Q-Learning example using Unity Machine Learning Agents with AiGym interface  
Patryk Stradomski s16716

Table of Contents

[Introduction 2](#_Toc70371686)

[Goal of this paper 2](#_Toc70371687)

[Tools(?) -*Nie jestem pewien czy tu dawac podpunkty bo opisuje je pozniej jeszcze.* 2](#_Toc70371688)

[Summary 2](#_Toc70371689)

[Reinforcement Learning 2](#_Toc70371690)

[Introduction 2](#_Toc70371691)

[Definitions 3](#_Toc70371692)

[Agent 3](#_Toc70371693)

[Environment 3](#_Toc70371694)

[State 3](#_Toc70371695)

[Action 3](#_Toc70371696)

[Markov chain stuff(?) *– nie wiem jak to ladnie zawrzec, moze jako przyklad ?* 3](#_Toc70371697)

[Explore v. Exploit(?) – *czy warto tu wspominac czy moze juz przy implementacji?* 3](#_Toc70371698)

[Q-learning 4](#_Toc70371699)

[What is Q-learning 4](#_Toc70371700)

[Q-table 4](#_Toc70371701)

[Q-Learning v. SARSA 4](#_Toc70371702)

[Gym by OpenAI 4](#_Toc70371703)

[What is Gym? 4](#_Toc70371704)

[Environments 5](#_Toc70371705)

[Examples 5](#_Toc70371706)

[Interface 5](#_Toc70371707)

[Unity ml-agents 5](#_Toc70371708)

[Unity game engine (or Unity Real-Time Development Platform) 5](#_Toc70371709)

[Machine learning with Unity 5](#_Toc70371710)

[Environments 6](#_Toc70371711)

[Q learning implementation using ml agents – ai gym hybrid.? 7](#_Toc70371712)

[Basic environment 7](#_Toc70371713)

[How to create a unity executable 8](#_Toc70371714)

[Using Ai gym with unity executable 9](#_Toc70371715)

[UnityEnvironment 9](#_Toc70371716)

[UnityToGymWrapper 9](#_Toc70371717)

[Algorithm implementation 9](#_Toc70371718)

[Experiments 9](#_Toc70371719)

[Conclusions 9](#_Toc70371720)

[References 9](#_Toc70371721)

# Introduction

## Goal of this paper

The goal of this paper is to present the ease of integration of reinforcement learning solutions with an environment prepared in Unity through Gym interface introduced by OpenAi.

## Tools(?) -*Nie jestem pewien czy tu dawac podpunkty bo opisuje je pozniej jeszcze.*

* Gym by OpenAi
* Unity
* Unity ML Agents
* Python 3.7

## Summary

# Reinforcement Learning

## Introduction

[ (Sutton & Barto, 2014, 2015)]

Reinforcement learning is a field of machine learning that focuses on maximizing rewards while performing a certain task or navigating a certain environment.

Reinforcement learning should feel very familiar and intuitive to most people since it is the most ‘natural’ way of learning. Sutton in his book mentions as an example how a child learns and interacts with its environment without any tutorage.

## Definitions

### Agent

We define an Agent as an entity capable of performing actions that influence the environment or its position therein. It’s acting based on a given policy – a set of instructions defining its behavior in a given state.

### Environment

Environment is a certain situation in which an agent is placed. It has it’s own set of rules depending the scope, the agent’s possible actions and the reward function associated with those actions

### State

State describes a snapshot of the environment, agent, and their relation to each other in a given moment. When agent performs an action, it can change the state

### Action

Reinforcement learning is fundamentally based on interaction.

## Markov chain stuff(?) *– nie wiem jak to ladnie zawrzec, moze jako przyklad ?*

## Explore v. Exploit(?) – *czy warto tu wspominac czy moze juz przy implementacji?*

# Q-learning

## What is Q-learning

(Hasselt)

Where:

* – represents the learning rate.
* – is the discount factor. If its smaller than one the then rewards received later are valued exponentially(?) less than those received earlier.
* is current state.
* represents the estimate Q value after the most optimal action.

## Q-table

Q-Learning utilizes an idea of a Q-Table – a dataset assigning a so called Q-value to each pair in a cartesian product of action and state. When an agent finds itself in a given state, with a correctly discovered values in its Q-Table, it should pick an action with the highest assigned Q-value.

