Reinforcement Learning for Continuous Action using Stochastic Gradient Ascent

Hajime KIMURA, Shigenobu KOBAYASHI
Tokyo Institute of Technology, 4259 Nagatsuda, Midori-ku Yokohama 226-8502 JAPAN

Abstract: This paper considers a reinforcement learning (RL) where the set of possible action is continuous and reward is considerably delayed. The proposed method is based on a stochastic gradient ascent with respect to the policy parameter space; it does not require a model of the environment to be given or learned, it does not need to approximate the value function explicitly, and it is incremental, requiring only a constant amount of computation per step. We demonstrate the behavior through a simple linear regulator problem and a cart-pole control problem.

1 Introduction

This paper considers a reinforcement learning (RL) where the set of possible action is continuous and reward is considerably delayed. RL is the on-line learning of an input-output mapping through a process of trial and error to maximize some statistical performance index. *Q-learning* [14] is a representative of the RL algorithms for Markov decision processes in which the set of possible action is discrete. However, many applications in real-world need to handle the continuous action space, and often mixed it with discrete action space. This paper describes a new approach to RL for continuous action space. We define the agent's policy as a distribution of the action output, and we present a policy improvement algorithm. The proposed method is based on a stochastic gradient ascent with respect to the policy parameter space; it does not require a model of the environment to be given or learned, it does not need to approximate the value function explicitly, and it is incremental, requiring only a constant amount of computation per step. We demonstrate an application to linear regulator problems.

2 Related Works

[3] and [1] have proposed DP-based RL methods for only LQR problems. RFALCON [10] uses Adaptive Heuristic Critic [2] combined with a fuzzy controller. It is a policy improvement method which needs to estimate the value function explicitly.

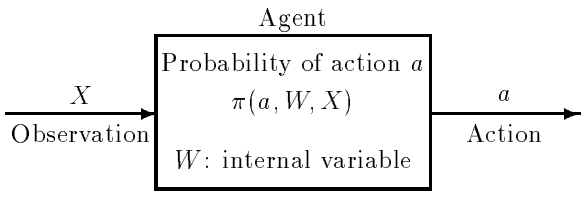

Figure 1: Stochastic policy; The agent can improve the policy π by modifying the parameter vector W.

3 Stochastic Gradient Ascent (SGA)

The objective of the agent is to form a *stochastic policy* [12], that assigns a probability distribution over actions to each observation, so that maximize some reward function. A policy $\pi(a, W, X)$ denotes probability of selecting action a in the observation X (Figure 1). The policy $\pi(a, W, X)$ is a probability density function when the set of possible action values a is continuous. The policy is represented by a parametric function approximator using the internal variable vector W. The agent can improve the policy π by modifying W. For example, W corresponds to synaptic weights where the action selecting probability is represented by neural networks, or W means weight of rules in classifier systems. The advantage of the parametric notation of π is that computational restriction and mechanisms of the agent can be specified simply by a form of the function, and we can provide a sound theory of learning algorithms for arbitrary types of agents.

1. Observe X_t in the environment.
2. Execute action a_t with probability $\pi(a_t, W, X_t)$.
3. Receive the immediate reward r_t.
4. Calculate $e_i(t)$ and $D_i(t)$ as

$$\begin{aligned} e_i(t) &= \frac{\partial}{\partial w_i} \ln\Big(\pi(a_t, W, X_t)\Big) \ , \\ D_i(t) &= e_i(t) + \gamma D_i(t-1) \ , \end{aligned}$$

   where γ $(0 \leq \gamma < 1)$ denotes the discount factor, and w_i does the i^{th} component of W.
5. Calculate $\Delta w_i(t)$ as

$$\Delta w_i(t) = (r_t - b)\, D_i(t) \ ,$$

   where b denotes the reinforcement baseline.
6. Policy Improvement: update W as

$$\begin{aligned} \Delta W(t) &= (\Delta w_1(t), \Delta w_2(t) \cdots \Delta w_i(t) \cdots) \ , \\ W &\leftarrow W + \alpha(1-\gamma)\, \Delta W(t) \ , \end{aligned}$$

   where α is a nonnegative learning rate factor.
7. Move to the time step $t+1$, and go to step 1.

