POLITECNICO DI TORINO

Department of Control and Computer Engineering

Master of Science in Computer Engineering (Software Career)

Master Thesis

Deep Reinforcement Learning algorithms for autonomous systems

Design and implementation of a control system for autonomous driving task of a small robot, exploiting state-of-the-art Model-Free Deep Reinforcement Learning algorithms

Supervisors
prof. Pietro Michiardi
prof. Elena Baralis

Candidate
Piero Macaluso
matricola: s252894

Academic year TODO (Macaluso P.): 2019-2020

Abstract

TODO (Macaluso P.): Abstract is the last thing to do

Acknowledgements

TODO (Macaluso P.): Acknowledgements must be prepared!

Contents

Chapter 1

Introduction

1.1 Motivation

Autonomous systems, and in particular self-driving for unsupervised robots and vehicles (e.g. self-driving cars) is a topic that has attracted a lot of attention from both the research community and industry, due to its potential to radically change mobility and transport. In general, most approaches to date focus on formal logic methods, which define driving behavior in annotated geometric maps. This can be difficult to scale, as it relies heavily on an external mapping infrastructure rather than using and understanding the local scene.

In order to make autonomous driving a truly ubiquitous technology, in this thesis we focus on systems which address the ability to drive and navigate in the absence of maps and explicit rules, relying – just like humans do – on a comprehensive understanding of the immediate environment while following simple high-level directions (e.g. turn-by-turn route commands). Recent work in this area has demonstrated that this is possible on rural country roads, using GPS for coarse localization and LIDAR to understand the local scene.

Recently, Reinforcement Learning (RL) – a machine learning subfield focused on solving Markov decision process (MDP), where an agent learns to select actions in an environment in an attempt to maximize some reward function – has been shown to achieve super-human results at games such as Go or chess, to be particularly suited for simulated environments like computer games, and to be a promising methodology for simple tasks with robotic manipulators.

In this thesis, we argue that the generality of RL makes it a useful framework to apply to autonomous driving. For this reason we design and implement a control system for an autonomous driving task with a small robot, exploiting state-of-the-art model-free Deep RL algorithms and discussing possible ways to make them data efficient.

1.2 Structure of the thesis

The aim of this section is to describe the main structure of the thesis.

Chapter 1 - Introduction

The current chapter contains the motivation of this work and the structure of the thesis.

Chapter 2 - Reinforcement Learning Fundamentals

The aim of this chapter is to present a description as detailed as possible about RL state-of-the-art in order to provide the reader with useful tools to enter in this research field. TODO (Macaluso P.): Da qui in poi questo capitolo è da fare

Chapter 3 - Tools and Frameworks

This chapter explains briefly what are the main tools, frameworks and languages used in the thesis. TODO (Macaluso P.): Continue this list

OpenAI Gym a framework that is proposed as toolkit for developing and comparing RL algorithms.

Anki Cozmo Cozmo looks like a simple toy at first sight, but it hides an infinite potential under the hood, which make it a perfect candidate for the purposes of this thesis.

Chapter 4 - Design of the control system

Chapter 5 - Algorithms for Autonomous Systems

Chapter 5 - Experiments

This is the most important chapter. It shows all the results obtained during the numerous experiments with comments and speculations about them.

Chapter 6 - Conclusions

A summary of the results obtained from experiments with a specific part dedicated to future improvements.

1.3 Hardware and Software

In this section I want to list all software tools and hardware used, providing a quick introduction.

•

Chapter 2

Reinforcement Learning

Reinforcement Learning (RL) is a field of Machine Learning that is experiencing a period of great fervour in the world of research, fomented by recent progress in Deep Learning (DL). This event opened the doors to function approximation with Neural Network (NN) and Convolutional Neural Network (CNN) developing what is nowadays known as Deep Reinforcement Learning.

RL represents the third paradigm of Machine Learning alongside supervised and unsupervised learning. The idea behind this research field is that the learning process to solve a decision-making problem consists in a sequence of trial and error where the *agent*, the protagonist of RL,could discover and discriminate valuable decisions from penalising ones exploiting information given by a *reward signal*. This interaction has a strong correlation with what human beings and animals do in the real world to forge their behaviour.

