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

TU Delft

Contents

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

TU Delft

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

Why cryptocurrency data?

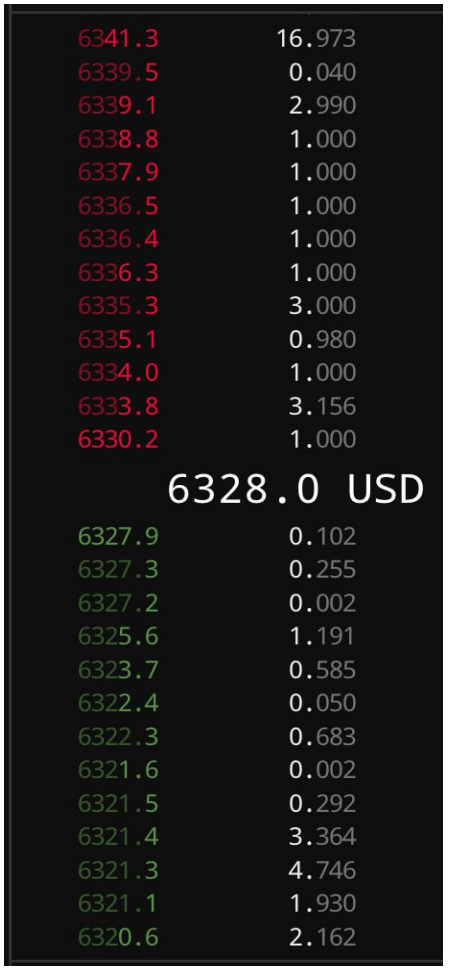
- Is traded in the same way as other currencies and securities (shares, bonds, etc.)
- Market data is free and available in real time
- Volatile market prices
- Low entry barrier → Many inexperienced traders
- Personal interest

Order book

Bid: price in a buy order
Ask: price in a sell order
Spread: gap between bid and ask

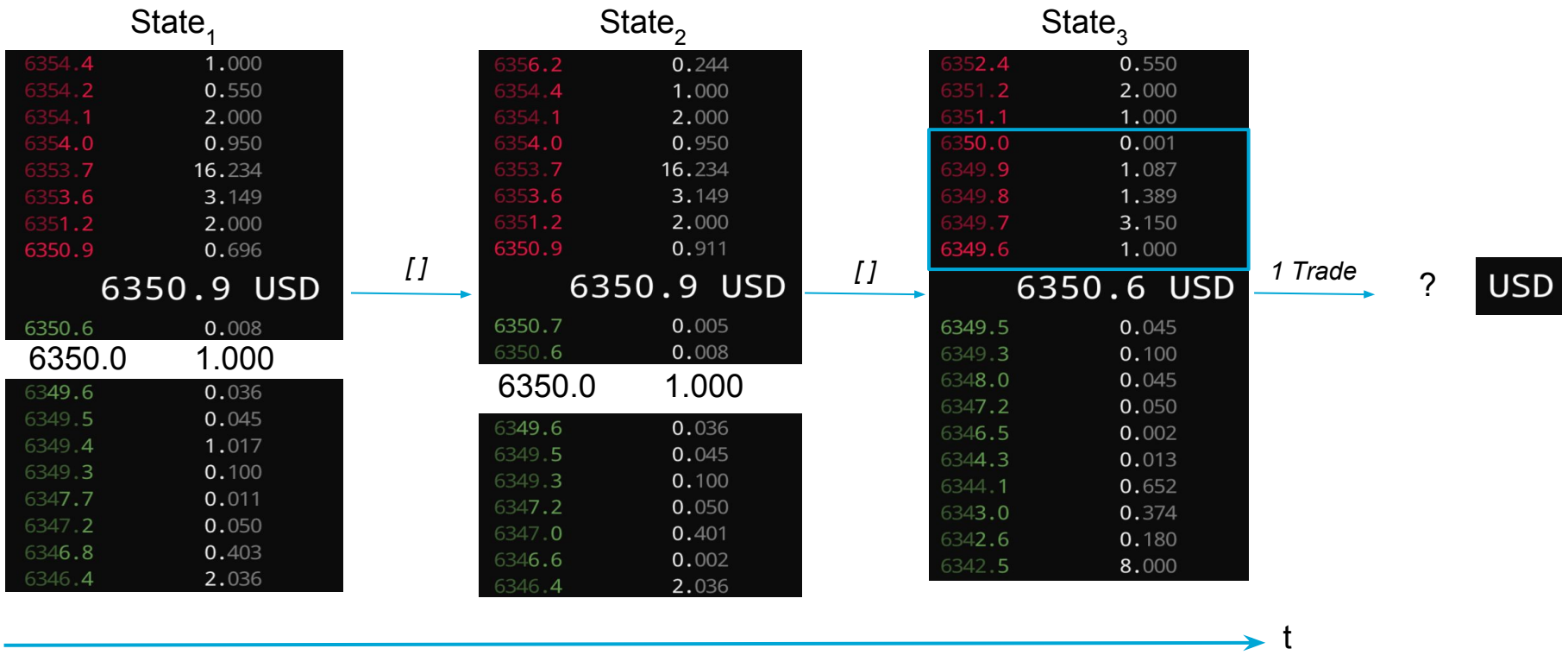
Order
side :: buy | sell
type :: limit(price, amount),
market(amount)

