(D)DPG:
(Deep) Deterministic Policy Gradients Network for Continuous Action Space

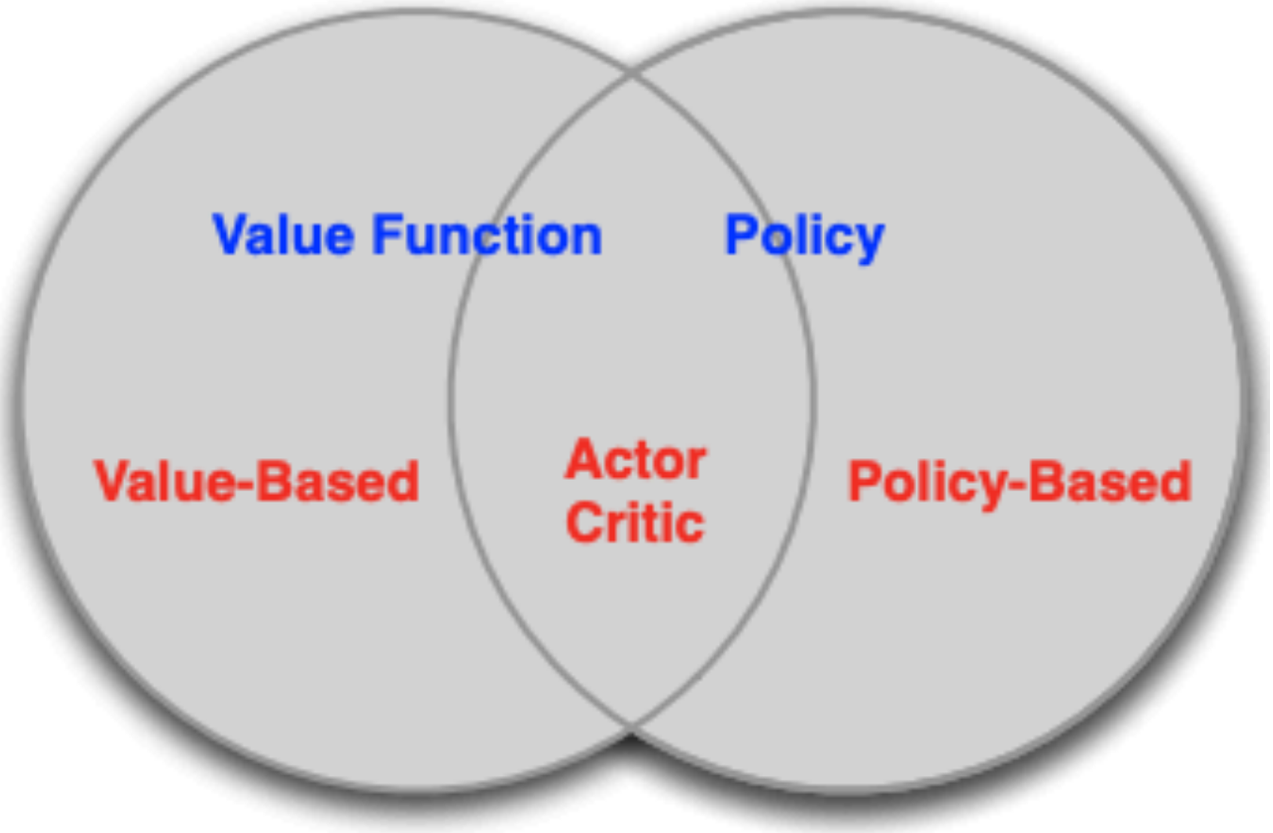
Reinforcement Learning Group Meeting

Yanjun Gao

Jan 24, 2019

Value-Based and Policy-Based RL

- Value Based
  - Learnt Value Function
  - Implicit policy (e.g. ϵ-greedy)
- Policy Based
  - No Value Function
  - Learnt Policy
- Actor-Critic
  - Learnt Value Function
  - Learnt Policy



Slides borrowed from: David Silver - Lecture 7: Policy Gradient Methods

Table of Content

Parameterized Policies

Recall Parameterized Policies: $\pi_\theta(s, a) = \mathbb{P}[a \mid s, \theta]$

- A family of policies indexed by parameter vector $\theta \in \mathbb{R}^d$
  - Deterministic: $a = \pi(s, \theta)$
  - Stochastic: $\pi(a \mid s, \theta)$
- Analogous to classification or regression with input s, output a.
  - Discrete action space: network outputs vector of probabilities
  - Continuous action space: network outputs mean and diagonal covariance of Gaussian

Stochastic: To represent the policy by a parametric probability over state space and action space and selects action a in state s according to parameter vector theta.

Deterministic: Models the decision policy only related to the state s.

