Reinforcement learning with delayed rewards on continuous state space

Hajime Kimura, Masayuki Yamamura, Shigenobu Kobayashi
Department of Intelligence Science, Graduate School of Interdisciplinary Science and Engineering,
Tokyo Institute of Technology, 4259,Nagatsuda,Midori-ku,Yokohama,227 JAPAN

Abstract

This paper extends reinforcement learning to treat continuous state environments.

First, we propose a mathematical model that represents a Markov decision process where each state corresponds to some region in continuous space. Second, we present a learning system that consists of state analyzer, stochastic action selecter, and incremental learner with DSG(discounted sum of gradient) method. We show efficiency of the proposed learning system in comparison with Q-learning through some examples. Finally,we apply it to a real task of robotics.

1 Introduction

Reinforcement learning aims to adapt a system to a given environment according to delayed rewards. The objective of learning is to find an optimal control policy, that is, a mapping from situations to actions so that maximize its performance. In this paper,the performance measure is defined as expected rewards per learning step. The learning system is not told about the environment, so it must explore by trying a variety of actions. There are two issues to handle delayed reward and uncertainty. The environment which is treated in reinforcement learning can be classified according to two characteristics. One is types of state space, the other state transition. Types of state space are divided into discrete and continuous. Types of state transition are divided into Markovian and non-Markovian.

Conventional works can be classified into an identification intensive approach and reinforcement intensive one. The former forcuses upon correctly identifying the environment, the later does upon rapidly reinforcing experiences. Q-learning[Watkins 92] is a representative of identification intensive approach, which can treat not only discrete state environments, but also continuous state[Lin 93]. Bucket brigade[Goldberg 89] and profit sharing[Grefenstette 88],[Miyazaki 92] in classifier systems are reinforcement intensive approaces, which can be only used for discrete state environments. There are few work on reinforcement intensive methods under continuous environments.

This paper extends reinforcement intensive learning to treat continuous state environments. Table1 shows a classification of reinforcement learning methods.

Table 1: Classification of reinforcement learning methods

	Identification intensive	Reinforcement intensive
Discrete environments	Q-learning	bucket-brigade profit-sharing
Continuous environments	Q-learning augmented by neural networks	DSG method (presented in this paper)

2 The domain

This paper discusses reinforce learning for a class of continuous and Markovian environments. We define these environments by extending a usual Markov decision process shown in Fig.1 . A state is defined as a region of sensory input vector space. The learning system senses a point in some region. The state transition follows Markovian rules. The learning system have to divide continuous sensory state space into appropriate state regions, and find an optimal control policy in a Markov decision process.

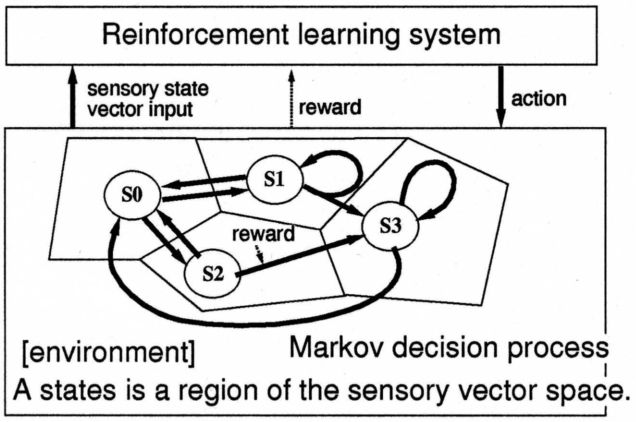
Reinforcement learning system
sensory state vector input
reward
action
S0
S1
S2
S3
reward
[environment]
Markov decision process
A states is a region of the sensory vector space.

Figure 1: The target environment.

3 Extended reinforcement learning system

In this section,we propose a reinforcement intensive learning algorithm. It is incremental procedure and can apply to continuous state environments. A framework of proposed learning system is shown in Fig.2 .

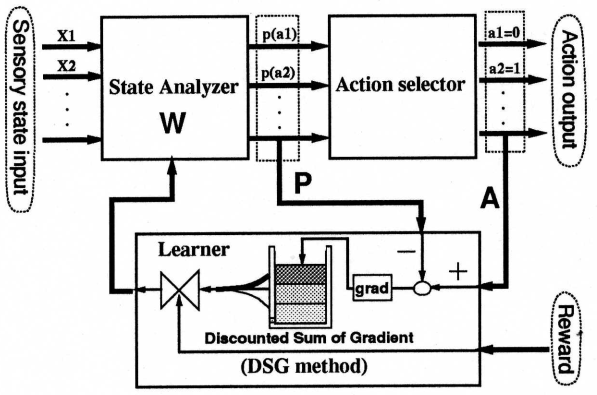

Figure 2: A proposed learning system.

