

# *PQ*-Learning: An Efficient Robot Learning Method for Intelligent Behavior Acquisition

Weiyu Zhu, Stephen Levinson

*Beckman Institute, 405 N. Mathews, Urbana, IL61801, USA*

*Dept. of Electrical and Computer Engineering, Univ. of Illinois at Urbana-Champaign  
{weiyuzhu, sel}@ifp.uiuc.edu*

**Abstract** This paper presents an efficient reinforcement learning method, called the *PQ*-learning, for intelligent behavior acquisition by an autonomous robot. This method uses a special action value propagation technique, named the spatial propagation and temporal propagation, to achieve fast learning convergence in large state spaces. Compared with the approaches in literature, the proposed method offers three benefits for robot learning. First, this is a general method, which should be applicable to most reinforcement learning tasks. Second, the learning is guaranteed to converge to the optimum with a much faster converging speed than the traditional  $Q$  and  $Q(\lambda)$ -learning methods. Third, it supports both self and teacher-directed learning, where the help from the teacher is directing the robot to explore, instead of explicitly offering labels or ground truths as in the supervised-learning regime. The proposed method had been tested with a simulated robot navigation-learning problem. The results show that this method significantly outperforms the  $Q(\lambda)$ -learning algorithm in terms of the learning speeds in both self and teacher-directed learning regimes.

## 1. Introduction

Intelligent behavior acquisition by a robot is a challenging topic in the field of Artificial Intelligence and Robotics. Traditional methods are usually based on planning or supervised learning, in which either complete task-specific knowledge from human beings is incorporated or extensive directions or feedbacks from an outside teacher are highly relied on. Although these methods have achieved impressive robotic intelligence in some fields, more often than not, the intelligent systems built in this way usually only work well on specific tasks and can hardly be transplanted from one to another. In addition, since these methods usually highly rely on humans' knowledge of the world, which might be incorrect sometimes, the acquired robotic systems may not be robust and adaptive in the real world.

In recent years, the Reinforcement Learning (RL) methodology has been paid increased attention for autonomous agent learning with little or no *a priori* knowledge of the environment [1][2][3]. An essential challenge of the current RL methods, such as the Watkins'  $Q$ -Learning algorithm [4], lies in the size of the learning state space. According to Watkins and Dayan [5],  $Q$ -Learning is guaranteed to converge to the optimum as long as all state-action pairs are continuously experienced and updated in learning. This requirement places great difficulties on many real learning tasks for two reasons. First, the size of the learning state spaces, which are usually obtained from the partitioned continuous feature spaces in the real world, could be very large in general. Secondly, learning with a real robot is practi-

cally time-consuming. It is usually unrealistic to let the robot exhaustively explore all situations in order for the learning to converge.

An intuitive scheme of coping with large state spaces is value function generalization. That is, the learning uses a function approximator, instead of a plain table, to represent value functions so that the robot is not necessary to explicitly visit all states in order to obtain a learned value function for each action. Some researchers have experimented with this scheme and some positive examples have been reported [6][7]. Nevertheless, in general this approach has been proved dangerous [8][9] because the estimate-based value updating cannot guarantee to reduce the errors of value functions at each step; consequently, the learning might not converge at all.

Adaptive resolution is another well-studied strategy for continuous feature space learning. In this scheme, feature spaces are partitioned adaptively according to a certain criterion so that different feature regions may receive different attention in learning. In [10], Moore proposed a PartiGame algorithm for solving puzzle problems, in which the agent learns to travel to a goal location in a cell graph by adaptively partitioning the graph to find the shortest feasible path. In [12], a decision tree based adaptive state partitioning method was proposed by Chapman and Kaelbling for solving learning problems in high dimensional binary spaces. In the real robot-learning domain, Yasutake *et al* [13] used local action models to partition feature spaces for robot soccer skill learning and Zhu *et al* have presented a hybrid state-partitioning strategy for autonomous robot navigation learning in [11]. Ideally, adaptive resolution approaches are capable of handling any state-partitioning learning problems. However, designing a clever adaptive strategy in general is not easy since it requires heuristic knowledge of each specific task. For instance, the Moore's PartiGame algorithm relies on the complete and known state transition models of the environment while Yasutake's local action model method assumes that the optimal actions of the physical states sharing the same action models must be the same, which might not hold in most other learning tasks.

