最佳訂單執行問題的強化學習演算法
A Reinforcement Learning Algorithm for Optimal Order Execution Problem

報告者：陳建宇
指導教授：郭美惠 教授

國立中山大學應用數學系

2019-1-23

Contents

Introduction

This study mainly discuss the optimal order execution problems in limit order book with two methods.

(1) Partition Algorithm

(2) Reinforcement Learning

- Given market depth (Q_i), spread (d_i) and limit order, the objective is to find the optimal order execution (u_i).
- We compare the execution costs for the two algorithms with the naive methods.

Limit order book

- Limit Order Books are the financial tools for investors to trade stock through the use of buy and sell orders.
- It contains three types of orders
  (1) Limit order
  (2) Market order
  (3) Cancellation order

- The general case of limit order book.

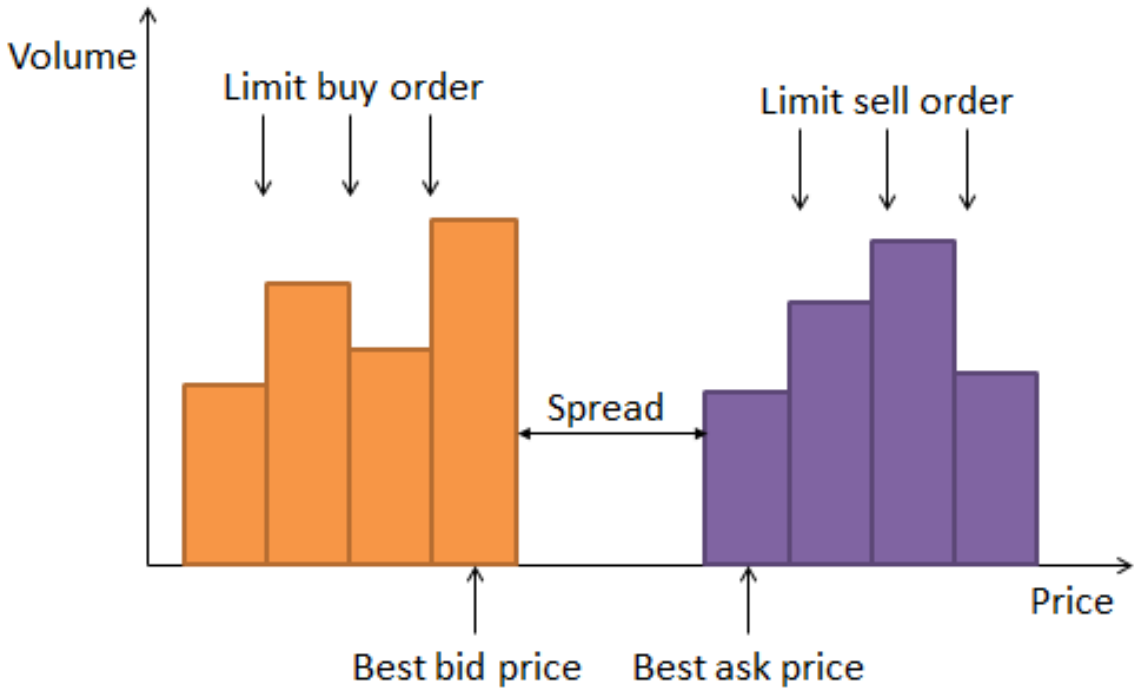


Figure: The illustration of limit order book

- When the market order is submitted.

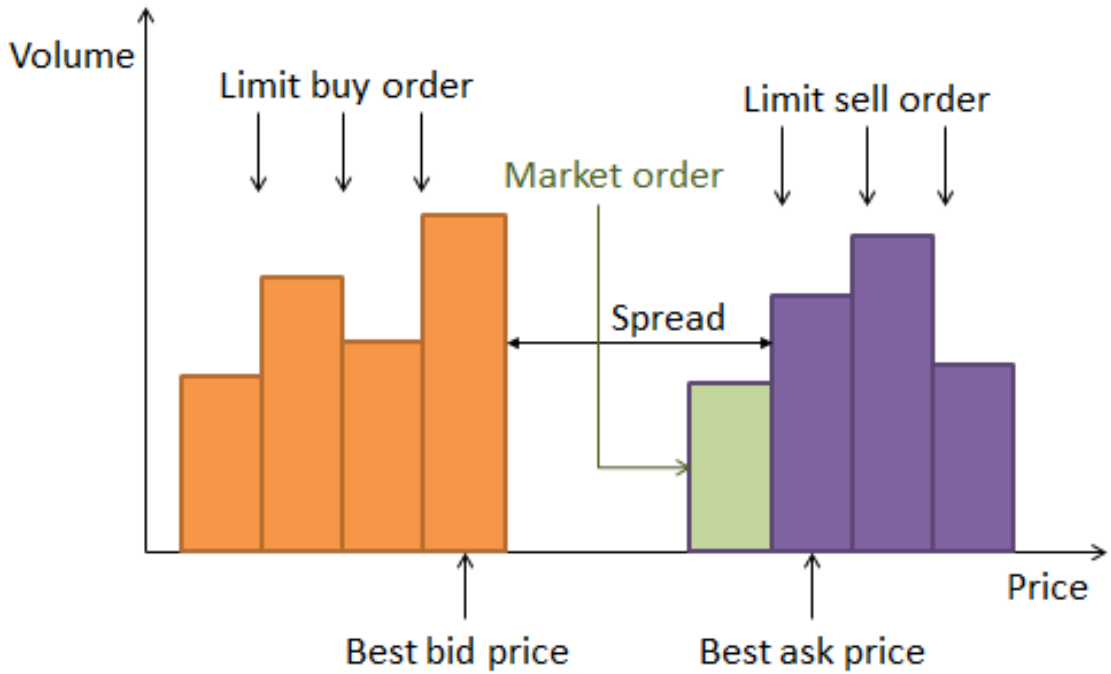
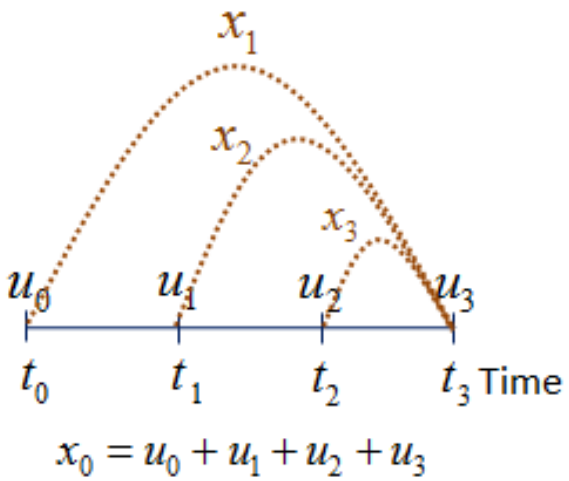


