We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

13

Proposal and Evaluation of the Improved Penalty Avoiding Rational Policy Making Algorithm

Kazuteru Miyazaki[1], Takuji Namatame[2] and Hiroaki Kobayashi[3]
[1]*National Institution for Academic Degrees and University Evaluation*
[2]*MAZDA*
[3]*Meiji University*
Japan

1. Introduction

Reinforcement learning (RL) is a kind of machine learning. It aims to adapt an agent to a given environment with a reward and a penalty. Traditional RL systems are mainly based on the *Dynamic Programming* (DP). They can get *an optimum policy* that maximizes an expected discounted reward in *Markov Decision Processes (MDPs)*. We know *Temporal Difference learning* (Sutton, 1988) and *Q-learning* (Watkins, 1992) as a kind of the *DP-based RL systems*. They are very attractive since they are able to guarantee the optimality in MDPs. We know that *Partially Observable Markov Decision Processes* (POMDPs) classes are wider than MDPs. If we apply the DP-based RL systems to POMDPs, we will face some limitation. Hence, a heuristic *eligibility trace* is often used to treat a POMDP. We know TD(λ) (Sutton, 1988), Sarsa(λ) (Singh & Sutton, 1996), (Sutton & Barto, 1998) and Actor-Critic (Kimura & Kobayashi, 1998) as such kinds of RL systems.

The DP-based RL system aims to optimize its behavior under given reward and penalty values. However, it is difficult to design these values appropriately for the purpose of us. If we set inappropiate values, the agent may learn unexpected behavior (Miyazaki & Kobayashi, 2000). We know the *Inverse Reinforcement Learning* (IRL) (Ng & Russell, 2000) as a method related to the design problem of reward and penalty values. If we input an expected policy to the IRL systems, it can output a *reward function* that can realize just the same policy. IRL has several theoretical results, i.e. *apprenticeship learning* (Abbeel & Ng, 2005) and *policy invariance* (Ng et.al., 1999).

On the other hand, we are interested in the approach where a reward and a penalty are treated independently. As examples of RL systems that we are proposed on the basis of the viewpoint, we know the rationality theorem of *Profit Sharing* (PS) (Miyazaki et.al., 1994), the *Rational Policy Making algorithm* (RPM) (Miyazaki & Kobayashi, 1998) and *PS-r** (Miyazaki & Kobayashi, 2003). They are restricted to the environment where the number of types of a reward is one. Furthermore, we know the *Penalty Avoiding Rational Policy Making algorithm* (PARP) (Miyazaki & Kobayashi, 2000) and the *Penalty Avoiding Profit Sharing* (PAPS) (Miyazaki et.al., 2002) as examples of RL systems that are able to treat a penalty, too. We call these systems *Exploitaion-oriented Learning (XoL)*.

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

XoL have several features: (1) Though traditional RL systems require appropriate reward and penalty values, XoL only requires *an order of importance* among them. In general, it is easier than designing their values. (2) They can learn more quickly since they trace successful experiences very strongly. (3) They are not suitable for pursuing an optimum policy. The optimum policy can be acquired with *multi-start method* (Miyazaki & Kobayashi, 1998) but it needs to reset all memories to get a better policy. (4) They are effective on the classes beyond MDPs since they are a *Bellman-free method* (Sutton & Barto, 1998) that do not depend on DP.

We are interested in XoL since we require quick learning and/or learning in the class wider than MDPs. We focus on PARP and PAPS especially because they can treat a reward and a penalty at the same time. The application of PARP to a real world is difficult since it requires $O(MN^2)$ memories where N and M are the number of types of a sensory input and an action. Though PAPS only require $O(MN)$ memories, it may learn an irrational policy. In this paper, we aim to reduce the memories of PARP to $O(MN)$ by updating a reward and a penalty in each episode and also selecting an action depending on the degree of a penalty. We show the effectiveness of this approach through a soccer game simulation and its real world experimentation.

2. The domain

2.1 Notations

Consider an agent in some unknown environment. For each discrete time step, after the agent senses the environment as a pair of a discrete attribute and its value, it selects an action from some discrete actions and executes it. In usual DP-based RL systems and PS, a scalar weight, that indicates the importance of a rule, is assigned to each rule. The environment provides a reward or a penalty to the agent as a result of some sequence of actions. In this paper, we foucus on the class where the numbers of types of a reward and a penalty are at most ones, respectively, as same as PARP and PAPS. We give the agent a reward for achievement of our purpose and a penalty for violation of our restriction.

We term the sequence of rules selected between the rewards as an *episode*. For example, when the agent selects *xb*, *xa*, *ya*, *za*, *yb*, *xa*, *za*, and *yb* in Fig. 1 a), there are two episodes (*xb*·*xa*·*ya*·*za*·*yb*) and (*xa*·*za*·*yb*), as shown in Fig. 1 b). Consider a part of an episode where the sensory input of the first selection rule and the sensory output of the last selection rule are the same although both rules are different. We term it as a *detour*. For example, an episode (*xb*·*xa*·*ya*·*za*·*yb*) has two detours (*xb*) and (*ya*·*za*), as shown in Fig. 1 b).

The rules on a detour may not contribute to obtain a reward. We term a rule as *irrational* if and only if it always exist on detours in any episodes. Otherwise, a rule is termed as *rational*. After obtaining the episode 1 of Fig. 1 b), rule *xb,ya* and *za* are irrational rules and rule *xa* and *yb* are rational rules. When the episode 2 is experienced furthermore, rule *za* changes to a rational rule. We term a rule *penalty* if and only if it has a penalty or it can transit to a *penalty state* in which there are penalty or irrational rules only.

The function that maps sensory inputs to actions is termed a *policy*. The policy that maximizes an expected reward per an action is termed as an *optimum policy*. We term a policy *rational* if and only if the expected reward per an action is larger than zero. We term a rational policy a *penalty avoiding rational policy* if and only if it has no penalty rule. Furthermore, the policy that outputs just one action for each sensory input is termed a *deterministic policy*.