In this paper, we proposed an efficient RL learning method called the *Propagated Q (PQ)*-Learning. This method is based on Watkins' *Q*-learning algorithm while action value functions are learned partially based on a recursive *spatial propagation* and *temporal propagation* process at each learning step, instead of the traditional one or multiple sequential step tracing scheme used in the *Q* or *Q*( $\lambda$ ) learning. Compared with the current RL methods in literatures, this method possesses several advantages. a) It is a general model-free RL method. b) The learning is guaranteed to converge to the optimum with much less required number of learning episodes than the *Q* and *Q*( $\lambda$ ) learning. c) It supports both self and teacher-directed learning, in which the correctness of the teacher, however, is not required. We have used the this method on a simulated navigation-learning problem (with 10,000 states defined), where it was proved to outperform the *Q*( $\lambda$ ) learning in terms of the converging speeds under both self and teacher-directed learning scenarios.

The rest of this paper is organized as follows. Section 2 presents the details of the proposed *PQ*-learning algorithm. Section 3 is focused on the experiments carried out with the *PQ* and the *Q*( $\lambda$ ) learning algorithms, followed by the conclusion in section 4.

## 2. Proposed Method

### 2.1 *Q*-learning

Before introducing the proposed *PQ*-learning method, we'd first give a brief overview to Watkins' *Q*-learning algorithm, on which this method is based. *Q* learning is a Temporal-

Difference (TD) based off-policy RL algorithm developed by Watkins in 1989 [4]. Its simplest form, the one-step-learning, is defined as

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_{t+1} + \gamma \max_{a'} Q(s', a') - Q(s, a)] \quad (1)$$

where  $Q(s_t, a_t)$  is called the action-value function which indicates the goodness of taking action  $a_t$  at state  $s_t$ .  $r_{t+1}$  is the instant reward received after the action is taken and  $\alpha$  is a constant step-size parameter.  $\gamma$  is called the *discount rate*, which reflects the influence of future rewards on the current state.

The  $Q$ -learning algorithm updates only one action value at each time. Therefore, the learning convergence could be very slow if the number of states is large since the number of required learning steps may increase exponentially as the size of the state space increases. An improvement is called the  $Q(\lambda)$ -learning [4], in which a sequence of action values are traced and updated at each learning step according to the so-called *eligibility traces* [4].  $Q(\lambda)$  algorithm learns much faster by updating multiple value functions at each step. However, since the updating only takes place on the states experienced, the learning still requires the explicit visits to all states in order for all value functions being updated.

## 2.2. Propagated $Q$ -learning

The basic idea of the proposed  $PQ$ -learning method is to update value functions of multiple states simultaneously at the each step. However, in addition to feeding the updates to the predecessors (temporal propagation), like what the  $Q(\lambda)$ -learning does, we also propagate the changes to the neighboring states meanwhile (spatial propagation) so that a state could be learned even if it has not been visited once before.

### General Assumptions

To simplify the analysis, the proposed method was developed based on a number of general assumptions regarding the learning environment and task natures.

- Negative rewards are received only at the end of the failed learning episodes.
- For any non-terminal state (neither a failed nor a goal state), there exists at least one action for the robot to arrive at another non-failed state. That is, at least one path exists for the robot to reach its goal from any non-terminal state.
- State transitions and action rewarding policies are fixed, although the robot does not know them in advance.

Based on these assumptions, the learning scenario of the  $Q$ -learning algorithm can be pictured like this: given all  $Q$  values being zero-initialized, these values would increase monotonically towards the optimums in learning unless a negative reward is received, when the corresponding  $Q$  value becomes negative. This analysis is helpful for the later on proof of the convergence of the proposed  $PQ$ -learning method.

