Learning non-parametric policies as random variable transformations

Hiroki Yokoyama[1] and Hiroyuki Okada[2]

***Abstract*— For reinforcement learning algorithms with non-parametric policies, we propose a method to learn random variable transformation whose resultant distribution maximizes a particular criterion. In our approach, the policy PDF is not explicitly represented, whereas the gradients of that can be directly estimated.**

I. INTRODUCTION

Learning how to act under uncertainty is a central problem in the field of machine learning. The main approach to attack this problem has been value-iteration, which forms the core of Q-learning[1] and other reinforcement learning algorithms. Value-iteration attempts to estimate a value function that indicates the value of taking an action in a state (or of being in a state) and use it to determine the actions. Action selection is done typically by maximizing the value function, which makes it difficult to deal with continuous action spaces. In contrast to the value-iteration, policy gradient approach (e.g. REINFORCE algorithm[2]) explicitly represents the policy and attempts to find the optimal one. This makes it straightforward to generate continuous action value as long as the policy is confined to a particular family of distributions, i.e. the policy is "parametric." To get rid of this limitation, there are many attempts (e.g. [3], [4]) to learn non-parametric policies. However, it is generally difficult to generate numbers from non-parametric distributions. That is why these approaches require to approximate the policies with more straightforward distributions (e.g. normal distribution[4]) or to restrict the action set to be finite after all (e.g.[3] with Boltzmann distribution).

In this paper, we propose a method to learn non-parametric policy without any concern about generating action values. Our approach is learning transformation from random variable whose distribution is known to one that follows desired distribution.

II. LEARNING RANDOM VARIABLE TRANSFORMATION

The most typical representation of stochastic policy is normal distribution whose parameters are functions of the state s_t, such as $a_t \sim N(\mu(s_t), \sigma^2(s_t))$, where a_t is the action value. In other words, the action value is determined by adding a noise $n_t \sim N(0,1)$ "after" calculating μ and σ^2 with particular function approximators, that is,

$$a_t = \mu(s_t) + \sigma(s_t) n_t. \tag{1}$$

[1]Hiroki Yokoyama is with Tamagawa University Brain Science Institute, Tamagawagakuen 6-1-1, Machida, Tokyo 194-8610, Japan yokoyama@lab.tamagawa.ac.jp

[2]Hiroyuki Okada is with the Faculty of Engineering, Tamagawa University, Tamagawagakuen 6-1-1, Machida, Tokyo 194-8610, Japan h.okada@tamagawa.ac.jp

Our idea is to feed a noise, whose distribution is known, as a part of input of a function approximator, namely,

$$a_t = f(s_t, n_t; \theta), \tag{2}$$

where θ is the parameter of the function approximator. We assume $n_t \sim N(0,1)$ for the sake of simplicity. In this case, the form of the distribution of a_t can vary depending on s_t and θ, at least non-Gaussian unless f is linear function. In particular, the distribution can be multimodal if the function approximator is sufficiently expressive. If there can be found f such that a_t follows desired distribution, random numbers of such distribution can be generated by calculating f.

The stochastic policy in our approach can be written as

$$\begin{aligned} p(a_t|s_t;\theta) &= \int p(a_t|n_t, s_t) p(n_t) \mathrm{d}n_t \\ &= \int \delta(a_t - f(s_t, n_t; \theta)) p(n_t) \mathrm{d}n_t. \end{aligned} \tag{3}$$

Since the calculation of $f(s_t, n_t; \theta)$ in this equation is deterministic, $p(a_t|n_t, s_t) = \delta(a_t - f(s_t, n_t; \theta))$, where $\delta(\cdot)$ is Dirac's delta function.

The goal of reinforcement learning is to maximize the expected reward $R(s,a) = E[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, a_t = a]$ ($0 < \gamma < 1$ is the discount factor) w.r.t. the stochastic policy. Thus, the objective function is

$$E = \int R(s_t, a_t) \int \delta(a_t - f(s_t, n_t; \theta)) p(n_t) \mathrm{d}n_t \mathrm{d}a_t. \tag{4}$$

Applying gradient method to the maximization thus requires differentiation of delta function of multivariate function. The next section deals with this kind of differential.