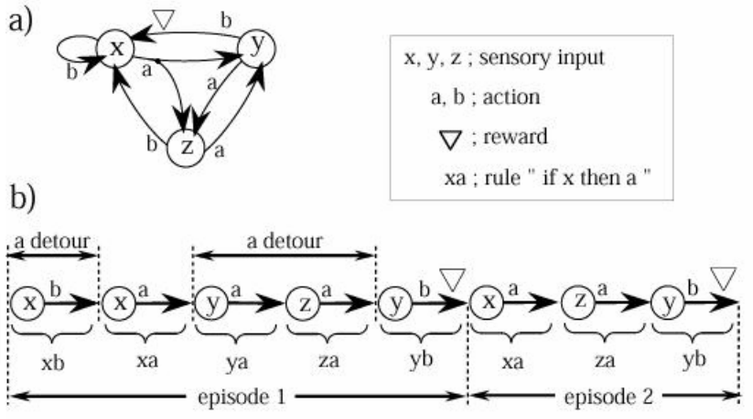


Fig. 1. a) An environment of 3 sensory inputs and 2 actions. b) An example of an episode and a detour

2.2 Previous works

The Penalty Avoiding Rational Policy Making algorithm

We know the *Penalty Avoiding Rational Policy Making algorithm* (PARP) as XoL that can make a penalty avoiding rational policy. To avoid all penalties, PARP suppresses all penalty rules in the current rule sets with the *Penalty Rule Judgment algorithm* (PRJ) in Fig. 2. After suppressing all penalty rules, it aims to make a deterministic rational policy by PS, RPM and so on.

procedure The penalty Rule Judgement (PRJ)
begin
  Put the mark in the rule that has been got a penalty directly
  do
    Put the mark in the following state;
      there is no rational rule or
      there is no rule that can transit to non-marked state
    Put the mark in the following rule;
      there are marks in the states that can be transited by it
  while (there is a new mark on some state)
end

Fig. 2. The Penalty Rule Judgment algorithm (PRJ); We can regard the marked rule as a penalty rule. We can find all penalty rules in the current rule set through continuing PRJ

Furthermore, PARP avoids a penalty stochastically if there is no deterministic rational policy. It is realized by selecting the rule whose penalty level is the least in all penalty levels at the same sensory input. The penalty level is estimated by *interval estimation* of transition probability to a penalty state of each rule. If we can continue to select only rational rule, we can get a penalty avoiding rational policy. On the other hand, if we have to refer to the penalty level, we get a policy that has a possibility to get a penalty.

PARP uses PRJ. PRJ has to memorize all rules that have been experienced and descendant states that have been transited by their rules to find a penalty rule. It requires $O(MN^2)$ memories where N and M are the number of types of a sensory input and an action. Furthermore, PARP requires the same memory to suppress all penalties stochastically. In applying PRJ to large-scale problems, we are confronted with the curse of dimensionality.

The Penalty Avoiding Profit Sharing

We know PAPS as XoL to make a penalty avoiding rational policy with *O(MN)* memories that is the same memory to storage all rules. The original PRJ requires $O(MN^2)$ memories since it scans all of the known rules. On the other hand, PAPS uses *PRJ on episode* (PRJ[episode]) where the scanning is restricted within each episode. Therefore it can find a penalty rule with *O(MN)* memories.

Furthermore, PAPS uses PS[+] and PS[-] to avoid a penalty stochastically with *O(MN)* memories. PS[+] is the same as PS whose weights are reinforced only by a reward. It is used in the case where there is a non-penalty rule in a current state. If there is no non-penalty rule in a current state, PAPS selects an action that is the least reinforced by PS[-] whose weights are reinforced by a penalty only.

PAPS aims to make a penalty avoiding rational policy with *O(MN)* memories. However there is a serious problem in the action selection based on PS[-]. Originally, PS aims to learn a rational policy that reaches to a reward. In the same way, we can get a policy that reaches to a penalty, if we select an action that is the most reinforced by PS[-] in each state. However, we cannot know how rules except for the most reinforced rule are reinforced. Therefore we may select an action that will get a penalty with rather high possiblity, even if we select the least inforced rule in the current state.

3. Improvement of the penalty avoiding rational policy making algorithm

3.1 Basic ideas

We aim to make a penalty avoiding rational policy with *O(MN)* memories. A penalty rule is found by PRJ[episode] as well as PAPS. Therefore, we can find a penalty rule with *O(MN)* memories.

If we cannot select any non-penalty rules in a current state, we should avoid a penalty stochastically. It is realize with penalty level of each rule as PARP. PARP has to memorize all state transitions in order to calculate the penalty level. It means that it requires $O(MN^2)$ memories. In order to reduce the memory to *O(MN)*, we aim to calculate the penalty level by each episode.

3.2 Approximation of the penalty level

We do not memorize all state transitions but memorize only state transitions of each episode to estimate the penalty level of each rule. It only requires *O(MN)* memories. The penalty level of rule S_1a, $PL(S_1a)$, is calculated by the following equation:

$$PL(S_1a) = \frac{N_p(S_1a)}{N(S_1a)} \tag{1}$$

where $N_p(S_1a)$ is the number of times that rule S_1a has judged as a penalty rule, and $N(S_1a)$ is the number of times that the rule has been selected so far. We show examples of $N_p(S_1a)$ and $N(S_1a)$ at a rectangle of Fig. 3 (b) and ellipses of Fig. 3 (a)(b), respectively.

The range of $PL(S_1a)$ is $1.0 \geqq PL(S_1a) \geqq 0.0$. If $PL(S_1a)$ is close to 1.0, we can regard that the possibility of getting a penalty by the action a in the state S_1 is high. On the other hand, if $PL(S_1a)$ is close to 0.0, we can regard that the possibility is low. We can calculate $PL(S_1a)$ every episodes. It requires the following two types memories only. One is the number of

times where each rule has been selected until now. The other is that where each rule has been judged to a penalty rule. They only require *O(MN)* memories.

