SICE Annual Conference in Fukui, August 4-6, 2003
Fukui University, Japan

Real time learning control of high d.o.f. robots: Automatic generation of discrete states and learning transition models

Hajime Kimura, Shigenobu Kobayashi
Interdisciplinary Graduate School of Science and Eng., Tokyo Institute of Technology

Abstract: We present a model-based RL approach to cope with continuous space of high D.O.F. robots, combining model learning and an actor-critic method. The model learner generates a discrete state-transition model, that helps improvement of both the policy and state-representation. In general, model-based methods tends to fail in Non-Markovian problems, but the proposed method, using actor-critic, can find good policies in such environments.

1 Introduction

We propose a new approach to construct control rules automatically for high-degree of freedom robots such as four-legged or humanoid robots. Learning problems in such robots have the following features:

1. States and actions are continuous and high-dimensional large spaces.
2. There exists uncertainty in the state transition.
3. The number of trial and error interactions is limited because of its cost, especially in real robots.

In the 3rd item (limitation of the trial), the learner would face to the following matters:

- It is desired to find allowable good control rules within the limited interactions.
- It should be able to improve control rules by gaining more experiences.
- Gaining new experiences is more expensive than storing all sequences in its memory.
- It should be able to give the expert's knowledge to the controller easily, and to revise the rules if the given rules include some errors.

State representation gives large influence to the learning speed and the quality of the control rules. It is remarkable in large and continuous state spaces, where simple partitioning continuous state-space into a multidimensional grid tends to fail because of curse of dimensionality. One promising approach to deal with high d.o.f. robots for state representation is StateNet[3]. It takes some representative poses as the states, and the actions correspond to moving the other poses. But the state representation must be given by designers in advance, and transition uncertainty is not considered in the model. There exists DP-based or model-based methods to find optimal control rules in uncertain environments, that is modeled by Markov decision processes. These methods need to construct environment models through trial and error interactions. This approach has several advantages that when the environment model is obtained, the controller can learn the other tasks easily. A problem is that the controller must construct a very accurate transition model in the model-based approach. However, the estimated model tends to inaccurate because of 1) The number of trial and error interactions are limited, 2) There exists uncertainty in state observations, especially in our robotics problem. Actually, control rules obtained from only DP-based planning in the learned model are useless in our experiments. Non-Markovian problems give too bad influence to DP-based approaches. It is caused from not only sensor noize but coarse discretization of state-space. To cope with this problem, a standard approach is to quantize the state spaces into smaller fine cells. But in high-dimensional space, it cause the curse of dimensionality. The other approach is to use adaptive grids, however, it is infeasible to construct any state representations that have no non-Markovian problems. That is because using only DP-based methods has limitations.

Recently, several direct approaches based on reinforcement learning are proposed, that do not have models. Q-learning and TD-methods are the representative methods. Although these are unstable in non-Markovian environments, sometimes it is used

PR0001/03/0000-3190 ¥400

because of simplicity of the scheme in such domains. Policy gradient methods such as actor-critic algorithms can construct desirable policies in a certain class of non-Markovian environments[5]. To cope with continuous state or action space, several methods are proposed, e.g., CMAC[6], an adaptive gridding method[1),7),9)]. Unfortunately, these methods would not learn sufficiently in the case of limited times of trials, because they requires enormous number of trials, Also, naive reinforcement learning approaches cannot make use of experiences obtained from executing another tasks.

In this paper we propose a model-based RL approach to cope with continuous space of high d.o.f. robots, combining model learning and an actor-critic method. The model learner constructs a discrete state-transition model based on StateNet, that helps improvement of both the policy and the state representation. Note that the feature of our method is to construct a rough transition model in coarse state representation that tends to cause non-Markovian, and the learner makes use of such an incomplete model for the following three items:

- The policy learning is accelerated by using state value estimated from the model.
- The learner can eliminate wasteful search by generating a good initial policy from the model and planning method.
- The learner can improve the state representation by detecting special states that have a large error in the model.

