Deep Reinfocement Learning For Robotics Manipulation (I) (Under Construction)

Harry Li [‡], Ph.D.
Computer Engineering Department, San Jose State University
San Jose, CA 95192, USA

Email[†]: harry.li@ctione.com

***Abstract*—This note describes the mathematic foundation of deep reinfocement learning for robotics system control. The objectives of deep reinforcement learning is to develop algorithms to make robot has the ability to learn new manipulation policies from scratch, without user demonstrations and without the need of a task-specific domain knowledge.**

I. Introduction

This note is prepared based on the review of the paper on reinforcement learning for robotics applications [Franceschetti, 2020]. The scope of this review includes

1. Trust Region Policy Optimization;
2. DeepQ-Network with Normalized Advantage Functions;
3. Deep Deterministic Policy Gradient;
4. Vanilla Policy Gradient.

The techniques have been applied in the reviewed paper to the robotic arms control to accomplish manipulation tasks such as

1. reaching a random target pose;
2. pick & place an object.

Both simulated and real-world experiments can be carried out to verify the review paper's findings.

We would like to gain good insights in the areas of

1. the procedures configuration to precisely estimate the algorithms hyperparameters, which can be carried out in Unity and/or ROS integrated environment;
2. the policies design;

Through the review of this paper, we will carry out the real-world experiments in our lab environment on FD100 robot to demonstrate our designed polices and smooth transation to a real world environment for fast readily deployable results.

II. Mathematic Formulation

Robotic operations can be characterized as a robot making sequential movement, e.g., control actions in a stochastic environment. The control actions are torques from the controllers to robot joints over a sequence of time steps. From the deep reinforcement learning point of view, the goal of robot operations is to maximize a long term reward. By the mathematical nature, these sequential decision process is modeled as a Markov Decision Process (MDP).

A large class of sequential decision making problems can be formuated as Markov decision processes (MDPs). An MDP consists of

1. a set S of states, denoted as S = $\{s_1, s_2, ..., s_N\}$, and
2. a set A of actions, denoted as A = $\{a_1, a_2, ..., a_M\}$ as well as
3. a transition function T and
4. a reward function R,

denoted as a tuple $< S, A, T, R >$. When in any state $s \in S$, an action $a \in A$ will lead to a new state with a transition probability $P_T(s, a, s\prime)$, and a reward R(s, a) function.

The stochastic policy $\pi : S \rightarrow D$ maps from a space state to a probability over the set of actions, and $\pi(a|s)$ represents the probability of choosing action a at state s.

The goal is to find the optimal policy π^* to produce the highest rewards [Rein, 2020]:

$$\arg\max_{\pi \in \Omega_\pi} \{E[\sum_{k=0}^{H-1} \gamma^k R(s_k, a_k)]\} \tag{1}$$

The composition of the above equation can be described as follows. First, define reward function at time k:

$$R(s_k, a_k) \tag{2}$$

with discounted factor γ at time k, and $\gamma < 1$ (to formulate the older state, as the k becomes bigger, its effects on the reward will get smaller):

$$\gamma^k R(s_k, a_k) \tag{3}$$

Note In finite-horizon or goal-oriented domains, choose discount factors close to 1 to encourage actions towards the goal, whereas in infinite-horizon domains choose lower discount factors to achieve a balance between short- and long- term rewards.

For all the experiments up to horizon H, we have summation:

$$\sum_{k=0}^{H-1} \gamma^k R(s_k, a_k) \tag{4}$$

For the stochastic nature of these rewards, we use statistical expectation as

$$E[\sum_{k=0}^{H-1} \gamma^k R(s_k, a_k)] = \int_{\inf} \gamma^k R(s_k, a_k) P(s_k, a_k) ds \tag{5}$$

Hence, we have the average discounted rewards under policy π.

