

**Faculty**

**Department**

Specialization

**Thesis author**

Student identification number

**Sarsamax implementation**

**with OpenAI Gym and Unity engine**

Thesis type (e. g. BSc, MSc, B.Eng., Project)

Thesis supervisor

Warsaw, month, year

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 6](#_Toc70371708)

[Unity game engine (or Unity Real-Time Development Platform) 6](#_Toc70371709)

[Machine learning with Unity 6](#_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)

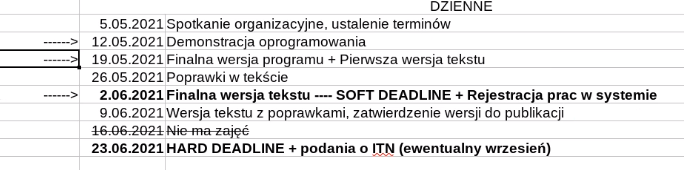
[UnityToGymWrapper 9](#_Toc70371717)

[Algorithm implementation 9](#_Toc70371718)

[Experiments 9](#_Toc70371719)

[Conclusions 9](#_Toc70371720)

[References 9](#_Toc70371721)



# Introduction

## Goal of this paper

Researchers working on reinforcement learning should be able to spend most of their time and resources on implementing the solution rather than on preparing environments and tools.  
The paper will present two tools that can be integrated in order to streamline the development process.

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

## Tools

* 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. In reinforcement learning the child would be considered an **agent** working in some **environment** trying to feel out its surrounding by performing different **actions** like touching or tasting different objects.

Slowno muzycznie opisz reinforcement learning, jego historie I wplesc w to te definicje np dajac bold na **agent**

## Definitions

### Agent

We define an Agent as an entity capable of performing actions that influence the environment or its position therein. It is 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 its own set of rules describing 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. Action is an interaction between the agent and the environment that may result in a changed state. Reward function provides a specific reward to the agent corresponding to its performed action.

## Decision Processes (Markov chain)

## Explore v. Exploit

# SARSA & Q-learning

## What is SARSA

SARSA stands for State–action–reward–state–action, it is 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.

## What is Q-learning

Q-learning or SARSAMAX is an algorithm relying on the same principle as SARSA but it’s algorithm for choosing the best value is to choose the max value.

Definitioon first then after explaining the equation you can show the equation – it has to be put in context so that you can understand it on the spot. F ex New Q value Is determined by …

(Hasselt)

Where:

* – represents the learning rate.
* – represents the reward.
* – 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 based on the q-value corresponding to that action. In sarsamax this will be the action which has accumulated the highest q-value.  
**Przyklady**

# OpenAI

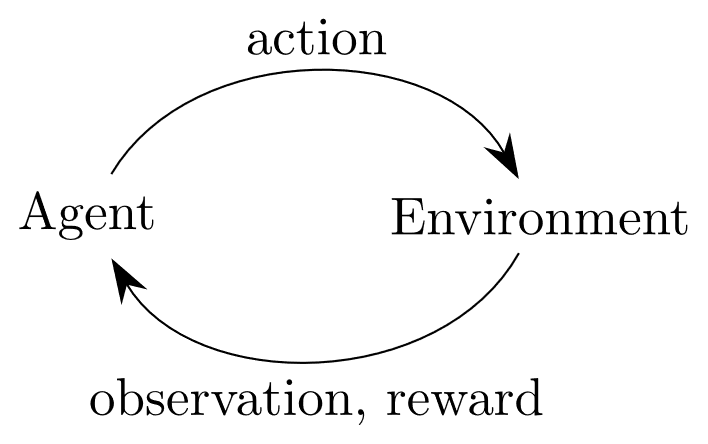
OpenAi is an organization that focuses on AI research. It was created in December 2016 as non-profit research company trying to develop an ethical AI solutions that would serve everybody. It has since developed several products both open-source and proprietary.

OpenAI’s mission is ensuring that artificial intelligence serves all of humanity. To meet this goal the organization intends to develop and help advance beneficial autonomous systems with its long-term safety in mind. The founders were concerned that the AI race may become so competitive that some could be willing to cut corners when it comes to safety precautions. They pledge that if some organization with similar views will come close to creating an artificial general intelligence before OpenAi, the company will stop competing against them and instead start supporting their project to ensure that no conflict of interest would jeopardize the systems’ safety.

