Deep RL in Parameterized Action Space

Reinforcement Learning Group Meeting

Yanjun Gao

Jan 31, 2019

Recall … Deep Deterministic Policy Gradients (DDPG)

Deterministic Policy Gradients Theorem

Regularity conditions A.1: $p(s'|s,a)$, $\nabla_a p(s'|s,a)$, $\mu_\theta(s)$, $\nabla_\theta \mu_\theta(s)$, $r(s,a)$, $\nabla_a r(s,a)$, $p_1(s)$ are continuous in all parameters and variables s, a, s' and x.

Theorem 1 (Deterministic Policy Gradient Theorem). *Suppose that the MDP satisfies conditions A.1 (see Appendix; these imply that $\nabla_\theta \mu_\theta(s)$ and $\nabla_a Q^\mu(s,a)$ exist and that the deterministic policy gradient exists. Then,*

$$\begin{aligned}\nabla_\theta J(\mu_\theta) &= \int_{\mathcal{S}} \rho^\mu(s) \nabla_\theta \mu_\theta(s)\ \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)}\, \mathrm{d}s \\ &= \mathbb{E}_{s\sim\rho^\mu}\left[\nabla_\theta \mu_\theta(s)\ \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)}\right] \quad (9)\end{aligned}$$

Silver, David, et al. "Deterministic policy gradient algorithms." *ICML*. 2014.

Recall … Deep Deterministic Policy Gradients (DDPG)

Implementation Details

- Copy of actor and critic networks to be the target network, with slow update rate:

$$Q'(s, a|\theta^{Q'}) \text{ and } \mu'(s|\theta^{\mu'})$$

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta' \text{ with } \tau \ll 1$$

To improve the stability of Q network, with a much smaller changes constrianed by the updates. (Recall Q learning "not explore enough" situation)

- Batch Normalization: scale the features into the same range so they could be trained easily

- Noisy process in critic networks to construct exploration policy

$$\mu'(s_t) = \mu(s_t|\theta_t^{\mu}) + \mathcal{N}$$

- Experience replay buffer is still useful

Lillicrap, Timothy P., et al. "CONTINUOUS CONTROL WITH DEEP REINFORCEMENT LEARNING." (2015).

This week …

Hausknecht, Matthew, and Peter Stone. "Deep reinforcement learning in parameterized action space." ICLR 2016.

- How DDPG is applied in real-word problem, especially in terms of "stability"
- How they designed the architecture
- What is "parameterized action space"

Background

RoboCup 2D Soccer Simulation Movie

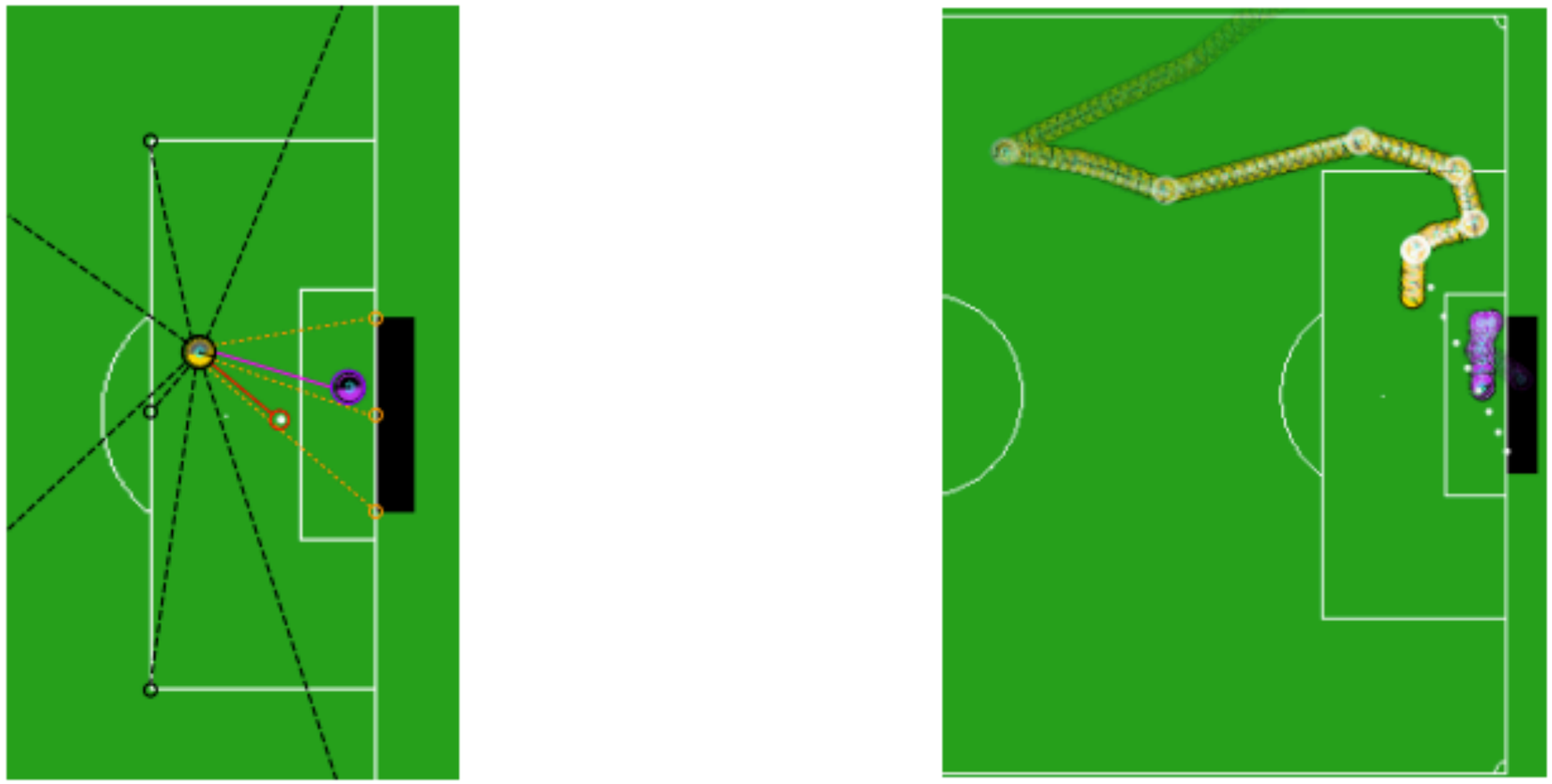
Full Soccer Task (Half Field Offense)

