LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

INSTITUT FÜR INFORMATIK

LEHRSTUHL FÜR MOBILE UND VERTEILTE SYSTEME

MASTER THESIS

Exploring the impact of markets on the credit assignment problem in a multiagent environment

Zarah Zahreddin

Entwurf vom September 11, 2021

LMU
LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
LEHRSTUHL FÜR MOBILE UND VERTEILTE SYSTEME

MASTER THESIS

Exploring the impact of markets on the credit assignment problem in a multiagent environment

Zarah Zahreddin

Professor:	Prof. Dr. Claudia Linnhoff-Popien
Supervisor:	Kyrill Schmid Robert Müller
Submission Date:	31. December 2021

I hereby affirm that I wrote this Master Thesis on my own and I did not use any other sources and aids than those stated.

Munich, 31. December 2021

. . . .
Signature

Abstract

Hier kommt der Abstract hin. Hier kommt der Abstract hin.

Contents

1 Introduction

- Motivation
- Goal
- Research Question
- Structure

2 Related Work

- Definition of field of research
- Scientific Scope
- Which comparable work in research exists?
- Separation from other works

2.1 Reinforcement Learning

rl components

Sutton and Barto wrote in “Reinforcement learning: An introduction”[SB18] that Reinforcement learning (RL) is based on two components that interact with each other: an environment and an agent, see Figure 2.1. Those interactions take part during a time period with discrete timesteps $t \in \mathbb{N}_0$ until a goal is reached or the ending condition applies. Initially the agent gets a starting environment state S_0, and can processes it to choose and execute an action A_0. This concludes the first timestep. The environment changes based on the action and transitions into the next state S_1. In return the the agent receives the new state with a reward R_1 rating the action A_0. Afterwards the agent proceeds to execute actions which leads to the displayed cycle of Figure 2.1.

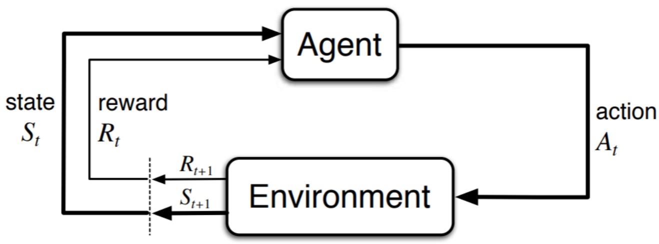

Figure 2.1: The cycle of agent-environment interaction as shown in “Reinforcement learning: An introduction”[SB18]

sets and values

The state S_t is part of a set S containing all possible environment states. Since its likely that not all actions are valid in each environment state the agents action selection is based on a restricted set $A_t \in A(S_t)$. The reward R_t is element of a set of possible rewards R, which is a subset of real numbers $R \subset \mathbb{R}$. Therefore the reward can potentially be negative or very low to emphasize a bad action. The general concept of RL, as defined by Sutton and Barto, is to maximize rewards. However, unlike machine learning approaches the agent starts with no knowledge about good or bad actions and enhances the decision-making by aiming to improve the reward.

policy

Sutton and Barto continue by defining the agents action selection with respect to the current state as a policy π. They explain further that a policy could be as simple as a lookup table, mapping states to actions or it could contain a complicated search process for the best decision. In most cases it is of stochastic nature, mapping actions and states with probabilities. During environment interactions agents gain rewards, which then can be used to update the policy accordingly. For example, should the reward be low or negative it could be interpreted as a penalty. In return the policy $\pi(a \mid s)$ could then be adapted to set a very low probability for that action in combination with that certain state. So next time the agent finds itself in that state the bad action is not very likely to be chosen again.

value function

While rewards only rate the immediate situation, a value function, i.e. the state-value function for a policy $v_\pi(s)$ can be used to estimate the long term value of a state s. The result is the total accumulated reward an agent could get down the line following that state and choosing actions based on the policy π. States that offer immediate high reward could end in low reward streaks. Or the opposite could be the case, where a low reward state could subsequently yield high rewards. Therefore, value functions are of great use to achieve the maximum reward.

exploration vs exploitation

The last part to note about RL is that it entails the problem of balancing exploration and exploitation. In order to learn, an agent has to explore the options given. Since maximizing rewards is the goal, so an agent could become greedy and always choose actions of which is known to result in small but positive reward. If an agent doesn't explore enough the best action sequence will stay hidden and if an agent always explores without exploiting the gained knowledge chances are that the reward will not be optimal.

2.2 Credit Assignment Problem

intro multi problems

Realistic RL scenarios often involve multiple agents solving problems together, for example robots working in warehouses and factories. Such multi-agent environments come with many difficulties. On the one hand in a scenario where agents work independently it is very probable that they get in each other's way in order to score highest or finish a task, preventing the overall goal to be achieved.

In cooperative environments on the other hand, agents share the reward and therefore can not tell who contributed useful actions and who did not. Hence all agents receive the same reward regardless of their contribution which aggravates learning. The independence problem is discussed in chapter 2.3 whereas the cooperation challenge is the focus point of this chapter.

coop and problem

Sutton and Barto [SB18] define a RL environment as cooperative, when agents execute their actions collectively each timestep but receive an overall reward. In this case individual learning is difficult or even impossible. Collective actions may contain bad choices that could be rewarded or, in case of a penalty, good actions that would be punished. Deciding which agent deserves more or less reward, when splitting it up is referred to as the credit assignment problem (CAP) [Min61].

