is occurred a reward of negative one is received. When a failure occurs, the cart-pole system is reset to the initial state, and begins a new attempt to balance the pole.

3.2 Simulation results

The simulation includes 10 experiments. There are 500 trials in one experiment. One trail is said to be ended if an attempt of balancing the pole is failed. When an attempt is lasting over 100,000 sampling steps, a successful trial occurs, and a new attempt is restarted by initializing the cart and pole. Sampling interval is set as 0.02 second.

Parameters of the ARM are: learning constant α=0. 000001, γ扩0.007, η扩0.045, $T_{ini} = 2.0$, $T_{\min} = 1.8$, $\Pr_{ini} = 1.0$, and $\Pr_{\min} = 0.09$; whereas parameters of the Q-learning are set to be: learning rate α扩0.2, discount rate γ扩0.999.

	Best	Worse
BOX Q	351	>500
ARM Q	7	82

Table 1. Summary of simulation results

ARM Q-learning and Box Q-learning are simulated for manifesting the improvement of the proposed method. Average results of 10 experiments of the both algorithms are depicted in figure 4. The ARM Q-learning method balances the pole within about 100 trials, while the Box Q-learning method needs more than 500 trials to learn to balance the pole. The stability of the ARM Q-learning also shows in the figure. The pole can be balance again quickly even if it should fall after a long time of balancing. The ARM Q-learning keeps its learned experiences well to equip it with superior stability.

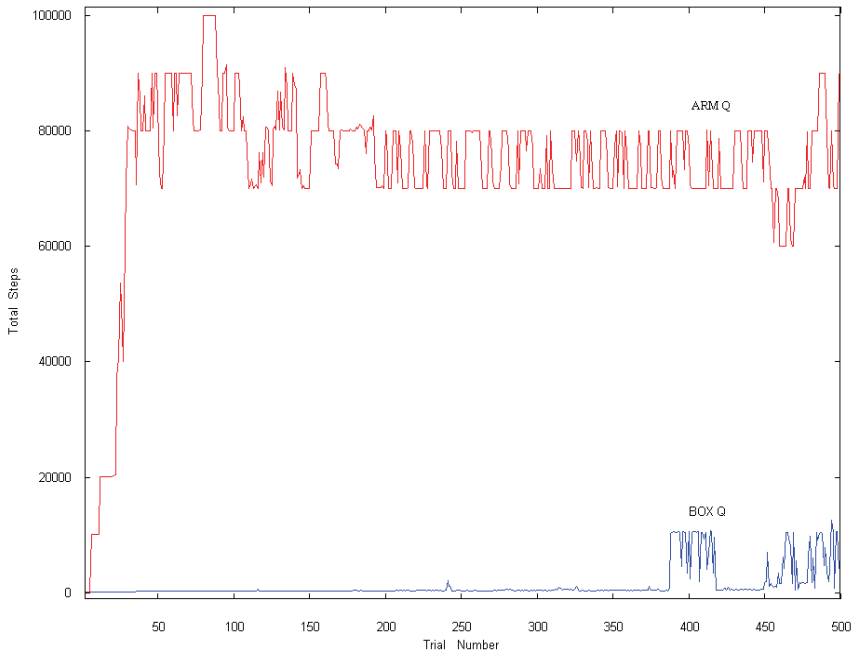


Fig. 4. Performance of ARM Q-learning and Box Q-learning system.

Table 1 demonstrates that the modified ARM Q took only 7 trails for the first time that it can successfully balance the pole for the best case. To accomplish the same job, the Box Q took 351 trials. Furthermore, the ARM generated 7 to 10 neurons to categorize the states. The Box instead pre-partitioned the input state space into 162 boxes.

In accordance with the above observation, for a plant like the cart-pole system, the ARM Q-learning method needed only a small amount of neurons to learn to balance the pole within much shorter time than the Box Q-learning method did.

4. Discussion

This article proposed a reinforcement learning architecture that combines ARM, a FAST–based algorithm, and Q-learning algorithm. The ARM, at the front end, is featured with multi-neuron triggering and dynamically adjusting its sensitive region as well, providing the Q-learning, at the back end, with more suitable clustered input states. In such a way, the Q-learning learns quick and stable.

There are future work can be tried to enhance the Q-learning for constructing more efficient reinforcement learning architecture, such as replaces the Q-learning in our architecture with the Q(λ)-learning [11] or SARSA [12].

5. Acknowledgment

The authors are grateful for financial support from the National Science Council of Taiwan, the Republic of China, under Grant NSC94-2213-E-150-025-.

