Deep Reinforcement Learning: A study of the CartPole problem

Ádám Budai, Kristóf Csorba

Abstract: One of the major challenges of artificial intelligence is to learn solving tasks which are considered to be challenging for even a human. Reinforcement learning is the most general learning framework among the three main type of learning methods: supervised, unsupervised and reinforcement learning. Most of the problems can easily fit into this framework. Experience shows that a lot of machine learning methods with non-linear function approximators suffers from instability regarding convergence. Reinforcement learning is more prone to diverge due to its ability to change the structure of its training data by modifying the way how it interacts with the environment. In this paper we investigate the divergence issue of DQN on the CartPole problem in terms of the algorithm's parameters. Instead of the usual approach we do not focus on the successful trainings but instead we focus on the dark side where the algorithm fails on such an easy problem like CartPole. The motivation is to gain some further insight into the nature of the divergence issues on a specific problem.

Keywords: reinforcement learning, ALE, CartPole, OpenAI gym, DQN

Introduction

Reinforcement learning (herafter RL) becomes more general by involving interactions into the learning process which makes RL an active learning method. Supervised and unsupervised learning use training data generated independently of the learning algorithm. But an RL agent generates the training data (samples) for itself. Fig. 1 shows the components of the RL framework. The agent is the manifestation of the learning algorithm and it provides an interface to interact with the environment through actions. The environment is described by its state. If we knew the behavior of the environment then we would forecast the next state by knowing the current one. The agent receives a feedback in the form of a reward indicating the success of its action. The objective of reinforcement learning is to find a methodology for choosing actions to gather as many rewards as possible [1, 2].
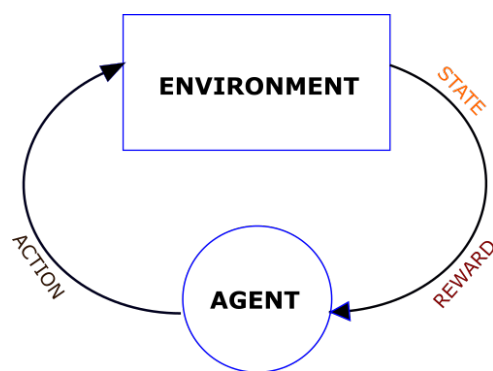

Figure 1: The components of machine learning.

A key concept in the traditional foundation of RL is the so called action-value function (Q):

$$Q(s,a) = E_\tau\big[G(\tau), s_0 = s, a_0 = a\big] \tag{11}$$

$$G(\tau) = \sum_{(s,a)\in\tau} r(s,a,s'). \tag{12}$$

The action-value function shows the value of the current state in terms of the expected return (G) by following trajectories started from s and make action a at first time. If the optimal action-value function is known then the function to choose the next action (policy) is given as:

$$\pi(s) = \arg\max_a Q(s, a). \tag{13}$$

The main purpose in RL is to find the optimal policy. If the optimal action-value function is given then the optimal policy is given by Eq. 13. If the state has high dimension (like an image) then storing the Q function in memory for each state is impossible. Therefore function approximators, like neural networks, are used to represent them. However, training a deep neural network in the RL framework was known to divergent. The DQN algorithm [5] was a breakthrough when it was able to learn a wide range of Atari games [11] from the raw sensory inputs (the image of the playing area) with the same hyper–parameters. After this breakthrough it seems considerable for us that solving the CartPole problem with DQN does not require a lot of parameter tuning but in reality the behavior of the algorithm sensitive for the parameters. In Section the DQN algorithm is introduced. Section introduces the CartPole problem and benchmarking tools to make comparisons with other algorithms. In Section the measured results are summarized after parameter scanning our DQN implementation (available on GitHub [13]). Section discusses conclusion thoughts.

