Report for Reading Paper
A Generalized Path Integral Control Approach to Reinforcement Learning

Yunian Pan

Abstract—This Report presents some substances of the paper "A Generalized Path Integral Control Approach to Reinforcement Learning", as the paper stated: "reinforcement learning has recently moved towards combining classical techniques from optimal control and dynamic programming with modern learning techniques from statistical estimation theory in order to develop more scalable algorithms with higher efficiency and fewer open parameters, the paper suggests to use the framework of stochastic optimal control with path integrals calle PI^2 to do parameterized policy approximation, value function estimation and optimal control based on the stochastic Hamilton-Jacobi-Bellman (HJB) equations."

The algorithm's only parameter is the exploration noise, it can be conceived of as model-based, hybrid, or even model free, depending on how the learning task is structured. The update equations have no danger of numerical instabilities because neither matrix inversions nor gradient learning rates are required.

This report will briefly present the theoretical development of the original paper, then show the experimental set up and results, some Discussion required by the assignments will be given.

Index Terms—stochastic optimal control, reinforcement learning, parameterized policies, path integrals.

I. Summary

A. Brief theoretical development

The theoretical development can be divided into 4 parts, the first part is how we can start to derive stochastic optimal control algorithm from Hamiltan-Jacobian-Bellman equation(HJB), the optimal control of a general stochastic control framework can be derived from following nonlinear PDE,

$$-\partial_t V_t = q_t + (\nabla_x V_t)^\top f_t - \frac{1}{2}(\nabla_x V_t)^\top G_t R d^{-1} G_t^\top (\nabla_{x_t} V_t) + \frac{1}{2} trace((\nabla_{xx} V_t) G_t \Sigma_\epsilon G_t^\top) \tag{1}$$

The second part is to make a logarithmic transfrom of this nonlinear PDE so that we get a linear PDE, here we make a very some strong assumptions that:

$$\lambda G_t R^{-1} G_t^\top = G_t \Sigma_\epsilon G_t^\top = \Sigma(x_t) \tag{2}$$

This assumption implies that the cost of the control is inverse proportional to the noise variance, and using Feyman-Kac lemma we can solve this PDE as a path integral:

$$\Psi_{t_i} = \lim_{dt \to 0} \int p(\tau_i | x_{t_i}) \exp[-\frac{1}{\lambda}(\phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} dt)] d\tau_i \tag{3}$$

The third part is to do some formulation of the MDP process and building the probability of the trajectory through Gaussian noise distribution, since the only stochastic part is the noise, we would get an explicit expression of the trajectory cost, and through the expression we can derive the optimal stochastic control which can be viewed as the probability combination of local optimal control.(This actually results in stochastic optimal control algorithm, but people don't like it since it's too complicated!)

The forth part is to parametrize the policy, and start from a simple DMP case,(which will be discarded later to prove this algorithm can be model free) we get the parameter updating formula:

$$\begin{aligned} \theta^{(new)} &= \frac{1}{N} \sum_{i=0}^{N-1} \delta\theta_{t_i} + \frac{1}{N} \sum_{i=0}^{N-1} \theta \\ &= \frac{1}{N} \sum_{i=0}^{N-1} \delta\theta_{t_i} + \theta \end{aligned} \tag{4}$$

Thus the main framework of Policy Improvement with Path Integrals (PI^2) has been constructed, with the equations below:

$$\begin{aligned} P(\tau_i) &= \frac{e^{-\frac{1}{\lambda} S(\tau_i)}}{\int e^{-\frac{1}{\lambda} S(\tau_i)} d\tau_i} \\ S(\tau_i) &= \phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} \\ &\quad + \frac{1}{2} \sum_{j=i}^{N-1} \frac{1}{2} (\theta_{t_j} + \epsilon_{t_j})^\top R^{-1} (\theta_{t_j} + \epsilon_{t_j}) \\ \delta\theta_{t_i} &= \int P(\tau_i) M_{t_i} \epsilon_{t_i} d\tau_i \\ [\delta\theta]_j &= \frac{\sum_{i=0}^{N-1} (N-i) w_{j,t_i} [\delta\theta_{t_i}]_j}{\sum_{i=0}^{N-1} (N-i) w_{j,t_i}} \\ \theta^{(new)} &= \theta^{(old)} + \delta\theta \end{aligned}$$

