Hindsight Experience Replay

Marcin Andrychowicz, Filip Wolski, Alex Ray et al.

Link to the paper: https://arxiv.org/abs/1707.01495

Note: this paper is briefly referenced in the course when presenting TDM:

https://www.tu-chemnitz.de/informatik/KI/edu/deeprl/notes/notes/4.2-LearnedModels.html

1 Abstract (3 pts)

Summarize the most important aspects of the article with your own words (max. 200 words).

Answer

Reinforcement learning uses trial-and-error to find a policy that maximizes the discounted sum of rewards obtained by an agent in the long-term. When the reward distribution is sparse, it is rather unlikely that an initially random exploration policy discovers interesting actions by chance, slowing down the speed of learning by lack of useful feedback. When using a goal-conditioned RL algorithm that tries to achieve an arbitrarily specified goal instead of using a shaped reward function, learning can be vastly improved by forcing the agent to learn from mistakes all the time, simply through replacing the initial goal with the achieved state. Hindsight Experience Replay (HER) is a simple method which can be applied to any off-policy algorithm (here DDPG) and is tested in the paper on a simulated robotic arm performing pushing, sliding and pick-and-place tasks. HER enables learning on complex tasks that DDPG alone cannot solve, even when only one goal is actually interesting and a single shaped reward function could have been designed.

2 Basic RL

2.1 Dense and sparse rewards (2 pts)

Explain the difference between dense and sparse rewards and provide examples of RL problems having both kinds of rewards.

Expected length: 3 or 4 sentences.

Answer

Dense reward have non-uniform (for example non-zero) values for most transitions in the MDP. When training a robot to go forward, the reward is defined as the velocity, which can vary between all time steps.

Sparse rewards are only occasionally different from the baseline, typically at the end of an episode. In the game of Go, the reward +1 for winning / -1 for losing is received at the end of game; all other transitions receive a reward of 0.

2.2 Optimal Bellman equations (2 pts)

RL, section 2.1, last sentence:

> It is easy to show that it satisfies the following equation called the Bellman equation.

It was not really that easy in the lectures… Explain why the optimal Q-function uses a maximum over the next Q-values and apparently does not depend on the policy.

Expected length: 2 or 3 sentences.

Answer

The optimal policy is greedy with respect to the optimal Q-values, i.e. it always select the action(s) with the highest Q* value. It is therefore deterministic. The Q-value of the next action taken by the optimal policy is always the maximum one, so this optimal Bellman equation actually depends on the optimal policy, it is just implied by the max operator.

2.3 Replay buffer (1 pt)

DQN, section 2.2:

> The transition tuples (s_t, a_t, r_t, s_{t+1}) encountered during training are stored in the so-called replay buffer. The generation of new episodes is interleaved with neural network training.

Why cannot they be used one-by-one to directly train the neural network when interacting with the environment?

Expected length: 1 or 2 sentences.

Answer

NN only work well when using stochastic gradient descent, i.e. by selecting random minibatches of i.i.d samples. If the samples are fed one by one (batch size of 1), they are going to be correlated with each other (consecutive video frames are not very different), what would impair learning.

2.4 Semi-gradient (3 pts)

DQN, section 2.2, footnote:

> The targets y_t depend on the network parameters but this dependency is ignored during back-propagation.

Explain what this means and how you can ignore this during backpropagation.

Expected length: 3 or 4 sentences.

Answer

The loss function depends twice on the output of the DQN network, once for the prediction $Q_\theta(s_t, a_t)$ and once for the greedy next action $Q_\theta(s_{t+1}, a)$. When computing the gradient with respect to the weights θ, one should theoretically differentiate twice. In practice, the Bellmann target y_t is not computed by the main network with weights θ, but by the target network θ'. The Bellman target therefore does not depend on the weights θ (as if it were constant), one only needs to differentiate $Q_\theta(s_t, a_t)$ once.

3 Multi-goal / Goal-conditioned RL

3.1 Goal-conditioned RL (1 pt)

Propose a practical application where goal-conditioned RL is necessary, as a single reward function cannot be designed (e.g. in robotics). It must be different from the examples in the paper or in the course.

Expected length: 1 or 2 sentences.

