Reinforcement Learning for Trading

Degree programme: BSc in Computer Science
Thesis advisor: Prof. Dr. Erich Baur
Expert: Andreas Fischer (VBS)

Effective trading strategies usually need a deep understanding of finance and market dynamics. We try to circumvent this problem by creating a Reinforcement Learning agent which should learn a policy to maximize the value of a single stock portfolio. Both Q-Learning and Deep Q-Learning are used to estimate this policy.

Introduction

Traditionally, supervised learning methods were used for machine learning agents in trading. This approach has two weaknesses. First, most supervised learning methods used for trading have an information bottleneck between input data and trading action, because of a two-staged procedure: A prediction is made which then results in a trading action based on this prediction. In this case there will be no optimization of a model which maximizes the profit but only the prediction model.

A second problem with supervised learning methods is their inability to model transaction costs. Reinforcement learning solves both problems: The agent tries to learn a policy which directly maximizes the profit and transaction costs can easily be modelled within the environment. Therefore, the goal of our thesis is to implement a Reinforcement Learning framework with an agent which tries to optimize a single stock portfolio. It is not our goal to connect our framework to a real stock exchange.

RL Basics

The goal in Reinforcement Learning is to learn a policy which maps states to actions and maximizes a predefined reward function. The procedure runs as follows: In a period of a finite or infinite number of timesteps, for each timestep, the current facts are mapped to a state which covers the information to learn the policy (stock prices, money held, etc.). Based on this policy, the agent generates an action (for example buy n shares of stock s). Then, the agent is rewarded based on the action taken, which will be used to improve the policy.

Trading-Agent

We designed a modular trading framework so that the following components can be easily exchanged: state generator (part of the environment which creates the state), reward generator (part of the environment which creates the reward), agent (creates actions and learns a policy) and model (neural network which represents the policy function). For each of these components, different combinations are implemented. We designed a Q-Learning and a Deep-Q-Learning agent to learn the policy function. In Q- and Deep-Q-Learning, the agent learns a function which maps states to so-called Q-values. Those values represent how good an action in a particular state is expected to be. In Q-Learning, the policy function is represented by a table, whereas in Deep-Q-Learning a neural network is used.

Some of our improvements seem to be novel concepts when compared to the literature. First, we introduce a new paradigm for the reward function. Traditionally, the reward function rewards only actions the agent has executed. However, there are situations when this classic reward paradigm confuses the agent. Suppose an agent wants to buy stock but has no money to do so. If the price went up the agent would have intended a hypothetically profitable action. Because no action has been executed in the classic reward paradigm, this intention would not be rewarded. We introduce a new reward paradigm where the agent will be rewarded not only for its actions but also for its intentions.

A second improvement we could not find in the literature concerns the update of the Q-function. Traditionally, the update will only be performed for the executed action, while all other values stay the same. We propose an update procedure where the environment computes the reward for all other possible actions and the agent updates its policy with rewards for all these actions. This update procedure will eliminate the necessity of exploration and uses less iterations to learn the optimal policy.

Conclusion

We tested different combinations of agents, reward generators and models in the context of Reinforcement Learning for trading. It seems promising that with an improved configuration, our agents could indeed execute a successful long-term trading strategy.

Remo Hanspeter Hofmann

Lukas Zoller