Reinforcement learning (RL) is a framework for solving sequential decision problems where an agent interacts with its environment and adapts its policy based on a reward signal. We present an RL algorithm respecting two main requirements while being most data efficient possible:

1. dealing with continuous action and state spaces,
2. knowledge added by the designer to the agent should be minimal.

### Contributions

Inspired by Fitted Q Iteration (FQI) [1] and Deep Deterministic Policy Gradients (DDPG) [2], we formulated a new off-policy, non-linear, off-line, model-free, actor-critic algorithm. Unlike FQI, it deals with continuous action and state spaces and performs better than DDPG on three experimental environments.

### Algorithm

**Data:** $D$ replay buffer of $N$ samples, $Q_k$ value-function, $\pi_k$ previous policies, $K$ number of fitted iteration, $G$ number of gradient descent for actor updates, inverting gradient strategy, reset_critic strategy

```
for $k \leftarrow 1$ to $K$ do
  for $(s_t, a_t, r_{t+1}, s_{t+1}) \in D$ do
    $Q_{k+1} \leftarrow r_{t+1}$
    if $s_{t+1} \notin S^*$ then
      $Q_{k+1} \leftarrow Q_{k+1} + \gamma Q_{k-1}(s_{t+1}, \pi_{k-1}(s_{t+1}))$
    end
  end
  $Q_k \leftarrow Q_{k-1}$
  if reset_critic then
    $Q_k \leftarrow$ randomly initialize critic network
  end
  Update critic by minimizing the loss:
  $$\frac{1}{N} \sum_{i=1}^{N} \min(1, \frac{\pi_{k-1}(a_i | s_i)}{\pi_k(a_i | s_i)}) \left( Q_{k+1} - Q_k(s_t, a_t) \right)^2$$
  Randomly initialize actor network $\pi_k$

for $g \leftarrow 1$ to $G$ do
  Update the actor policy using the batch gradient over $D$:
  If inverting gradient then
  $$\nabla_a = \nabla_{\pi_k} \left\{ \begin{array}{ll} a_{max} - a & \text{if } \nabla_a < 0 \\ a_{max} - a_{min} & \text{otherwise} \end{array} \right. \nabla_a$$
  end
  $$\nabla_{\pi_k} \pi_k = \frac{1}{N} \sum_{i=1}^{N} \nabla_a Q(s_t, a)_{a=\pi_k(s_t)} \nabla_{\pi_k} \pi_k(s_t)$$
end
```

### Results

Median and quartile of the best registered performance in Acrobat (lower better) and Cartpole (higher better) environment during RL learning.

### References


### Future Directions

In order to increase data efficiency, it should be analyzed if a First-In First-Out (FIFO) queue is the best choice for $D$. Slowing down the change in the policy might increase his stability [5]. Finally, a better exploration strategy, that also take into account the previous collected data, is another possibility to deepen.

### Source Code

The source code and data are available at:

drl.gforge.inria.fr