亚马逊云科技

Reinforcement Learning in Finance

Kaige Yang
Applied Scientist Intern
AWS

Reinforcement Learning

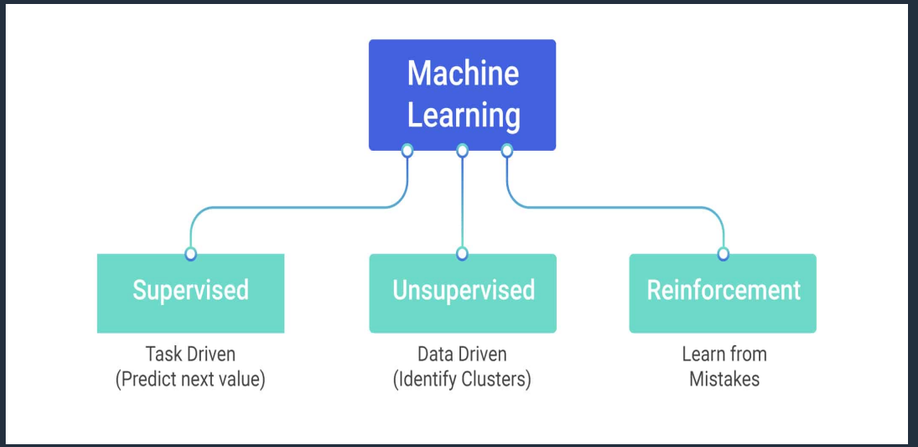
Reinforcement Learning (RL) is a branch of Machine Leanring (ML) for sovling sequential decision making under uncertainty

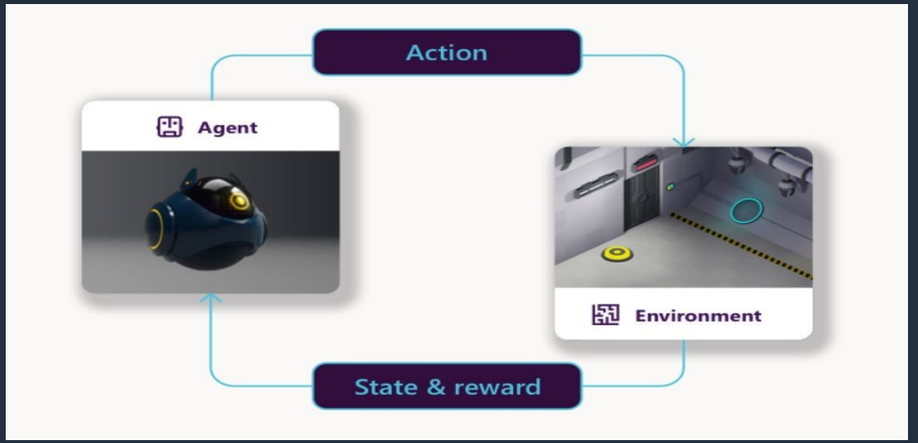
RL problem consists of agent and environment.

At each interaction,

1. The agent sees an observatio of the state of the environment.
2. Then decides on an action to take.
3. The agent received a reward (feedback) upon the action and current state.
4. The state of environment changes when the agent acts in it.
5. The process repeats until research the terminate state.

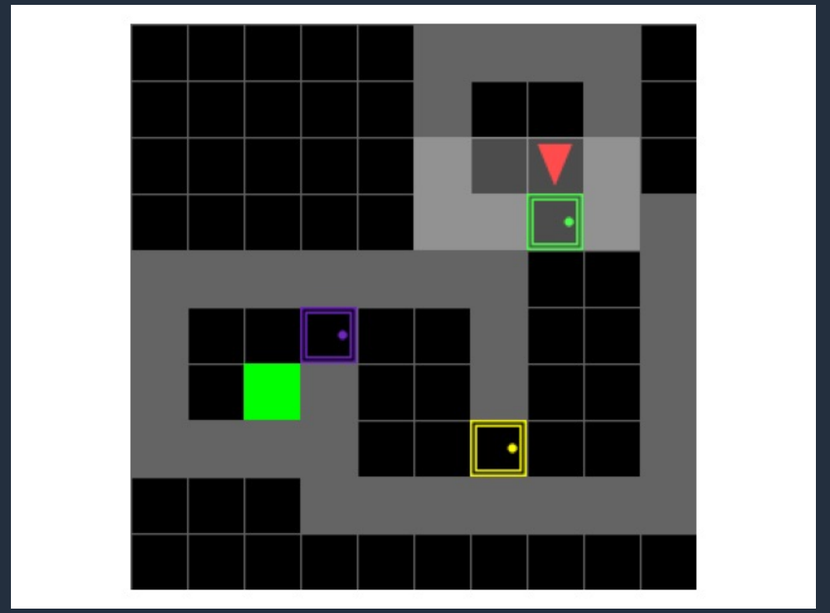
The goal of the agent is to maximize the cumulative rewards.