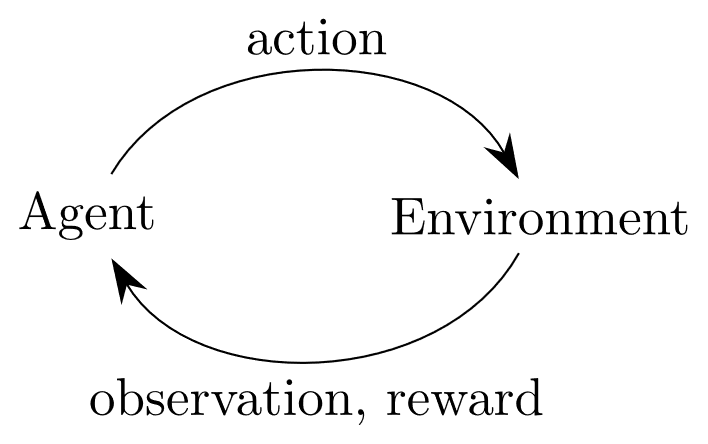
## Q-Learning v. SARSA

SARSA stands for State–action–reward–state–action, it’s a reinforcement learning algorithm Q-learning is SARSAMAX. Q learning is SARSA but with the assumption that the policy for updating the Q-value is based on the maximum possible reward for available actions while for SARSA itself it could be a different policy, for example taking the mean value.

# Gym by OpenAI

## What is Gym?

Gym is a toolkit for developing and comparing reinforcement learning algorithms. It provides a set of standardized problems which can be solved with the use of reinforcement learning. The problems, referred to as *Environments*, encamps a simulation with an agent. The toolkit exposes a concise interface that wraps the environment allowing for an agent to easily influence it. After each tick, the interface can provide us with the following: The current state of the environment – the representation of which varies, the latest reward obtained by the agent, information on whether it’s time to end the current episode, and diagnostic information that can be used to further understand what happened during the last step. The agent can make use of the first three to then provide an action that will influence the environment in the next step.



(OpenAi Gym Documentation) (OpenAI Gym whitepaper, n.d.)

## Environments

### Examples

Gym has a lot of examples on their website.

Some are classical retro games – often used as reinforcement learning problem benchmarks(**citation)** They have some physics problems, they have MUJOCOCO (PAYED SCAM) and some others,

The envs registry can be found in *gym.envs.registry*, import gym and run the snippet below to see it.

>>> from gym import envs  
>>> print(envs.registry.all())

## Interface

Gym provides a simple interface for their environment:

# Unity ml-agents

## Unity game engine (or Unity Real-Time Development Platform)

Unity is a game engine created in \_ by \_

It has over 80 different case studies listed on their site (<https://unity.com/case-study>) created by companies and organizations from a range of industries including: Gaming, Engineering, Automotive, Film and more.

The team behind unity wants to empower everyone with a universal kit that will let them focus on their goal rather than on the essential tools. (https://unity3d.com/whitepapers/adopting-unity)