Recently RL has known a remarkable development and interests in video games: it managed to beat world champions at the game of Go [1] and Dota with superhuman results and to master numerous Atari video games [2] from raw pixels. Decisions, actions and consequences make video games a simulated reality on which to exploit and test the power of RL algorithms. It is essential to realise that the heart of RL is the science of decision making. This fact makes it compelling and general for many research fields ranging from Engineering, Computer Science, Mathematics, Economics, to Psychology and Neuroscience.

Before discussing the results of this thesis, it is good to clarify everything that today represents the state-of-the-art in order to understand the universe behind this new paradigm better. Indeed, the exploration of this field of research is the main aim of this chapter: the first section begins with the definition of the notation used and with the theoretical foundations behind RL, then in the second section it moves progressively towards what is Deep RL through a careful discussion of the most important algorithms paying more attention to those used during the thesis project.

The elaboration of this chapter is inspired by [3], [4], [5], [6].

2.1 Elements of Reinforcement Learning

In all bibliography books and papers about RL, it is likely to find different choices about notation used. For this reason, this thesis is written using one of the conventional notation of RL, trying to be as coherent as possible through all reported demonstrations.

As a general guideline, this exposition uses uppercase letters (e.g. $\mathcal{A}$) to describe sets of elements, while lowercase letters (e.g. a) to represent the specific instance of a set. Some entities are related to a specific timestep t and, for this reason, added as subscript (e.g. a_t).

2.1.1 The Reinforcement Learning Problem

Reinforcement Learning is a computational approach to Sequential Decision Making. It provides a framework that is exploitable with decision-making problems that are unsolvable with a single action and need a sequence of actions, a broader horizon, to be solved. In this context, RL algorithms learn how to improve and maximise a future reward from interactions between two main components: the agent and the environment.

The *agent* is the entity that interacts with the environment by making decisions based on what it can observe from the state of the surrounding situation. The decisions taken by the agent consist of *actions* (a_t). The agent has no control over the environment, but actions are the only means by which it can modify and influence the environment.

Usually, the agent has a set of actions it can take, which is called *action space*. Some environments have discrete action spaces, where only a finite number of moves are available (e.g. $\mathcal{A} = [\text{North}, \text{South}, \text{East}, \text{West}]$ choosing the direction to take in a bidimensional maze). On the other side, there are continuous action spaces where actions are vectors of real values. This distinction is fundamental to choose the right algorithm to use because not all of them could be compatible with both types: according to the needs of the specific case, it may be necessary to modify the algorithm to make it compatible.

The *environment* represents all the things that are outside the agent. At every action received by the agent, it emits a reward, an essential aspect of RL, and an observation of the environment.

The *reward* r_t is a scalar feedback signal that defines the objective of the RL problem. This signal allows the agent to be able to distinguish positive actions from negative ones in order to reinforce and improve its behaviour. It is crucial to notice that the reward is local: it describes only the value of the latest action. Furthermore, actions may have long term consequences, delaying the reward. As it happens with human beings' decisions, receiving a conspicuous reward at a specific time step does not exclude the possibility to receive a small reward immediately afterwards and sometimes it may be better to sacrifice immediate reward to gain more rewards later.

In this context, many features make RL different from supervised and unsupervised learning. Firstly, there is no supervisor: when the agent has to decide what action to take, there is no entity that can tell him what the optimal decision is in that specific moment. The agent receives only a reward signal which may delay compared to the moment in which it has to perform the next action. This fact brings out another significant difference: the importance of time. The sequentiality links all actions taken by the agent, making resulting data no more indipendent and identically distributed (i.i.d.).

Given these definitions, it is noticeable that the primary purpose of the agent is to maximise the cumulative reward called *return*.

