Persistence Length Based Exploration for Continuous Control

Riashat Islam
(Joint work with Maziar Gomrokchi, Susan Amin & Doina Precup)

Reasoning and Learning Lab

McGill University

20th April 2017
Deep Reinforcement Learning

Locomotion Tasks

Swimmer  Hopper  Half Cheetah  Walker
Ant  Simplified Humanoid  Full Humanoid
Exploration in Continuous Control

- Exploring environment $\leftrightarrow$ Exploiting good behaviour
- In continuous control:
  - default exploration is through random control noise
- High dimensional continuous actions
  - Many directed exploration methods ($\epsilon$-greedy, Boltzmann) are limited to discrete action spaces
  - Current exploration strategies are insufficient

We propose trajectory based exploration method suited for continuous control tasks
Motivation

Off-Policy Actor-Critic

- DDPG in continuous control [Lillicrap et. al., 2016, Silver et. al., 2014]

However, no good exploration strategy to collect off-policy samples

- this talk: propose exploration method for off-policy actor-critic for continuous control

- Related current benchmark: VIME in on-policy TRPO [Houthooft et. al., 2016]
Persistence Length Exploration

Intuition:

- Choice of next exploratory action should depend on the trajectory so far
- Trajectories should fill up the entire state space
Persistence Length Exploration

- Mechanism of locally self avoiding random walk
- Adopted from physics literature to describe behaviour of polymer chains
- Consider trajectory upto current state to decide next action
- Pure exploration $\rightarrow$ plan trajectory to fill up entire environment
Persistence Length Exploration

- Self avoiding chains in d-dimensional action space
- Self avoiding trajectory
- Travel quickly around environment depending on parameterization
- Persistence length $L_p$ quantifies stiffness of the chain
Algorithm 1: PolyRL Algorithm (2D Action Space) on top of DDPG

1. Randomly initialize critic network $Q(s,a; \theta^Q)$ and actor network $\mu(s;\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$;
2. Initialise target network $Q'$ and policy network $\mu'$;
3. Initialise two replay buffers $B^e$ and $B^d$;
4. for episode = 1, 2, ..., $M$ do
5. PolyRL pure exploration phase → for expl epoch until $e = E$ do
6. if $e == 0$ then
7. Sample $A_0$ and $S_0$ w.r.t $\rho$;
8. else if $e == 1$ then
9. Initialize $H_1$ s.t. $|H_1| = b_o$;
10. $A_1 \leftarrow A_0 + H_1$;
11. else
12. Draw a sample $\theta$ from $\mathcal{N}(\mu, \sigma)$;
13. $\theta_t \leftarrow$ toss a coin and choose between $\theta$ and $-\theta$;
14. $A_e \leftarrow A_{e-1} + \alpha \|H_e\|_2$ on $H_{e-1}$;
15. if $A_e$ is not valid then
16. Terminate the episode;
17. else
18. Apply step function on action $A_e$ and observe $S_{e+1}$ and $R_{e+1}$;
19. if $S_{e+1}$ is valid then
20. Continue;
21. else
22. End the episode and re-start the chain;
23. Sample a random minibatch of transitions from buffer $B^e$;
24. Update the Q critic network using off-policy exploration samples;
25. Return trajectory of states and actions;
26. Return end of trajectory state and action;
27. Return updated Q critic network from PolyRL exploration phase;

Deep Deterministic Policy Gradient (DDPG);
for $t=1, 2, ..., T$ do
1. Select action $a_t$ according to current policy $\mu(s_t;\theta^\mu)$;
2. Execute action $a_t$ and observe reward $r_{t+1}$ and next state $s_{t+1}$;
3. Store transition to replay buffer $B^d$;
4. Sample random minibatch from replay buffer $B^d$;
5. Update the critic network by minimizing the loss;
6. Update the actor policy network using sampled policy gradient;
7. Update the target networks;
PolyRL Exploration (2D Action Space)

(a) Episode 1
(b) Episode 2

**Figure**: Exploratory action trajectory
PolyRL + DDPG (MuJoCo Hopper)
PolyRL + DDPG (MuJoCo Swimmer)
# Few Benchmark Results (Max Return)

<table>
<thead>
<tr>
<th>Task</th>
<th>Action Dim</th>
<th>TRPO</th>
<th>DDPG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swimmer</td>
<td>2D</td>
<td>110</td>
<td>150</td>
</tr>
<tr>
<td>Reacher</td>
<td>2D</td>
<td>-6.7</td>
<td>-6.6</td>
</tr>
<tr>
<td>Hopper</td>
<td>3D</td>
<td>2486</td>
<td>2604</td>
</tr>
<tr>
<td>HalfCheetah</td>
<td>6D</td>
<td>4734</td>
<td>7490</td>
</tr>
<tr>
<td>Walker</td>
<td>6D</td>
<td>3567</td>
<td>3626</td>
</tr>
<tr>
<td>Humanoid</td>
<td>17D</td>
<td>918</td>
<td>552</td>
</tr>
</tbody>
</table>
Current Benchmark - VIME
MuJoCo Walker2D, Swimmer

Figure 3: Performance of TRPO with and without VIME on the high-dimensional Walker2D locomotion task.

Figure 4: VIME: performance over the first few iterations for TRPO, REINFORCE, and ERWR i.f.o. $\eta$ on MountainCar.

Figure 5: Performance of TRPO with and without VIME on the challenging hierarchical task SwimmerGather.
Thank You

Questions...