# OpenAI Gym

## 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 is 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

Gym boasts a considerable number of premade environments.

Their website lists them in the following categories:

* Algorithms – Simple computations or logic actions such as addition or reversing of symbols provided by the environment. While such a task would be trivial for a computer, the goal is to have the agent learn by examples, very much like a person would.
* Box2D – 2-dimensional simulations for both discrete and continuous control tasks.
* Classic control – A set of reinforcement learning problems previously proposed in literature. Notable examples include the cart pole problem(the reverse pendulum) and the mountain car example which can be found here https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-209.pdf
* Atari – an integration of the Arcade Learning Environment(<https://github.com/mgbellemare/Arcade-Learning-Environment>), numerous Atari game emulations can be found in this package with most if not all offering the on screen image as the environments observation. An agent performing in such an environment would be no different to a person playing the very same game on the console- having visual observation as an input and pressing appropriate buttons as actions.
* MuJoCo – 3d simulations prepared in MuJoCo physics engine. The name stands for Multi-Joint dynamics with Contact.
* Robotics -
* Toy text

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())

### Mujoco

Why bad why good what can we do…

## Interface

Gym provides a simple interface for their environment.  
Having started the environment, an order to perform an action can be sent to the agent with the following method:

observation, reward, done, info = env.step(action)

The action is performed, provided that it matches the environments action space, and in turn the environment returns four variables:

* Observation: list - agent's observation of the current environment
* Reward: float – the value that has been awarded to the agent after last action
* Done: Boolean – specifies whether the environment is ready to be reset. Returned as true in cases like the agent arriving at its final destination or failing beyond fixing.
* Info: Dict – diagnostic information on the environment, irrelevant from the algorithms point of view.

# 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>

https://blogs.unity3d.com/2020/05/12/announcing-ml-agents-unity-package-v1-0/

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 is on its 16th stable release. The project is open source and has a growing community.

## Environments

The ml-agents package comes with 17 example environments each designed with a different task, reward function and proposed solution. Should I give examples?(yes, a little) However, the great thing about ml-agents is that the provided environments are just an example of what can be achieved. The documentation provides a thorough guide on creating new environments. [https://github.com/Unity-Technologies/ml-agents/blob/main/docs/Learning-Environment-Create-New.md] This gives people the freedom to design simulations fit to their research.