### Spatial Propagation

The motivation of spatial propagation comes from the observation that, given the feature space being “smooth” enough, the action values of two states close to each other are usually very close too. Therefore, it might be reasonable to let the neighboring states “share” the update of the current state so that more than one states could be learned simultaneously. To achieve this idea, two issues should be considered in designing the value-propagation strategy in order to guarantee the correct convergence of the learning. First, the propagation should not affect the learning of “critical” regions, i.e., the regions where optimal value functions change dramatically, instead of smoothly as in most cases. Secondly, the propaga-

tion should not introduce oscillations in value function changes so that the updating would take place monotonically (or near monotonically) toward the optimums.

Figure 1 displays the propagation algorithm used in the proposed method, in which two facts are implied: 1) Propagations only “promote” the  $Q$  values of the neighboring states while the propagated updates never exceed the values of the current states. 2) Negative value functions of the neighboring states never receive promotions in propagations.

For each neighboring state  $s_i$  do:  
 Compute the Euclidean distance  $d_i$  to the current state  $s$   
 Compute the *reducing factor*  $\alpha_i = 1 - \text{MIN}(1, \beta d_i)$ , where  $\beta$  is a positive constant factor  
 Update  $Q(s_i, a) \leftarrow \text{MAX}[Q(s_i, a), \alpha_i Q(s, a)]$  if  $Q(s_i, a)$  is non-negative or  $s_i = s$

Figure 1. Algorithm of spatial propagation of action value  $Q(s, a)$

By choosing a proper distance factor  $\beta$ , the promoted  $Q$  values are guaranteed not to exceed the respective optimums. In addition, since negative value functions are never promoted, the updating introduced by propagations would be made monotonically toward the optimums for most states, except for the actions with negative  $Q$  values, when the promotions might be completely wrong. However, according to the algorithm details, the incorrectly promoted value functions still have chance to be adjusted if that state is visited in the future, when the learning of that value function is degraded to the  $Q$ -learning scheme.

The idea of the spatial propagation process is similar as that of the function approximation method. However, according to the above analysis, our algorithm guarantees the union of the two criteria placed before, i.e., value functions are updated monotonically toward the optimums while critical regions can still be correctly learned.

### Temporal Propagation

Similar to the eligibility trace idea in  $Q(\lambda)$ -learning, the aim of temporal propagation is to exploit the updating information of the current state to reevaluate the previous experience. However, unlike the scheme in  $Q(\lambda)$ -learning, the proposed method propagates the updating information to, instead of the “eligible” states experienced in the current learning episode only, all state transitions in history so that the consistency of value functions among states are better maintained.

Figure 2 illustrates the temporal propagation process in the  $PQ$ -learning method. Once the value of a given state is updated, the new value function is used to “re-process” the learning of all predecessors (the triples of state, action and reward) by recursively “playing-back” the previous experience in mind. The “replay” process, which is similar to the “experience replay” technique used by Thrun [1] and Lin [14], does not affect the learning convergence in general.

If  $(\Delta V(s) > \text{threshold})$   
 For each predecessor triplet  $(s', a', r')$ , do:  
 $Q(s', a') \leftarrow Q(s', a') + \alpha[r' + \gamma V(s) - Q(s', a')]$   
 where  $V(s)$  is the value function of state  $s$  defined as  $V(s) = \text{MAX}_a Q(s, a)$

Figure 2. Algorithm of temporal value propagation for state  $s$

### Summary of the proposed method

By combining the spatial and temporal propagation modules, a recursive value-updating scheme in  $PQ$ -learning is obtained, which is summarized in Figure 3.

```

Initialize  $Q(s, a)$  to zeros, and do for each learning episode:
  Initiate an initial state  $s$ 
  Repeat (for each step of episode):
    Choose action  $a$  for  $s$  using a certain policy (e.g.,  $\epsilon$ -greedy)
    Take action  $a$ , observe the instant reward  $r$  and the next state  $s'$ 
    Initiate a queue with the state  $s$ 
    Repeat
      Take the head element  $s$  from the queue
      Do spatial propagation (SP), including the state  $s$  itself
      Do temporal propagation (TP) for all states whose  $V$  values are updated in SP
      Insert all states whose  $Q$  values are updated in TP process into the queue
    Until the queue is empty
     $s \leftarrow s'$ ;
  until  $s$  is terminal

```

Figure 3. Summary of the  $PQ$ -learning algorithm

According to the analysis of the propagation properties, the  $PQ$ -learning method guarantees: a) The  $Q$  values of “non-failed” actions will increase monotonically toward the optimums and b) the values of “failed” actions could be learned and maintained thereafter if the corresponding state-action pair is explicitly experienced in learning.

