

Limit order placement with Deep Reinforcement Learning

Learning from patterns in raw historical cryptocurrency market data

Marc B. Juchli

<https://github.com/backender/ctc-executioner>

Thesis committee:

Prof. dr. ir. M.J.T. Reinders

Dr. M. Loog

Dr. J. Pouwelse



1

Welcome to my master defence presentation.

My name is Marc Juchli

The topic of today is: Limit order placement optimization with deep reinforcement learning: learning from patterns in raw historical cryptocurrency market data

In the thesis committee are:

Marco Loog as my supervisor

Johan Pouwelse as my co-supervisor

and Marcel Reinders as an examiner

Contents

- Introduction
- Previous approaches
- Methodology
- Data curation
- Framework construction
- Evaluation
- Results and limitations
- Recommendations and conclusion

The steps above explain the structure of this presentation and their order corresponds with the steps i have taken while proceeding my research

Why limit order placement?

Financial institutions buy or sell assets based on various reasons:

- Customer request
- Fundamental- or technical analysis
- ...

→ Invariable outcome is the decision to buy or sell assets.

Limit order placement (optimization):

the way how to attempt to make a purchase or sale

- aims for best possible price for trader

But first, why limit order placement and what is it?

Financial institutions buy and sell assets based on various reasons, this can be as simple as a: customer requests or of more complex nature such as a fundamental or technical analysis, and the like.

However, their shared and invariable outcome is the decision to buy and sell assets.

Limit order placement therefore concerns about the way how to make these purchases and sales.

And thereby we aim for the best possible price to be paid or received.

Why cryptocurrency data?

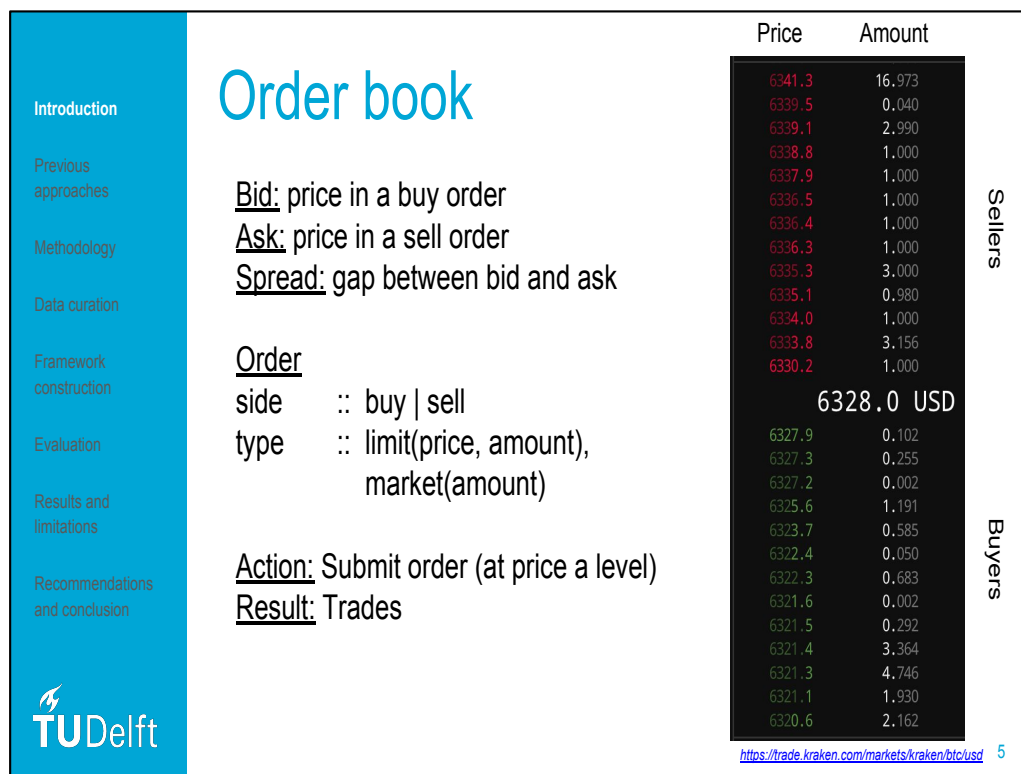
- Is traded in the same way as other currencies and securities (shares, bonds, etc.)
- Market data is free and available in real time
- Volatile market prices
- Low entry barrier → Many inexperienced traders
- Personal interest

I chose for the cryptocurrency domain because data is freely available in real time and cryptos are traded in the same way as other currencies or common securities. Furthermore crypto market prices are very volatile and therefore i hoped for great optimization potential.

Specifically I choose for the bitcoin as this has the highest market volume.

In addition, the low entry barrier for trading cryptocurrencies attract many inexperienced traders, which would greatly benefit from optimized buying and selling capabilities.

Last but not least it was also in my personal interest.



Before i can state my research question, i will provide a brief introduction of some of the components involved in the limit order placement process.

This graphic shows a screenshot of a real order book.

It is divided into buyer and seller side.

Each side lists the offerings of other traders in form of price and amount these parties are willing to buy and sell.

We call the difference between the bid prices on the buyer side and the ask prices on the seller side the spread.

And this is the last traded price.

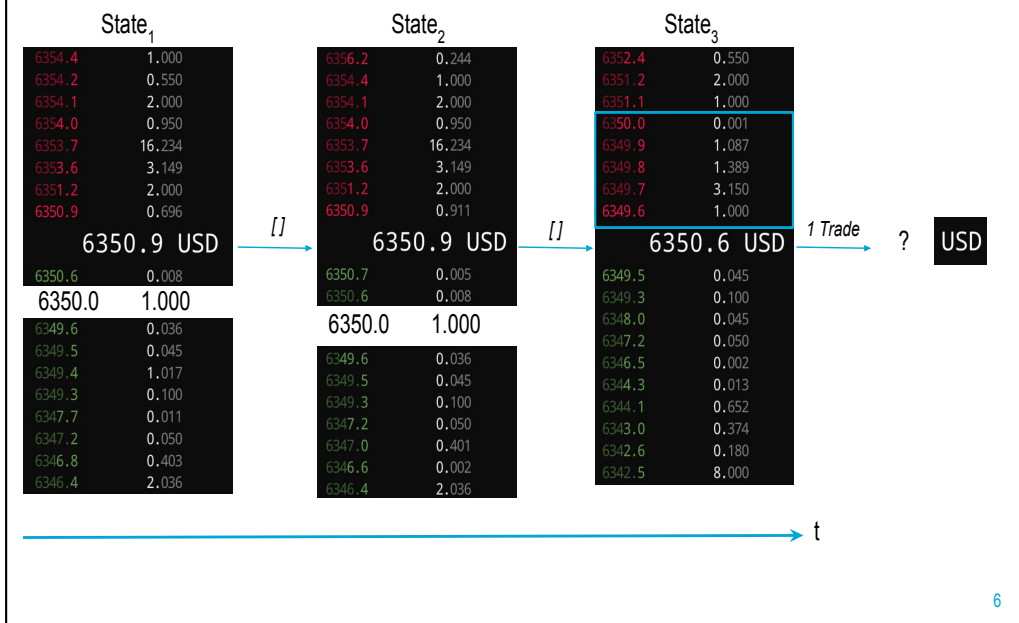
In order to make an entry in the order book we have to submit an order.