Example

Maze Navigation: Find the shortest path in a Maze environment.



Goal: arrive the green spot as soon as possible.

State: location of the agent

Action: up, down, left and right

Reward: -1 for each step, and -10 for hitting the wall.

To maximize the cumulative rewards, the agent is encouraged to find the shortest path to the goal.

Applications

RL methods have recentely enjoyed a wide variery of successes

- Playing games
- Resource relocation
- Traffic light controller
- Robotics
- Personalized Recommendations
- Manage an investment portfolio
- Control a power station

Markov Decision Processes (MDPs)

The class of problems solved by RL can be described as a mathematical framework Markov Decision Processes (MDPs).

A stochastic process has the Markov property if the conditional probability distribution of future states of the process depends only upon the present state. i.e., given the present, the future does not depend on the past.

An MDP is a 5-tuple $< S, A, R, P, P_0 >$

- S: state space is the set of all valid states.
- A: action space is the set of all valid actions.
- R: the reward function $r_t = R(s_t, a_t, s_{\{t+1\}})$.
- P: transition probability function $P\left(s_{\{t+1\}} \middle| s_t, a_t\right)$ is the proability of transitioning into state s' if action a is taken in state s.
- P_0: initial state distribution $s_0 \sim P_0(s)$.

Suppose at time step t, the state of environment is denoted as s_t. The agent takes an action a_t. Then, the environment state transits to the next state according to the transition probability $s_{\{t+1\}}$.The agent receives a reward upon the action taken according to the reward function r_t. This process repeats a course of T steps. The goal of the agent is to maximize the cumulative rewards.

The RL problem

The goal of RL is to select a policy which maximizes expected return.

The policy is a rule used by an agent to decide what actions to take under a state. The policy is usually denoted by π with

$$a_t \sim \pi(\cdot|s_t)$$

A trajectory τ is a sequence of states and actions

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

A reward is defined as

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

The undiscounted/discounted return of a trajectory is defined as

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

Where the discount factor $\gamma \in (0,1]$.

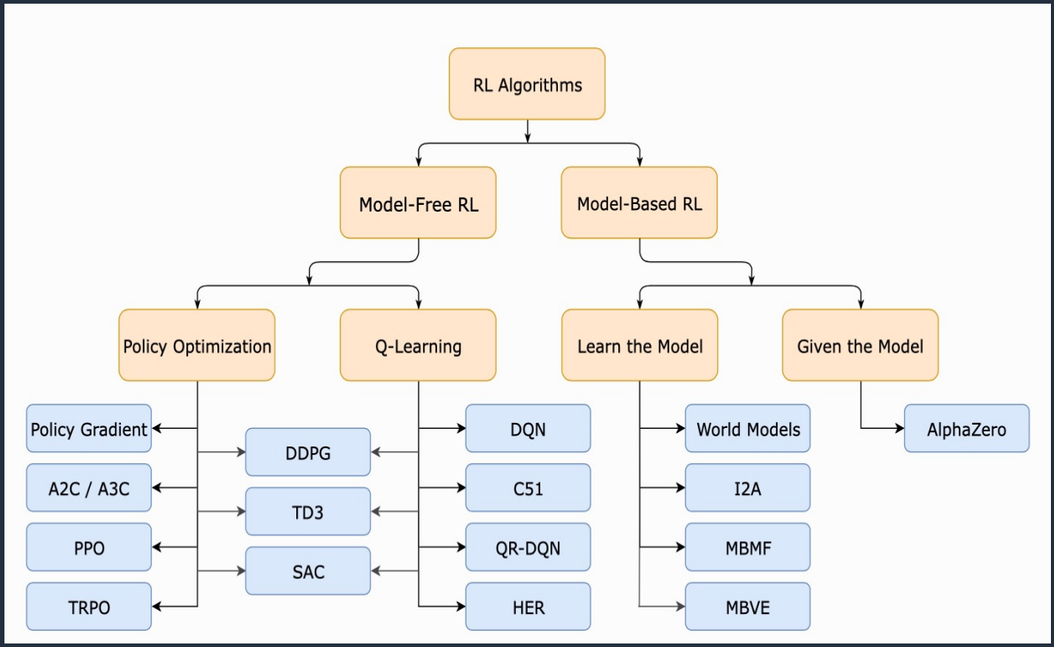
The expected return denoted by $J(\pi)$ is defined as