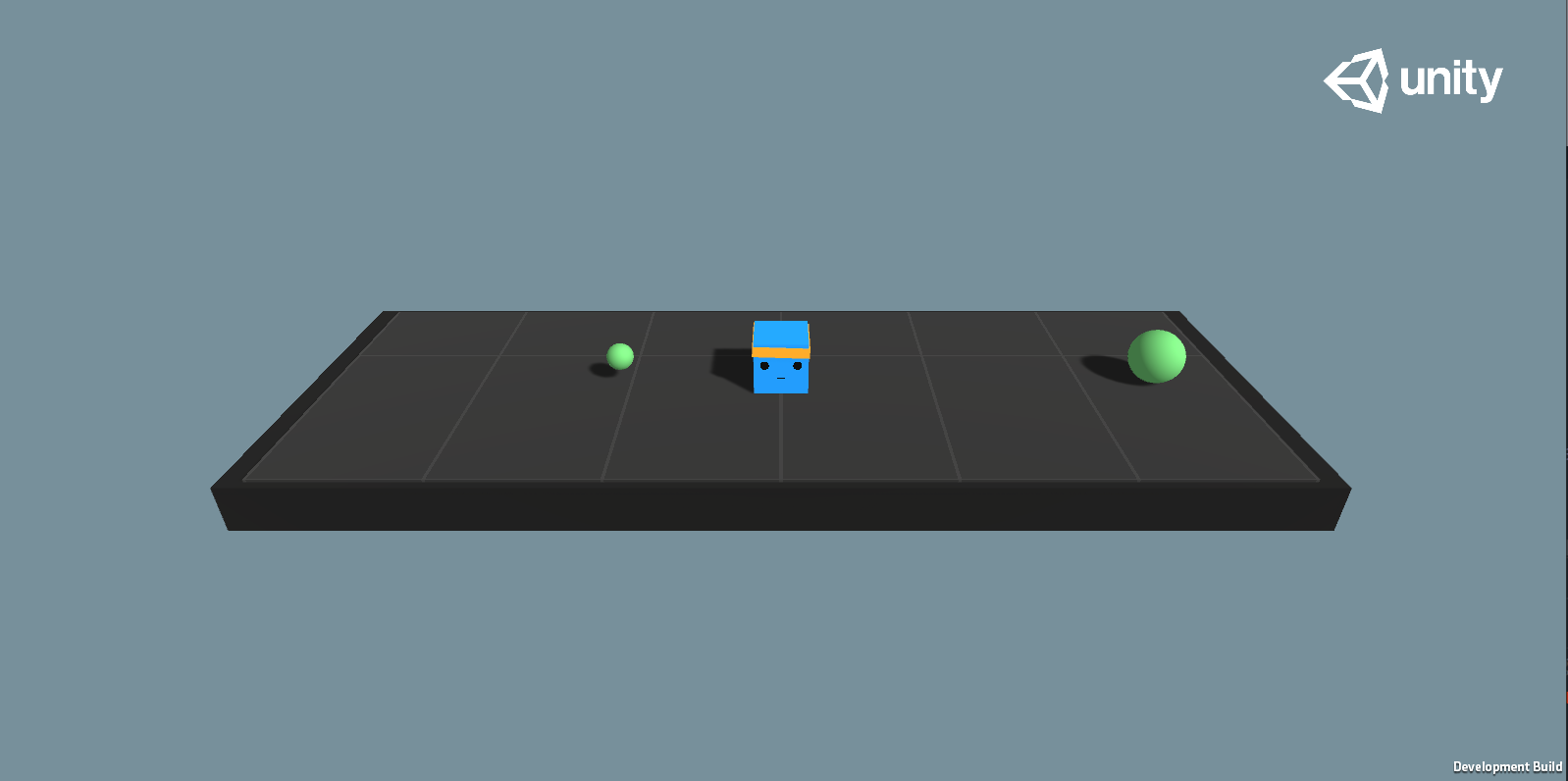
Robotic simulation (mujoco sth) https://github.com/Unity-Technologies/articulations-robot-demo/tree/mlagents

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

## Basic environment

I have chosen the ‘Basic’ environment provided by the ml-agents framework 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.

The agent, represented by the blue cube, start is placed in a 1-dimensional space that holds 20 positions. The agent starts at the eleventh position from the left while the small and big reward are on the positions 8 and 18, respectively.



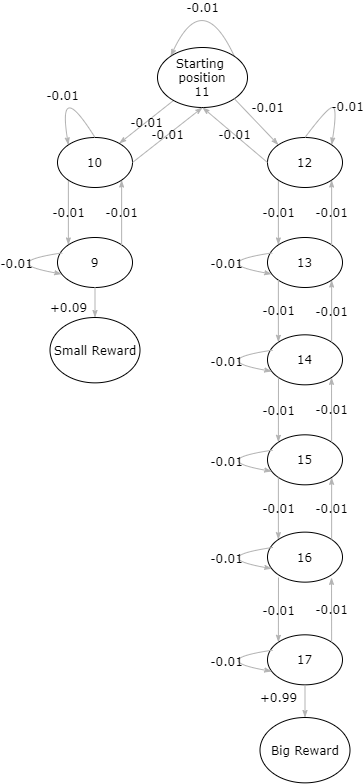
1 "Basic" environment in its initial state

Reward function for the environment:

* +1.0 on touching the bigger reward.
* +0.1 on touching 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.  
To achieve the lower reward however, the agent needs to only perform 3 steps. The path is more than 2 times shorter, but the reward is ten times smaller.

