Persistence Length Based Exploration for Continuous Control

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

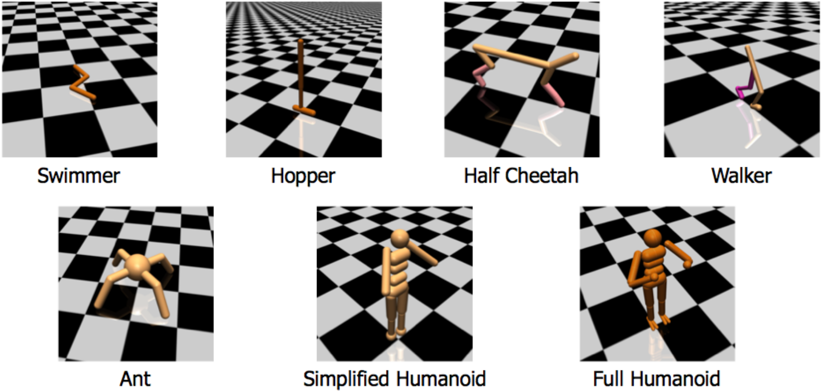
Reasoning and Learning Lab

RLLab
McGill University

20th April 2017

Deep Reinforcement Learning

Locomotion Tasks



Exploration in Continuous Control

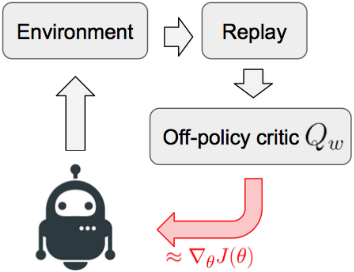
- Exploring environment $\longleftrightarrow$ Exploiting good behaviour
- In continuous control :
  default exploration is through random control noise
- High dimensional continous actions
  - Many directed exploration methods (ϵ-greedy, Boltzmann) are limited to discrete action spaces
  - Current exploration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



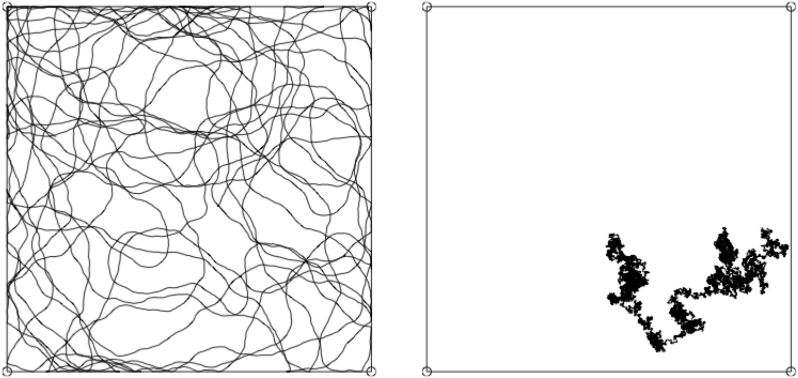
However, no good exploration strategy to collect off-policy samples

- 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 dependent on the trajectory so far
- Trajectories should fill up the entire state space

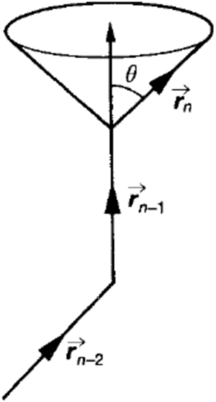


Persistence Length Exploration

- 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



PolyRL + DDPG

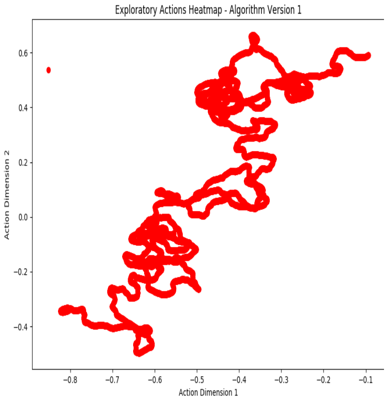
Algorithm 1: PolyRL Algorithm (2D Action Space) on top of DDPG

Randomly initialize critic network Q(s,a | θ^Q) and actor network $\mu(s|\theta^\mu$ with weights θ^Q and θ^μ;
Initialise target network Q' and policy network μ';
Initialise two replay buffers B^e and B^d;
for *episode=1, 2, ... M* **do**
  PolyRL pure exploration phase $\rightarrow$ **for** *expl epoch until $e = E$* **do**
    if $e == 0$ **then**
      Sample $\mathbf{A_0}$ and S_0 w.r.t ρ;
    else if $e == 1$ **then**
      Initialize $\mathbf{H_1}$ s.t. $||\mathbf{H_1}|| = b_o$;
      $\mathbf{A_1} \leftarrow \mathbf{A_o} + \mathbf{H_1}$;
    else
      Draw a sample θ from $\mathcal{N}(\mu, \sigma)$;
      $\theta_t \leftarrow$ toss a coin and choose between θ and $-\theta$;
      $\mathbf{A_e} \leftarrow \mathbf{A_{e-1}} +$ apply $\prod_2^{\theta_e}$ on $\mathbf{H_{e-1}}$;
    if *A_e is not valid* **then**
      Terminate the episode;
    else
      Apply step function on action $\mathbf{A_e}$ and observe S_{e+1} and R_{e+1};
      if *S_{e+1} is valid* **then**
        Continue;
      else
        End the episode and re-start the chain;
    Sample a random minibatch of transitions from buffer B^e;
    Update the Q critic network using off-policy exploration samples;
  Return trajectory of states and actions;
  Return end of trajectory state and action;
  Return updated Q critic network from PolyRL exploration phase;
  Deep Deterministic Policy Gradient (DDPG);
  for *t=1, 2,...T* **do**
    Select action a_t according to current policy $\mu(s_t|\theta^{mu})$;
    Executve action a_t and observe reward r_{t+1} and next state s_{t+1};
    Store transition to replay buffer B^d;
    Sample random minibatch from replay buffer B^d;
    Update the critic network by minimizing the loss;
    Update the actor policy network using sampled policy gradient ;
    Update the target networks;