$$J(\pi) = E_{\tau\sim\pi}[R(\tau)]$$

The goal of RL algorithm is to find the optimal policy such that

$$\pi^* = argmax_\pi \ J(\pi)$$

RL Algorithms



Model-based and model-free algorithms

Whether the agent has access to (or learns) a model of the environment. Model referes to transition probability and reward $P(s_{\{t+1\}}|s_t, a_t)$ and $R(s_t, a_t, s_{\{t+1\}})$.

Policy-based and value-based

- Policy-based algorithms represent the policy explicitly as $\pi_\theta(\cdot|s)$ and optimize the parameter θ.

- Value-based algorithms represent the expected return of state-action pair as $Q_\theta(s, a)$. The agent takes the action with highest value

$$a_t = \arg\max\_\{a \in A\}\, Q_\theta(s_t, a)$$

Q-Learning

Q-learning is an off-policy RL approach that aims to learn the optimal action-value function.

Action-value function

$$Q(s_t, a_t) = r_t + \gamma \max_{\{a \in A\}} Q(s_{\{t+1\}}, a)$$

This function can be updated recursively as

$$Q^{\pi}(s_t, a_t) = E_{\{s_{\{t+1\}}\}}[R(s_t, a_t) + \gamma \max_{\{a \in A\}} Q^{\pi}(s_{\{t+1\}}, a)\ ]$$

After learning the action-value function, the optimal policy can be retrived by selecting the best action at every state.

$$\pi^*(s) = \arg\max_{\{a \in A\}} Q^{\pi}(s, a)$$

REINFORCE

REINFORCE is a policy gradient approach that aims to learn the policy directly instead of learning state-action value function and acting greedily.

The policy is parameterized by θ, denoted as $\pi_{\{\theta\}}(s)$.

Recall that the goal is to find the pollicy that maximize the expected return

$$\pi^* = \arg\max\_\{\pi\} J(\pi)$$

As π is a function of θ, the optimization problem is

$$\theta^* = \arg\max\_\theta \ J(\theta)$$

The problem can be solved by gradient descent to search local optimum. What we need is the gradient of θ.

Due to the Policy Gradient Theorem

$$\nabla_\theta J(\theta) = E[Q_\theta(s,a)\nabla_\theta \log \pi_\theta(s,a)]$$

Therefore, θ can be updated via

$$\theta \ \leftarrow \theta + \alpha \, Q(s_t,a_t)\nabla_\theta \log \pi_\theta(s_t,a_t)$$

RL in finance

Motivation

Given that actions taken in finance and trading may have long-term effects not immediately measurable, some financial problems can be viewed as sequential decision problems, and the environment in which these areas work may be very large or continuous, reinforcement learning can be well-suited to solving finance problems.

Applications

- Stock trading
- Portfolio management
- Market making
- Pricing and hedge
- Order excution

Use RL as a solution

1. Formalize the problem as a MDP
2. Chose a suitable RL algorithm

Papers

- Modern perspective on reinforcement learning in finance
- Deep Reinforcement Learning in Quantitative Algorithmic Trading: A Review
- Deep Reinforcement Learning for Active High Frequency Trading
- Double Deep Q-Learning for Optimal Execution
- Deep Graph Convolutional Reinforcement Learning for Financial Portfolio Management - DeepPocket
- Market Making via Reinforcement Learning
- Hedging Derivatives Under Generic Market Frictions Using Reinforcement Learning

Deep Reinforcement Learning for Active High Frequency Trading

Problem setting

Trade one unit of INTEL stock at each second during a trading day. The goal is to maximize the net profit at the end of the day.

