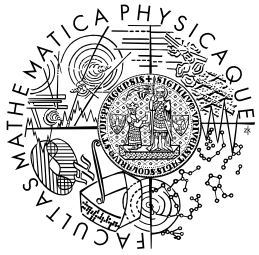


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Bc. Přemysl Bašta

Thesis title

Department of Theoretical Computer Science and Mathematical Logic

Supervisor of the master thesis: : Mgr. Martin Pilát, Ph.D.
Study programme: Computer Science
Study branch: Artificial Intelligence

Prague 2023

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

Dedication.

Title: Thesis title

Author: Bc. Přemysl Bašta

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: : Mgr. Martin Pilát, Ph.D., Department of Theoretical Computer Science and Mathematical Logic

Abstract: Abstract.

Keywords: key words

Contents

Introduction

In recent years people are witnessing fast progress in artificial inteligence(AI) in all sort of domains. Beating world champions in chess or Go is no longer any issue for AI models. Same pattern is showing in more recent popular games such as Dota (OpenAI [2019]) or Starcraft, where even great Human-AI team cooperation behavior has been achieved. However, all of these examples share common property of being competetive. Ultimate goal of our society is to create AI that will be cooperating with humans, not competing.

Recent work has shown us that AI cooperative models trained together for purely cooperative tasks tend to rely mostly on expect near optimal behavior from their partners and fail to cooperate with partners who don't satisfy this condition.

Which is bad news for us humans as our behavior is rarely optimal.

Great example of human-AI cooperation domain where humans do not always perform perfectly are self driving cars. In a situation where an accident is imminent humans have to react quickly without having enough time to consider all possible reactions or even analyze entire current road situation. However car accidents maybe even too extreme example of human unoptimal behavior. People often fail at even simplier task of obeying the standard traffic rules when having enough time to react. We can imagine how predicting human behavior is not an easy task for self driving car.

In this work we will firstly revisit definition of Markov decision process, building block of reinforcement learning, then mention basic aproaches of Q-learning. We will focus on policy branch of reinforcement algorithms, mention their variants and conclude with policy learning algorithm Proximal policiy optimization which is considered as state of art algorithm masively deployed in many succesful projects.

We will utilize simplified cooperative cooking environment based on popular video game Overcooked, where two partners are forced to coordinate shared task of cooking and delivering soup to customer. Here we are going to summarize what aproaches have been tested in related work in terms of ad-hoc agent cooperation. Some of which will we reimplement for our evaluating purposes. We mention problem of definition of robustness of agent cooperation.

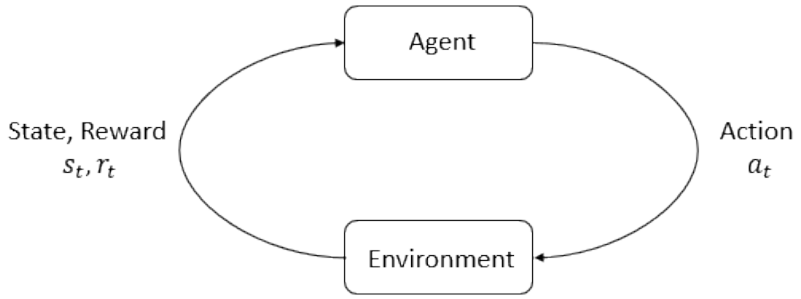
And finally we contribute with our ideas of diversificated partners population.

1. Introduction to reinforcement learning

1.1 Markov decision process

Environment cycle

Whole problem of reinforcement learning (RL) can be best described by following visualisation.



Environment represents some kind of world with its inner rules and properties. Agent is then an entity that exists inside this world, observes **state s** of the world, decides on the basis of this state to react with **action a** and as consequence of this action recieves **reward r**. Entire mechanism of this environment can be then broken into these cycles of states, actions and rewards. The goal of an agent is to interact with environemt in such a way to maximize its cumulative reward.

In order to be fully able to formalize problems of reinforcement learning we have to expand further our notation.

Observability

State s contains all information about environment at given time. However, agent in some environment can percieve **observation o** where some information about environment can be missing. In that case we say environment is **partially observable** as opposed to **fully observable** environment where agent has available entire information at it's observation.