- The world: players (two teams), ball, field are all 2-D objects
- Goal: scoring and defending the ball
- Episodic Multi-agent POMDP; episode ends when one team scores or the ball leaves the field
- Each agent recieve its own state space and selects its action
- At the beginning of each episode, the agent and ball will be placed randomly on the field.

Full Soccer Task is challenging because …

- Full games are lengthy;
- High variance in outcome;
- Require special handling of rules;
- Involving many decision-making logic

State Space



(a) State Space

(b) Helios Champion

Figure 1: **Left:** HFO State Representation uses a low-level, egocentric viewpoint providing features such as distances and angles to objects of interest like the ball, goal posts, corners of the field, and opponents. **Right:** Helios handcoded policy scores on a goalie. This 2012 champion agent forms a natural (albeit difficult) baseline of comparison.

Action Space

4 Discrete Actions:

- Each action associates with 1-2 contiuously-valued parameters
- Dash(power, direction)
- Turn(direction)
- Tackle(direction)
- Kick(power, direction)

Power: [0, 100]

Direction: [-180, 180]

Reward Signal

- Move to Ball: scalar, proportional to changes between agents and the ball in distance d(a,b); becomes negative if the ball moves too far from the agent

- Kick to Ball: scalar, proportional to distance between the ball and the center of the goal d(b,g); addtional reward will be given if score

$$r_t = d_{t-1}(a,b) - d_t(a,b) + \mathbb{I}_t^{kick} + 3\big(d_{t-1}(b,g) - d_t(b,g)\big) + 5\mathbb{I}_t^{goal} \quad (1)$$

Recall Q-learning:

$$Q(s,a) = Q(s,a) + \alpha\big(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\big)$$

Recall Actor Network:

$$L_Q(s,a|\theta^Q) = \Big(Q(s,a|\theta^Q) - \big(r + \gamma Q(s', \mu(s'|\theta^\mu)'|\theta^Q)\big)\Big)^2$$

Recall Critic Network:

$$L_\mu(s|\theta^\mu) = \big(a - a^*\big)^2 = \big(\mu(s|\theta^Q) - a^*\big)^2$$

Gradients computed by:

$$\nabla_{\theta^\mu}\mu(s) = \nabla_a Q(s,a|\theta^Q)\nabla_{\theta^\mu}\mu(s|\theta^\mu)$$

DRL with more details

Recall implementation details

Employing these two techniques the critic loss in Equation 4 and actor update in Equation 5 can be stably re-expressed as follows:

$$L_Q(\theta^Q) = \mathbb{E}_{(s_t,a_t,r_t,s_{t+1})\sim\mathcal{D}}\left[\left(Q(s_t,a_t) - \left(r_t + \gamma Q'(s_{t+1},\mu'(s_{t+1}))\right)\right)^2\right] \tag{7}$$

$$\nabla_{\theta^\mu}\mu = \mathbb{E}_{s_t\sim\mathcal{D}}\left[\nabla_a Q(s_t,a|\theta^Q)\nabla_{\theta^\mu}\mu(s_t)|_{a=\mu(s_t)}\right] \tag{8}$$