### 3. Experiments

#### 3.1. Simulation setup

The proposed learning method has been tested with a simulated robot navigation-learning task. The simulated robot started at a random location in a 5m×5m playing ground and learned to navigate toward a goal somewhere in the environment. The input to the robot consisted of the distance from the robot to the goal and the panning angle of its head, by moving which the robot tried to keep the goal in the middle of its viewing field. The outputs were four actions: going forward for 60 or 30 cm and turning left or right for 5°. Initially, the robot did not have any *a priori* knowledge of the environment except for knowing there were four actions to choose and also being aware of success and failure, when a learning episode terminated. A situation was said to be a success if the robot was within 30cm to the goal and the head angle was less than  $\pm 3^\circ$ , when a positive reward +1000 was received. Similarly, the robot was said to have failed if it could not see the goal anymore (the head angle  $\geq \pm 50^\circ$ ), when a negative reward -1000 was imposed. Learning states were obtained by uniformly partitioning the feature space with a resolution of 5cm and 1°, respectively. That is, the learning space consisted of a total 100×100 states.

The learning performance was evaluated in terms of decision accuracy and the speed of convergence to the ground truth, which was obtained by solving the following iterative Bellman equations [15] according to the known action and state-transition models.

$$\begin{aligned}
 \text{Policy Evaluation : } & V(s) \leftarrow r_{ss'}^{a^*} + \gamma \mathcal{W}(s'), \text{ where } a^* \text{ is the current optimal action at } s \\
 \text{Policy Improvement : } & a^* \leftarrow \arg \max_a \left[ r_{ss'}^a + \gamma \mathcal{W}(s') \right]
 \end{aligned} \tag{2}$$

#### 3.2. Self-learning without directions

The first experiment was to let the robot learn completely by itself without any external directions. The robot autonomously explored the environment and chose actions to perform

according to the  $\epsilon$ -greedy algorithm, with which the agent would always choose the current optimal action to perform with the probability of  $1-\epsilon$  while other actions are randomly selected with the remaining likelihood of  $\epsilon$ . Initially all action values were set as zero, so the robot has to select actions randomly at the beginning.

Figure 4 shows the percentage of states learned with the  $PQ$  and  $Q(\lambda)$  (to be concise,  $Q(\lambda)$  is denoted as  $Q$  in all figures) algorithms in a varying number of learning episodes, where  $\epsilon$  was set as 0.8 and all statistics were collected and averaged in five repeated experiments. A state  $s$  was said to be learned if its state value function  $V(s)$  became positive, which means that the robot had already obtained a certain knowledge about action selection at state  $s$  to achieve its goal. The statistics in Figure 4 shows the superior efficiency of the  $PQ$ -learning method, where most states were learned after 100 episodes while only half states were learned with  $Q(\lambda)$ -learning in 1000 episodes.

Figure 5 depicts the averaged errors, which were defined as  $|Q_{learned} - Q_{truth}| / Q_{truth}$ , of the learned value functions. Combined with Figure 4, one may observe that our method learned not only faster but more accurate than the  $Q(\lambda)$ -learning as well. In 5000 episodes, the value functions had almost (less than 8% in error) converged to the ground truth while the error of  $Q(\lambda)$ -learning was around 40% and there were still some states not learned yet. A visual representation of action values learned for each state in 5120 learning episodes is given in Figure 6 (a), where the values are linearly scaled into gray levels 0~255. The  $PQ$ -learning, as expected, offered a very good approximation to the ground truth.