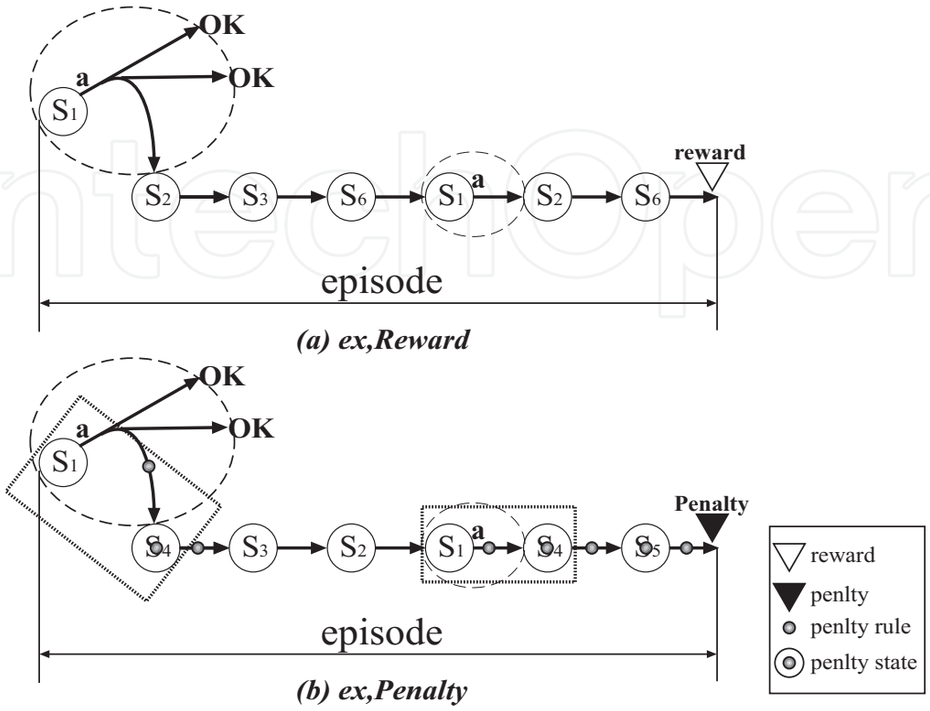


Fig. 3. Approximation of the penalty level

3.3 The action selection based on the penalty level

If there is a rational rule in the current rule set after excluding all penalty rules, we should select the rule to make a penalty avoiding rational policy. It is realized by PS(PS[+]) or RPM. On the other hand, if we cannot select such rule, we select an action in the rule whose penalty level calculated by equation (1) is the least in order to reduce the probability of getting a penalty.

From section 2.1, the rule that has a possibility to get a penalty is defined as a penalty rule. This definition has a possibility of regarding all rules as penalty rules. We introduce the cutting parameter γ ($1.0 \geqq \gamma \geqq 0.0$) to reduce the number of penalty rules. If $PL(S_1a)$ is larger than γ, the rule S_1a is a penalty rule. Otherwise, it is not a penalty rule.

It means that the rule is not regarded as a penalty rule if the possibility of getting a penalty is low. If we set $\gamma=0.0$, the rule that has been given a penalty even once is regarded a penalty rule. It is coincident to the case where we do not introduce the parameter γ to PARP. On the other hand, if γ is closed to *1.0*, the number of penalty rules will decrease, and in turn, the number of rules that can be selected by PS(PS[+]) or RPM will increase significantly. If we set $\gamma=1.0$, it is coincident to PS(PS[+]) or RPM that do not avoid any penalty. We can control the number of rules that can be selected with γ.

3.4 Features

We call the proposal method *Improved PARP*. Improved PARP has the following features.

- We can avoid a penalty stochastically with *O(MN)* memories through combining of both PRJ[episode] and the approximation of a penalty level.
- We can cotrol the number of penalty rules with the cutting parameter γ.

4. Numerical experimentation

We use the simulator shown in Fig. 4 that is based on the Small Size Robot League on RoboCup. There are two learning agents for one side. One agent puts a pass out for the other agent. The other agent aims to receive it. There is an opponent agent for the other side. The agent aims to cut off the pass. We show the initial positions of these agents in Fig. 4. We know the *keepaway task* (Stone et.al. 2005) as another soccer game. The performance of the keepaway task strongly depneds on the designing of a reward and a penalty (Tanaka & Arai, 2006). In the future, we will challenge to the keepaway task, too.

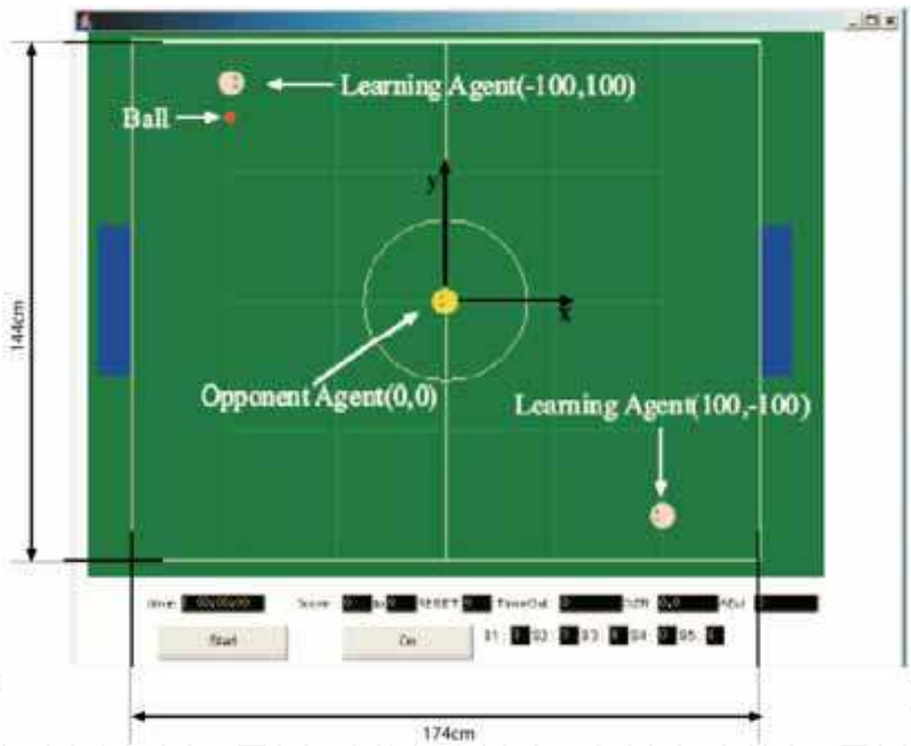


Fig. 4. Soccer game simulator

4.1 Detail design of the task

The learning agents

The state spaces of a learning agent is classified to the following five cases; 9 states of a ball from the learning agent (Fig. 5 a)), 10 states of an opponent agent from the learning agent (Fig. 5 b)), 5 states of the other learning agent from the learning agent (Fig. 5 c)), 4 states of destination between two learning agents (Fig. 5 d)) and 5 states of the learning agent from the other learning agent (Fig. 5 c)).

There are the following seven actions; stop, go forward, turn by five degrees to direction of a ball, turn right by five degrees, turn left by five degrees, kick for dribble and kick for pass.