Figure: The illustration of limit order book with market order

LOB Model

We follow the LOB model of Chen, Kou, and Wang (2017)[2], and the parameters of the model at time t_i are introduced below

(1) Best amount of order (u_i)
(2) Remaining shares (x_i)
(3) Fundamental value of stock (M_i)
(4) Spread (d_i)
(5) Market depth (Q_i)
(6) Transition probability (P_{jk})
e.g. three state (s_1, s_2, s_3)
then $P_{jk} = P(s_k|s_j)$



Optimizing problems

- The objective is to find the executed orders which have the total minimum capital cost.
- If we buy u_i shares at time t_i, then the average price per share for the transcation is $M_i + d_i + \frac{u_i}{2Q_i}$
- The capital cost of target is

$$\min_{u_i} \mathbb{E}\left[\sum_{i=0}^{n} u_i \left(M_i + d_i + \frac{u_i}{2Q_i}\right)\right]$$

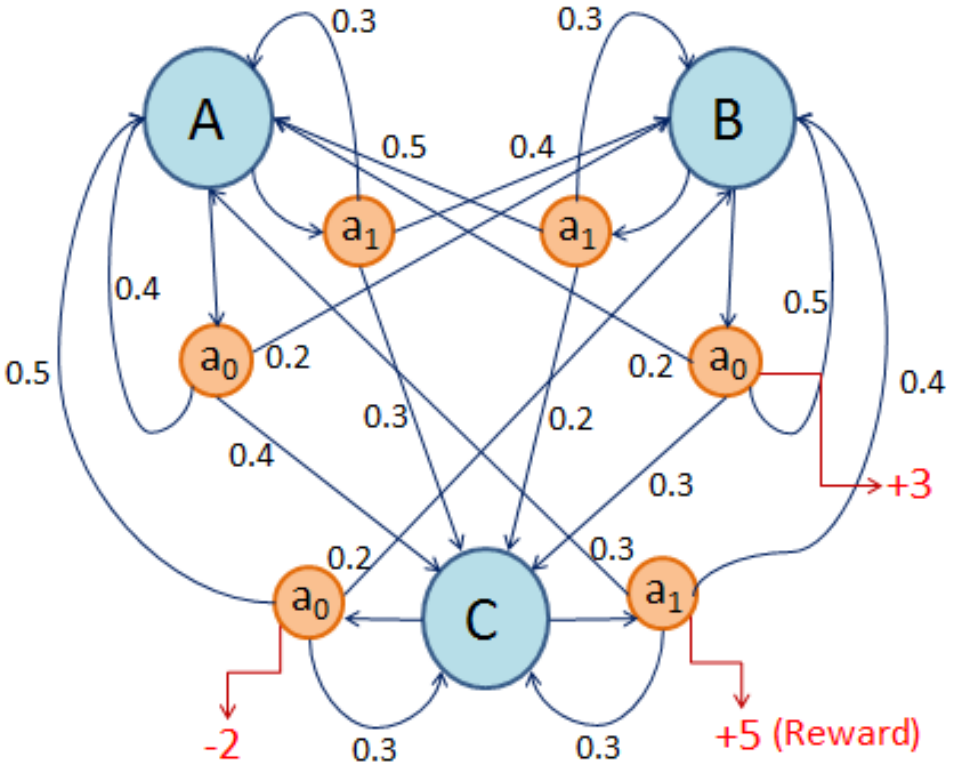
- Since M_i is martingale, then the capital cost of target can be simplified as

$$\min_{u_i} \mathbb{E}\left[\sum_{i=0}^{n} u_i \left(d_i + \frac{u_i}{2Q_i}\right)\right]$$

Markov decision process

- A Markov decision process (MDP) is a discrete time stochastic process.
- A Markov decision process mainly consists of four parts
  (1) State (S)
  (2) Transition probaibilies (P_{ij})
  (3) Different actions in each state (A)
  (4) Reward after execution (R_i)
- If we select an action a_i, the probability distribution over next states is the same as the last time we try the same action in the same state.

- A Markov decision process for the limit order book.



- Three state A, B and C(market depth).
- Two actions a_0 and a_1(amount of orders).
- The black numerical values near blue lines represent transition probabilities.
- The red numerical values near red lines represent reward.

- Combine the Markov decision process with the optimizing problem, we have the equation

$$\min_{u_i} \mathbb{E}\left[\sum_{i=0}^{n} u_i \left(d_i + \frac{u_i}{2Q_i}\right)\right] = \min_{\{u_i\}_{i=0}^{n}} \mathbb{E}\left[\sum_{i=0}^{n} \begin{pmatrix} u_i \\ \mathbf{x}_i \end{pmatrix}' C_{S_i,i} \begin{pmatrix} u_i \\ \mathbf{x}_i \end{pmatrix}\right]$$

where $C_{S_i,i} = \begin{pmatrix} 1/2Q_i & 0 & 1/2 \\ 0 & 0 & 0 \\ 1/2 & 0 & 0 \end{pmatrix}$ and state variables $\mathbf{x}_i = (x_i, d_i)$
(x_i = remaining shares, d_i = spread)

- We consider the following optimization problem for the LOB model

$$\min_{\{u_i\}_{i=0}^n} \mathbb{E}\left[\sum_{i=0}^{n} \begin{pmatrix} u_i \\ \mathbf{x}_i \end{pmatrix}' C_{S_i,i} \begin{pmatrix} u_i \\ \mathbf{x}_i \end{pmatrix}\right]$$

$$\text{s.t.} \quad F'_{S_i,i}\mathbf{x}_i \le u_i \le G'_{S_i,i}\mathbf{x}_i, \ i = 0, 1, \cdots, n$$

$$\mathbf{x}_{i+1} = A_{S_i,i}\mathbf{x}_i + B_{S_i,i}u_i, \ i = 0, 1, \cdots, n-1$$

where $A \in \mathbb{R}^{m\times m}$, $B \in \mathbb{R}^m$, $C \in \mathbb{R}^{(m+1)\times(m+1)}$, $F, G \in \mathbb{R}^m$

- Since $F'_{S_i,i}\mathbf{x}_i \le u_i \le G'_{S_i,i}\mathbf{x}_i$ is the constraint for the amount of order, then we have

$$F_{S_i,i} = \begin{cases} (0,0)', & i = 0, 1, \cdots, n-1 \\ (1,0)', & i = n \end{cases} \quad \text{and } G_{S_i,i} = (1,0)'$$

- Since $\mathbf{x}_{i+1} = A_{S_i,i}\mathbf{x}_i + B_{S_i,i}u_i$ indicates the relationship for remaining shares and spread between t_i and t_{i+1}, the functions are shown below

$$\begin{cases} x_{i+1} = x_i - u_i \\ d_{i+1} = e^{-\rho\Delta t}d_i + \frac{e^{-\rho\Delta t}}{Q_i}u_i \end{cases}$$

then we have

$$A_{S_i,i} = \begin{pmatrix} 1 & 0 \\ 0 & e^{-\rho\Delta t} \end{pmatrix}, \; B_{S_i,i} = \left(-1, \frac{e^{-\rho\Delta t}}{Q_i}\right)'$$