We have to choose whether we want to buy or sell and in which way we want to do that.

The two most popular order types are: limit order, where we specify the price and amount

the market order where we only specify the amount and therefore are willing to buy or sell at the best available price.

Limit order matching process



Lets now make an example order submission and see what happens during the matching process.

Assuming we submit a limit order to buy 1.0 bitcoin at price 6350.0, we would be listed here on the buyer side.

We then wait for other traders to hopefully sell at this price.

In the first few seconds noone is willing to sell, instead someone is now willing to buy for a higher price.

So we wait a bit longer and because we are lucky, the market price moved down and suddenly there are traders willing to sell at 6350.0 and better.

What do you think, at which price will our trade be executed?

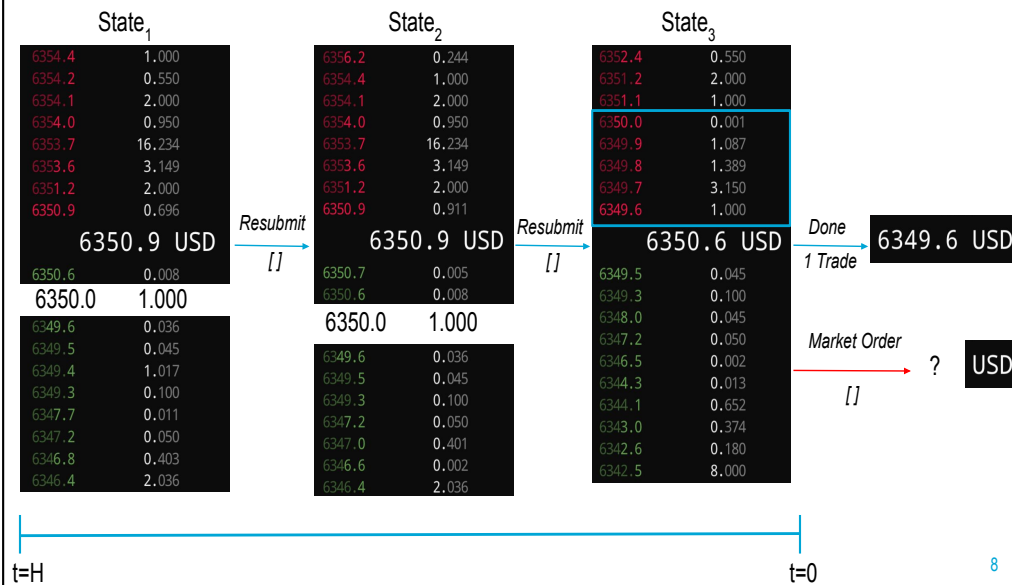
Limit order matching process



Limit order placement (optimization)

Fixed time horizon (H=100 seconds)

Fixed inventory (I=1.0 BTC)



Limit order placement is actually the same process, but allows us to intercept.

We can define a certain time after which we can replace the order at a different price level.

All that matters is that the order is fully executed after a time horizon between 10 and 100 seconds.

It also requires to define a fixed inventory of assets to be bought or sold, which we fix to 1.0 bitcoins.

That also means, if the price would not have developed in such a convenient way and no trade would have resulted after 100 seconds, we are forced to buy our inventory using a market order.

Introduction

Previous approaches

Methodology


Data curation

Framework construction

Evaluation

Results and limitations

Recommendations and conclusion



Research objectives

How can the application of deep reinforcement learning contribute to the optimization of limit order placement?

1. Which historical market data patterns drive market participants to buy or sell assets, and how can these patterns be incorporated into features used by a deep reinforcement learning agent?
2. How should one design a reinforcement learning environment and agents, in the context of order placement?
3. How can one evaluate a reinforcement learning environment and agent in the context of order placement?
4. In which way do the previously constructed features enable a reinforcement learning agent to improve the way it places orders?

9

My main research question is: how to deep reinforcement learning contribute to the optimization of limit order placement?

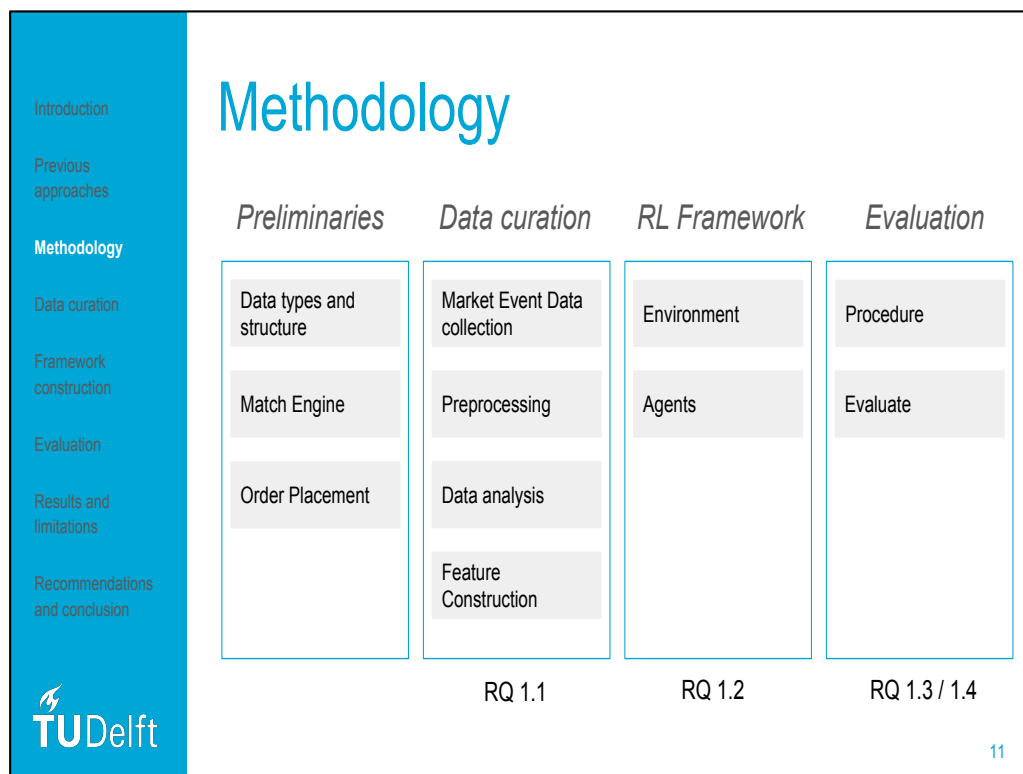
I decided to split this question into subquestions.