A proposed reinforcement learning system is composed of a learner, a stochastic action selector and a state analyzer as shown in this figure. In the following,a function of each component and a flow of control are explained.

3.1 State analyzer

t shows a present time step, and $t-1$ shows a time step before 1 step from the present. The state analyzer receives a vector of continuous values as a state input from the environment. The state input vector is denoted as $\boldsymbol{X}(t)$. We suppose that there are k kinds of alternative actions. The state analyzer transforms a state input vector $\boldsymbol{X}(t)$ into a k-dimensional vector $\boldsymbol{P}(t)$ by using an inside vector $\boldsymbol{W}$. Here,$\boldsymbol{W}$ is updated by the learner as descrobed later. Each signal $p(a_i)$,that is the i-th element of $\boldsymbol{P}(t)$, denotes a weight of actiona_i. The range of $p(a_i)$ is from 0 to 1.

3.2 Action selector

An action selector receives a vector $\boldsymbol{P}(t)$ from the state analyzer. It chooses an action stochastically by the so-called roulette wheel selection according to $\boldsymbol{P}(t)$. Then it outputs an action vector $\boldsymbol{A}(t)$. Here,$\boldsymbol{A}(t)$ is a k-dimensional unit vector. If i-th element of $\boldsymbol{A}(t)$ is 1, then action a_i is taken up.

3.3 Learner

The difference between a vector $\boldsymbol{P}(t)$ and an action vector $\boldsymbol{A}(t)$ is recognized by the learner. An error function $E(t)$ is defined as

$$E(t) = |\boldsymbol{A}(t) - \boldsymbol{P}(t)|^2 .$$

By using a gradient of the error function $E(t)$ about the inside variable $\boldsymbol{W}$ of the state analyzer, a vector $D(t)$ is defined as

$$\boldsymbol{D}(t) = -\nabla E(t) + \beta \boldsymbol{D}(t-1),$$

here

$$\nabla E(t) = (\frac{\partial E(t)}{\partial W_1}, \frac{\partial E(t)}{\partial W_2}, ...).$$

β is a constant from 0 to 1. If the learner receives a reward r, an inside variable $\boldsymbol{W}$ of the state analyzer is updated by the following equation.

$$\boldsymbol{W} \leftarrow \boldsymbol{W} + r\boldsymbol{D}(t)$$

If the learner receives no reward, $\boldsymbol{W}$ is not updated. We call this learner DSG(Discounted Sum of Gradient) method.

3.4 Advantages of the proposed system

The proposed learning algorithm with the incremental procedure has three advantages. First,a calculation cost is not so high. Second,it requires a few memory capacity. Last,parallel implementation on hardware is possible. The DSG method has the same characteristic as Q-learning because the both have incremental procedures.

Furthermore,the design of state analyzer under the given environment is easy by the following reason. Environments are limited to Markov decision process of which states are regions of the state space,so the most suitable action against a state input is invariable. Therefore, the reinforcement learning system must get an invariable mapping from state input to action output in the final stage. This means that the probability signal vector $\boldsymbol{P}(t)$ must converge to an unit vector. In other words,the state analyzer should divide state space into only 2 values correctly about each behavior a_i in the final stage.

As mentioned above,it is easy to design a function of the state analyzer. In addition, the evasion of inappropriate behaviors in the middle stage and the speed-up of the convergence of the state analyzer can be expected.

3.5 A simple example

3.5.1 The test environment

The following exercise is considered to confirm that the proposed reinforcement learning system behaves as the expectation. We suppose that are four kinds of actions and states. At each state only one action is effective and the others are ineffective. The state transitional

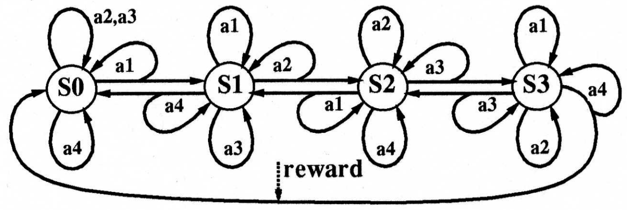

Figure 3: An environment of the experiment.Action attributes of the state space is linearly separable.

diagram of this environment used for the experiment is shown in Fig.3 . The objective of experiment is to confirm that the system can learn only effective actions properly.

Arrows in the figure shows actions,and the branching of an arrow indicates a stochastic transition. Each state is a super-cube territory with a range of ± 0.5 around the following coordinates in each axis direction; $S0 = (1,0,0,0)$,$S1 = (0,1,0,0)$, $S2 = (0,0,1,0)$,$S3 = (0,0,0,1)$.