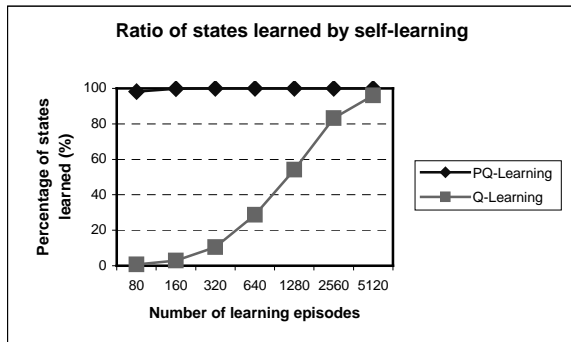


Figure 4. Learning efficiency by self-learning.

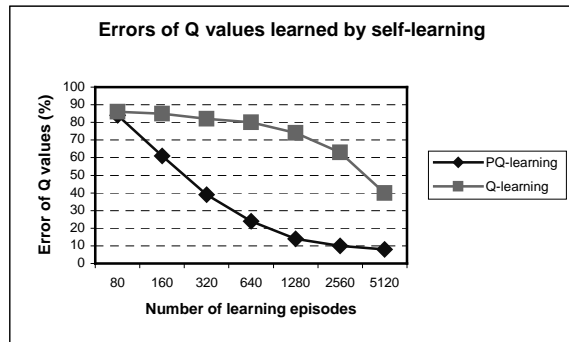


Figure 5. Averaged value errors by self-learning.

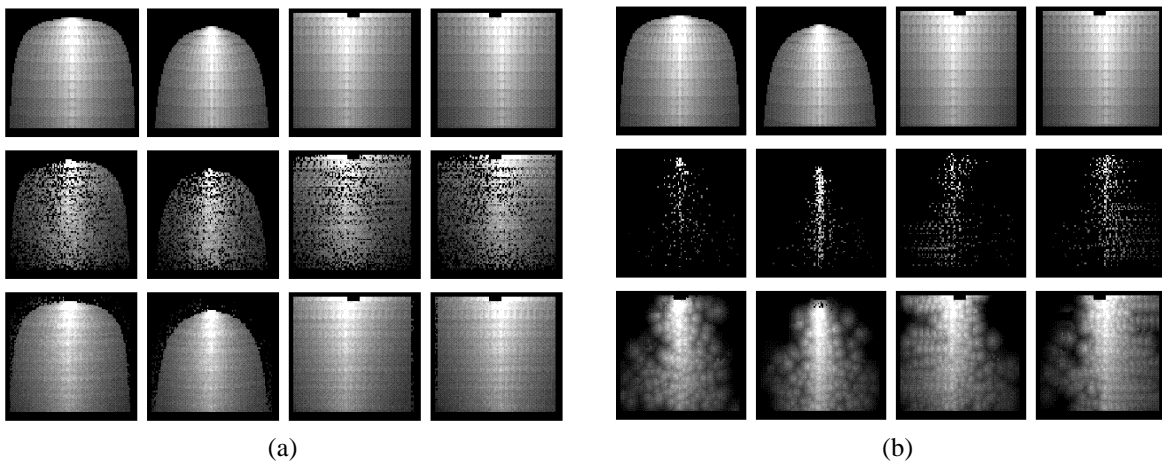


Figure 6. Visual representation of action values learned with  $PQ$  and  $Q(\lambda)$ -learning. (a) Self-learning in 5120 episodes. (b) Teacher-directed learning in 160 episodes. Top to bottom: ground truth,  $Q(\lambda)$ -learning and  $PQ$ -learning. Left to right: forward 30cm, forward 60cm, turn left and turn right. For each figure: x-axis: head angle ( $-50^\circ \sim 50^\circ$  from left to right); y-axis: distance to the goal (0~500cm, from top to bottom).