TABLE I

- Given:
  - The immediate cost: $r_t = r(x_t, u_t, t) = q_t + \frac{1}{2}\theta_t^\top R\theta_t$
  - A terminal cost term Φ_{t_N}
  - A stochastic parameterized policy $\alpha_{t_i} = g_{t_i}^\top(\theta + \epsilon_{t_i})$
  - The basis function g_{t_i} from the system dynamics
  - The variance Σ_ϵ of the mean zero noise ϵ_t
  - The initial parameter vector θ
- Repeat until convergence of the tracjectory cost S
  - Create K roll-outs of the system from the same start state x_0 using stochstic parameters $\theta + \epsilon_t$ at every time step
  - For $k = 1, \ldots, K$, compute:
    - $P(\tau_{i,k}) = \frac{e^{-\frac{1}{\lambda}S(\tau_{i,k})}}{\sum_{k=1}^K [e^{-\frac{1}{\lambda}S(\tau_{i,k})}]}$
    - $S(\tau_{i,k}) = \phi_{t_N,k} + \sum_{j=i}^{N-1} q_{t_j,k} + \frac{1}{2}\sum_{j=i}^{N-1}\frac{1}{2}(\theta + M_{t_j,k}\epsilon_{t_j,k})^\top R^{-1}(\theta + M_{t_j,k}\epsilon_{t_j,k})$
    - $M_{t_j,k} = \frac{R^{-1}g_{t_j,k}g_{t_j,k}^\top}{g_{t_j,k}^\top R^{-1} g_{t_j,k}}$
  - Probability of a trajectory: $P(\tau_i) = \frac{e^{-\frac{1}{\lambda}\tilde{S}(\tau_i)}}{\int e^{-\frac{1}{\lambda}\tilde{S}(\tau_i)} d\tau_i}$
  - Generalized trajectory cost: $\tilde{S}(\tau_i) = S(\tau_i) + \frac{\lambda}{2}\sum_{j=i}^{N-1}\log|H_{t_j}|$
  - Local controls: $u_L(\tau_i) = R^{-1}G_{t_i}^{(c)\top}(G_{t_i}^{(c)}R^{-1}G_{t_i}^{(c)\top})^{-1}(G_{t_i}^{(c)}\epsilon_{t_i} - b_{t_i})$
- For $i = 1, \ldots, (N-1)$,compute:
  - $\delta\theta_{t_i} = \sum_{k=0}^K [P(\tau_{i,k}) M_{t_i,k}\epsilon_{t_i,k}]$
- Compute: $[\delta\theta]_j = \frac{\sum_{i=0}^{N-1}(N-i)w_{j,t_i}[\delta\theta_{t_i}]_j}{\sum_{i=0}^{N-1}(N-i)w_{j,t_i}}$
- Update: $\theta^{(new)} = \theta^{(old)} + \delta\theta$
- Create one noiseless roll-out to check the trajectory cost $R = \Phi_{t_N} + \sum_{i=0}^{N-1} r_{t_i}$. In case the noise cannot be turned off, that is, a stochastic system, multiple roll-outs need be averaged.

It's remarkable that in the derivation we actually discard another projection matrix term in order to get time invariant policy!

B. How was it done?

Analogous to the generalized formulation of stochastic optimal control with path integrals, the algorithm was derived using the constructed equations, I describes the algorithm.

C. Why is it worth doing?

Cited from the original paper: It's simple formed, numerically robust in high dimensional problems, with no open algorithmic tuning parameters except exploration noise. What's more, it makes interesting connection to previous work on RL based on probability matching and motivates why probability matching algorithms can be successful. In general, Policy Improvement with Path Integrals (PI^2) offers currently one of the most efficient, numerically robust, and easy to implement algorithms for RL based on trajectory roll-outs.

D. results

There were 3 learning tasks done in the paper:

1) 1 DOF Via-Point Task: Forcing the movement to pass through an intermediate via-point at a particular time, which is an abstract approximation of hitting a target. The cost function was:

$$r_{300ms} = 100000000(G - y_{t_{300ms}})^2$$
$$\phi_{t_N} = 0$$

with $G = 0.25$. The results shown in 1 indicates that PI^2 performs the best:

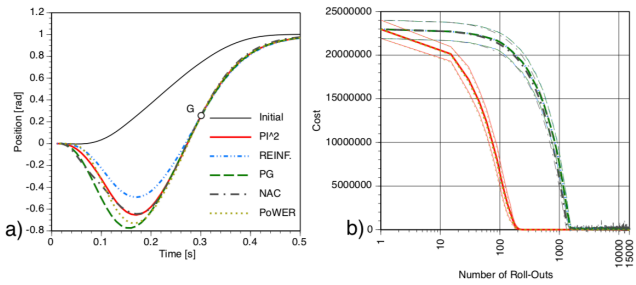


Fig. 1. 1 DOF via-point comparison

2) Multi-DOF Via-Point Task: The learning task was to pass through an intermediate target, just that a d = 2, 10, 50 dimensional motor primitive was employed thus result in a redundant learning problem. Denote the joint angles of the robots as ξ_i, $i = 1, \ldots, d$, s.t. the directly actuated part is: $\ddot{\xi}_{i,t} = f_{i,t} + g_{i,t}^\top(\theta_i + \epsilon_{i,t})$, The end-effector position is computed as:

$$x_t = \frac{1}{d}\sum_{i=1}^d \cos(\sum_{j=1}^i \xi_{j,t})$$
$$y_t = \frac{1}{d}\sum_{i=1}^d \sin(\sum_{j=1}^i \xi_{j,t})$$

The cost function was designed as:

$$r_t = \frac{\sum_{i=1}^d (d+1-i)(0.1 f_{i,t}^2 + 0.5\theta_i^\top\theta_i)}{\sum_{i=1}^d (d+1-i)}$$
$$\Delta r_{300ms} = 10^8((0.5 - x_{t_{300ms}})^2 + (0.5 - y_{t_{300ms}})^2)$$
$$\phi_{t_N} = 0$$

The comparison shown in 2 indicates that PI^2 is superior to others.