When the system is transitted from state x to y, it is moved to somewhere in the y territory with an uniform ditribution. If actions are chosen at random in the environment such as Markov decision process, the system gets rewards at the theoretical expected interval of 80 steps. Those intervals are called episode steps in the following. If the most suitable actions are chosen, an optimal length of episode step is 8.

3.5.2 The system used for the experiment

All attributes of the state space about an action a_i are linearly separable. Therefore, the neural network of 2 layers used as a state analyzer of the DSG system, and an experiment was performed. An output of the state analyzer must be $0 < p(a_i) < 1$. So the Sigmoid function was used for each unit which had two saturation points,0 and 1. β was set to 0.7.

An experiment was performed and compared with the 1 step Q-learning system that had the state analyzer same as DSG system. The Q-vector according to actions is outputted from the state analyzer of Q-learning against a state input. The neural network of 2 layers against the above was used for the Q-learning system. But it wasn't restricted an upper bound and a lower bound of Q-value, so a linear function was used for each unit. A discount rate γ was set to 0.7. As an action selector of Q-learning system, an usual method was used shown in the following . 10% of the time an action is selected at random, and 90% the action with maximum Q value is selected. The numbers of parameters of DSG method and Q-learning was adjusted to the same.

3.5.3 Simulation results

The average and a standard deviation of episode steps on 100 trials are shown in Fig.4 .

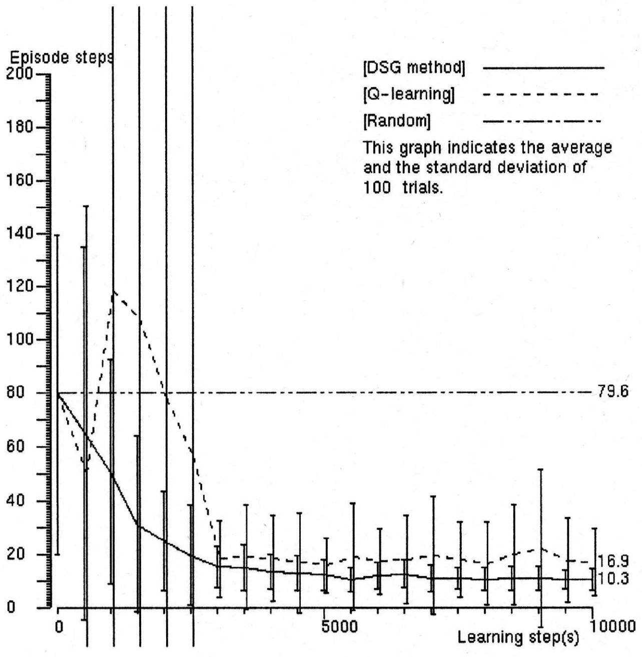

Figure 4: A plot of the average number of episode-steps to solve the linearly separable environment.

A vertical axis shows its episode step, and a horizontal axis does learning steps. The point of 0 step of the horizontal axis means the performance when actions were chosen at random.

A standard deviation of DSG method is smaller than Q-learning system, therefore it is clear that the DSG method is more efficient than Q-learning.

4 Application

The system learned a task of storing a block in the position of the right hollow using a robot hand as shown in Fig.5 . Because it is unacceptable in the restriction of the dimension to move a seized block into the hollow, the closed robot hand must push a free block at the end of the arrangement. The information that it can be measured is the hand position x_1 and the block position x_2 and the hand condition x_3 , and it is inputted as a continuous state vector to the system. Behavior output from the system is four ways of "open the hand(open)", "close the hand(close)", "move the hand 1step righthand(right)", "move the hand 1step lefthand(left)". A state don't change even if "open" is carried out under the condition that a hand has been already opened, and "close" is carried out under the condition that a hand has been already closed either. When "right" or "left" is carried out, a hand is moved for the distance of only 1 step with the error of the uniform ditribution. Even if a block isn't taken, a block can be pushed by the closed hand. A block can't be moved by the opened hand. It can't be moved any further when a hand or a block touches a wall. Even if a hand moves how, a block or a hand never does rotation. A block and a hand are returned to the initial state after a reward is given to the system when a block is pushed into the right clearance completely. The dimension used for the experiment is as the following.

