Bipedal Walking - Reinforcement Learning

Roberto Figueiredo

April 2022

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Acknowledgement

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Contents

Chapter 1

Introduction

In this chapter the target problem will firstly be introduced, as will a proposed solution to solve it. To understand the problem, background information will be covered to better understand the problem and what it is meant to target.

1.1 RoboCup

RoboCup is an attempt to advance the field of robotics by providing a common problem and an environment for sharing knowledge and collaborate. The first RoboCup took place in 1997 in Nagoya, Japan. The Objective of RoboCup is to achieve fully autonomous soccer playing robots that are able to defeat the FIFA World Cup champions by 2050.

RoboCup has since evolved from just soccer and now includes multiple fields, Rescue, Soccer, @Home, Industrial and Junior. [3]

1.2 Soccer League

RoboCup Soccer is split into multiple leagues, each with different challenges and focuses. The Small Size league uses small wheeled robots, each team is composed of six robots and play using a orange golf ball while tracked by a top-view camera; This enables the robots to abstract from challenges such as complex vision detection, walking and others, enabling the teams to focus on strategy and multi-robot/agent cooperation and control in a highly dynamic environment with a hybrid centralized/distributed system.

Middle size League assimilates to small size league, but in this case, the robots are of a larger size and must have all sensors on board; The main focus

of the league is on mechatronics design, control and multi-agent cooperation at plan and perception levels.

RoboCup also has simulation for most of its leagues, allowing the teams to focus on software and avoid the problems originated by using real robots hardware.

The Standard Platform is a step-up from the simulation league as while it uses real robots, NAOs, it allows the teams to focus mainly on software while using real robots and the challenges of using real robots without having to develop custom hardware. Each robot is fully autonomous and takes its one decisions.

The Humanoid league, assimilates the most to humans, using robots assimilating its shape and unlike humanoid robots outside the Humanoid League, the task of perception and world modeling is not simplified by using non-human like range sensors, making this, the most transversal league, requiring hardware and software development. In addition to soccer competitions technical challenges take place. Dynamic walking, running, and kicking the ball while maintaining balance, visual perception of the ball, other players, and the field, self-localization, and team play are among the many research issues investigated in the Humanoid League.

1.3 Bold Hearts

1.4 Introduction to the Project

1.4.1 Problem

Robotic locomotion has, until a few years ago, been focused on wheel-based movement. Although it is very stable and easy to implement, it lacks flexibility, the ability to move on uneven, unpredictable terrain and overcome obstacles such as stairs.

As a member of the Bold Hearts team, which competes in the Teen size league, a branch of the Humanoid League, robots must walk and, in addition to the high complexity of the challenges imposed by the humanoid leagues, RoboCup rules periodically change, these changes are put in place as the RoboCup objective is to achieve the most realistic environment. Rule changes affect both robots, changing the required height, sensors and others, as well as affecting the environment such as moving from flat ground to synthetic grass.

One of the main challenges faced by the team is walking, which is one of the most complex movements performed by humans, requiring simultaneous

control of multiple joints to move while maintaining balance. This is one of the most energy and time-consuming problems faced by the team additionally, the rules changes and continuous improvements require updates to the robot's structural architecture, leading to new updates to the walking algorithm.

1.4.2 Proposed Solution

Walking algorithms can be developed using various techniques, including explicit programming, supervised learning and unsupervised learning. Walking is very complex as there are a lot of variables involved in it, such as the ground contact, maintaining balance and the complex gait movement. The two main aspects important to highlight are that the walking algorithm requires changes when both the robot or external factors change and that it requires manual work from the team to achieve this.

The best way to solve the first mentioned problem requires having a robot and environment agnostic approach, this is possible by using Reinforcement Learning as the principles of the movement maintains therefore it should be possible to develop an agnostic reward function. From the moment a reinforcement learning solution is successfully implemented, the team should be able to use the same implementation to retrain the policy using the updated robot/environment that has been modeled in the simulator. This also allows to reduce the complexity of the problem as the input into the system is sensory data and the output is a set of actions to execute on the joints.

1.4.3 Aims and Objectives

This project proposes to develop a reinforcement learning implementation for walking, to achieve the objectives of the project, three main topics will be covered, robotics biped locomotion, reinforcement learning algorithms and the training framework.

While achieving a working walking pattern would be desireable, the project aims to develop an implementation of reinforcement learning to achieve walking, along with it aims to develop resourcefull documentation on the process and its results, allowing for reproducibility and further development and adaptations of the current implementation to learn not only walking but any suitable problem.