Why deterministic?

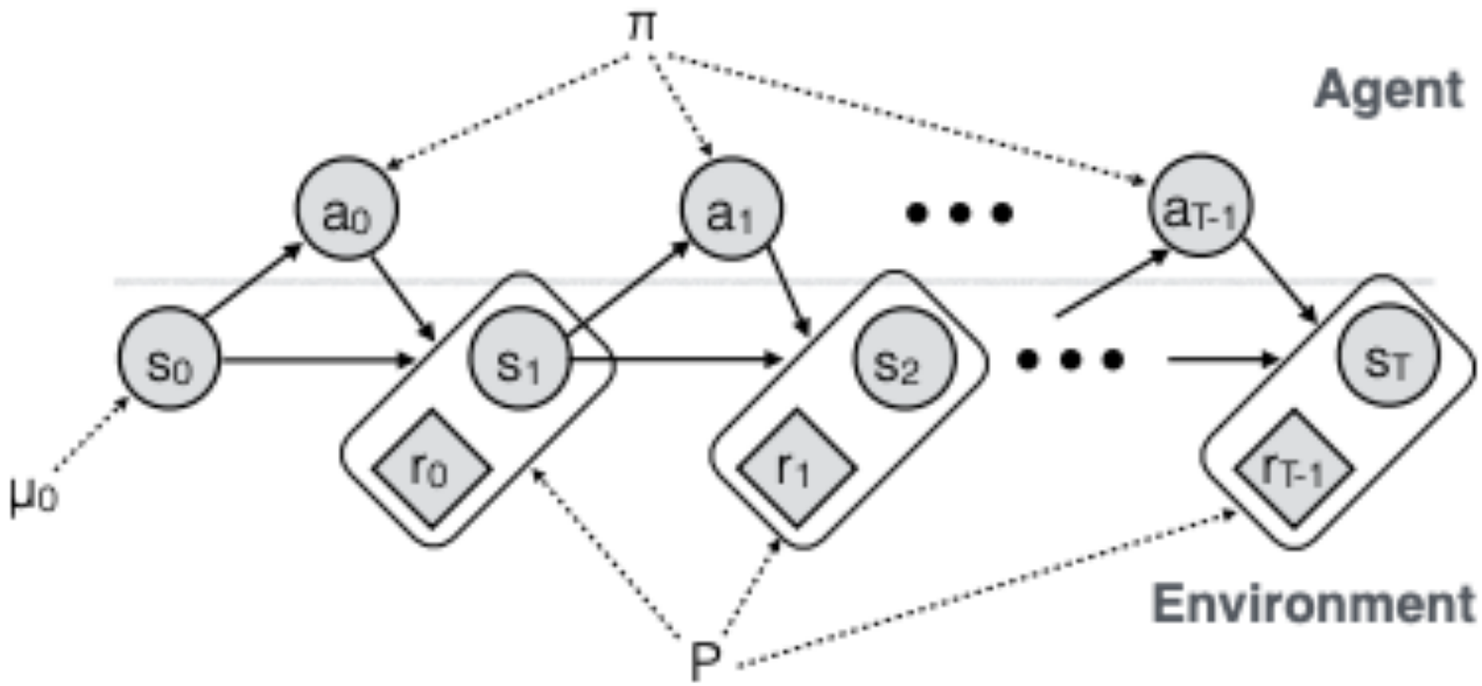
It helps to deal with high-dimensional action spaces, continuous action spaces, etc.

Slides borrowed from: David Silver - Lecture 7: Policy Gradient Methods
UC Berkeley, CS 294: Deep Reinforcement Learning, Spring 2017, Lec2: Policy Gradient Methods

Deterministic Policy Gradients

Preliminaries: Policy Gradients

Recall MDP



Objective:

$$\text{maximize}\,\eta(\pi),\ \text{where}$$
$$\eta(\pi) = E[r_0 + r_1 + \cdots + r_{T-1} \mid \pi]$$

Agent goal: obtain a policy that maximize the cumulative discounted reward

$$r_t^\gamma = \sum_{k=t}^{\infty} \gamma^{k-t} r(s_k, a_k) \text{ where } 0 < \gamma < 1$$

Deterministic Policy Gradients

Preliminaries: Policy Gradients

We denote the density at state s' after transitioning for t time steps from state s by $p(s \to s', t, \pi)$. We also denote the (improper) discounted state distribution by $\rho^\pi(s') := \int_\mathcal{S} \sum_{t=1}^\infty \gamma^{t-1} p_1(s) p(s \to s', t, \pi) \mathrm{d}s$. We can then write the performance objective as an expectation,

p1(s): initial state distributions

$$
\begin{aligned}
J(\pi_\theta) &= \int_\mathcal{S} \rho^\pi(s) \int_\mathcal{A} \pi_\theta(s, a) r(s, a) \mathrm{d}a \mathrm{d}s \\
&= \mathbb{E}_{s \sim \rho^\pi, a \sim \pi_\theta} [r(s, a)] \qquad (1)
\end{aligned}
$$

where $\mathbb{E}_{s \sim \rho}[\cdot]$ denotes the (improper) expected value with respect to discounted state distribution $\rho(s)$.[2] In the remainder of the paper we suppose for simplicity that $\mathcal{A} = \mathbb{R}^m$ and that $\mathcal{S}$ is a compact subset of $\mathbb{R}^d$.