Markov decision process representation of the Basic environment



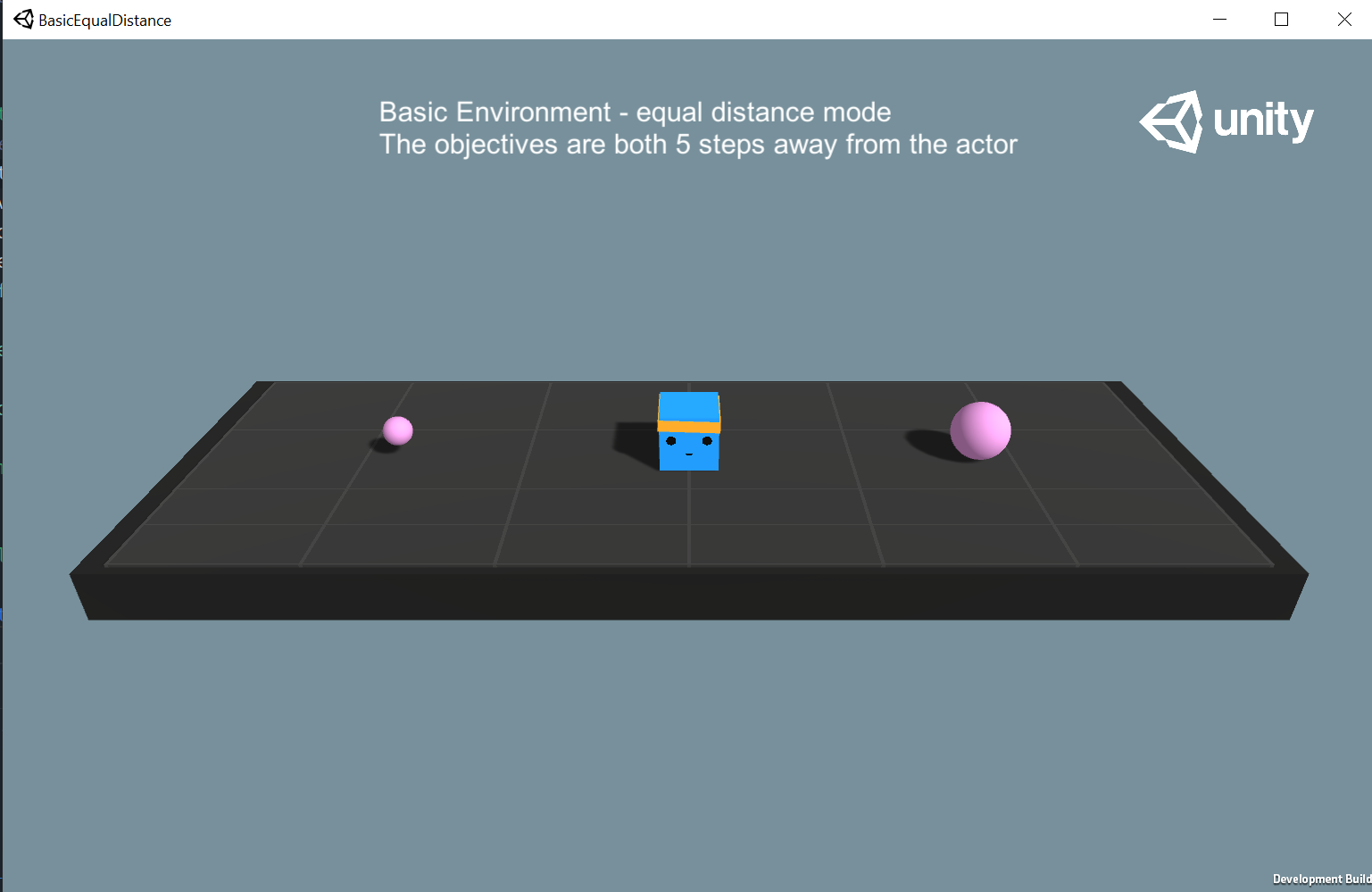
# Experiments

The original run had the following arguments:  
  
And resulted in this:  
  
Equal distance to the rewards.

In the ‘Basic’ environment provided by Unity the agent has to learn to choose the reward even though it is twice as far as the small reward. This can lead to a situation where the agent gets stuck on the local maximum and decides to always go for the small reward.