First, can we find patterns in market data data drive traders to buy or sell assets, and how can these patterns be incorporated into features to be used by a deep reinforcement learning agent.

Having these features, I asked myself how to design a reinforcement learning environment and agent to make the transposition of this financial problem into the reinforcement learning context.

In a next step, then I asked how can we evaluate this reinforcement learning framework?

And last but not least, in which way do the features enable a reinforcement learning agent to learn and improve the way of order placement.



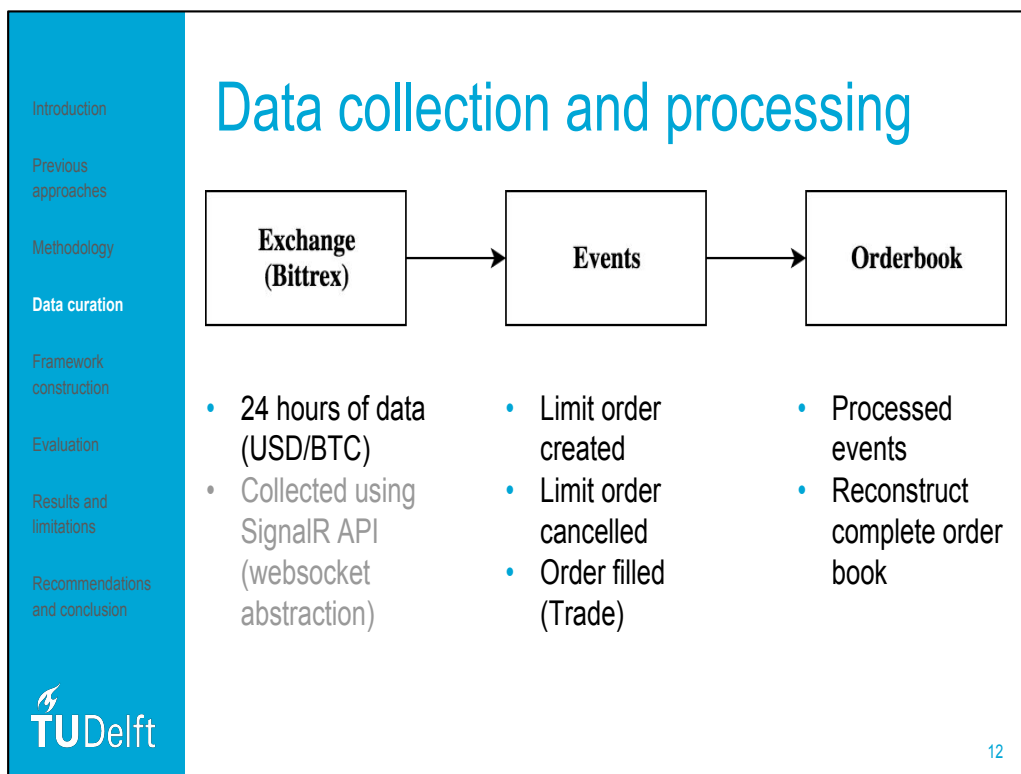
I will now carry out the methodology chosen for my research, however, I will not go into detail as this will become clear over the course of this presentation.

Before I could attempt to work on my research questions, I had to do preliminary work related to the many components involved that were required in all the subsequent steps.

Having this completed I worked on the data preparation and preprocessing part.

After which I built the reinforcement learning framework.

Followed an evaluation of my setup.



Regarding the data collection.

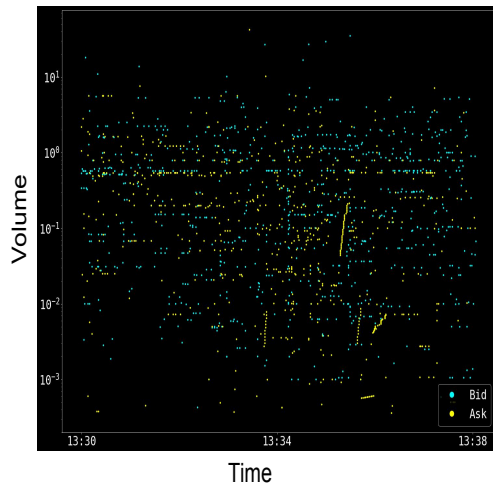
This was a straightforward process after I have understood what data was actually required.

I have understood that only way how to reconstruct a complete order book is to consider all market events.

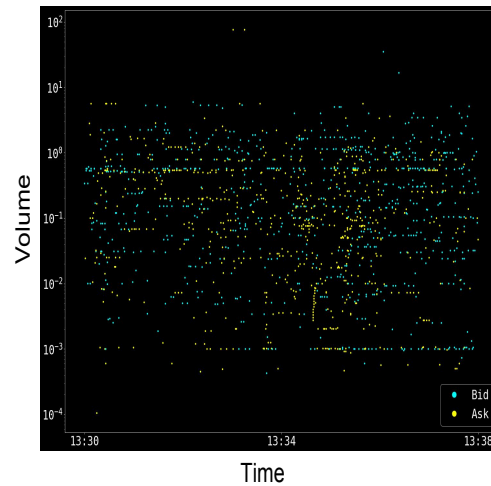
Therefore I collected market events over 24 hours from the Bittrex cryptocurrency exchange and processed them to an order book.

Data analysis: Orders

Volume of created orders

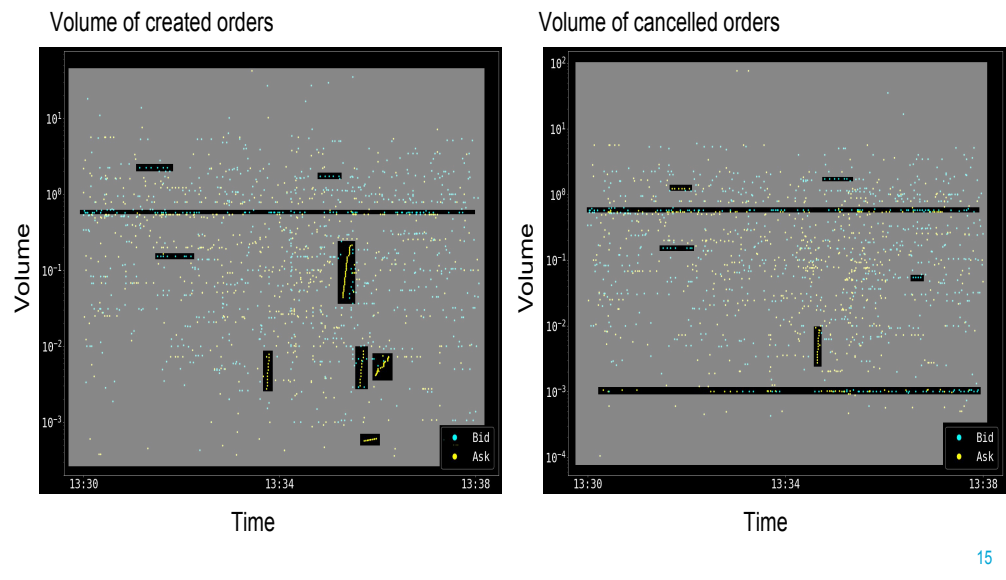


Volume of cancelled orders



I was able to visualize market events using volume maps.
The first category of these events to look at were orders.
Here is an example of the volume of created and cancelled orders over 10 minutes.