Actions and policies

Environments can also differ from point of what actions are possible inside of given world. Set of possible actions is called **action space** which again can be divided into two types. **Discrete** action space contains finite number of possible actions. And **continuous** action space which allows for action to be any real-valued number or vector.

Agent's action selection can then be described by a rule called **policy**. Common notation is established that if the action selection is deterministic, we say policy is **deterministic** and denote

$$\boldsymbol{a_t = \mu(s_t)}.$$

If policy is **stochastic** it is usually noted as

$$\boldsymbol{a_t} \sim \boldsymbol{\pi}(\cdot|\boldsymbol{s_t}).$$

Policies are main object of interest of reinforcement learning as this action selection mechanism of an agent is what we are trying to learn. Policy, for optimization purposes, is function often **parametrized** by a neural network whose parameters are usually denoted by symbol $\boldsymbol{\theta}$, therefore parametrized deterministic and stochastic policy are represented by symbols $\boldsymbol{\mu_\theta(s_t)}, \boldsymbol{\pi_\theta(\cdot|s_t)}$ respectively.

Trajectory

Next important definition is notion of trajectory also known as episode or rollout. Trajectory is a sequence of states and actions in an environment.

$$\tau = (s_0, a_0, s_1, a_1, ...)$$

Initial state s_0 of an environment is sampled from **start-state distribution**. Subsequent states follow transition laws of environment. These can be again deterministic

$$s_{t+1} = f(s_t, a_t)$$

or stochastic,

$$s_{t+1} \sim P(\cdot|s_t, a_t)$$

Return

We already mentioned agent aspiration of maximization of cumulative rewards. Now we combine it with trajectories and derive formulation of **return**.

$$\boldsymbol{R(\tau)} = \sum_{t=0}^{T} r_t$$

for finite-horizon. And infinite-horizon discounted return:

$$\boldsymbol{R(\tau)} = \sum_{t=0}^{T} \gamma^t r_t$$

Discounting infinite-horizon is both intuitive and convinient for mathematical purposes.

$$r_t = R(s_t, a_t, s_{t+1})$$

is a function of states and action. However, this definition is often abreviated only to pair of current state and taken action

$$r_t = R(s_t, a_t)$$

Optimal policy

2. Policy gradient methods

2.1 Comparison with off policy Q learning methods

2.2 Idea, motivation and brief technical description of algorithm

2.3 Variants of policy theorem

Vanilla

PPO

3. Multi agent environments for RL ???

3.1 Definitions

3.2 Possible mention of MAPPO success

3.3 Cooperation harder than competition

4. Overcooked environment

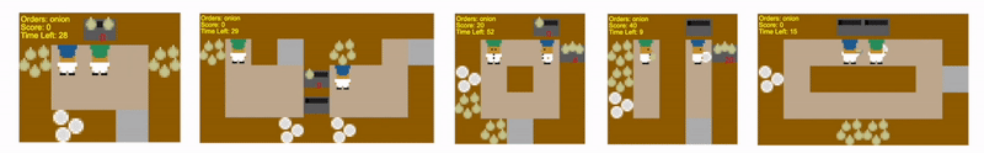
4.1 Overcooked game

Before we get into our problems with cooperation let us first examine the environment. We will be working with environment based on popular cooking video game `https://ghosttowngames.com/overcooked/`. Overcooked is multiplayer cooperative game where the goal is to work in a kitchen as a team with partner cooks and prepare together various dishes within limited time. However, the game is dynamic to a great extent. In many maps the kitchen itself is not static and may be changing on a run. Moreover, random events such as pots catching fire add to the chaos. The challenge lies in coordination with rest of the team and dividing subtasks efficiently.

The aforementioned game was simplified and reimplemented to simpler environment `https://github.com/HumanCompatibleAI/overcooked_ai` to serve a purpose of scientific common ground for studying multi agent cooperation in somehwat complex settings. Lot of additional features of original game were removed and remained only essential coordination aspects. In its simplest form, environment is taking place in small static kitchen layout where only available recipe is onion soup which can be prepared by putting three onions in a pot and waiting for given time period. Somewhere in the kitchen there is unlimited source of onions and dish dispenser, where player can grab a dish to carry cooked onion soup in to the counter. Team of cooks is rewarded as team by abstract reward of value 20 every time cooked soup is delivered to the counter. It may seem that the task is quite straightforward. However, players face problems on multiple levels.