We apply the algorithm to an imaginary four-legged crawling robot, and demonstrate its dynamics. Also we show preliminary experiments applying to a real four-legged robot.

2 Problem Formulation

In this paper, we aim at obtaining control rules so that the robot shown in Fig.1 is to walk through trial and error within 15 minutes . But it is difficult to execute sufficient experiments using real robots for comparing several algorithms. We consider an imaginary four-legged crawling robot shown in Fig.2, and we evaluate the proposed methods from the experiments in that domain.

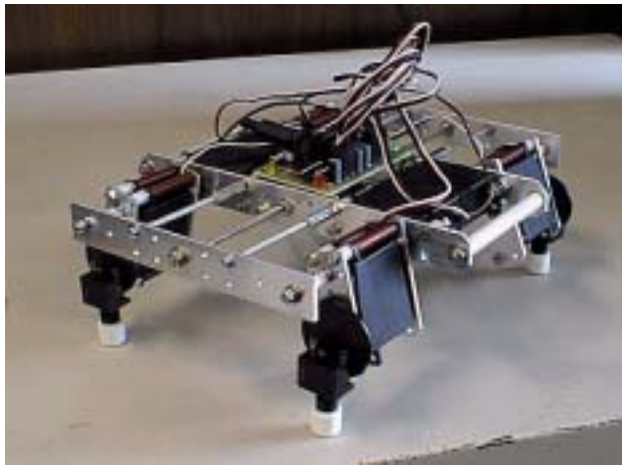

Figure 1: Real four-legged robot.

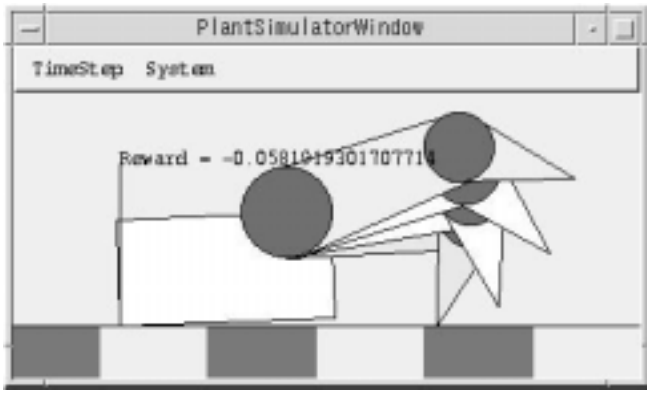

Figure 2: Imaginary four-legged crawling robot.

The objective of learning is to find control rules to move forward, but the controller does not know the dynamics ahead of time. The controller improves its behavior through a process of trial and error. At each time step, the agent observes current state, and select action, and immediate reward is given as a result of the action and state transition, and the time step proceeds to the next step. The reward signal reflects achievement of the given task. The designer must specify the task to learn as the reward representation. Since we want the robot to go forward, the immediate reward is defined as the speed of the body at each step.

These robots have bounded continuous and discrete state variables. Continuous state variables are angular-position of the eight joints, and discrete state variables represent four touch sensors for each legs. The learning agent observes these state variables. The action the agent selects is an objective angular-position of 8 joint-motors. That is,

the same dimension of the continuous state. When the agent select action, the robot moves the motors towards the commanded positions. When the joint-angles move to the commanded position, or changing the sensor variables, then the reward is given as the result of the transition, and the time step proceeds to the next step. When the case of sensor variable changing in the way of moving joint-motors, the angular-position would not correspond to the selected objective position. For this reason, there exists uncertainty of the state transition. The imaginary crawling robot has four legs, and each leg has two joints. The body moves forward or backward when some legs are touching the ground and moving it. Note that the controller should find good assignment of crawling control rules for each leg. The state space and action space are the same as the four-legged real robot. For simplicity, we assume that there is no noise in the state observation.