Figure 2: General form of the SGA algorithm.

Figure 2 shows a general form of the algorithm. The notation $e_i(t)$ in the 4th procedure is called *eligibility* [15], that specifies a correlation between the associated policy parameter w_i and the executed action a_t. $D_i(t)$ is a discounted running average of eligibility. It accumulates the agent's history. When a positive reward is given, the agent updates W so that the probability of actions recorded in the history is increased.

Some theorems have shown in [7], [8] and [9] that the weight changes in the direction of the expected discounted reward biased by the state occupancy probability. Although any convergence theory of this algorithm have not shown, it has the following practical advantages.

- It is easy to implement multidimensional continuous action, that is often mixed with discrete action.
- Memory-less stochastic policies can be considerably better than memory-less deterministic policies in the case of partially observable Markov decision processes (POMDPs) [12] or multi-player games [11].
- It is easy to incorporate an expert's knowledge into the policy function by applying conventional supervised learning techniques.

Algorithms for Continuous Action

Remember that the policy $\pi(a, W, X)$ is a probability density function when the set of possible action values a is continuous. The normal distribution is a simple and well-known multiparameter distribution for a continuous random variable. It has two parameters, the mean μ and the standard deviation σ. When the policy function π is given by the equation 1, the eligibility of μ and σ are

$$\pi(a, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp(\frac{-(a-\mu)^2}{2\sigma^2}) \tag{1}$$

$$e_\mu = \frac{a_t - \mu}{\sigma^2} \tag{2}$$

$$e_\sigma = \frac{(a_t - \mu)^2 - \sigma^2}{\sigma^3} \ . \tag{3}$$

One useful feature of such a *Gaussian unit* [15] is that the agent has a potential to control it's degree of exploratory behavior. Because the parameter σ is occupying the denominators of equation 2 and 3, we must draw attention to the fact that the eligibility is to divergent when σ goes close to 0. The divergence of the eligibility has a bad influence on the algorithm. One way to overcome this problem is to control the step size of the update parameter vector using σ. Such an algorithm is obtained by setting the learning rate parameter proportional to σ^2, then the eligibility is given by

$$e_\mu = a_t - \mu \tag{4}$$

$$e_\sigma = \frac{(a_t - \mu)^2 - \sigma^2}{\sigma} \ . \tag{5}$$

4 Preliminary Experiment: A LQR Problem

The following linear quadratic regulator problem can serve as a benchmark. At a given discrete-time t, the state of the environment is the real value x_t. The agent chooses a control action a_t which is also real value. The dynamics of the environment is:

$$x_{t+1} = x_t + a_t + noise \ , \tag{6}$$

where $noise$ is the normal distribution that follows the standard deviation $\sigma_{noise} = 0.5$. The immediate reward is given by

$$r_t = -x_t^2 - a_t^2 \ . \tag{7}$$

The goal is to maximize the total discounted reward,

$$\sum_{t=0}^{\infty} \gamma^t r_t \ , \tag{8}$$

where γ is some discount factor $0 \leq \gamma < 1$. Because the task is a linear quadratic regulator (LQR) problem, it is possible to calculate the optimal control rule. From the Riccati equation, the optimal regulator is given by

$$a_t = -k_1 x_t \quad \text{, where} \quad k_1 = 1 - \frac{2}{1 + 2\gamma + \sqrt{4\gamma^2 + 1}}. \tag{9}$$

The optimum value function is given by $V^*(x_t) = -k_2 x_t^2$, where k_2 is a some positive constant. In this experiment, the possible state is constrained to lie in the range $[-4, 4]$. When the state transition given by Equation 6 does not result in the range $[-4, 4]$, the value of x_t is truncated. When the agent chooses an action which is not lie in the range $[-4, 4]$, the action executed in the environment is also truncated.