The *return* g_t is the total discounted reward starting from timestep t defined by eq. (2.1) where γ is a *discount factor*.

$$g_t = r_{t+1} + \gamma r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \quad \gamma \in [0,1) \tag{2.1}$$

Not only the fact that animal and human behaviour show a preference for immediate rewards rather than for the future ones motivates the presence of this factor, but it is also mathematically necessary: an infinite-horizon sum of rewards may not converge to a finite value. Indeed, the return function is a geometric series, so, if $\gamma \in [0,1)$, the series converges to a finite value equal to $1/(1-\gamma)$. For the same convergence sake, the case with $\gamma = 0$ makes sense only with a finite-horizon cumulative discounted reward.

The other data emitted by the environment is the *observation* (o_t) that is related to the *state* (s_t). It represents a summary of information that the agent uses to select the next action, while the *state* is a function of the *history* the sequence of observation, actions and rewards at timestep t as shown in eq. (2.2).

$$h_t = o_1, r_1, a_1, \ldots, a_{t-1}, o_t, r_t, \qquad s_t = f(h_t) \tag{2.2}$$

The sequence of states and actions is named *trajectory* (τ): it is helpful to represent an episode in Reinforcement Learning (RL) framework.

The state described above is also called *agent state* s_t^a, while the private state of the environment is called *environment state* s_t^e. This distinction is useful for distinguishing fully observable environments where $o_t = s_t^e = s_t^a$, from partially observable environments where $s_t^e \neq s_t^a$. In the first case, the agent can observe the environment state directly, while in the second one, it has access to partial information about the state of the environment.

Beyond the fact that this chapter will focus on fully observable environments, the distinction between state and observation is often unclear and, conventionally, the input of the agent is composed by the reward and the state as shown in fig. 2.1.

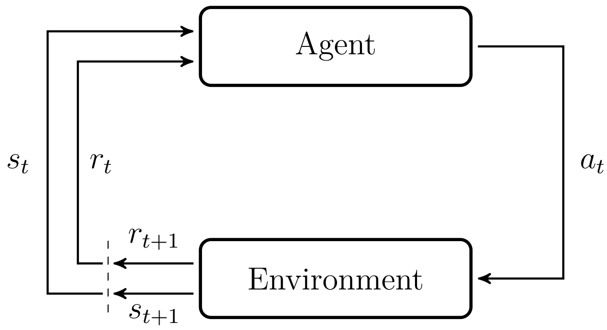

Figure 2.1. Interaction loop between Agent and Environment. The reward and the state resulting from taking an action become the input of the next iteration.

Furthermore, a state is called *informational state* (or *Markov state*) when it contains all data and information about its history. Formally, a state is a Markov state if and only if satisfies eq. (2.3).

$$\mathbb{P}[s_{t+1}|s_t] = \mathbb{P}[s_{t+1}|s_1, \ldots, s_t] \tag{2.3}$$

It means that the state contains all data and information the agent needs to know to make decisions: the whole history is not useful anymore because it is inside the state. The environment state s_t^e is a Markov state.

With all the definitions shown so far, it is possible to formalise the type of problems on which RL can unleash all its features: the Markov decision process (MDP), a mathematic framework to model decision processes. Its main application fields are optimization and dynamic programming.

An MDP is defined by

$$< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$$

$$\begin{aligned}
\text{where} \quad & \mathcal{S} \text{ is a finite set of states} \\
& \mathcal{A} \text{ a finite set of actions} \\
& \mathcal{P} \text{ a state transition probability matrix } \mathcal{P}^a_{ss'} = \mathbb{P}[s_{t+1} = s'|s_t = s, a_t = a] \\
& \mathcal{R} \text{ a reward function } \mathcal{R}^a_s = \mathbb{E}[r_{t+1}|s_t = s, a_t = a] \\
& \gamma \text{ a discount factor such that } \gamma \in [0,1]
\end{aligned} \tag{2.4}$$

The main goal of an MDP is to select the best action to take, given a state, in order to collect the best reward.

2.1.2 Bellman Equation

In this quick overview of the main unit of RL the components that may compose the agent, the brain of the RL problem can not be missing: they are the *model*, the *policy* and the *value function*.