Action: Submit order (at price a level)
Result: Trades

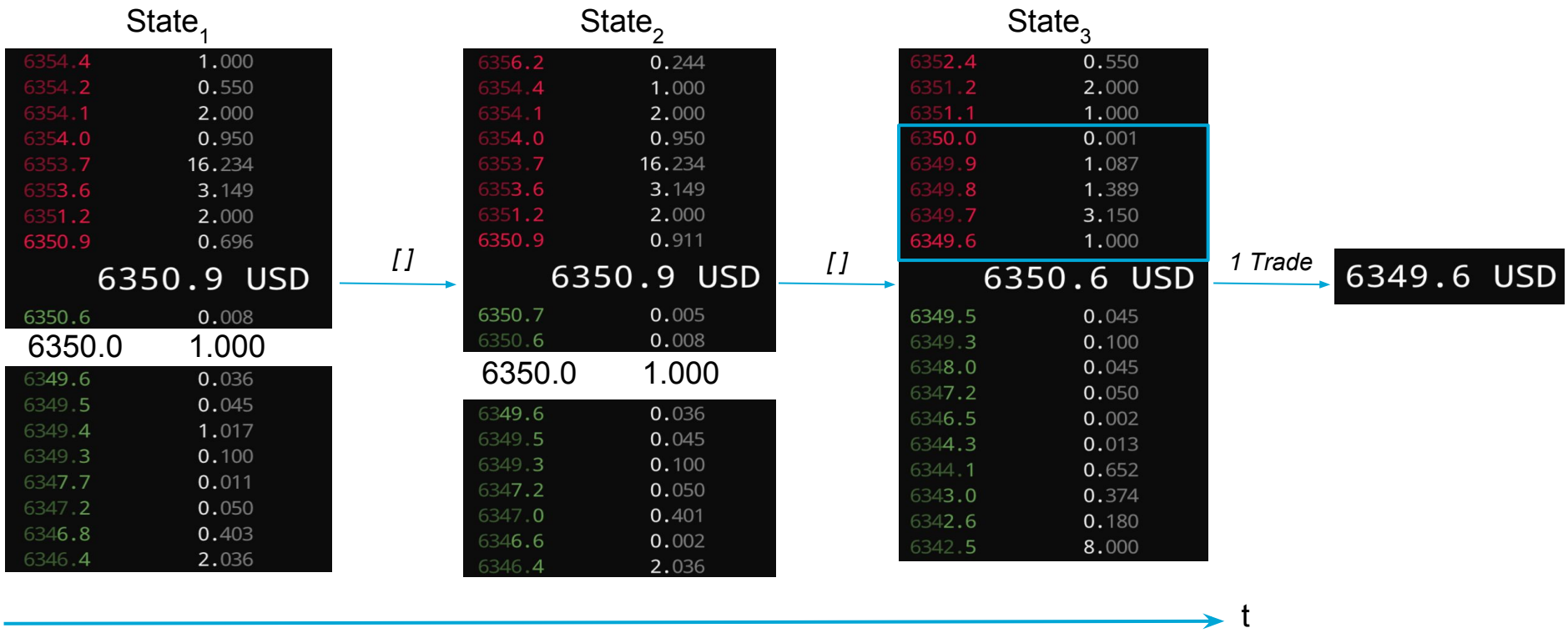


https://trade.kraken.com/markets/kraken/btc/usd

Limit order matching process



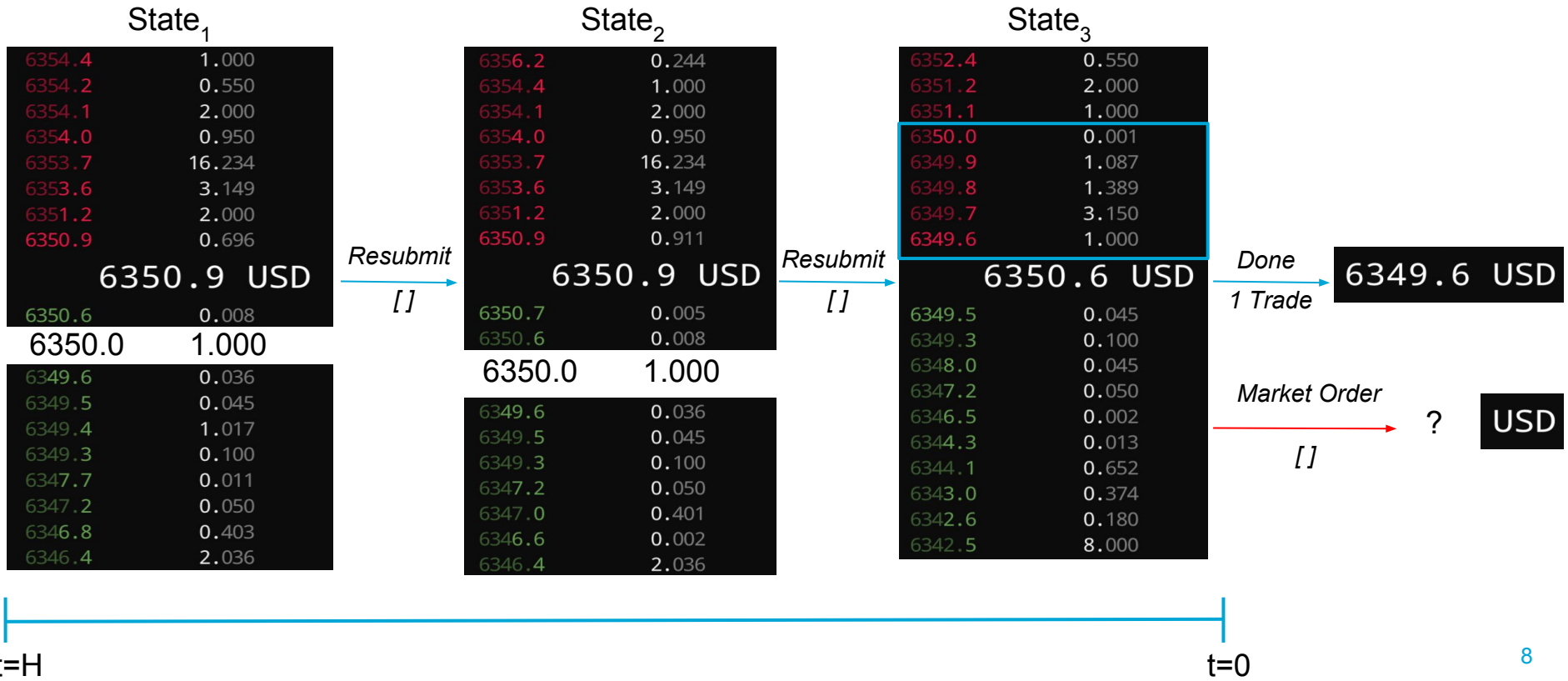
Limit order matching process



Limit order placement (optimization)

Fixed time horizon (H=100 seconds)

Fixed inventory (I=1.0 BTC)



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?

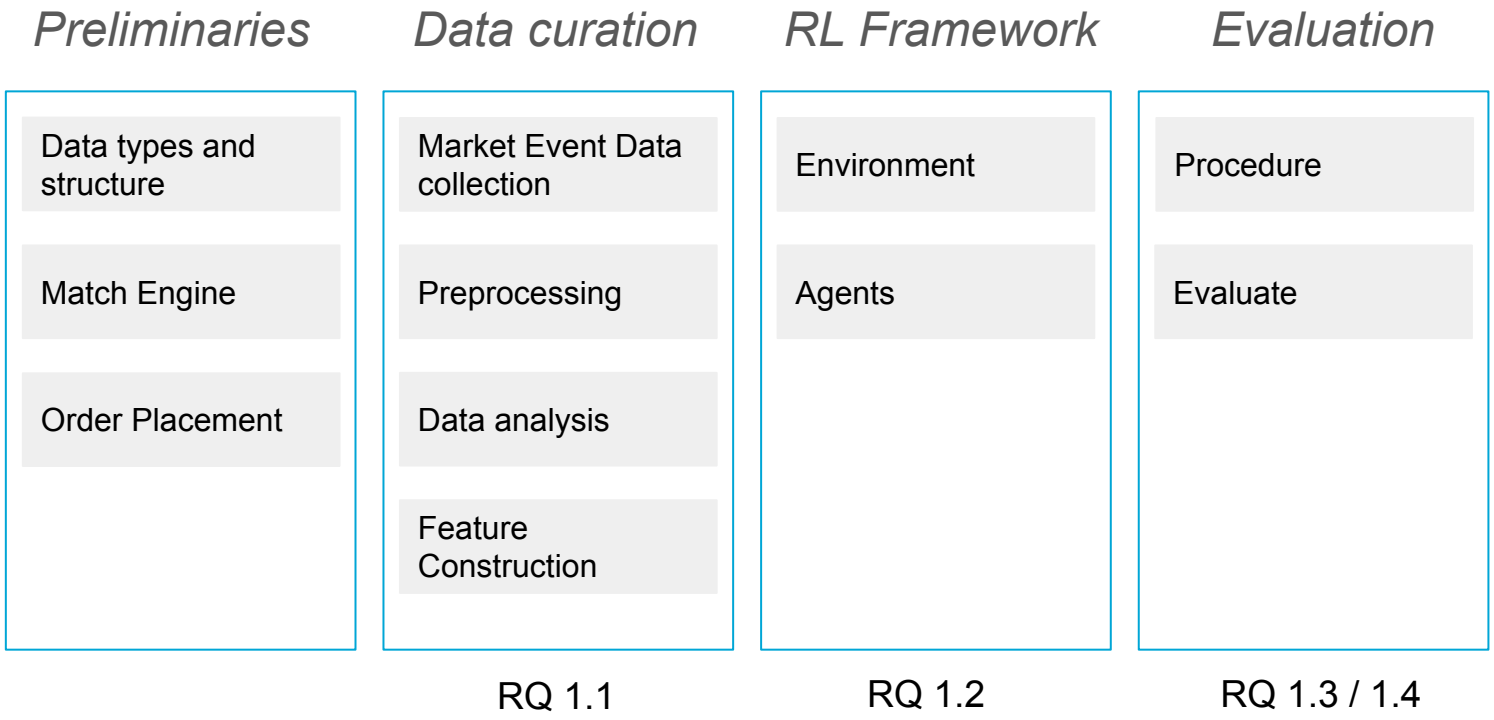
2. How should one design a reinforcement learning environment and agents, in the context of order placement?

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?