4.1 Implementation for SGA

The agent would first compute values of μ and σ deterministically and then draw its output from the normal distribution that follows mean equal to μ and standard deviation equal to σ. The agent has two internal variables, w_1 and w_2, and it computes the value of μ and σ according to

$$\mu = w_1 x_t \quad , \quad \sigma = \frac{1}{1 + \exp(-w_2)}. \tag{10}$$

Then, w_1 can be seen as a feedback gain parameter. The reason for the calculation of σ is to guarantee the value to keep positive. We write e_1, e_2 as the characteristic eligibility of w_1 and w_2 respectively. From Equation 4 and 5, e_1 and e_2 are given by

$$e_1 = e_\mu \frac{\partial}{\partial w_1} \mu = (a_t - \mu) x_t \tag{11}$$

$$e_2 = e_\sigma \frac{\partial}{\partial w_2} \sigma = ((a_t - \mu)^2 - \sigma^2)(1 - \sigma) \ . \tag{12}$$

The learning rate is fixed to $\alpha = 0.01$, reinforcement baseline $b = 0.0$, discount rate $\gamma = 0.9$. The value of w_1 is initialized to 0.35 ± 0.15, and $w_2 = 0$, i.e., $\sigma = 0.5$.

4.2 Implementation for Actor/Critic Algorithms

We compare the algorithm with an actor/critic algorithm. The critic quantizes the continuous state-space ($-4 \leq x \leq 4$) into an array of boxes. We have tried two types of the quantizing: one is discretizing x evenly into 3 boxes, the other is 10 boxes. The critic attempts to store in each box a prediction of the value $\hat{V}$ by using TD(0) [13]. The critic provides TD-error $= r_t + \gamma \hat{V}(x_{t+1}) - \hat{V}(x_t)$ to the actor as a reinforcement. The actor updates policy parameters with using Equation 11, 12 as:

$$w_1 = w_1 + \alpha \times (\text{TD-error}) \times e_1$$
$$w_2 = w_2 + \alpha \times (\text{TD-error}) \times e_2$$

The learning rate for TD(0) is fixed to 0.2, and all the other parameters are the same as the SGA method.

4.3 Results

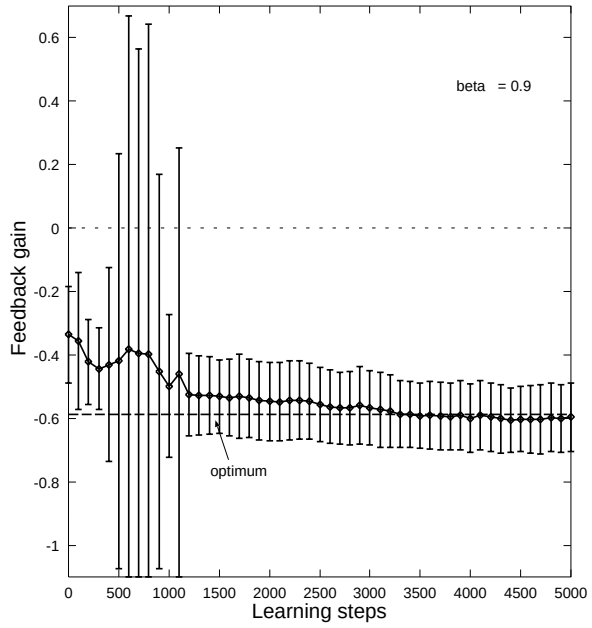

Figure 3: The average performance of the proposed method over 100 trials.

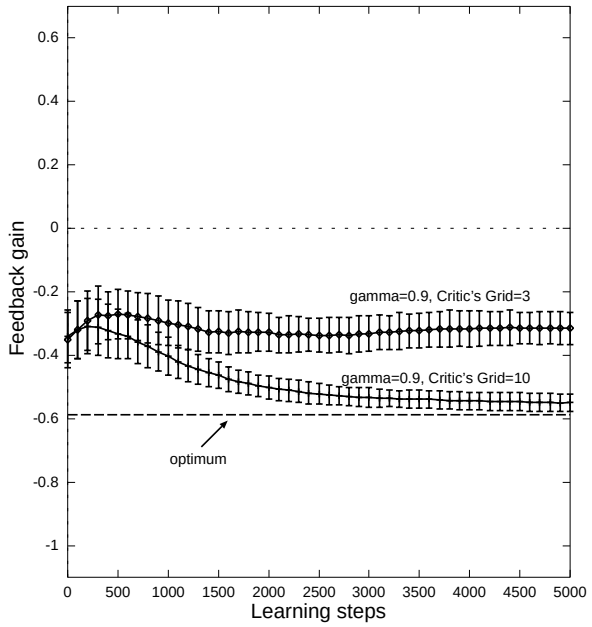

Figure 4: The average performance of the actor/critic algorithm over 100 trials. The critic uses 3 or 10 boxes.