A *model* is composed by information about the environment. These data must not be confused with *states* and *observations*: they make it possible to infer prior knowledge about the environment, influencing the behaviour of the agent.

A *policy* is the core of RL because it is the representation of the agent's behaviour. It is a function that describes the mapping from states to actions. The *policy* is represented by π and it may be deterministic $a_t = \pi(s_t)$ or stochastic $\pi(a_t|s_t) = \mathbb{P}[a_t|s_t]$.

In this perspective, it is evident that the central goal of RL is to learn an optimal policy π^*. The optimal policy is a policy which can show agent what the most profitable way to achieve the maximum return is, what is the best action to do in a specific situation. In order to learn the nature of the optimal policy, RL exploits value functions.

A *value function* represents what is the expected reward that the agent can presume to collect in the future, starting from the current state. The reward signal represents only a local value of the reward, while the value function provides a broader view of future rewards: it is a sort of prediction of rewards.

It is possible to delineate two main value functions: the *state value* function and the *action value* function.

- The *State Value Function* $V^\pi(s)$ is the expected return starting from the state s and always acting according to policy π.

$$V^\pi(s) = \mathbb{E}_{\tau\sim\pi}[g_t|s_0 = s] \tag{2.5}$$

- The *Action Value Function* $Q^\pi(s)$ is the expected return starting from the state s, taking an action a and then always acting according to policy π.

$$Q^\pi(s,a) = \mathbb{E}_{\tau\sim\pi}[g_t|s_0 = s, a_0 = a] \tag{2.6}$$

Both these functions satisfy recursive relationships between the value of a state and the values of its successor states. It is possible to see this property deriving *Bellman equations* as shown in appendix A.1 on page 25.

The results are shown in eq. (2.7) on the next page where $s_{t+1} \sim E$ means that the next state is sampled from the environment E and $a_{t+1} \sim \pi$ shows that the next action is taken following the policy π. TODO (Macaluso P.): Rivedi queste formule inserendo estensioni

$$\begin{aligned}
V^\pi(s_t) &= \mathbb{E}_{a_t\sim\pi, s_{t+1}\sim E}[r(s_t,a_t) + \gamma V^\pi(s_{t+1})] \\
&= \sum_{a\in\mathcal{A}} \pi(a|s) \sum_{s'\in\mathcal{S}, r\in\mathcal{R}} P(s',r|s,a)\Big[r + \gamma V^\pi(s')\Big] \\
Q^\pi(s_t,a_t) &= \mathbb{E}_{s_{t+1}\sim E}[r(s_t,a_t) + \gamma \mathbb{E}_{a_{t+1}\sim\pi}[Q^\pi(s_{t+1},a_{t+1})]]
\end{aligned} \tag{2.7}$$

$r(s_t, a_t)$ is a placeholder function to represent the reward given the starting state and the action taken. As discussed above, the goal is to find the optimal policy π^* to exploit. It can be done using the optimal value functions defined in eq. (2.8).

$$\begin{aligned} V^*(s_t) &= \max_a \mathbb{E}_{s_{t+1}\sim E}[r(s_t, a) + \gamma V^*(s_{t+1})] \\ Q^*(s_t, a_t) &= \mathbb{E}_{s_{t+1}\sim E}[r(s_t, a_t) + \gamma \max_{a'}[Q^*(s_{t+1}, a')]] \end{aligned} \tag{2.8}$$

Value functions allow defining a partial ordering over policies such that

$$\pi \geq \pi' \text{ if } V_\pi \geq V_{\pi'}, \forall s \in \mathcal{S}$$

This definition is helpful to enounce the *Sanity Theorem*. It asserts that for any MDP there exists an optimal policy π^* that is better than or equal to all other policies, $\pi^* \geq \pi, \forall \pi$, but also that all optimal policies achieve the optimal state value function and the optimal action-value function.

The solution of Bellman Optimality Equation is not linear and, in general, there is no closed-form solution. For this reason, there are many iterative methods. TODO (Macaluso P.): Bisognerà aggiungere riferimento a sezioni successive.