Finally, these updates are applied to the respective networks, where α is a per-parameter step size determined by the gradient descent algorithm. Additionally, the target-actor and target-critic networks are updated to smoothly track the actor and critic using a factor $\tau \ll 1$:

$$\begin{aligned}
\theta^Q &= \theta^Q + \alpha\nabla_{\theta^Q}L_Q(\theta^Q)\\
\theta^\mu &= \theta^\mu + \alpha\nabla_{\theta^\mu}\mu\\
\theta^{Q'} &= \tau\theta^Q + (1-\tau)\theta^{Q'}\\
\theta^{\mu'} &= \tau\theta^\mu + (1-\tau)\theta^{\mu'}
\end{aligned} \tag{9}$$

Architecture

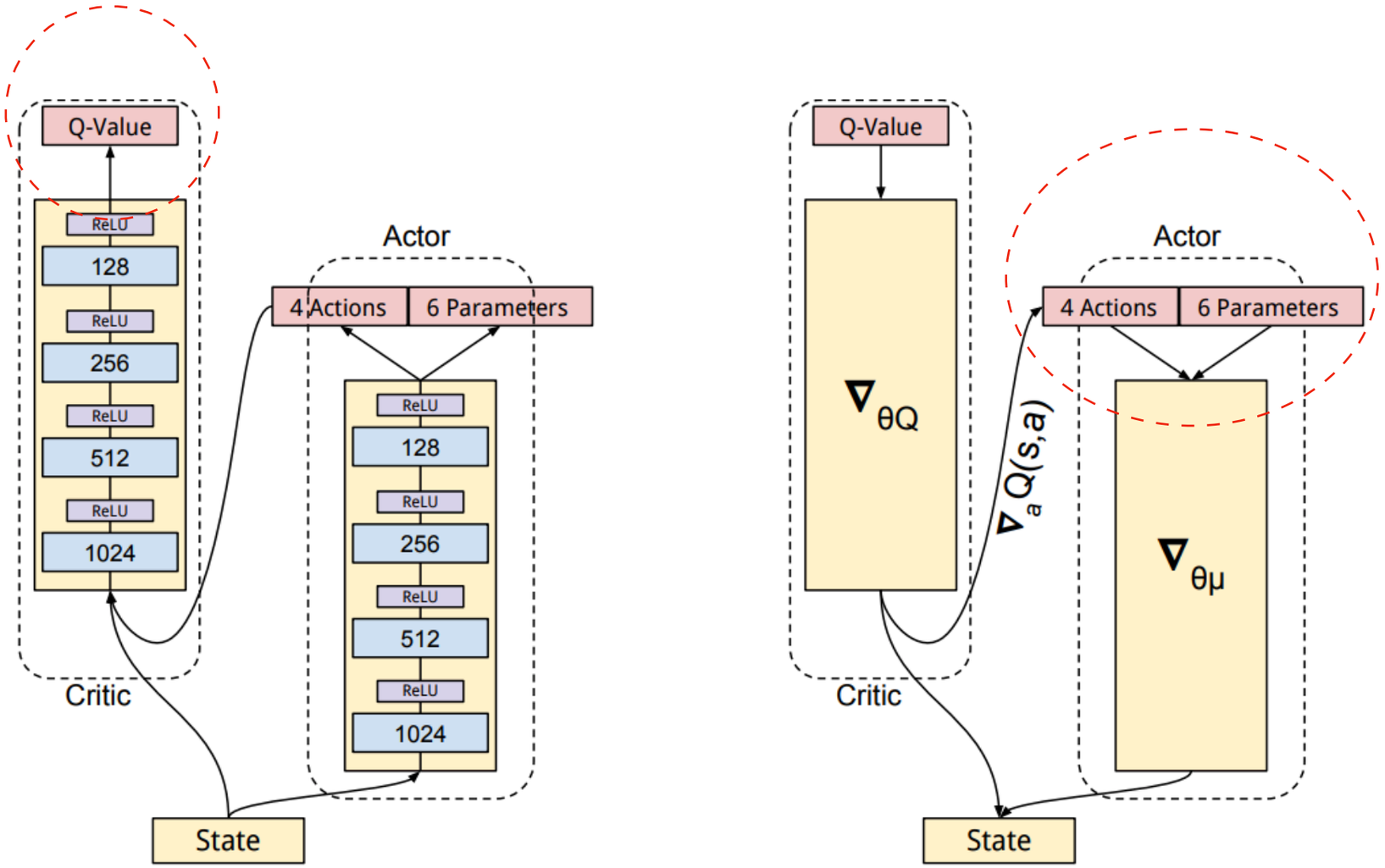


Figure 2: **Actor-Critic architecture (left)**: actor and critic networks may be interlinked, allowing activations to flow forwards from the actor to the critic and gradients to flow backwards from the critic to the actor. The gradients coming from the critic indicate directions of improvement in the continuous action space and are used to train the actor network without explicit targets. **Actor Update (right)**: Backwards pass generates critic gradients $\nabla_a Q(s, a|\theta^Q)$ w.r.t. the action. These gradients are back-propagated through the actor resulting in gradients w.r.t. parameters ∇_{θ^μ} which are used to update the actor. Critic gradients w.r.t. parameters ∇_{θ^Q} are ignored during the actor update.