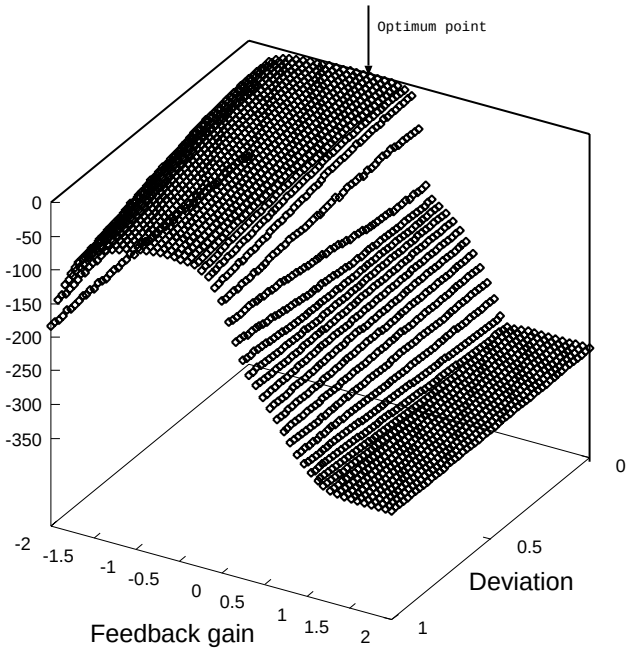

Figure 5: Value function over the parameter space in the LQR problem, where $\gamma = 0.9$. It is fairly flat around the optimum: $\mu = -0.5884$, $\sigma = 0$.

Figure 3 shows the performance of the SGA algorithm in the LQR problem. The variable of the feedback gain has a tendency to drift around the optimum. The parameter of σ decreased, but in most case, the growing stopped around 0.2. This result is not so pessimistic. Figure 5 shows the value function which are defined by Equation 7 and 8 over the parameter space (μ and σ). The value of performance is fairly flat around the optimal solution. For this reason, we can conclude that the proposed method would obtain a good policy for the LQR without estimating value function.

Figure 4 shows the performance of the actor/critic algorithms. The actor/critic algorithm using 3 boxes converged not close to the optimum feedback gain. The reason for this is that the critic's ability of the function approximation (3 boxes) is insufficient for learning policy, whereas the policy representation is the same.

5 Applying to a Cart-Pole Problem

The behavior of this algorithm is demonstrated through a computer simulation of a cart-pole control task, that is a multi-dimensional nonlinear nonquadratic problem. We modified the cart-pole problem described in [2] so that the action is taken to be continuous.

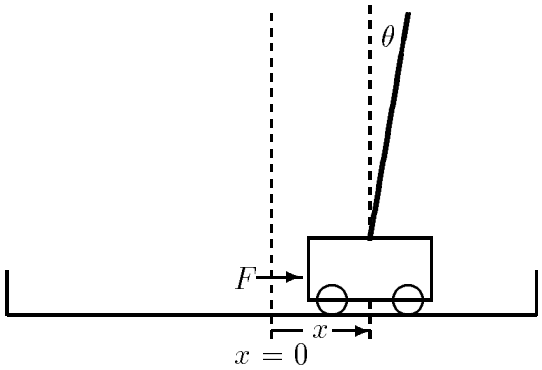

Figure 6: The cart-pole problem.