Chapter 2

Background Research

2.1 Reinforcement Learning

Reinforcement learning is one of the main machine learning paradigms, alongside supervised learning and unsupervised learning. The objective of reinforcement learning is essentialy to map states to actions while maximizing a reward signal, in same cases, actions may affect not only the present but future situations. To learn how to map the states to actions the agent must try them, this charactheristics, trial-and-error and delayed reward are the most distinguishable features of reinforcement learning.

The agent must be able to perform actions affecting the environment followed by a perception, this perception should indicate to what state the robot has transitioned and the reward signal associated with the result of the action taken given the previous state. This interaction is summerized in the following figure.

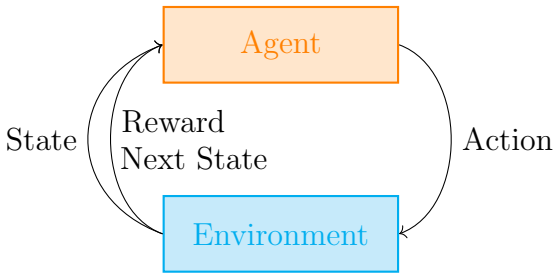

Figure 2.1: Reinforcement learning model

One unique challange in reinforcement learning is balancing exploration and exploitation. To succed a task the agent must exploit actions that, through experience have yielded the most rewards, although, to have experience and perform better in the future the agent must explore new actions. The dilemma is that neither exploration nor exploitation can be pursued exclusively without failing at the task [8]

2.1.1 Markov Decision Process

Reinforcement learning can be described using the Markov Decision Process(MDP). MDP is the final summary concept of the individual elements:

- The Markov Decision Chain
- The Markov Property
- The Markov Reward Process

Markov Decision Chain

A Markov chain is a mathematical system that experiences transitions from one state to another according to certain probabilistic rules. The defining characteristic of a Markov chain is that no matter how the process arrived at its present state, the possible future states are fixed. In other words, the probability of transitioning to any particular state is dependent solely on the current state.

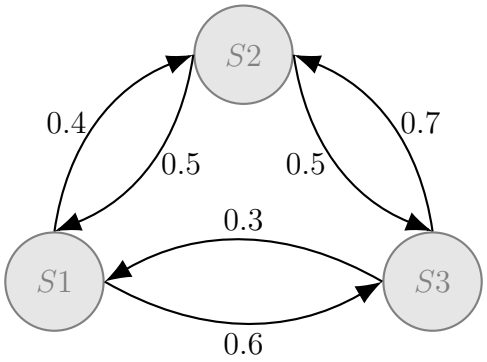

Figure 2.2: Example of a Markov Decision Chain

As can be observed by the example Markov Decision Chain, the transition probabilities are fixed and are only dependent on the current state, this is the Markov property.

$$P(S_{t+1}|S_t)$$

Definition of the probability of transitioning to any particular state given the current state.

Markov Property

This means that the transistion to state t_{+1} from state t is independent of the past, meaning that our current state already captures the information of the past states. Defined by the following equation:

$$P[S_{t+1}|S_t] = P(S_{t+1}|S_1, ..., S_t)$$

Markov Reward Process

As it suggests, the Markov Reward Process is a Markov process with the difference that it includes a reward system that indicates how much reward is accumulated through a particular sequence. An additional factor is applied, the **discount factor** γ that indicates how much the future reward is discounted. if $\gamma = 0$ then the agent will only consider the immediate reward, if $\gamma = 1$ then the agent will consider all subsequent rewards. In practice this extreme rewards are inneffective and γ is usually set to values between 0.9 and 0.99

2.1.2 Bellman equation

The bellman equation is the basis to solve RL problems, it helps us solving the MDP, it is defined as:

$V(s) = max_a(R(s,a) + \gamma V(s'))$

The Bellman equation sums to the present reward the discount factor multiplied by the best next value outputed by the value function.

The optimal value function $V^*(s)$ maximizes the expected reward. To achieve this the Bellman equation is solved by iteratively updating the value function until $V^*(s)$ is reached.

2.2 Learning Algorithms

One of the main decisions in implementing a reinforcement learning algorithm is the learning algorithm, to understand the algorithm decision its important to understand how this differ.

The first major split in RL algorithms is whether it is model-free or model-based, model-based algorithms use a model of the environment, which is capable of mimicking its behaviour, this allows inferences to be made about how the environment will behave given an action and what the next reward will be. This method of reinforcement learning isn't adequate for the current problem due to the high complexity of the environment in which the agent is interacting, making it impossible to model it.

The second decision is between **policy optimization** and **Q-Learning**. While both aproaches have similar concepts and are driven by MDP they are internally different.

