Implementation Report: Project 2 Continuous Control

Deep Reinforcement Learning Nanodegree

Thomas Gallien

October 7, 2019

Problem Description

The objective is to train a deep reinforcement agent being capable to control a multi-body model to follow a moving target. To solve the task two different versions of Unity environments are available instantiating a single or 20 double-jointed arms, respectively. The solution reported here deals with 20 agent environment.

Environment

The environment consists of 20 instances of a double-jointed arm being controlled in order to follow a moving target indicated by a green sphere. Figure 1 illustrates the configuration.

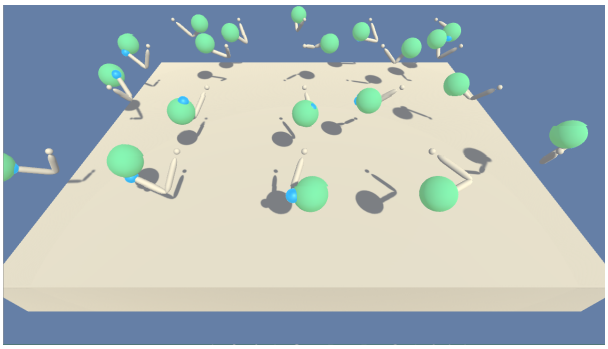

Figure 1: Unity's reacher environment utilizing 20 double-jointed arms. The corresponding target location is highlighted by the green sphere.

Observation Space

For each instance of the two-joint arm a 33-dimensional observation vector can be observed corresponding to position, rotation, velocity, and angular velocities of the arm.

Action Space

The action space is 4-dimensional for each instance and consists of and corresponds to the the normalized torque (numbers between -1 and 1) that is to be applied to each of the two joints.

Reward

A reward of 0.1 is returned if the tip of the double-jointed arm is the target location. Otherwise the agent does not get reward.

Goal

The environment is considered solved, when the average (over 100 episodes) of those average scores (average over all 20 instances) is at least 30.

Training Algorithm

For training the agent the Deep Deterministic Policy Gradient(DDPG) approach (see Lillicrap et.al. [1] for details) was used. Essentially, the core implementation is based on an example used in class. However, since the agent interacts with 20 instances of the double-jointed arm simultaneously, the experiences (states, rewards, done flags) of all instances are stored in one single experience buffer (shared experience buffer). As a further expansion gradient clipping is used train the critic network. This measure significantly improved the performance of the algorithm.

Architecture

The architecture of the actor and critic networks was chosen according to the supplementary information shown in the DDPG paper [1].

Actor Network

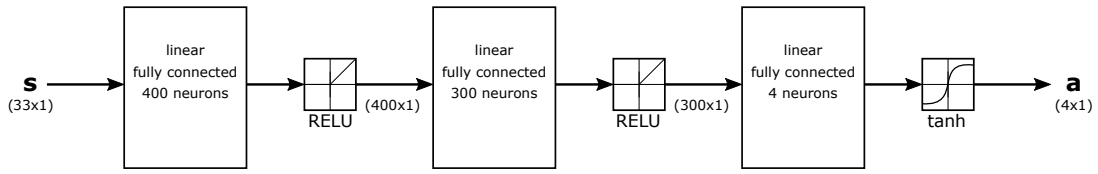

Figure 2: Forward path of the agent's fully connected actor network(s).

Critic Network

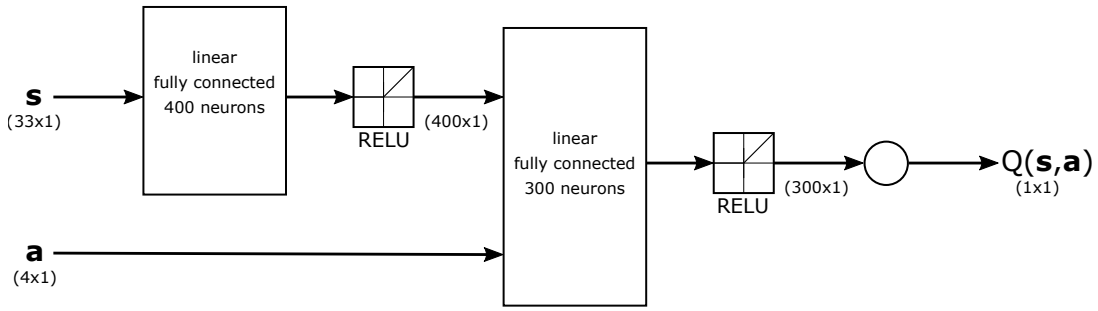

Figure 3: Forward path of the agent's fully connected critic network(s).