III. DRL Connection to Robot Control

Definition 1. *Trajectory τ. A trajectory τ of a robot motion is defined as a sequence of state-action pairs in time sequence $t_1, t_2, ..., t_N$, denoted as $\tau = (s_1, a_1, s_2, a_2, ..., s_N, a_N)$.*

Consider any trajectory τ of a robot motion, e.g., $\tau = (s_1, a_1, s_2, a_2, ..., s_N, a_N)$, the trajectory of the end effector in 3D space. See an end effector (a set of three in this case) from CTI's FD100 robot below,

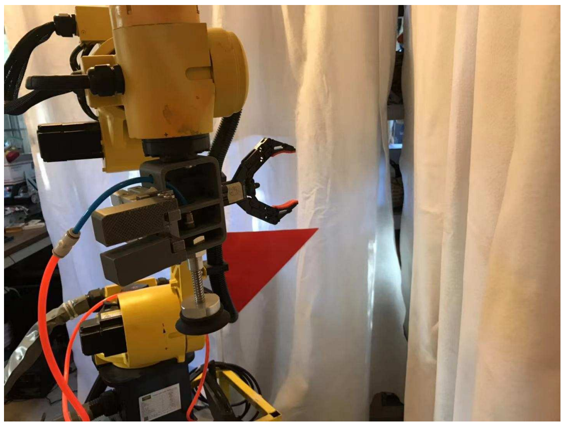

Fig. 1. An end effector (a set of three in this case) from CTI's FD100 robot below, whose movement forms a trajectory in 3D space".

The bigger view of the FD100 robot with an end effector (a set of three in this case) in the view.

Fig. 2. The bigger view of the FD100 robot with an end effector (a set of three in this case) in the view.

Definition 2. *Reward r. A reward r is defined as numerical value assigned to each state-action pair $s_i a_i$, e.g., formulate as $r : S \times A \rightarrow R$, where $S \times A = (s_1 a_1, s_2 a_1, ..., s_1 a_N, s_2 a_N, ..., s_N a_1, ..., s_N a_N)$.*

A reward r is formulated as $r : S \times A \rightarrow R$, where $S \times A = (s_1 a_1, s_2 a_1, ..., s_1 a_N, s_2 a_N, ..., s_N a_1, ..., s_N a_N)$.

We can build tables for S, A and R repectively, as follows

TABLE I
Table I. State Table for FD100 Robot

Category	Description	Note % Improvement
$s_{j1-angle}$	[,]	Continuous
$s_{j1-speed}$	[,]	Continuous
$s_{j1-accel}$	[,]	Continuous
$s_{j2-angle}$	[,]	Continuous
$s_{j2-speed}$	[,]	Continuous
$s_{j2-accel}$	[,]	Continuous
...	...	...
$s_{j6-angle}$	[,]	Continuous
$s_{j6-speed}$	[,]	Continuous
$s_{j6-accel}$	[,]	Continuous

TABLE II
Table II. Acition Table for FD100 Robot

Category	Description	Note % Improvement
$a_{j1-angle}$	[,]	Continuous
$a_{j1-speed}$	[,]	Continuous
$a_{j1-accel}$	[,]	??? Check this
$a_{j2-angle}$	[,]	Continuous
$a_{j2-speed}$	[,]	Continuous
$a_{j2-accel}$	[,]	Continuous
...	...	...
$a_{j6-angle}$	[,]	Continuous
$a_{j6-speed}$	[,]	Continuous
$a_{j6-accel}$	[,]	??? Check this

TABLE III
TABLE III. REWARD TABLE FOR FD100 ROBOT

Category	Description	Note % Improvement
$s_{j1-angle} \times a_{j1-angle}$	$[,]$	Continuous
$s_{j1-speed} \times a_{j1-angle}$	$[,]$	Continuous
$s_{j1-accel} \times a_{j1-angle}$	$[,]$	Continuous
...	...	...

Note, check the datasheet of FD100 to fill in the numerical values in the description section of the above tables.

IV. POLICY ON ROBOT OPERATIONS

Definition 3. *Policy π. A policy π in Robotics is a set of guidelines for a robot controller to follow to deliver its control action a_i upon its current state s_i, which is denoted as $\pi(a_i|\ s_i)$.*

A policy π leads to the robot control to deliver its control action as a mapping function $s_i \rightarrow a_i$.