## Machine learning with Unity

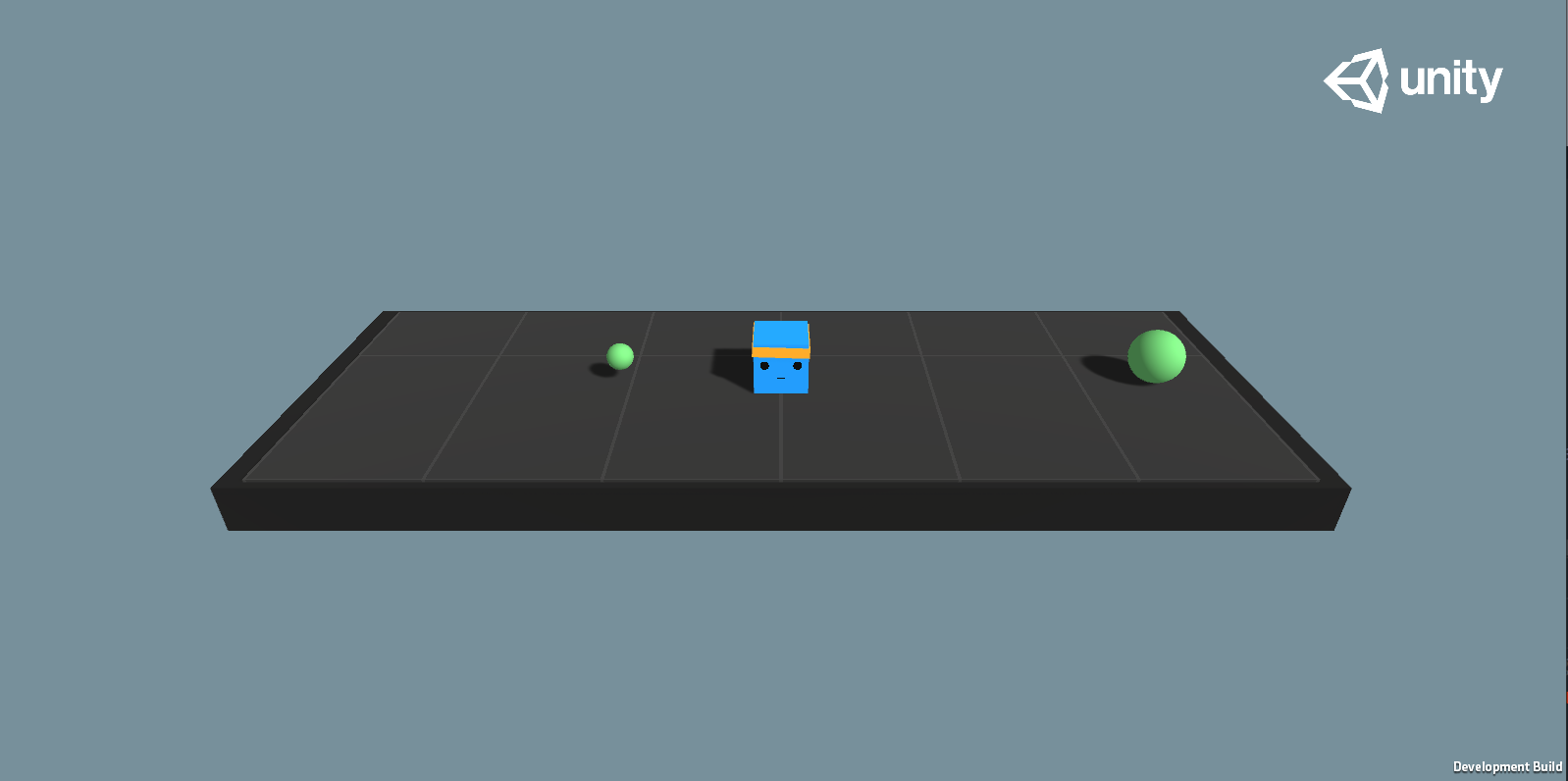
<https://github.com/Unity-Technologies/ml-agents>

The Unity Machine Learning Agents is a toolkit that empowers the creation of games and simulation for the purpose of using them as environments for intelligent agents. It comes with a number of sample environments prepared with sample machine learning solutions that could utilize them. Alternatively, new solutions can be tested against those either by creating policies for the agents or by utilizing a provided python API. It uses. The toolkit first appeared in Beta on Sep 19, 2017[https://github.com/Unity-Technologies/ml-agents/releases/tag/v0.1] and as of April 21, 2021 it’s on its 16th stable release. The project is open source and has a growing community.