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

Deterministic Policy Gradients

Preliminaries: Stochastic Policy Gradients
Intutition: Update parameters theta with the direction of gradients $\nabla_\theta J(\pi_\theta)$

$$\nabla_\theta J(\pi_\theta) = \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \nabla_\theta \pi_\theta(a|s) Q^\pi(s,a) \mathrm{d}a \mathrm{d}s$$
$$= \mathbb{E}_{s\sim\rho^\pi, a\sim\pi_\theta} \left[\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s,a)\right] \quad (2)$$

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

Deterministic Policy Gradients

Deterministic Policy Gradients Theorem

Motivation: in continous action space, evaluating the action-value function (maximization) becomes expensive

We now formally consider a deterministic policy $\mu_\theta : \mathcal{S} \rightarrow \mathcal{A}$ with parameter vector $\theta \in \mathbb{R}^n$. We define a performance objective $J(\mu_\theta) = \mathbb{E}\left[r_1^\gamma | \mu\right]$, and define probability distribution $p(s \rightarrow s', t, \mu)$ and discounted state distribution $\rho^\mu(s)$ analogously to the stochastic case. This again lets us to write the performance objective as an expectation,

$$\begin{aligned} J(\mu_\theta) &= \int_{\mathcal{S}} \rho^\mu(s) r(s, \mu_\theta(s)) \mathrm{d}s \\ &= \mathbb{E}_{s \sim \rho^\mu}\left[r(s, \mu_\theta(s))\right] \end{aligned} \tag{8}$$

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

Deterministic Policy Gradients

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.

Deep Deterministic Policy Gradients (DDPG)

Motivation: High-dimensional, continous action space cant be simply discretized (insufficient exploration space; might throw away the information); see example about human arm system with 7 degree of freedom. DQN is not applicable to continous action space due to the efficiency.

Contributions:

- A novel model-free, off-policy actor-critic RL architecture that applies DPG and DQN for solving RL problems in both high-dimensional and low-dimensional action space.

- Avoids inefficient computation from action-value function maximization

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

Deep Deterministic Policy Gradients (DDPG)

The action-value function is used in many reinforcement learning algorithms. It describes the expected return after taking an action a_t in state s_t and thereafter following policy π:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{\pi}\left[R_t | s_t, a_t\right] \tag{1}$$

Many approaches in reinforcement learning make use of the recursive relationship known as the Bellman equation:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{s_{t+1} \sim E, a_t \sim \pi}\left[r(s_t, a_t) + \gamma\, \mathbb{E}_{\pi}\left[Q^{\pi}(s_{t+1}, a_{t+1})\right]\right] \tag{2}$$

If the target policy is deterministic we can describe it as a function $\mu : \mathcal{S} \leftarrow \mathcal{A}$ and avoid the inner expectation:

$$Q^{\mu}(s_t, a_t) = \mathbb{E}_{s_{t+1} \sim E}\left[r(s_t, a_t) + \gamma Q^{\mu}(s_{t+1}, \mu(s_{t+1}))\right] \tag{3}$$

Notice that the outer expectation depends only on the environment. This means that it is possible to learn Q^{μ} off-policy, using transitions which are generated from a different behaviour policy μ'.

Q-learning [9], a commonly used off-policy algorithm, uses the greedy policy $\mu(s) = \arg\max_a Q(s, a)$. We consider function approximators parameterized by θ^Q, which we optimize by minimizing the loss:

$$L(\theta^Q) = \mathbb{E}_{\mu'}\left[\left(Q(s_t, a_t | \theta^Q) - y_t\right)^2\right] \tag{4}$$

where

$$y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \mu(s_{t+1}) | \theta^Q). \tag{5}$$

While y_t is also dependent on θ^Q, this is typically ignored.

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

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).