In real four-legged robot, two wheels detect a movement of its body, and generate the reward signal; the average of the moved distance of the wheels is the moved distance of the robot, and the differential of the wheels indicates the amount of turning the head. Since we want the robot to go straight, the immediate reward is defined as the average of the wheels minus absolute value of the differential of the wheels. Each foot has a touch sensor, and when the variable changes, an event of decision making would occur.

In this crawling robot domain, there are many control rules that can move the body ahead. However, in the domain of the four-legged robot, finding walking rules are difficult because there are many local optimal behavior such as "not moving" that can avoid negative rewards.

3 Combining Model Learning with Reinforcement Learning

We propose a new approach that is combined model learning and reinforcement learning as below.

3.1 State and Action Representation

The key points are:

1) State quantization by representative points: The representative points specify the divided regions, and the region is based on distance measure of L2 norm.

2) Generative discrete state representation: When the smallest distance between the observed state and representative points exceeds some threshold, the learner generates a new representative point at that location.

3) The actions are defined as objective states: The actions correspond to moving the other discrete states, but the resulting states are not always the same objective states. In this paper, the number of objective action states are fixed to avoid wasteful increasing of similar actions in the process of learning.

This state-action representation is similar to StateNet[3] that can cope with high-dimensional space by using small number of representative points. The new feature of our method is to generate new representative points. This type of state-action representation is easy to understand, since the behavior for the task is shown by some representative poses and transitions between those poses. Therefore, it is easy for giving prior knowledge, and also we can easily transform the knowledge by natural languages. The other advantage is that the edge of the state region can be easily improved by moving the position of its representative points. A drawback of this approach is that a local controller is needed to move robots towards the objective positions, since the action is corresponding to the objective states. The robots in this paper already have such a local controller only in the continuous state variables. But the local controller cannot treat properly in the context of discrete state variables (sensors), the learner is to control at the level of the StateNet-like representation.

3.2 Learning of Transition Models

The agent constructs environment models as a Markov decision process, translating the continuous state (and action) observation into the above discrete representation, and executing most-likelihood estimation by counting the visiting times and accumulating reward signals. However, the estimated transition model tends to inaccurate, because 1) State discretization is too coarse, 2) The amount of the trial data is not sufficient.

1. Observe state s_t, choose action a_t with probability $\pi(a_t|\theta, s_t)$, and perform it.

2. Observe immediate reward r_t, resulting state s_{t+1}, and calculate the TD-error according to

$$(\text{TD-error}) = r_t + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t) \ , \quad (1)$$

where $0 \leq \gamma \leq 1$ is the discount factor, $\hat{V}(s)$ is an estimated value function by the critic.

3. Update the estimating value function $\hat{V}(s)$ in the critic according to DP in the estimated model.

4. Update the actor's stochastic policy by

$$\begin{aligned} e_\pi(t) &= \frac{\partial}{\partial\theta} \ln\Big(\pi(a_t|\theta, s_t)\Big) \ , \quad (2) \\ \overline{e_\pi}(t) &\leftarrow e_\pi(t) + \overline{e_\pi}(t) \ , \\ \Delta\theta(t) &= (\text{TD-error})\, \overline{e_\pi}(t) \ , \\ \theta &\leftarrow \theta + \alpha_\pi \, \Delta\theta(t) \ , \end{aligned}$$

where e_π is the eligibility of the policy parameter θ, $\overline{e_\pi}$ is its trace, and α_π is a learning rate.

5. Discount the eligibility traces as follows:

$$\overline{e_\pi}(t+1) \leftarrow \gamma \, \lambda_\pi \, \overline{e_\pi}(t) \ , \quad (3)$$

where λ_π $(0 \leq \lambda_\pi \leq 1)$ is a discount factor in the actor.

6. Let $t \leftarrow t+1$, and go to step 1.

Figure 3: An actor-critic algorithm using eligibility traces in the actor, and the critic is making use of the model and DP.