A. Directional derivative of Dirac's delta function

Here, we generalize the form of the objective function and discuss its derivative. It can be seen that the derivative of Eq.4 takes the following form:

$$\int_{\mathbb{R}^2} \delta'(\psi(x,y)) \phi(x,y) \mathrm{d}x \mathrm{d}y \tag{5}$$

(we use $\int$ to denote multiple integral). To deal with this form of expression, we apply generalized stokes' theorem: $\int_D \mathrm{d}w = \oint_{\partial D} w$ to a differential form $w = -v_y f g \mathrm{d}x + v_x f g \mathrm{d}y$, where f, g are scalar fields and $\boldsymbol{v} = (v_x, v_y)^\top$ is a vector field. This yields

$$\int_D (\nabla_{\boldsymbol{v}} f) g \mathrm{d}x \wedge \mathrm{d}y = \oint_{\partial D} w - \int_D (f \cdot (\nabla_{\boldsymbol{v}} g) + (\operatorname{div} \boldsymbol{v}) f g) \mathrm{d}x \wedge \mathrm{d}y, \tag{6}$$

which is a generalization of the partial integral and implies the possibility of eliminating the directional derivative of f. Substituting $\delta'(\psi)$ for f and ϕ for g in Eq.6 yields

$$\begin{aligned} &\int_D \delta'(\psi) \phi \, \mathrm{d}x \wedge \mathrm{d}y = \oint_{\partial D} \delta(\psi) \frac{\phi}{\nabla_{\boldsymbol{v}} \psi} (-v_y \mathrm{d}x + v_x \mathrm{d}y) \\ &- \int_D \delta(\psi) \left(\frac{\nabla_{\boldsymbol{v}} \psi \nabla_{\boldsymbol{v}} - \boldsymbol{v}^\top H_\psi \boldsymbol{v} - \nabla_{\nabla_{\boldsymbol{v}} \boldsymbol{v}} \psi}{(\nabla_{\boldsymbol{v}} \psi)^2} + \frac{\operatorname{div} \boldsymbol{v}}{\nabla_{\boldsymbol{v}} \psi} \right) \phi \, \mathrm{d}x \wedge \mathrm{d}y, \end{aligned} \tag{7}$$

where the last integral can be considered as a variant of Eq.5 with δ instead of δ', and the path integral as the residual.

B. Update rule

By substituting (a, n) for (x, y), $a - f(s, n; \theta)$ for ψ, $R(s, a) f_\theta(s, n; \theta) p(n)$ for ϕ, and employing, say, constant vector field $(v_a, v_n) = (0, 1)$, we obtain

$$\frac{\partial}{\partial \theta} E[r] \approx \lim_{D \to \mathbb{R}^2} \int_D \delta(a - f) \left(\frac{f_{nn} - f_n \frac{\partial}{\partial n}}{f_n^2} \right) (R f_\theta p(n)) \mathrm{d}a \mathrm{d}n$$

$$= E\left[R \frac{(f_{nn} - (\log p(n))' f_n) f_\theta - f_n f_{n\theta}}{f_n^2} \right], \tag{8}$$

where $f_n = \frac{\partial f}{\partial n}, f_{nn} = \frac{\partial^2 f}{\partial n^2}, f_{n\theta} = \frac{\partial^2 f}{\partial n \partial \theta}$. If $p(n)$ converges sufficiently fast as $n \to \pm\infty$, the residual

$$\oint_{\partial D} \delta(a - f) \frac{R f_\theta p(n)}{f_n} \mathrm{d}a \tag{9}$$

also converges as $D \to \mathbb{R}^2$. This suggests the efficacy of an update rule:

$$\theta \leftarrow \theta + \alpha_t r_t \frac{(f_{nn} - (\log p(n))' f_n) f_\theta - f_n f_{n\theta}}{f_n^2} \tag{10}$$

where α_t is the learning rate. If $n \sim N(0, 1)$, the derivative of the logarithm of noise PDF $(\log p(n))'$ is simply $-n$. Although the change of θ at each step is random, the expected reward increases through iteration.

C. Implementation with TensorFlow

Here, we show that the maximization of E can be easily implemented with TensorFlow, which is a software library for numerical computation that enables to deal with multi-dimensional data array and to deploy computation to GPUs with small codes.

