Q-Learning: randomized rewards

Valentin Samokhin, Daniil Merkulov
Optimization Class Project. MIPT

Introduction

The Reinforcement Learning (RL) is a quite old branch of learning based on interaction between the environment and the agent. The agent makes actions, receives reward, and makes a step according to some policy. The task of the RL is to find out the optimal policy that provides the biggest cumulative reward. There are two main approaches in RL - model-based and model-free. The latter one is considered by some specialists to be more simple in sense of memory consumption and knowledge about the environment. The most difficult problems that algorithms should solve is the exploitation-exploration trade-off [1] and the credit assignment problem. Instead of introducing arbitrariness into action choice we add Gaussian noise to the reward received by the agent. Thus we want to study the possible effect the noise[1] can bring into the results. To our disappointment, the majority of existing papers are not supported by any numerical experiments. We want to set them.

Value (V-) and action-value (Q-) functions

Let us intrduce *the Value function*[2]:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_i \gamma^i r_i(s_i, a_i^{\pi}) \mid s_0 = s, \pi\right] \tag{1}$$

It is the expectation of the sum of discounted rewards, received by the agents in case it follows the policy π [1] starting from the state s.

Then we should introduce *the Q-function* [3]. It, given the state s and action a, returns the expected cumulative reward in case of folowing the policy π

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t \mid s_0 = s, a_0 = a, \pi\right] = \\ = r_0(s,a) + \gamma \mathbb{E}\left[V^{\pi}(s_1{}^2, a)\right] \tag{2}$$

It makes sence, if we suppose that interaction between agent and environment is a Markov Decision Process. This assumption allows us not to take into the account the previous observations.

Optimality theorem[4] states that for the optimal policy $\pi^{\star}$ the following equation takes place.

$$V^{\star} = max_a Q^{\star}(s,a) \tag{3}$$

If we assume that the state and action spaces, and the horizon, than we can claim that the optimal policy always exists. This equation and the recursive definition of Q-function (2) make possible the introduction of the various iterative schemes (like (4))that update the Q-values step-by-step.

$$Q_{i+1}(s,a) = \mathbb{E}\left[r + \gamma max_{a'} Q_i(s', a') \mid s, a\right] \tag{4}$$

Algorithm

The main algorithm we want to study here is a variant of Q-learning that incorporates UCB exploration.

1: Initialize $Q_h(s,a) \leftarrow H$ and $N_h(s,a) \leftarrow 0 \; \forall (s,a,h) \in S \times A \times [H]$
2: **for** episode $k = 1, .., K$ **do**
3: Receive x_1
4: **for** step $h = 1, \ldots, H$ **do**
5: Take action $a_h \leftarrow \mathrm{argmax}_{a'} Q_h(x_h, a')$, and observe x_{h+1}
6: $t = N_h(s_h, a_h) \leftarrow N_h(s_h, a_h) + 1; b_t \leftarrow c\sqrt{H^3\iota/t}$
7: $Q_h(s_h, a_h) \leftarrow$
$(1-\alpha_t)Q_h(s_h, a_h) + \alpha_t[r_h(s_h, a_h) + \mathcal{N}(0,1) + V_{h+1}(s_{h+1}) + b_t]$.
$V_h(s_h) \leftarrow min\{H, \max_{a' \in A} Q_h(s_h, a')\}$.
8: **end for**
9: **end for**

Algorithm 1: Q-learning with UCB-Hoeffding [1]

Here ι is a log-factor: $\iota := \log(SAT/p)$

We would compare this approach with ϵ-greedy policy without bonus: instead of choosing the best action from the Q-table (as in step 5 of the algorithm 2), we would balance between exploitation and exploration with ϵ probability of choosing exploration. We would also try to add some small noise to the reward in the UCB-algorithm (step 7).

