Reinforcement Learning of Motor Skills with Policy Gradients

Ruiming Zhang,

{**amber.zhang,** }**@tum.de**

Problem Setting

- Task: Learning complex motor skills with an anthropomorphic robot arm using reinforcement learning.
- Typical characteristics:
  - high-dimensional and continuous space and action space
  - has to be model-free
  - high degrees of freedom cannot deal with parameterised policies

Problem statement in Mathematics

- The general goal is to optimize the policy parameters $\theta \in R^K$ so that the expect return

$$J(\theta) = \frac{1}{a_\Sigma} E\{\sum_{k=0}^{H} a_k r_k\} \tag{1}$$

is optimized where a_k denote time-step dependent weighting factors and r_k the reward the learning system received by each instant of time.

Policy Gradient Method

- The policy parameter θ is updated at each time step by:

$$\theta_{m+1} = \theta_m + \alpha_m \nabla_\theta J|_{\theta=\theta_m} \tag{2}$$

where $\alpha_m \in R^+$ denotes a learning rate and fulfill:

$$\sum_{m=0}^{\infty} \alpha_m > 0 \quad and \quad \sum_{m=0}^{\infty} \alpha_m^2 = 0 \tag{3}$$

- The main problem is obtaining a good estimator of the policy gradient $\nabla_\theta J|_\theta = \theta_m$.

Motor Primitive Framework

- We use motor commands to well represent a trajectory, which forms the stochastic policy:

$$\pi(\ddot{q}_d|q_d) = \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{(\ddot{\ddot{q}}_d - \ddot{q}_d)^2}{2\sigma^2}) \tag{4}$$

The $\ddot{\ddot{q}}_d$ term denotes for a perturbed target output considering the exploration controlled only through σ.

Likelihood Ratio Method

- Use τ to represent a real generated trajectory, $\tau \sim p_\theta(\tau) = p(\tau|\theta)$, with rewards $r(\tau) = \sum_{k=0}^{H} a_k r_k$.
- LEMMA 1 *save for use.....*
- Then the expect return of a policy can be written as an expectation over all possible trajectories:

$$J(\theta) = \int p_\theta(\tau) r(\tau) d\tau \tag{5}$$

- Subsequently, the gradient can be rewritted by:

$$\begin{aligned} \nabla_\theta J(\theta) &= \int \nabla_\theta p_\theta(\tau) r(\tau) d\tau \\ &= \int p_\theta(\tau) \nabla_\theta log p_\theta(\tau) r(\tau) d(\tau) \\ &= E\{\nabla_\theta log p_\theta(\tau) r(\tau)\} \end{aligned} \tag{6}$$

- The derivative $\nabla_\theta log p_\theta(\tau)$ can be computed by:

$$\nabla_\theta log p_\theta(\tau) = \sum_{k=0}^{H} \nabla_\theta log \pi_\theta(u_k|x_k) \tag{7}$$

- A constant baseline can be inserted since it has no effect to the derivative, and results in the final form of the gradient estimator:

$$\nabla_\theta J(\theta) = E\{\nabla_\theta log p_\theta(\tau)(r(\tau) - b)\} \tag{8}$$

Usually b is chosen with the goal to minimize the variance of the gradient estimator.

Natural Actor-Critic

- We first introduce second-order Taylor expansion to approximate the closeness of two distribution, i.e. the gradient descent with respect to an invariant measure of the closeness between the current policy and the updated policy based on the distribution of the paths generated by each of these:

$$d_{KL}(p_\theta(\tau) \;||\; p_{\theta+\Delta_\theta}(\tau)) \approx \frac{1}{2} \Delta\theta^T F_\theta \Delta\theta \tag{9}$$

where

$$F_\theta = \int p_\theta(\tau) \nabla log p_\theta(\tau) \nabla log p_\theta(\tau)^T d\tau = \left\langle \nabla log p_\theta(\tau) \nabla log p_\theta(\tau)^T \right\rangle \tag{10}$$

is known as the Fisher information matrix.

- Assume that the amount of change is fixed using step size ε. Then we have an Optimization problem:

$$\begin{aligned} &\max_{\Delta\theta} J(\theta + \Delta\theta) \approx J(\theta) + \Delta\theta^T \nabla_\theta J \\ &s.t.\ \varepsilon = d_{KL}(p_\theta(\tau) \;||\; p_{\theta+\Delta\theta}(\tau) \approx \tfrac{1}{2} \Delta\theta^T F_\theta \Delta\theta) \end{aligned} \tag{11}$$

and has the solution

$$\Delta\theta = \alpha_n F_\theta^{-1} \nabla_\theta J \tag{12}$$

- The direction $\Delta\theta$ is called the natural gradient.

Compatible Function Approximation

- to be added
- to be added

Natural Actor-Critic Algorithm

- For episodic tasks the natural actor-critic can be simpified as:

$$\sum_{t=0}^{H} a_t \nabla log \pi(u_t, x_t)^T w + J_0 = \sum_{t=0}^{H} a_t r(x_t, u_t) \tag{13}$$

- This means for non-stochastic tasks we can obtain a natural gradient after $\dim\theta + 1$ roll-outs using least-squares regression:

$$\begin{bmatrix} \omega \\ J_0 \end{bmatrix} = (\Psi^T \Psi)^{-1} \Psi^T R \tag{14}$$

with

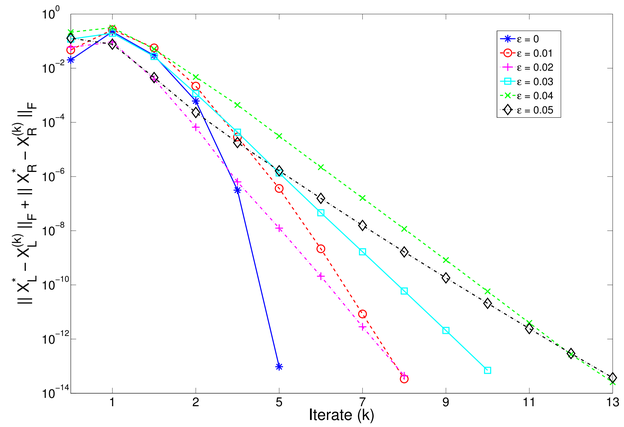
$$\Psi_i = \left[\textstyle\sum_{t=0}^{H} a_t \nabla log \pi(u_t, x_t)^T, 1\right] \tag{15}$$

$$R_i = \textstyle\sum_{t=0}^{H} a_t r(x_t, u_t) \tag{16}$$

- In order to take time-variance rewards significantly better into account,
- to be added...

A Block-Jacobi NsJD Algorithm

Numerical Experiments



Research Group for Geometric Optimization and Machine Learning
Technische Universität München, Germany