4.2 Basic layouts

Although the Overcooked implementation has its own generator that can be used to generate new random kitchen layouts, the majority of the related scientific work has so far experimented with a fixed set of predefined layouts, where each of them capture some important aspect of coordination.



(From left to right: Cramped room, Assymetric advantages, Coordination ring, Forced coordination, Counter circuit)

Cramped room as a name suggests represents cramped kitchen layout where all important places are relatively easy to reach. Challenge lies in low level coordination of movement with the other partner as there is no spare room.

In Assymetric advantages both players are located in separated regions where each region is fully self-sustaining. However, each region has better potential for specific subtask. And it is only when both players make the most of their own region's potential that the maximal shared efficacy is reached.

The Coordination ring is another example of a layout where clever coordination is required as the only possible movement around the kitchen is along a narrow circular path that can be used in a given direction. For example, if one player decides to move in clockwise direction, the other player would automatically get stuck if persuing counter-clockwise movement.

Forced coordination kitchen layout is significantly different from others. In this layout, each player is located in a separate region where neither player has all the resources necessary to prepare a complete onion soup. Thus, players are forced to cooperate with each other with the resources they have.

In the last layout, the situation may look similar to the coordination ring. However, in this case, carrying onions around the entire kitchen is highly suboptimal no matter which direction the players choose. To deliver onions efficiently, players must pass them over the counter to shorten the distance. However, the cooks still need to decide who will be responsible for bringing the plates.

4.3 Environment description

4.3.1 Actions space, episode horizont, shaped rewards, state representation MLP vs CNN

4.3.2 Reset state static position, index switching, Randomization function

5. Related work

5.1 Human-ai cooperation results

5.1.1 Human cooperation

Most of the previous work in this area has focused on one of two types of coordination. The first being coordination between a human and an AI partner. And second, focusing solely on the fully AI-driven pair.

While perfect AI-human coordination is generally a more desirable goal to achieve in all sorts of domains, it will not be our main focus. Several previous scientific papers have addressed this issue. A particularly noteworthy contribution is the article On the Utility of Learning about Humans for Human-AI Coordination `https://arxiv.org/abs/1910.05789`, whose authors are also responsible for creating the overcooked environment implementation. They collected many human-human episodes and incorporated these experiences in the form of human behavior clone models into the training. Their main conclusions were that AI models often rely heavily on the optimal behavior of their partner. However, when such a model is paired with generally suboptimal human behavior, it often fails to cooperate at all, regardless of the approach used for training of the AI model.

One of the other important conclusions they came to was that even AI-AI coordination often fails when models are paired with another AI model trained using a different approach. We will discuss some of these popular approaches in the next section.

TODO: How much further focus on human-ai coordination if not our interest?

5.2 Problem of robustness

5.2.1 Problem of robustness definition

Ad hoc agent playing? Trivial states failure (unit-test based aproach)?

5.3 AI-AI coordination

5.3.1 Aproaches

Self-play, Population

5.3.2 Results

It fails

6. Our work - Preparation

6.1 Utilized framework

Comparision of rllib and StableBaselines3

Rllib framework was used in original paper, however for our usages stable baselines seemed sufficient and reasonably easy to extend. Stable baselines has no explicit support of multi-agent environments.

6.1.1 Modifications of stable baselines

CNN policy wrapper, Partner embedded into environment

6.1.2 NN structure modification

6.1.3 Hyperparameters random search

6.1.4 Randomization function correction

6.2 Self-play

6.2.1 Training

6.2.2 Results

7. Our work - Contribution

7.1 Our definition(s?) of robustness

Probably just average of pair results (non diagonal in case of same sets). Maybe percentage of pairs who surpassed some threshold reward?

7.2 Population construction

7.2.1 SP agents initialization

One agent is not enough?

7.2.2 population partner sampling during training

See if playing with whole population at once differs from one random partner for episode

7.2.3 Final agent training

7.3 Diverzification

maximize kl divergence among population partners policies

7.3.1 Population policies difference rewards augmentation

7.3.2 Population policies difference loss

Conclusion

Bibliography

OpenAI, 2019. URL https://openai.com/blog/openai-five-finals/.

List of Figures

List of Tables

List of Abbreviations

A. Attachments

A.1 First Attachment