Deep Reinforcement Learning

Overview of main articles
Part 3. Advanced topics

Sergey Ivanov

November 2, 2018

MSU

Table of contents i

Continuous Control (2016)

Continuous Control Task

Consider that action space is $\mathbb{R}^n$, i. e. continuous.

Continuous Control Task

Consider that action space is $\mathbb{R}^n$, i. e. continuous.

* policy gradients algorithms can be used with $\pi_\theta(a \mid s)$ parameterized as some distribution (for example, $\mathcal{N}(\mu_\theta(s_t), \sigma_\theta(s_t)I)$)

Continuous Control Task

Consider that action space is $\mathbb{R}^n$, i. e. continuous.

* policy gradients algorithms can be used with $\pi_\theta(a \mid s)$ parameterized as some distribution (for example, $\mathcal{N}(\mu_\theta(s_t), \sigma_\theta(s_t)I)$)
    - limits model's capacity

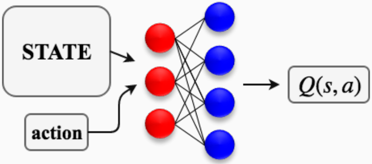
Continuous Control Task

Consider that action space is $\mathbb{R}^n$, i. e. continuous.

* policy gradients algorithms can be used with $\pi_\theta(a \mid s)$ parameterized as some distribution (for example, $\mathcal{N}(\mu_\theta(s_t), \sigma_\theta(s_t)I)$)
    - limits model's capacity
- Deep Q-learning based methods become extremely expensive.

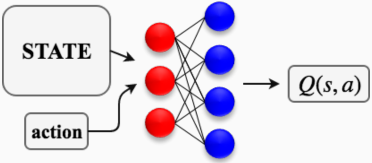
Problem with «continuous Q-learning»

The only possible architecture of Q-network:



Problem with «continuous Q-learning»

The only possible architecture of Q-network:

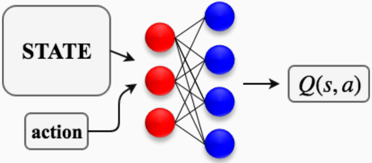


Problem: we require $\underset{a}{argmax}\, Q(s, a)$ for policy and target generation.

* well, maximize $Q(s, a)$ with respect to input a.

Problem with «continuous Q-learning»

The only possible architecture of Q-network:

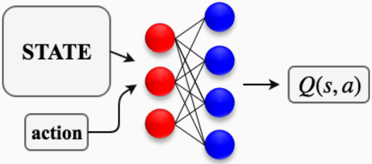


Problem: we require $\underset{a}{argmax}\, Q(s, a)$ for policy and target generation.

* well, maximize $Q(s, a)$ with respect to input a.
  - needs a lot of passes through the network on each step $\Rightarrow$ strong slowdown.

Problem with «continuous Q-learning»

The only possible architecture of Q-network:

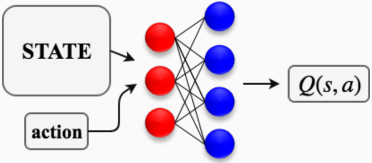


Problem: we require $\underset{a}{argmax}\, Q(s, a)$ for policy and target generation.

* well, maximize $Q(s, a)$ with respect to input a.
  - needs a lot of passes through the network on each step $\Rightarrow$ strong slowdown.
* approximate $Q_\theta(s, a) \approx \phi(a, \mu_\theta(s))$, e. g. quadratic function.

Problem with «continuous Q-learning»

The only possible architecture of Q-network:



Problem: we require $\underset{a}{argmax}\, Q(s,a)$ for policy and target generation.

* well, maximize $Q(s,a)$ with respect to input a.
  - needs a lot of passes through the network on each step $\Rightarrow$ strong slowdown.
* approximate $Q_\theta(s,a) \approx \phi(a, \mu_\theta(s))$, e. g. quadratic function.
  - same problems as in policy gradients methods.

DDPG

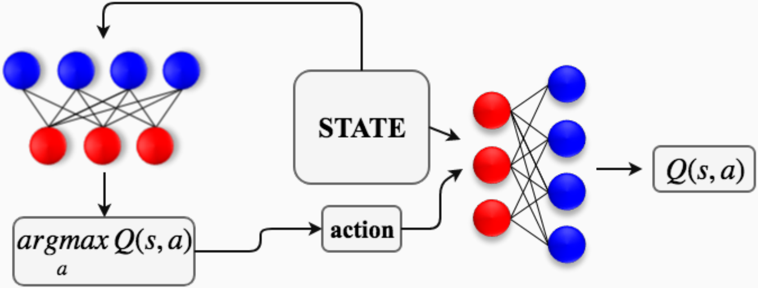
Continuous control with deep reinforcement learning (2016):

Use one more neural net to approximate $\underset{a}{argmax}\, Q(s, a)$!