In Q-learning, the goal is to determin a single action from a set of discrete actions by finding the action with the highest Q-value. While in policy optimization the goal is to learn a map from state to action, this can be stochastic and, opposed to Q-Learning, workds in continuous action spaces.

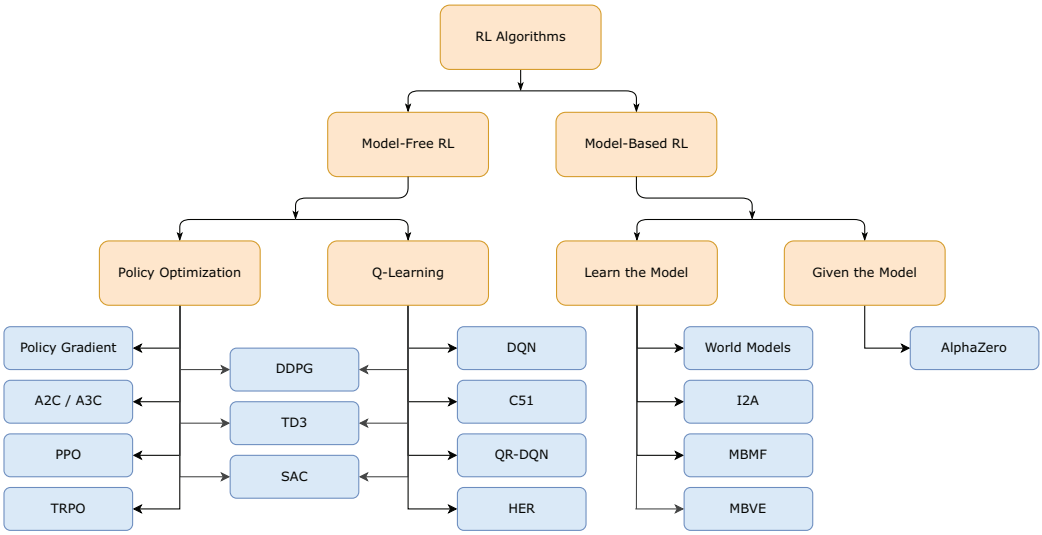

Figure 2.3: A non-exhaustive, but useful taxonomy of algorithms in modern RL. [5]

To the problem in study, Q-Learning was a better fit as the action space was discretized(covered further in chapter 3), apart from the action space, the Implementation of Q-learning is simpler and more common.

The algorithm decision was set on Deep Q-Network (DQN). The main difference between DQN and Q-Learning is that DQN implements a neural network replacing the Q-table. This is a benefit due to the complexity of the environment and the total number of possible states, using standard Q-learning would require a very large q-table, which is a huge memory and computational burden.

2.3 Training Framework

Followed by the new advances in RL there was a growing need for a common benchmarking framework, that would allow for comparing algorithms and implementations.

Gym was releasead by the OpenAI team to enable this, by providing an accessible API and common environments. Gym is now the most widely used tool in RL research, it has a large community and documentation and a growing number of environments.

Gym was choosen as the framwork to standardize the implementation which allows for an easier understanding and comparison, behond this, gym

also facilitates the implementation by using all the pre-built functions.

2.4 Previous Implementations

To develop this project it was important to first study previous implementations, what problems were faced and the reasoning behind there decisions, this brought light to many problems and details of implementing reinforcement learning in special for walking.

One important implementation was the **Deep Q-Learning for Humanoid Walking**[9], an implementation of reinforcement walking on the Atlas platform. this implementation higlits some of the general pre-conceptions regarding the reward system and how to handle the complexity of controlling all the joints simultaneously.

2.5 Logging and Reproducibility

One of the most important aspects of machine learning and Reinforcement Learning in particular is data logging and reproducibility. This project required extensive testing of hyperparameters and reward systems. To understand the results and its correlation with the variables in testing its important to log all the results and to be able to reproduce the results. **Requirements for logging:**

- Hyperparameters
- Performance metrics
- Code
- Models
- Renderings

From previous experiences with machine learning and logging platforms MLflow was the first option analysed, although, Weights & Biases (WandB) was the choosen platform, WandB offered an hosted version option compared to MLflow, WandB also integrates with Keras allowing for a seamless implementation. WandB fills all the requirements and allows for a comparison of the renderedings and performance results between runs.

Chapter 3

Development Structure

Due to the complexity of the project, a development structure has been put in place, this includes multiple steps of increasing complexity and realism, the increasing complexity allowed for detecting problems at an earlier, simpler stage, making the transition easier.