Hyperparameter

The table below shows the set of hyperparameter used to solve the 20 instances Reacher environment. Similar to the network architectures, the majority of the hyperparameters were chosen in accordance with the supplementary information shown in the DDPG paper [1]. However, using L2 regularization for the optimization of the critic did not reveal in an increased performance. Hence, the weight decay parameter was set to 0. Moreover, the size of the replay buffer was reduced by a factor of 10 and the size of the mini-batches was doubled to 128.

Hyperparameter	Abbreviation	Value
Buffer size replay memory	M	10^5
Batch size	N	128
Discount factor	γ	0.99
Learn rate actor network	α_{actor}	10^{-3}
Learn rate critic network	α_{critic}	10^{-4}
Interpolation parameter soft update	τ	10^{-3}
Update rate	l	1
Decay rate UO-noise	θ	0.15
Standard deviation UO-noise	σ	0.2

Table 1: Used set of hyperparameter solving the 20 instances environment utilizing a DDPG agent.

Results

The figure below shows the scores calculated by averaging the individual rewards across all instances of the double-jointed arm during training (blue line) and the corresponding moving average with respect to 100 filter taps. As can be easily seen, the agent approaches the desired number +30 of the average score already after 68 periods. Moreover, after about 18 episodes the score stabilizes at approximately 38. This indicates that the training essentially succeeded already at this time.

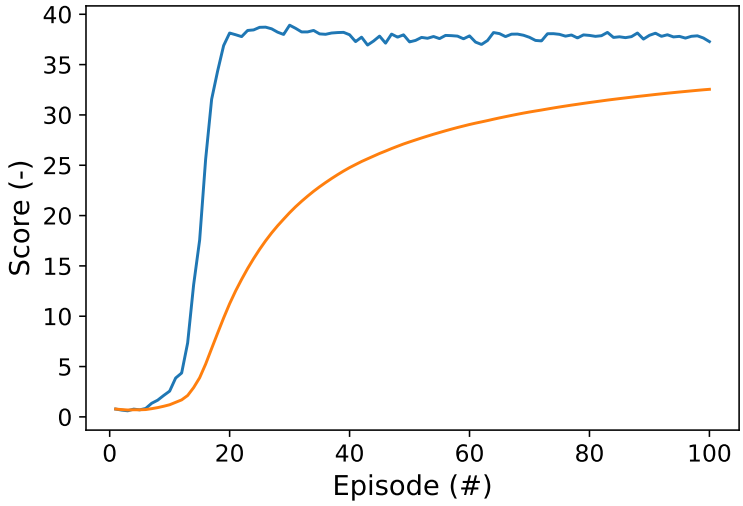

Figure 4: Performance of the DDPG agent in terms of episodic cumulative reward (score, blue line) for 100 episodes and corresponding moving average filtered signal (red line). In order to access the performance, the average reward was calculated across all instances of the environment.

Conclusion & Outlook

The DDPG algorithm utilizing the used set of hyperparameter achieved fairly good performance in solving the 20 instances Reacher environment. However, in a first attempt the proposed algorithm was used to solve the single instance Reacher environment and failed to converge to an useful configuration. This essentially indicates a design flaw in the environment's reward signal, since the agent only achieves reward if the tip of the double-jointed arm is in the desired location. Hence, the probability that the agent receives exploitable information is quite little. This explains why parallelization was able to overcome this problem, since the amount of samples from the state space is significantly increased. For a real world problem, the reward signal might use an error signal encoding the distance from the current to the target location.

Nevertheless, in order to solve the single instance Reacher environment attempts using Trust Region Policy Optimization (TRPO)[3] and Truncated Natural Policy Gradient (TNPG)[4] will be made and most recent approaches such as Distributed Distributional Deterministic Policy Gradients (D4PG)[2] will be investigated.

References

[1] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel et. al. *Continuous Control with Deep Reinforcement Learning*. Proceedings of the 2016 International Conference on Learning Representation (ICLR), 2016.

[2] Gabriel Barth-Maron, Matthew W. Hoffman, David Budden et. al. *Distributed Distributional Deterministic Policy Gradients*. Proceedings of the 2016 International Conference on Learning Representation (ICLR), 2018.

[3] John Schulman, Sergey Levine, Philipp Moritz et. al. *Trust Region Policy Optimization*. Proceedings of the 2015 International Conference on Machine Learning (ICML), 2015.

[4] Sham Kakade *A natural policy gradient*. Neural Information Processing Systems 2002, pp. 1531–1538.