Formalize as a MDP

- State: the LOB current state and historical LOB state within a window
- Action: [buy, sell, stay]
- Reward: net value change $P_t - P_{\{t-1\}}$

Algorithm

PPO

Data

Train: 60 days 04-02-2019, 30-04-2019
Test: 22 days 01-05-2019, 31-05-2019

Results

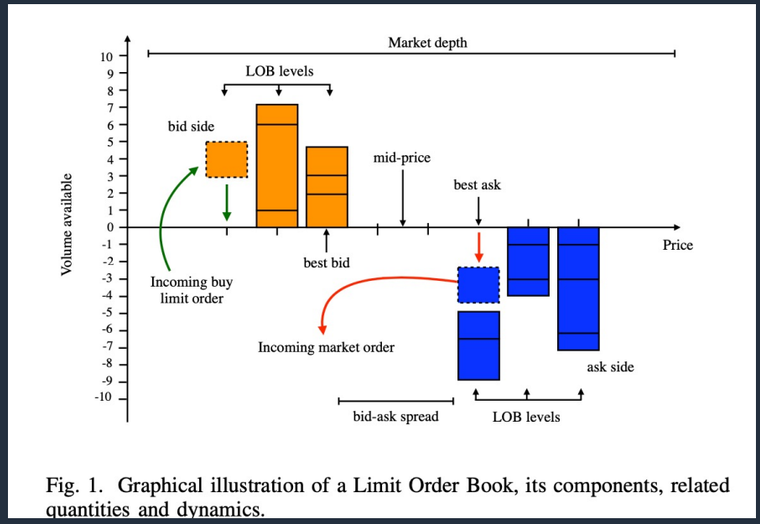
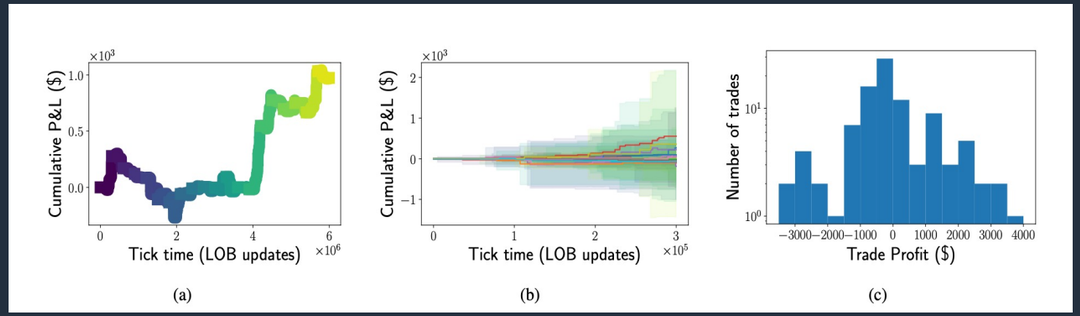


Fig. 1. Graphical illustration of a Limit Order Book, its components, related quantities and dynamics.



Deep Graph Convolutional Reinforcement Learning for Financial Portfolio Management - DeepPocket

Portfolio management

The process of selecting and managing a group of financial instruments such as stocks, bonds and securities, is called portfoli management. Portfolio management aims at maximizing the profit on investment while minimizing the risk.

Problem Setting

Given a mount of fund P_0 and a set of assets m. At each trading day t, set the weight of each asset $w_t = [w_{\{0,t\}}, w_{\{1,t\}}, \dots, w\_\{m, t\}]$. The goal is to maximize the portfolio value P_T at day T.