2.1.3 Approaches of Reinforcement Learning

It is possible to explain the main strategies in RL to solve problems using *policy*, *model* and *value function* defined previously.

Every agent has a specific application field which depends on the different approach it supports. Understanding differences among these approaches is useful to adequately understand what type of algorithm satisfies better the needs of a specific context.

The distinctions presented in this part are just a part of the complete set because this section aims to describe the most crucial distinctions that are useful in the context of the thesis without claiming to be exhaustive.

Model-Free vs Model-Based

One of the most crucial aspects of an RL algorithm is the question of whether the agent has access to (or learns) a model of the environment. A model of the environment enables the agent to predict state transitions and rewards.

A method is *model-free* when it does not build a model of the environment. All the actions made by the agent results from direct observation of the current situation in which the agent is. It takes the observation, does computations on them and then select the best action to take.

This last representation is in contrast with *model-based* methods. In this case, the agent tries to build a model of the surrounding environment in order to infer information useful to predict what the next observation or reward would be.

Both groups of methods have strong and weak sides. Ordinarily, *model-based* methods show their potential in a deterministic environment (e.g. board game with rules). In these contexts, the presence of the model enables the agent to plan by reasoning ahead, to recognise what would result from a specific decision before taking action. The agent can extract all this knowledge and learn an optimal policy to follow. However, this opportunity is not always achievable: the model may be partially or entirely unavailable, and the agent would have to learn the model from its experience. Learning a model is radically complex and may lead to various hurdles to overcome: for instance, the agent can exploit the bias present in the model, producing an agent which is not able to generalise in real environments.

On the other hand, model-free methods tend to be more straightforward to train and tune because it is usually hard to build models of a heterogeneous environment. Furthermore, model-free methods are more popular and have been more extensively developed and tested than model-based methods.

Policy-Based vs Value-Based

The use of policy or value function as the central part of the method represents another essential distinction between RL algorithms.

The approximation of the policy of the agent is the base of *policy-based* methods. The representation of the policy is usually a probability distribution over available actions. This method points to optimise the behaviour of the agent directly and, because of its on-policy nature, may ask manifold observations from the environment: this fact makes this method not so sample-efficient.

On the opposite side, methods could be *value-based*. In this case, the agent is still involved in finding the optimal behaviour to follow, but indirectly. It is not interested anymore about the probability distribution of actions. Its main objective is to determine the value of all actions available, choosing the best value. The main difference from the policy-based method is that this method can benefit from other sources, such as old policy data or replay buffer.

On-Policy vs Off-Policy

It is possible to classify this method also by different types of policy usage.

An *off-policy* method can use a different source of valuable data for the learning process instead of the direct experience of the current policy. This feature allows the agent to use, for instance, large experience buffers of past episodes. In this context, these buffers are usually randomly sampled in order to make the data closer to being independent and identically distributed (i.i.d): random extraction guarantees this fact.

On the other hand, *on-policy* methods profoundly depend on the training data to be sampled according to the current policy.

2.1.4 Dynamic Programming

Dynamic programming is one of the approaches used to solve RL problems. Formally, it is a general method to solve complex problems by breaking them into sub-problems that are more convenient to solve. After solving all sub-problems, it is possible to sum them up in order to obtain the final solution to the whole original problem.

This technique provides a practical framework to solve MDP problems and to observe what is the best result achievable from it, but it assumes to have full knowledge about the specific problem. For this reason, it applies primarily to model-based problems.

This thesis will not focus on this type of approach, so this section aims to present only the basic concept of *policy iteration* and *value iteration* which are worth quoting: the fourth chapter of [4] provides further details about this section.

Policy Iteration

The *policy iteration* aims to find the optimal policy by directly manipulating the starting policy. However, before proceeding with this process, a proper evaluation of the current policy is essential. This procedure can be done iteratively following algorithm 2.1 on the next page where θ is the parameter that defines the accuracy: the more the value is closer to 0, the more the evaluation would be precise.