3.1 Structure

3.1.1 Cartpole

Cartpole is a classic exercise of reinforcement learning, it consists in balancing a pole in a cart moving on an horizontal plane, this environment allows for 2 discrete actions, which consist of applying force on the right or left side of the cart making it move in the oposite direction.

3.1.2 2D Walker

In this step a 2D environment of a simplified humanoid was used in order to train a walking behaviour, this new environment introduced a lot of new variables such as controlling multiple joints in a step, understanding how to efficiently calculate the best action and which algorithms to use. New challenges such as implementing a custom reward system, rendering and step functions where an important step in order to transition to 3D simulation.

3.1.3 3D Walker

3d simulation brings new challenges, such as a larger range of motion and more joints to controll, along with a more complex environment, requiring more processing power and more time to solve the problem. Along with this it requires a more complex reward system as a new dimention poses new problems.

3.2 Environment Definition

3.2.1 Cartpole

Its observation space consists of position of the cart on the horizontal axis and its velocity and the angle of the pole and its angular velocity. The objective of this environment is to balance the pole over 500 episodes To balance the pole the angle needs to stay in between $\pm 12°$ and the cart position stay in between the bounds of ± 2.4 The reward system is for cartpole is very simple, it earns 1 point for each time step survived. [4]

3.2.2 2D Walker

The 2d environment uses as a physics engine Pymunk [7], a python implementation of Chipmunk[1] in conjunction with pygame [6] to render the simulation.

To achieve walking a 2D simplified humanoid was developed in this environment, it consists of 8 joints, shoulder, hips, knees and ankle.

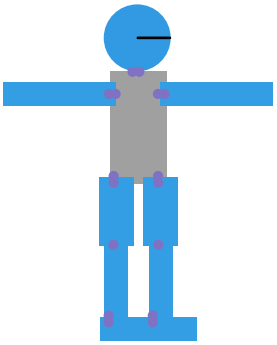

Figure 3.1: Representation of 2D humanoid

The reward system developed for this environment:

- Moves back: penalty of 200 points
- Stays in place: penalty of 100 points
- Moves forward: receives 0 points
- Both feet lose contact with the ground: cumulative penalty of 50 points
- Reaches target position: reward of 100 points
- Falls: penalty calculated as $\frac{1}{1-\gamma} \cdot$(highest penalty)

3.2.3 3D Walker

[2]

Chapter 4

Experiments

This chapter will cover the development process, the decisions that have been taken as well as the results and outcomes of the individual development parts.

4.1 Cartpole Outcomes

The first task developed was the classic cartpole environment, this was helpfull in understanding core concepts of reinforcement learning and neural networks, along with it, cartpole was essential in testing and setting up the logging interface as well as testing different implementations of the learning algorithm

Keras-rl

Keras-rl is a community maintained high level implementation of keras agents for reinforcement learning, this was the first implementaiton tested.

The implementation of keras-rl is very easy and diesn't require a deep understanding of reinforcement learning and the leanring structure.

Keras API

The second implementaiton tested was using the plain keras API, while this provides more flexibility it also requires a much deeper understandng of how reinforcement learning works. The implementaiton using the Keras API was essential to develop a necessary knowledge for the project and to progress to the next stage.

While an implementaiton using Keras-rl would be simpler and even possibly ease the itteration process, this implementation provides less flexibility and given the target of the project and desire to develop a deeper understanding of reinforcement learning the implementation using the Keras API was choosen to implemnt the next phases.

Results

4.2 2D Environment Outcomes

4.3 3D Environment Outcomes

4.4 Reward Function

Chapter 5

Future Research

5.1 Empowerment

5.2 Reward Function Development

5.3 Policy Gradients

5.4 Mujoco Implementation

5.5 Real Robot Training

Chapter 6

Project Evaluation

Chapter 7

Conclusion

Bibliography

[1] Chipmunk. Chipmunk. http://chipmunk-physics.net.

[2] Alberto Ezquerro, Miguel Angel Rodriguez, and Ricardo Tellez. openai ros. http://wiki.ros.org/openai_ros, 2016.

[3] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. Robocup: The robot world cup initiative, 1995.

[4] OpenAI. Cartpole. https://github.com/openai/gym/blob/master/gym/envs/classic_control/cartpole.py, 2016.

[5] OpenAI. Introduction to rl. https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html, 2018.

[6] Pygame. Pygame. https://www.pygame.org.

[7] Pymunk. Pymunk. http://www.pymunk.org.

[8] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning, second edition: An Introduction*. MIT Press, Nov 2018. Google-Books-ID: uWV0DwAAQBAJ.

[9] Alec Jeffrey Thompson and Nathan Drew George. Deep q-learning for humanoid walking, Apr 2016.