CAP diff kinds

However the CAP originated in a one-agent environment that only rewarded the agent once the goal was reached or the terminating condition applied. A popular example of this is a chess game. In 1961, Minsky [Min61] elaborated on this by explaining that a player wins or loses the game, but cannot retrace which decision got him there.

2.3 Markets

markets and advantage
sm
am
what is expected?

3 Background

- Describe the technical basis of your work
- Do not tell a historical story - make it short

3.1 Proximal Policy Optimization

intro

In 2017 Schulman et al. introduced the concept of Proximal Policy Optimization (PPO) in the article "Proximal Policy Optimization Algorithms"[SWD+17]. This section is solely based on that article in order to explain the Algorithm. Policy optimization is the improvement of the action selection strategy π based on the current state s_t. This is achieved by rotating two steps: 1. sampling data from the policy and 2. optimizing that data through several epochs.

TRPO, Advantage func

The origin of PPO lies in a similar approach called Trust Region Policy Optimization (TRPO). TRPO strives to maximize the following function:

$$\underset{\theta}{maximize}\, \hat{\mathbb{E}}_t[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)}\hat{A}_t - \beta\, KL[\pi_{\theta_{old}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]] \tag{3.1}$$

with $\hat{A}_t$ as an estimator of the advantage function. The advantage function often calculated with the state-value function $V(s)$, a reward r and a discount coefficient λ over a period of Time t

$$\hat{A}_t = -V(s_t) + r_t + \lambda r_{t+1} + \ldots + \lambda^{T-t+1} r_{T-1} + \lambda^{T-t} V(s_T) \tag{3.2}$$

The fraction in the Minuend of (3.1) can be replaced by $r(\theta)$ and represents the probability ratio of an action in the current policy in comparison to the old policy, with θ being a policy parameter. The result of $r(\theta)$ is greater than one, if an action is very probable in the current policy. Otherwise the outcome lies between zero and one. Schulman et al. further describe that TRPO maximizes the "surrogate" objective

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)}\hat{A}_t] = \hat{\mathbb{E}}_t[r(\theta)\hat{A}_t] \tag{3.3}$$

However, maximized on its own without a penalty this results in a large outcome and leads to drastic policy updates.

problem TRPO

In order to stay in a trust region, as the name suggests, a penalty is subtracted from the surrogate function (3.3). The penalty is the Subtrahend of equation (3.1) and contains the fixed coefficient β. Regardless of the function details and outcome of KL, the coefficient β is hard to choose, since different problems require different penalty

PPO

PPO Algo

degrees. Even in a single problem it could be necessary to adapt the coefficient, due to changes within the setting.

Therefore Schulman et al. introduced

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(r(\theta)\hat{A}_t, clip(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{3.4}$$

which is very similar to (3.1) but does not require coefficients. The first part of min contains L^{CPI} (3.3). The second part contains a *clip* function which narrows the space of policy mutation with the small hyperparameter ϵ. After applying the clip function $r(\theta)$ lies between $[1-\epsilon, 1+\epsilon]$. Calculating the minimum of the clipped and unclipped probability ratio produces the lower bound of the unclipped $r(\theta)$, preventing the policy to change drastically.

PPO is defined by the following equation

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)] \tag{3.5}$$

with c_1 and c_2 as coefficients. The authors point out that the loss function $L_t^{VF} = (V_\theta(s_t) - V_t^{targ})^2$ combines the policy surrogate and the value function error term and is necessary once a neural network shares parameters between policy and value function. Finally an entropy bonus S is added to ensure exploration. Schulman et al. continues to show an example of an Algorithm using PPO, see Fig. 3.1. N detonates (parallel) actors collecting data in T timesteps in each Iteration. Afterwards the policy is optimized in K epochs by computing the Loss function (3.5) on the corresponding NT timesteps of data, using a minibatch.

Algorithm 1 PPO, Actor-Critic Style

```
for iteration=1, 2, ... do
    for actor=1, 2, ..., N do
        Run policy π_θold in environment for T timesteps
        Compute advantage estimates Â_1, ..., Â_T
    end for
    Optimize surrogate L wrt θ, with K epochs and minibatch size M ≤ NT
    θ_old ← θ
end for
```

Figure 3.1: Exemplary use of PPO, as shown in "Proximal Policy Optimization Algorithms"[SWD+17]

3.2 Deep Q-Learning

4 Concept

- What is your plan?
- How do you proof that it worked? -> Metric and Experiments

5 Implementation

- How exactly did you do it?
- Experiment parameters
- Experiment setup
- No need to mention framework, software libraries or tools

6 Results

- Result presentation
- Description of images and charts

7 Discussion

- Are the findings as expected?
- Why are the things as they were observed?
- New experiments that provide further insights
- Make your results more comprehensible

8 Conclusion

(Briefly summarize your work, its implications and outline future work)
- What have you done?
- How did you do it?
- What were the results?
- What does that imply?
- Future work

List of Figures

List of Tables

Listings

Bibliography

[Min61] Marvin Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, 1961.

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[SWD+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.