Answer

A self-driving car that automatically choses an empty spot in a parking lot. A service robot in a factory that needs to put items on a shelf. etc.

3.2 Limitations (2 pts)

Name an application where goal-conditioned RL **cannot** be applied and explain precisely why.

Expected length: 2 or 3 sentences.

Answer

Some problems do not have unique goal states. When you want to train a humanoid robot to go forward, the exact state (joint angles, position in the room) does not matter, only the forward velocity.

3.3 UVFA (2 pts)

UVFA, section 2.4: Which structure should the DQN network have in order to learn the proposed Q-function when playing Atari games? Specify the inputs and outputs.

Expected length: 2 or 3 sentences.

Answer

Input: 4 last frames (state) plus one frame for the goal. Output: one Q-value per available action. Otherwise it can be the usual CNN.

4 Hindsight Experience Replay

4.1 Experience replay (2 pts)

Explain the main difference between prioritized experience replay and hindsight experience replay. Why can they be used together?

Expected length: 2 or 3 sentences.

Answer

In PER, transitions are sampled based on how "hard" they are (absolute value of the TD error). In HER, transition tuples are created artificially by varying the goal. HER does not deal with sampling the replay buffer, only with how to fill it, so it can be used in combination with PER.

4.2 Shaped rewards (1 pt)

A shaped reward function defines the reward as the remaining distance to a goal $r_g(s, a) = -|s - g|$ according to some metric (e.g. Euclidian). Although this provide a dense (i.e. at each time step) feedback signal for the learning agent, it cannot apply to all RL problems. Provide an example (other than bit-flipping) where such shaped rewards are not likely to help.

Expected length: 2 or 3 sentences.

Answer

The metric between a goal state and any other state $|s - g|$ can be hard to compute (highly dimensional spaces such as images) or poorly informative: in video games, the goal could be to see the "you won" frame, but the pixel-based Euclidian distance between this goal and any other frame does not tell you whether you are close to winning or not.

4.3 Final strategy (1 pt)

In HER, an episode $(s_0, \dots, s_T)$ of length T is first executed with a goal g. How many transitions are added to the replay buffer using the `final` strategy? Briefly justify your answer.

Expected length: 1 or 2 sentences.

Answer

$2 \times T$. T transitions with the goal g and T transitions with the goal s_T.

4.4 Markov property (2 pts)

Section 4.1: "The state of the system is represented in the MuJoCo physics engine and consists of angles and velocities of all robot joints as well as positions, rotations and velocities (linear and angular) of all objects." Does this state representation have the Markov property? What should you do if that is not the case?

Expected length: 2 or 3 sentences.

Answer

It is probably Markov depending on the simulator. If the physics engine is realistic, accelerations might also be needed, but this is usually enough. If state representations are not Markov, one should either stack enough vectors together or use a RNN somewhere in the neural network.

4.5 DQN (3 pts)

Why are the robotic experiments made with DDPG instead of DQN? Compare the two methods in terms of possible action spaces, on/off-policy, actor-critic architecture, exploration mechanisms.

Expected length: 3 or 4 sentences.

Answer

The robotic arm is controlled through continuous joint angles, so an actor-critic algorithm like DDPG is needed. DQN = discrete action spaces; DDPG = discrete and continuous action spaces. Both are off-policy. Only DDPG has an actor-critic architecture. DQN explores with ϵ-greedy over the Q-values, DDPG adds exploratory noise over the output of the actor. Both could use parameter noise, though.

4.6 PPO (1 pt)

Why couldn't they use PPO instead?

Expected length: 1 or 2 sentences.

Answer

PPO is on-policy, it does not work with a replay buffer.

5 Miscellaneous

Various questions on unrelated parts of the course. Just one example here.

5.1 On- and off-policy methods (3 pts)

Summarize the relative advantages and drawbacks of on-policy and off-policy algorithms.

Expected length: 4 or 5 sentences.

Answer

On-policy methods follow the learned policy, so they form less biased estimates of the value functions (which should be expectations under the current policy) and find better policies. However, they constantly need new samples (no replay buffer) so they have a higher sample complexity. Off-policy methods benefit from the

use of a replay buffer as they can learn from past transitions (or even expert knowledge), but, unless we use importance sampling, they form biased estimated and often converge to suboptimal policies.