Width of the hand	:	0.24
Width of the block	:	0.40
Range of the block movement	:	$0 < x_2 < 1.0$
Length of the hand's 1 step movement	:	0.315 ± 0.105

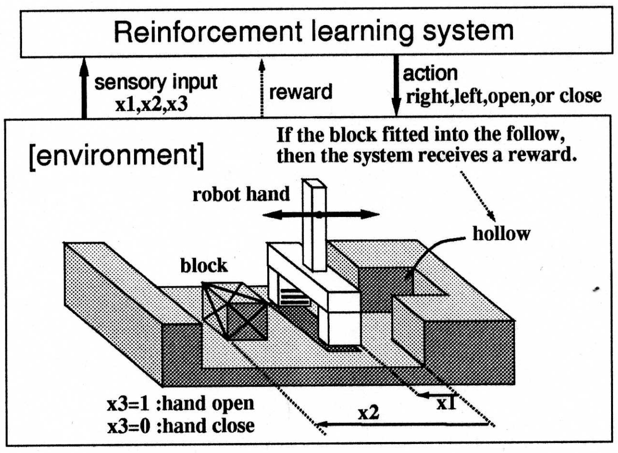
Reinforcement learning system
sensory input x1,x2,x3
reward
action right,left,open,or close
[environment]
If the block fitted into the follow, then the system receives a reward.
robot hand
hollow
block
x3=1 :hand open
x3=0 :hand close
x2
x1

Figure 5: The block storage problem

It is the problem which can be surely learned if a state analyzer can do that state space is even divided into the polygon territory properly because it is a structure that the only one effective action competes with the other ineffective ones except for a polygonal state. The network that can divide optional polygonal regions approximately was shown in the following was used as the state analyzer. The unit outputs 1 if it is fired,else it outputs 0. This unit was arranged in lattice of 27 toward the state input. The network of nearest-neighbor was made with the neighborhood of 6 units. It was compared with Q-learning system which has structure same as DSG method. Parameters are the same as the value used front section. The results are shown in Fig.6 . Fig.6 shows that they got the arrangement of block storage task in both techniques. But, diviations of the DSG system is smaller than Q-learning, and it is known that efficiency is excellent, too.

5 Conclusion

This paper presented a DSG based learning system as a reinforcement intensive approach. It has an incremental learning procedure, it can be applied to continuous state Markovian environments, and it is efficient because it doesn't accumulate information for inappropriate actions.

A proposed system was applied to some problems. Performance was compared with Q-learning system.

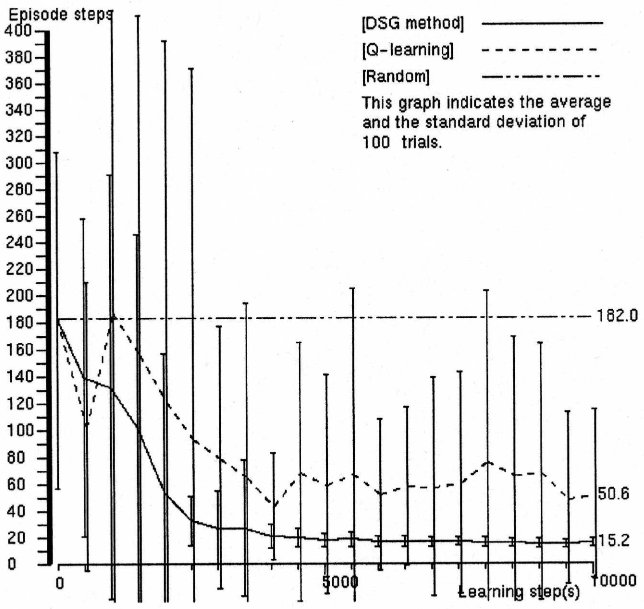

Figure 6: Performance on the block storage task.

The future work is to analyze theoretically the behavior of the proposed system.

References

[Goldberg 89] Goldberg,D.E.: *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley (1989).

[Grefenstette 88] Grefenstette,J.J.: *Credit Assignment in Rule Discovery Systems Based on Genetic Algorithms*, Machine Learning 3, pp.225-245 (1988).

[Holland 78] Holland, J. H., and Reightman. J. S.: *Cognitive Systems Based on Adaptive Algorithms*, Pattern-Directed Inference Systems. Waterman, D. A., and Hayes-Roth, F. ed., Academic Press (1987).

[Sutton 88] Sutton, R. S.: *Learning to Predict by the Methods of Temporal Differences*, Machine Learning 3, pp.9-44 (1988).

[Watkins 92] Watkins, C.J.C.H., and Dayan, P.: *Technical Note:Q-Learning*, Machine Learning 8, pp.55-68 (1992).

[Miyazaki 92] Miyazaki K.,Yamamura M.,Kobayashi S.: *A theory of Profit Sharing in Reinforcement Learning*, Journal of Japanese Society for Artificial Intelligence, Vol.9,No.4,1994 to appear.(in Japanese)

[Lin 93] Long-Ji Lin: *Scaling Up Reinforcement Learning for Robot Control*, Proc. of the tenth International Conference on Machine Learning, pp.182-189 (1993).