Formalize as a MDP

- State S : $s_t = [f_{\{0,t\}}. f_{\{1,t\}}, \dots, f\_\{m, t\}]$ the feature of each asset and the market
- Action A: $a_t = w_t$ valid weights of assets
- Reward r: $r_t = \ln \frac{P_t}{P_{\{t-1\}}}$ the log ratio between portfolio value as day t and day $t-1$.
- Goal G: $G = \ln \frac{P_T}{P_0}$

Deep Graph Convolutional Reinforcement Learning for Financial Portfolio Management - DeepPocket

Algorithm

1. Autoencoder: feature extraction
2. GCN: exploit the correlation between assets
3. Actor: learn policy function
4. Critic: learn action-value function

State: each asset is described by a vector $f_{\{i,t\}} = [open, close, low, high, moving\ averaging\ momentum, \dots]$

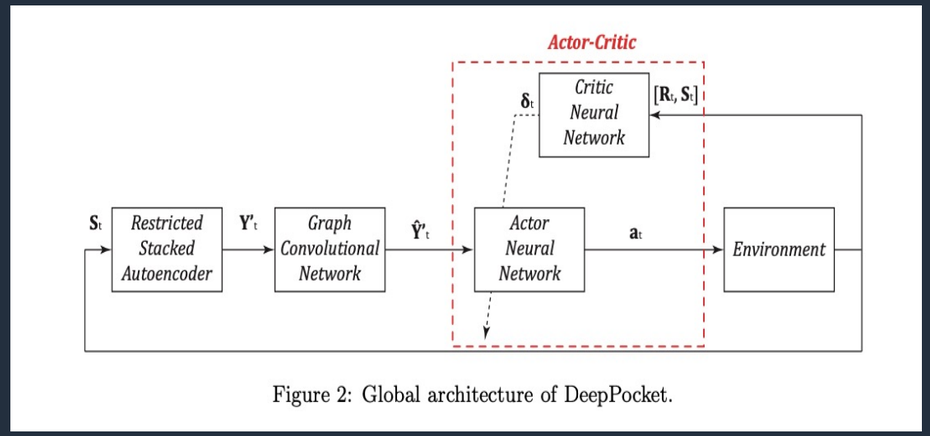


Figure 2: Global architecture of DeepPocket.

Algorithm 4 Actor-Critic Algorithm

Input: Initial network weights: $\theta_{a,init}, w_{c,init}$
Data: $\alpha_a, \alpha_c \in [0,1]$: Actor and critic learning rate
Data: θ_a, w_c : Actor and critic neural network weights
Data: N_t : Number of iterations
Initialization:
$iteration = 0$
$\pi_\theta(\mathbf{s}, \mathbf{a}) \in \mathcal{A}$
while $iteration \leq N_t$ **do**
  $\Delta_w = \alpha_c \delta_v \nabla_w V(\mathbf{s})$
  $w_c \leftarrow w_c + \Delta_w$
  $\Delta_\theta = \alpha_a \delta_v \nabla_\theta \log \pi_\theta(\mathbf{s}, \mathbf{a})$
  $\theta_a \leftarrow \theta_a + \Delta_\theta$
  Update state and action:
  $\mathbf{s} \leftarrow \mathbf{s}'$
  $\mathbf{a} \leftarrow \mathbf{a}'$

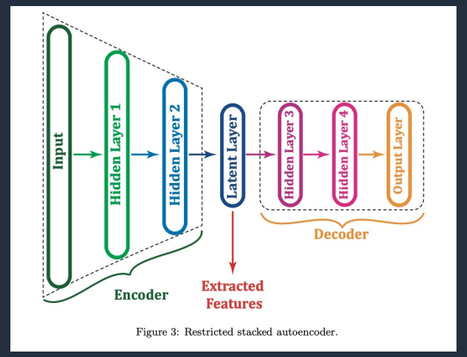


Figure 3: Restricted stacked autoencoder.

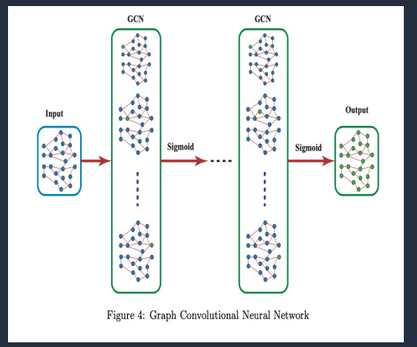


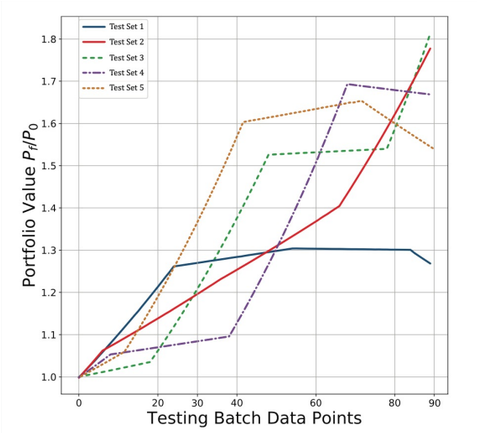
Figure 4: Graph Convolutional Neural Network