- Using the framework for dynamic programming, we denote $J_i(\mathbf{x}_i, S_i)$ as the optimal value function at time t_i and $u_i^*(\mathbf{x}_i, S_i)$ as the optimal policy, by Bellman equation, we have the value of a state-action pair as

$$V_i(\mathbf{x}_i, S_i, u_i) = \begin{pmatrix} u_i \\ \mathbf{x}_i \end{pmatrix}' C_{S_i,i} \begin{pmatrix} u_i \\ \mathbf{x}_i \end{pmatrix} + \mathbb{E}[J_{i+1}(A_{S_i,i}\mathbf{x}_i + B_{S_i,i}u_i^*, S_{i+1})|S_i]$$

where

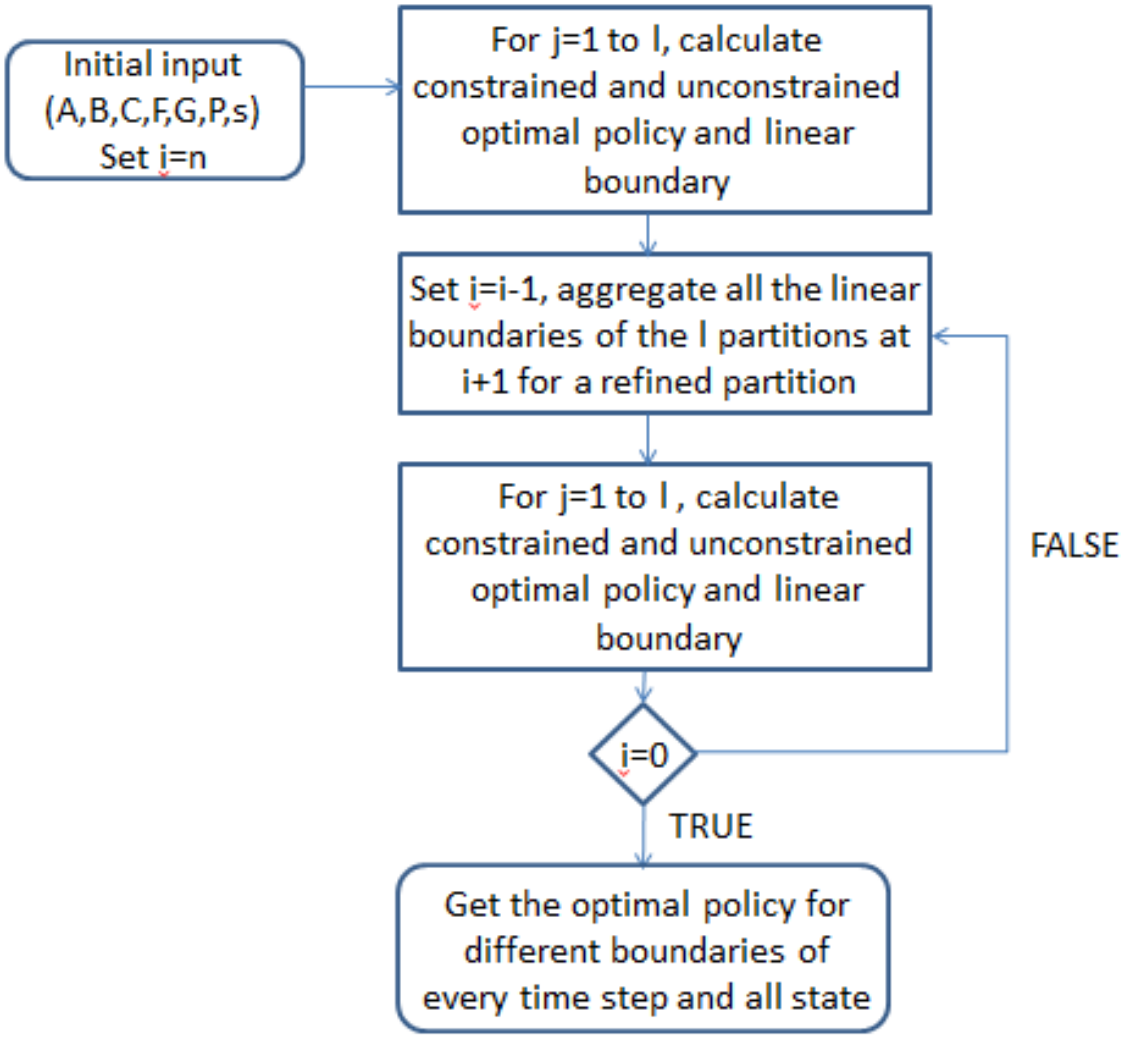
$$J_i(\mathbf{x}_i, S_i) = \min_{F'_{S_i,i}\mathbf{x}_i \le u_i \le G'_{S_i,i}\mathbf{x}_i} \{V_i(\mathbf{x}_i, S_i, u_i)\}$$

.

Partition algorithm

- Since the optimal policy is a linear function of the state variables $\mathbf{x}_i$, and the optimal value function is a quadratic function of the state variables $\mathbf{x}_i$.
- Chen, Kou and Wang (2017)[2] propose a partitioning algorithm to characterize the analytical solution of this class of Markov decision process.

- The flow chart of partition algorithm



The process of partition algorithm contains the following steps

1. Set parameters: We first set the coefficients matrix , denoted $A_{s_j,i}$, $B_{s_j,i}$, $C_{s_j,i}$, $F_{s_j,i}$ and $G_{s_j,i}$, transition probability matrix of market depth P_{jk} and the state sets, $S = \{s_1, s_2, \cdots, s_l\}$.
2. Set $i = n$ and $j = 1$, compute the cost function of the last stage

$$au_i^2 + b'\mathbf{x}_i u_i + \mathbf{x}_i' c\mathbf{x}_i \triangleq (u_i, \mathbf{x}_i) C_{s_j,i} (u_i, \mathbf{x}_i)'$$

   then find the unconstrained optimal solution $\tilde{u}_i^*$.
3. Partition the state space into four regions and compute the constrained optimal policy of following regions:

Regions	Optimal Policy u_i^*
$(F_{s_j,i} - G_{s_j,i})'\mathbf{x}_i \geq 0$	infeasible
$\tilde{u}_i^* \leq F'_{s_j,i}\mathbf{x}_i \leq G'_{s_j,i}\mathbf{x}_i$	$F'_{s_j,i}\mathbf{x}_i$
$F'_{s_j,i}\mathbf{x}_i \leq \tilde{u}_i^* \leq G'_{s_j,i}\mathbf{x}_i$	$\tilde{u}_i^*$
$F'_{s_j,i}\mathbf{x}_i \leq G'_{s_j,i}\mathbf{x}_i \leq \tilde{u}_i^*$	$G'_{s_j,i}\mathbf{x}_i$

4. Compute the value function in the above regions by letting $u_i = u_i^*$ in cost function.
5. Repeat the step 2 to step 4 for $j = 2$ to $j = l$.
6. Set $i = n - 1$, $j = 1$ and aggregate all the linear boundaries of the l partitions into a refined partition.
7. Compute the cost function in all feasible regions $\mathcal{P}$ for some $S_{t+1} = s_k$ and $P_{jk}^{(i)} > 0$, then the cost function denoted as

$$au_i^2 + b'\mathbf{x}_i u_i + \mathbf{x}_i' c\mathbf{x}_i \triangleq V_i(\mathbf{x}_i, S_i, u_i)$$

then find the unconstrained optimal solution $\tilde{u}_i^*$.