Background

DQN (Deep Q-Network) algorithm [5, 6] is a Q-learning based algorithm where the action-value function (Q-learning) is represented by a neural network. DQN uses two techniques in order to address the issue of convergence. 1) It applies the so called experience replay [8]. It has a significant impact on the learning because the samples (experiences, which are the tuple of state, action, reward and next state, a cycle in Fig. 1) are stored in a buffer. Therefore it can be reusable more then once. This is important in RL where the samples are gather during the interactions with the environment but some part of the environment is difficult to visit, so samples from that part are rare. Throwing them away is wasteful. On the other hand drawing the training samples uniformly from the experience replay to feed into the Q-network breaks the correlations between the samples. This makes the learning process more stable [9]. 2) The so called iterative (or delayed) update uses a copy of the Q-network but it is frozen for a while across bootstrap updates Eq. 14.

$$\Theta_{t+1} = \Theta_t - \alpha \cdot \left(Q(s_t, a_t, \Theta_t) - (r_t + \gamma \max_{a'} Q(s', a', \Theta^-)) \right) \frac{\partial Q(s_t, a_t, \Theta_t)}{\partial \Theta_t}, \tag{14}$$

where Θ^- are the frozen weights and it is updated after more steps (in $t + k$ for a well-chosen k), γ is the discounting factor between 0 and 1, r_t is the immediate reward. The usage of experience replay requires an off-policy learning algorithm like Q-learning. Without this the samples should be used sequentially (in the sequence they were sampled from the environment) for updates, otherwise convergence is not ensured even in the case of a linear-approximator. Unfortunately, off-policy algorithms more proned to divergence then on-policy ones. In case of DQN it is ridiculous that experience replay is able to balance this out. The authors claim that in the future developing stable off-policy algorithms is essential for scalable RL which is the base for real life applications. The intuition behind this is that off-policy algorithms make possible to use the same experience (sample) for train different agents parallel. Therefore we investigates DQN. In recent years more versions of DQN were developed [7] but in this paper we implemented the original version because it has many available benchmarks to compare.

Benchmarking tools

When newer and newer algorithms are developed it is vital to establish testbeds in order to compare the results of new algorithms to older ones. In RL, OpenAI gym is a popular [3] tool. Especially the Atari-games [11] which are based on the Arcade Learning Environment [4]. The Atari games are fairly complicated to solve but the training is time consuming and does not appropriate for parameter scanning. Therefore a simpler problem was necessary which has the same characteristic (the state is an image about the playing area) as the Atari in OpenAI gym. Classic control problems, like CartPole, is suitable for that. Unfortunately, the original implementation does not support giving back images as states, therefore we changed the implementation in OpenAI gym to get rid of this insufficiency [12]. The CartPole problem: "A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled by applying a force of $+1$ or -1 to the cart. The pendulum starts upright, and the goal is to prevent it from falling over [10]".
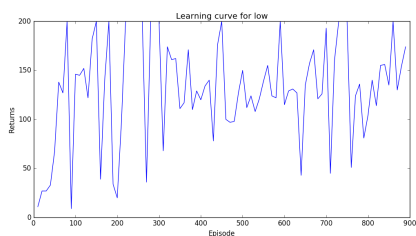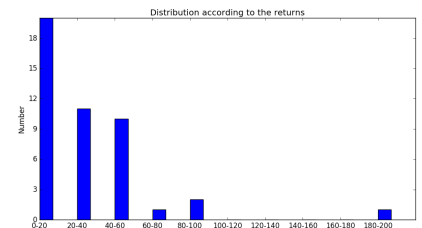
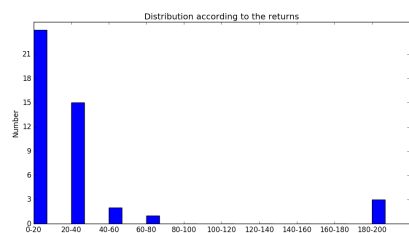