The next test was using the learned value functions to predict actions. Since what we want is the robot being able to perform rationally to achieve its goal; therefore, in addition to optimal actions, we introduced a concept called the *semi-optimal action*, which was defined as the actions whose  $Q$  values exceeded 90% of that of the optimal actions. With this definition, decision accuracies could therefore be evaluated with respect to either optimal actions or semi-optimal actions. Figure 7 shows both prediction accuracies with the two algorithms. Again, the statistics were all collected from the learned state as in the previous experiment. Although the  $PQ$ -learning had to make more predictions (because more states were learned), it still outperformed the  $Q(\lambda)$ -learning in terms of both accuracies.

### 3.3. Learning under directions

In this experiment, the robot explored the environment by following the directions from a teacher at each step. The direction could be the optimal action or others according to the specified  $\epsilon$ -greedy policy. However, the robot learned only from its own experience, which means that the correctness of the teacher was not required.

The similar learning statistics, as well as the visual representation of value functions, are given in Figure 6(b), 8, 9, 10, respectively, where  $\epsilon=0.4$ . Encouragingly, remarkable accuracies in action prediction and value estimations were achieved with the  $PQ$ -learning in tens of learning episodes. According to the learning performance obtained, one may tell that, in addition to simulations, the proposed method is also desirable for real robot learning by interacting with a teacher.

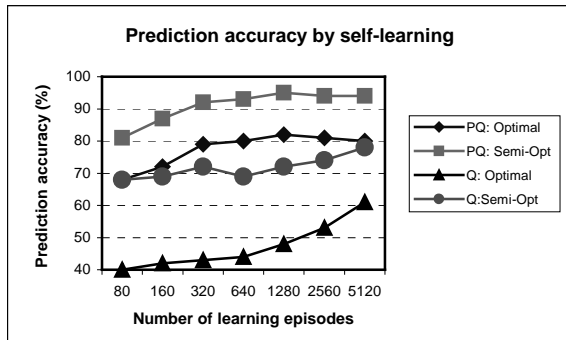


Figure 7. Prediction accuracy by self-learning.

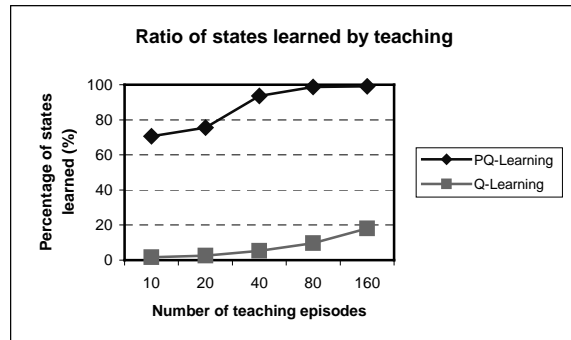


Figure 8. Learning efficiency by directed-learning.

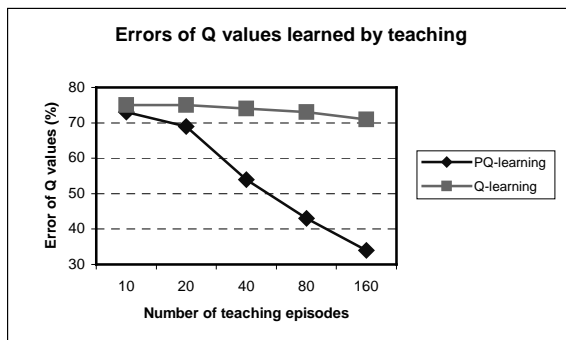


Figure 9. Averaged value errors by directed-learning.

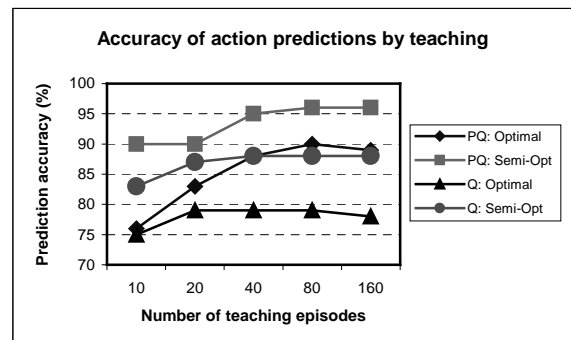


Figure 10. Prediction accuracy by directed-learning.