DDPG

Continuous control with deep reinforcement learning (2016):

Use one more neural net to approximate $\underset{a}{argmax}\, Q(s, a)$!



DDPG: Training

* new neural network is used every time we need $\underset{a}{argmax}\, Q(s, a)$!
* very similar to actor-critic models!

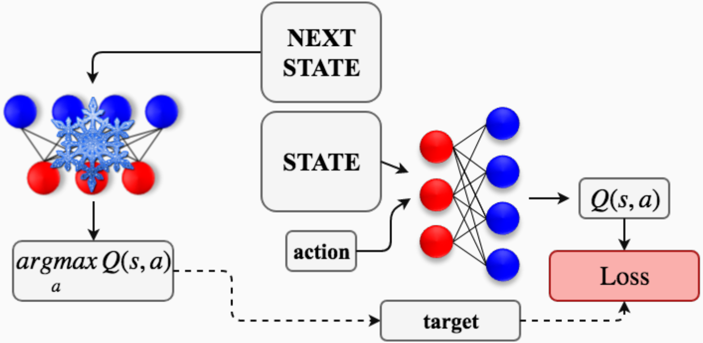
DDPG: Training

* new neural network is used every time we need $\underset{a}{argmax}\, Q(s, a)$!
* very similar to actor-critic models!

DDPG: Training

* new neural network is used every time we need $\underset{a}{argmax}\, Q(s, a)$!
* very similar to actor-critic models!

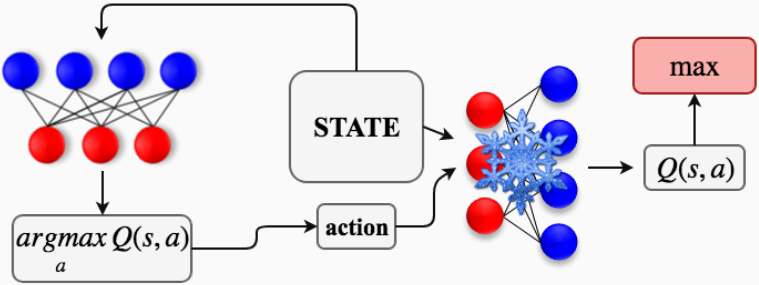
On each step of optimization target for Q-network (critic) is generated using actor.



DDPG: Training

* new neural network is used every time we need $\underset{a}{argmax}\, Q(s, a)$!
* very similar to actor-critic models!

Actor is trained to maximize $Q(s, a)$ (critic) by input.



DDPG: Resume

+ number of network passes stays the same!
    - though now we have two networks.

DDPG: Resume

- $+$ number of network passes stays the same!
  - \- though now we have two networks.
- $+$ can be extended on policy gradient methods! [1]
  - \- though the policy becomes deterministic.

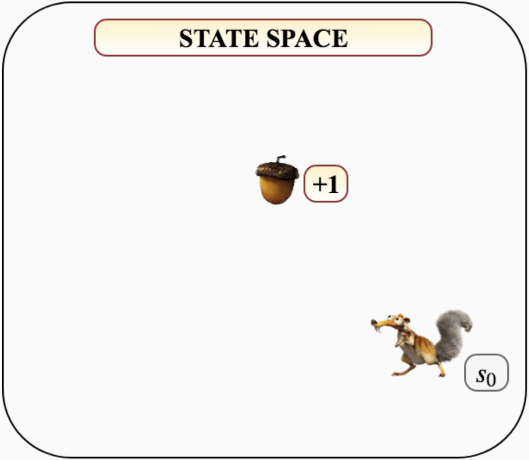
[1] how?

Hindsight Experience Replay (2018)

Problem of sparse rewards

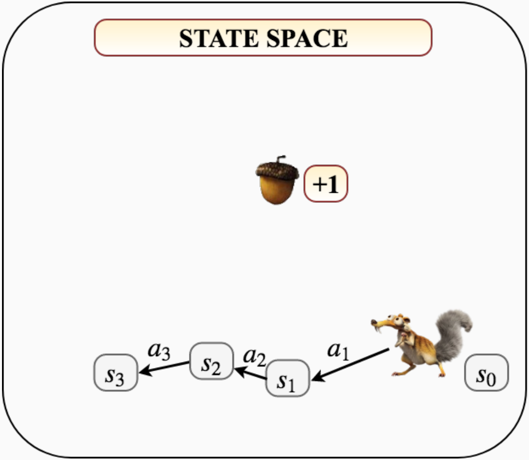
Consider the extreme sparse rewards case:

- $+1$ reward for achieving the goal, 0 otherwise
- game ends after achieving the goal



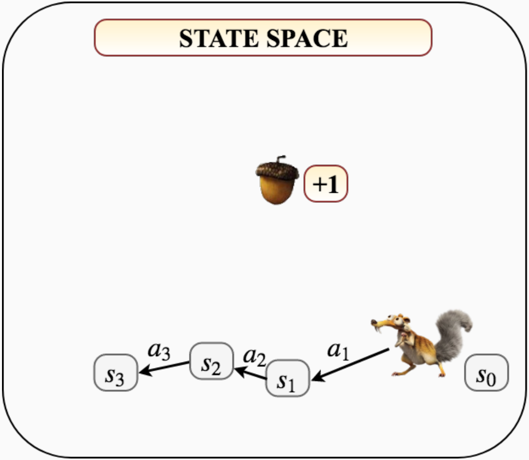
Problem of sparse rewards

At the beginning of learning, agent usually behaves randomly and can never achieve the goal.



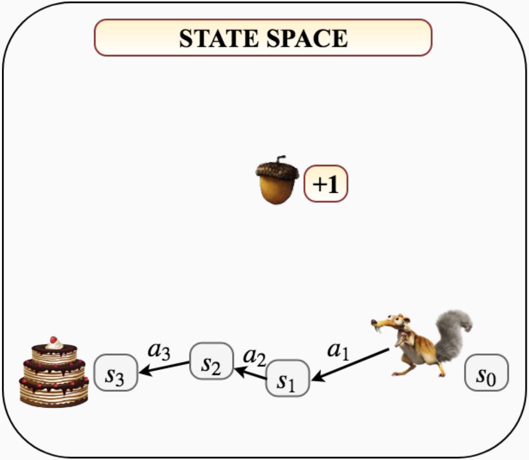
Problem of sparse rewards

Usually game session is terminated after some time and restarts. Having obtained nothing but zero reward the agent can't learn anything at all.



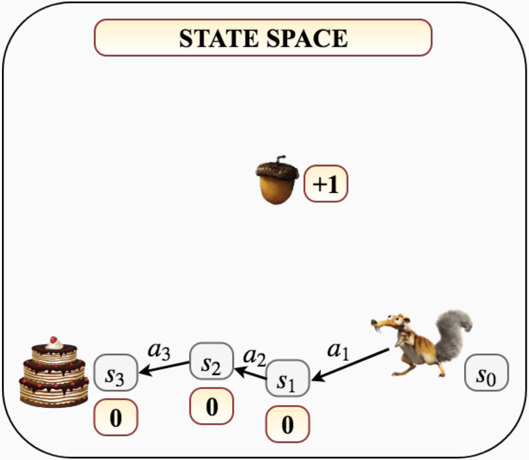
Hindsight Experience Replay

Suppose that in the state we terminated in there was another goal.



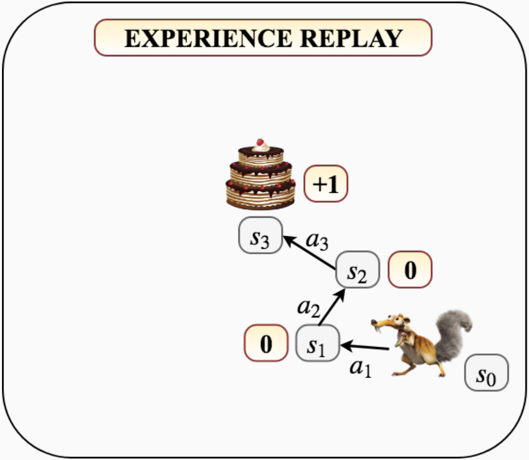
Hindsight Experience Replay

This goal was undesired, so we still receive 0 rewards.



Hindsight Experience Replay

But we may store in experience replay, that it was desired.



HER: resume

Hindsight Experience Replay (2018):

Pretend that reached goal was desired from the very beginning.

HER: resume

Hindsight Experience Replay (2018):

> *Pretend* that reached goal was desired from the very beginning.

* Concatenation of current state and goal state is given to agent as input.

HER: resume

Hindsight Experience Replay (2018):

Pretend that reached goal was desired from the very beginning.

* Concatenation of current state and goal state is given to agent as input.
* During simulations agent tries to reach true goal state [2].

[2]what if we do not know it's description?

HER: resume

Hindsight Experience Replay (2018):

Pretend that reached goal was desired from the very beginning.

* Concatenation of current state and goal state is given to agent as input.
* During simulations agent tries to reach true goal state [2].

+ Allows agent to learn navigation in state space without reward supervision.
+ "Learning from own mistakes".

[2] what if we do not know it's description?

NEXT: ?