Methodology

Methodology



Data collection and processing

- 24 hours of data (USD/BTC)
- Collected using SignalR API (websocket abstraction)

- Limit order created
- Limit order cancelled
- Order filled (Trade)

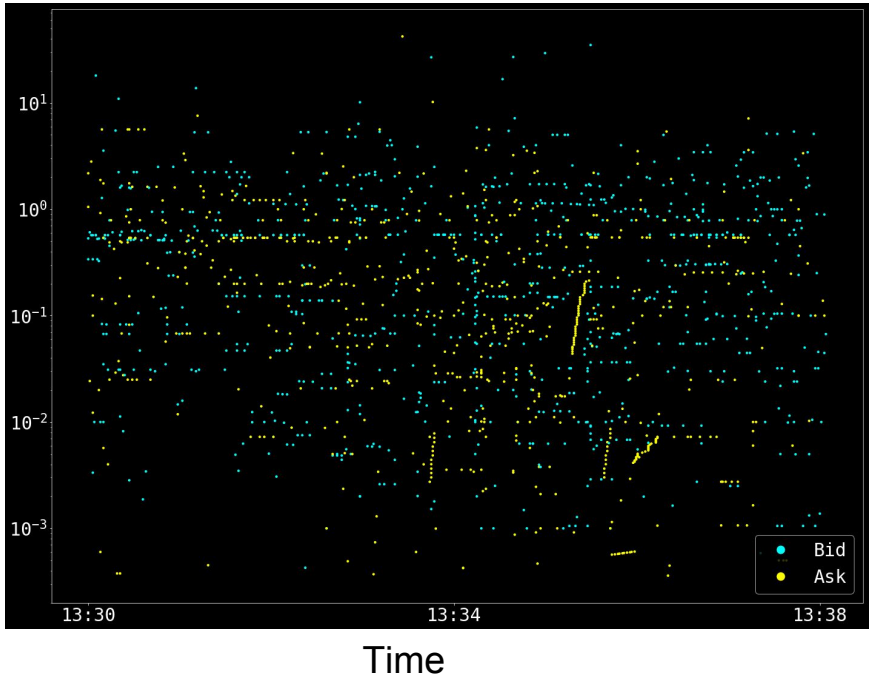
- Processed events
- Reconstruct complete order book

Data analysis

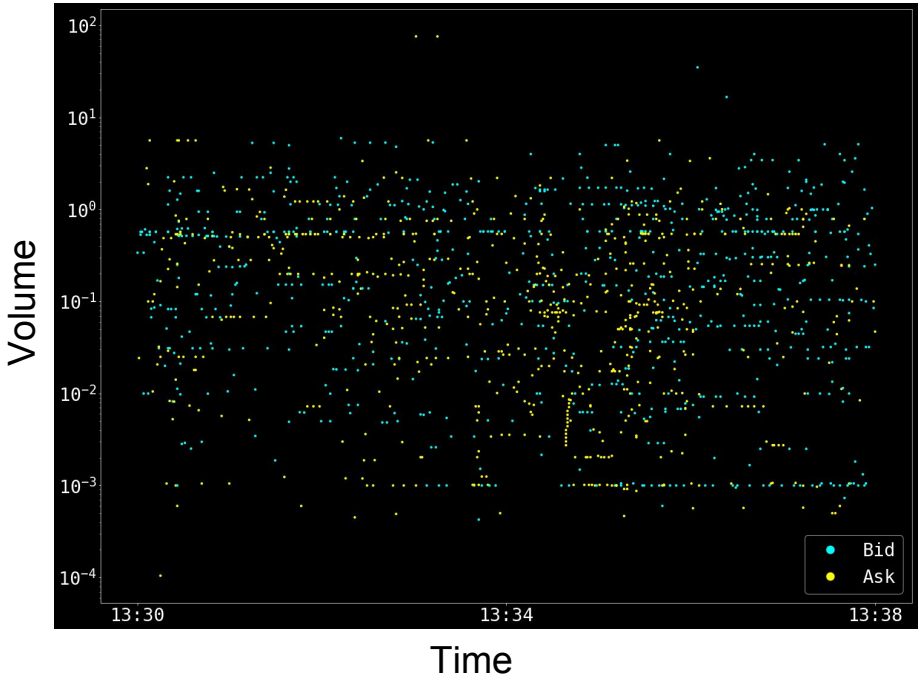
- First logical choice (given prior research) were hand-crafted features
  - Number of created/cancelled orders/trades
  - Transaction volume
  - ...
  - 200+ indicators possible (see TA-Lib: Technical Analysis Library)
- Come with difficulties and inconveniences:
  - Benefit for limit order placement not immediately evident
  - Computation required for each market event (>1/second) and each feature

Data analysis: Orders

Volume of created orders

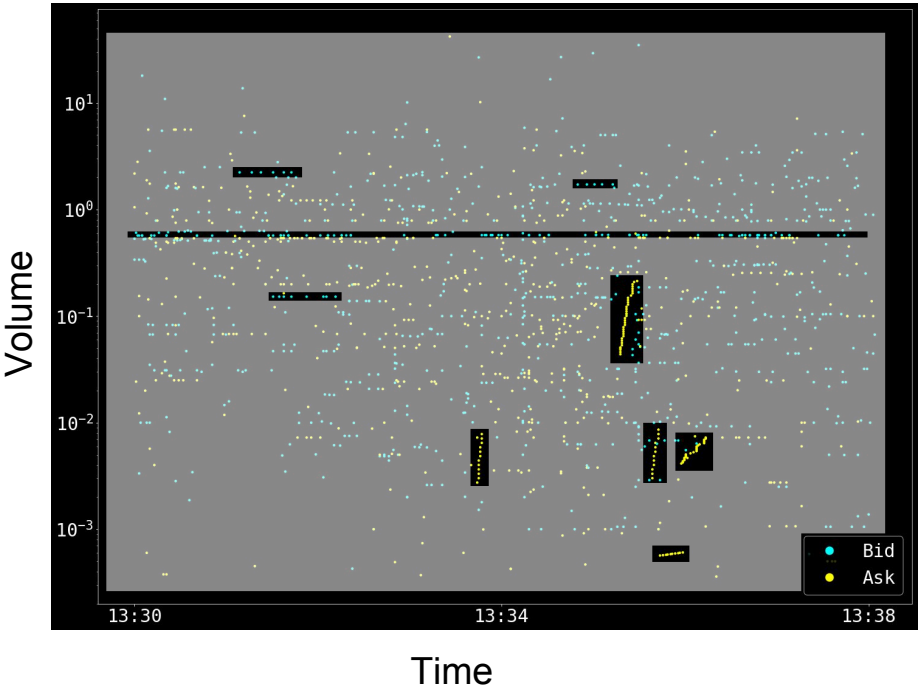


Volume of cancelled orders

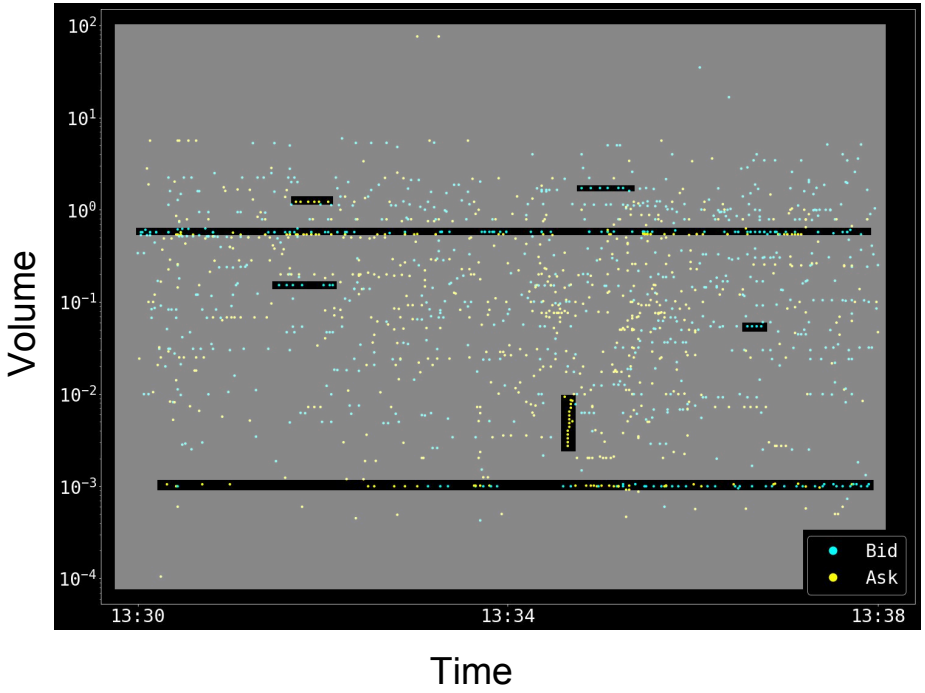


Data analysis: Orders

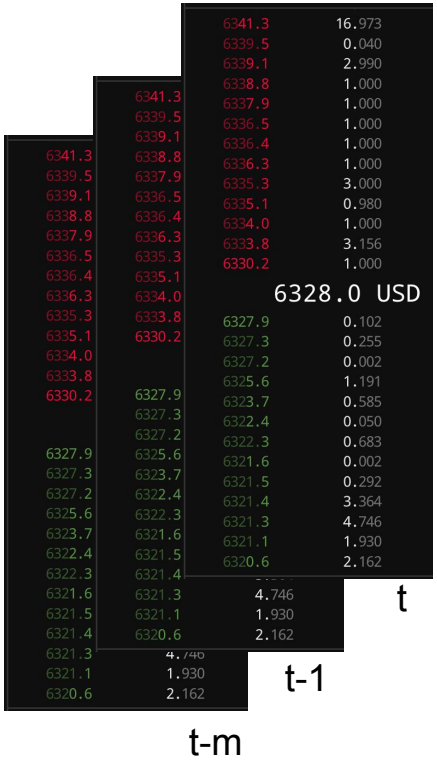
Volume of created orders



Volume of cancelled orders



Construction of Feature I (RQ 1.1)