8. Partition the state space into three regions and compute the constrained optimal policy of following regions:

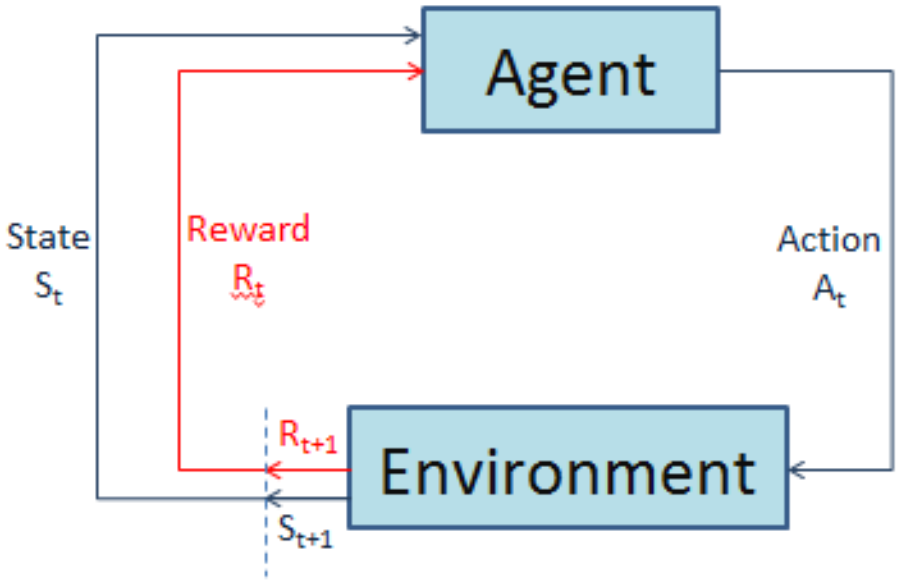
Regions	Optimal Policy u_i^*
$A_{s_j,i}\mathbf{x}_i + B_{s_j,i}(F_{s_j,i})'\mathbf{x}_i \in \mathcal{P},\ \tilde{u}_i^* \leq F'_{s_j,i}\mathbf{x}_i$	$F'_{s_j,i}\mathbf{x}_i$
$A_{s_j,i}\mathbf{x}_i + B_{s_j,i}\tilde{u}_i^* \in \mathcal{P},\ F'_{s_j,i}\mathbf{x}_i \leq \tilde{u}_i^* \leq G'_{s_j,i}\mathbf{x}_i$	$\tilde{u}_i^*$
$A_{s_j,i}\mathbf{x}_i + B_{s_j,i}(G_{s_j,i})'\mathbf{x}_i \in \mathcal{P},\ G'_{s_j,i}\mathbf{x}_i \leq \tilde{u}_i^*$	$G'_{s_j,i}\mathbf{x}_i$

9. Compute the value function in the above regions by letting $u_i = u_i^*$ in cost function.
10. Repeat step 6 to step 9 for $j = 1$ to $j = l$
11. Repeat step 6 to step 10 for $i = n - 1$ to $i = 0$

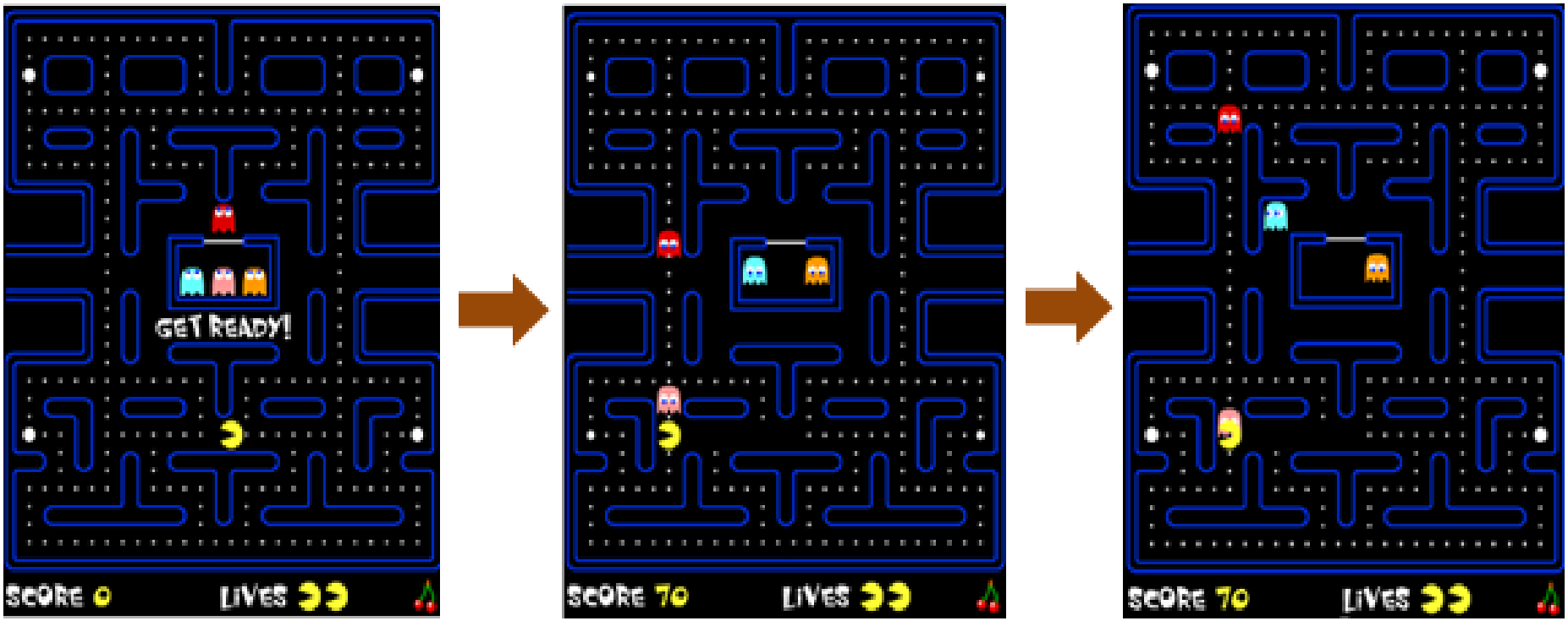
Reinforcement learning

- Reinforcement learning is a method that we find the connection with environment by taking some policy, which produces a wealth of information about cause and effect, about the consequences of actions, and about how to do in order to achieve goals we set.
- The learner is not told which actions to take and must discover which actions yield the most reward by trying them.