Data analysis: Orders



Looking at this in more detail allowed me to find patterns evolved through the posting of these orders by other traders.

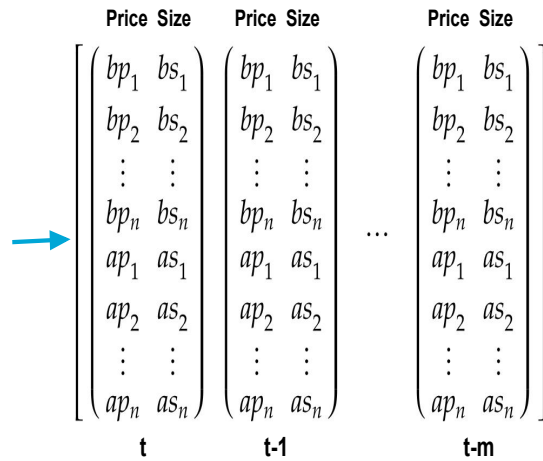
For example consecutive posting of buy and sell orders here.

But also posting of orders with increased volume. This is what we call a buy or sell wall.

Oftentimes I was even identify traders that submitted and cancelled their orders, such as this pair.

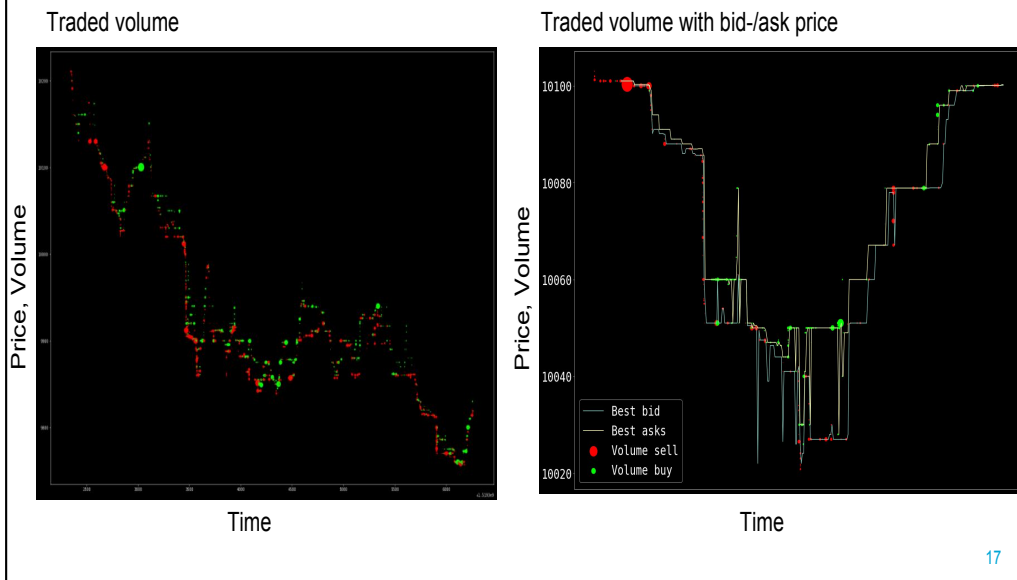
And furthermore, I was able to determine direct correlation to future market price movements based on these patterns.

		641.3	16.973
		639.5	0.040
		638.1	2.999
		638.8	1.000
	641.3	637.9	1.000
	639.5	636.5	1.000
	639.1	636.4	1.000
641.3	638.6	636.3	1.000
639.5	637.9	635.3	3.000
638.1	636.5	635.1	0.980
638.8	636.4	634.0	1.000
637.9	636.3	633.8	3.156
636.5	636.2	630.2	1.000
636.4	635.1		
636.3	634.0		
635.3	633.8		
635.1		632.9	0.102
634.0		632.7	0.255
633.8		632.2	0.002
633.2	632.9	635.6	1.191
	632.7	633.7	0.385
	632.2	632.4	0.350
		632.0	0.683
622.9	635.6	631.6	0.002
622.7	632.9	631.5	0.292
622.2	632.4	631.4	3.364
621.6	632.3	631.3	4.746
621.7	631.6	631.1	1.930
621.4	630.6	630.6	2.162
621.6	632.3	4.746	
621.5	632.1	1.930	
621.4	630.6	2.162	
621.3	4.746		
621.1	1.930		
630.6	2.162		



State: (lookbackWindow m, 2*bookSide n, 2)

Data analysis: Trades



The second category of market events are trades.

Looking at the price and volume over time one can clearly see that this corresponds to the market price movement.

Looking at it in more detail, I was able to see that either consecutive or impulsive sell trades led the price to fall.

Likewise, the market price oftentimes moved up after consecutive or impulsive buying.

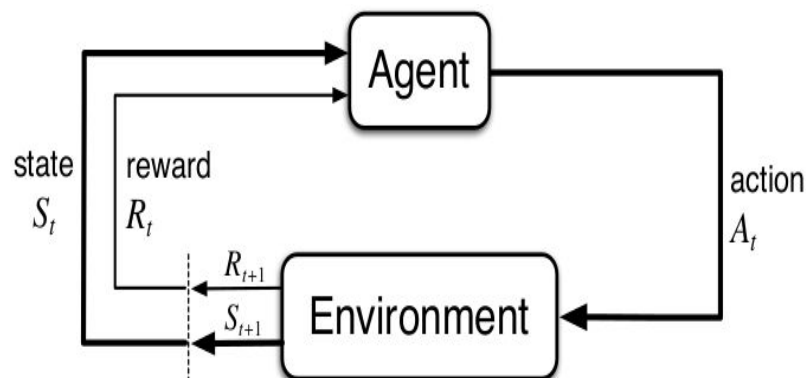
Construction of Feature II (RQ 1.1)

$$s_{trade} = \begin{pmatrix} \Delta t s_1 & p_1 & q_1 & os_1 \\ \Delta t s_2 & p_2 & q_2 & os_2 \\ \vdots & \vdots & \vdots & \vdots \\ \Delta t s_n & p_n & q_n & os_n \end{pmatrix} \forall p, q, os, ts \in Trade$$

Time difference is measured from the time of the order placement t

With that in mind, I constructed a second feature, that contains a number of trades. Thereby i took the price, quantity and the side (buy or sell) of each trade. But I also calculate the time difference, relative to time when we place an order, therefore maintaining the frequency in which the trades have occurred. Okay, with that my data preparation part was completed.