Deep Graph Convolutional Reinforcement Learning for Financial Portfolio Management - DeepPocket

Experiments

- Data: historical data covers Jan 2002– March 2020
- Asset set: 28 stocks cover several industries
- Training: training set and test set

Results



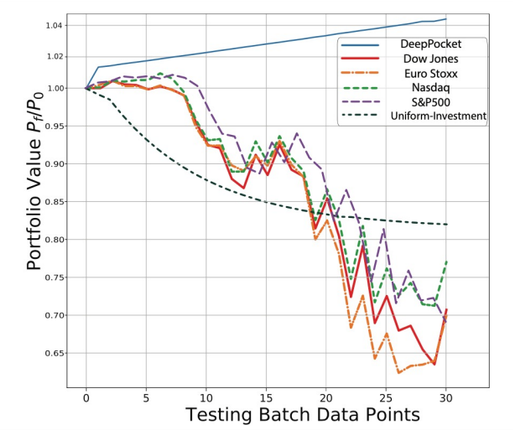


Table 2: Composition of our Portfolio for the period in between January 2, 2002, and February 10, 2020.

Sector	Name
Technology	Apple.Inc (AAPL) Cisco Systems Inc. (CSCO) Intel Corporation (INTC) Oracle Corporation (ORCL) Microsoft Corporation (MSFT) International Business Machines Corporation (IBM) Honeywell International Inc. (HON) Verizon Communications Inc. (VZ)
Financial Services	Manulife Financial Corporation (MFC) JPMorgan Chase & Co. (JPM) Bank of America Corporation (BAC) The Toronto-Dominion Bank (TD)
Industries	3M Company (MMM) Caterpillar Inc. (CAT) The Boeing Company (BA) General Electric Company (GE)
Consumer Defensive	Walmart Inc. (WMT) The Coca-Cola Company (KO)
Consumer Cyclical	The Home Depot, Inc. (HD) Amazon.com, Inc. (AMZN)
Healthcare	Johnson & Johnson (JNJ) Merck & Co., Inc. (MRK) Pfizer Inc. (PFE) Gilead Sciences, Inc. (GILD)
Energy	Enbridge Inc. (ENB) Chevron Corporation (CVX) BP p.l.c. (BP) Royal Dutch Shell plc (RDSB.L)

Table 3: Training and test sets

ID	Training Set	Test Set
Test 1	2002-01-02 to 2009-04-16	2010-03-15 to 2010-07-21
Test 2	2002-01-02 to 2012-12-06	2013-11-04 to 2014-03-14
Test 3	2002-01-02 to 2016-08-01	2017-06-28 to 2017-11-02
Test 4	2002-01-02 to 2018-10-19	2019-06-09 to 2019-10-16
Test 5	2002-01-02 to 2019-06-23	2019-11-12 to 2020-03-24

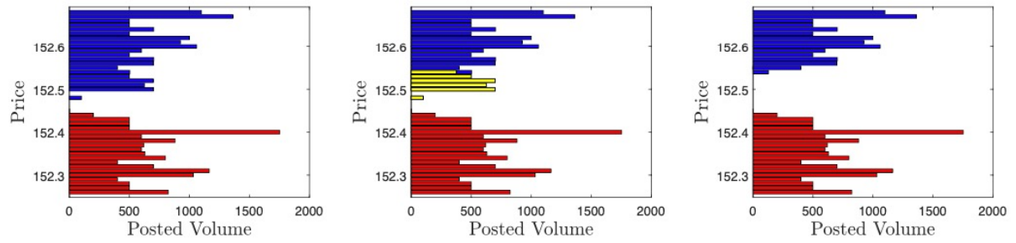
Double Deep Q-Learning for Optimal Execution

Optimal Execution

How to execute large positions over a given trading horizon is an important problem faced by investors. Naively rebalancing a portfolio could result in significant adverse price movements as other traders may read off the signal.
The trader must balance trade quickly and obtaining poor execution prices with trading slowly which exposes them to unknown market fluctation.