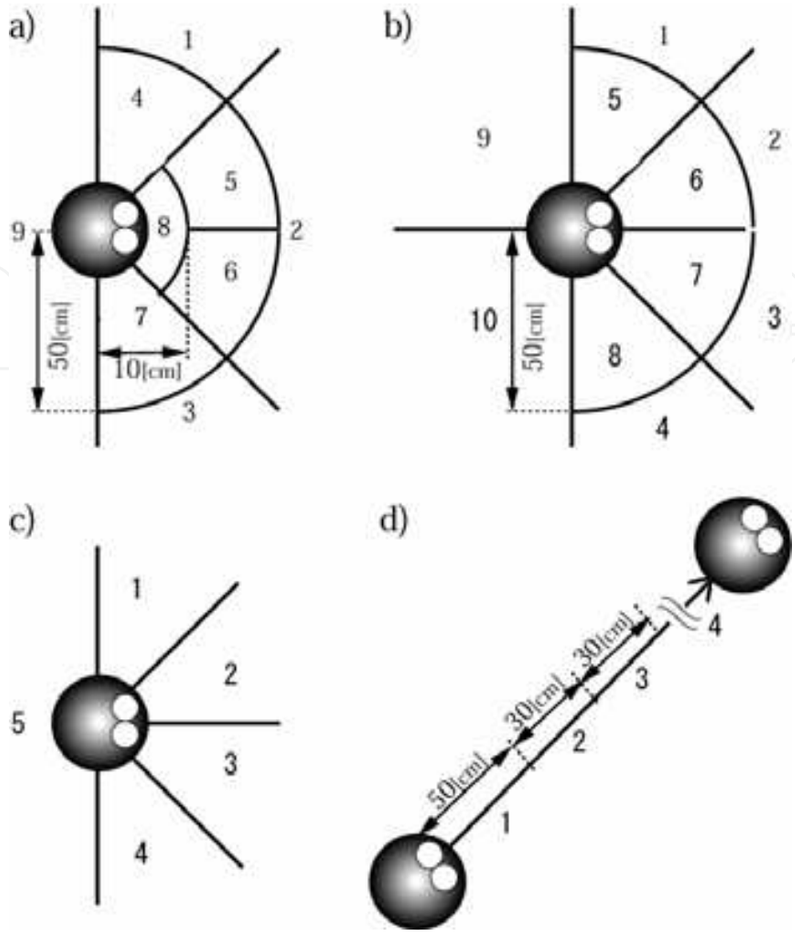


Fig. 5. a) 9 states of a ball from the learning agent. b) 10 states of an opponent agent from the learning agent. c) 5 States between two learning agents. d) 4 states of destination between two learning agents

A reward and a penalty

The learning is judged to *success* when a ball in a learning agent had reached to the other learning agent and the distance between two learning agents is larger than 40cm. A reward is given to two learning agents when the condition for success is achieved. A penalty is given to two learning agents when a ball had reached to an opponent agent, the learning time had exceeded 12000 msec that is called *TimeOut*, or a ball in a learning agent had reached to the other learning agent but the distance between two learning agents is not larger than 40cm. Though there are three types of penalties, learning agents do not distinguish them, that is, they are treated as the same penalty. In the future, we will try to the case where these penalties are treated independently.

The opponent agent

There are two policies in the opponent agent. One policy aims to reach to a ball. The other policy aims to reach to the middle position of two learning agents to block the pass. The former is selected when the distance from the agent to a ball is closer than the distance from the agent to the middle position of two learning agents. Otherwise, the latter policy is selected.

Setting of the experiment

The learning agents sense the environment each 3 msec. When they had received a reward or a penalty, the trial has been ended and new trial begins with the initial position. We take 100 experiments with different random seeds.

We introduce two noises to simulate a stochastic state transition. One is added to the movements of all agents and a ball. The maximum value is *±1/10 [mm/s]*. The other is added to the angles of the turn actions of learning agents and the direction of a ball when agents kicked it. The maximum value is *±1/100 [deg]*.

4.2 Results and discussion

About the setting of γ

We have compared our proposed method called Improved PARP with PAPS. On the other hand, PARP cannot apply to this task since it requires many memories. We have changed γ every *0.1* from *0.5* up to *1.0*. Fig. 6 and Fig. 7 are the results of success rates of pass behaviors of PAPS and Improved PARP, respectively. The horizontal axis is the number of trials and the vertical axis is the percentage of the success rate. One trial includes a pair of sensing the environment and selecting an action. The plots are shown every 100 trials. The initial point of these plots are the results of random walk.

Fig. 6 shows that PAPS cannot learn. It comes from the fact that the penalty avoiding by PS[-] does not function properly. On the other hand, we can conclude that Improved PARP gets high performance than PAPS from these figures.

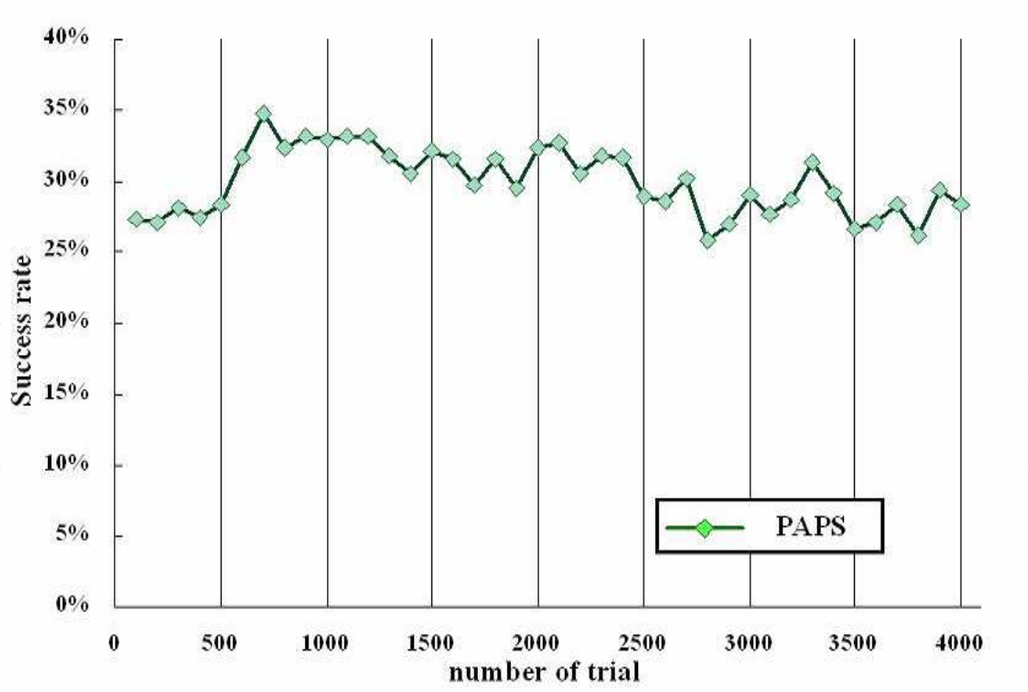


Fig. 6. Success rate (%) of pass behavior by PAPS

If we decrease γ, the number of penalty rules of Improved PARP increases. It means that the penalty avoiding is more important than the reward getting. Therefore, if we decrease γ, the

action selection based on PS(PS[+]) decreases, and the success rates become bad as shown in Fig. 7. On the other hand, if we increase γ until *0.8*, the success rates become good. It comes from the fact that the action selection based on PS(PS[+]) increases because of decreasing the number of penalty rules. However, the results of γ=*0.9* and *1.0* show bad. It means that if we set γ excessively largely, the learning agents regard an important penalty rule as a non-penalty rule.