(a) Learning curve for low dimensional input.
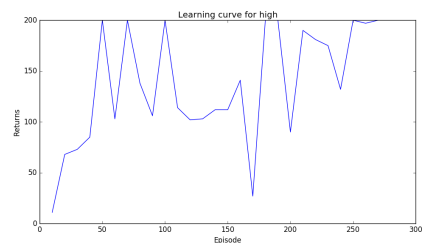
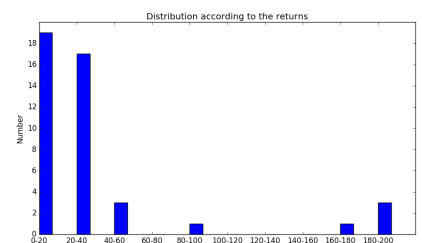(b) Learning curve for high dimensional input.
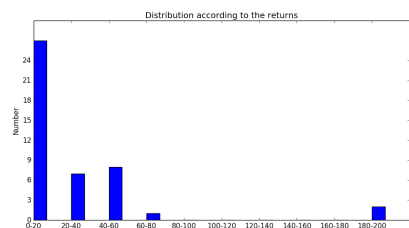
(c) Distrib. of the returns for low dim. input.

(d) Distrib. of the returns for low dim. input.

(e) Distrib. of the returns for high dim. input.

(f) Distrib. of the returns for high dim. input.

Figure 2: The first row shows the learning curves for the low and the high dimensional cases for the best learners. The second and the third row shows the returns achieved by the algorithm on different parameters and organized according to bins with width 20.

Experimental results

Fig. 2 summarizes the results. It can be seen that most of the parameters were wrong to achieve good performance. Among 46 experiments only 2-3 were able to solve the task. The

two columns in the second and third rows are runs of the same algorithm. The parameter table available at [13]. The best parameters for high dimensional inputs is in the 13rd row while for the low case the 14th row in the corresponding files. The corresponding parameters not at the extreme. Deeper, wider networks do not performs better by definition. The result strongly depends on the chosen activation too but as the best algorithms exemplifies [10], both relu and tanh can be successful. Fig. 2 makes the impression that high dimensions are easier to solve. We encourage interested readers to play with the code.

Conclusions

The results show how difficult to find a right parameter settings for this easy problem and the results can vary across executions as well. But regarding the fact that off-policy learning is so important it is worthy to find methods to stabilize it. Furthermore achieving stable, robust results on problems like CartPole can be significant as well.

Acknowledgement

References

[1] R. Sutton, Introduction: The Challenge of Reinforcement Learning, *Machine Learning*, Vol. 8, pages 225–227, 1992.

[2] R. Sutton and A. Barto, Reinforcement Learning: An Introduction, *MIT Press*, 2018

[3] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang and W. Zaremba. OpenAI Gym, *CoRR*, arXiv:1606.01540, 2016.

[4] M. G. Bellemare, Y. Naddaf, J. Veness and M. Bowling. The Arcade Learning Environment: An Evaluation Platform for General Agents, *Journal of Artificial Intelligence Research*, Vol. 47, pages 253-279, 2013.

[5] V. Mnih et. al., Playing Atari with Deep Reinforcement Learning, *CoRR*, Vol. abs/1312.5602, 2013.

[6] V. Mnih et. al., Human-level control through deep reinforcement learning, *Nature*, Vol. 518, doi:10.1038/nature14236, 2015.

[7] Z. Wang et. al., Dueling Network Architectures for Deep Reinforcement Learning, *CoRR*, Vol. abs/1511.06581, 2015.

[8] L. Lin, Self-Improving Reactive Agents Based On Reinforcement Learning, Planning and Teaching, *Machine Learning*, Vol. 8, pages 293–321, 1992.

[9] Y. LeCun et. al., Gradient-Based Learning Applied to Document Recognition, *Proc, of the IEEE*, 1998.

[10] OpenAI gym leader board, https://github.com/openai/gym/wiki/Leaderboard.

[11] Atari games, https://en.wikipedia.org/wiki/Atari_Games.

[12] OpenAI gym with new CartPole environment, https://github.com/adamtiger/gym/tree/openai-goal-based-atari.

[13] The implementation of DQN for the CartPole problem, https://github.com/adamtiger/CartpoleCSCS.