Describing the role of Artificial Neural Networks in Reinforcement Learning

Thilo Stegemann
University of Applied Sciences
Applied Computer Science
12459 Berlin, Wilhelminenhofstraße 75A
Email: t.stegemann@gmx.de

***Abstract*—TODO : Write an awesome abstract at the end!**

I. Introduction

Artificial Intelligence is a huge, rewarding, rising and complex research field. More and more people are interested in AI every day. Students, researchers, economics, engineers, CEO's and investors are highly encouraged to use, understand and/or improve AI technologies. At some point in time an AI newcomer will get to the problems of Reinforcement Learning (RL) and therefore to Artificial Neural Networks (ANN's). Andrew Ng. describes AI as the new upcomming electricity: AI will change many different industries and it will have a huge general impact in everyday life.

II. Related Work

A. Asynchronous Methods for Deep Reinforcement Learning

The scientists from Google DeepMind and Montreal Institute for Learning Algorithms introduced asynchronous deep learning algorithms [1]. These asynchronous algorithms are based on four standard reinforcement learning algorithms: One-step Q-learning, one-step Sarsa, n-step Q-learning and advantage actor-critic. The paper explains the background of reinforcement learning and how the asynchronous reinforcement learning methods works. The study was approved by an experiment in an Atari 2600 evaluation environment . All four asynchronous algorithms where tested within the test environment . The Atari 2600 environment tests where used to compare the performance of the four algorithms. The main finding of this study is that all four asynchronous deep reinforcement learning algorithms are able to train neural network controllers on a variety of domains in a stable manner. In addition their results show that stable train ing of neural networks through reinforcement learning is possible with both value-based and policy-based methods, off-policy as well as on-policy methods, and in discrete as well as continuous domains.

B. Deep Reinforcement Learning with Double Q-Learning

Aim of this paper is to determine if the recent DQN (Deep Q Network) algorithm, which combines Q-learning with a deep neural network, suffers from substantial overestimations in some games in the Atari 2600 domain [2]. Furthermore the Google DeepMind contributors point out how the Double Q-learning algorithm can be generalized to work with large-scale function approximation to successfully reduce the DQN overoptimism, resulting in more stable and reliable learning. Finally they propose a specific adaptation to the DQN algorithm and show that the resulting algorithm (Double DQN) not only reduces the observed overestimation, as they hypothesized, but that this also leads to much better performance on several Atari 2600 games.

C. Human-level controll through deep reinforcement learning

The paper is about how to reach human-level control through a deep Q-network, that can learn successful policies directly from high-dimensional sensory inputs using end-to-end reinforcement learning [3]. The deep Q-network agent (in reinforcement learning an agent is the executing learning algorithm) is tested on the challenging classic Atari 2600 game environment. The result of this test demonstrated that the deep Q-network agent, receiving only the pixels and the game score as inputs, was able to surpass the performance of all previous algorithms and achieve a level comparable to that of a professional human games tester across a set of 49 games, using the same algorithm, network architecture and hyperparameters.

D. Mastering the game of Go with deep neural networks and tree search

The paper from Google concerns two different algorithm approaches for deep reinforcement learning [4]. The first approach uses 'value networks' to evaluate board positions and 'policy networks' to select moves. These deep neural networks are trained by a novel combination of supervised learning from human expert games and reinforcement learning from games of self-play. They proof through experiments that this deep RL algorithm approach is capable of playing Go at the level of state-of-the-art Monte Carlo tree search. The second deep RL approach is a new seach algorithm that combines Monte Carlo simulation with value and policy networks. They used this search algorithm inside the application AlphaGo and the application achieved a 99.8% winning rate against other Go programs and it defeated the human European Go champion by 5 games to 0.

E. Playing Atari with Deep Reinforcement Learning

The paper from DeepMind Technologies is about a deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning [5]. They defined the model as a convolutional neural network (CNN). This CNN is trained with a variant of Q-learning. Input of the CNN is row pixels and output is a value function estimating future rewards. They apply this deep learning model to seven Atari 2600 games from the Arcade Learning Environment, with no adjustment of the architecture or learning algorithm. Result of this experiment is that it outperforms all previous approaches on sex ot the games and surpasses a human expert on three of them.

III. Reinforcement Learning

Reinforcement learning (RL) problems consider an agent-environment interaction framework. Basics of reinforcement learning are mentioned in [1]–[5]. As an in depth guide for reinforcement learning see [6]. The following part is about summarizing those background RL introductions. The agent (implementation of the learning algorithm) will interact with the environment (a Markov Decision Process). The agent will get rewards and states from the environment and the environment will get actions from the agent. The agent tries to learn optimal behaviour through trial and error attempts. The agent wants to know which actions in which states get the most long-term reward and fit this knowledge into a policy representation. A few main problems of this RL framework are:

- The agent only gets a numerical reward from the environment at the end of a decision-sequence.
  $\sim$ *Delayed Reward*
- How should the reward be assigned to the different steps of a decision-sequence?
  $\sim$ *Credit Assignment Problem*
- How to handle vast action- and state-spaces?
  $\sim$ *Generalization Problem*