## Environments

# Q learning implementation using ml agents – ai gym hybrid.?

## Basic environment

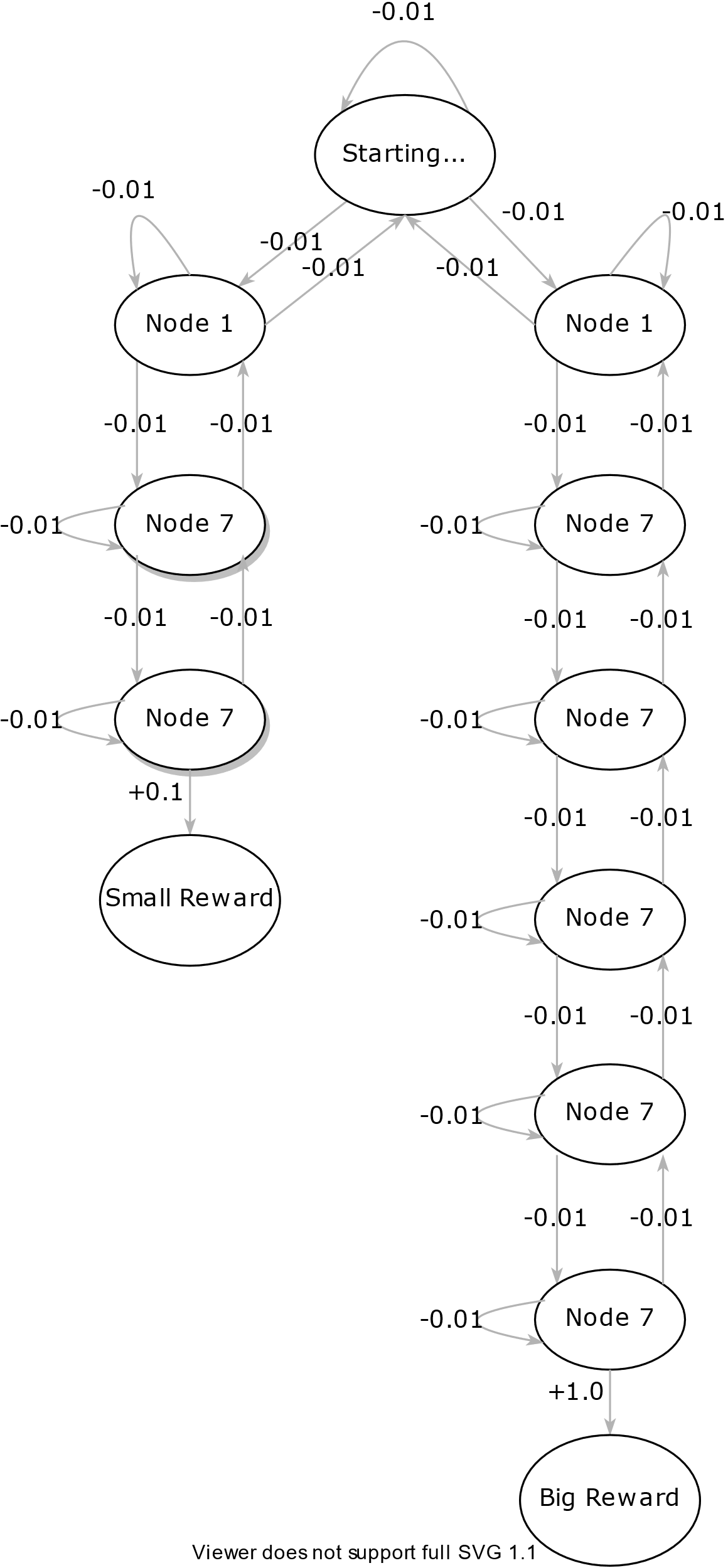
I have chosen the ‘Basic’ environment provided by the ml-agents framework. It contains a single cube that can move left, right, or not move at all. There are two rewards in the level: one small but close and one far bigger but a little further away. The goal is to obtain the most reward state.  


1 "Basic" environment in its initial state

Reward function for the environment:

* +1.0 on arriving on the bigger reward.
* +0.1 on arriving on the smaller reward.
* -0.01 on each step

The documentation(**citation**) lists the benchmark Mean reward as 0.93. According to this benchmark we should expect the agent to arrive at the bigger reward state after 7 steps.

Markov decision process representation of the Basic environment: 

## How to create a unity executable

? Should I write that down?

## Using Ai gym with unity executable

After creating a unity executable with our environment we’re able to import it into our runtime by creating an object of ***mlagents\_envs.environment.UnityEnvironment*** and wraping it into a Gym with the help of ***UnityToGymWrapper*** from ***gym\_unity.envs***.

### UnityEnvironment

On initiation, this object runs the unity environment executable under a provided path and establishes a connection between python runtime and the environment through an unsecured socket. By default, the connection is established on port 5005.

### UnityToGymWrapper

## Algorithm implementation

# Experiments

# Conclusions

# References

Hasselt, H. v. (n.d.). *Reinforcement Learning: State-of-the-Art.* (M. Wiering, & M. v. Otterlo, Eds.) Springer Science & Business Media. Retrieved 3 27, 2021

Juliani, A., Berges, V.-P., Teng, E., Cohen, A., Harper, J., Elion, C., . . . Lange, D. (2020, May 6). Unity: A General Platform for Intelligent Agents. *2*. Retrieved from https://arxiv.org/abs/1809.02627

*OpenAi Gym Documentation*. (n.d.). Retrieved March 24, 2021, from https://gym.openai.com/docs/

*OpenAI Gym whitepaper*. (n.d.). Retrieved from https://arxiv.org/abs/1606.01540

Sutton, R. S., & Barto, A. G. (2014, 2015). *Reinforcement Learning: An Introduction.* Cambridge, Massachusetts: The MIT Press.