1: Initialize $Q_h(s,a) \leftarrow H$ and $N_h(s,a) \leftarrow 0 \; \forall (s,a,h) \in S \times A \times [H]$
2: **for** episode $k = 1, .., K$ **do**
3: Receive x_1
4: **for** step $h = 1, \ldots, H$ **do**
5: Sample action $a_h \leftarrow \mathrm{argmax}_{a'} Q_h(x_h, a')$ with $p = 1 - \epsilon$ or chose any other action with $p = \epsilon$, and observe x_{h+1}
6: $t = N_h(s_h, a_h) \leftarrow N_h(s_h, a_h) + 1; b_t \leftarrow c\sqrt{H^3\iota/t}$
7: $Q_h(s_h, a_h) \leftarrow (1-\alpha_t)Q_h(s_h, a_h) + \alpha_t[r_h(s_h, a_h) + V_{h+1}(s_{h+1}) + b_t]$.
$V_h(s_h) \leftarrow min\{H, \max_{a' \in A} Q_h(s_h, a')\}$.
8: **end for**
9: **end for**

Algorithm 2: Q-learning with ϵ-greedy exploration [1]

Numerical example

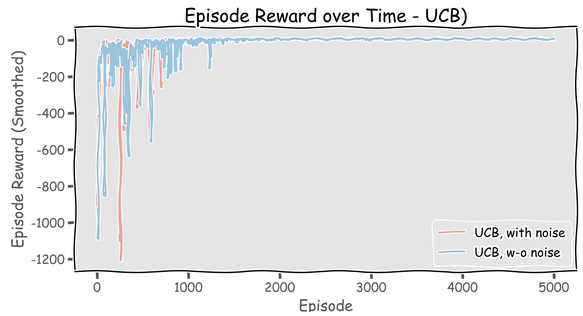
To get numerical results we use the environment from Open-AI gym package "Taxi-v2" [5]. It has small finite action and observation spaces and allows user to select the arbitrary initial state.

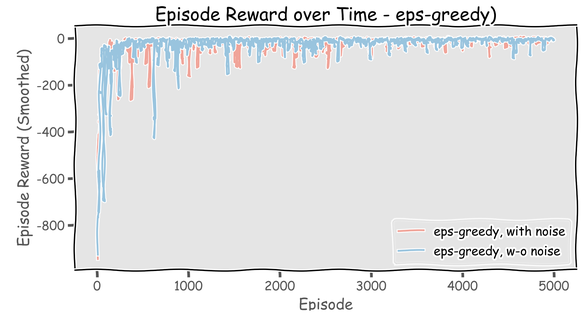
> There are 4 locations (labeled by different letters) and your job is to pick up the passenger at one location and drop him off in another. You receive +20 points for a successful dropoff, and lose 1 point for every timestep it takes. There is also a 10 point penalty for illegal pick-up and drop-off actions.

The second environment to study the approaches is the discrete variant of CartPole problem. This classic control problem allows to choose discrete approximation as we wish. To do it precisely, the observation space should be quite big, what sets a more difficult task than the previous one.

Results

The results show, that UCB-exploration finds the optimal policy quicker than the ϵ-greedy algorithm for Taxi problem. The addition of noise makes the curve of the episode reward smoother for both policies.





The algorithm worked poorely on CartPole problem, due to the lack of assumptions required for the algorithm.

Conclusion

We showed that in some cases bonus and noise addition may help to improve the exploration-exploitation trade-off, giving better observed episode rewards. However, there is strong dependence on algorithm success and the ability to choose the initial state arbitrary. In the Taxi case, fine tuning on parameters of the algorithm improved results, reducing the length of the episode. One can try to study the impact the random noise can have in Deep Q-Learning. The tasks, usually solved by such approach (e.g. Atari games [6]), incorporate enough arbitrariness in the initial state.

References

[1] Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. Is q-learning provably efficient? In *Advances in Neural Information Processing Systems*, pages 4868–4878, 2018.

[2] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[3] Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

[4] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[5] Greg Brockman et al. Openai gym. *CoRR*, abs/1606.01540, 2016.

[6] Volodymyr Mnih et al. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.