Framework construction



<https://www.oreilly.com/learning/introduction-to-reinforcement-learning-and-openai-gym>

It is now time to construct the reinforcement learning framework, that consists of 2 components: the environment and the agent.

The environment provides a new state to the agent for which it decides which action to take.

The environment rewards the agent for how well it did.

The goal now is to make the transposition of the limit order placement into this context.

Introduction

Previous approaches

Methodology

Data curation

Framework construction

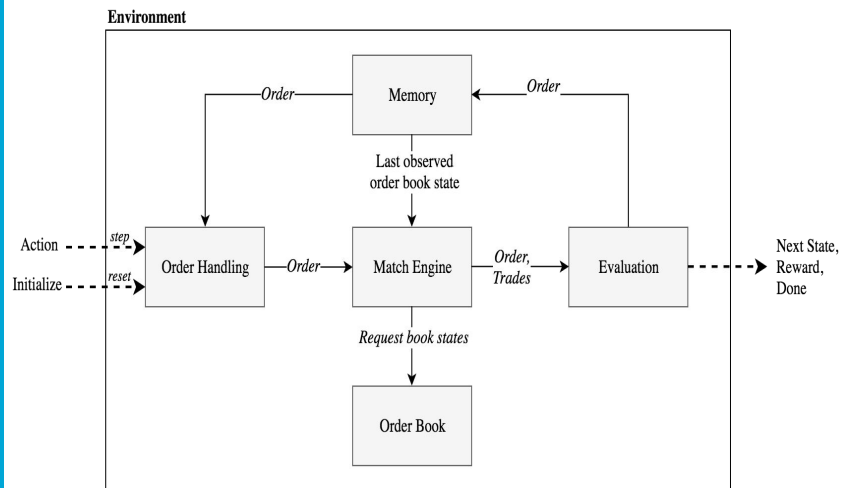
Evaluation

Results and limitations

Recommendations and conclusion

RL Environment – Overview

Enables an agent to find a policy to place limit orders



20

Let us first have a look at the environment that i developed.

Id like to demonstrate its working with a workflow diagram.

Starting with the agent that initializes an learning epoch, receives the first state and sends an action using the step function to the environment.

The order handling components will create an order and sets the price of the order according to the action received.

It then forwards the order to the match engine, which tries to match the order by using the historical order book.

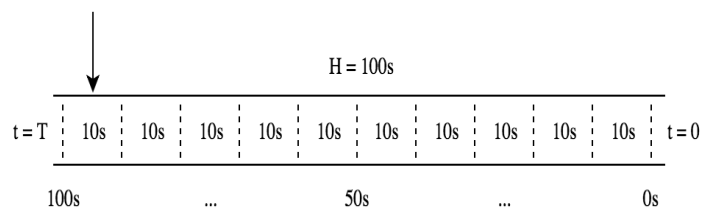
The resulted trades, if any, are evaluated and a reward is sent back to the agent, alongside the next state and whether or not the order is completed.

Internally, the order is stored in a memory and with the next step the agent takes, the price at which to place the is updated again.

This process continues until the order is completed or the time horizon is exceeded.

RL Environment – Configuration

- Feature Type: I, II
- Order Side: Buy, Sell
- Inventory (I): 1.0 BTC
- Time Horizon (H): 100 seconds
- Time step length (Δt):



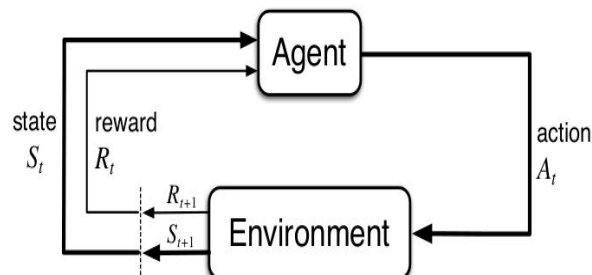
To make this process working, I had to configure some parameters.

Most importantly, I decided to make the environment discrete by introducing a time step size, which limits the number of steps to be taken by the agent and defines the duration of the matching process.

That is to trade 1.0 bitcoin within 100 seconds with a maximum of 10 steps and matching duration of 10 seconds for each step.

RL Environment – State space

- Observation (Agent's input)
 - Private Variables (inventory left i , time left t)
 - Market feature (I or II) of current order book state: FT_t



When we talk about states, we either refer to the internal state of the environment. That is the pointer in the historical order book and the inventory and time left for the order.

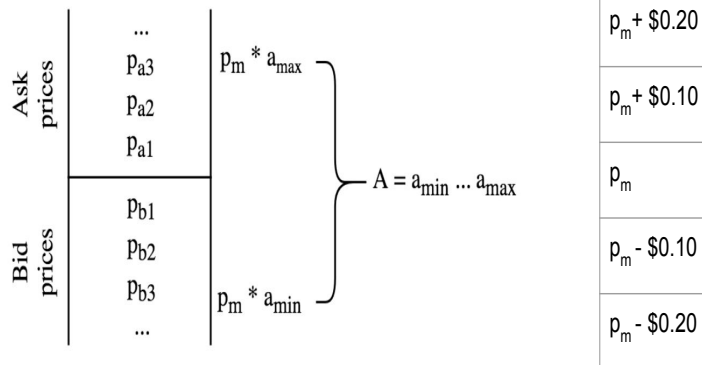
The other state is the observation that is provided to the agent.

That is again the private variables, but it is also the market feature that is derived from the order book pointer.

RL Environment – Action space

Order book state
(OS) at some time
step t

Example: $\Delta a = \$0.10$
 $|A| = 5$



The actions the environment allows an agent to take refer to the level in the order book.

Looking at a simplified orderbook here, we can go price levels up and down relative to the market price that is in the middle.

For example with a step size of 10 cents, this would be market price +10, +20 or in the opposite direction -10, -20.

RL Environment – Reward

$$\text{Measure: } VWAP = \frac{\sum \text{Number of shares} * \text{Share price}}{\sum \text{Total shares}} \quad \text{of trades}$$

$$\begin{aligned} \text{Reward: } r &= p_{m^T} - p_{vwap} && \text{for selling} \\ r &= p_{vwap} - p_{m^T} && \text{for buying} \end{aligned}$$

The last thing to define is the reward function, that is how the agent ultimately learns. We measure the vwap as it tells the average price paid or received for the resulted trades.

The environment calculates the difference of the volume weighted average price of the resulted trades and the market price prior the order was submitted.

Remember: our goal is to get a better price than what was offered at the market before we submitted the order.

Introduction

Previous approaches

Methodology

Data curation

Framework construction

Evaluation

Results and limitations

Recommendations and conclusion

RL Agents

- Aims to find a policy that optimizes limit order placement
- Uses the environment
- With/without market features



25

Now that we have constructed the environment, what is left to do is the agent setup. I actually built two types of agents, one that uses market features, and one that does not.

RL Agent – Q-Learner

Q-Table:

State: (time, inventory)	Action	Value
...