- Reinforcement learning is learning from the feedback after taking actions, which may affect not only the immediate reward but also the next situation and all subsequent rewards.
- A basic structure of reinforcement learning is shown as follows



- The Pac-Man game is shown below



Reward=+70

Reward= -100

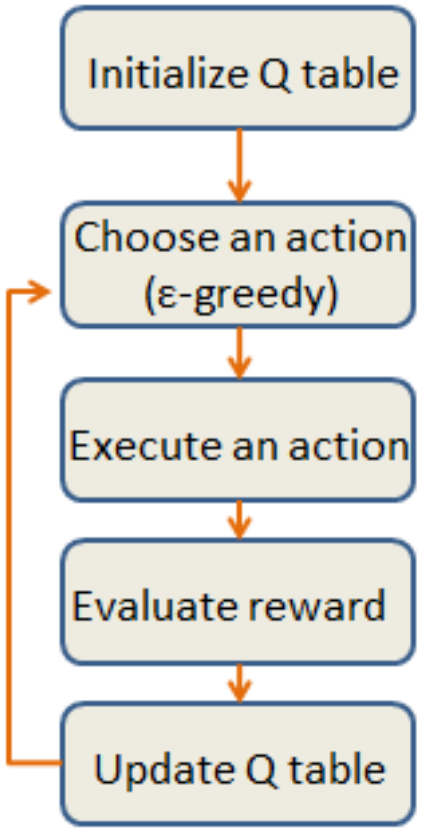
- Q-learning is a form of model-free reinforcement learning, it provides agents with the capability of learning to act optimally in Markov process by experiencing the consequences of actions.
- By trying all actions in all state repeatedly, it judges by long-term discounted reward to learn which are best outcome.
- We define the action-value function $Q(s, a)$

$$Q(s, a) = \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots | s_t = s, a_t = a]$$

- Q-table contains every action-value function of corresponding state and action
- We choose actions by epsilon greedy.
- We update Q-table by the function

$$Q(s,a) \leftarrow Q(s,a)+\alpha[r_t+\gamma \max_{a'} Q(s',a')-Q(s,a)]$$

- The flow chart of Q-learning is shown.



The process of algorithm is shown in following step

1. We first set the initial value to every parameter, such as discount rate γ, learning rate α, and etc., and denote the state sets, $S = \{s_1, s_2, \cdots, s_l\}$, the action sets, $A = \{0, 1, 2, \cdots, x_0\}$, and the reward function, $R_t = -a_t \times \left(d_t + \frac{a_t}{2 \times S[t]} \right)$, then set the Q-table for each time step with initial value zero. Recall x_0 is total demand of shares that investor wants to buy.
2. Given the remaining shares x_0, the spread d_0, and the state s_t at $t = 0$.
3. For episode $= 1$ and for $t = 0$, we select a random action a_t with probability ϵ or when the q-value is all zero in that state or select $a_t = \underset{a}{\operatorname{argmax}} Q(S[t], a)$ where $a_t \in \{0, 1, \cdots, x_t\}$.

4. Update x_t and d_t by the function, $x_{t+1} = x_t - a_t$ and $d_{t+1} = e^{-\rho\Delta t} d_t + (e^{-\rho\Delta t} \times a_t)/S[t]$.
5. We choose the next state s_{t+1} with transition probability at time t.
6. Repeat step 3 to step 5 from $t = 1$ to $t = T$.
7. Update the value of Q-table by

$$Q[S[t], a_t] = Q[S[t], a_t] + \alpha[R_t + \sum_{i=1}^{T-t} (\gamma^i Q[S[t+i], a_{t+i}]) - Q[S[t], a_t]]$$

for $t < T$

$$Q[S[t], a_t] = Q[S[t], a_t] + \alpha[R_t - Q[S[t], a_t]]$$

for $t = T$.

8. Repeat step 2 to step 7 from episode $= 2$ to episode $= M$.

Numerical example

We set $n = 2$, $s = \{100, 200\}$, $P^i_{11} = P^i_{22} = 0.5$, $G_{s_j,i} = (1, 0)'$, $\mathbf{x}_i = (x_i, d_i)$, $\rho = 6$, $\Delta_t = t_{i+1} - t_i = 1/n$, $e^{-\rho(t_{i+1}-t_i)} \approx 0.05$

$$A_{s_j,i} = \begin{pmatrix} 1 & 0 \\ 0 & 0.05 \end{pmatrix}, \; B_{s_j,i} = \left(-1, \frac{0.05}{s_j}\right)', \; C_{s_j,i} = \begin{pmatrix} 1/2s_j & 0 & 1/2 \\ 0 & 0 & 0 \\ 1/2 & 0 & 0 \end{pmatrix}$$

and $F_{s_j,i} = \begin{cases} (0,0)', & i = 0, 1 \\ (1,0)', & i = 2 \end{cases}$

When the remaining shares is 2 and the spread is 0.1, then the policy of partition algorithm and Q-learning are as follows

Table: The policy of two methods for example 1

Market Depth	Partition Algorithm	Q Learning
(100,100,100)	(0,1,1)	(0,0,2)
(100,100,200)	(0,1,1)	(0,0,2)
(100,200,100)	(0,1,1)	(0,1,1)
(100,200,200)	(0,1,1)	(0,1,1)
(200,100,100)	(0,1,1)	(0,0,2)
(200,100,200)	(0,1,1)	(0,0,2)
(200,200,100)	(0,1,1)	(0,1,1)
(200,200,200)	(0,1,1)	(0,1,1)

- The red one is the policy which has minimum cost.

