Deep Reinforcement Learning Beginners Tutorial Documentation

Julian Bernhart, Robin Guth

June 22, 2019

Contents

1 Introduction

Deep Reinforcement Learning is a huge step towards the creation of an universal artificial intelligence[1]. In 2013 a company, owned by Google, called "Deep Mind", was able to create an astonishing implementation of RL[2], which was capable to play a range of retro games of the console "Atari 2600". In many cases, the AI[3] was not only able to play the games successfully, but also exceeded human performances significantly [1]. After these impressive results, it is definitely worth to take a closer look at *reinforcement learning*.
Playing games is fun, but *reinforcement learning* has more to offer, as discussed in [2]. Apart from playing video games, there are many use cases in fields like robotics, traffic control or even personalized advertisement. Apart from the Atari paper [1], multiple papers have been written about RL. Some noteworthy examples are the creation of an RL algorithm, which recommends personalized news [3] or a survey about different applications for RL in robotics [4]. While supervised and unsupervised learning are already widely used in production,

[1] can be used for multiple problems without notable changes
[2] Reinforcement Learning
[3] Artificial Intelligence

reinforcement learning is still in development and further research is needed. As an interesting topic with many application possibilities, there is a demand for tutorials and hands-on guides for beginners, especially students. Beginners often struggle to find a good starting point into the world of AI and specifically RL. Many existing tutorials are written for more advanced users, who already have a deeper understanding of machine learning.
For this reason, the *"Deep Reinforcement Learning Beginners Tutorial"* was created, as an effort to create an easy to understand, hands-on guide to learn RL without needing extensive knowledge about *machine learning*. After completing the tutorial, the reader will be familiar with the basics of RL and be able to create its own implementation based on the examples given.
This document serves as a documentation of the making and thought-process of the authors, as well as a presentation of the result and a critical reflection of the work. First, the basic contents of the tutorial will be explained, to give the reader a short overview about the topic of RL. Then, the initial targets will be outlined as well as the methods and materials, that were used in the making. Finally, all results will be presented and evaluated against the initial targets. This document will be concluded by the discussion of possible improvements and expansions of the final product.

2 Deep Reinforcement Learning

In this section we will discuss the content of two developed notebooks, mainly the theory behind Deep Reinforcement Learning (DRL). We will get to know the basic concept of RL, see on an exemplary neural network model the deep learning aspect and work out the necessary formulas in order to translate the theory into code. Further we will introduce a framework, which helps building this project by providing a game to play, more on that later. All the theory, explanations and code can be viewed in completion in the enclosed notebooks:

1. Deep Reinforcement Learning Beginners Tutorial (1) - Theory
2. Deep Reinforcement Learning Beginners Tutorial (2) - Practice

motivation

2.1 The Software Agent

Befor we start with the topic of RL we have to adress some basics in order to clarify the following explaination. We will use the term of *software agent*, which, as part of artificial intelligence, is a program, that acts self sufficient to solve a task. There are three important aspects an agent should fulfill. First are autonomous actions. An agent needs to make decisions without external help. Further an agent needs to execute multiple actions to complete its task, so it should be proactive. Last but not least, an agent has to be reactive, which means it reacts on changes in the environment it is in. An optional ability for an agent like this is independent improvement. For this the agent builds up knowledge after repetitively doing its task and improves itself this way.
Basically, the part of the agent, which controls its actions, can be filled with different algorithms. In our case, this will be an implementation of RL, which fulfills all requirements of an agent.

2.2 Reinforcement Learning

Reinforcement Learning is considered one of the three machine learning paradigms, alongside supervised learning and unsupervised learning. The main goal of RL is to create an agent, which can navigate by actions in an environment in a way to maximize a representation of a cumulative reward. The evironment is a space that contains the agent. We will talk about the environment later on. In order to miximize the reward, the agent has to learn by trial and error which actions most likely lead to a reward and which lead to a penalty. After some training, the agent will use its knowledge to avoid previous mistakes. It still has to explore the environment, because otherwise we can not be sure if we actually find the global maximum or just a local one and if there might be a better chain of actions. If we think about our real world, this method of learning is pretty close the human learning. If we get hurt for example, we are more likely to try to avoid the situation. Still curiosity or necessity leads us to exploring and thus maybe getting us into danger, but in the best case, will lead us to some kind of positive reward.

2.2.1 The concept of RL

Our agent is contained by an environment. A momentary snapshot of the environment is called state. A state contains all information at a certain point in time for example the position of the agent or enemies. There is a permanent exchange of information between the agent and its environment. The agent receives the actual state and has to choose an action based on its logic. Everything the agent can use to alter its environment is called an action. It changes the environment based on a certain set of rules. After executing an action the agents receives the new state and a reward, which helps to decide whether its decision leads to a positive result. Keep in mind, that a reward can also be negative. The main goal for the agent is to learn the best action for each state, so that it can maximize the reward. Basically, an environment is just a set of states. We can move between states by executing actions. A good example for an environment is our own world. We are agents, moving in the world. All the time we have to look at our surroundings and choose an action like moving accross the street or waiting until it is free. The laws of physics restrain us, take gravity for example. We receive rewards like geting hurt oder feeling satisfied, thats how we evalute our actions. Our environment is constantly changing, so we have to reevalute our decisions and choose an action again. The following picture shows the whole procedure:

2.2.2 Deep Learning Aspect

The agent has to recognize and differentiate elements in the environment, to be able to act accordingly and choose a proper action. In the best case, we already have the input reduced into some important numbers, which describe the actual state. In real applications this may not be possible or feasible with our knowledge. Most of the time we will have a picture available for example the image of a camera or a game screen, but the input we receive can basically

be any sensory data.