- Not suitable when using market features
- Same state would likely not appear twice
- Suitable when using private variables only

The q-learning agent, whose inner workings I'm not going to explain in detail due to time limitations, does not use market features.

What I would like to mention is that the q-learner makes use of a q-table that stores all the visited observations and aims to find the optimal action for such a state.

Let us remember that the market features are derived from the order book.

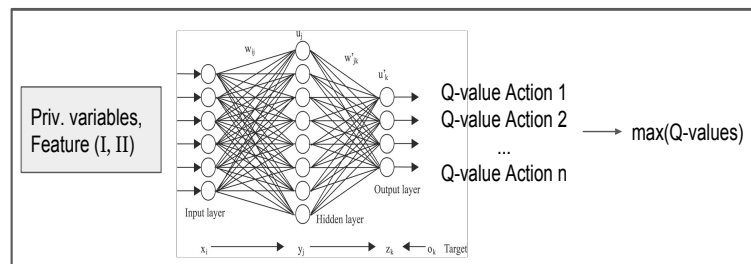
Since the order book changes constantly, a new state is likely not visited more than once.

Therefore this table would just be extended by states and the agent wouldn't be able to learn.

However, when just the time and inventory, that is left, is in use, then it is actually a very straightforward and simple approach.

RL Agent – DQN

- Action-value function approximation with neural network
- Widely-used CNN in use as well as a MLP (2-hidden layers)
- Recent successes with raw signal data [1,2]
- Experience replay: train on random mini-batches
- MSE loss function



[1] <https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf>

[2] <https://web.stanford.edu/class/psych209/Readings/MnihEtAlHassibis15NatureControlDeepRL.pdf>

In order to be able to consider the constructed features, the second agent introduces a function approximator instead of this table.

This allows to approximate arbitrarily complex features.

I have chosen to implement a deep q network agent that makes use of a neural network as a function approximator.

I have used the widely known CNN implementation that has shown to be successful in other fields.

In addition, I implemented a more simple multilayer perceptron to have means for comparisons.

The basic idea is that we feed features to the neural network, which in turn adjusts its weights and outputs the expected rewards for each action.

The agent will choose the action with the maximum expected reward.

Evaluation Procedure (RQ 1.3)

- Empirical analysis
 - Expected return for (1) market order and (2) optimal placed limit order
 - Environment parameter tuning
- Q-Learning agent policy
 - Average return (without market features)
- DQN agent policy
 - Average return (with market features)
- DQN agent limitations

Previous literature has not suggested any other evaluation methods besides measuring rewards, as it is standard in reinforcement learning. However, in my case this was not sufficient as this does not give indication of how well my approach performs compared to straightforward buying and selling at market prices.

Therefore I propose an evaluation procedure consisting of 4 steps:

first I proceeded an empirical analysis which provides expected returns for (1) market orders and (2) optimal placed limit orders

I observed the performance of the q-learner which does not use market features. both results combined will then serve as a baseline for the second agent, the DQN agent that does make use of market features.

Last but not least I proceeded an investigation of the limitations of the DQN agent.

Introduction

Previous approaches

Methodology

Data curation

Framework construction

Evaluation

Results and limitations

Recommendations and conclusion

 TU Delft

Real world data sets

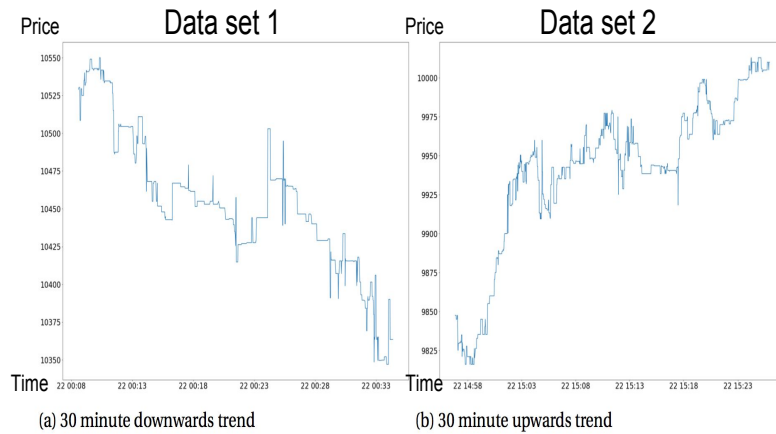


Figure 6.1: Bid/ask mid-price of 30 minute order book recordings.

Regarding the data sets, I have chosen two sample data sets of the collected data. The first one contains a downwards trend and the second one an upwards trend, therefore ensuring that different price constellations are covered during the training and testing.

Artificial data sets

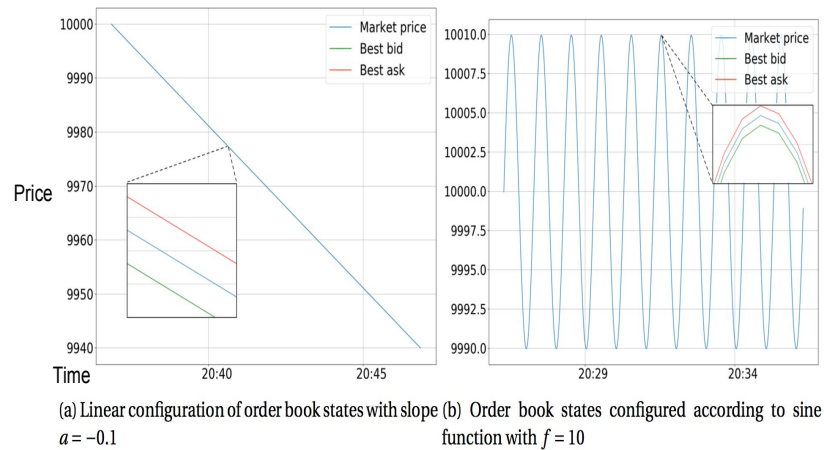


Figure 6.15: Artificial order books with duration of 10 minutes

In addition, I created a method that allows me to generate artificial order books. Thereby I define a function which is translated into a multivariate time series. The first data set is based on a linear function with a downwards slope and the second one is based on a sine function. The the zoomed area you can see the limit levels built around the market price.

Introduction

Previous approaches

Methodology

Data curation

Framework construction

Evaluation

Results and limitations

Recommendations and conclusion

TU Delft

Training and Testing

Order book (sequence of states)

Training epochs

Testing epochs

Train data
 Test data

Time horizon at random state

- Data sets are split with ratio 2 : 1
- 5000 epochs training
- 1000 epochs testing
- 1 epoch = 1 order fulfilled

31

I was especially careful in the way how to carry out training and testing. I made sure that with each new epoch initialized by the agent, a random starting point in the order book is chosen. Thereby I ensure that same states are not visited over and over again and avoid overfitting of similar states. Training consists of 5000 epochs and testing of 1000 epochs.