Algorithm 2.1: Iterative Policy Evaluation for estimating V_π

Input: π the policy to be evaluated; a small threshold θ which defines the accuracy of the estimation

Initialise $V(s)$ $\forall s \in \mathcal{S}$ arbitrarily except that $V(terminal) = 0$
repeat
    $\Delta \leftarrow 0$
    for *each* $s \in \mathcal{S}$ **do**
        $\text{v} \leftarrow V(s)$
        $V(s) \leftarrow \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} P(s', r|s, a)\Big[r + \gamma V(s')\Big]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
    end
until $\Delta \leq \theta$
$V_\pi \leftarrow V(s)$

Output: V_π

Policy improvement represents the second step towards policy iteration. Intuitively, it is possible to find a more valuable policy than the starting one by changing the action to take in a specific state with a more rewarding one. The key to check if the new policy is better than the previous one is to use the action-value function

$Q_\pi(s, a)$. This function returns the value of taking action a in the current state s and after that following the existing policy π. If $Q_\pi(s, a)$ is higher than $V_\pi(s)$, so the action selected is better than the action chosen by the current policy, and consequently, the new policy would be better overall.

Policy improvement theorem is the formalisation of this fact: appendix A.2 on page 25 shows its demonstration. Thanks to this theorem, it is reasonable to act greedily to find a better policy starting from the current one iteratively selecting the action that produces the higher $Q_\pi(s, a)$ for each state.

The iterative application of policy improvement stops after an improvement step that does not modify the initial policy, returning the optimal policy found.

Algorithm 2.2: Policy Improvement for estimating $\pi \sim \pi'$

Input: π the policy to be improved
$is\_policy\_stable \leftarrow true$
for *each* $s \in \mathcal{S}$ **do**
  $old\_action \leftarrow \pi(s)$
  $\pi(s) \leftarrow \arg\max_a \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} P(s', r|s, a)\left[r + \gamma V_\pi(s')\right]$
  if $old\_action \neq \pi(s)$ **then**
    $is\_policy\_stable \leftarrow false$
  end
end
if is_policy_stable **then**
  Stop and return $V \sim V^*$ and $\pi \sim \pi^*$
end
Output: $V*$ and $\pi*$

TODO (Macaluso P.): Restart from here

2.1.5 Monte Carlo

2.1.6 Temporal Difference (TD) Learning method

2.1.7 Function Approximation

2.2 Deep Reinforcement Learning

2.2.1 Taxonomy of Deep RL Algorithm

After the quick overview of the basics of RL terminology and notation provided in the previous section, it is possible to explore more in-depth the universe behind the algorithms of modern Deep RL. Because of the nature of this work, the focus of this section will be on the types of algorithms used in the thesis, without leaving out a quick overview of other types of algorithms most used today in Deep RL.

2.2.2 Deep Deterministic Policy Gradient (DDPG)

2.2.3 Soft-Actor Critic (SAC)

2.3 Summary

Chapter 3

Tools and Frameworks

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

3.1 Environment Setup

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

3.2 OpenAI Gym

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

3.2.1 The importance of a Framework

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

3.2.2 Main features

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

3.2.3 How to create an environment

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

3.3 PyTorch

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

3.3.1 Tensor and Gradients

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

3.3.2 Building a CNN

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

3.3.3 Loss function and Optimizers

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

3.3.4 TensorboardX

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

3.4 Anki Cozmo

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

3.4.1 Features of Cozmo

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

3.4.2 Cozmo SDK

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

Chapter 4

Design of the Control System

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

4.1 The main concept

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

4.1.1 Related Work

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

4.2 The Track

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

4.2.1 Track Design and Materials

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

4.2.2 Problems and solutions

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

4.3 Cozmo Control System

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

4.3.1 Formalization as an MDP

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

4.3.2 Design of the OpenAI Gym Environment

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

4.3.3 Main setup of the system

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

Chapter 5

Experiments

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

5.1 Results of the experiments

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

5.1.1 Track A

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

5.1.2 Track B

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

Chapter 6

Conclusions

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

6.1 Future Work

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

Appendix A

A.1 Bellman Equation

TODO (Macaluso P.): Check correctness and completeness

The value function is decomposable in the immediate reward r_t and the discounted state value of the next state. It is possible to obtain the result in eq. (A.1) by writing expectations explicitly.