3.3 Finding Control Rules to Achive Tasks

The actor-critic algorithm using eligibility traces in the actor[4] is a promising approach to deal with some class of non-Markovian environments. The critic estimates state values under the current policy, and the actor selects action improving its policy by using the estimated state values and reward signal. Note that if the critic fails to learn state values, the actor can improve the policy. In many cases, the critic plays a role of accelerating the actor's policy learning.

In this paper, we propose a new scheme in the critic replacing TD-method with model-learning combined with DP planning. It will be able to provide more accurate TD-errors to the critic for the purpose of accelerating the actor's learning. Since this approach remains the feature of the actor-critic, it can execute the model learning and the policy improvement concurrently. It can also obtain some degree of improved stochastic policy if the learning is terminated in arbitrary steps. When the agent is to learn a new policy, if the agent has the appropriate transition models, it can eliminate wasteful trial-and error interaction with the environment by generating a good initial policy from the model and the planning method. In the planning, estimated optimal action would be approximately correct excepts for the states that have observation uncertainty. Since the agent is to revise only such wrong actions by using reinforcement learning, it will be more efficient than learning from non-prior knowledge.

Fig.3 specifies the proposed method. The actor holds parameterized stochastic policy function, and updates the policy parameters towards the value gradient. Let $\pi(a|\theta, s)$ denote probability of selecting action a under the policy π in the state s, and θ is the policy parameters. The agent improves the policy π by modifying the parameter θ. The parameter λ_π specifies the actor's eligibility traces. When λ_π is close to 0, the policy would be updated according to the gradient of the estimated value function $\hat{V}$, and when λ_π is close to 1, the policy would be updated by the gradient of the actual return. When we want to cope with non-Markovian or incompleteness of the critic, $\lambda_\pi = 1$ is preferable.

In the critic, the state value $\hat{V}(s)$ under the policy π is estimated from the transition model and a DP scheme, given by:

1. For all s, substitute the following into $\hat{V}(s)$: $\sum_{s'} \sum_a \Pr(s'|s,a)\pi(a|\theta,s) \Big[R^a(s,s') + \gamma \hat{V}(s')\Big]$

2. Repeat the above scheme to converge.

Where the transition probability $Pr(s'|s,a)$ from the state s taking action a to the state s', and reward function $R^a(s,s')$ are given by the estimated model, and action selection probability $\pi(a|\theta,s)$ is given by the actor's policy.

3.4 Improvement of The State Representation

Moving the representative points to the center of gravity: In our method, the performance of

the model would vary in the different trials, because the position of the generating new representative point that depends on the corresponding current state input is quite accidental. Moving the representative points to the center of gravity may improve the performance.

State splitting using the likelihood of the model: In order to decrease the non-Markovian influence of coarse state discretizing, we propose a state splitting scheme using the likelihood of the model to the sequence data. When state discretizing is too coarse in a state region, the state transition probability would be considerably different in the same region. This fact lead us that if the likelihood of the model that is based on the splited region is considerably larger than the likelihood of the original model, we should split the state region. The similar state splitting method based on the likelihood is proposed using decision trees[9] . In their method, the state space is quantized by hyper rectangles, but our method is discretized in the way of Voronoi graph. The details of the state splitting algorithm is omitted because of the limited space.