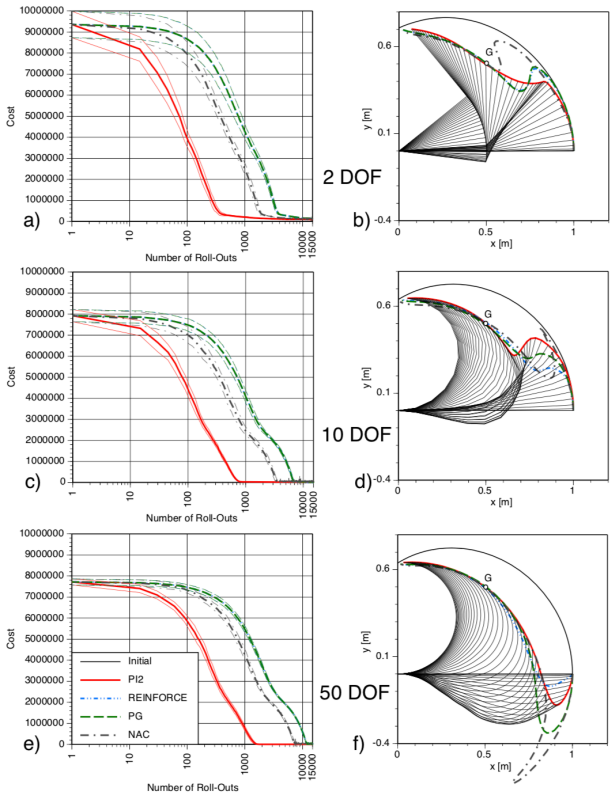


Fig. 2. multi-DOF via-point comparison

3) Dog jumping: The robot dog is to jump across as gap. The paper only uses a simulation to show the results as real robot was not available. The dog has 3 DOFs per leg, thus a total of d=12, each DOF represented with 50 basis functions.

PI^2 learning used primarily the forward progress as a reward, a penalty was incurred if the yaw or the roll exceeded a threshold value, which encouraged the robot to jump straight across the gap instead of to the side or falling over. The exact cost design is:

$$r_t = r_{roll} + r_{yaw} + \sum_{i=1}^{d}(a_1 f_{i,t}^2 + 0.5 a_2 \theta_i^\top \theta_i)$$

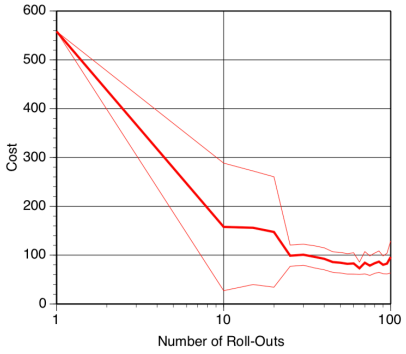
$$r_{roll} = \begin{cases} 100(|roll_t| - 0.3)^2 & if |roll_t| > 3 \\ 0 & otherwise \end{cases}$$

$$r_{yaw} = \begin{cases} 100(|yaw_t| - 0.1)^2 & if |yaw_t| > 0.1 \\ 0 & otherwise \end{cases}$$

$$\phi_{t_N} = 5 \times 10^4 (goal - x_{nose})^2,$$

Roll and yaw are the roll and yaw angles of the robot's body, and x_{nose} is the position of the "nose". The multipliers for each reward component were tuned to have a balanced influence of all terms. Ten learning trials were performed initially for the first parameter update.

It should be noted that here manual tuning is focused on generating a good cost function, which is beyond the scope of reinforcement learning.



(b) Learning curve for Dog Jump with **PI**^{2} $\pm 1 std$

Fig. 3. learning curve

II. How the paper relates the algorithms seen in class?

This algorithm has been presented in our class, as we are entering the era of DQN, we learned that we can use a parametrized function to approximate the best policy, in the context of this idea and after some beautiful mathematical development and derivations, Theodorou et. came up with this idea which can be model-based semi-model based and even model-free, thus give the readers infinite possibility to solve many reinforcement learning tasks. What's more, the algorithm, absorbing some nice property of Monte-Carlo sampling, provides a very mathematically solid ground.

III. Discussion

As can be seen from the previous figures, the PI^2 Algorithm has a surprisingly good performance, without any need for tuning parameters. And it's scalable to high dimensionality. Yet there's indeed space to improve such as the cost function design, which future work might be focus on to move the algorithm to a "black box" character.

The real beauty in the development is that through making a simple assumption but gigantically reduce the complexity of the PDE equation, future work might also fall into the effort of removing this assumption or change it into more generalized version.

References

[1] Theodorou, Evangelos, Jonas Buchli, and Stefan Schaal. "A generalized path integral control approach to reinforcement learning." journal of machine learning research 11.Nov (2010): 3137-3181.