Deep Deterministic Policy Gradients Algorithm (DDPG)

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q)^2)$
    Update the actor policy using the sampled gradient:

$$\nabla_{\theta^\mu}\mu|_{s_i} \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

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

Deep Deterministic Policy Gradients Algorithm (DDPG)

Experiments (Movie)

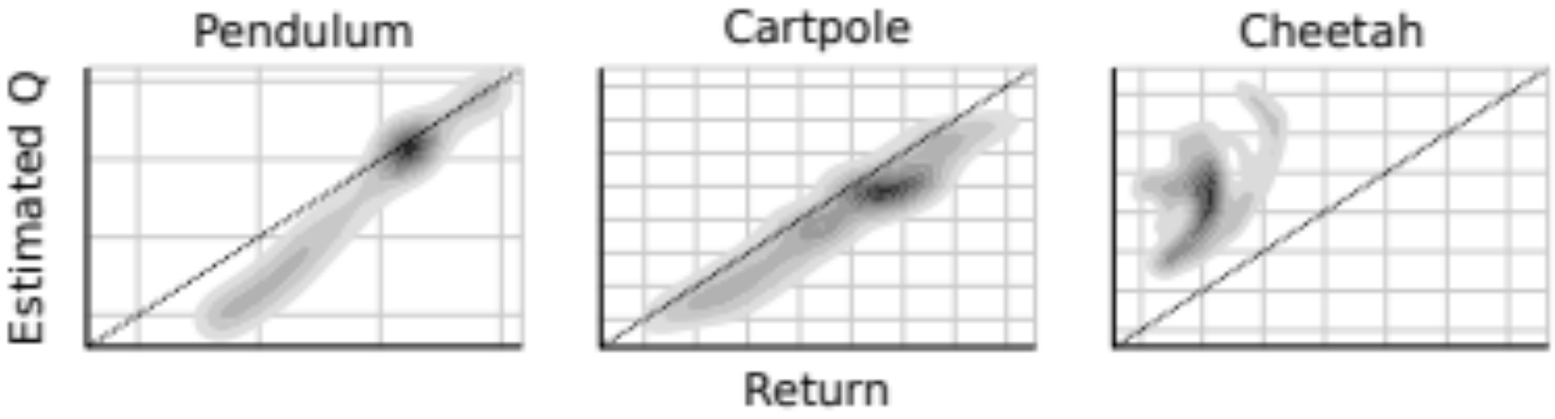


Figure 3: Density plot showing estimated Q values versus observed returns sampled from test episodes on 5 replicas. In simple domains such as pendulum and cartpole the Q values are quite accurate. In more complex tasks, the Q estimates are less accurate, but can still be used to learn competent policies. Dotted line indicates unity, units are arbitrary.

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

Deep Deterministic Policy Gradients Algorithm (DDPG)

Experiments (Movie)

Table 1: Performance after training across all environments for at most 2.5 million steps. We report both the average and best observed (across 5 runs). All scores, except Torcs, are normalized so that a random agent receives 0 and a planning algorithm 1; for Torcs we present the raw reward score.

environment	$R_{av,lowd}$	$R_{best,lowd}$	$R_{av,pix}$	$R_{best,pix}$
blockworld1	1.156	1.511	0.466	1.299
blockworld3da	0.340	0.705	0.889	2.225
canada	0.303	1.735	0.176	0.688
canada2d	0.400	0.978	-0.285	0.119
cart	0.938	1.336	1.096	1.258
cartpole	0.844	1.115	0.482	1.138
cartpoleBalance	0.951	1.000	0.335	0.996
cartpoleParallelDouble	0.549	0.900	0.188	0.323
cartpoleSerialDouble	0.272	0.719	0.195	0.642
cartpoleSerialTriple	0.736	0.946	0.412	0.427
cheetah	0.903	1.206	0.457	0.792
fixedReacher	0.849	1.021	0.693	0.981
fixedReacherDouble	0.924	0.996	0.872	0.943
fixedReacherSingle	0.954	1.000	0.827	0.995
gripper	0.655	0.972	0.406	0.790
gripperRandom	0.618	0.937	0.082	0.791
hardCheetah	1.311	1.990	1.204	1.431
hopper	0.676	0.936	0.112	0.924
hyq	0.416	0.722	0.234	0.672
movingGripper	0.474	0.936	0.480	0.644
pendulum	0.946	1.021	0.663	1.055
reacher	0.720	0.987	0.194	0.878
reacher3daFixedTarget	0.585	0.943	0.453	0.922
reacher3daRandomTarget	0.467	0.739	0.374	0.735
reacherSingle	0.981	1.102	1.000	1.083
walker2d	0.705	1.573	0.944	1.476
torcs	-393.385	1840.036	-401.911	1876.284

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

Take-away

- A novel model-free, off-policy actor-critic RL architecture that applies DPG and DQN for solving RL problems in both high-dimensional and low-dimensional action space.
- Avoids inefficient computation from action-value function maximization
- We dont see continuous action space often in NLP, but there are many cases that the action space are continuous or in high-dimensional