## 4. Conclusion

An efficient reinforcement learning method, called the  $PQ$ -learning, is presented in this paper. The proposed method is based on Watkins'  $Q$ -learning while a recursive value

propagation process, which consists of spatial propagation and temporal propagation, is introduced to boost the learning in large state spaces. The new algorithm offers several benefits for autonomous robot learning. First, it is a general delayed rewarding based RL method. Second, the learning is guaranteed to converge to the optimum with a much faster speed than the  $Q$  and  $Q(\lambda)$ -learning. Finally, this method supports both self and teacher-directed learning, in which the correctness of the teacher, however, is not required, which offers a relaxed and friendly interacting interface between the human and the robot.

In addition to the above benefits, the proposed method also has some limits in real applications. First, it assumes negative rewards only available when the robot fails the task, which might not be true in some cases. Secondly, this method does not consider uncertainties in state transitions and rewarding. Consequently, the learning might be noise sensitive since the influence of the noise would also be recursively propagated to all related states. How to ensure the robustness of the learning in real world is a crucial but still outstanding issue of this method.

## Acknowledgement

This project is partly supported by the NSF Grant with the award number IIS-00-85980. The authors would like to acknowledge the anonymous reviewers for their invaluable comment and suggestion.

## References

- [1] S. Thrun, "An Approach to learning mobile robot navigation", *Robotics and Autonomous Systems*, vol. 15, pp.301-319, 1995.
- [2] J. Millan and R. del, "Reinforcement learning of goal-directed obstacle-avoiding reaction strategies in an autonomous mobile robot", *Robotics & Autonomous Systems*, vol. 15, pp.275-99, 1995.
- [3] J. Rando and P. Alstrom, "Learning to drive a bicycle using reinforcement learning and shaping", *International Conference on Machine Learning*, pp.463-471, 1998.
- [4] CJCH. Watkins, *Learning from delayed reward*, Ph.D. thesis, Cambridge University, 1989.
- [5] CJCH. Watkins and P. Dayan, "Q-learning", *Machine Learning*, vol. 8, pp.279-92, 1992.
- [6] G. Gordon, "Stable function approximation in dynamic programming", *Proc. of 12<sup>th</sup> Int'l Conf. on Machine Learning*, pp. 261-68, San Francisco, CA, 1995.
- [7] L. Baird, "Residual algorithms: Reinforcement learning with function approximation", *Proc. of 12<sup>th</sup> Int'l Conf. on Machine Learning*, pp. 30-37, San Francisco, CA, 1995.
- [8] S. Thrun and A. Schwartz, "Issues in using function approximation for reinforcement learning", *Proc. of 1993 Connectionist Models Summer School*, Hillsdale, NJ, 1993.
- [9] J. Boyan and A. Moore, "Generalization in reinforcement learning: Safely approximating the value function", *Advances in Neural Informatin Processing Systems*, MIT Press, 1995.
- [10] A. Moore and C. Atkeson, "The Parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces", *Machine Learning*, vol. 21, pp. 199-233, 1995.
- [11] W. Zhu and S. Levinson, "Vision-based reinforcement learning for robot navigation", *Proc. of Int'l Joint Conf. on Neural Network*, vol. 2, pp. 1025-30, Washington DC, July, 2001.
- [12] D. Chapman and L. Kaelbling, "Input generalization in delayed reinforcement learning: An algorithm and performance comparisons", *Proc. of Int'l Joint Conf. on Artificial Intelligence*, pp. 726-31, 1991.
- [13] Y. Takahashi, M. Asada and K. Hosoda. "Reasonable Performance in Less Learning Time by Real Robot Based on Incremental State Space Segmentation", *Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pp. 1518-24, 1996.
- [14] L-J. Lin, *Self-supervised Learning by Reinforcement and Artificial Neural Networks*, Ph.D. Thesis, Carnegie Mellon University, Pittsburgh, PA, 1992.
- [15] R. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, 1957.