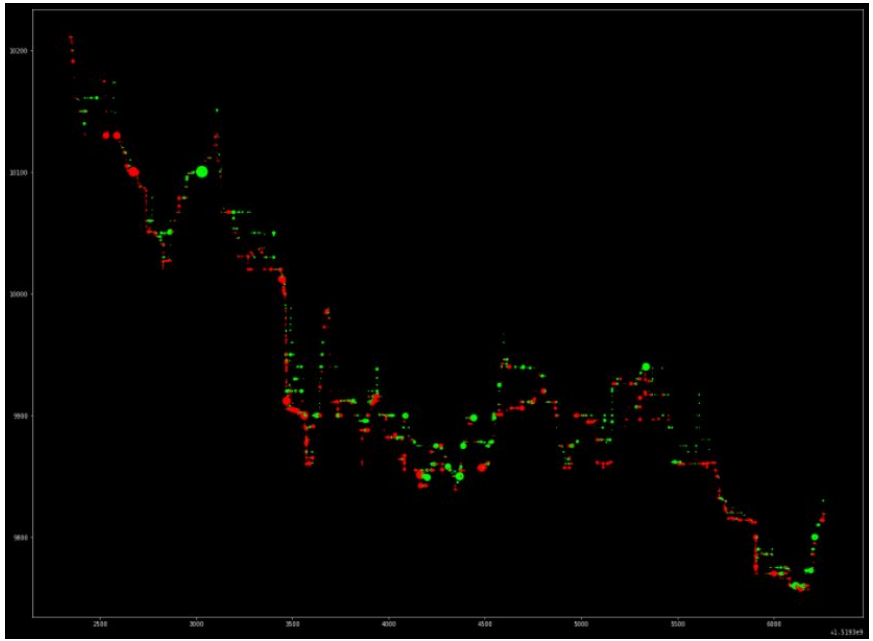
$$\begin{bmatrix} \begin{pmatrix} bp_1 & bs_1 \\ bp_2 & bs_2 \\ \vdots & \vdots \\ bp_n & bs_n \\ ap_1 & as_1 \\ ap_2 & as_2 \\ \vdots & \vdots \\ ap_n & as_n \end{pmatrix}_{t} & \begin{pmatrix} bp_1 & bs_1 \\ bp_2 & bs_2 \\ \vdots & \vdots \\ bp_n & bs_n \\ ap_1 & as_1 \\ ap_2 & as_2 \\ \vdots & \vdots \\ ap_n & as_n \end{pmatrix}_{t-1} & \cdots & \begin{pmatrix} bp_1 & bs_1 \\ bp_2 & bs_2 \\ \vdots & \vdots \\ bp_n & bs_n \\ ap_1 & as_1 \\ ap_2 & as_2 \\ \vdots & \vdots \\ ap_n & as_n \end{pmatrix}_{t-m} \end{bmatrix}$$

Price Size | Price Size | Price Size

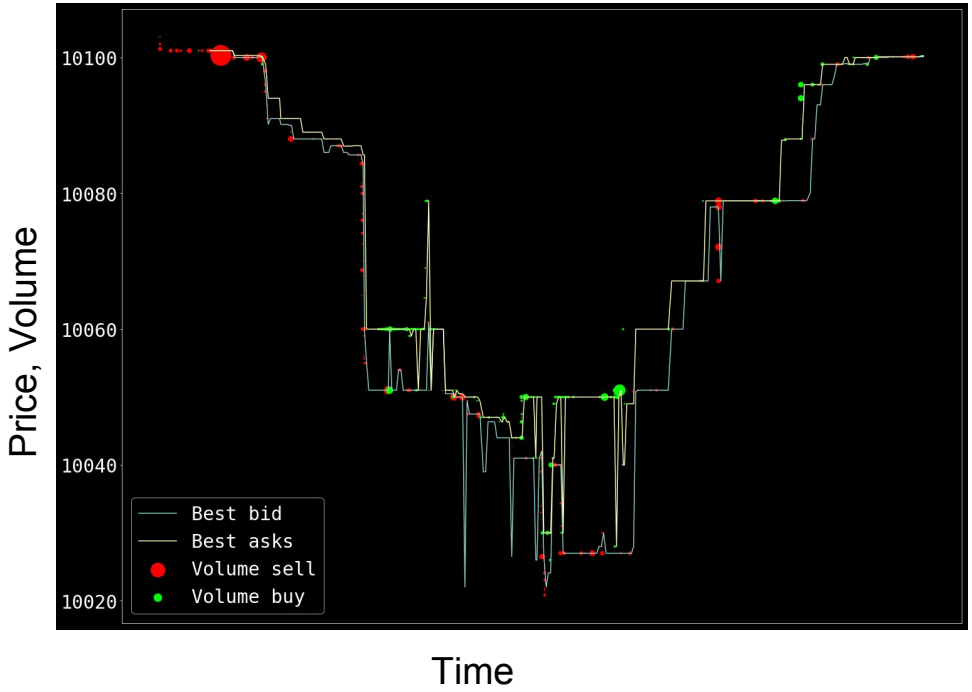
State: (lookbackWindow m, 2*bookSide n, 2)

Data analysis: Trades

Traded volume



Traded volume with bid-/ask price

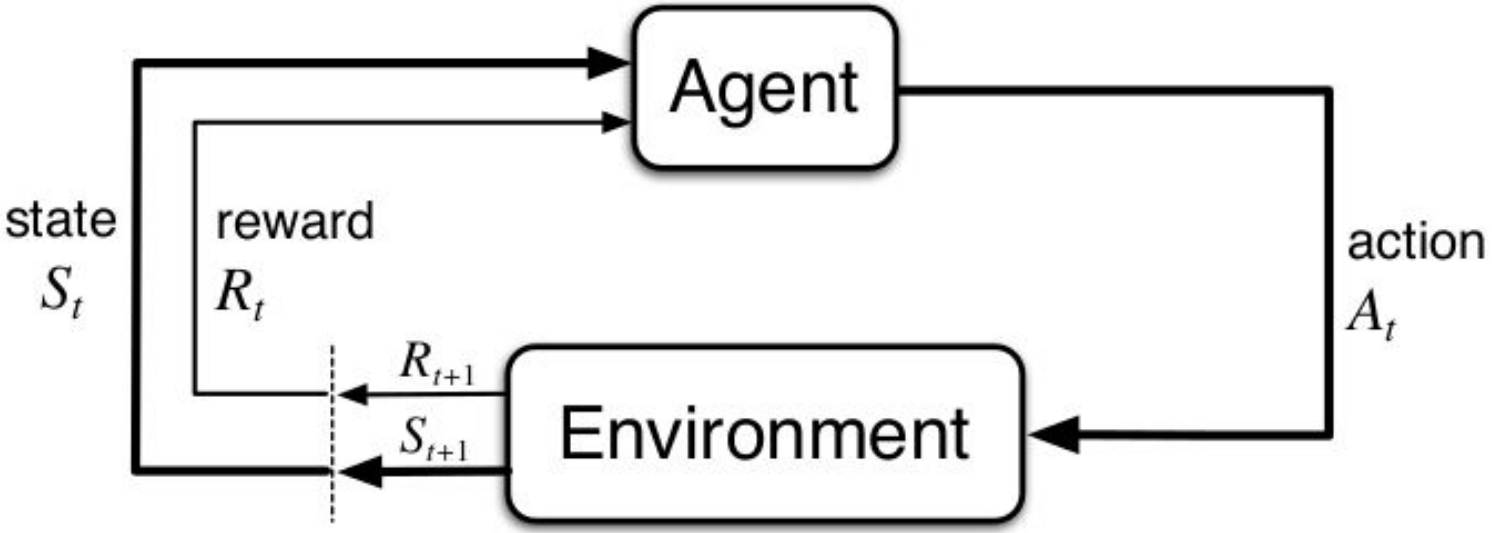


Construction of Feature II (RQ 1.1)

$$s_{trade} = \begin{pmatrix} \Delta ts_1 & p_1 & q_1 & os_1 \\ \Delta ts_2 & p_2 & q_2 & os_2 \\ \vdots & \vdots & \vdots & \vdots \\ \Delta ts_n & p_n & q_n & os_n \end{pmatrix} \forall\ p, q, os, ts \in Trade$$