Parameterized action space

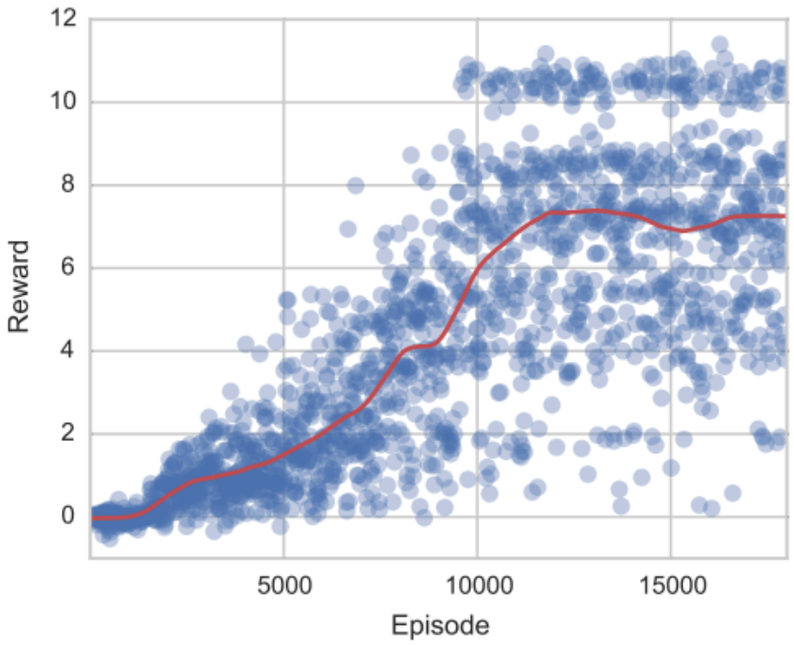
Following notation in (Masson & Konidaris, 2015), a Parameterized Action Space Markov Decision Process (PAMDP) is defined by a set of discrete actions $A_d = \{a_1, a_2, \ldots, a_k\}$. Each discrete action $a \in A_d$ features m_a continuous parameters $\{p_1^a, \ldots, p_{m_a}^a\} \in \mathbb{R}^{m_a}$. Actions are represented by tuples $(a, p_1^a, \ldots, p_{m_a}^a)$. Thus the overall action space $A = \cup_{a \in A_d}(a, p_1^a, \ldots, p_{m_a}^a)$.

In Half Field Offense, the complete parameterized action space (Section 2.2) is $A = (\text{Dash}, p_1^{\text{dash}}, p_2^{\text{dash}}) \cup (\text{Turn}, p_3^{\text{turn}}) \cup (\text{Tackle}, p_4^{\text{tackle}}) \cup (\text{Kick}, p_5^{\text{kick}}, p_6^{\text{kick}})$. The actor network in Figure 2 factors the action space into one output layer for discrete actions $(\text{Dash}, \text{Turn}, \text{Tackle}, \text{Kick})$ and another for all six continuous parameters $(p_1^{\text{dash}}, p_2^{\text{dash}}, p_3^{\text{turn}}, p_4^{\text{tackle}}, p_5^{\text{kick}}, p_6^{\text{kick}})$.

Action selection and exploration

- Actor chooses action with the parameters
- While training, critic network will not know which action is executed in the world, it just provides gradients to actor; this proved to work well in experiments
- Exploration is done by random discretization of continous action space follwing epsilon-greedy style

Scores Evaluation



(a) Learning Curve

	Scoring Percent	Avg. Steps to Goal
Helios' Champion	.962	72.0
SARSA	.81	70.7
$DDPG_1$	1	108.0
$DDPG_2$	.99	107.1
$DDPG_3$	.98	104.8
$DDPG_4$	.96	112.3
$DDPG_5$	.94	119.1
$DDPG_6$	.84	113.2
$DDPG_7$	.80	118.2

(b) Evaluation Performance

Figure 4: **Left:** Scatter plot of learning curves of DDPG-agents with Lowess curve. Three distinct phases of learning may be seen: the agents first get small rewards for approaching the ball (episode 1500), then learn to kick the ball towards the goal (episodes 2,000 - 8,000), and start scoring goals around episode 10,000. **Right:** DDPG-agents score nearly as reliably as expert baseline, but take longer to do so. A video of $DDPG_1$'s policy may be viewed at `https://youtu.be/Ln0Cl-jE_40`.