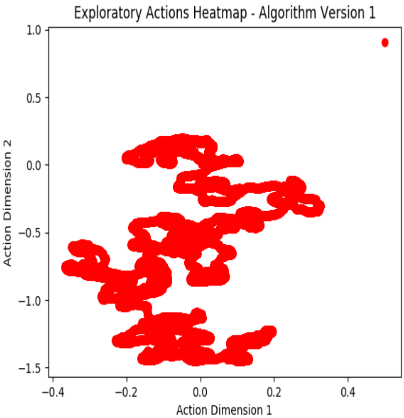
Persistence Length Based Exploration

DDPG

PolyRL Exploration (2D Action Space)



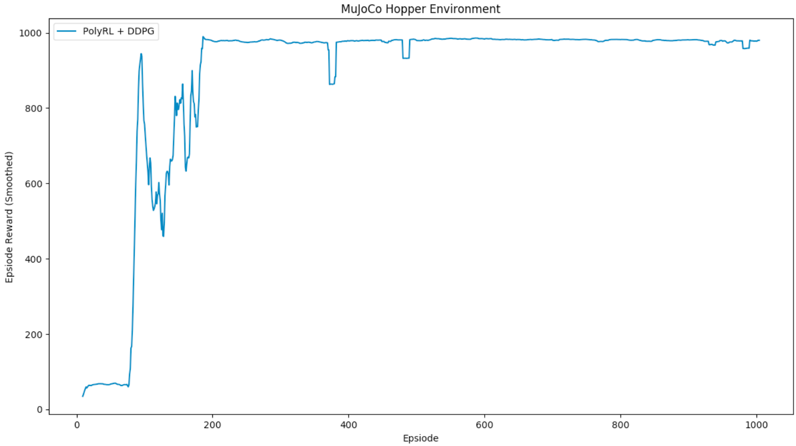
(a) Episode 1



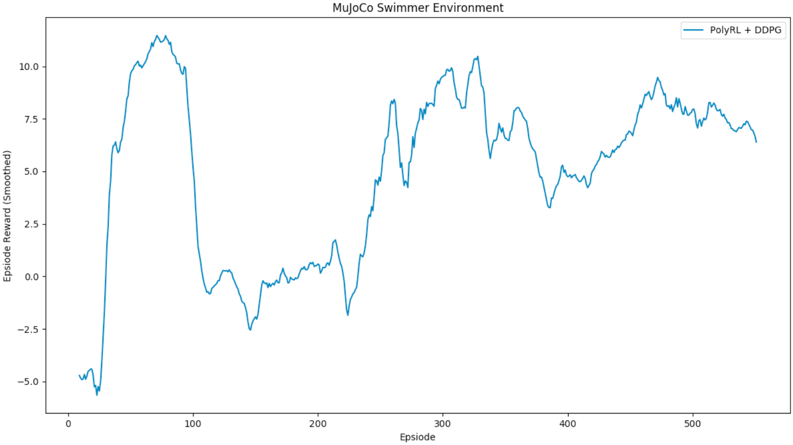
(b) Episode 2

Figure: Exploratory action trajectory

PolyRL + DDPG (MuJoCo Hopper)



PolyRL + DDPG (MuJoCo Swimmer)



Policy Gradients on MuJoCo Tasks

Few Benchmark Results (Max Return)			
Task	**Action Dim**	**TRPO**	**DDPG**
Swimmer	2D	110	150
Reacher	2D	-6.7	-6.6
Hopper	3D	2486	2604
HalfCheetah	6D	4734	7490
Walker	6D	3567	3626
Humanoid	17D	918	552

Current Benchmark - VIME

MuJoCo Walker2D, Swimmer

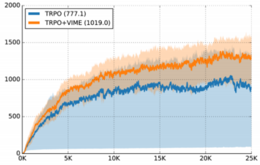


Figure 3: Performance of TRPO with and without VIME on the high-dimensional Walker2D locomotion task.

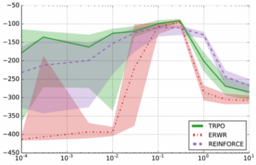


Figure 4: VIME: performance over the first few iterations for TRPO, REINFORCE, and ERWR i.f.o. η on MountainCar.

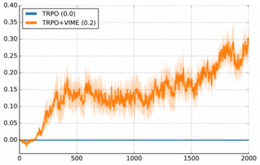


Figure 5: Performance of TRPO with and without VIME on the challenging hierarchical task SwimmerGather.

Thank You

Questions...