Introduction

Previous approaches

Methodology

Data curation

Framework construction

Evaluation

Results and limitations

Recommendations and conclusion

 TU Delft

Empirical analysis

- Return (VWAP)
- 201 actions
- Limit levels -100...100
- 100 orders (e.g. epochs) for each level
- 20'100 orders in total

$\mathbb{E}[\text{Limit order}]$ (optimal)	$\mathbb{E}[\text{Market order}]$
-9.88	-30.53

32

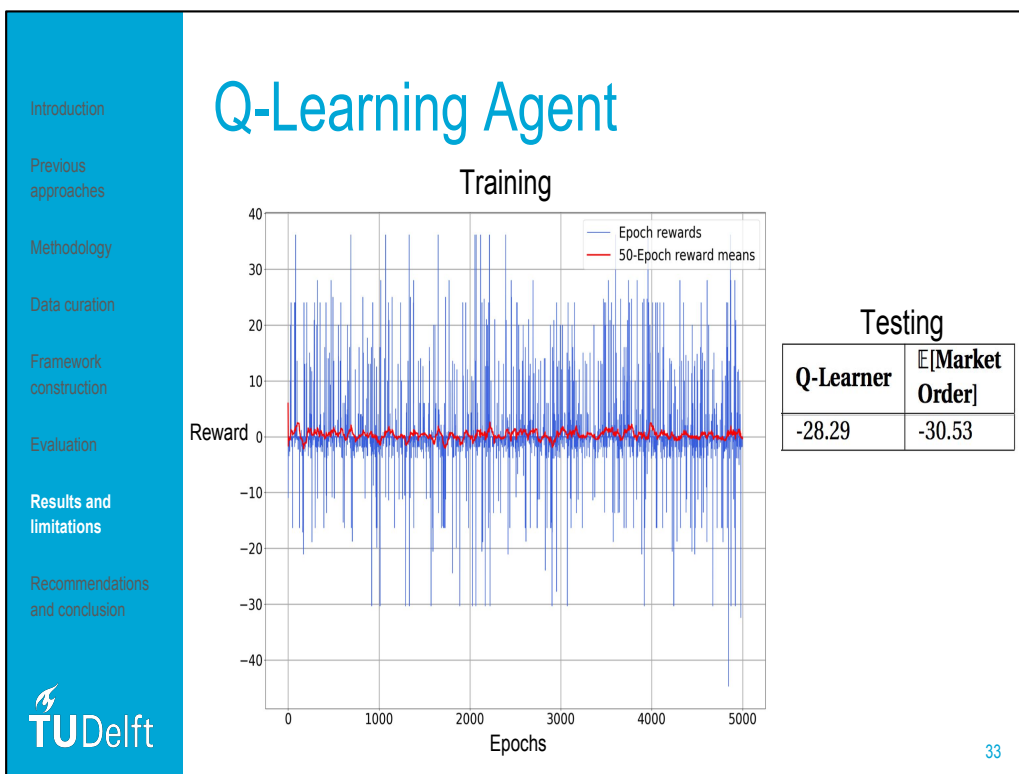
The results of the first evaluation step, the empirical analysis, were derived by simulating the limit order placement process over 100 epochs for a range of 201 actions.

This results in a total of 20'100 order being submitted.

By doing so I was able to determine the expected return for a market order and the expected return for an optimally placed limit order.

As we can see in the table, a market order is much more expensive than a properly placed limit order.

Our goal was now to improve over the market order return.



The training of the Q-learner on the two data sets, for buying and selling, was a very noisy process.

Taking the moving average of 50 epochs into consideration, it can be seen over 5000 epochs, the learner was not able to improve.

The table confirms this result and we see that the q-learner was only marginally better than buying and selling at a market price.

DQN Agent – Real world data

Multiple window sizes tested.

Feature I $\Sigma[B1, S1, B2, BS2]$	Feature II $\Sigma[B1, S1, B2, BS2]$	$\mathbb{E}[\text{Market Order}]$
<u>-18.62</u>	<u>3.36</u>	-30.53

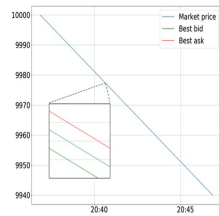
Table 6.6: Summary of rewards during backtest of DQN-CNN agent using different feature vector sizes.

The same training and testing procedure was proceeded for the DQN agent. I ran the evaluation with multiple window sizes and the table shows the best found results.

The values are the sum of all test for buying and selling on both real world data sets. As we can see, feature 2, the historical trades, performed superior to the use of feature 1, the historical order book.

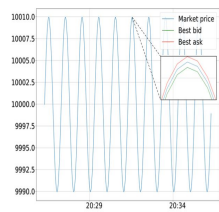
However, both approaches performed better than the expected market order return.

DQN Agent – Artificial



(a) Linear configuration of order book states with slope $a = -0.1$

(a)	Optimal (Reward)	Agent
Buy	\$10.0	\$9.45
Sell	\$-0.10	\$-0.10



(b) Order book states configured according to sine function with $f = 10$

(b)	Optimal (VWAP)	Agent
Buy	\$9'990.0	\$9'992.0
Sell	\$10'010.0	\$10'007.0

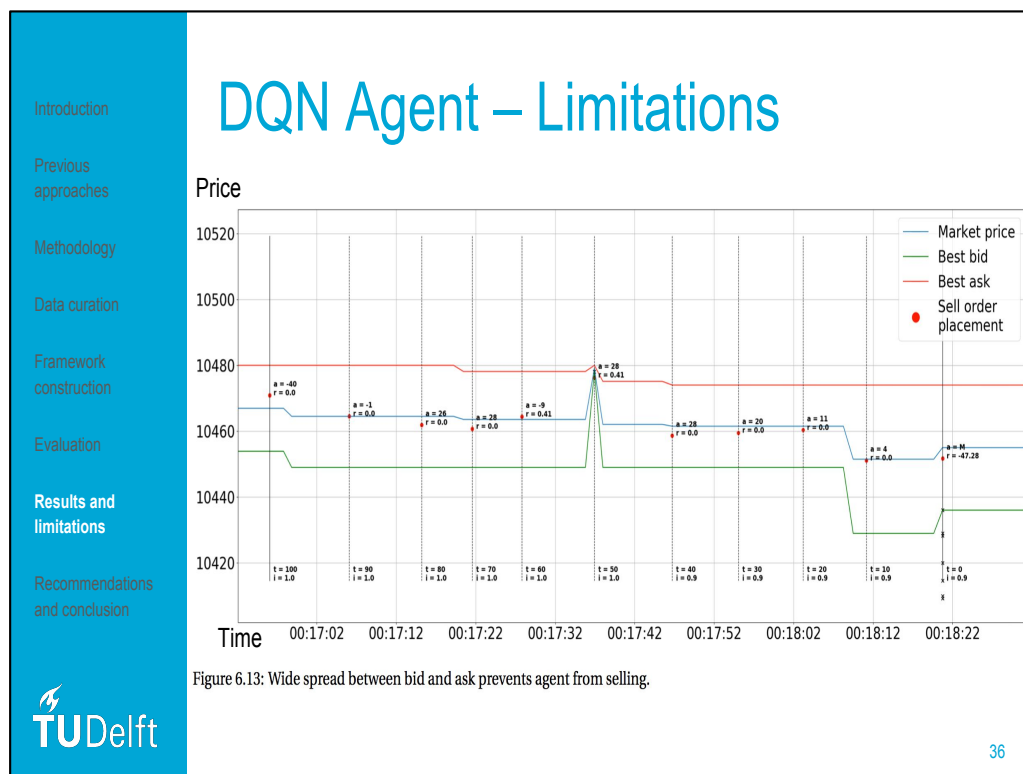
- Near-optimal placement
- Absence of market noise
- Stationary evolution of order book
- Liquidity constantly provided