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

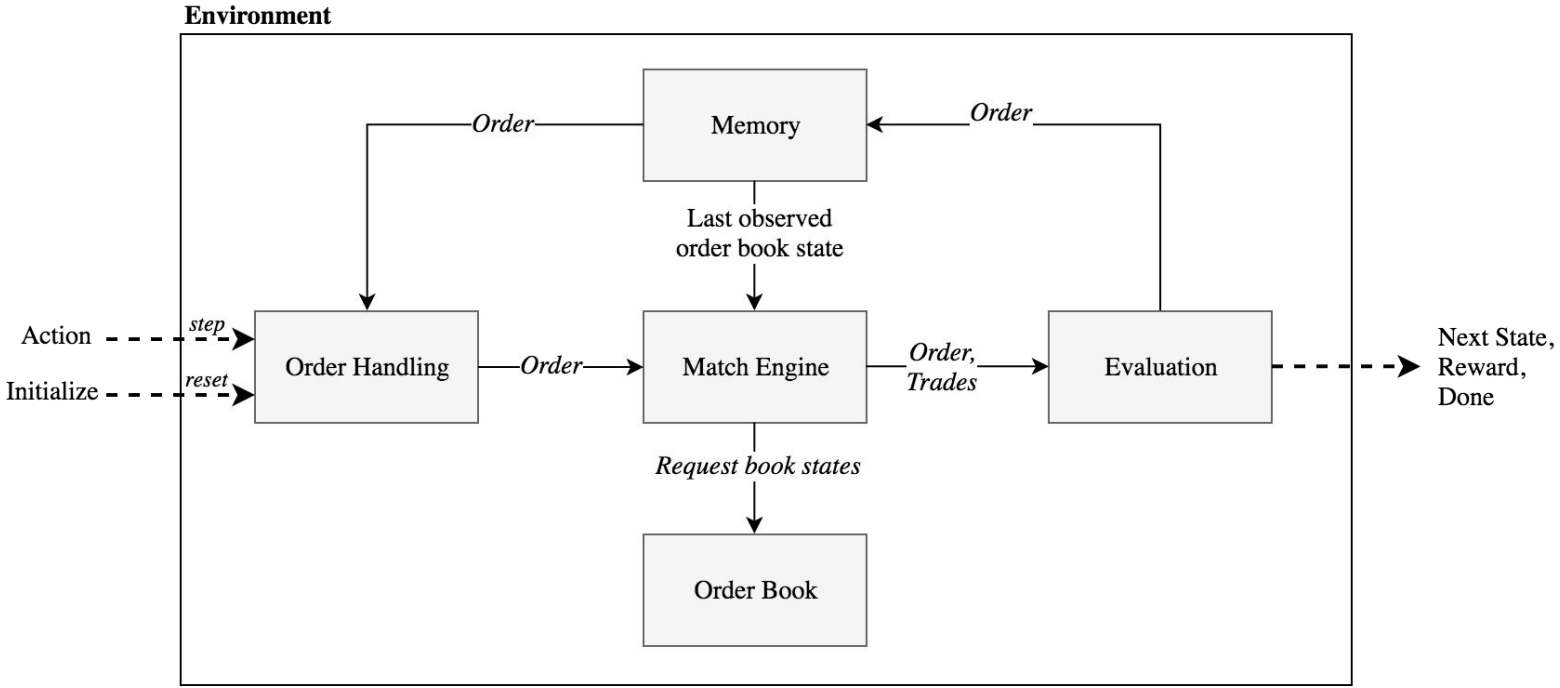
Framework construction



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

RL Environment – Overview

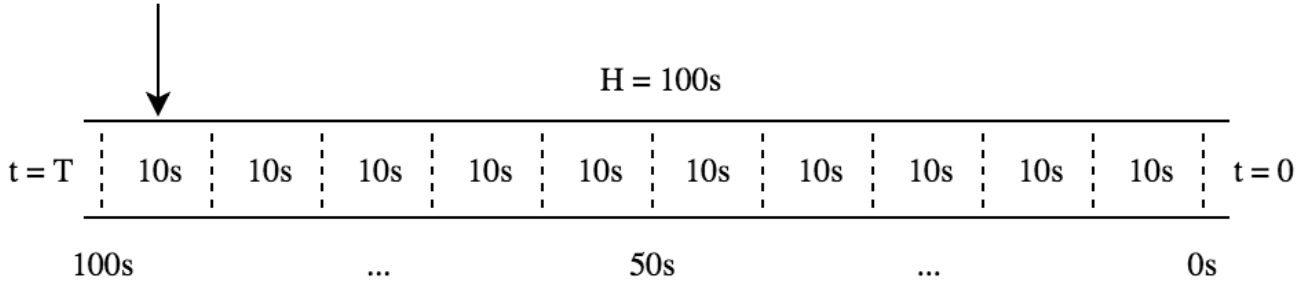
Enables an agent to find a policy to place limit orders



RL Environment – Configuration

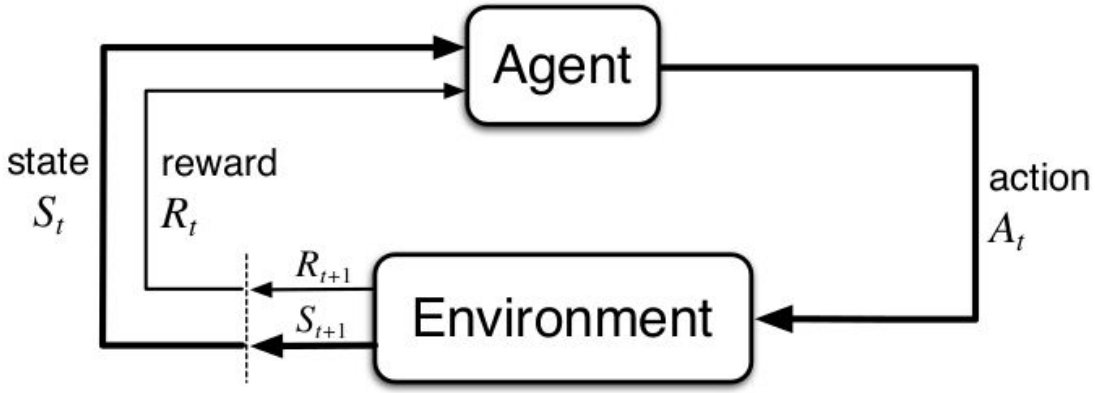
- Feature Type: I, II
- Order Side: Buy, Sell
- Inventory (I): 1.0 BTC
- Time Horizon (H): 100 seconds

- Time step length (Δt):