4 Experiments

In order to confirm the effects of our model-based actor-critic and state splitting, we compare the other two algorithms: one is estimated optimal policy derived by DP in the estimated model, the other is a standard actor-critic[4]. We set the discount rate $\gamma = 0.9$, and the actor's learning coefficient of our method and the standard actor-critic are set to 0.02, and the critic's learning rate is 0.4. The discrete action representation is corresponds to the initial representative points for discrete state representation, given by: (1 1 1 1 1 1 1 1), (0 0 0 0 0 0 0 0), (1 1 1 1 0 0 0 0), (0 0 0 0 1 1 1 1), (1 1 0 0 1 1 0 0), (0 0 1 1 0 0 1 1), (1 0 1 0 1 0 1 0), (0 1 0 1 0 1 0 1). Fig.4 shows the performance of learned policy averaged over 5 trials. The distance threshold of the generating new state is 2.0, and all the learning methods use the same state representations containing 13 states. Fig.5 and 6 are the same configuration except for the number of states, 24 and 46 respectively. The policies obtained by only the model learning and DP planning hardly move to the front. The proposed method tends to find better policies than other methods as smaller the number of states.

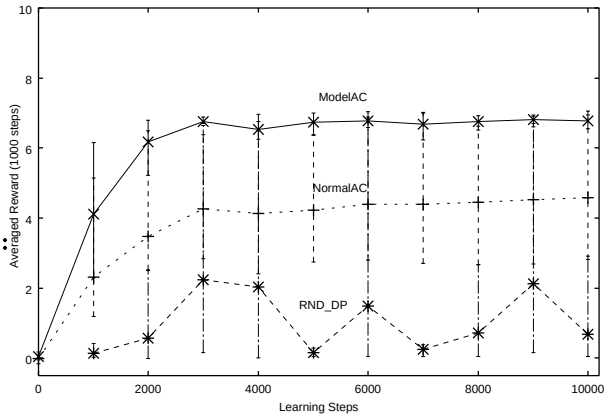

Figure 4: The performance of learned policy averaged over 5 trials in 13 states. The bars are upper or lower bound of the data. ModelAC is the proposed method, RND-DP is model-learning plus DP-planning with random search behavior.

Fig.7 shows the effect of improving state representation by the proposed splitting scheme. The effect is evaluated by the quality of learned policies in the model learning combined with DP-planning only method. The scheme of moving state representative points improves performance, then it finds good policies frequently. The more increase splitting, the agent finds good policies more stably. But in our method, the likelihood of the model is decreased when the splitting is increased.

We applied the proposed method to the real robot shown in Fig.1, and the learner finds good policies to walk within about 15 minutes.

5 Discussion

Insufficiency of visiting important states: On the learning of the transition models, arbitrary action selection policy can be taken. This feature often cause insufficient visiting to important states. For this reason, the effect of the proposed method becomes weak as increasing states. Although we omitted detailed experimental results, when the crawling robot learns moving backward from the model that is constructed by a random action selection, it is failed to find such control rules, because such a random policy would not provide useful experiences for learning to move backward.

Accidental state representation: The proposed method generates discrete state representations on

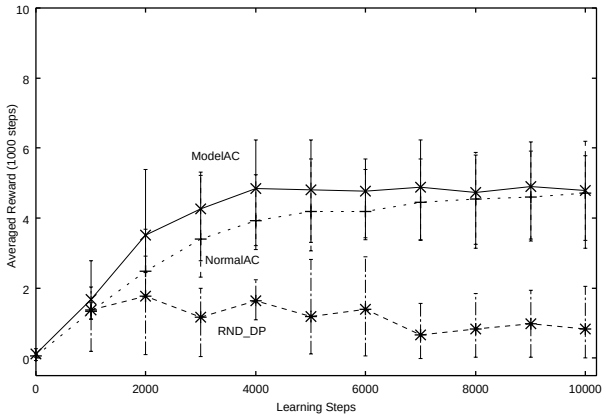

Figure 5: The performance of learned policy averaged over 5 trials in 24 states. ModelAC is the proposed method, RND-DP is model-learning plus DP-planning with random search behavior.