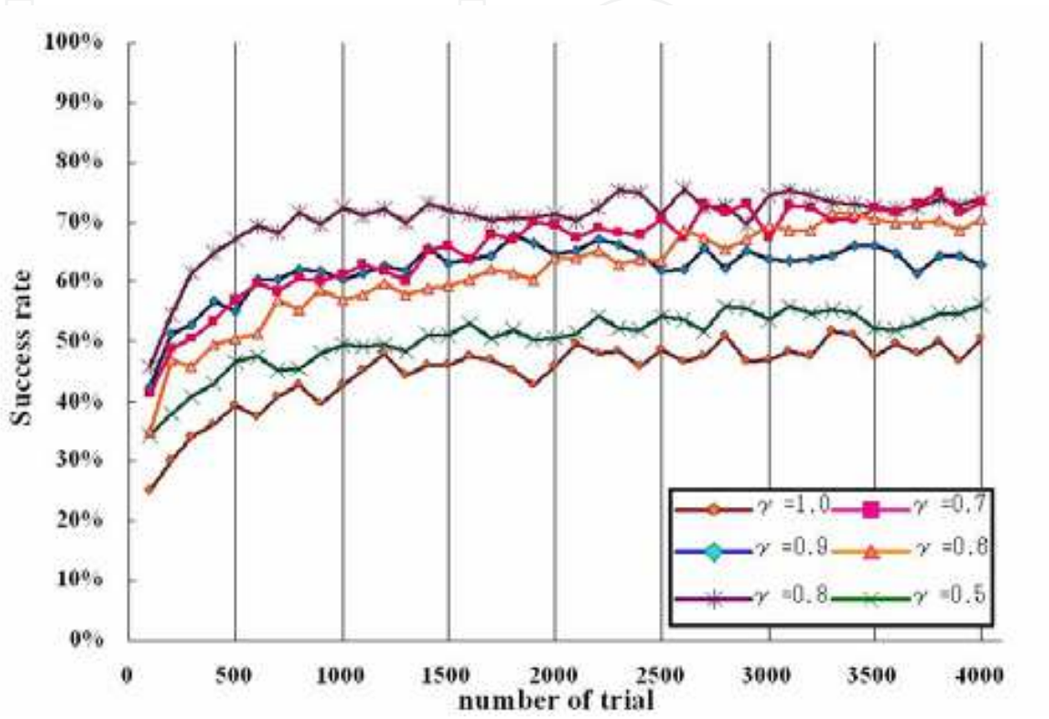


Fig. 7. Success rate (%) of pass behavior by Improved PARP

The result of Improved PARP depends on γ. If we set γ small, we can get a policy that is focused on a penalty avoiding. If we set γ large, we can get a policy that is focused on a reward getting. Therefore, we should change γ dynamically. For example, we should set γ large to get a goal in the initial phase of a game or in the situation where the game may lose. On the other hand, we should set γ small to protect our goal when our score being more than the other.

About the setting of penalties

We give a learning agent the same penalty when the learning reaches TimeOut, a ball had reached to an opponent agent, or a ball in a learning agent had reached to the other learning agent but the distance between two learning agents is not larger than 40cm. We confirm the effects of first two penalties. Fig. 8 shows the TimeOut rate plotted against the number of trials. Fig. 9 shows miss rate of pass behavior plotted against the number of trials.

Though we can confirm the effect of learning in Fig. 8, we cannot do it in Fig. 9. It comes from the fact that the policy that is specialized in TimeOut has been learned than the other policies, since TimeOut occurs frequently than the others. Therefore, we should not give the same penalty to the different types of behaviors. In the future, we aim to combine Improved

PARP with the method in the paper (Miyazaki & Kobayashi, 2004) that can treat several types of rewards and penalties at the same time.