- Next we set the remaining shares x_0 to be $\{20, 40, 60, 80, 100\}$, that is, the more actions can be executed for each state.
- We use partition algorithm and reinforcement learning to find the optimal execution policy in each stage and compare the gap rate on average.
- The gap rate is count by

$$\frac{\text{method cost} - \text{naive cost}}{\text{naive cost}} \times 100\%$$

Table: The gap rate of two methods for different shares

	Partition Algorithm		Q Learning	
Shares	All	Remove	All	Remove
20	-31%	-32%	-31%	-32%
40	-13%	-15%	-13%	-14%
60	-8%	-10%	-8%	-10%
80	-6%	-8%	-5%	-7%
100	-5%	-7%	-5%	-6%

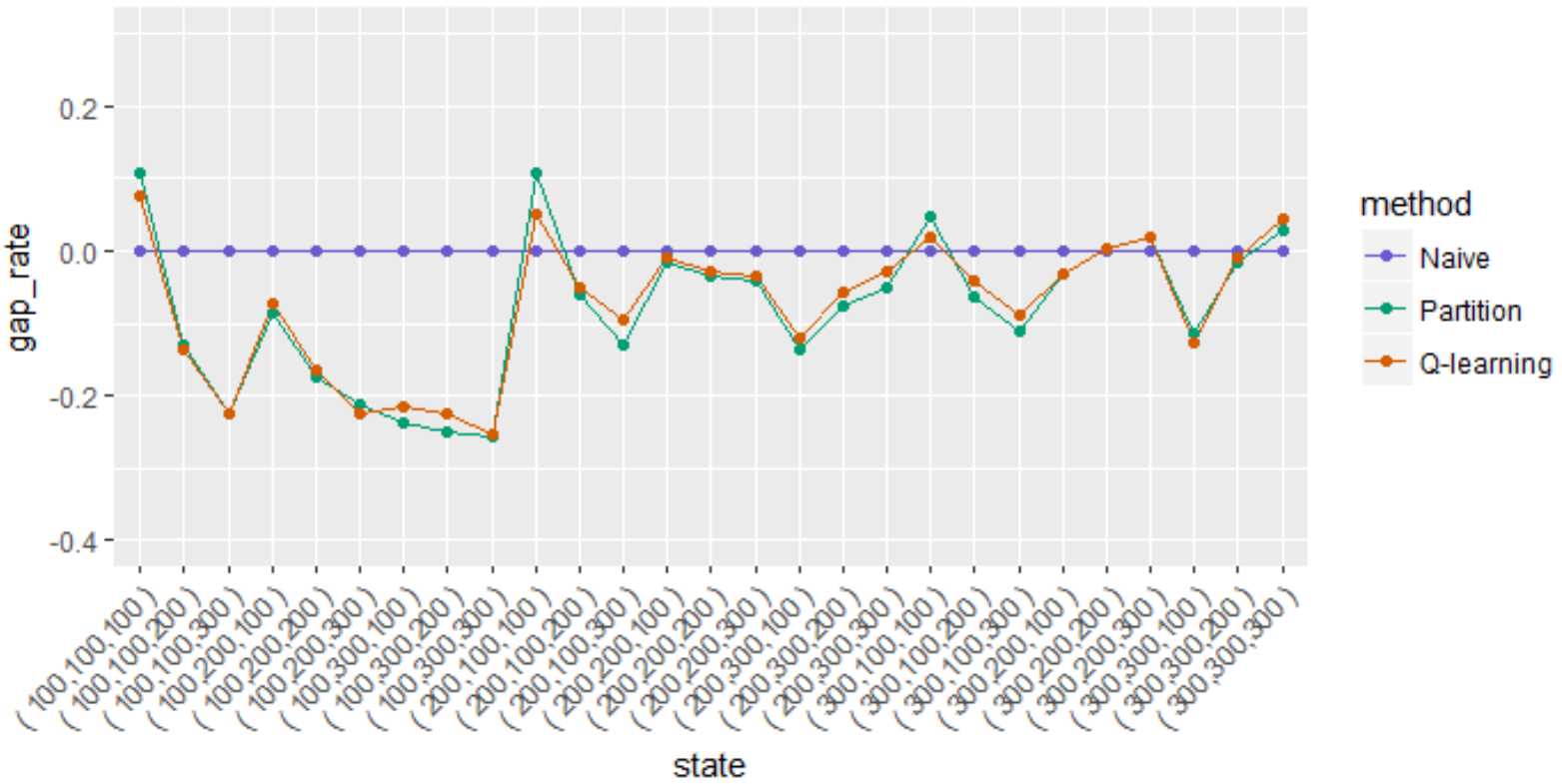
- The left column is all state and the the right column is the state except the unchanged state.

Table: The policy of two methods for example 2

Market Depth	Partition Algorithm	Q Learning
(100,100,100)	(19,34,47)	(23,32,45)
(100,100,200)	(19,34,47)	(23,32,45)
(100,200,100)	(19,48,33)	(23,41,36)
(100,200,200)	(19,48,33)	(23,41,36)
(200,100,100)	(32,28,40)	(31,32,37)
(200,100,200)	(32,28,40)	(31,32,37)
(200,200,100)	(32,40,28)	(31,41,28)
(200,200,200)	(32,40,28)	(31,41,28)

- Although there are different policies for different method, the gap rate on average are almost the same.

- Fix the coefficients of matrix A, B, C, F and G.
- We increase the state to be three situations, $s = \{100, 200, 300\}$, then we use the two methods to find the optimal execution policy in each stage and plot the gap rate of every situation.



- From the above figure, we then calculate the average gap rate of two methods and record it in following table

method	All	Remove
Partition Algorithm	-8%	-9%
Q Learning	-8%	-9%

- Comparing with the naive policy, we find that the average gap rate of two methods are the same.

Disney high-frequency data

Data introduction

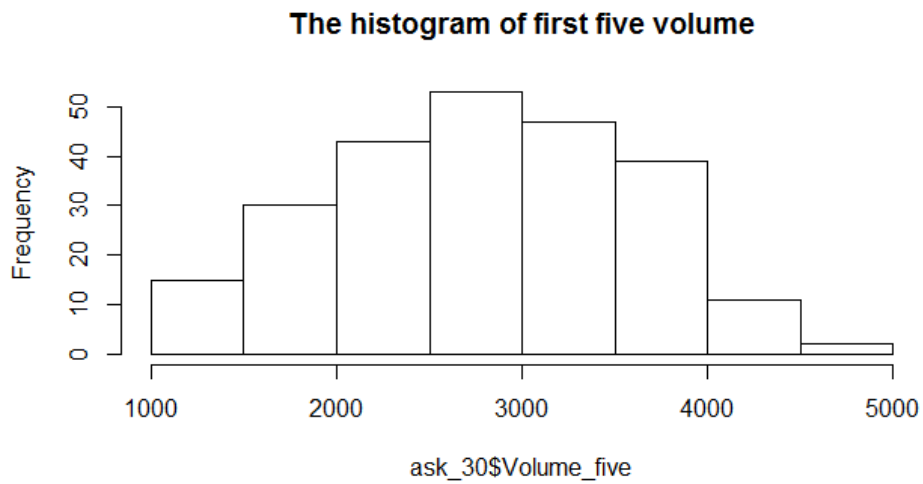
- The data is from LOBSTER which contains high quality limit order book data.
- We choose the limit oreder book data of Walt Disney Company at May 19,2017.
- The data contains about 210000 raw data and it contains the list of all transactions of one day.
- The dataset contains variables such as every ask price and bid price with corresponding volume, the trading time, event type, price, etc.