The dynamics of the cart-pole system is modeled by

$$\ddot{\theta} = \frac{g\sin\theta + \cos\theta\left(\frac{-F-m\ell\ddot{\theta}^2\sin\theta+\mu_c sgn(\dot{x})}{M+m}\right) - \frac{\mu_p\dot{\theta}}{m\ell}}{\ell\left(\frac{4}{3}-\frac{m\cos^2\theta}{M+m}\right)},$$

$$\ddot{x} = \frac{F + m\ell\left(\dot{\theta}^2\sin\theta - \ddot{\theta}\cos\theta\right) - \mu_c sgn(\dot{x})}{M+m},$$

where $M = 1.0$(kg) denotes mass of the cart, $m = 0.1$(kg) is mass of the pole, $2\ell = 1$(m) is a length of the pole, $g = 9.8(m/sec^2)$ is the acceleration of gravity, F(N) denotes the force applied to cart's center of mass, $\mu_c = 0.0005$ is a coefficient of friction of cart, $\mu_p = 0.000002$ is a coefficient of friction of pole. In this simulation, we use discrete-time system to approximate these equations, where $\Delta t = 0.02$sec. At each discrete time step, the agent observes $(x, \dot{x}, \theta, \dot{\theta})$, and controls the force F. The agent can execute action in arbitrary range, but the possible action in the cart-pole system is constrained to lie in the range $[-20, 20](N)$. When the agent chooses an action which is not lie in that range, the action executed in the system is truncated. The system begins with $(x, \dot{x}, \theta, \dot{\theta}) = (0, 0, 0, 0)$. The system fails and receives a reward (penalty) signal of -1 when the pole falls over ± 12 degrees or the cart runs over the bounds of its track $(-2.4 \leq x \leq 2.4)$, then the cart-pole system is reset to the initial state.

In this experiment, the state space is normalized as $(x, \dot{x}, \theta, \dot{\theta}) = (\pm 2.4 \text{ m}, \pm 2 \text{ m/sec}, \pm\pi \times 12/180 \text{ rad}, \pm 1.5 \text{ rad/sec})$ into $(\pm 0.5, \pm 0.5, \pm 0.5, \pm 0.5)$. The agent's policy function has five internal variables $w_1 \cdots w_5$, and computes the μ and σ according to

$$\begin{aligned} \mu &= w_1 \frac{x_t}{2.4} + w_2 \frac{\dot{x}_t}{2} + w_3 \frac{\theta_t}{12\pi/180} + w_4 \frac{\dot{\theta}_t}{1.5} \quad , \\ \sigma &= 0.1 + \frac{1}{1 + \exp(-w_5)}. \end{aligned} \tag{13}$$

The eligibilities $e_1 \cdots e_5$ are given by

$$\begin{aligned} e_1 &= (a_t - \mu)\, x_t \ , \quad e_2 = (a_t - \mu)\, \dot{x}_t \\ e_3 &= (a_t - \mu)\, \theta_t \ , \quad e_4 = (a_t - \mu)\, \dot{\theta}_t \\ e_5 &= ((a_t - \mu)^2 - \sigma^2)(1 + 0.1 - \sigma) \ . \end{aligned}$$

The critic discretizes the normalized state space evenly into $3 \times 3 \times 3 \times 3 = 81$ boxes, and attempts to store in each box $\hat{V}$ by using TD(0) algorithm [13]. Parameters are set to $\gamma = 0.95$, $\alpha = 0.01$, and learning rate for TD(0) in the actor/critic is 0.5.

Figure 7 shows the performance of two learning algorithms in which the policy representation is the same. The proposed algorithm achieved best results. In contrast, the actor/critic algorithm couldn't learn the control policy because of the poor ability of function approximation in the critic.