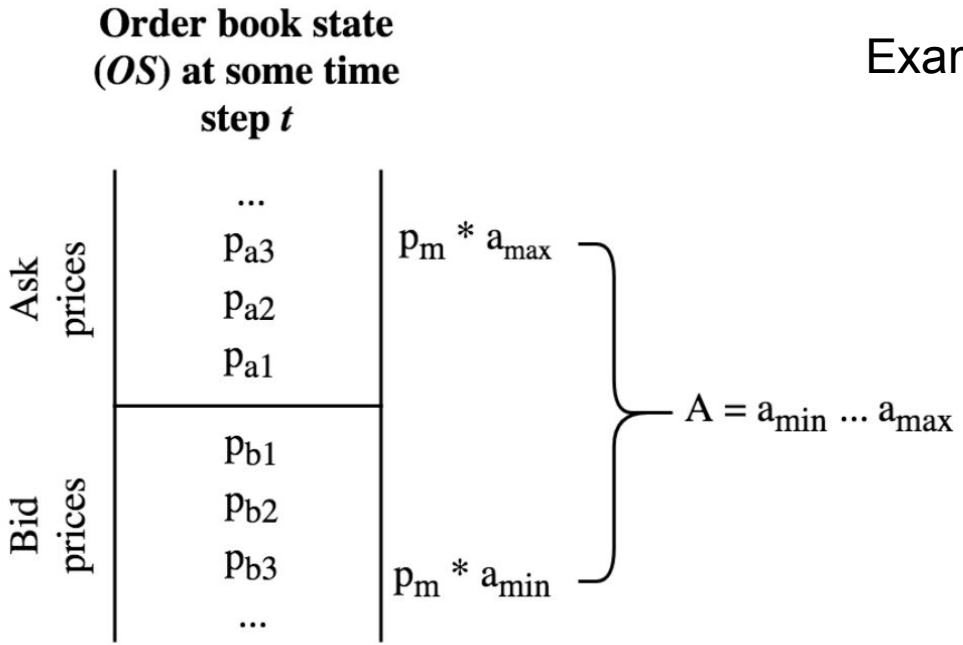


RL Environment – State space

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



RL Environment – Action space



Example:

$\Delta a = \$0.10$

$|A| = 5$

$p_m + \$0.20$
$p_m + \$0.10$
p_m
$p_m - \$0.10$
$p_m - \$0.20$

RL Environment – Reward

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

Reward:

$r = p_{m^T} - p_{vwap}$ for selling

$r = p_{vwap} - p_{m^T}$ for buying

RL Agents

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

RL Agent – Q-Learner

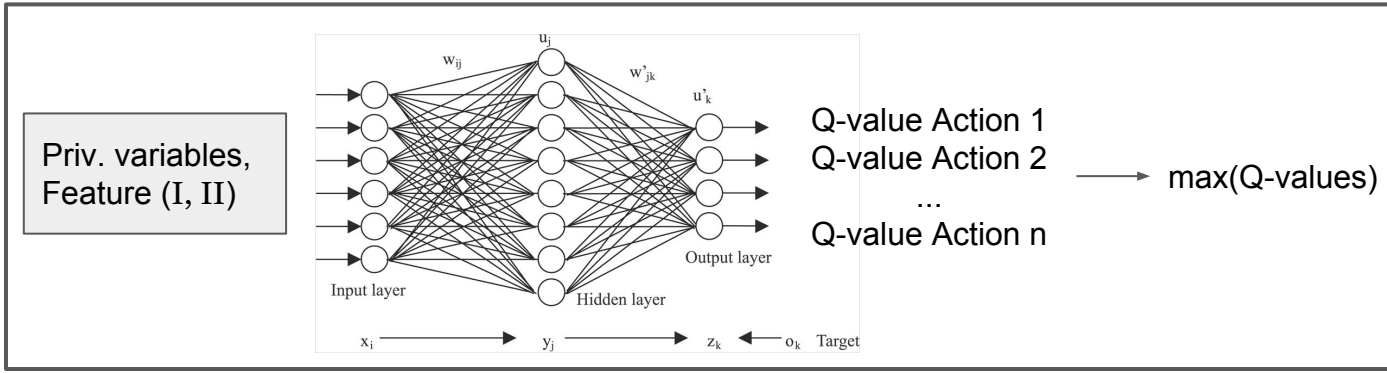
Q-Table:

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

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

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

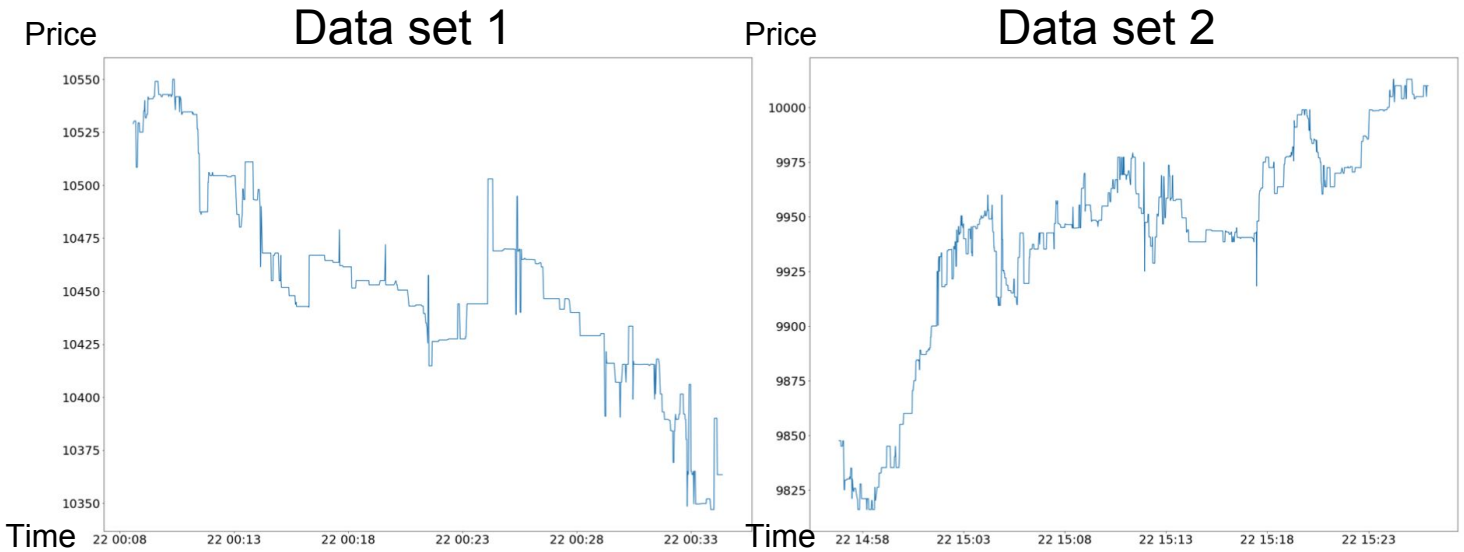


[1] https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf
[2] https://web.stanford.edu/class/psych209/Readings/MnihEtAlHassibis15NatureControlDeepRL.pdf