Disney high-frequency data

Data preprocessing

In order to define the different states, we first find the market depth and divide it into three states through the market depth.

- We calculate the mean volume of first five ask price for every 30 seconds, then we divide the data into three parts and calculate the average of its volume as the estimation of its state.



- Then the state is (1900,2800,3700).

- Since the spread can be calculated by

$$d_{i+1} = e^{-\rho\Delta t}(d_i + \frac{u_i}{s_i})$$

we take log at both side and the equation becomes

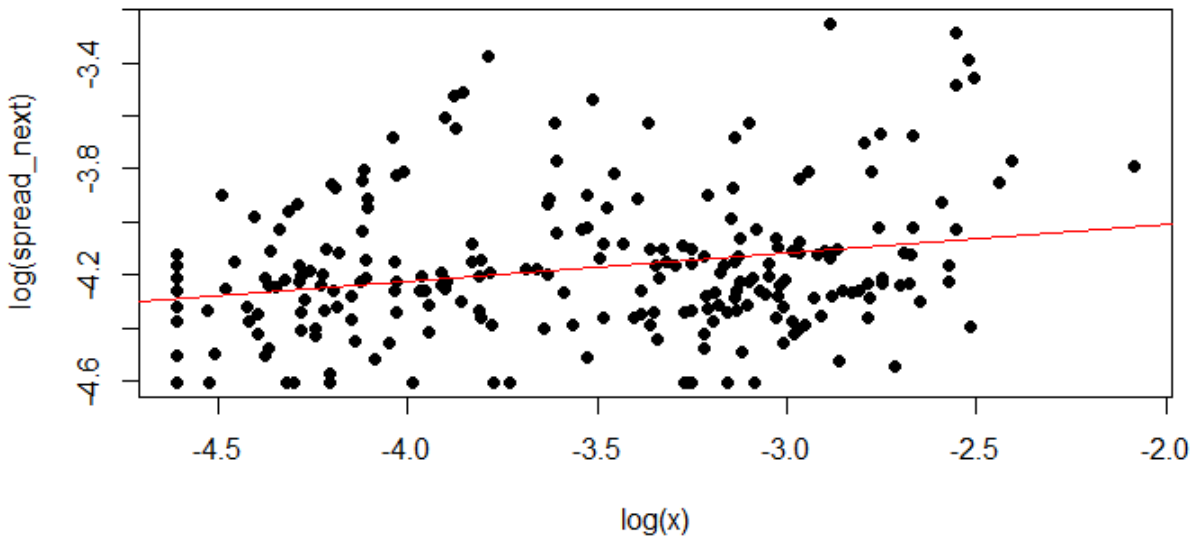
$$ln(d_{i+1}) = -\rho\Delta t + ln(d_i + \frac{u_i}{s_i})$$

- We assume $\Delta t = 1$(30 secs), then we fit the linear model to estimate the coefficient of ρ