In a next step I trained and tested the DQN agent on the artificial data sets. Thereby I was able to determine the optimal reward prior training, that is \$10.0 for buying and \$-0.10 for selling.

As we can see, the model was able to find a near optimal policy for the linear development of the order book.

For the sine function I calculated the optimal volume weighted average price prior training, as this is easier than the reward in this case scenario.

Again, my model was able to come close to an optimal policy for both buying and selling.



In the last step of the evaluation I tried to find limitations in my approach. Although my model performed well in the real world scenario, it was not as good as with artificial books.

With my method to follow the decision-making process of the agent, i was able to see that at times, the spread between best bid and best ask became very large. And at this point the spectrum of the action space was not large enough such that the agent did not come close to the prices offered at the market.

As a result the order had to be filled with market orders once the time horizon was exceeded.

This is a limitation in the environment itself, were the action space should be larger.

Introduction

Previous approaches

Methodology

Data curation

Framework construction

Evaluation

Results and limitations

Recommendations and conclusion

 TU Delft

Performance overview (RQ 1.4)

E[Market Order]	E[Limit order] (optimal)	Q-Learning	DQN-CNN (Feature I)	DQN-CNN (Feature II)
-30.53	-9.88	-28.29	-18.62	<u>3.36</u>

Table 6.7: Summary of expected and achieved average rewards from empirical evaluations and reinforcement learning applications.

38

To conclude the evaluation the table shows the performance achieved from all the models presented.

We can see that no market features, as tested with the q-learner did not lead to great improvements.

Feature 1 improved already significantly and feature 2 performed very well. Even better than the expected optimal limit order.

Introduction

Previous approaches

Methodology


Data curation

Framework construction

Evaluation

Results and limitations

Recommendations and conclusion



Conclusion

- *RQ 1.1*
 - Found market patterns
 - 2 features constructed
- *RQ 1.2*
 - Sequential decisions make RL suitable
 - Built around given exchange functionality
 - Extendible in terms of features and capabilities
- *RQ 1.3*
 - Evaluation procedure developed
 - Indication of optimization capabilities
 - Found limitations
- *RQ 1.4*
 - Limit order placement was optimized
 - There is more potential

39

Now I would like to conclude my work by reconsidering the research questions.

First of all, I have found patterns in historical market data and was able to construct features containing these patterns.

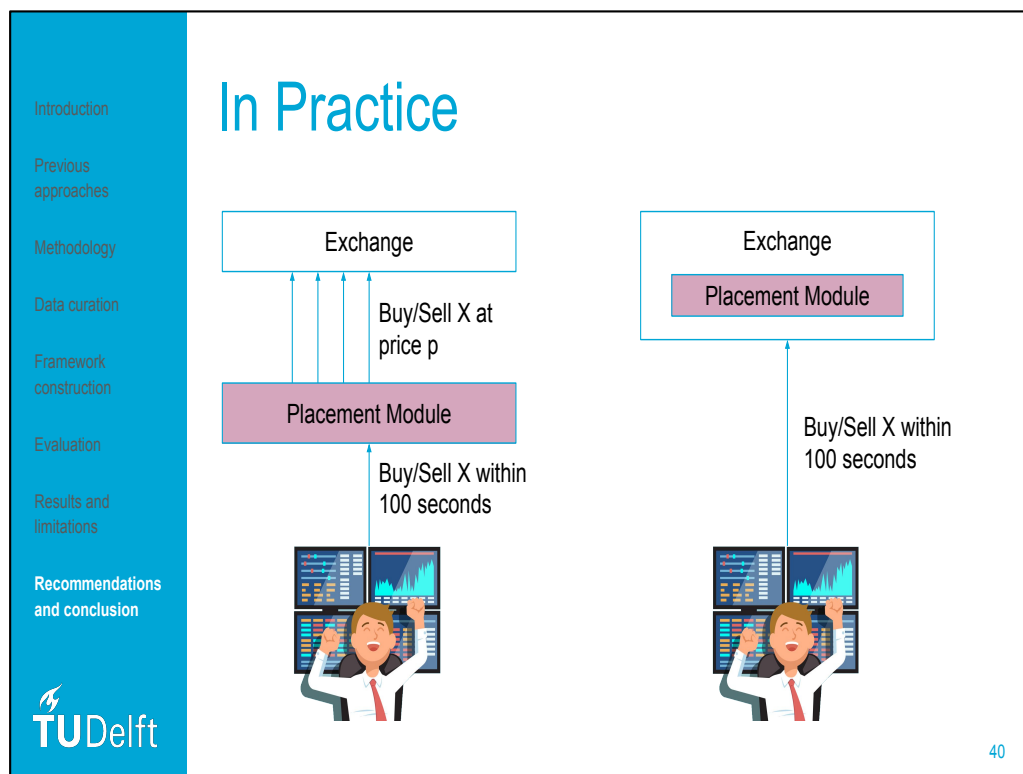
I was able to show that the sequential decision making process of limit order placement can be perfectly transposed to the reinforcement learning process. Moreover, I was able to build the environment around given methods offered by financial exchanges.

Furthermore, the way i have constructed the reinforcement learning framework allows for great flexibility when it comes to extensions of features or how to attempt the optimize the process.

The evaluation procedure developed allowed my to determine the capabilities of the environment and the agents and also allowed to find limitations.

Last but not least, the process was actually optimized and I determined which features contributed the most to that.

Overall, I have shown an how deep reinforcement learning can indeed optimize the limit order placement process.



In addition, and i think that is equally as important as answering the research questions, i developed a useful product.

<https://cryptomenow.com/one-of-the-worlds-top-etfs-trader-moves-into-trading-crypto/>

Introduction

Previous
approaches

Methodology

Data curation

Framework
construction

Evaluation

Results and
limitations

Recommendations
and conclusion

 **TU**Delft

Recommendations

- Test on live-market with actual purchases and sales
- Tune DQN agent parameters
- Combine with a statistical framework
- Extend to a “market maker”: $\text{Buy} + \text{Sell} = \text{Profit}$

Questions?