A policy pi can be either stochastic which is characterized by a conditional probability as

$$\pi(a_i|s_i) : s_i \rightarrow Pr(a_i|s_i), \tag{6}$$

or deterministic

$$\pi(a_i|s_i) : s_i \rightarrow a_i = \mu(s_i). \tag{7}$$

V. POLICY AS DNN

We now introduce a notation to policy π as π_θ where a set of variables which affects π. In deep reinforcement learning (DRL), a policy π_θ is formulated as a deep neural network (DNN), where θ is the general parameter storing all the networks weights and biases $W = (w_{i,j})$.

Definition 4. *Policy π_θ. A policy π_θ is a deep neural network (DNN), where θ is the collection of all parameters of the neural networks (NN) weights and biases, simply denoted as $W = (w_{i,j})$.*

A typical realization of π_θ is a gaussian Multi-Layer Perceptron (MLP) net, which samples the action to be taken from a gaussian distribution of actions over states as follows

$$\pi_\theta(a_i|s_i) = \frac{1}{\sqrt{(2\pi)^{n_a} det \sum_\theta(s)}} E[-\frac{(||a-\mu_\theta(s)||)^2}{2\sum_\theta(s)}] \tag{8}$$

Example 1. Supppose we have $s_1 = -1, s_2 = 2$ and $a_1 = 10$ and $a_2 = 20$, find

(1) $\sum_\theta(s)$ and $det \sum_\theta(s)$,
(2) $\mu_\theta(s)$,
(3) $\pi_\theta(a_i|s_i) =?$
Sol:
(1) $\sum_\theta(s) = \frac{1}{2}(s_1 + s_2)$,
???
???

Algorithm 1 Adaptive Learning

```
Require: input video frame $I(x, y; t_i)$
Ensure: Start 2 Processes $P_i(w_i, b_i)$ for i=1, 2;
  while $P_1(w_i, b_i) \wedge P_2(w_i, b_i)$ do
    if $flag\_adaptive$ then
      Run $P_2(w_i, b_i)$ with new added images $I(x, y; t_i) \in \Omega_a$
      if $flag\_update$ then
        Update $P_1(w_i, b_i)$
      else
        Run $P_2(w_i, b_i)$ with new added images $I(x, y; t_i) \in \Omega_a$
      end if
    else
      Run $P_1(w_i, b_i)$ with original image dataset $\Omega$
    end if
  end while
```

VI. QUICK REVIEW

The objectives of deep reinforcement learning is to develop algorithms to make robot has the ability to learn new manipulation policies

(1) from scratch, without user demonstrations and
(2) without the need of a task-specific domain knowledge.

A. Trust Region Policy Optimization

Trust Region Policy Optimization (TRPO) [1]

B. Deep Q-Network

Deep Q-Network with Normalized Advantage Func- tions (DQN-NAF) [

C. Deep Deterministic Policy Gradient

Deep Deterministic Policy Gradient (DDPG) [3] and

D. Vanilla Policy Gradient

Vanilla Policy Gradient (VPG) [4].

E. hyper-parameters selection and tuning procedures

The hyper-parameters selection and tuning procedures
as well as demonstrate the robustness and adaptability of TRPO and DQN-NAF while performing manipulation tasks such as reaching a random position target and pick & placing an object.

Moreover, their model-freedom guarantees good performances even in case of changes in the dynamic and geometric models of the robot (e.g., link lengths, masses, and inertia).

???

ACKNOWLEDGMENT

I would like to express my thanks to CTI One engineering member and CTI One engineering intern team, as well as to SJSU graduate research assistants for letting me introducing DRL in Robotics to our next project and for the codings in Deep Learning, Robotis and embedded systems.

References

[1] [Franceschetti, 2020] Andrea Franceschetti, Elisa Tosello, Nicola Castaman, and Stefano Ghidoni, ”Robotic Arm Control and Task Training through Deep Reinforcement Learning”, https://arxiv.org/pdf/2005.02632.pdf, May 2020.

[2] [Reinforcement, 2020] Reinforcement learning, https://en.wikipedia.org/wiki/Reinforcement _learning, 2020.