However, if the goal were to check the very obvious assumption that the actor will always go for the bigger of two rewards then it could be more reasonable to prepare an environment that will check just that.

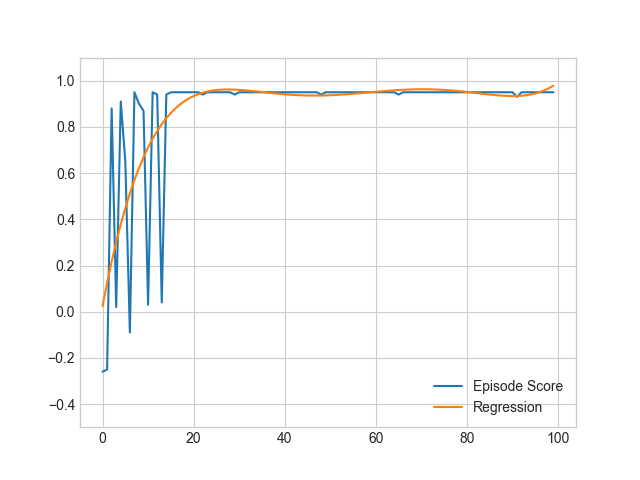
Unity provides us with a way to easily create new environments and alter existing ones. Using the intuitive UI, I was able to quickly alter the position of the rewards, their color, and add a description of this mode.

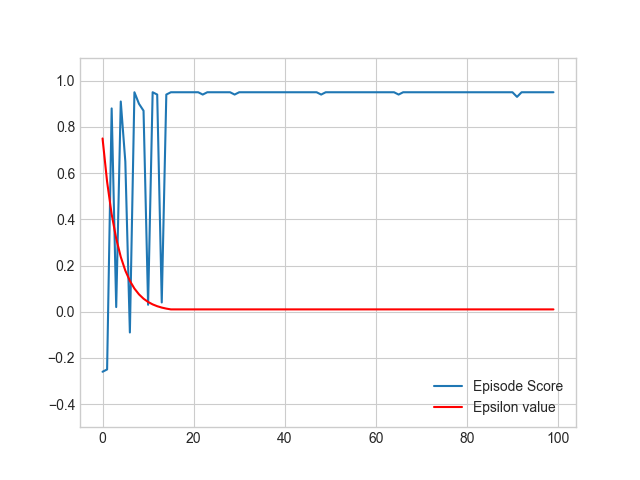


After building the executable I performed the Sarsamax training with the following attributes:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Discount factor | Learning rate | Beginning epsilon | Epsilon decay | Minimum epsilon |
| 0.95 | 0.1 | 1 | 0.75 | 0.01 |

The Q-table has been initialized with all zeros and the training run for 100 episodes.







# 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

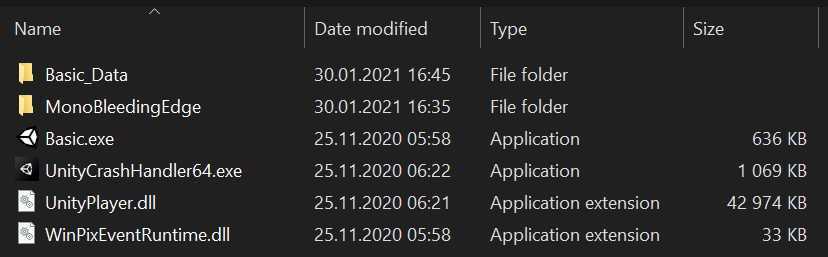
Sutton, R. S., & Barto, A. G. (2014, 2015). *Reinforcement Learning: An Introduction.* Cambridge, Massachusetts: The MIT Press.

# Appendix

Write a tutorial on what u did.

## How to create a unity executable – to appendix

After building the project we receive the following directory:



## Using Ai gym with unity executable – to appendix

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 wrapping it into a Gym with the help of ***UnityToGymWrapper*** from ***gym\_unity.envs***.

### UnityEnvironment

https://github.com/Unity-Technologies/ml-agents/tree/main/ml-agents-envs

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 – to appendix

Should I put code here or just describe it?