Most gradient methods use derivatives of the objective function with respect only to trainable variables (θ in Eq.10), whereas our update rule contains derivative w.r.t. n as well. To deal with such derivatives, we obtain f_n and f_{nn} with `gradients` method and use `stop_gradient` method to prevent from further differentiation. A simple example of python code with TensorFlow (imported as `tf`) is as follows.

Listing 1. An example of implementation in python with TensorFlow.

```
a = f(s,n)
dfdn = tf.gradients(a, n)[0]
d2fdn2 = tf.gradients(dfdn, n)[0]
E = r*(a*tf.stop_gradient((d2fdn2+n*dfdn)/(dfdn**2))
    - dfdn/tf.stop_gradient(dfdn))
opt = tf.train.GradientDescentOptimizer(learning_rate)
train_step = opt.minimize(-E)
```

The variables `s`, `n` and `r` correspond to the state, Gaussian noise and the reward, respectively, and the function `f` represents a function approximator such as neural network. The training can be done by repeatedly running `train_step` operation prepared in the above code. The optimizer automatically calculates the derivative w.r.t. trainable variables to update them. In practice, it may be needed to clip the gradients and/or to decay the learning rate, which can also be easily implemented.

III. NUMERICAL EXPERIMENT

To confirm that multimodal distribution can be learned with proposed update rule, we trained a fully-connected four-layer neural network, whose input and output are n and a respectively. The second and third layer has 16 and 8 units respectively. We constrained the weight parameters to be non-negative, resulting in non-decreasing function, which is sufficient for one-dimensional random variable transformation. We employed $\tanh$ for the activation function.

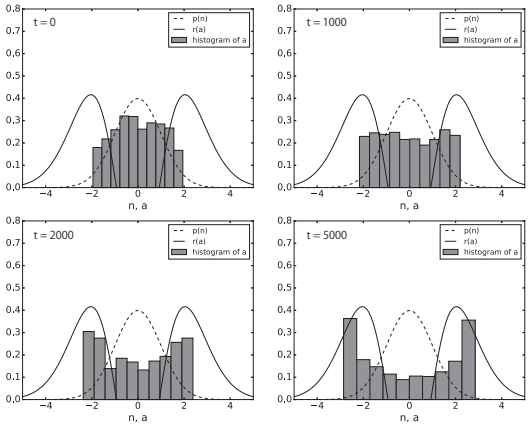

Fig. 1. Multimodal distribution obtained by the neural network.

Fig.1 shows the result. The input noise n was drawn from $N(0, 1)$, which is indicated by dashed curves (the horizontal axes are shared by n and a). The histograms show the distribution of $a = f(n)$. The solid curves show the reward $r = 0.7e^{-(a+1)^2/4} + 0.7e^{-(a-1)^2/4} - 1.5e^{-a^2/2}$.The frequency of a around zero decreased to avoid negative reward, whereas that around ± 2 increased pursuing the reward. If the policy were restricted to normal distribution, one of the possibilities would be chosen depending only on early steps of the training.

IV. CONCLUSIONS

In this paper, we confirmed that non-parametric policy can be learned in a simple toy problem without state. We are currently applying our update rule to neural networks with state input, to deal with control problems for future work.

Stochastic policy is a major factor in reinforcement learning, because it allows to explore unrevealed state of the environment. In addition, it is also important in social contexts, e.g., where the agent competes with others and its action is desired not to be predicted. It is expected that our framework enables robots to learn a variety of behaviors in such situations.

ACKNOWLEDGMENT

This work was supported by CREST, JST.

REFERENCES

[1] C. J. C. H. Watkins, "Learning from Delayed Rewards," Ph.D. dissertation, Cambridge Univ., Cambridge, England, 1989.

[2] R. J. Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". Machine Learning, vol. 8, no. 3, pp. 229–256, 1992.

[3] K. Kersting and K. Driessens, " Non-Parametric Policy Gradients: A Unified Treatment of Propositional and Relational Domains," in Pro. 25th Int. Conf. Machine Learning, Helsinki, Finland, 2008.

[4] H. van Hoof and J. Peters, "Learning of Non-Parametric Control Policies with High-Dimensional State Features," in Proc. 18th Int. Conf. Aritificial Intelligence and Statistics, San Diego, CA, USA, 2015.