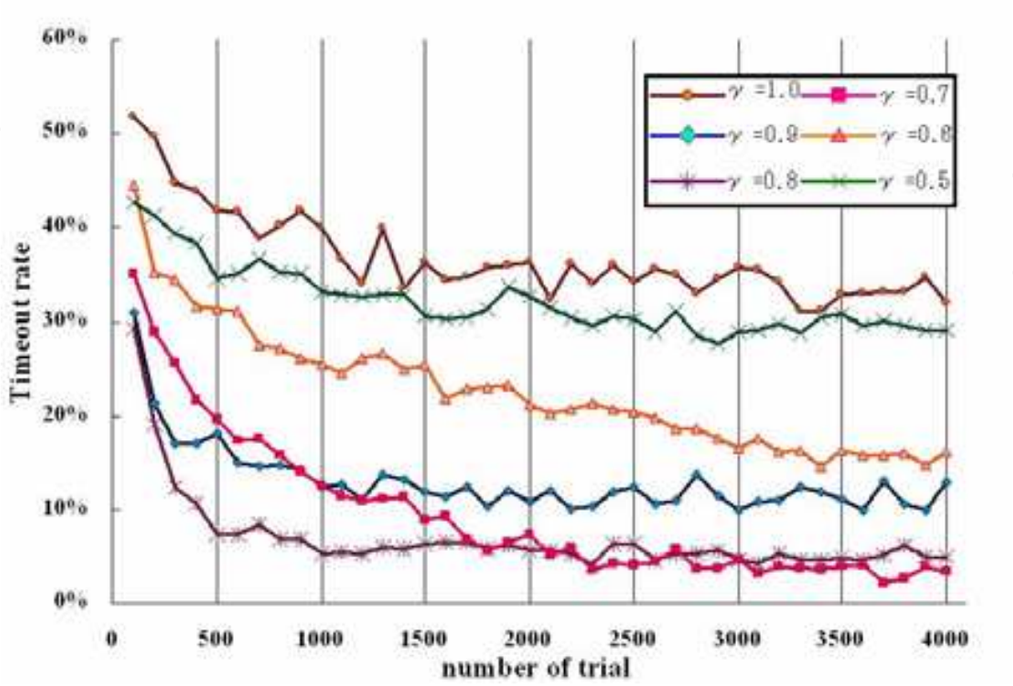


Fig. 8. Timeout rate (%) by Improved PARP

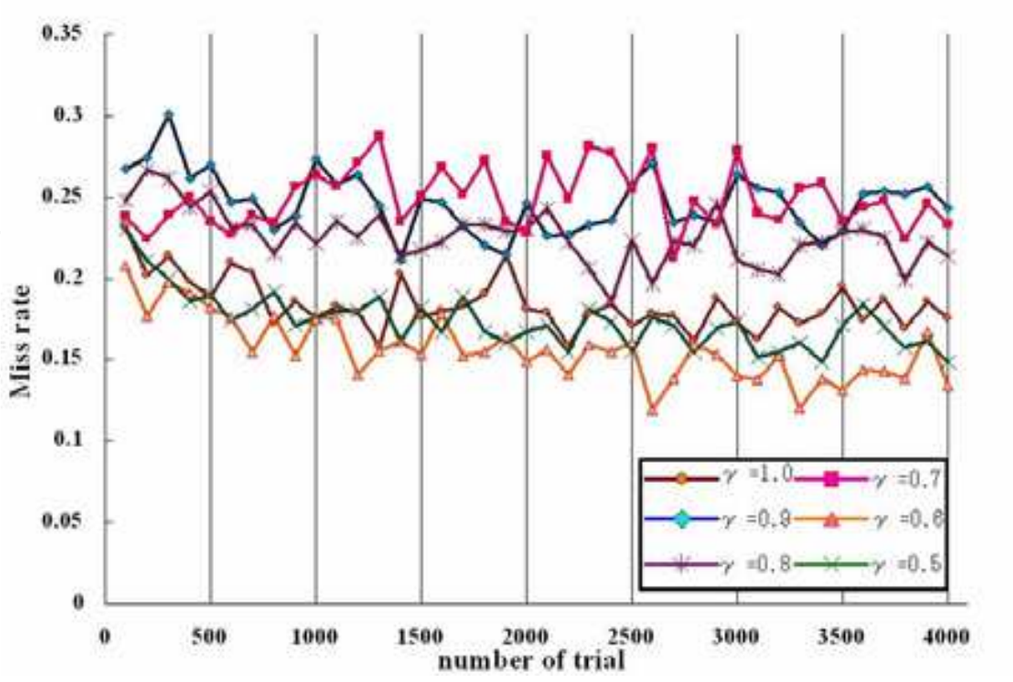


Fig. 9. Miss rate of pass behavior by Improved PARP

5. Real world experimentation

5.1 The soccer robot system

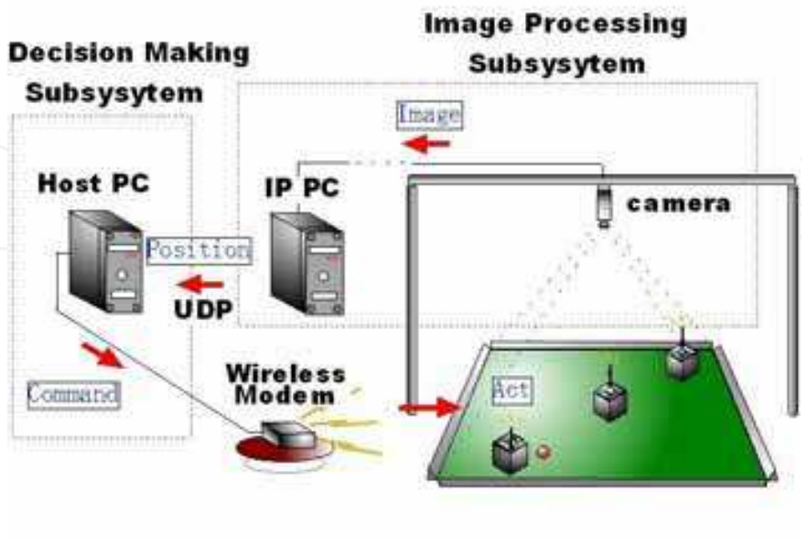


Fig. 10. The soccer robot system

We have developed a soccer robot system (Fig.10) to evaluate impoved PARP. It has the decision making subsystem and the image processing subsystem. The former is to decide an action and the latter is to process images from camera. It can adapt on the reguration of Robo Cup Small Size Robot League excet for the field size.

5.2 Soccer robot

We use three robots called Robo-E (Fig. 11). It has 12 light sensors. They are used to escape from clashing with a wall or another robots. Each robot has different markers each other to distinguish each robot. We show the combinations of colors for robots in table.1.

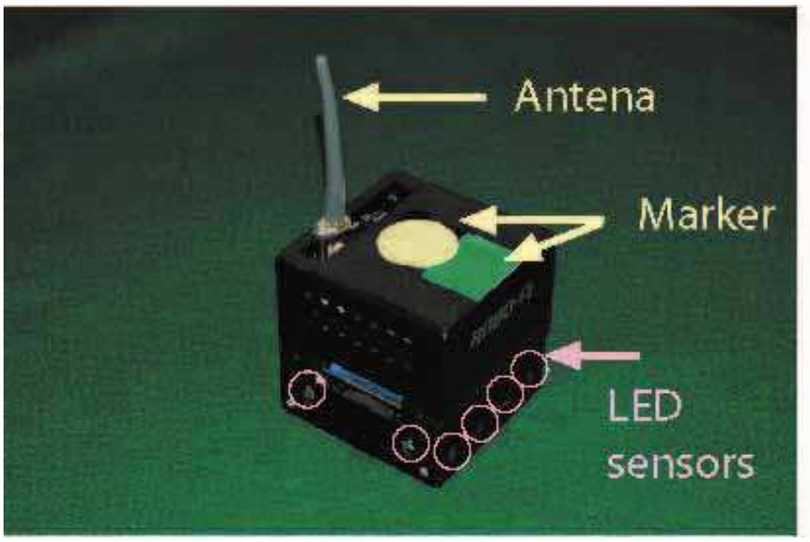


Fig. 11. Soccer Robot Robo-E

Robot 1	Yellow and light green
Robot 2	Yellow and Cyan
Robot 3	Blue and light green

Table 1. Combinations of colors for robots

5.3 Host system

The host system is consturcted by two computers, that is, Host and Image Processing (IP) PCs. IP and Host PCs are connected by TCP/IP. We use UDP (User Datagram Protocol) as the protocol on IP.
Host PC is to decide an action. We have impremented Improved PARP on Host PC. The decision of Host PC is transmitted by willess modem.