Furthermore the agent must associate the occurence of individual elements on screen with its own actions and the subsequent reward or penalty, which may occur only after several steps in the future. Depending on the situation it may be also necessary to estimate what the next state of the game could be. There are two possible ways to use the Deep Learning aspect.

1. Image processing with a Convolutional Neural Network (CNN), in order to process the image information and send this to the agent.
2. A Deep Q-Learning Network (DQN), in order to model the Q-function, which is discussed below.

Following is an exemplary presentation of a neural network, which is designed according to the tasks just mentioned.

There are two ways to learn in deep reinforcement learning. Eiher we can use Value Learning or Policy Learning. With Value Learning, we assign values to each state-action pair, which correspond to the probabilities of the pair beeing the best option. In Policy Learning, we will learn a strategy instead, which gives as the best estimated action for a given state s. We will focus at Value Learning.

2.2.3 Q-Learning

The task of the agent is the maximization of a cumulative future reward. An example of this can be the score in a game. In an environment, there is no guarantee for an immediate reward after executing an action. In many cases, only specific chains of actions lead to a positive reward, so the agent needs to learn multiple moves in succession in order to fulfill the given task. For example, if the agent needs to collect an object in order to get a higher score, it may be necessary to take multiple moves to reach it. The moves in between may not increase the reward significantly or even reduce it, however they are needed to complete the task in order to be successfull. So the agent may have to learn to sacrifice some rewards, to earn an overall higher reward in the end. A simple way to implement RL is Q-Learning. We assign a numerical value to each action we can execute per state. This value is called Q-value. It represents an estimate of which action will result in the highest reward for the actual state. This is the knowledge of our agent. In the beginning all these values are initialized by 0. Each step we have to choose an action for the actual state. This is either random or knowledge based, but all the actions taken, provide data the agent is trained with. We use random actions to explore our environment at first. If we use the knowledge of the agent, the next action is chosen by exploitation, so the agent uses the action with the highest Q-value for the given state. Our next step is the associated function. The Q-function is the name giving aspect of a Deep Q-Learning Network, while the "Q" stands for "Quality". The higher the reward, the higher the estimated Q-value and associated quality. That means this function calculates a Q-value for the actual state, which represents the best action to take. Therefor taking into account the highest Q-value associated with the heighest estimated future total Reward $Q(s', a')$, the last gained reward and a discount factor, in order to make rewards in the near future more dicisive for

the agent. The main formula to model such a behaviour is the so called *Bellman equation.*

$Q(s_t, a_t) = r_t + \gamma * \max_{a'} Q(s', a')$

2.3 The OpenAi Gym Framework

2.4 The Exercises

3 Methods and Materials

keras etc jupyter lab open ai gym: As we discussed before, Reinforcement Learning can be used to solve a range of different problems. Developing Machine Learning algorithms is often not easy to understand nor comprehensible especially for beginners. Furthermore, it is important to be able to compare the performance of different iterations of our algorithm, to be able to improve it.

So bascially we need an environment, that we can use to test and train our RL agent, which fulfills the following requirements:

repeatable test/training epochs finite set of inputs finite set of actions easy state representation easy to control agent deliver a score for a given state

In practice, not all of these points will be fulfilled, but as this is a beginners guide, we will start with a simple environment. Luckily, many video games can be used as quite good environments for machine learning purposes. Many implementations of RL are tested with games as Benchmark and there are some good reasons for this. Developing a whole test environment would be labour intensive and would require dedicated work towards a useable simulator. Using an existing game is also easier to compare to human performance and therefore the evaluation of different algorithms is easier. Another important point is the size of possible inputs and actions. The AI replaces the human player. Depending on the game, the input for our agent is an image, like a human player would see it. The set of actions is a combination of different buttons, which can be pressed on a controller. Finally, games are fun and most people can relate to them. It is also easier to understand what we want to accomplish, because we can transfer aspects from our human play style to the behaviour of an AI.

4 Results

critic/ expensions: using a framework policy learning / value learning -¿ both variants as cartpole agent policy gradient

notebooks: theory, exercises installation guide

outlook

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013.

[2] Garychl, "Applications of reinforcement learning in real world."

[3] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N. J. Yuan, X. Xie, and Z. Li, "Drn: A deep reinforcement learning framework for news recommendation," in *Proceedings of the 2018 World Wide Web Conference*, WWW '18, (Republic and Canton of Geneva, Switzerland), pp. 167–176, International World Wide Web Conferences Steering Committee, 2018.

[4] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.