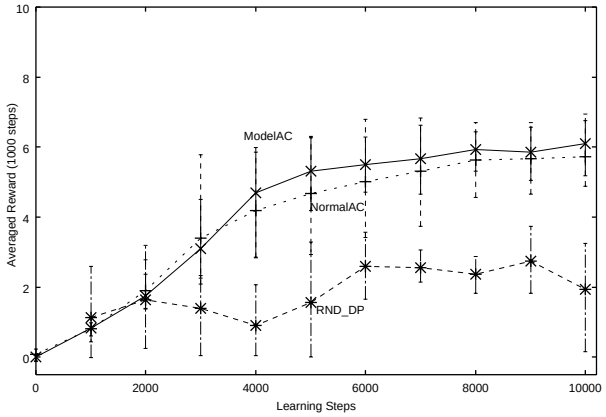

Figure 6: The performance of learned policy averaged over 5 trials in 46 states. ModelAC is the proposed method, RND-DP is model-learning plus DP-planning with random search behavior.

demand. As a result, the position of the generated new representative points are quite accidental, and the quality of the control rules depend on the position of the states. Finding stable generative state representation is a challenging future work.

Limitations of local search in the policy space: Since the proposed method is an extension of the actor-critic algorithm, It remains the same problem due to the local policy search algorithm. In order to avoid this problem, it may be necessary to employ a method holding several policies and improving them concurrently using techniques such as importance sampling.

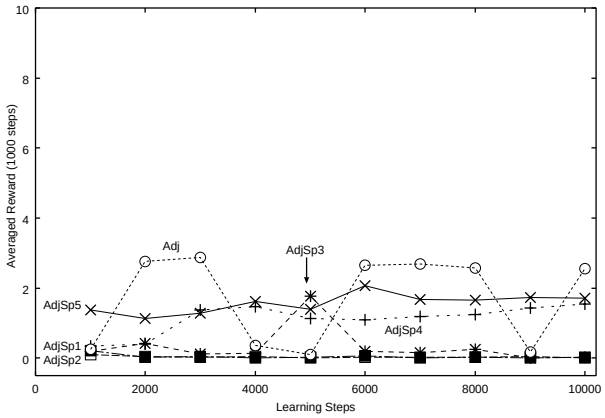

Figure 7: The effect of splitting states. Adj is moving representative state points to the center of gravity of the data, AdjSp1 is splitting one state from Adj, AdjSp2 is splitting one state from AdjSp1, and so on.

References

1) Asada, M., Noda, S. & Hosoda, K.: Action-Based State Space Construction for Robot Learning, Journal of the Robotics Society of Japan, Vol.15, No.6, pp.886–892, 1997 (in Japanese).

2) Barto, A. G., Bradtke, S. J. & Singh, S. P.: Learning to act using real-time dynamic programming, Artificial Intelligence 72, pp.81–138. (1995).

3) Kanehiro, F. Inaba, M. and Inoue, H.: StateNet: State Transition Graph Description of Action Space that Includes Error Recovery Function, Journal of the Robotics Society of Japan, Vol.20, No.8, pp.835–843, 2002 (in Japanese).

4) Kimura, H. & Kobayashi, S.: An Analysis of Actor/Critic Algorithms using Eligibility Traces: Reinforcement Learning with Imperfect Value Function, 15th International Conference on Machine Learning, pp.278–286 (1998).

5) Kimura, H. and Kobayasi, S.: Reinforcement learning of walking behavior for a four-legged robot, Trans.IEE of Japan, Vol.122-C, No.3, pp.330–337 (2002)

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

7) Takahashi, Y. and Asada, M.: Incremental State Space Segmentation for Behavior Learning by Real Robot, Journal of the Robotics Society of Japan, Vol.17, No.1, pp.118–124, 1999 (in Japanese).

8) Watkins, C.J.C.H. & Dayan, P.: Technical Note: Q-Learning, Machine Learning 8, pp.279–292 (1992).

9) Yairi, T. & Hori, K. & Nakasuka, S.: State Generalization Based on Maximum Likelihood Estimation Considering Multiple Behavior Outcomes, Journal of Japanese Society for Artificial Intelligence, Vol.16, No.1, pp.128–138 (2001 in Japanese).