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -3.79519    0.09903 -38.324  < 2e-16 ***
log(x)       0.10722    0.02742   3.911  0.00012 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2634 on 237 degrees of freedom
Multiple R-squared:  0.06062,   Adjusted R-squared:  0.05666
```

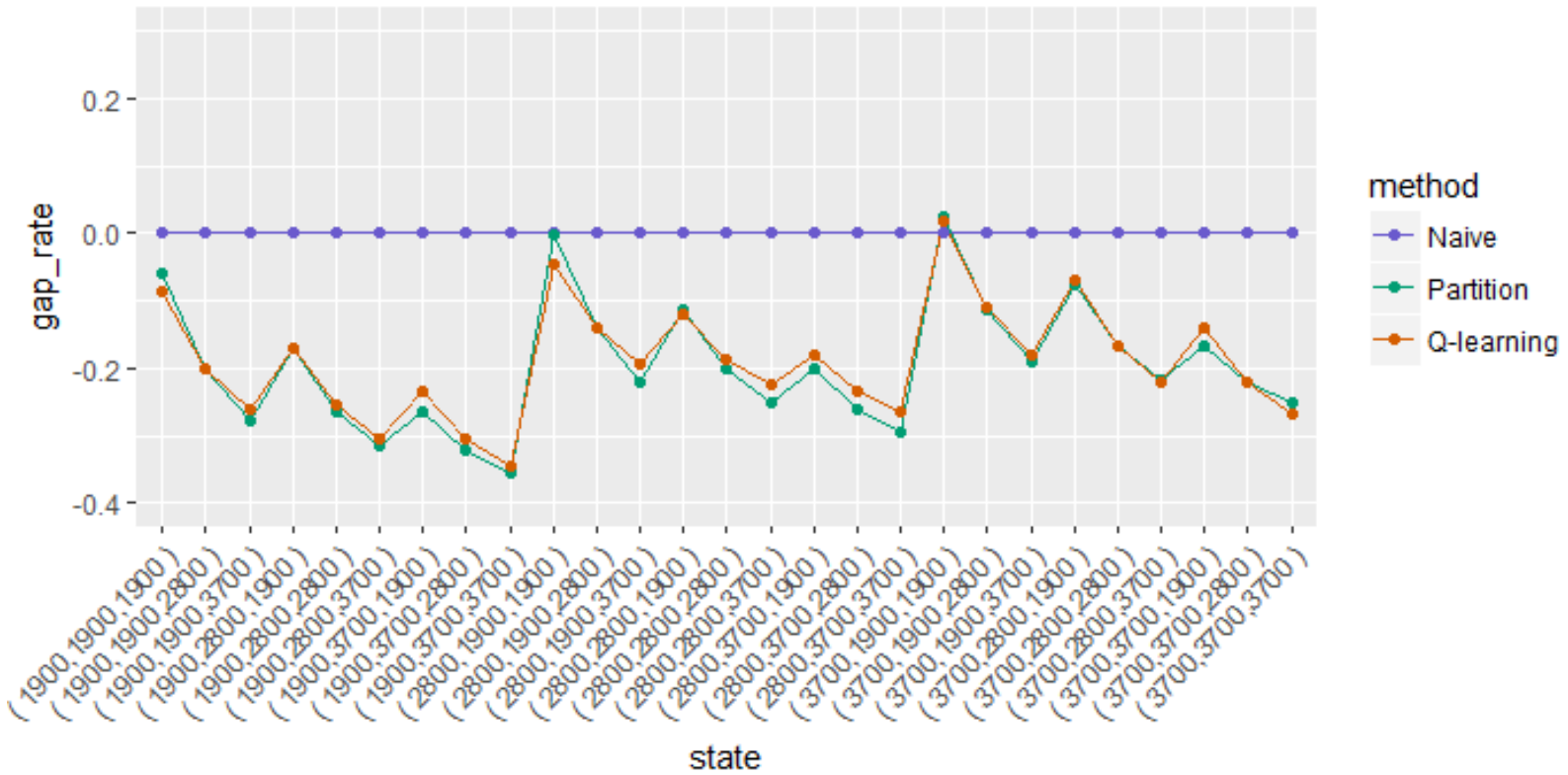
- Then we have the estimation of ρ is 3.8.
- Although the R-square of the model is only about 0.06, that is, there exists other varibales to affect the spread of next step, but the variable ρ is also an important variable for the next spread.



Disney high-frequency data

Comparison

- We assume that the transtition probability $P^i_{11} = P^i_{22} = P^i_{33} = 1/3$, $P^i_{12} = P^i_{13} = P^i_{21} = P^i_{23} = P^i_{31} = P^i_{32} = 1/3$.
- Here we want to buy 100 shares and the spread is 0.02, i.e. $(x_0, d_0) = (100, 0.02)$.



- From the output of real data, we can count the average gap rate of every state for each method and the average gap rate without invariable state.

Table: The gap rate of limit order book data

method	All	Remove
Partition Algorithm	-20%	-20%
Q Learning	-20%	-19%

- The gap rate of each state is somewhat different, the gap rate on average are almost the same, that is, the two method can decrese similar rate comparing with the naive method.

Disney high-frequency data

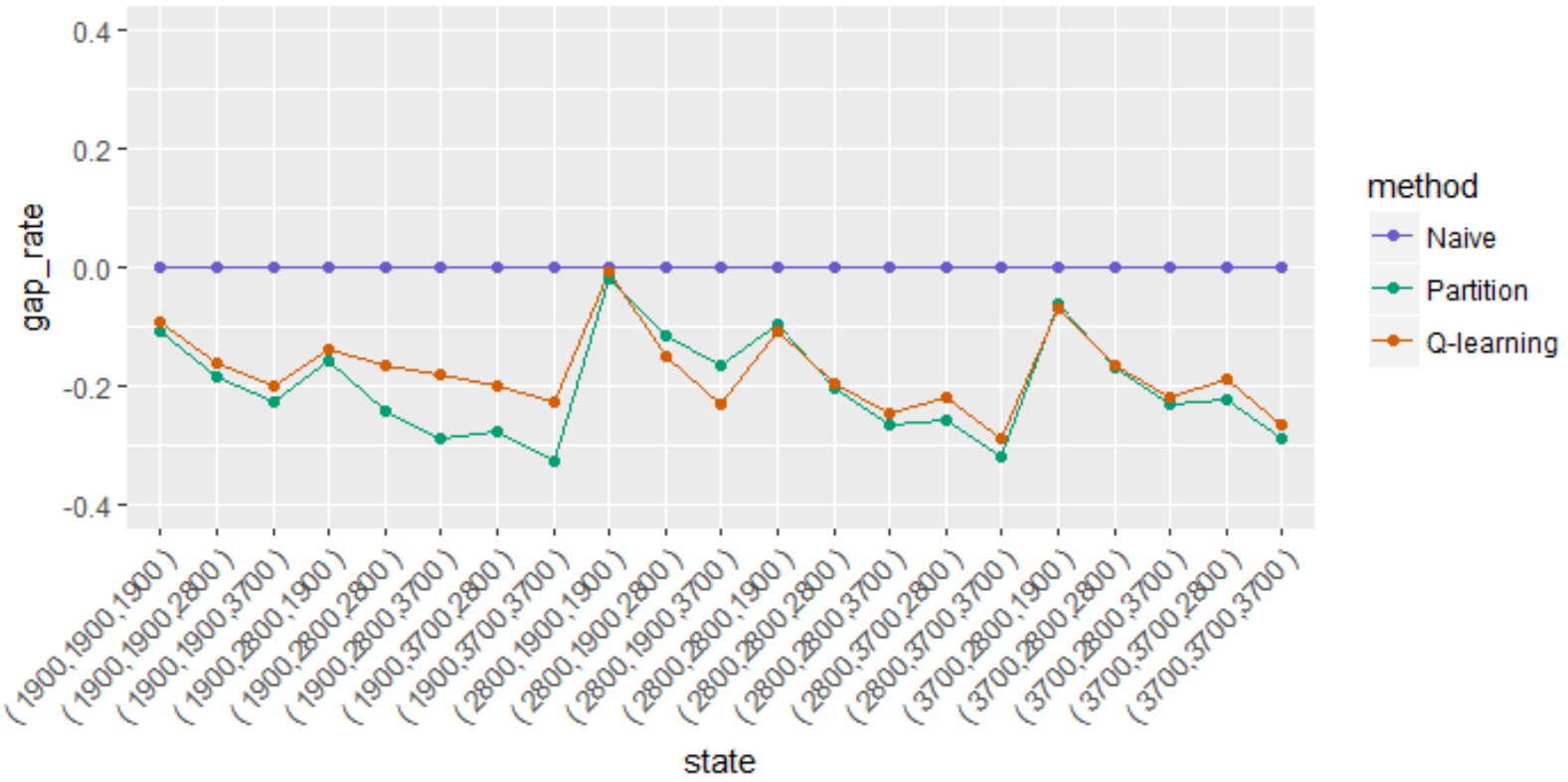
Comparison

- We use the historical data to estimate the transition probability

	s_1	s_2	s_3
s_1	0.85	0.138	0.013
s_2	0.139	0.608	0.253
s_3	0	0.262	0.738

- Here we want to buy 100 shares and the spread is 0.02, i.e. $(x_0, d_0) = (100, 0.02)$.

- The gap rate for each state is plotted below



Conclusion

- From the simulation study and empirical study, we find that when the market depth is higher, the policy tend to buy more shares, when the market depth is lower, the policy tends to buy less shares.
- Since the cost function we use is quadratic function, the unconstrainted optimal solution of partition algorithm can be easily sloved, otherwise, it may be more complex than the reinforcement learning.
- Reinforcement learning we use in the study spend more time than partition algorithm, because the algorithm needs to go through all situation. We can try deep Q-network to improve it.

References


Abergel, F., Anane, M., Chakraborti, A., Jedidi, A. and Muni Toke, I. (2016). *Limit Order Books*. Cambridge, England.


Chen, N., Kou, S. and Wang, C. (2017). A Partitioning Algorithm for Markov Decision Process with Applications to Market Microstructure. *Management Science*, **64**,784-803.


Grinstead, C. M. and Snell J. L. (1997). *Introduction to Probability*, 2nd Rev edition. American Mathematical Society, US.


Mnih, V., Kavukcuoglu, K. , Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S. and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, **518**, 529-533.

References


Sutton, R. S. and Barto, A. G. (2017). *Reinforcement Learning: An Introduction*. The MIT Press, England.


van Otterlo, M. (2009). *Markov Decision Processes: Concepts and Algorithms*. Course on 'Learning and Reasoning'.

Thank You for your Attention