Fig. 12. 3CCD Camera

On the other hand, IP PC is to process images from camera. It calculates positions of robots and a ball. Their images are given by the camera (SONY XC-003; Fig. 12). We had the camera mounted at the top of the field to get global vision. We use machine vision tool *HALCON* with its *Hdevelop* programming environment to get robots and a ball positions.
The positions of each robots are calculated with markers on Table. 1 and their body color (black). We use RGB parameter to distinguish each color. Table 2 shows these threshholds.

Color	R	G	B
Field (green)	10-40	75-100	40-65
Robot (black)	0-20	0-55	0-220
Wall (white)	255	255	255
Ball (orange)	125-200	95-125	45-85
Yellow	170-255	180-255	110-190
Blue	30-110	40-30	80-200
Light green	40-130	130-230	60-180
Cyan	100-170	140-240	140-240

Table 2. RGB parameter of each color

The position of the robot is the center of gravity of two markers. It is (X_{robot}, Y_{robot}) on Fig. 13 (a). The position of a ball is the center of ball marker (orange). It is (X_{ball}, Y_{ball}) on Fig. 13 (b).

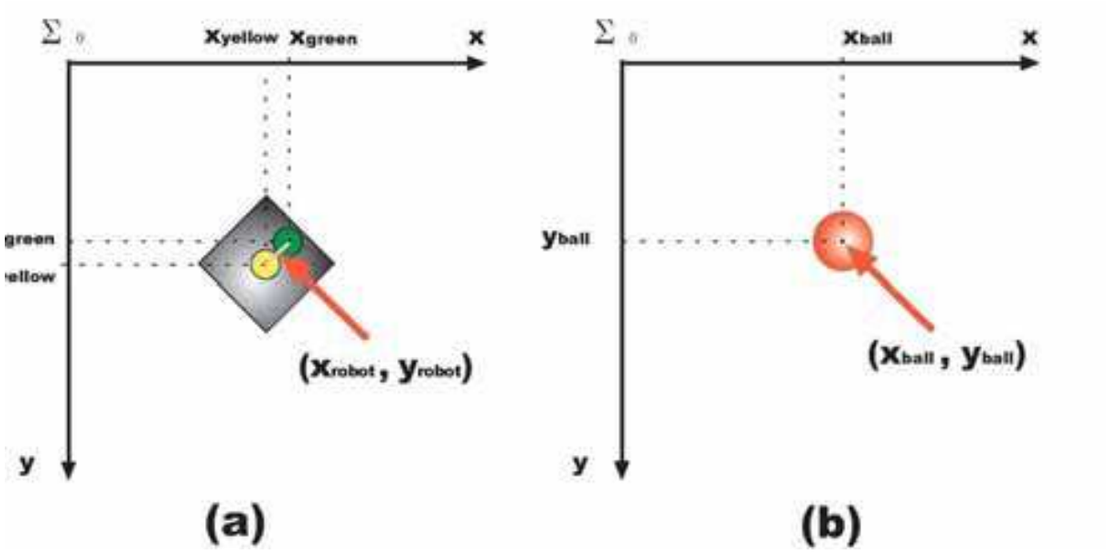


Fig. 13. Positions of robots and a ball

The orientation of the robot is calculated by the center of gravity of two markers (Fig. 14). If we set the center of gravity of center marker (X_{center}, Y_{center}) and it of another marker ($X_{outside}$, Y_{outsde}), we can get the angrle θ is as the following,

$$\theta = \text{atan2}(d_{y,} d_x), \quad (180° < \theta < 180°) \tag{2}$$

where dx, dy is $d_x = x_{outside} - x_{center}, d_{y=} y_{outside} - y_{center}$.

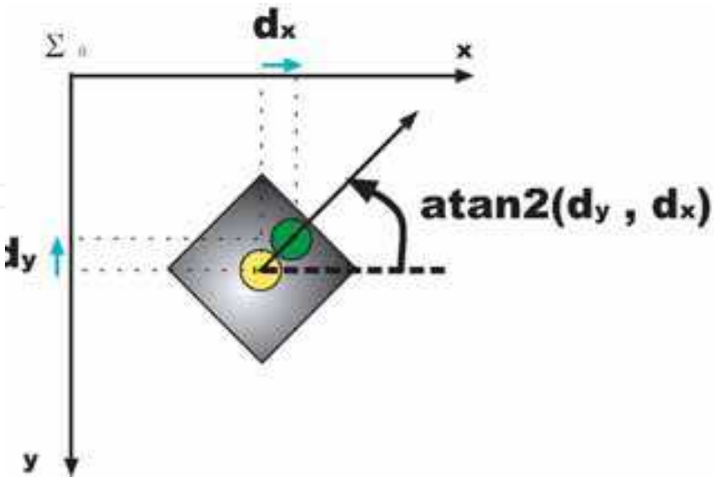


Fig. 14. Orientation of a robot

5.4 Experimental results

We use three Robo-E type robots in our real world experimentation. One is an opponent agent and the others are learning agets. Three robots and a ball are set on the field in Fig.15.

Fig. 15. Real world field

Fig. 16 shows the results of success rates of pass behavior of Improved PARP, the Q-learning and random agets. Improved PARP has 4 experiments with different random seeds from SEED_1 to SEED_4. We set γ=*0.8*. We can confirm the effectiveness of Improved PARP as same as numerical experimentation. Especially, it can learn very quickly than Q-learning agent that is the most famous DP-based RL system.