A major goal of RL is to find a global optimal policy. A policy is a function which maps states to actions. This policy will additionally get a vector of parameters. The parameter-vector changes the policy output. This parametrisation of the policy function is called "function approximation" and Artificial Neural Networks are a really great approach for approximating a policy function. Combining ANN's with reinforcement learning algorithms have so far shown spectacular results e.g.: The Google DeepMind Team programmed an AI which plays Go (a very complex strategy board game) at human grandmaster level [4]. With this approximation the problem of vast action- and state-spaces can be solved. To optimise the parameter-vector methods like "Monte-Carlo Policy Gradient" or "Actor-Critic Policy Gradient" are applied to objective functions. Applications like TD-Gammon by Gerald Tesauro proved that learning complex strategy games with Artificial Neural Networks is possible and promising.

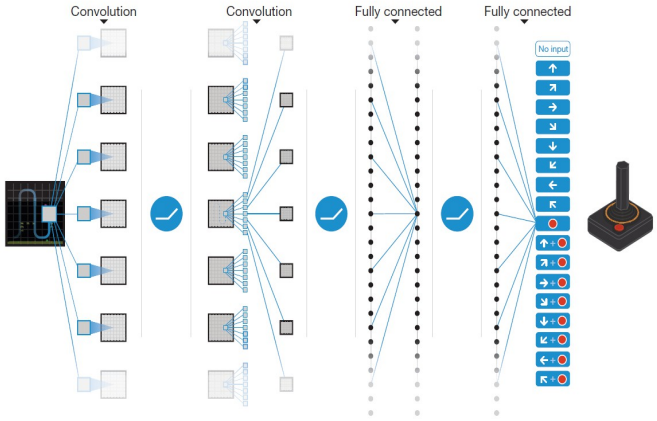

Fig. 1. Schematic illustration of the convolutional neural network [3].

A. Q-learning

B. Actor-critic approaches

IV. Convolutional Neural Networks

A convolutional neural network (CNN) consists of layers. There are different kinds of layers: input layer, convolutional layer (CONV), rectified linear unit layer (RELU), pooling layer (POOL), fully connected layer (FC) and an output layer. Some of these layers can appear multiple times. For example CONV $\rightarrow$ RELU $\rightarrow$ CONV $\rightarrow$ RELU $\rightarrow$ POOL ... FC. In neural networks in general between the output and the input layers are hidden layers. If there is more then one hidden layer between input and output layer, then the neural network is called a deep neural network. To understand the Deep Q-networks it is helpful to first think about how those convolutional neural networks work in detail.

a) The Input layer: can be high dimensional sensory data. Considering the Atari 2600 arcade gaming environment [1]–[3] the input is a video stream of pixels at different time steps. At time step t a video signal reduces to just an image of pixels and the complete sequence of images at all time steps equals the video signal. There is still more then the raw pixels from the video stream like the score value at each time step [3]. In terms of reinforcement learning the score signal is a reward signal and the video stream at each time step describes the state in which the agent is in.

b) The convolution layer: is about shifting a fixed sized window (a filter) around the whole input image. The size of the filter should be about 3x3 or 5x5 pixels small. Each time the 3x3 filter is applied on an image position it outputs exact one value for that position. A filter can be a representation for edges, lines or other shapes. A combination of those low level filters results in more complex filters like an eye or an ear. Andrej Karpathy created a great visualisation of the computation inside the convolution layer see section "Convolution Demo" in [7].

V. Deep Q-network

A deep Q network (DQN) is a combination of a convolutional neural network and the reinforcement learning algorithm Q-learning.

"A deep Q network (DQN) is a multi-layerd neural network that for a given state s outputs a vector of action values $Q(a, ; \theta)$, where θ are the parameters of the network. For an n-dimensional state space and an action space containing m actions, the neural network is a function from $\mathbb{R}^n$ to $\mathbb{R}^m$. Two important ingredients of the DQN algorithm as proposed by Mnih et al. (2015) are the use of a target network and the use of experience replay. The target network, with parameters θ^-, is the same as the online network except that its parameters are copied every t steps from the online network, so that then $\theta_t^- = \theta_t$, and kept fixed on all other steps. The target used by DQN is then

$$Y_t^{DQN} \equiv R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta_t^-).$$

For the experience replay (Lin, 1992), observed transitions are stored for some time and sampled uniformly from this memory bank to update the network. Both the target network and the experience replay dramatically improve the performance of the algorithm (Mnih et al. 2015)."

References

[1] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," *CoRR*, vol. abs/1602.01783, 2016. [Online]. Available: http://arxiv.org/abs/1602.01783

[2] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *CoRR*, vol. abs/1509.06461, 2015. [Online]. Available: http://arxiv.org/abs/1509.06461

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, and D. H. S Legg, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, 2015. [Online]. Available: https://storage.googleapis.com/deepmind-media/dqn/DQNNaturePaper.pdf

[4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," in *NIPS Deep Learning Workshop*, 2013.

[6] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[7] A. Karpathy. (2017) Cs231n convolutional neural networks for visual recognition. [Online]. Available: http://cs231n.github.io/convolutional-networks/