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

Real world data sets

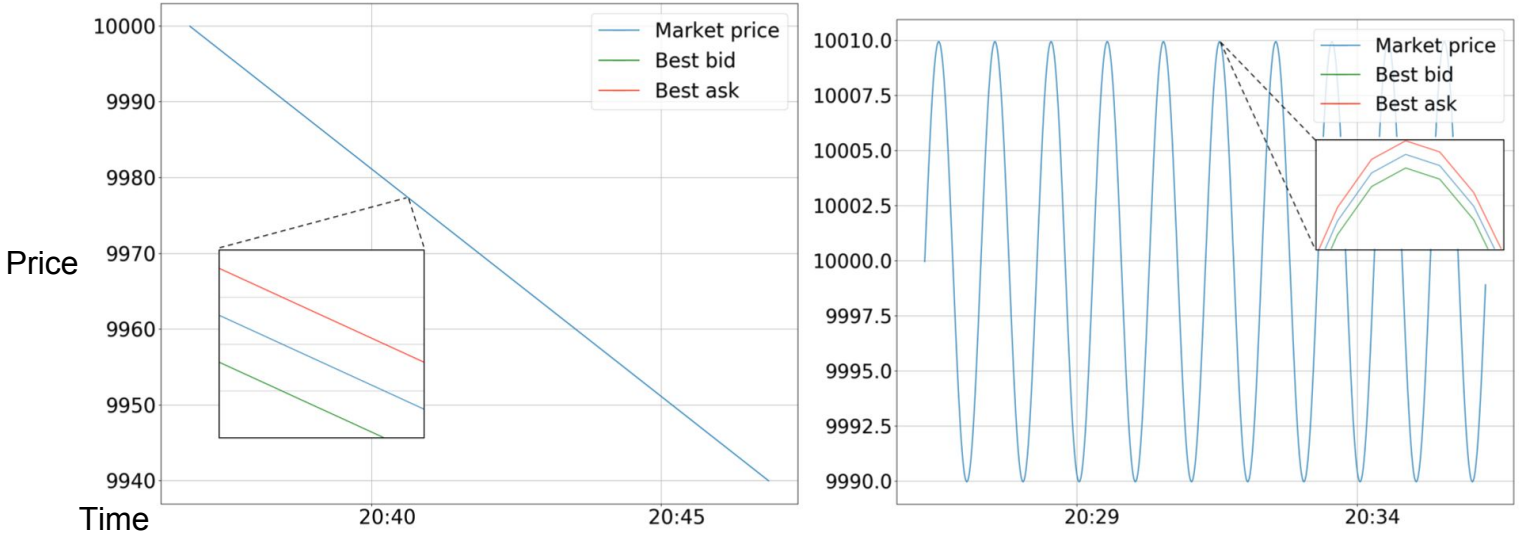


(a) 30 minute downwards trend

(b) 30 minute upwards trend

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

Artificial data sets

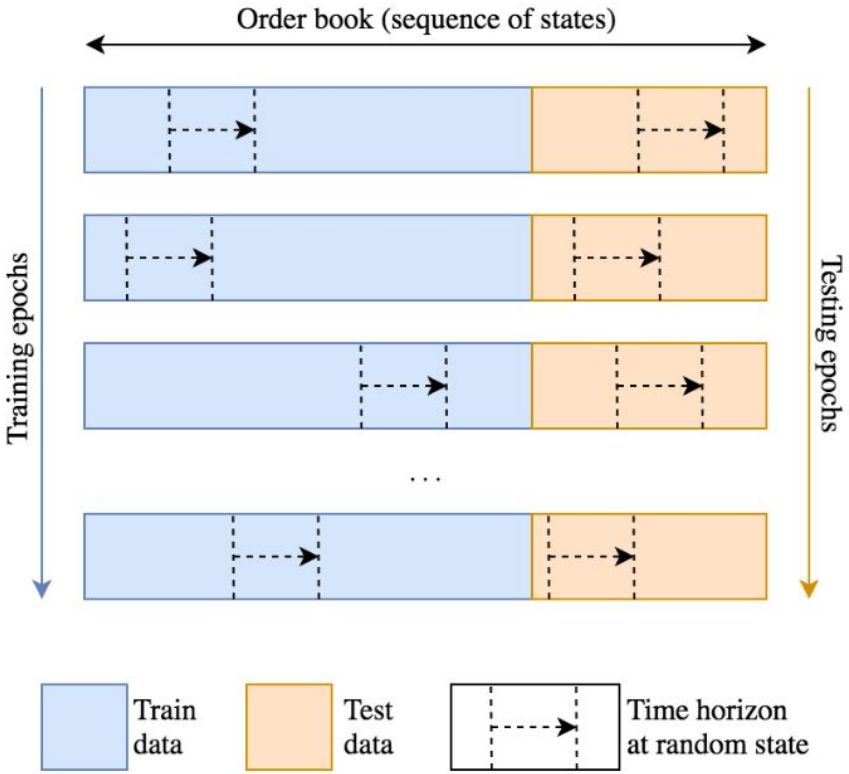


(a) Linear configuration of order book states with slope $a = -0.1$

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

Figure 6.15: Artificial order books with duration of 10 minutes

Training and Testing



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

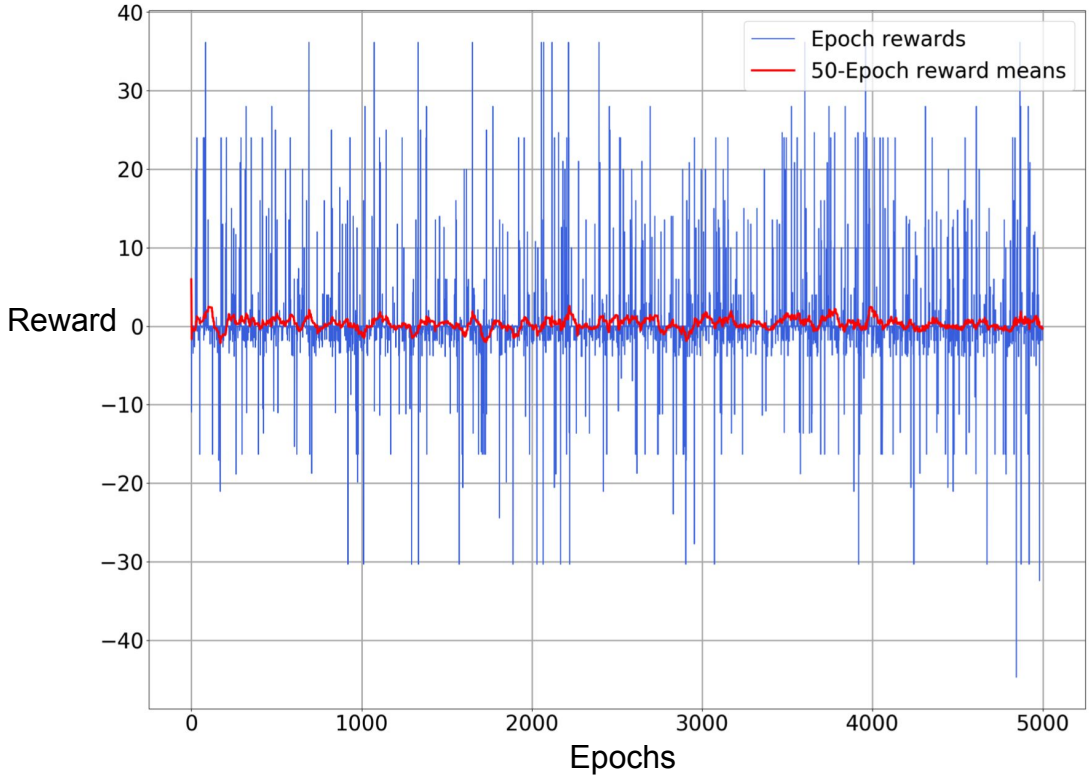
Empirical analysis

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

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

Q-Learning Agent

Training



Testing

Q-Learner	𝔼[Market Order]
-28.29	-30.53

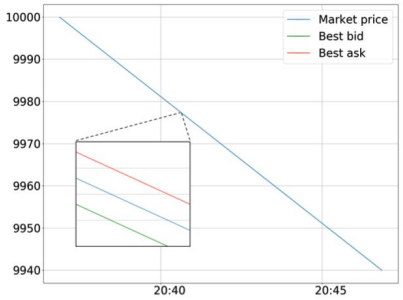
DQN Agent – Real world data

Multiple window sizes tested.

Feature I $\Sigma[B1, S1, B2, BS2]$	**Feature II** $\Sigma[B1, S1, B2, BS2]$	$\mathbb{E}$**[Market Order]**
-18.62	3.36	-30.53

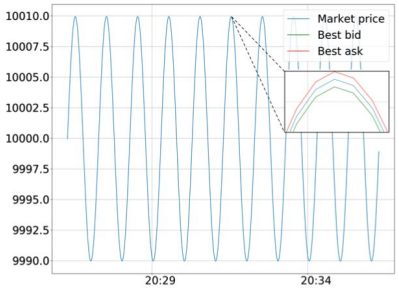
Table 6.6: Summary of rewards during backtest of DQN-CNN agent using different feature vector sizes.