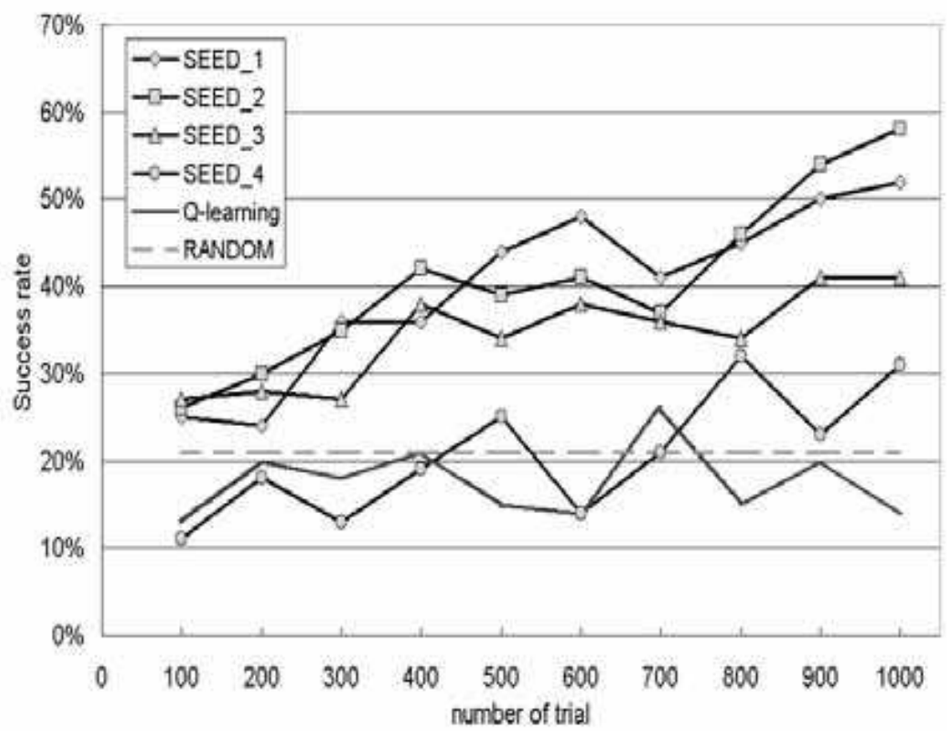


Fig. 16. Success rate (%) of pass behavior in real world experimentation

6. Conclusion

As examples of XoL that use both a reward and a penalty simultaneously, we know PARP and PAPS. The application of PARP to real worlds is difficult since it requires $O(MN^2)$ memories where N and M are the number of types of a sensory input and an action. Though PAPS only requires $O(MN)$ memories, it may learn unexpected behavior.

In this paper, we have proposed *Improved PARP* in order to overcome the difficulty by updating a reward and a penalty in each episode and selecting actions depending on the degree of a penalty. We have shown the effectiveness of this approach through a soccer game simulation and its real world experimentation.

Improved PARP cannot treat multiple rewards and penalties. In the future, we try to combine Improved PARP with the method in the paper (Miyazaki & Kobayashi, 2004) to treat several types of rewards and penalties at the same time. Furthermore, we will apply Improved PARP to the *keepaway task* (Stone et.al. 2005) and the other real world applications.

7. References

Abbeel, P. & Ng., A. Y. (2005). Exploration and apprenticeship learning in reinforcement learning. Proceedings of the 21th International Conference on Machine Learning, pp.1-8

Kimura, H. & Kobayashi, S. (1998). An analysis of actor/critic algorithms using eligibility traces: reinforcement learning with imperfect value function. Proceedings of the 15th International Conference on Machine Learning, pp. 278-286.

Miyazaki, K. & Kobayashi, S. (1998). Learning Deterministic Policies in Partially Observable Markov Decision Processes. Proceedings of the International Conference on Intelligent Autonomous System 5, pp. 250-257.

Miyazaki, K. & Kobayashi, S. (2000). Reinforcement Learning for Penalty Avoiding Policy Making. In Proceeding of the 2000 IEEE International Conference on Systems, Man and Cybernetics, pp. 206-211.

Miyazaki, K. & Kobayashi, S. (2003). An Extension of Profit Sharing to Partially Observable Markov Decision Processes: Proposition of PS-r* and its Evaluation. Journal of the Japanese Society for Artificial Intelligence, Vol.18, No.5, pp. 166-169. (in Japanese).

Miyazaki, K. & Kobayashi, S. (2004). Reinforcement Learning in Multiple Rewards and Penalties Environments. Proceedings of Joint 2nd International Conference on Soft Computing and Intelligent Systems and 5th International Symposium on Advanced Intelligent Systems.

Miyazaki, K., Yamamura, M. & Kobayashi, S. (1994). On the Rationality of Profit Sharing in Reinforcement Learning. Proceedings of the Third International Conference on Fuzzy Logic, Neural Nets and Soft Computing, pp. 285-288.

Miyazaki, K, Saitou, J. & Kobayashi, H. (2002). Reinforcement Learning for Penalty Avoiding Profit Sharing and its Application to the Soccer Game. Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning, pp. 335-339.

Ng, A. Y. & Russell, S. J. (2000). Algorithms for Inverse Reinforcement Learning. Proceedings of the 17th International Conference on Machine Learning, pp. 663-670.

Ng, A. Y., Harada, D. & Russell, S. J. (1999). Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping Proceedings of the 17th International Conference on Machine Learning, pp. 278-287.

Singh, S. P. & Sutton, R. S. (1996). Reinforcement Learning with Replacing Eligibility Traces. Machine Learning, Vol.22, pp. 123-158.

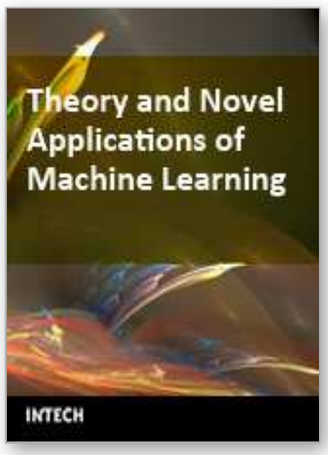
Stone. P., Sutton, R. S. & Kuhlmann, G. (2005). Reinforcement Learning for RoboCup Soccer Keepaway. Adaptive Behavior, Vol.13, No.3, pp. 165-188.

Sutton, R. S. & Barto, A. (1998). Reinforcement Learning: An Introduction. A Bradford Book, The MIT Press.

Sutton, R. S. (1988). Learning to Predict by the Methods of Temporal Differences. Machine Learning, Vol.3, pp. 9-44.

Tanaka, N. & Arai, S. (2006). Teamwork Formation for Keepaway in Robotics Soccer (Reinforcement Learning Approach), Pacific Rim International Workshop on Multi-Agents, Lecture Notes in Computer Science, Vol.4088, pp. 279-292.

Watkins, C. J. H. & Dayan, P. (1992). Technical note: Q-learning.Machine Learning, Vol.8, pp. 55-68.



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.

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

IntechOpen