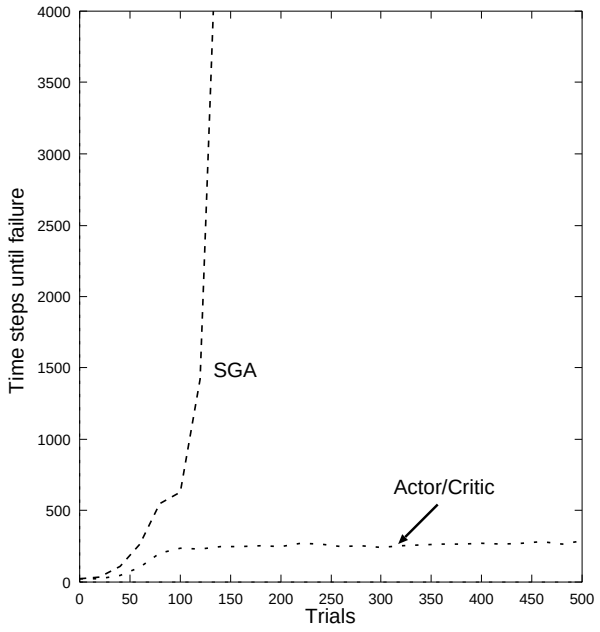

Figure 7: The average performance of the algorithms on 100 trials. The critic uses $3 \times 3 \times 3 \times 3$ boxes. A trial means an attempt from initial state to a failure.

6 Conclusion

This paper has considered a reinforcement learning where the set of possible action is continuous and reward is considerably delayed. We have proposed an policy improvement method that is based on a stochastic gradient ascent with respect to the policy

parameter space; it does not require a model of the environment to be given or learned, it does not need to approximate the value function explicitly, and it is incremental, requiring only a constant amount of computation per step. To our knowledge, this is the first study of the stochastic gradient method on discounted reward applying to RL tasks which have continuous action space. We have demonstrated the performance in comparison with an actor/critic algorithm. The proposed method enables to learn an acceptable policy with less cost rather than increasing the critic's ability of function approximation in our test cases.

References

[1] Baird, L. C.: Reinforcement Learning in Continuous Time: Advantage Updating, *Proceedings of IEEE International Conference on Neural Networks*, Vol. IV, pp. 2448-2453 (1994).

[2] Barto, A. G., Sutton, R. S. and Anderson, C. W.: Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems,

[3] Bradtke, S. J.: Reinforcement Learning Applied to Linear Quadratic Regulation, *Advances in Neural Information Processing Systems 5*, (1992).

[4] Clouse, J. A. & Utogoff, P. E.: A Teaching Method for Reinforcement Learning, *Proc. of the 9th International Conference on Machine Learning*, pp. 93-101 (1992).

[5] Crites, R. H. and Barto, A. G.: An Actor/Critic Algorithm that is Equivalent to Q-Learning, *Advances in Neural Information Processing Systems 7*, pp. 401-408 (1994).

[6] Doya, K. : Efficient Nonlinear Control with Actor-Tutor Architecture, *Advances in Neural Information Processing Systems 9*, pp. 1012–1018 (1996).

[7] Kimura, H., Yamamura, M., & Kobayashi, S.: Reinforcement Learning by Stochastic Hill Climbing on Discounted Reward, *Proceedings of the 12th International Conference on Machine Learning*, pp.295-303 (1995).

[8] Kimura, H. & Yamamura, M. & Kobayashi, S.: Reinforcement Learning in Partially Observable Markov Decision Processes: A Stochastic Gradient Method, *Journal of Japanese Society for Artificial Intelligence*, Vol.11, No.5, pp.761-768 (1996 in Japanese).

[9] Kimura, H., Miyazaki, K. and Kobayashi, S.: Reinforcement Learning in POMDPs with Function Approximation, *Proceedings of the 14th International Conference on Machine Learning*, pp. 152–160 (1997).

[10] Lin, C. J. and Lin, C. T.: Reinforcement Learning for An ART-Based Fuzzy Adaptive Learning Control Network, *IEEE Transactions on Neural Networks*, Vol.7, No. 3, pp. 709-731 (1996).

[11] Littman, M. L.: Markov games as a framework for multi-agent reinforcement learning, *Proc. of 11th International Conference on Machine Learning*, pp. 157-163 (1994).

[12] Singh, S. P., Jaakkola, T. and Jordan, M. I.: Learning Without State-Estimation in Partially Observable Markovian Decision Processes, *Proceedings of the 11th International Conference on Machine Learning*, pp. 284-292 (1994).

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

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

[15] Williams, R. J.: Simple Statistical Gradient Following Algorithms for Connectionist Reinforcement Learning, *Machine Learning 8*, pp. 229-256 (1992).