Problem setting

Assume a trader who at $t = 0$ holds an inventory of q_0 shares and must fully liquidated it by the end of time $t = T$. The trader sells x_t shares at evenly distributed time-steps $t = \{0,1,2 \dots, T-1\}$ at the mid-price p_t on the LOB. The goal is to maximize the total profit after selling all shares at time T



(a) Before MO Arrival (b) Buy MO walking the LOB (c) After MO Arrival

Figure 1. *The LOB of Facebook Inc. stock at 10:36 on 28 Mar, 2018 as a buy MO of size 3,000 arrives. The blue (red) bars represent available volume of sell (buy) orders at the corresponding price. The yellow bars in the middle panel indicate which LOs match the incoming MOs.*

Double Deep Q-Learning for Optimal Execution

Formalize as a MDP

- State: LOB current state and previous state, the remaining invertory
- Action: the number of share to sell x_t
- Reward: the profit due to selling x_t
- Goal: sell all shares and maximize the total profit after time horizon T

Algorithm

Double DQN

Data

LOB data ranges from Jan-02-2017 to March-30-2018.
Stocks: APPL, AMZN, FB, GOOG, INTC, MSFT and NTAP

Result

Compare profit with as basedline called TWAP (Time-Weighted Average Price) i.e, selling the same number of shares at each action-executable timestep

Table 1

Relative P&L performance for all stocks with respect to TWAP strategy.

		$\Delta P\&L$				
Ticker	Features	Median	Mean	Std.Dev.	GLR	$\mathbb{P}(\Delta P\&L > 0)$
AAPL	TIP	2.92	2.85	6.1	1.0	77.4%
	TIPQV	2.68	2.68	5.3	1.2	76.2%
AMZN	TIP	0.06	-0.28	11.3	0.9	50.1%
	TIPQV	-0.02	-0.15	11.7	1.0	49.9%
FB	TIP	2.61	2.52	7.6	1.2	68.1%
	TIPQV	2.29	2.26	8.3	1.1	64.3%
GOOG	TIP	-0.59	0.21	7.4	1.3	45.7%
	TIPQV	-0.40	0.05	11.2	1.1	47.2%
INTC	TIP	11.63	11.08	5.6	2.5	95.8%
	TIPQV	11.96	11.40	4.0	3.6	97.8%
MSFT	TIP	5.97	5.93	3.0	3.1	97.4%
	TIPQV	5.89	5.95	3.8	2.8	94.8%
NTAP	TIP	10.13	9.62	5.0	1.8	96.4%
	TIPQV	10.05	9.64	5.0	2.1	96.2%
SMH	TIP	3.80	2.67	6.7	1.0	73.5%
	TIPQV	4.90	4.86	3.9	2.1	91.8%
VOD	TIP	14.68	13.99	5.2	4.1	97.8%
	TIPQV	15.72	15.43	3.1	18.6	99.2%

Deep Reinforcement Learning in Quantitative Algorithmic Trading: A Review

- Many of the reviewed studies had only proof-of-concept ideals with experiments conducted in unrealistic settings and no real-time trading applications.
- If their new algorithm performed as well in the real-world, they certainly would not publish a paper about it and give away their edge.
- More extensive experiments on live-trading platforms rather than back-testing or very limited real-time trading.
- Direct comparisons between DRL trading agents with human traders
- More comparisons among state-of-the-art DRL approaches under similar conditions and data sources.

Frameworks

- **FinRL**: Deep Reinforcement Learning for Quantitative Finance
- **MAXE**: Market simulator designed by oxford man institute
- **Trading Gym**: A toolkit for developing and comparing reinforcement learning trading algorithms
- **Stock market reinforcement learning**: OpenAI Gym Environment with Deep Reinforcement Learning using Keras
- **PGPortfolio**: A toolkit of portfolio management research