$$\begin{aligned} V^\pi(s) &= \mathbb{E}[g_t|s_t = s] \\ &= \mathbb{E}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots |s_t = s] \\ &= \mathbb{E}[r_{t+1} + \gamma g_{t+1}|s_t = s] \\ &= \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} P(s', r|s, a) \Big[r + \gamma \mathbb{E}[g_{t+1}|s_{t+1} = s']\Big] \\ &= \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} P(s', r|s, a) \Big[r + \gamma V^\pi(s')\Big] \end{aligned} \tag{A.1}$$

This equation expresses the relationship between the value of a state and the values of its successor states. It is further possible to derive the Bellman Equation for Action-Value function using the same procedure described above.

The resulting formulas are shown in eq. (2.7) on page 9.

Furthermore, it is possible to obtain the Bellman Equation solution in eq. (A.2) working with matrix notation.

$$\begin{aligned} V^\pi &= \mathcal{R}^\pi + \gamma \mathcal{P}^\pi V^\pi \\ (I - \gamma \mathcal{P}^\pi) V^\pi &= \mathcal{R}^\pi \\ V^\pi &= (I - \gamma \mathcal{P}^\pi)^{-1} \mathcal{R}^\pi \end{aligned} \tag{A.2}$$

A.2 Policy Improvement Theorem

Let π and π' be any pair of deterministic policy such that

$$Q_\pi(s, \pi'(s)) \geq V_\pi(s) \; \forall s \in S \tag{A.3}$$

Then the policy π' leads to

$$V'_\pi(s) \geq V_\pi(s) \tag{A.4}$$

Therefore, the presence of strict inequality in eq. (A.3) on the preceding page for a state leads to a strict inequality of eq. (A.4).

The proof of this theorem is shown in eq. (A.5).

$$\begin{aligned}
V_\pi(s) &\leq Q_\pi(s, \pi'(s)) \\
&= \mathbb{E}[r_{t+1} + \gamma V_\pi(s_{t+1}) | s_t = s, a_t = \pi'(s)] \\
&= \mathbb{E}_{\pi'}[r_{t+1} + \gamma V_\pi(s_{t+1}) | s_t = s] \\
&\leq \mathbb{E}_{\pi'}[r_{t+1} + \gamma Q_\pi(s_{t+1}, \pi'(s_{t+1})) | s_t = s] \;\; \text{(by A.3)} \\
&= \mathbb{E}_{\pi'}[r_{t+1} + \gamma \mathbb{E}_{\pi'}[r_{t+2} + \gamma V_\pi(s_{t+2}) | s_{t+1}, a_{t+1} = \pi'(s_{t+1})] | s_t = s] \\
&= \mathbb{E}_{\pi'}[r_{t+1} + \gamma r_{t+2} + \gamma^2 V_\pi(s_{t+2}) | s_t = s] \\
&\leq \mathbb{E}_{\pi'}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 V_\pi(s_{t+3}) | s_t = s] \\
&\vdots \\
&\leq \mathbb{E}_{\pi'}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \ldots | s_t = s] \\
&= v_{\pi'}(s)
\end{aligned} \tag{A.5}$$

Acronyms

CNN Convolutional Neural Network. 4

DL Deep Learning. 4

i.i.d. indipendent and identically distributed. 6

MDP Markov decision process. 1, 7, 9

NN Neural Network. 4

RL Reinforcement Learning. 1, 2, 4, 5, 7, 8

Bibliography

[1] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[3] David Silver. University college london course on reinforcement learning. http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html, 2015.

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

[5] OpenAI. Open AI spinning up, 2019. https://spinningup.openai.com/.

[6] Maxim Lapan. *Deep Reinforcement Learning Hands-On: Apply modern RL methods, with deep Q-networks, value iteration, policy gradients, TRPO, AlphaGo Zero and more*. Packt Publishing Ltd, 2018.