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

DQN Agent – Limitations

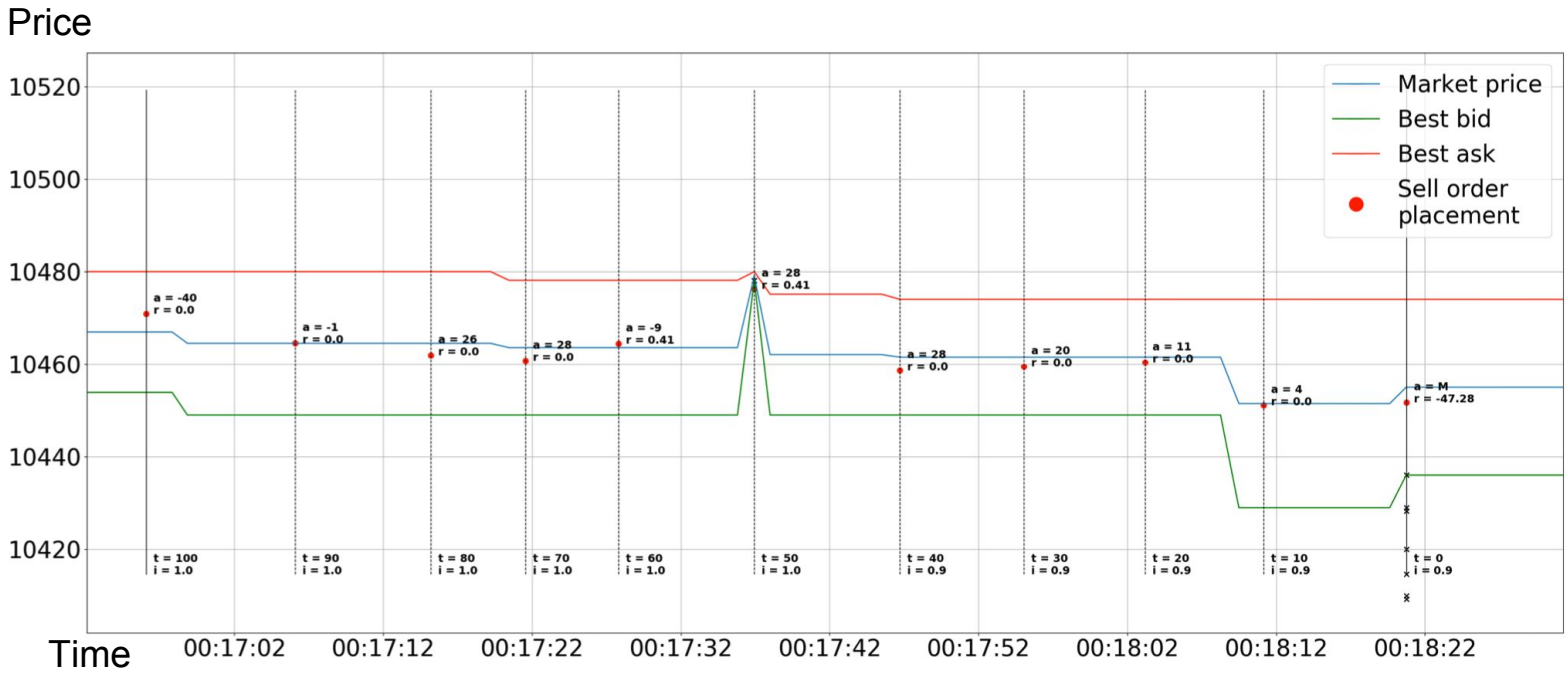


Figure 6.13: Wide spread between bid and ask prevents agent from selling.

DQN Agent – Limitations

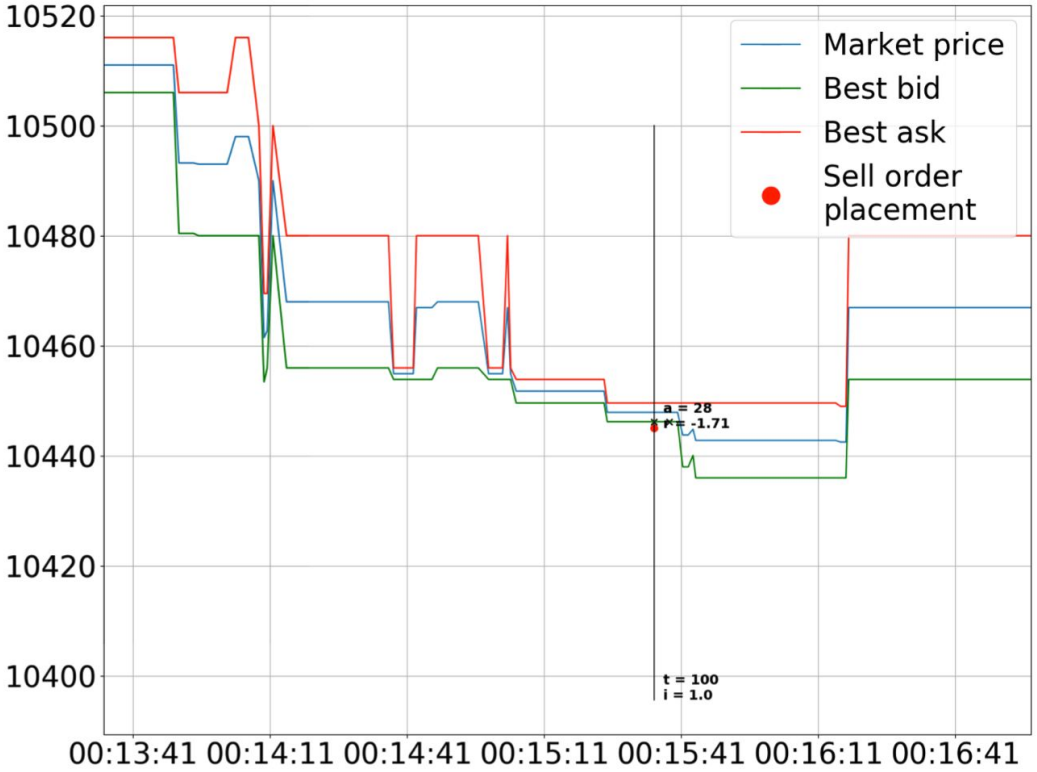


Figure 6.14: Agent is impatient and does not foresee trend change .

Performance overview (RQ 1.4)

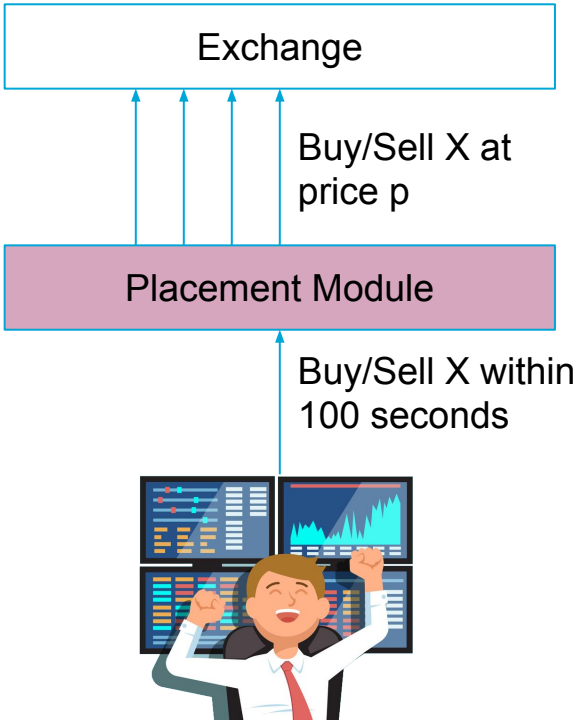
$\mathbb{E}$[Market Order]	$\mathbb{E}$[Limit order] (optimal)	Q-Learning	DQN-CNN (Feature I)	DQN-CNN (Feature II)
-30.53	-9.88	-28.29	-18.62	3.36

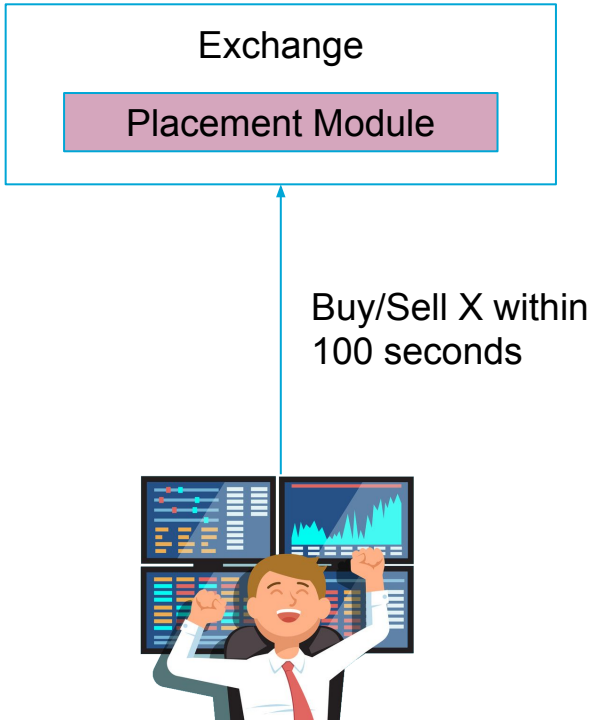
Table 6.7: Summary of expected and achieved average rewards from empirical evaluations and reinforcement learning applications.

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

In Practice





Recommendations

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

Questions?