6. Reference

[1] Watkins, C. J. C. H., and Dayan, P., Technical note: Q-Learning, *Machine Learning*, 8(3-4): pp. 279-292, 1992.

[2] Claude F. Touzet, "Q-Learning for Robot," in M. A. Arbib, editor, *Handbook of Brain Theory and Neural Networks*, pp. 934-937, 2003.

[3] L. E. Parker, C. Touzet and F. Fernandez, "Techniques for learning in multi-robot teams," in *Robot Teams: From Diversity to polymorphism* (T Balch and L. E. Parker, Eds.), Natick, MA: A. K. Peters, 2001.

[4] K.-S. Hwang, S.-W. Tan; C.-C. Chen, "Cooperative strategy based on adaptive Q-learning for robot soccer systems," *IEEE Transactions on Fuzzy Systems*, Vol. 12, Issue: 4, pp. 569-576 Aug. 2004.

[5] G. A. CARPENTER and S. GROSSBERG, "The ART of adaptive pattern recognition by a self-organizing neural network," *IEEE Computer*, 21(3), pp77-88, March 1988.

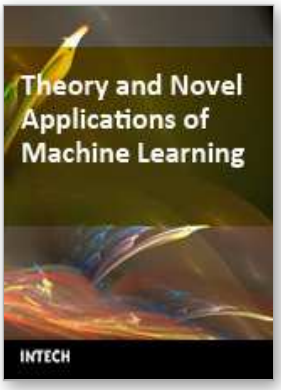
[6] J. S. Albus, "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)," Trans. ASME, J. Dynamic Syst. Meas., Contr., Vol. 97, pp. 220-227, Sept. 1975.

[7] A. Pérez-Uribe, *Structure-adaptable digital neural networks*, PhD Thesis 2052, Swiss Federal Institute of Technology-Lausanne, Lausanne, 1999.

[8] A. Perez and E. Sanchez, "The FAST architecture: a neural network with flexible adaptable-size topology," *Proceedings of Fifth International Conference on Microelectronics for Neural Networks*, pp. 337–340, 12-14 Feb. 1996

[9] A. E. Alpaydin, *Neural models of incremental supervised and unsupervised learning*. PhD thesis, Swiss Federal Institute of Technology-Lausanne, Lausanne, DPFL, 1990. Thesis 863.

[10] Barto, Andrew G., Sutton, Richard S. and Anderson, Charles W., "Neuronlike adaptive elements that can solve difficult learning control problems." *IEEE Transactions on System, Man, and Cybernetics* SMC-13: 834-846. 1983.
[11] Peng, J. and Wiliams, R.J., "Incremental Multi-Step Q Learning," *Machine Learning, 22*, pp. 283-290, 1996.
[12] R. S. Sutton and A. G. Barto, *Reinforcement Learning An Introduction*, Cambridge, Mass., MIT Press, 1998.



Theory and Novel Applications of Machine Learning

Edited by Meng Joo Er and Yi Zhou

ISBN 978-953-7619-55-4
Hard cover, 376 pages
Publisher InTech
Published online 01, January, 2009
Published in print edition January, 2009

Even since computers were invented, many researchers have been trying to understand how human beings learn and many interesting paradigms and approaches towards emulating human learning abilities have been proposed. The ability of learning is one of the central features of human intelligence, which makes it an important ingredient in both traditional Artificial Intelligence (AI) and emerging Cognitive Science. Machine Learning (ML) draws upon ideas from a diverse set of disciplines, including AI, Probability and Statistics, Computational Complexity, Information Theory, Psychology and Neurobiology, Control Theory and Philosophy. ML involves broad topics including Fuzzy Logic, Neural Networks (NNs), Evolutionary Algorithms (EAs), Probability and Statistics, Decision Trees, etc. Real-world applications of ML are widespread such as Pattern Recognition, Data Mining, Gaming, Bio-science, Telecommunications, Control and Robotics applications. This books reports the latest developments and futuristic trends in ML.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Kao-Shing Hwang, Yuan-Pao Hsu and Hsin-Yi Lin (2009). A FAST-Based Q-Learning Algorithm, Theory and Novel Applications of Machine Learning, Meng Joo Er and Yi Zhou (Ed.), ISBN: 978-953-7619-55-4, InTech, Available from: http://www.intechopen.com/books/theory_and_novel_applications_of_machine_learning/a_fast-based_q-learning_algorithm

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821